Τ.Ε.Ι. ΗΠΕΙΡΟΥ
ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ & ΟΙΚΟΝΟΜΙΑΣ
ΤΜΗΜΑ ΤΗΛΕΠΛΗΡΟΦΟΡΙΚΗΣ
& ΔΙΟΙΚΗΣΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ & ΚΑΤΑΣΚΕΥΗ ΕΝΟΣ ETHERNET INTERFACE

ΔΕΛΛΙΟΣ ΚΛΕΑΝΘΗΣ – ΧΡΟΝΑΚΗΣ ΑΡΙΣΤΕΙΔΗΣ
ΑΜ:3399 ΑΜ:3651

ΑΡΤΑ, ΦΕΒΡΟΥΑΡΙΟΣ 2006

ΠΕΡΙΛΗΨΗ

Τα επικοινωνιακά συστήματα υπολογιστών και συγκεκριμένα το Internet παίζουν έναν πολύ σημαντικό, γρήγορα αυξανόμενο, ρόλο στο καθημερινό μας περιβάλλον. Φανταστείτε εφαρμογές, οι οποίες είναι ικανές να διαχειρίζονται εξοπλισμούς μέσω του Internet, να αποστέλλουν και να αντιλαμβάνονται τις καταστάσεις των ανιχνευτών ή αυτόματα να δημιουργούν και να στέλνουν E-mails στις περιπτώσεις σημαντικών γεγονότων-λειτουργιών (λόγου χάριν σε περιπτώσεις σκοπών ηλεκτρονικής ασφάλειας).

Γνωρίζοντας την χρήση των δικτύων, μικρό κομμάτι των τηλεπικοινωνιών, στους προσωπικούς υπολογιστές, το Ethernet μπορεί να προσφέρει επίσης μία κατανοητή και λογικής τιμής τεχνολογία σε εφαρμογές δικτύων, παραπάνω από ότι ένας υπολογιστής γραφείου.

Ο σκοπός αυτού του project είναι να δείξουμε πόσο εύκολο είναι να σχεδιάσουμε ένα σωρό του TCP/IP σαν ένα Ethernet Interface, πιο συγκεκριμένα την υλοποίηση ενός HTTP Server με χρήση του σωρού TCP/IP με τον μικροελεγκτή οικογένειας MSP430, και πιο συγκεκριμένα τον MSP430F149IPMG4, σε διεπαφή με την βοήθεια του Ethernet Controller CS8900A.

ΠΡΟΛΟΓΟΣ

Η παρούσα πτυχιακή εκπονήθηκε στη σχολή ΣΧΟΛΗ ΔΙΟΙΚΗΣΗΣ & ΟΙΚΟΝΟΜΙΑΣ στο τμήμα ΤΗΛΕΠΛΗΡΟΦΟΡΙΚΗΣ & ΔΙΟΙΚΗΣΗΣ του ΤΕΙ ΗΠΕΙΡΟΥ, στα πλαίσια του τομέα δικτύων και τηλεπικοινωνιών. Η εκκίνηση της πτυχιακής εργασίας τοποθετείται χρονικά τον Μάϊο του 2005 και η ολοκλήρωσή της τον Φεβρουάριο του 2006.

Θα θέλαμε να ευχαριστήσουμε θερμά τον καθηγητή και προϊστάμενο του τμήματος κύριο Κωνσταντίνο Αγγέλη, επιβλέποντα της παρούσας πτυχιακής εργασίας για την ευκαιρία που μας έδωσε να εργαστούμε σε ένα τόσο σύγχρονο και συνάμα ενδιαφέρον αντικείμενο, καθώς και για τη συνεχή βοήθεια και καθοδήγηση που μας παρείχε καθόλη την διάρκεια της εργασίας.

Κλείνοντας τον πρόλογο αυτό θα θέλαμε να ευχαριστήσουμε τις οικογένειές μας για την συνεχή υποστήριξή τους που υπήρξε καταλυτική καθόλη την διάρκεια των σπουδών μας.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΕΦΑΛΑΙΟ 4ο

ΚΕΦΑΛΑΙΟ 5ο

ΚΕΦΑΛΑΙΟ 6ο

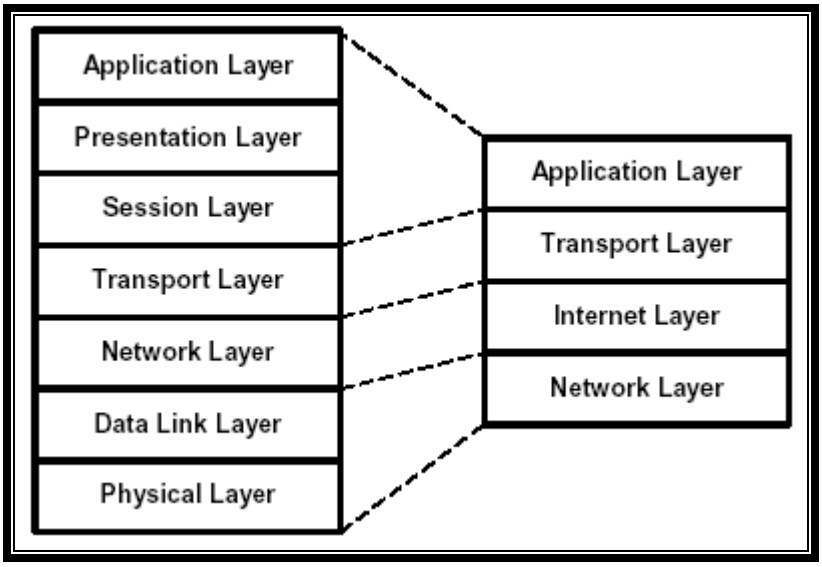
ΚΕΦΑΛΑΙΟ 1ο

1. ΒΑΣΙΚΑ ΠΡΩΤΟΚΟΛΛΑ ΕΠΙΚΟΙΝΩΝΙΑΣ

Ο σωρός των πρωτοκόλλων περιγράφεται από ένα μοντέλο, το οποίο βασίζεται σε ιεραρχημένα επίπεδα. Κάθε ένα από τα επίπεδα παρέχει, μία λειτουργία στο αμέσως επόμενο του καθώς επίσης παρέχει υπηρεσίες και στο αμέσως προηγούμενο επίπεδο.

Οι λειτουργικές λεπτομέρειες των κατωτέρων επιπέδων δεν είναι γνωστές στα ανώτερα επίπεδα. Αυτό βοηθάει στην καλύτερη ανάπτυξη του λογισμικού και τη σωστή διαχείρισή του. Για παράδειγμα, σε περίπτωση που τροποποιήσουμε τον κώδικα ενός επιπέδου, δεν χρειάζεται να τροποποιηθεί και ο κώδικας των υπόλοιπων επιπέδων.

Το μοντέλο αυτό είναι γνωστό με το όνομα ISO/OSI εφτά επιπέδων μοντέλο. Το παρακάτω σχήμα δείχνει την διαφορά ανάμεσα στα επίπεδα των μοντέλων ISO/OSI και Internet. Το Internet Layer και το Network Layer του Internet Model γίνεται implementation από την πλακέτα που σχεδιάστηκε με την βοήθεια των αρχείων κώδικα που θα αναφερθούν παρακάτω.



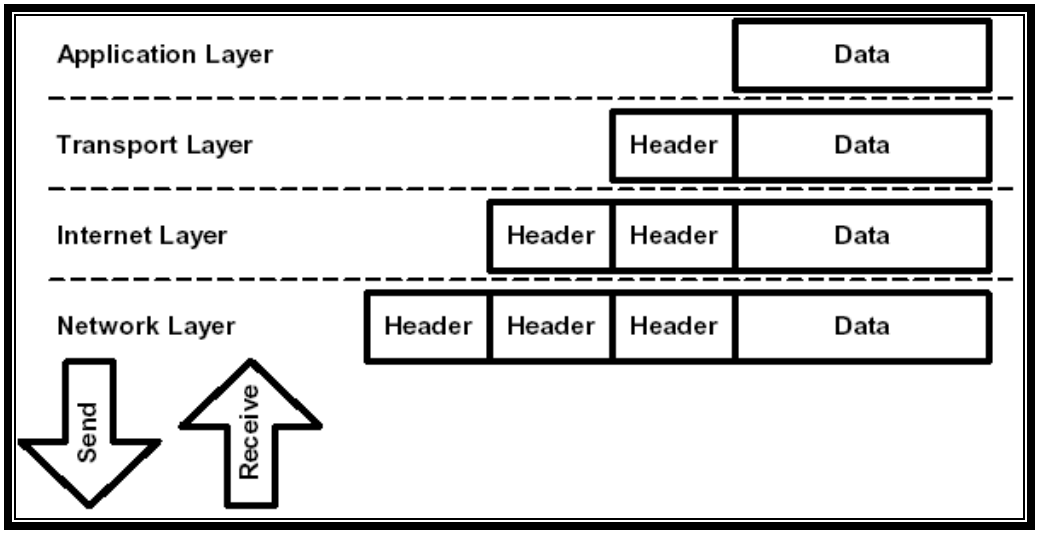
Σχήμα 1: ISO/OSI vs. Internet Model.

Όνομα Επιπέδου	***Λειτουργία***	***Χρήση***
Επίπεδο Εφαρμογής	Περιέχει πολλά πρωτόκολλα τα οποία καθορίζονται από εφαρμογές για το πώς θα παρέχουν υπηρεσίες	HTTP, Telnet, e-mail (SMTP, POP).
Επίπεδο μεταφοράς	Δημιουργούν την επικοινωνία εφικτή μεταξύ endpoints	TCP, UDP.
Επίπεδο Internet	Δρομολόγηση και παράδοση των Datagrams μεταξύ σημείων του Internet	IP, ICMP, ARP,
Επίπεδο Δικτύου	Host-specific υλοποίηση της εκπομπής datagrams.	Ethernet (IEEE 802.3) point-to-point protocol, AX.25 .

Πίνακας 1: Λειτουργίες των επιπέδων του Internet μοντέλου.

Κατά την αποστολή δεδομένων, κάθε επίπεδο προσθέτει την δική του επικεφαλίδα στο πακέτο δεδομένων. Αυτή η διαδικασία ονομάζεται ενσωμάτωση δεδομένων (data encapsulation) (βλέπε εικόνα 1).

Για παράδειγμα, κατά τη λήψη ενός πακέτου δεδομένων του TCP/IP σωρού από το Ethernet, το TCP/IP πρέπει να εξετάσει και να απομακρύνει κάθε επικεφαλίδα, από το κάθε ένα επίπεδο, για να μπορεί να επεξεργαστεί τα δεδομένα που θα λάβει [3].



Εικόνα 1: Διάγραμμα Επιπέδων

1.1 ETHERNET

Σήμερα το Ethernet είναι το πιο χρηστικό μέσο για να μεταφέρουμε δεδομένα σε ένα τοπικής περιοχής Δίκτυο (LAN). Τα πρότυπα που έχουν οριστεί από IEEE 802.3 καθορίζουν τους εφικτούς ρυθμούς αποστολής bit, τον τύπο του πακέτου που θα χρησιμοποιηθεί και το μέγεθος των bits που μπορούν να κωδικοποιηθούν.

Το Ethernet μοιράζει τους διαύλους, με αποτέλεσμα κάθε δίκτυο να έχει τα ίδια δικαιώματα πρόσβασης δεδομένων με την μέθοδο CSMA/CD (carrier sense multiple access with collision detection). Εάν εντοπιστεί κάποια σύγκρουση πακέτων, σταματάει η μετάδοση και χρησιμοποιείται ένας ειδικός αλγόριθμος υποστήριξης για επαναποστολή.

Κάθε δίκτυο έχει τη δική του ξεχωριστή φυσική διεύθυνση. Αυτή η διεύθυνση είναι η MAC διεύθυνση και αποτελείται από 48-bit. Το μέγιστο μήκος ενός Ethernet πακέτου είναι 1518 bytes, μη συμπεριλαμβανόμενου του προοιμίου.

Το προοίμιο αποτελείται από 1 και 0 που χρησιμοποιούνται για σκοπούς συγχρονισμού. Τα πρώτα 48 bits αναφέρονται στον προορισμό και τα επόμενα 48 bits στη διεύθυνση MAC.

Τα επόμενα 2 bytes περιγράφουν τον τύπο του πακέτου δεδομένων. Αυτά τα δύο bytes χρησιμοποιούνται για να αποφασιστεί σε πιο από τα επίπεδα προορίζεται το πακέτο. Στη συνέχεια, το μέγιστο μήκος των 1500 bytes ενός πακέτου, μπορούν να μεταδοθούν, ακολουθούμενα από μία cyclic redundancy check (CRC) τιμή των 4 bytes.

Χρησιμοποιώντας την CRC το Ethernet εξασφαλίζει ότι δεν θα υπάρξει αλλοίωση δεδομένων αλλά δεν εξασφαλίζει ότι τα δεδομένα θα φτάσουν με την σειρά που στάλθηκαν [3].

1.2 ΠΡΩΤΟΚΟΛΛΟ ARP (Address Resolution Protocol)

Η χρησιμοποίηση ενός δρομολογητή σε κάθε περίπτωση επικοινωνίας μεταξύ δύο σταθμών εργασίας, φορτώνει υπερβολικά κάποιο υποδίκτυο. Στην περίπτωση όπου και οι δύο ανήκουν στο ίδιο υποδίκτυο, αρκεί ο σταθμός πηγή να γνωρίζει τη φυσική διεύθυνση του σταθμού προορισμού, προκειμένου να του προωθήσει ένα IP datagram.

Το πρωτόκολλο το οποίο εκτελεί την παραπάνω λειτουργία ονομάζεται ARP. Το ARP είναι ένα χαμηλού επιπέδου πρωτόκολλο, το οποίο κρύβοντας τη μορφή των φυσικών διευθύνσεων, μας επιτρέπει να χρησιμοποιούμε τις IP διευθύνσεις με όποιον τρόπο θέλουμε [3].

1.3 ΠΡΩΤΟΚΟΛΛΟ ICMP (Internet Control Message Protocol)

Η λειτουργία του Internet παρακολουθείται στενά από τους δρομολογητές. Όταν συμβεί κάτι αναπάντεχο, το γεγονός αυτό αναφέρεται από το πρωτόκολλο μηνυμάτων ελέγχου του Internet, ICMP. Τα ICMP μηνύματα μεταφέρουν πληροφορία σχετικά με διάφορες δυσλειτουργίες, καθώς και λειτουργίες ελέγχου του δικτύου.

Τέτοιες λειτουργίες είναι:

- Αναφορά σφαλμάτων.
- Δοκιμή δυνατότητας πρόσβασης σε κόμβο.
- Έλεγχος συμφόρησης.
- Ειδοποίηση αλλαγής διαδρομής.
- Μέτρηση επιδόσεων.
- Διευθυνσιοδότηση υποδικτύων.

Κάθε τύπος μηνύματος ICMP ενσωματώνεται σε ένα πακέτο IP. Το λογισμικό υλοποίησης των TCP/IP πρωτοκόλλων επεξηγεί τα ICMP μηνύματα και εκτελεί τις απαραίτητες ενέργειες ανάλογα με το περιεχόμενό τους.

Στον παρακάτω πίνακα παρατείθονται οι κύριοι τύποι ICMP μηνυμάτων.

Τύπος μηνύματος	Περιγραφή
Destination unreachable	Το πακέτο δεν μπορούσε να παραδοθεί
Time exceeded	Το πεδίο διάρκειας ζωής έφθασε το 0
Parameter problem	Άκυρο πεδίο επικεφαλίδας
Source quench	Πακέτο φραγής
Redirect	Μάθε γεωγραφία σ' έναν δρομολογητή
Echo request	Ρώτα μία μηχανή αν είναι ζωντανή
Echo reply	Ναι, είμαι ζωντανή
Timestamp request	Όπως το Echo request αλλά με χρονική σφραγίδα
Timestamp reply	Όπως το Echo reply αλλά με χρονική σφραγίδα

Πίνακας 2: Κύριοι τύποι μηνυμάτων ICMP.

Από τα παραπάνω μηνύματα, τα μόνα τα οποία χρησιμοποιούνται στην υλοποίηση της πλακέτας, είναι τα Echo Request και Echo Reply. Αυτά τα δύο μηνύματα χρησιμοποιούνται κυρίως από την εντολή PING. Η εντολή PING αποστέλλει ένα Echo μήνυμα σε ένα σταθμό εργασίας, ο οποίος απαντά με ένα Echo Reply μήνυμα, στέλνοντας πίσω δεδομένα.[3].

1.4 ΠΡΩΤΟΚΟΛΛΟ TCP (Transmission Control Protocol)

Το πρωτόκολλο ελέγχου μεταφοράς TCP, σχεδιάστηκε ειδικά για να προσφέρει έναν αξιόπιστο συρμό bytes από άκρο σε άκρο μέσω ενός αναξιόπιστου δικτύου.

Ένα δίκτυο σαν το Internet διαφέρει από ένα απλό τοπικό δίκτυο επειδή τα διαφορετικά του μέρη μπορούν να έχουν:

- διαφορετικές τοπολογίες,
- διαφορετικό εύρος ζώνης,
- διαφορετικές καθυστερήσεις,
- διαφορετικά μεγέθη πακέτων,
- άλλες διαφορετικές παραμέτρους.

Το TCP σχεδιάστηκε έτσι ώστε να προσαρμόζεται δυναμικά στις ιδιότητες του διαδικτύου και να είναι ανθεκτικό ως προς πολλά είδη αστοχιών. Το TCP ορίστηκε επίσημα στο RFC 793 [3].

Για την παροχή αυτών των υπηρεσιών χρησιμοποιούνται οι ακόλουθοι μηχανισμοί:

1. ***Βασική Μεταφορά Δεδομένων:*** το TCP διαχωρίζει μία συνεχής ροή bytes σε πακέτα και τα στέλνει ως IP πακέτα.
2. ***Αξιοπιστία:*** σε περίπτωση απώλειας ή αντιγραφής των δεδομένων επανακτά την αρχική του λειτουργία, βάζοντας αριθμούς σε κάθε byte και flags.
3. ***Έλεγχος Ροής:*** κάθε φορά ένα TCP λαμβάνει ένα πακέτο από τον άλλον σταθμό διευκρινίζοντάς του πόσα bytes μπορεί να στείλει.
4. ***Πολυπλεξία:*** το TCP εισάγει αριθμούς θυρών (port numbers), το οποίο επιτρέπει την πολυπλεξία των IP διευθύνσεων. Ο συνδυασμός IP διευθύνσεων και αριθμών θυρών ονομάζεται Socket. Μία μοναδική TCP σύνδεση καθορίζεται από ένα ζευγάρι Sockets.

5. ***Συνδέσεις:*** πριν τη μετάδοση δεδομένων μία σύνδεση μεταξύ του αποστολέα και του παραλήπτη πρέπει να εγκατασταθεί. Κατά την διάρκεια της εγκατάστασης της σύνδεσης η συνέχεια των αριθμών συγχρονίζεται. Στη συνέχεια η μεταφορά των δεδομένων μπορεί να ξεκινήσει.

1.5 ΠΡΩΤΟΚΟΛΛΟ IP (Internet Protocol):

Το πρωτόκολλο IP έχει σχεδιαστεί για χρήση σε δίκτυα δρομολόγησης πακέτων στο Internet. Χρησιμοποιεί μηχανισμούς για τη μετάδοση IP-datagrams, από μία πηγή προς έναν προορισμό. Επίσης έχει έναν μηχανισμό κατακερμάτισης πακέτων για την αποστολή δεδομένων διαμέσου δικτύων με μικρή χωρητικότητα πακέτων δεδομένων.

Η πλακέτα αυτή χρησιμοποιεί το IP ver.4.0. Το πρωτόκολλο IP προσφέρει ένα σύνολο από βασικές λειτουργίες οι οποίες είναι απαραίτητες για τη λειτουργία της διασύνδεσης.

Τέτοιες λειτουργίες είναι:

- Λειτουργίες κατακερματισμού των μηνυμάτων και επανασύνδεσης αυτών.
- Λειτουργίες δρομολόγησης των IP-datagrams διαμέσου των κόμβων του δικτύου προκειμένου να φτάσουν στον προορισμό τους.
- Λειτουργίες αναφοράς σφαλμάτων, που έχουν ως σκοπό την ενημέρωση του κόμβου-πηγή για σχετικές απώλειες.

Το IP πρωτόκολλο ορίζει τη μορφή που πρέπει να πάρουν τα πακέτα και την επεξεργασία που πρέπει να δεχτούν προκειμένου να μεταδοθούν μέσα από το μέσο μεταφοράς.

1.6 ΠΡΩΤΟΚΟΛΛΟ HTTP (HyperText Transfer Protocol)

Το πρωτόκολλο HTTP είναι ένα πρωτόκολλο επιπέδου εφαρμογής, το οποίο μπορεί να χρησιμοποιηθεί για πολλές λειτουργίες. Χρησιμοποιεί το σύστημα client-server και βασίζεται στο TCP.

Η πιο σημαντική του χρησιμοποίηση βρίσκεται στο να μεταδίδει HTML σελίδες μεταξύ Internet Servers και πελατών. Κατά διάρκεια λειτουργίας του πρωτοκόλλου HTTP, ένας πελάτης δημιουργεί μία σύνδεση σε έναν server και κάνει αίτηση για μία ιστοσελίδα.

Αφού ο server αποστείλει την ιστοσελίδα στον πελάτη, η σύνδεση μπορεί να τερματιστεί [3].

►ΠΙΝΑΚΑΣ 3: ΣΗΜΑΝΤΙΚΟΙ ΜΕΘΟΔΟΙ HTTP:

Μέθοδος	***Περιγραφή***
GET	Ένας πελάτης ζητά μία ιστοσελίδα από έναν server και ο server αποστέλλει το πλαίσιο επικεφαλίδας και την ιστοσελίδα.
HEAD	Ομοίως με την εντολή GET αλλά αποστέλλει μόνο το πλαίσιο επικεφαλίδας.
POST	Χρησιμοποιείται για την μεταφορά πληροφοριών μεταξύ server και client.

ΚΕΦΑΛΑΙΟ 2^{Ο}

2. ΠΕΡΙΓΡΑΦΗ ΤΟΥ HARDWARE

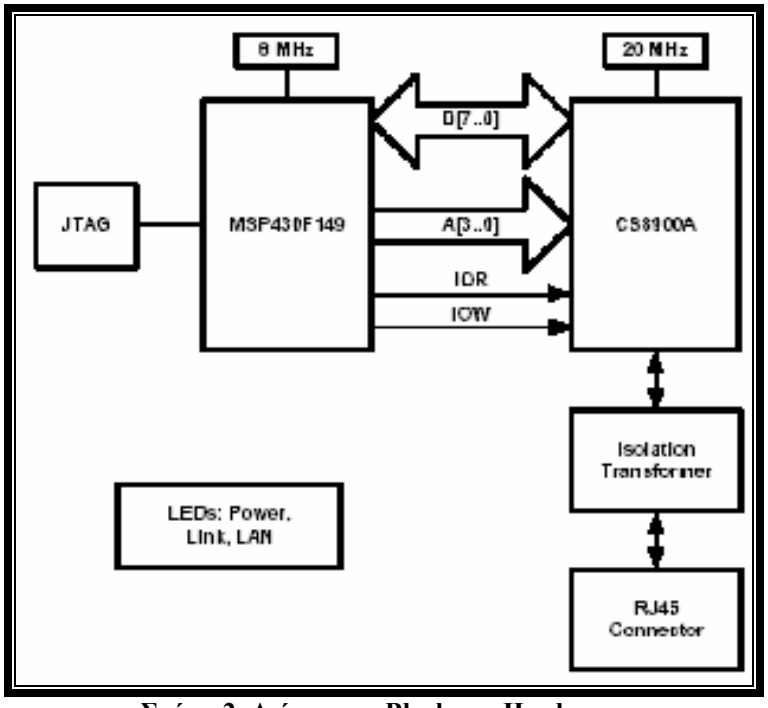
Τα δύο κύρια συστατικά της πλακέτας είναι ο MSP430F149IPMG4 επεξεργαστής της Texas Instrument και ο CS8900A Ethernet Controller της Crystal Semiconductor Corporation.

Ο MSP430F149 που χρησιμοποιείται έχει 60KBytes Flash memory και 2KBytes RAM. Έχει επίσης, 6 I/O θύρες, οι οποίες χρησιμοποιούνται όχι μόνο για το interface με το LAN Controller, αλλά και για άλλες γενικότερες εφαρμογές. Γενικά, τα I/O port pins του MSP430F149 χρησιμοποιούνται για να παρέχουν ένα Bus Interface στον LAN Controller. Το πιο ενδιαφέρον σημείο στο project αυτό, είναι η διασύνδεση μεταξύ των LAN Controller (IC2, βλέπε εικόνα 7) και MSP430F149 (IC1, βλέπε εικόνα 7).

Ο CS8900A είναι ένας LAN Controller μικρού κόστους και αυτό τον κάνει κατάλληλο για αυτήν την εφαρμογή. Το Bus Interface του CS8900A είναι εύκολο να έχει άμεση διεπαφή με το μικροεπεξεργαστή. Ο CS8900A μπορεί να λειτουργήσει σε τρεις διαφορετικές καταστάσεις: I/O space, memory space, και σαν DMA slave.

Το πιο σημαντικό είναι ότι μπορεί να χρησιμοποιήσει έναν 8-bit data bus [2]. Αυτός ο data bus είναι συνδεδεμένος στο general I/O port 5 του MSP430F149. Ο CS8900A σε I/O mode προσπελαύνεται μέσω 8 16-bit I/O port που αντιστοιχούν σε 16 registers και να προσπελαστούν χρησιμοποιείται ένας 4-bit address bus width. Επίσης χρησιμοποιούνται δύο γραμμές ελέγχου, η IOR και η IOW. Αυτά τα σήματα είναι active-low και δείχνουν πότε έχουμε εγγραφή ή ανάγνωση σε εξέλιξη.

Η όλη διεπαφή ολοκληρώνεται χρησιμοποιώντας μόνο δεκατέσσερα ηλεκτρικά σήματα. Όλα τα μη χρησιμοποιούμενα pins του CS8900A οδηγούνται στα κατάλληλα επίπεδα για να επιλεχθεί το κατάλληλο mode και για να γίνει configuration του bus interface. (σχήμα 1).



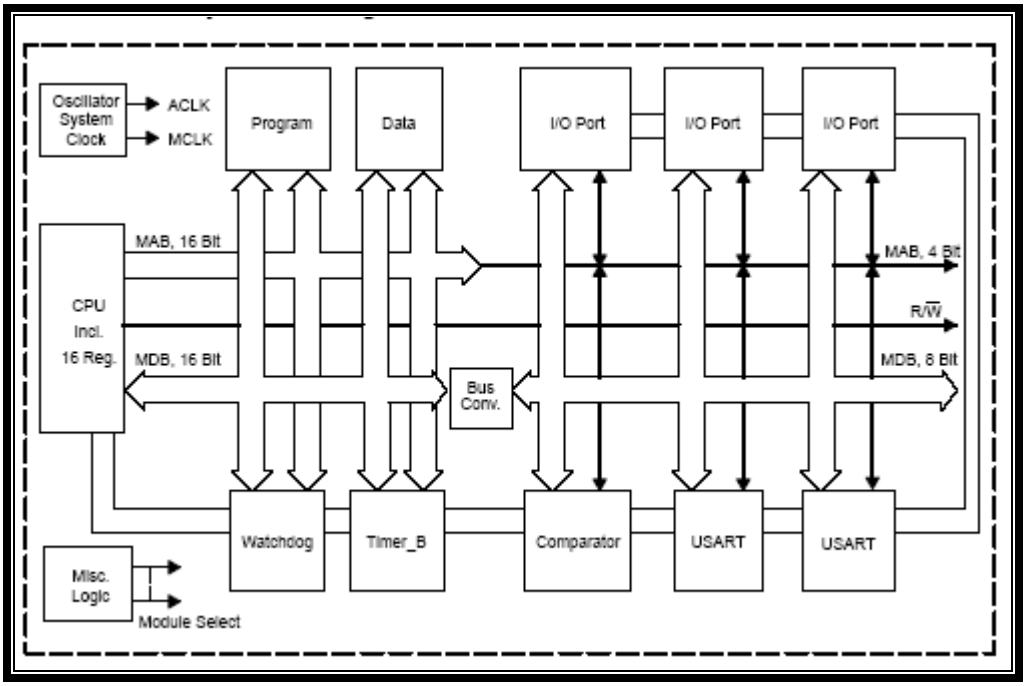
Σχήμα 2: Διάγραμμα Block του Hardware.

► ΠΙΝΑΚΑΣ 4: ΛΙΣΤΑ ΜΕΡΩΝ

ΟΝΟΜΑ ΜΕΡΟΥΣ	ΤΙΜΗ/ΠΕΡΙΓΡΑΦΗ	ΤΥΠΟΣ ΠΑΚΕΤΟΥ
C1	560 pF	SMT 0805
C2, C3, C6, C13, C14, C15, C16, C17, C18, C19, C20, C21, C22	0.1 µF (100n)	SMT 0805
C4, C5	15 pF	SMT 0805
C24, C25	4700 pF / 2 kV (4n7)	
D1	LED red, 3 mm, 2 mA (rt)	
D2	LED yellow, 3 mm, 2 mA (ge)	
D4	LED green, 3 mm, 2 mA (gn)	
IC1	MSP430F149	QFP-64
IC2	ISA ethernet controller CS8900A-IQ3 (Crystal Semiconductor)	TQFP-100
IND1	Transformer E2023 (Pulse Engineering)	SO-16L
Q1	20 MHz	HC-49
Q2	8 MHz	HC-49
R1	100 Ω	SMT 0805
R2, R3	8.2 Ω	SMT 0805
R4	4.7 kΩ (4K7)	SMT 0805
R5	4.99 kΩ, 1% (4K99)	SMT 0805
R6, R7, R10	560 Ω	SMT 0805
R8, R9	100 kΩ	SMT 0805
X2	RJ45 LAN connector	
X3, X4	Header, 26-pin	ML26
X6	Header, 14-pin	ML14

2.1 ΑΡΧΙΤΕΚΤΟΝΙΚΗ & ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΗ MSP430F149

Η αρχιτεκτονική της οικογένειας μικροεπεξεργαστών τύπου MSP430 βασίζεται στην αρχιτεκτονική τύπου memory-to-memory, ένα κοινό κενό διεύθυνσης για όλα τα σχετιζόμενα μπλοκ, και ένα σετ μειωμένων εντολών, εφαρμόσιμο σε όλα τα μπλοκ όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 2: Αρχιτεκτονική MSP430.

2.1.1 ΜΝΗΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ (Program Memory)

Ο μικροεπεξεργαστής ενσωματώνει ένα μειωμένο και εύκολα διαχειρίσιμο σετ εντολών, καθώς επίσης έχει ορθογωνικό σχεδιασμό. Αποτελείται από μία 16-bit αριθμητική λογική μονάδα (ALU) και 16 καταχωρητές και καθοδηγούμενη λογική ελέγχου.

Τέσσερις από αυτούς τους καταχωρητές χρησιμοποιούνται για ειδικούς σκοπούς. Αυτοί είναι ο μετρητής προγράμματος (Program Counter), ο δείκτης σωρού (stack Pointer), ο καταχωρητής κατάστασης (Status register), και η γεννήτρια μεταβλητών (constant generator).

Όλοι οι καταχωρητές εκτός από τους τελευταίους costant-generators, με το όνομα R3/CG2 και μέρος του R2CG1, μπορούν να προσπελαθούν με χρήση ολόκληρου του σετ εντολών. Ο constant generator εφοδιάζει με μεταβλητές τον μικροελεγκτή και δεν χρησιμοποιείται για αποθήκευση δεδομένων. Ο τρόπος διευθυνσιοδότησης που χρησιμοποιείται στο CG1, ξεχωρίζει τα δεδομένα από τις μεταβλητές.

Ο έλεγχος της CPU πάνω από τον μετρητή προγράμματος, καταχωρητή κατάστασης και δείκτη σωρού, με τη χρήση πάντοτε του μειωμένου σετ εντολών, επιτρέπει την ανάπτυξη εφαρμογών με πολύπλοκες διευθυνσιοδοτήσεις και αλγόριθμους λογισμικών.

Κάθε πρόσβαση χρησιμοποιεί τον 16-bit memory data bus (MDB) και όσες από τις γραμμές διεύθυνσης του memory address bus (MAB), χρειάζονται για την πρόσβαση σε τοποθεσίες της μνήμης.

Τα μπλοκ μνήμης επιλέγονται αυτόματα διαμέσου module-enable σημάτων. Αυτή η τεχνική μειώνει το ρυθμό της υπάρχουσας κατανάλωσης. Η μνήμη του προγράμματος είναι υλοποιημένη σαν programmable ή mask-programmed μνήμη.

Επί προσθέτως στον κώδικα προγράμματος, τα δεδομένα είναι δυνατόν να τοποθετούνται στον τομέα ROM του χάρτη της μνήμης και να είναι προσβάσιμα με τη χρήση λέξεων ή byte εντολών, χρήσιμο για πίνακες δεδομένων.

Αυτό το μοναδικό χαρακτηριστικό δίνει ένα πλεονέκτημα στον MSP430 σε σχέση με τους άλλους μικροελεγκτές, διότι οι πίνακες δεδομένων δεν χρειάζεται να αντιγραφούν στη μνήμη για χρήση.

Τέλος μνήμη χώρου δεκαέξι λέξεων κρατείται για επανεκκίνηση του MSP430 & interrupted vectors, στον αρχικό χώρο των 64KBytes διεύθυνσης, από την 0FFFh έως την 0FFE0h.

2.1.2 ΜΝΗΜΗ ΔΕΔΟΜΕΝΩΝ (Data Memory)

Η μνήμη δεδομένων είναι συνδεδεμένη στην CPU διαμέσου των ίδιων διαύλων όπως και η μνήμη του προγράμματος, ROM, δηλαδή με την MAB και την MDB. Αυτή μπορεί να προσπελαστεί είτε με πλήρη δεδομένα, είτε με μειωμένο εύρος δεδομένων.

Επιπροσθέτως, επειδή η RAM και η ROM είναι συνδεδεμένες στην CPU μέσω των ίδιων διαύλων, ο κώδικας του προγράμματος μπορεί να φορτωθεί και να εκτελεστεί επίσης από την RAM. Με τον τρόπο αυτό ο MSP430 παρέχει μεγάλη ευκολία στον προγραμματισμό (eay-to-use debugging capability).

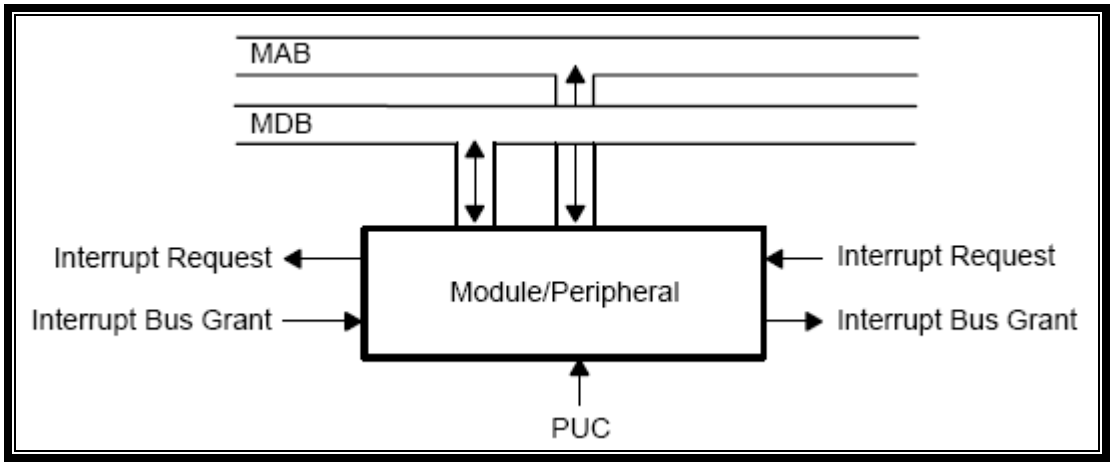
2.1.3 ΕΛΕΓΧΟΣ ΛΕΙΤΟΥΡΓΙΩΝ

Η λειτουργία των διαφορετικών τύπων της οικογένειας των μικροεπεξεργαστών MSP430 ελέγχονται κυρίως από τις πληροφορίες που αποθηκεύονται στους ειδικών-συναρτήσεων καταχωρητές. (SFRs). Τα διαφορετικά bits στους SFRs ενεργοποιούν διακοπές, που παρέχουν πληροφορίες σχετικά με την κατάσταση των διακοπών των flags και ορίζουν τους λειτουργικούς τρόπους των περιφερειακών στοιχείων. Με την απενεργοποίηση των περιφερειακών που δεν χρησιμοποιούνται και χρειάζονται κατά την διάρκεια μίας λειτουργίας, η συνολική κατανάλωση εκείνης της στιγμής μπορεί να μειωθεί.

2.1.4 ΠΕΡΙΦΕΡΕΙΑΚΑ

Περιφερειακά modules, τα οποία είναι συνδεδεμένα στη CPU διαμέσου της MAB, MDB και υπηρεσιών διακοπών και αιτούμενων γραμμών. Ο MAB συνήθως είναι ένας 5-bit δίαυλος για τα περισσότερα από τα περιφερειακά. Ο MDB είναι ένας 8-bit ή 16-bit δίαυλος.

Τα περισσότερα περιφερειακά μπορούν να λειτουργήσουν σε byte format. Modules με δίαυλο 8-Bit είναι συνδεδεμένα με διαύλους σε κυκλώματα των 16-bit της CPU.



Εικόνα 3: Λειτουργία για 8-bit περιφερειακά/ Σύνδεση Διαύλων

2.1.5 ΤΑΛΑΝΤΩΤΕΣ (Oscillators)

Ο LTX1 ταλαντωτής είναι σχεδιασμένος για την πιο συχνή χαμηλή κατανάλωση του clock crystal, των 32,768 Hz, ή με την χρήση ενός υψηλής ταχύτητας κρυστάλλου.

Όλα τα αναλογικά εξαρτήματα για τον 32,768 Hz ταλαντωτή είναι τοποθετημένα στον MSP430 και δεν χρειάζονται άλλα επιπρόσθετα εξωτερικά εξαρτήματα να χρησιμοποιηθούν. Όταν χρησιμοποιείται ο LTX1 ταλαντωτής με ένα υψηλής ταχύτητας κρύσταλλο, χρειάζονται επιπρόσθετοι πυκνωτές (βλέπε εικόνα 7).

Επιπλέον οι MSP430 μικροεπεξεργαστές χρησιμοποιούν και περιέχουν έναν ψηφιακά ελεγχόμενο RC ταλαντωτή (DCO). Αυτός είναι διαφορετικός από τους υπόλοιπους RC ταλαντωτές, διότι είναι ψηφιακά ελεγχόμενος .

2.1.6 ΨΗΦΙΑΚΗ I/O ΡΥΘΜΙΣΗ

Ο γενικός σκοπός των I/O θυρών του MSP430 είναι ότι σχεδιάστηκαν για να αποδώσουν την μέγιστη ελαστικότητα. Κάθε I/O γραμμή μπορεί να ρυθμιστεί ανεξάρτητα, και όλες διαθέτουν την δυνατότητα διακοπής.

Στους μικροεπεξεργαστές τύπου MSP430 υπάρχουν δύο είδη θυρών η P1 θύρα και η P2, οι οποίες έχουν δυνατότητα διακοπής και edge sensitivity για κάθε bit ξεχωριστά.

Επίσης είναι η P3 και η P6 που είναι διαφορετικού τύπου, αλλά όλες μαζί έχουν δυνατότητες ελέγχου εισερχόμενη/εξερχόμενης διαδρομής και εξερχόμενο επίπεδο, για να διαβάζει το επίπεδο που υπάρχει στο κάθε pin, και για τον έλεγχο εάν κάποια θύρα ή κάποια συνάρτηση είναι ενεργοποιημένη σε κάποιο pin.

2.2 ΑΡΧΙΤΕΚΤΟΝΙΚΗ & ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ETHERNET CONTROLLER CS8900A

Ο CS8900A είναι ένας χαμηλού κόστους Ethernet LAN Controller κατάλληλος για την ISA (Industry Standard Architecture) bus και γενικά για busses μικροελεγκτών. Ο υψηλά ολοκληρωμένος σχεδιασμός της εξαλείφει την ανάγκη για ακριβά εξωτερικά επιπρόσθετα μέρη που είναι απαιτούμενα από άλλους Ethernet Controllers. Ο CS8900A περιέχει μία on-chip RAM , 10Base-T αποστολής και λήψης φίλτρα, και μία απευθείας ISA-Bus διεπαφή με 24 mA Drivers.

Επιπροσθέτως, ο CS8900A προσφέρει μία μεγάλη γκάμα από αποδοτικά στοιχεία και στοιχεία ρυθμίσεων. Η μοναδικού τύπου αρχιτεκτονική του προσαρμόζεται αυτόματα σε αλλαγές κίνησης του δικτύου και διαθέσιμους πόρους του συστήματος.. το αποτέλεσμα αυτών είναι μία αυξημένη απόδοση του συστήματος.

Ο CS8900A είναι διαθέσιμος σε πακέτο τύπου LQFP 100 pin, όταν το κόστος για μία εφαρμογή του Ethernet παίζει πρωταρχικό ρόλο. Με τον CS8900A, είναι δυνατόν να δημιουργήσουμε ένα ολοκληρωμένο κύκλωμα Ethernet το οποίο καταλαμβάνει λιγότερο από 1,5 τετραγωνικές ίντσες (10sq. cm) χώρου στην πλακέτα.

2.2.1 ISA-Bus Interface

Τα στοιχεία ρύθμισης του CS8900A περιέχουν μία επιλογή τεσσάρων interrupts και τριών DMA καναλιών, όπου το καθένα επιλέγετε κατά την διάρκεια της αρχικοποίησης (βλέπε αρχεία κώδικα cs8900a.c & cs8900a.h).

Στο Memory mode αυτού, υποστηρίζονται Standard or Ready Buses κύκλοι, χωρίς να εισάγονται επιπρόσθετες καταστάσεις αναμονής. Ο δίαυλος μπορεί να ρυθμιστεί για την υποστήριξη πολλών διαύλων μικροελεγκτών και επεξεργαστών.

2.2.2 Integrated Memory

Ο CS8900A ενσωματώνει μία 4-KByte σελίδα σε on-chip memory, εξαλείφοντας το κόστος και το μέγεθος της πλακέτας που σχετίζεται εξωτερικά chips μνήμης χαρακτηριστικό που κάνει την σχεδίαση της πλακέτας πιο εύκολη.

Σε αντίθεση με τους άλλους Ethernet Controllers, ο CS8900A κρατάει αποθηκευμένα ολόκληρα frames αποστολής και λήψης στο chip, εξαλείφοντας την ανάγκη για πολύπλοκα ασαφή σχέδια διαχείρισης μνήμης.

Επιπροσθέτως, ο CS8900A μπορεί να λειτουργήσει σε Memory space, σε I/O space, ή σαν εξωτερικός DMA controller, παρέχοντας τη μέγιστη σχεδιαστή ευελιξία.

2.2.3 803.3 Ethernet Mac Engine

Η μηχανή του CS8900A Ethernet Mac (Media Access Control) είναι απολύτως συμβατή με το IEEE 802.3 Ethernet standard (ISO/IEC 8802-3, 1993), και υποστηρίζει αμφίδρομη λειτουργία. Διαχειρίζεται όλα τα Ethernet πακέτα λήψης και αποστολής, εμπεριέχοντας επίσης: αναγνώριση συγκρούσεων, προοίμιο δημιουργίας και αναγνώρισης και δημιουργία CRC και διάγνωση-τεστ.

Τα προγραμματιζόμενα MAC χαρακτηριστικά περιέχουν αυτόματη επαναποστολή σε περίπτωση σύγκρουσης, και αυτόματο συμπλήρωμα των πακέτων που αποστέλλονται.

ΠΙΝΑΚΑΣ 5: ΠΕΡΙΓΡΑΦΗ ΤΩΝ PINS ΤΟΥ CS8900A ΠΟΥ ΚΑΝΟΥΜΕ ΧΡΗΣΗ

ΟΝΟΜΑ PIN	ΠΕΡΙΓΡΑΦΗ
SA[0:19]	20 από 24 bit του System Address Bus που χρησιμοποιούνται για την πρόσβαση της CS8900A μνήμης. Το SA.0 και SA.15 χρησιμοποιούνται για I/O λειτουργίες.

SD[0:15]	Αμφίδρομα bit που χρησιμοποιούνται για την μεταφορά δεδομένω μεταξύ του CS8900A και MCU.
RESET	Pin που χρησιμοποιείται για τη επανεκκίνηση του CS8900A.
AEN-Address Enable	Όταν το pin AEN είναι σε κατάσταση high, CS8900A δεν θα λειτουργεί σαν I/O slave, ενώ αν είναι σε κατάσταση low όταν έχουμε πρόσβαση στην μνήμη του.
IOR-I/O Read	Όταν βρίσκεται σε κατάσταση low, μπορούμε να διαβάσουμε από τους registers του CS8900A.
IOW-I/O Write	Όταν είναι σε κατάσταση low, μπορούμε να γράψουμε στους registers του CS8900A. Όταν όμως το REFRESH είναι low τότε αγνοείται.
TXD+/TXD-/10BASE-T Transmit	Διφορούμενα pins εξόδου 10MBps Manchester-encoded δεδομένα.
RXD+/RXD-/10BASE-T Receive	Διφορούμενα pins εισόδου 10MBps Manchester-encoded δεδομένα.
XTAL[1:2]	Σε αυτά τα pins συνδέουμε έναν 20MHz κρύσταλλο (crystal).
SLEEP	Pin που σε κατάσταση low χρησιμοποιείται για την εκκίνηση των Sleep-Modes (Hardware-Suspend και Hardware-Standby).
LINKLED	Όταν δεν λειτουργεί το Led είναι σε κατάσταση Low.
LANLED	Όταν λειτουργεί το Led είναι σε κατάσταση High.
TEST	Όταν είναι low ο CS8900A μπαίνει σε Test mode.

2.3 ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΠΛΕΟΝΕΚΤΗΜΑΤΑ ΤΟΥ CS8900A

Ο CS8900A σχεδιάστηκε για χρήσεις όπως ολοκληρωμένες εφαρμογές, φορητές μητρικές πλακέτες, non-ISA bus συστήματα και κάρτες προσαρμογής.

Μερικά βασικά πλεονεκτήματα που μειώνουν το κόστος σε περιπτώσεις κατασκευής πλακετών με το συγκεκριμένο Ethernet Controller είναι:

Η ενσωματωμένη RAM εξαλείφει την ανάγκη για δαπανηρά εξωτερικά chips μνήμης.

- Τα on-chip 10Base-T φίλτρα επιτρέπουν στους σχεδιαστές να χρησιμοποιούν απλούς Isolation Transformers, αντί δαπανηρών πακέτων φίλτρων/transformers. Ο 10 Base-T transceiver που περιλαμβάνει είναι απόλυτα συμβατός με το στάνταρ (ISO/IEC 8802-3 1993) και υλοποιεί όλα τα αναλογικά και ψηφιακά κυκλώματα που χρειάζονται για την διεπαφή του CS8900A απευθείας με τον E2023 Isolation Transformer που χρησιμοποιήθηκε στην κατασκευή της συγκεκριμένης πλακέτας.
- Ο CS8900A είναι σχεδιασμένος να χρησιμοποιείται σε πλακέτες δύο στρωμάτων, αντί ακριβότερων περισσότερων επιπέδων.

2.3.1 ΥΨΗΛΗ ΑΠΟΔΟΣΗ

Ο CS8900A είναι ένας 16-bit Ethernet Controller σχεδιασμένος να προσφέρει μέγιστη απόδοση του συστήματος ελαχιστοποιώντας τον χρόνο στον ISA bus και στην CPU overhead ανά πακέτο. Προσφέρει παρόμοια ή ανώτερη απόδοση με λιγότερο κόστος συγκριτικά με άλλους Ethernet Controllers.

Η αρχιτεκτονική του επιτρέπει στο χρήστη να έχει πρόσβαση με το λογισμικό του, με όποια μέθοδο πρόσβασης τον διευκολύνει λόγω της ύπαρξης του λεγόμενου από την εταιρία που τον κατασκευάζει και προμηθεύει PacketPage. Είναι το χαρακτηριστικό που τον κάνει γρηγορότερο, απλοϊκότερο και αποδοτικότερο.

Για την περαιτέρω εκτόξευση της απόδοσης του CS8900A, περιέχει ο ίδιος, αρκετά χαρακτηριστικά, μερικά από τα οποία είναι:

- Ο Auto-Switch DMA επιτρέπει στο CS8900A να μεγιστοποιήσει την καλύτερη ροή δεδομένων εξόδου, ενώ ελαχιστοποιεί τα χαμένα πακέτα.
- Σύντομες διακοπές επιτρέπουν στον host να προεπεξεργαστεί τα εισερχόμενα πακέτα.
- Η αποθήκευση των πακέτων που γίνεται on-chip περιορίζει το ποσοστό του Bandwidth που χρειάζεται για τη διαχείριση της κίνησης στο Ethernet.

2.3.2 ΧΑΜΗΛΗ ΤΑΣΗ ΡΕΥΜΑΤΟΣ & ΧΑΜΗΛΟΣ ΘΟΡΥΒΟΣ

Για τις ανάγκες της χαμηλής τάσης, σημαντικός παράγοντας επιλογής του συγκεκριμένου Ethernet Controller, ο CS8900A προσφέρει τρεις επιλογές power-down: Hardware Stand-by, Hardware Suspend και Software Suspend.

Στην πρώτη επιλογή το chip τροφοδοτείται, με εξαίρεση τον 10Base-T δέκτη, ο οποίος ενεργοποιείται για την κινητικότητα σύνδεσης. Στις άλλες δύο περιπτώσεις ο δέκτης είναι απενεργοποιημένος και το κύκλωμα ‘πέφτει’ στην τάξη των mA(μικρο ampere).

Επιπροσθέτως, ο CS8900A είναι σχεδιασμένος για πολύ χαμηλά επίπεδα θορύβου εκπομπής, καθώς ελαχιστοποιεί τον χρόνο που απαιτείται για EMI test και επαλήθευση.

ΚΕΦΑΛΑΙΟ 3ο

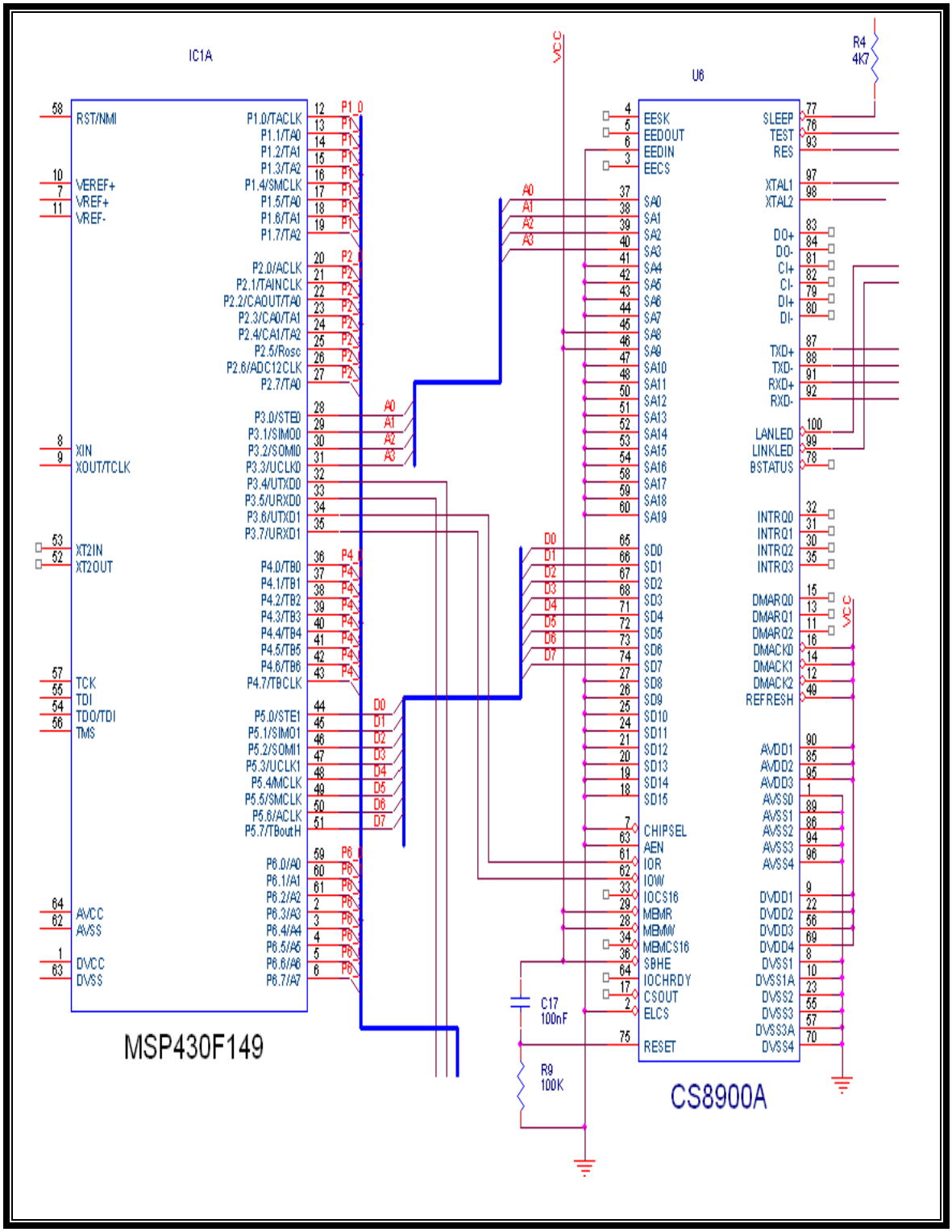
3. ΠΕΡΙΓΡΑΦΗ ΚΥΚΛΩΜΑΤΟΣ

Ο MCU συνδέεται με τον Ethernet Controller στις θύρες P.5 και P.3 αντίστοιχα με τις SD, SA. Πιο συγκεκριμένα τα pins IOR, IOW του CS8900A συνδέονται στα pins P3.6 και P3.7 αντίστοιχα.

Ειδικότερη περιγραφή της αντιστοίχισης φαίνεται στο πίνακα συνδεσιμότητας στην επόμενη σελίδα καθώς και στο Schematic (εικόνα 4).

►Πίνακας 6: Συνδεσιμότητα MSP430F149IPMG4 και CS8900A

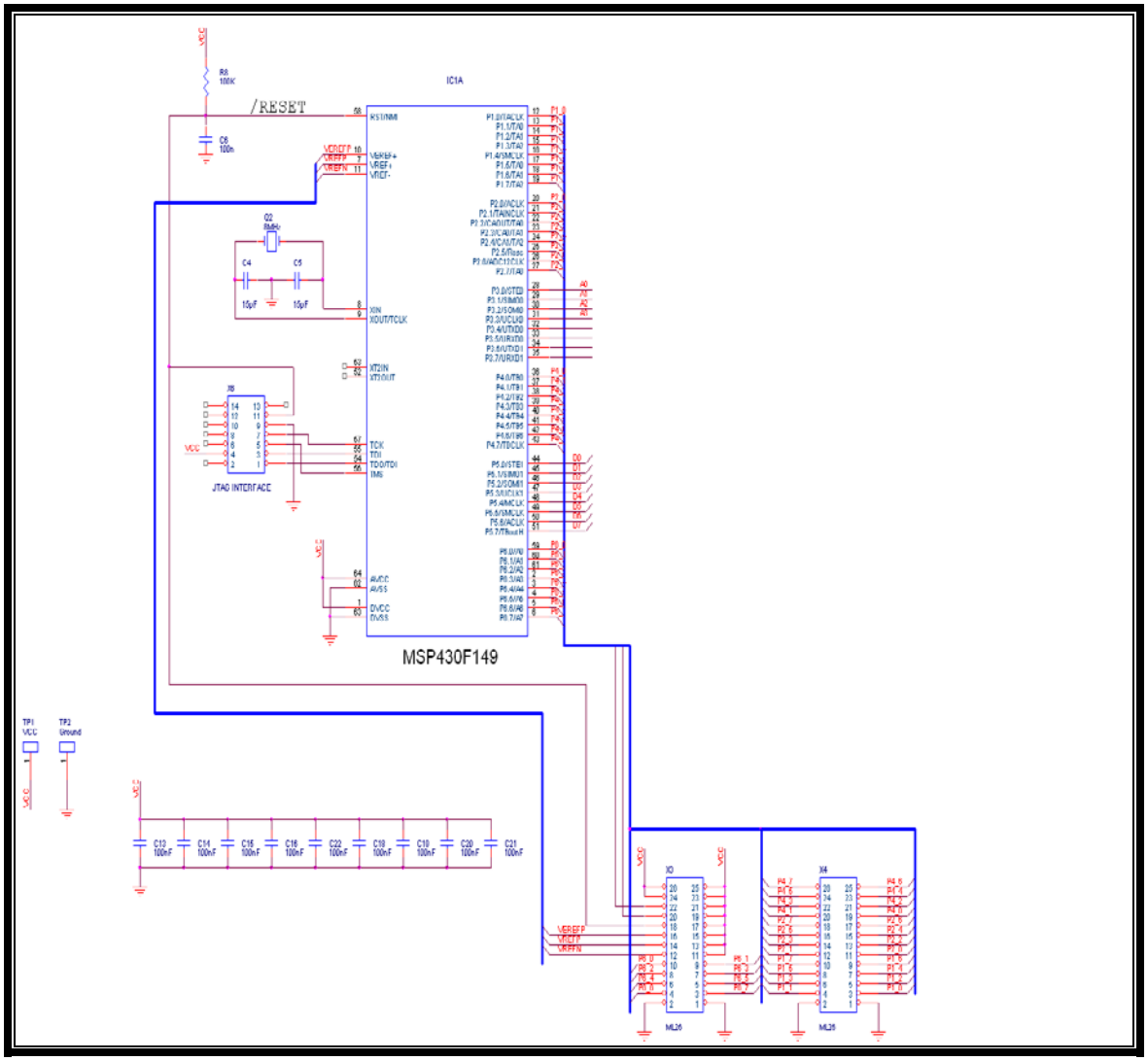
MSP430F149IPMG4 *MCU*	*CS8900A* *ETHERNETCONTROLLER*
PIN number ID	***PIN number ID***
D[7…0]	
#44. P5.0	*#65. SD.0*
#45. P5.1	*#66. SD.1*
#46. P5.2	*#67. SD.2*
#47. P5.3	*#68. SD.3*
#48. P5.4	*#71. SD.4*
#49. P5.5	*#72. SD.5*
#50. P5.6	*#73. SD.6*
#51. P5.7	*#74. SD.7*
A[3…0]	
#28. P3.0	*#37. SA.0*
#29. P3.1	*#38. SA.1*
#30. P3.2	*#39. SA.2*
#31.P3.3	*#40. SA.3*
[IOR & IOW]	
#34. P3.6	*#61. IOR*
#35. P3.7	*#62. IOW*



Εικόνα 4: Συνδεσιμότητα MSP430F149 & CS8900A.

Η καλωδίωση γύρω από το MCU έχει ως εξής τα pins TCK, TDI, TMS συνδέονται αντίστοιχα με τα pins 5,3 και 1 του Header X6, οποίος χρησιμοποιείται για την υλοποίηση του JTAG Interface.

Το JTAG Interface χρησιμοποιείται για προγραμματιστικούς και για debugging σκοπούς. Το pin του MCU, RST/NMI συνδέεται με το pin 11 του Header. Οι θύρες P.1, P.2, P.3, P.4 και P.6 του MCU, με την βοήθεια διαύλων οδηγούνται στους Headers X3 και X4 (βλέπε εικόνα 5).



Εικόνα 5: Καλωδίωση MCU.

Ένα RS232 Interface, μπορεί να χρησιμοποιηθεί εάν χρειαστεί να εγκαταστήσουμε μία SLIP ή PPP Internet σύνδεση. Όλα τα pins του μικροεπεξεργαστή που δεν χρησιμοποιούνται είναι συνδεδεμένα στα pins του Header X3 και X4 (βλέπε εικόνα 7). Ένας 8MHz κρύσταλλος (crystal) είναι συνδεδεμένος στα pins XOUT/TCLK και V_REF/Ve_REF pin 9 και 11 αντίστοιχα του MCU. Τα pins XT2OUT και XT2IN δεν χρησιμοποιούνται (βλέπε εικόνα 6). Η αναλογική καλωδίωση γύρω από τον CS8900A στην πλακέτα περιγράφεται διεξοδικότερα στην πηγή [2]. Ένας 20MHz κρύσταλλος (crystal) είναι συνδεδεμένος στα pins XTAL1 (pin 97) και XTAL2 (pin 98) του CS8900A. Λόγω της χωρητικότητας από κατασκευής των XTAL pins δεν χρειάζονται καθόλου εξωτερικοί πυκνωτές. Το σήμα επανεκκίνησης power-on δημιουργείται από R/C σε συνδιασμό με R9/C17. Ο LAN Controller έχει διαφορετικά pins εξόδου για τον έλεγχο των LED. Το pin 100 (LANLED) αλλάζει σε κατάσταση active-low όταν ο CS8900A εκπέμπει ή λαμβάνει δεδομένα και είναι συνδεδεμένος στο κόκκινο LED (D1). Ένα κίτρινο LED, το οποίο είναι συνδεδεμένο στο pin 99 (LINKLED) ανοίγει εάν βρεθεί κατάλληλος 10BASET παλμός, ενώ το τρίτο LED καταλήγει σε γείωση.

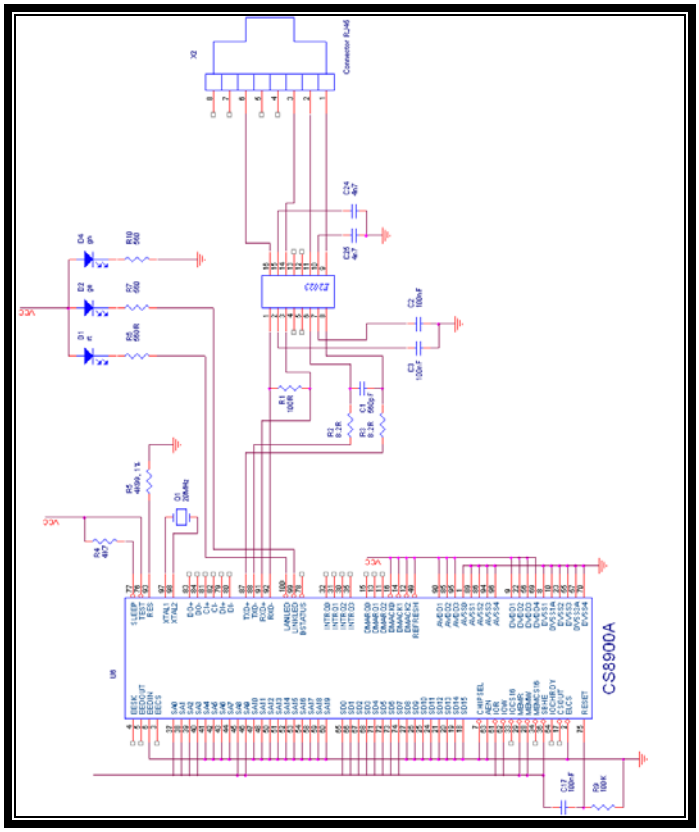
►***Πίνακας 7: Τα pins από το CS8900A που δεν χρησιμοποιούνται είναι τα παρακάτω:***

# pin 48	**Όνομα: SA11**	**# pin 5**	**Όνομα: EEDATAOUT**
# pin 34	**Όνομα: MEMCS16**	**# pin 3**	**Όνομα: EECS**
# pin 33	**Όνομα: IOCS16**	**# pin 17**	**Όνομα: CSOUT**
# pin 64	**Όνομα: IOCHRDY**	**# pin 78**	**Όνομα: BSTATUS/HC1**
# pin 32	**Όνομα: INTRQ0**	**# pin 81**	**Όνομα: CI+**
# pin 31	**Όνομα: INTRQ1**	**# pin 82**	**Όνομα: CI-**
# pin 30	**Όνομα: INTRQ2**	**# pin 79**	**Όνομα: DI+**
# pin 35	**Όνομα: INTRQ3**	**# pin 80**	**Όνομα: DI-**
# pin 15	**Όνομα: DMRQ0**	**# pin 83**	**Όνομα: DO+**
# pin 13	**Όνομα: DMRQ1**	**# pin 84**	**Όνομα:DO-**
# pin 11	**Όνομα: DMRQ2**	**# pin 4**	**Όνομα: EESC**

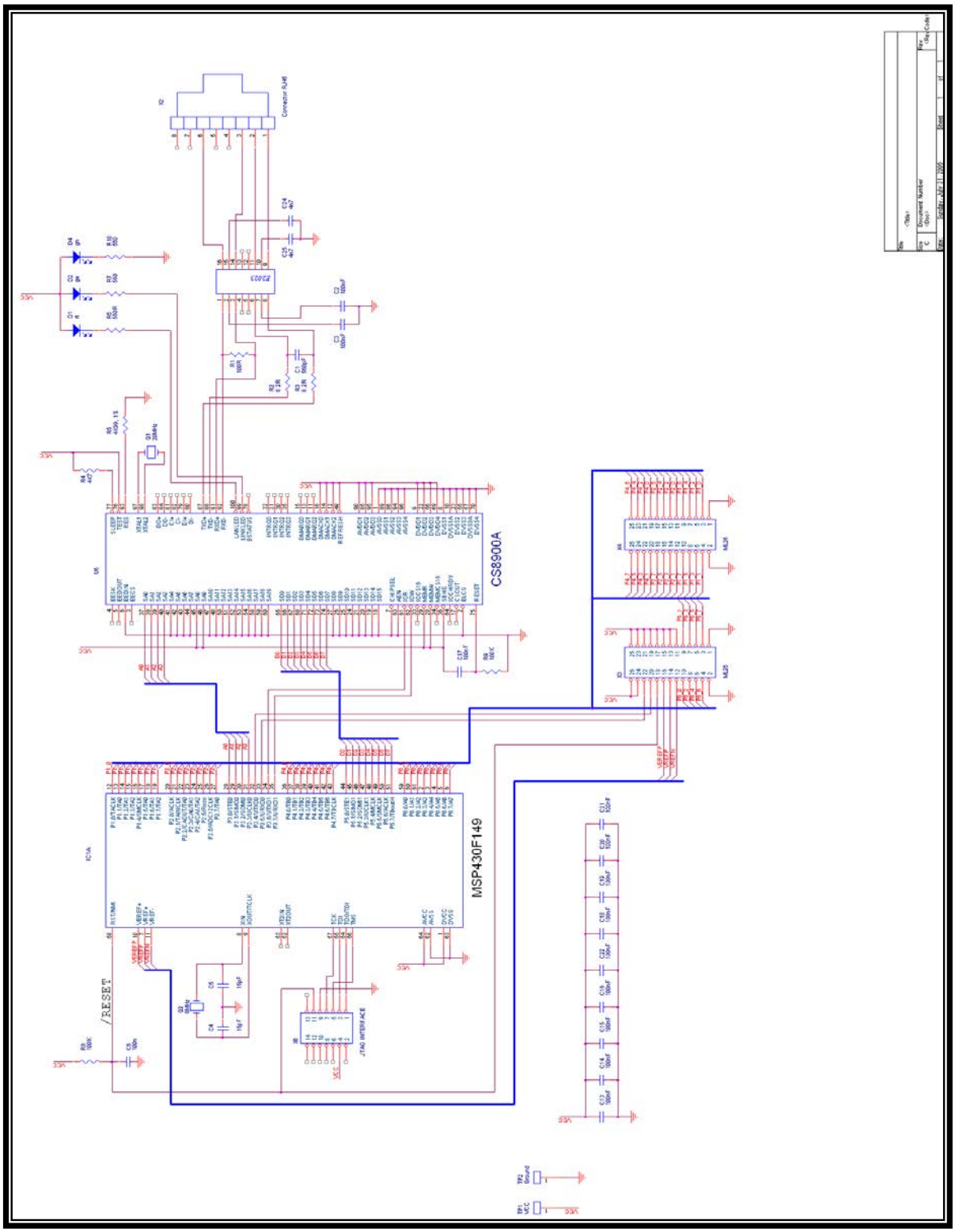
Τα pins TXD+, TXD-, RXD+, RXD-, συνδέονται στα pins 7,6,3 και 1 αντίστοιχα του E2023 Isolation Tranformer. Τα pins 2,7 και 14,11 του E2023 Isolation Tranformer οδηγούνται σε δύο αναλογικά κυκλώματα αποτελούμενα από δύο πυκνωτές το καθένα με τελική κατάληξη δύο γειώσεις αντίστοιχα.

Τα pins του E2023 Isolation Tranformer που δεν χρησιμοποιούνται είναι τα 4, 5, 12, 13. Ο E2023 Isolation Tranformer επικοινωνεί με τον RJ45 LAN Connector, με αντιστοιχία των pins 16→6, 14→3, 11→2, 9→1. Τα pins 4, 5, 7, 8 του RJ45 LAN Connector είναι εσωτερικές γειώσεις του πακέτου, οι οποίες καταλήγουν σε μία εξωτερική όπως φαίνεται στην παρακάτω εικόνα (βλέπε εικόνα 6) και δεν χρησιμοποιούνται για την μεταφορά δεδομένων στην πλακέτα.

Ένα RJ45 καλώδιο χρησιμοποιείτε για να συνδεθεί με ένα10 ή 100Mbs hub. Το 100Mbs hub όταν βρει τον CS8900A να δουλεύει στα 10 Mbs αυτομάτως θα μειώσει την ταχύτητά του στα 10 Mbs.



Εικόνα 6:CS8900A ΜΕ E2023 Isolation Transformer & RJ45 LAN Connector



Εικόνα 7: Διάγραμμα Κυκλώματος Πλακέτας (SCHEMATIC)

ΚΕΦΑΛΑΙΟ 4^{Ο}

4. ΠΕΡΙΓΡΑΦΗ SOFTWARE

Αυτό το κεφάλαιο περιγράφει το ολοκληρωμένο TCP/IP σωρό, τον Ethernet driver, και τον HTTP Server. Ολόκληρος ο κώδικας έχει γραφτεί σε κώδικα της C.

Για λόγους καλύτερης κατανόησης είναι χωρισμένοι σε διαφορετικά μοντέλα (modules). Στον πίνακα της επόμενης σελίδας παρατίθεται μία γενική περιγραφή των μοντέλων αυτών.

Application	• Μεταφέρει δεδομένα μέσω Ethernet & TCP/IP. • Χρησιμοποιεί τις API συναρτήσεις του TCP/IP module, το οποίο ενσωματώνει ολόκληρο το σωρό και τον κρύβει από την εφαρμογή.
TCP/IP module *(tcp/ip.c, tcpip.h)*	• Είναι μία βιβλιοθήκη για ανάπτυξη εφαρμογών. • Δημιουργεί τα πρωτόκολλα ARP, ICMP, IP, TCP. • Αντιδρά σε γεγονότα (frame reception, API calls by the user).
Ethernet module *(cs8900a.c, cs8900a.h, msp430x14x.h)*	• Hardware Drivers για την χρησιμοποίηση του CS8900A LAN Controller. • Παρέχει συναρτήσεις για configuration, για Read/Write Registers και για Αποστολή/Λήψη Ethernet Frames. • Αρχικοποιεί τον μικροεπεξεργαστή
Ethernet	• Το φυσικό επίπεδο στο οποίο έχουμε μεταγωγή των Data.

Πίνακας 8: Βασικά modules Software

4.1 ΜΟΝΤΕΛΟ ETHERNET

Ο βασικός σκοπός του μοντέλου Ethernet cs8900a.c, είναι η ενσωμάτωση συναρτήσεων για την μετάδοση δεδομένων. Επίσης το Ethernet μοντέλο ενεργοποιεί το clock scheme, το οποίο χρησιμοποιείται για την πρόσβαση στους εσωτερικούς registers του CS8900A.

Στο αρχείο επικεφαλίδας cs8900a.h η ποιο σημαντική παράμετρος που αρχικοποιείται είναι η διεύθυνση MAC της διεπαφής του δικτύου.Ο παρακάτω πίνακας μας δίνει μια γενικότερη ιδέα των συναρτήσεων του και το παρακάτω διάγραμμα ροής μας δείχνει μία γενική ιδέα για την λειτουργία της κάθε συνάρτησης.

Όνομα/Παράμετρος:	***Περιγραφή:***
void Init8900(void)	Αρχικοποιεί τα βασικά MCU port pins, κάνει reset στον Lan Controller και θέτει την MAC διεπαφή.
void Write8900(unsigned char Address, unsigned int Data)	Γράφει την 16-bit τιμή των δεδομένων στους έναν από τους οκτώ registers του CS8900A. Μεταφέρει μία λέξη ακολουθία byte στη Lan Controller μνήμη.
void WriteFrame8900(unsigned int Data)	Γράφει την 16-bit τιμή των δεδομένων στην I/O διεύθυνση TX_FRAME_PORT του CS8900A. Χρησιμοποιείται για να μεταφέρει μία λέξη στο transmit buffer.
Unsigned int Read8900(unsigned int Address)	Γράφει την 16-bit τιμή των δεδομένων στην I/O διεύθυνση στον Lan Controller.
unsigned int ReadFrame8900(void)	Γράφει την 16-bit τιμή από την RX_Frame_Port διεύθυνση του Lan Controller. Χρήση για μεταφορά δεδομένων σε MCU μνήμη.

Όνομα/Παράμετρος:	***Περιγραφή:***
unsigned int ReadFrameBE8900(void)	Γράφει την 16-bit τιμή από την διεύθυνση RX_FRAME_PORT του Lan Controller. Χρήση για μεταφορά δεδομένων σε MCU μνήμη. Χρήση αυτής, για αποφυγή συμφόρησης σε Byte όταν διαβάζονται από μεγαλύτερο επίπεδο πρωτοκόλλων.
void CopyToFrame8900(void *Source, unsigned int Size)	Αντιγράφει Size Bytes, ξεκινώντας από τη Source διεύθυνση της MCU μνήμης στην CS8900A TX_FRAME_PORT. Χρησιμοποιείται για την αποστολή ενός ολόκληρου έτοιμου πακέτου.
void CopyFromFrame8900(void *Dest, unsigned int Size)	Αντιγράφει Size Bytes, από την θύρα πακέτων του Lan Controller και τα μεταφέρει στην MCU μνήμη.
void DummyReadFrame8900(unsigned int Size)	Διαβάζει και αποβάλει τα Size bytes από την CS8900A TX_FRAME_PORT. Χρησιμοποιείται για την παραμέριση ενός αποσταλθέντος πακέτου. Το μέγεθος πρέπει να είναι μονός αριθμός.
void RequestSend(unsigned int FrameSize)	Ζητά FrameSize Bytes στην αναμετάδοση του buffer στον Lan Controller. Αυτή η function πρέπει να καλείται πριν από την εγγραφή δεδομένων στην TX_FRAME_PORT.
unsigned int Rdy4Tx(void)	Ελέγχει εάν προηγουμένως ζητήθηκε διαθέσιμος χώρος στο buffer, στον Lan Controller (επιστρέφει τιμή 0 και για αυτόν το λόγο το πακέτο μπορεί να αντιγραφεί στον TX buffer).

Πίνακας 9: Functions του Ethernet Μοντέλου

4.1.1 Επεξήγηση Διαγράμματος Ροής Μοντέλου Ethernet

Αρχικά, καλώντας την συνάρτηση Init8900(),

```
 void Init8900(void)
{
 unsigned int i;
 P3OUT = IOR | IOW;                          // επανεκκίνηση εξόδου, γραμμές ελέγχου high
 P3DIR = 0xff;                               // γραμμή ελέγχου γίνεται έξοδος
 P5OUT = 0;                                  //  επανεκκίνηση εξόδου
 P5DIR = 0xff;                               // γραμμή ελέγχου γίνεται έξοδος
 DelayCycles(40000);                         // delay 10ms @ 8MHz MCLK
 DelayCycles(40000);                         // time for CS8900 POR
 Write8900(ADD_PORT, PP_SelfCTL);            // θέτουμε την register
 Write8900(DATA_PORT, POWER_ON_RESET);        // επανεκκίνηση του Ethernet-Controller
 do
 Write8900(ADD_PORT, PP_SelfST);             // επανεκκίνηση της register
 while (!(Read8900(DATA_PORT) & INIT_DONE)); // αναμονή μέχρι επανεκκίνηση του chip
  for (i = 0; i < sizeof InitSeq / sizeof (TInitSeq); i++)
 {
  Write8900(ADD_PORT, InitSeq[i].Addr);
  Write8900(DATA_PORT, InitSeq[i].Data);
 }
}
```

γίνεται η αρχικοποίηση σε σημαντικά pins του MCU και γίνεται ένα software reset του CS8900A.

Στην συνέχεια καλείται η συνάρτηση Write8900(ADD_PORT,PP_RxEvent).

```
void Write8900(unsigned char Address, unsigned int Data)
{
 P5DIR = 0xff;                          // Θύρα 5 γίνεται έξοδος
 P3OUT = IOR | IOW | Address;           // θέτουμε διεύθυνση στο δίαυλο
 P5OUT = Data;

 P3OUT &= ~IOW;                         // toggle IOW-signal
 P3OUT = IOR | IOW | (Address + 1);     // και θέτουμε την επόμενη διεύθυνση στον δίαυλο

 P5OUT = Data >> 8;
 P3OUT &= ~IOW;                         // toggle IOW-signal
 P3OUT |= IOW;
}
```

Το πρόγραμμα συνεχίζεται σε ένα βρόγχο στον οποίο καλούμαι την συνάρτηση if (Read8900(DATA_PORT)&RX_OK));.

```
unsigned int Read8900(unsigned char Address)
{
 unsigned int ReturnValue;

 P5DIR = 0x00;                          // Θύρα 5 γίνεται είσοδος
 P3OUT = IOR | IOW | Address;           // θέτω διεύθυνση στο δίαυλο
 P3OUT &= ~IOR;                         // IOR-signal low
 ReturnValue = P5IN;

 P3OUT = IOR | IOW | (Address + 1);     // IOR high and και θέτω επόμενη διεύθυνση στο δίαυλο
 P3OUT &= ~IOR;                         // IOR-signal low
 ReturnValue |= P5IN << 8;

 P3OUT |= IOR;
 P5DIR = 0xff;                          // θύρα 5 έξοδος
  return ReturnValue;
}
```

Σε αυτό το σημείο έχουμε δύο καταστάσεις: εάν τα δεδομένα είναι επιθυμητά καλείται η συνάρτηση CopyFromFrame8900(…),

```
void CopyFromFrame8900(void *Dest, unsigned int Size)
{
 unsigned int *pDest = Dest;
 P5DIR = 0x00;                                  // Θύρα 5 γίνεται είσοδος
 while (Size > 1)
 {
  P3OUT = IOR | IOW | RX_FRAME_PORT;            // πρόσβαση στην RX_FRAME_PORT
  P3OUT &= ~IOR;                                // IOR-signal low
  *pDest = P5IN;                                // λαμβάνω το 1ο byte από τον data bus (low-byte)
  P3OUT = IOR | IOW | (RX_FRAME_PORT + 1);      // IOR high και θέτω επόμενη διεύθυνση στον bus
  P3OUT &= ~IOR;                                // IOR-signal low
  *pDest++ |= P5IN << 8;                        // λαμβάνω το 2ο byte από τον data bus (high-byte)
  P3OUT |= IOR;
  Size -= 2;
 }
  if (Size)
 {
  P3OUT = IOR | IOW | RX_FRAME_PORT;            // access to RX_FRAME_PORT
  P3OUT &= ~IOR;                                // IOR-signal low
  *(unsigned char *)pDest = P5IN;               // λαμβάνω byte from data bus
  P3OUT |= IOR;                                 // IOR high
 }
 P5DIR = 0xff;                                  // θύρα 5 έξοδος
}
```

Ειδάλλως συνεχίζεται η ροή του προγράμματος, κατά την οποία μπαίνουμε σε έναν ακόμη βρόγχο που διερωτάται αν η εφαρμογή θέλει να στείλει δεδομένα.

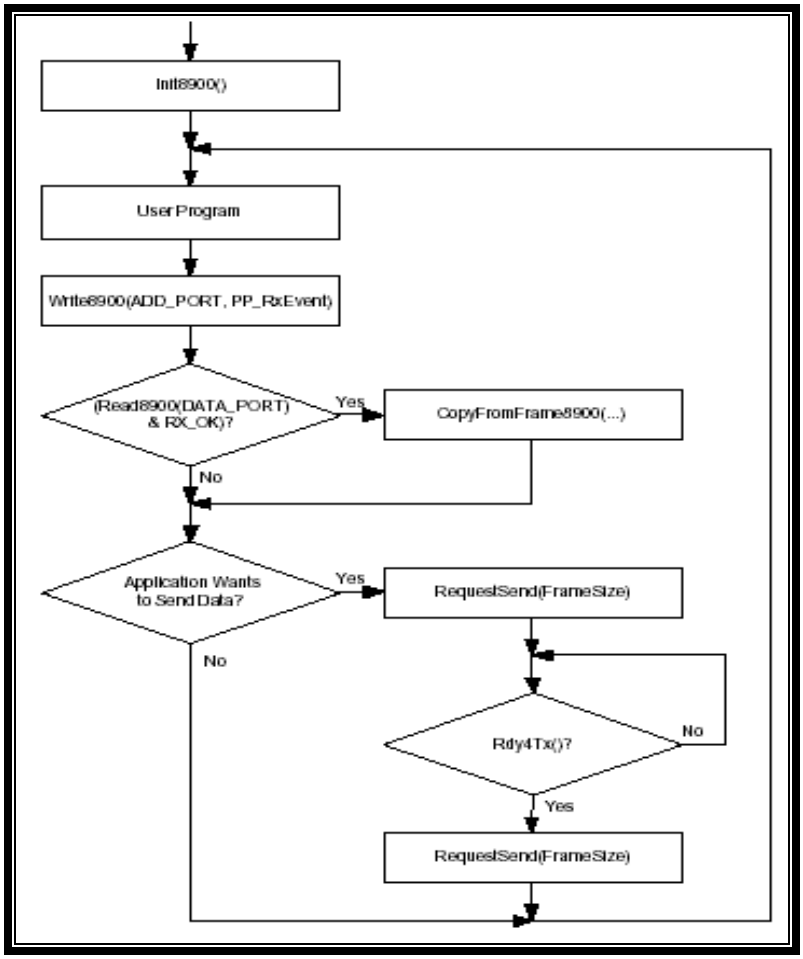
Εάν δεν θέλουμε να στείλουμε δεδομένα τερματίζεται η ροή του προγράμματος, εάν θέλουμε τότε καλείται η συνάρτηση RequestSend(FrameSize).

```
void RequestSend(unsigned int FrameSize)
{
 Write8900(TX_CMD_PORT, TX_START_ALL_BYTES);
 Write8900(TX_LEN_PORT, FrameSize);
}
```

Εφόσον βρισκόμαστε σε αυτήν τη φάση της εφαρμογής όπου καλείται η προαναφερθέντα συνάρτηση μπαίνουμε στον επόμενο βρόγχο όπου καλείται η συνάρτηση Rdy4Tx().

```
 unsigned int Rdy4Tx(void)
{
 Write8900(ADD_PORT, PP_BusST);
 return Read8900(DATA_PORT) & READY_FOR_TX_NOW;
}
```

Εάν τα δεδομένα δεν είναι τα απαιτούμενα ο βρόγχος επιστρέφει στην προηγούμενη κατάσταση, αλλιώς η ροή του προγράμματος συνεχίζεται καλώντας την συνάρτηση RequestSend(FrameSize).



Διάγραμμα 1: Χρησιμοποιώντας το μοντέλο Ethernet.

4.2 Μοντέλο TCP/IP

Το μοντέλο αυτό αναπαριστά το πιο σημαντικό κομμάτι της πλακέτας, διότι τα πρωτόκολλα για την μεταφορά δεδομένων μέσω του TCP/IP δημιουργούνται εδώ. Φυσικά χρησιμοποιεί συναρτήσεις από το Ethernet μοντέλο για την μεταφορά δεδομένων.

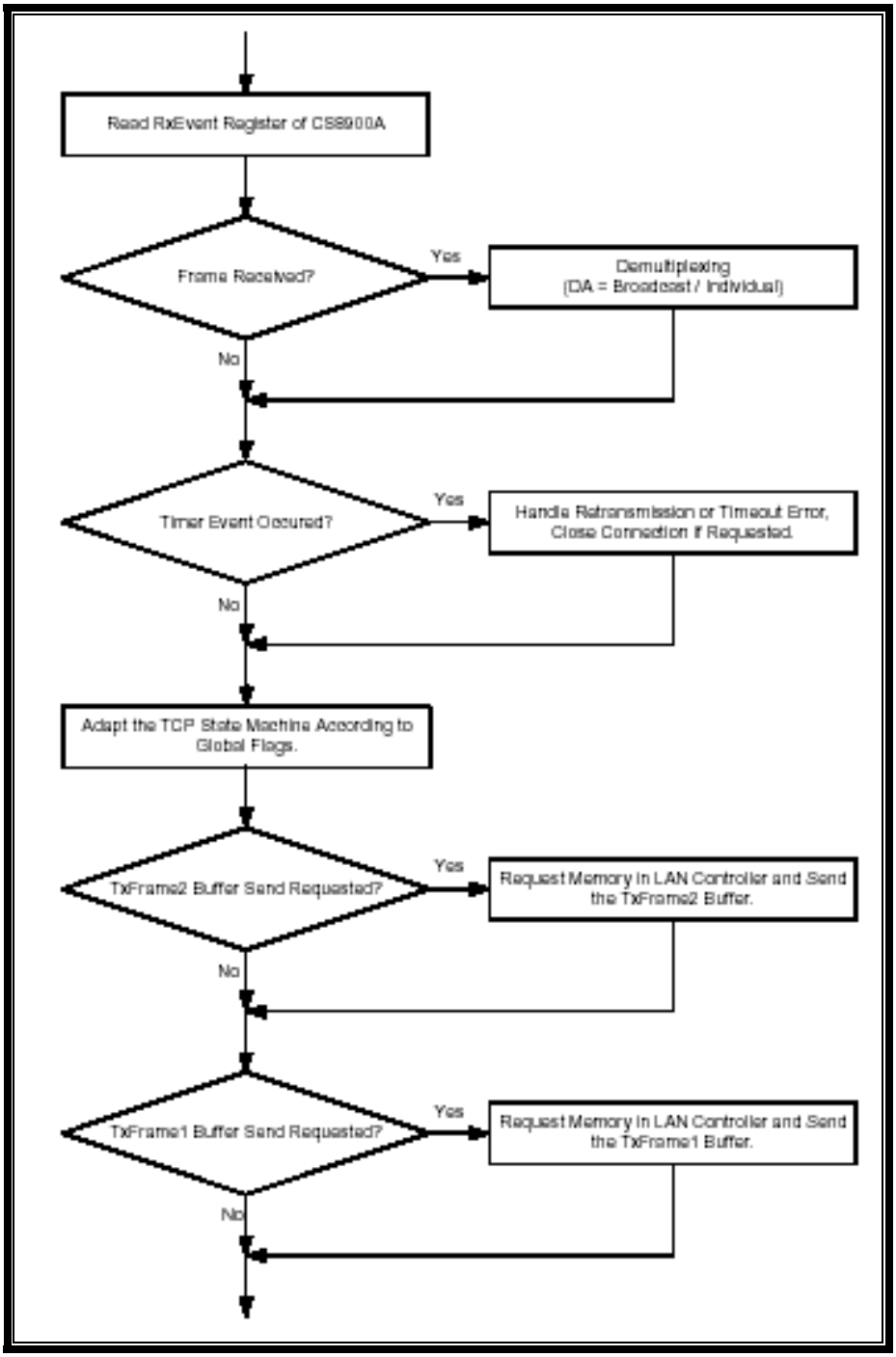
Σε ένα TCP/IP σωρό μπορούν να συμβούν τα παρακάτω γεγονότα:

- Ένα Frame λαμβάνεται από το LAN.
- Η εφαρμογή ξεκινάει ένα γεγονός (για παράδειγμα, μεταφορά δεδομένων, άνοιγμα μίας σύνδεσης…).
- Όταν ένα σφάλμα συμβαίνει (σφάλμα δικτύου).

Το software εμπεριέχει όλα τα μέρη των βασικών RFC 791, 792 και 793. Η πιο σημαντική συνάρτηση του σωρού είναι η DoNetworkStuff().

- Στα εισερχόμενα Frames κάνει Decoding & Receiving.
- Ελέγχει για Timeouts στο TCP/IP.
- Απαντάει σε ARP (request).
- Κάνει Update στη μεταβλητή Socket Status.

Αυτή η συνάρτηση πρέπει να καλείται περιοδικά, εδώ γίνεται ο χειρισμός του TCP/IP. Όσο πιο συχνά καλείται αυτή η συνάρτηση τόσο καλύτερη επίδοση έχει το TCP.



Διάγραμμα 2: Διάγραμμα Ροής DoNetworkStuff().

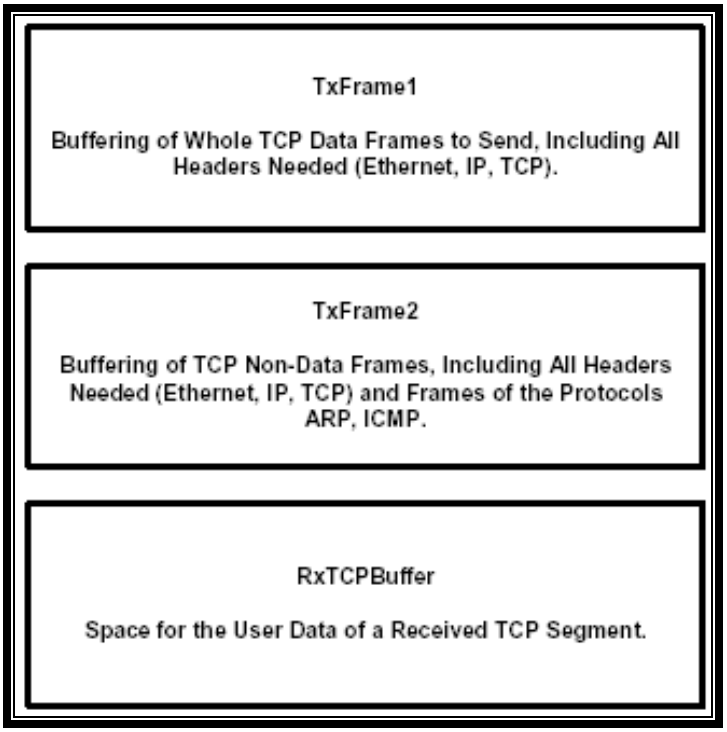
```
void DoNetworkStuff(void)
{
 unsigned int ActRxEvent;
 Write8900(ADD_PORT, PP_RxEvent);                    // point to RxEvent
 ActRxEvent = Read8900(DATA_PORT);
 if (ActRxEvent & RX_OK)
 {
  if (ActRxEvent & RX_IA) ProcessEthIAFrame();
  if (ActRxEvent & RX_BROADCAST) ProcessEthBroadcastFrame();
 }
 if (TCPFlags & TCP_TIMER_RUNNING)
  if (TCPFlags & TIMER_TYPE_RETRY)
  {
   if (TCPTimer > RETRY_TIMEOUT)
   {
    TCPRestartTimer();
    if (RetryCounter)
    {
     TCPHandleRetransmission();
     RetryCounter--;
    }
    else
    {
     TCPStopTimer();
     TCPHandleTimeout();
    }    }   }
  else if (TCPTimer > FIN_TIMEOUT)
  {
   TCPStateMachine = CLOSED;
   TCPFlags = 0;                                     // επανεκκινεί τα flags, σταματά την επαναποστολή
   SocketStatus &= SOCK_DATA_AVAILABLE;
  }
 switch (TCPStateMachine)
 {
  case CLOSED :
  case LISTENING :
   if (TCPFlags & TCP_ACTIVE_OPEN)
    if (TCPFlags & IP_ADDR_RESOLVED)
     if (!(TransmitControl & SEND_FRAME2))
     {
      TCPSeqNr = ((unsigned long)ISNGenHigh << 16) | TAR;
      TCPUNASeqNr = TCPSeqNr;
      TCPAckNr = 0;
      TCPUNASeqNr++;
      PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_SYN);
      LastFrameSent = TCP_SYN_FRAME;
      TCPStartRetryTimer();
      TCPStateMachine = SYN_SENT;
     }
   break;
  case SYN_RECD :
  case ESTABLISHED :
```

```
    if (TCPFlags & TCP_CLOSE_REQUESTED)
      if (!(TransmitControl & (SEND_FRAME2 | SEND_FRAME1)))
        if (TCPSeqNr == TCPUNASeqNr)
        {
          TCPUNASeqNr++;
          PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_FIN | TCP_CODE_ACK);
          LastFrameSent = TCP_FIN_FRAME;
          TCPStartRetryTimer();
          TCPStateMachine = FIN_WAIT_1;
        }
    break;
  case CLOSE_WAIT :
    if (!(TransmitControl & (SEND_FRAME2 | SEND_FRAME1)))
      if (TCPSeqNr == TCPUNASeqNr)
      {
        TCPUNASeqNr++;
        PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_FIN | TCP_CODE_ACK);  // we NEED
a retry-timeout
        LastFrameSent = TCP_FIN_FRAME;
        TCPStartRetryTimer();
        TCPStateMachine = LAST_ACK;
      }
    break;
 }
 if (TransmitControl & SEND_FRAME2)
 {    RequestSend(TxFrame2Size);
  if (Rdy4Tx())
  {
    CopyToFrame8900((unsigned char *)TxFrame2Mem, TxFrame2Size);
  }
  else
  {
    TCPStateMachine = CLOSED;
    SocketStatus = SOCK_ERR_ETHERNET;         // δείχνει λάθος στο χρήση
    TCPFlags = 0;                              // clear all flags, stop timers etc
  }
  TransmitControl &= ~SEND_FRAME2;             // clear tx-flag
 }
 if (TransmitControl & SEND_FRAME1)
 {    PrepareTCP_DATA_FRAME();
  RequestSend(TxFrame1Size);
  if (Rdy4Tx())
  {
    CopyToFrame8900((unsigned char *)TxFrame1Mem, TxFrame1Size);
  }
  else
  {
    TCPStateMachine = CLOSED;
    SocketStatus = SOCK_ERR_ETHERNET;         //δείχνει λάθος στο χρήστη
    TCPFlags = 0;                              // clear all flags, stop timers etc.
  }
  TransmitControl &= ~SEND_FRAME1;           // clear tx-flag  }}
```

4.2.1 ΑΠΟΘΗΚΕΥΤΙΚΗ ΜΝΗΜΗ

Για τα εισερχόμενα και εξερχόμενα πακέτα τρεις αποθηκευτικές μνήμες (buffer memories) έχουν κρατηθεί στην SRAM. Το μέγεθος των μνημών αυτών μπορεί να τροποποιηθεί, τροποποιώντας τον κώδικα στις σταθερές μεταβλητές, στο αρχείο επικεφαλίδας tcpip.h.

Μεγαλώνοντας το μέγεθος των buffer αυξάνεται η ταχύτητα μεταφοράς διότι ο κώδικας χρησιμοποιεί μόνο ένα buffer για την εκπομπή δεδομένων στο TCP/IP. Το buffer θα πάρει καινούρια δεδομένα μόνο όταν το τελευταίο κομμάτι δεδομένων έχει σταλεί στο παραλήπτη.

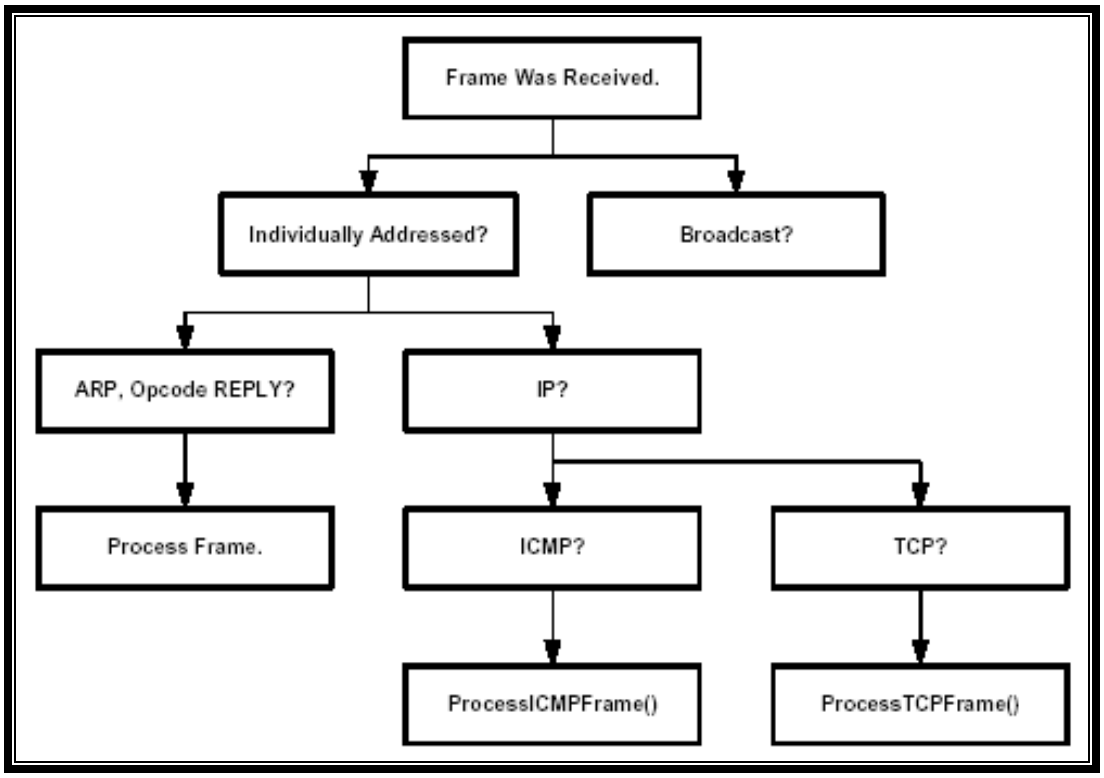


Σχήμα 3: Βασική ιδέα διαχείρισης Buffer

4.2.2 ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΣΕΡΧΟΜΕΝΩΝ ΠΑΚΕΤΩΝ ΔΕΔΟΜΕΝΩΝ (Frames)

Η συνάρτηση DoNetworkStuff() ελέγχει πότε ένα Frame έχει εισέλθει. Μετά τον έλεγχο του προορισμού του πακέτου, για να εξακριβωθεί εάν το πακέτο είναι broadcast, η συνάρτηση ProcessEthIAFrame() καλείται.

Στο παρακάτω διάγραμμα δίνεται μία γενική ιδέα για το ποια διαδικασία καλείται, σε συνάρτηση με τον τύπο του πακέτου. Εάν είναι broadcast πακέτο και ο τύπος του είναι ARPrequest ένα ARP απάντηση πακέτο δημιουργείται και γράφεται στο TxFrame2 buffer.



Διάγραμμα 3: Επεξεργασία Εισερχόμενων Πακέτων

Εάν το πακέτο έχει συγκεκριμένη διεύθυνση, αρχικά ελέγχεται εάν είναι ένα ARP απάντηση πακέτο. Σε αυτήν την περίπτωση η διεύθυνση MAC του αποστολέα εξάγεται και ο μηχανισμός αυτός χρησιμοποιείται για ένα δυναμικό άνοιγμα (Active Open) μίας σύνδεσης.

Στην περίπτωση που το εισερχόμενο πακέτο είναι τύπου IP μετακινούμαστε στην συνάρτηση ProcessICMPFrame() η οποία και εκτελείται εξαρτώμενη από το IP Protocol number.

Η συνάρτηση ProcessICMPFrame() ελέγχει εάν το πακέτο είναι ένα ICMP echo αίτησης και δημιουργεί ένα ICMP echo απάντησης. Όλα τα υπόλοιπα ICMP δεν λαμβάνονται υπόψιν.

Στη συνέχεια ελέγχεται εάν το πακέτο είναι TCP και εάν υπάρχει σε εξέλιξη ένα TCP session και ελέγχεται εάν το πακέτο ανήκει σε αυτό, ή είναι ένα πακέτο αίτησης για την εγκατάσταση μίας σύνδεσης.

Όταν τα δεδομένα στέλνονται και η μνήμη είναι ελεύθερη αυτά τα δεδομένα αντιγράφονται στο RxTCPBuffer. Το πρόγραμμα δέχεται πακέτα κατάλληλα αριθμημένα και για αυτόν το λόγο πακέτα που δεν έχουν εισέλθει με τη σειρά που θα έπρεπε αποβάλλονται λόγω της περιορισμένης μνήμης

4.2.3 ΔΗΜΙΟΥΡΓΩΝΤΑΣ ΜΙΑ ΣΥΝΔΕΣΗ

Μία σύνδεση μπορεί να δημιουργηθεί είτε με τρόπο ενεργητικό ή παθητικό, καλώντας την κατάλληλη συνάρτηση, δηλαδή ή την TCPPassiveOpen() ή την TCPActiveOpen().

Η συνάρτηση TCPPassiveOpen() τοποθετεί τον σωρό σε αναμονή για μία εισερχόμενη σύνδεση. Η TCPLocalPort μεταβλητή θα πρέπει να ορίσει ποια θύρα ο σωρός πρέπει να ελέγχει. Πριν ξεκινήσουμε ένα Active Open η IP διεύθυνση καθώς και οι θύρες θα πρέπει να οριστούν.

Αφότου καλέσουμε τη συνάρτηση TCPActineOpen() ο σωρός αρχικά προσπαθεί να ορίσει τη διεύθυνση MAC του αιτώντας στέλνοντας μία ARP αίτηση. Ο σωρός ελέγχει τον προορισμό της IP για να διευκρινίσει εάν ο αιτών είναι μέλος του υποδικτύου.

Εάν ο αιτών δεν είναι μέλος του υποδικτύου η ARP αίτηση γίνεται στην προκαθορισμένη πύλη (Gateway), και τα δεδομένα μεταφέρονται από την πύλη σαν router [3].

Από την στιγμή που θα βρεθεί η MAC διεύθυνση για να γίνει η επικοινωνία, ο σωρός στέλνει ένα TCP πακέτο που περιέχει ένα SYN flag και το TCP option MSS (maximum segement size).

Μαζί με το πακέτο αυτό ο αρχικός αριθμός της σειράς δεδομένων που χρειάζονται για την εγκατάσταση μίας σύνδεσης και επίσης το μέγεθος του λαμβανόμενου buffer.

4.2.4 ΜΕΤΑΦΟΡΑ ΔΕΔΟΜΕΝΩΝ

Μόλις μια σύνδεση εγκατασταθεί η μεταφορά δεδομένων μπορεί να γίνει. Η αποστολή και η λήψη των δεδομένων στο buffer γίνεται καλώντας την συνάρτηση ProcessTCPFrame(). Ο αριθμός των εισερχόμενων bytes αποθηκεύεται στην global μεταβλητή TCPRxDataCount. Τα καινούργια δεδομένα μπορούν να ληφθούν μόνο όταν η μνήμη είναι ελεύθερη.

Για να ελευθερώσουμε την μνήμη καλείται η συνάρτηση TCPReleaseRxBuffer(). Για την αποστολή δεδομένων στον παραλήπτη η εφαρμογή πρέπει να αποθηκεύσει αυτά τα δεδομένα στο TxFrame1 buffer, για την άμεση πρόσβαση σε αυτήν τη περιοχή χρησιμοποιείται ο δείκτης TCP_TX_BUF. Την αποστολή την ξεκινάει τελικά η συνάρτηση TCPTransmitTxBuffer(), η οποία ελέγχει εάν η αποστολή μπορεί να γίνει.

Για να αποφύγουμε μία πιθανή αστοχία μίας TCP σύνδεσης που δημιουργείται είτε από χαμένα αποστελλόμενα δεδομένα, ένας μηχανισμός επανεκπομπής δεδομένων δημιουργείται. Ο TCP σωρός καταγράφει τον τύπο του κάθε πακέτου που αποστάλθηκε στον LastFrameSent καταχωρητή.

Όταν ο μετρητής χρόνου (time counter) μπορεί μεγαλύτερη τιμή από τη μεταβλητή RETRY-TIMEOUT (tcpip.h), το πακέτο επανεκπέμπεται χρησιμοποιώντας τη συνάρτησηTCPHandleRetransmission().

Ο μέγιστος αριθμός επανεκπομπών πακέτων ορίζεται στη μεταβλητή MAX_RETRYS. Εάν αυτός ο μετρητής πάρει μεγαλύτερη τιμή από την απαιτούμενη η TCP σύνδεση κλείνει.

4.2.5 ΚΛΕΙΣΙΜΟ ΜΙΑΣ ΣΥΝΔΕΣΗΣ

Μία TCP σύνδεση μπορεί να κλείσει με διαφορετικούς τρόπους. Κανονικά αυτό γίνεται είτε τοπικά καλώντας τη συνάρτηση TCPClose(), είτε από τον άλλον χρήστη της σύνδεσης. Κατά το κλείσιμο μίας σύνδεσης πακέτα που περιέχουν FIN flags ανταλλάσσονται.

4.2.6 ΧΡΗΣΙΜΟΤΗΤΑ ΤΩΝ TIMERS

Κατά την υλοποίηση τουTCP διαφορετικά έργα πρέπει να γίνονται με τέλειο συγχρονισμό. Ο σωρός χρησιμοποιεί τον Timer_A του MSP430 που περιγράφεται από την συνάρτηση TCPLowLevelInit(). Το πρωτόκολλο TCP απαιτεί ένα 32-bit μετρητή (counter) ο οποίος δουλεύει στα 250 kHz.. Χρησιμοποιείτε για να δέχεται τον ISN (initial sequence number) αριθμό που χρησιμοποιείτε για να ξεκινήσει μια σύνδεση. Ο 8-MHz κρύσταλλος "οδηγεί"τον επεξεργαστή . Σε κάθε timer που συμβαίνει κάποιο "γεγονός" η μεταβλητή TCPTimer αυξάνει.

4.2.7 υναρτήσεις TCP/IP Σωρού

▸Void TCPLowLevelInit(void)

Αυτή η συνάρτηση κάνει τη βασική εγκατάσταση του Ethernet Controller και άλλων μεταβλητών.

Επίσης αρχικοποιεί τις θύρες και το μετρητή A (Timer A) του MCU. Πρέπει πάντα να καλείται πριν γίνει αποστολή δεδομένων.

```
void TCPLowLevelInit(void)
{  BCSCTL1 &= ~DIVA0;                                  // ACLK = XT1 / 4 = 2 MHz
  BCSCTL1 |= DIVA1;
  TACTL = ID_3 + TASSEL_1 + MC_2 + TAIE;        // παύση timer use ACLK / 8 = 250 kHz, gen. int.
                                                 // εκκίνηση  timer in continuous up-mode
  Init8900();
  TransmitControl = 0;
  TCPFlags = 0;
  TCPStateMachine = CLOSED;
  SocketStatus = 0; }
```

► VoidTCPPassiveOpen(void)

Όταν καλείται αυτή η συνάρτηση ο σωρός αλλάζει στον server mode για να ελέγξει αν υπάρχει εισερχόμενη σύνδεση. Το flag SOCK_ACTIVE αρχικοποιείται δείχνοντας ότι ο σωρός είναι απασχολημένος.

Πριν καλέσουμε αυτήν την συνάρτηση το τοπικό TCP port πρέπει να αρχικοποιηθεί. Η τοπική IP διεύθυνση διευκρινίζεται από μία σταθερή μεταβλητή στο αρχείο επικεφαλίδας tcpip.h.

```
        TCPLocalPort = 2025; // θύρα του HTTP server
        TCPActiveOpen();      // έλεγχος για εισερχόμενες συνδέσεις
(…)
void TCPPassiveOpen(void)
{
 if (TCPStateMachine == CLOSED)
 {
  TCPFlags &= ~TCP_ACTIVE_OPEN;          //  δυναμικό άνοιγμα (a passive open!)
  TCPStateMachine = LISTENING;
  SocketStatus = SOCK_ACTIVE;            // reset, socket now active
 }
```

► void TCPActiveOpen(void)

Αυτή η συνάρτηση δημιουργεί μία σύνδεση σε ένα απομακρυσμένο TCP server. Το flag SOCK_ACTIVE αρχικοποιείται και στέλνει ένα ARP μήνυμα αίτησης για να βρει τη διεύθυνση MAC του TCP. Εάν η IP διεύθυνση του προορισμού δεν ανήκει στο συγκεκριμένο υποδίκτυο, τότε η IP διεύθυνση της πύλης (Gateway) χρησιμοποιείται για την αποστολή του ARP μηνύματος.

Οι IP διευθύνσεις και του τοπικού και του απομακρυσμένου TCP port πρέπει να αρχικοποιηθούν πριν ανοίξουμε μία σύνδεση με τη συνάρτηση αυτήν.

Μετά το άνοιγμα αυτής της σύνδεσης αρχικοποιείται το flag SOCK_CONNECTED και η μεταφορά δεδομένων μπορεί να γίνει ελεύθερα χρησιμοποιώντας την κατάλληλη συνάρτηση.

Στην συνέχεια παρατίθεται ένα παράδειγμα για το πώς δημιουργούμε μία active-open σύνδεση.

```
*(unsigned char *)RemoteIP = 24;            // uncomment those lines to get the
*((unsigned char *)RemoteIP + 1) = 8;        // quote of the day from a real
*((unsigned char *)RemoteIP + 2) = 69;       // internet server! (gateway must be
*((unsigned char *)RemoteIP + 3) = 7;        // set to your LAN-router)
TCPLocalPort = 2025;
TCPRemotePort = TCP_PORT_QOTD;
TCPActiveOpen();
while (SocketStatus & SOCK_ACTIVE)           // read the quote from memory
{                                            // by using the hardware-debugger
  DoNetworkStuff();
}
```

```
void TCPActiveOpen(void)
{
 if ((TCPStateMachine == CLOSED) || (TCPStateMachine == LISTENING))
 {
   TCPFlags |= TCP_ACTIVE_OPEN;              //  an active open!
   TCPFlags &= ~IP_ADDR_RESOLVED;            // we haven't opponents MAC yet

   PrepareARP_REQUEST();                     // ζητά μιαMACστέλνοντας broadcast
   LastFrameSent = ARP_REQUEST;
   TCPStartRetryTimer();
   SocketStatus = SOCK_ACTIVE; }}            // reset, socket now active
```

► Void TCPClose(void)

Αυτή η συνάρτηση σταματά μία σύνδεση. πριν γίνει η αποσύνδεση ο σωρός εξασφαλίζει ότι ένα πακέτο στο buffer εξόδου έχει μεταδοθεί και διαβαστεί σωστά. Μετά το κλείσιμο της σύνδεσης η εφαρμογή πρέπει να επαναρχικοποιήσει IP διευθύνσεις αριθμούς θυρών, για το άνοιγμα μίας καινούριας σύνδεσης.

```
 void TCPClose(void)
{
 switch (TCPStateMachine)
 {
  case LISTENING :
  case SYN_SENT :
    TCPStateMachine = CLOSED;
    TCPFlags = 0;
    SocketStatus = 0;
    break;
  case SYN_RECD :
  case ESTABLISHED :
    TCPFlags |= TCP_CLOSE_REQUESTED;
    break;
 }
        }
```

► void TCPRealeseRxBuffer(void)

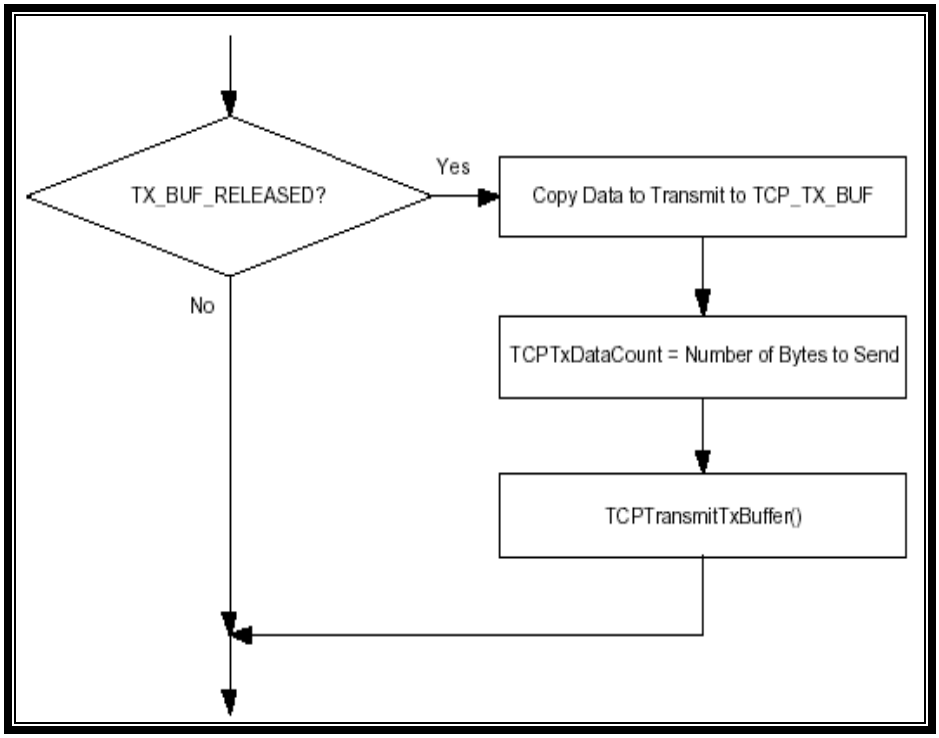
Μετά το διάβασμα της μνήμης εισερχομένων, καλώντας αυτήν τη συνάρτηση, ειδοποιείται ο σωρός ότι τα περιεχόμενα της μνήμης αυτής δεν χρησιμοποιούνται πια και μπορεί να γραφτούν άλλα. Αυτή η συνάρτηση, επίσης καθαρίζει το SOCK_TX_BUF_RELEASED flag που μας δείχνει αν υπάρχουν δεδομένα.

```
void TCPReleaseRxBuffer(void)
{
 SocketStatus &= ~SOCK_DATA_AVAILABLE;
}
```

► Void TCPTransmitTxBuffer(void)

Χρησιμοποιώντας αυτήν την συνάρτηση, μία εφαρμογή μπορεί να στείλει δεδομένα διαμέσου μίας ήδη ανοιχτής σύνδεσης. Αρχικά η εφαρμογή ελέγχει αν μπορεί να γράψει δεδομένα στο buffer μετάδοσης. Αυτό γίνεται στέλνοντας το flag SOCK_TX_BUF_RELEASED του SocketStatus register.

Εάν αυτό το flag είναι ορισμένο η εφαρμογή μπορεί να γράψει ένα μέγιστο μέγεθος δεδομένων (MAX_TCP_TX_DATA_SIZE) στο buffer μετάδοσης, ξεκινώντας από τη διεύθυνση TCP_TX_BUF. Στη συνέχεια ο μετρητής των bytes πρέπει να γραφτεί στον TCPTxDataCount register. Τέλος, η TCPTransmitTxBuffer συνάρτηση καλείται και τα δεδομένα αποστέλλονται.



Διάγραμμα 4: Μετάδοση Δεδομένων

```
void TCPTransmitTxBuffer(void)
{  if ((TCPStateMachine == ESTABLISHED) || (TCPStateMachine == CLOSE_WAIT))
   if (SocketStatus & SOCK_TX_BUF_RELEASED)
   {     SocketStatus &= ~SOCK_TX_BUF_RELEASED;            // occupy tx-buffer
    TCPUNASeqNr += TCPTxDataCount;                  // advance UNA
        TxFrame1Size = ETH_HEADER_SIZE + IP_HEADER_SIZE + TCP_HEADER_SIZE +
TCPTxDataCount;
    TransmitControl |= SEND_FRAME1;
        LastFrameSent = TCP_DATA_FRAME;
    TCPStartRetryTimer();
   }}
```

Συμπερασματικά, η μέγιστη ταχύτητα μεταφοράς του μοντέλου δεν μπορεί να συγκεκριμενοποιηθεί λόγω του ότι εξαρτάται περισσότερο από τον άλλον ‘χρήστη’ (δηλαδή τον άλλον σταθμό εργασίας με τον οποίο κάνει σύνδεση).

Κανονικά στο TCP πρωτόκολλο έχουμε τη δυνατότητα να λαμβάνουμε και να αποθηκεύουμε περισσότερα από ένα πακέτα κάθε φορά.

Επειδή όμως ο MSP430F149 έχει μικρή μνήμη σε σχέση με τους προσωπικούς υπολογιστές, μπορεί να λάβει/αποστείλει ένα πακέτο δεδομένων κάθε φορά.

ΚΕΦΑΛΑΙΟ 5ο

5.1 ΠΕΡΙΓΡΑΦΗ SOFTWARE HTTP SERVER

Για την κατανόηση της χρήσης του σωρού TCP/IP που υλοποιείται στο αρχείο κώδικα tcpip.c [παράρτημα Α], δημιουργήθηκε ένας HTTP Server. Αυτός ο Server δημιουργεί HTML ιστοσελίδα η οποία είναι αποθηκευμένη στη Flash μνήμη του MCU.

Η υλοποίηση του HTTP Server γίνεται στο αρχείο easyweb.c [παράρτημα Α] και η πιο σημαντική του συνάρτηση είναι η HTTPServer( ). Το διάγραμμα (4), μας δείχνει το διάγραμμα ροής του μοντέλου αυτού. Μετά την αρχικοποίηση μέρους του Hardware και του ίδιου του σωρού η τοπική TCPPort παίρνει την τιμή 80, τιμή η οποία είναι στάνταρ για ένα HTTP Server. Ο Server περιμένει για ένα Client να συνδεθεί.

Κατά την πρώτη εκτέλεση της συνάρτησης HTTPServer( ) μετά τη σύνδεση η flag HTTP_SEND_PAGE στη μεταβλητή HTTPStatus είναι άδεια.

```
static void HTTPServer(void)
{
 if (SocketStatus & SOCK_CONNECTED)            // έλεγχος αν κάποιος έχει συνδεθεί
 {
  if (SocketStatus & SOCK_DATA_AVAILABLE)      // έλεγχος αν απομακρυσμένοςTCP έστειλε data
   TCPReleaseRxBuffer();                       // και αποβάλλεται

  if (SocketStatus & SOCK_TX_BUF_RELEASED)     // έλεγχος αν το buffer είναι ελεύθερο για TX
  {
   if (!(HTTPStatus & HTTP_SEND_PAGE))         // αρχικοποίηση byte-counter και pointer στο webside
   {                                  // if called the 1st time
    HTTPBytesToSend = sizeof(WebSide) - 1;          // πάρε HTML μήκος, αγνόησε το trailing zero
    PWebSide = (unsigned char *)WebSide;            // δείκτης στον  HTML-code
   }

   if (HTTPBytesToSend > MAX_TCP_TX_DATA_SIZE)    // εκπομπή ενός segment  MAX_SIZE
   {
    if (!(HTTPStatus & HTTP_SEND_PAGE))            // 1η φορά, include HTTP-header
    {
     memcpy(TCP_TX_BUF, GetResponse, sizeof(GetResponse) - 1);
     memcpy(TCP_TX_BUF + sizeof(GetResponse) - 1, PWebSide,
      MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1);
     HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
     PWebSide += MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
    }
    else
    {
     memcpy(TCP_TX_BUF, PWebSide, MAX_TCP_TX_DATA_SIZE);
     HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE;
     PWebSide += MAX_TCP_TX_DATA_SIZE;       }

    TCPTxDataCount = MAX_TCP_TX_DATA_SIZE;   // bytes to xfer
    InsertDynamicValues();                   // συναλλαγή μερικών strings...
    TCPTransmitTxBuffer();                   // xfer buffer
   }
   else if (HTTPBytesToSend)                 // εκπομπή εναπομείναντα  bytes
   {
    memcpy(TCP_TX_BUF, PWebSide, HTTPBytesToSend);
    TCPTxDataCount = HTTPBytesToSend;        // bytes to xfer
    InsertDynamicValues();                   // συναλλαγή μερικών strings...
    TCPTransmitTxBuffer();                   // στείλε τελευταίο segment
    TCPClose();                              // και κλείσε την συνδεση
    HTTPBytesToSend = 0;                     // όλα τα data στάλθηκαν
   }     HTTPStatus |= HTTP_SEND_PAGE;       // 1ο loop εκτελέστεικε
  } }
 else
  HTTPStatus &= ~HTTP_SEND_PAGE;             // reset την help-flag εά δεν έχει συνδεθεί
}
```

Η μεταβλητή αυτή χρησιμοποιείται για να επεξεργαστούν κάποια ειδικά κομμάτια κώδικα κατά την πρώτη εκτέλεση της συνάρτησης HTTPServer( ). Ο Web Server ελέγχει για εισερχόμενα TCP δεδομένα και τα απορρίπτει. Εδώ γίνεται υπόθεση ότι τα εισερχόμενα δεδομένα εμπεριέχουν μία GET-REQUEST από έναν Internet Browser. Επειδή όμως ο server που χρησιμοποιούμε υποστηρίζει μόνο μία ιστοσελίδα, η Request από τον Browser δεν λαμβάνεται καθόλου υπόψιν.

Ο Server ξεκινά μετά από τη σύνδεση ενός client στέλνοντας την ιστοσελίδα που είναι αποθηκευμένη στη μνήμη flash του MCU. Ο κώδικας της ιστοσελίδας βρίσκεται την C-constant WebSide[ ], που βρίσκεται στο αρχείο κώδικα webside.c [παράρτημα Α].

αφού ελεγχθεί η κατάσταση του buffer αποστολής ένας δείκτης στο webside, ενεργοποιείται και ο συνολικός αριθμός bytes για αποστολή αποθηκεύονται στη μεταβλητή HTTPBytesToSend.

Κατά το αρχικό κάλεσμα της HTTPServer( ) μία HTTP επικεφαλίδα μεταδίδεται απευθείας πριν την ιστοσελίδα. Αυτό εξηγεί στο client ότι η αίτησή του αποκωδικοποιήθηκε επιτυχημένα και του γνωστοποιείται τι είδους πηγαίος κώδικας θα μεταδοθεί (HTML).

Αποθηκεύεται στην σταθερή μεταβλητή GetResponse[] στο αρχείο επικεφαλίδας easyweb.h. μετά από αυτήν τη διαδικασία η ιστοσελίδα στέλνεται και η σύνδεση τερματίζεται καλώντας τη συνάρτηση TCPClose( ). Η σύνδεση ξαναγίνεται από τη συνάρτηση main( ), ώστε ο επόμενος client να ζητήσει και να του αποσταλεί η ιστοσελίδα.

```
void main(void)
{
  InitOsc();
  InitPorts();
  InitADC12();
  TCPLowLevelInit();
  __enable_interrupt();                    // enable interrupts
/*
  *(unsigned char *)RemoteIP = 24;         // uncomment those lines to get the
  *((unsigned char *)RemoteIP + 1) = 8;    // quote of the day from a real
  *((unsigned char *)RemoteIP + 2) = 69;   // internet server! (gateway must be
  *((unsigned char *)RemoteIP + 3) = 7;    // set to your LAN-router)
```

```
 TCPLocalPort = 2025;
 TCPRemotePort = TCP_PORT_QOTD;

 TCPActiveOpen();
 while (SocketStatus & SOCK_ACTIVE)           // read the quote from memory
 {                                            // by using the hardware-debugger
  DoNetworkStuff();  }
*/
 HTTPStatus = 0;                             // clear HTTP-server's flag register
 TCPLocalPort = TCP_PORT_HTTP;                // set port we want to listen to
  while (1)                                  // repeat forever
 {
  if (!(SocketStatus & SOCK_ACTIVE)) TCPPassiveOpen();   // listen for incoming TCP-connection
  DoNetworkStuff();                                      // handle network and easyWEB-stack
                                                         // events
  HTTPServer();  }}
```

Πριν την αποστολή ενός πακέτου δεδομένων TCP η συνάρτηση InsertDynamicValues( ) εκτελείται. Αυτή η συνάρτηση ψάχνει στο buffer μετάδοσης για ειδικά strings. Εάν ένα τέτοιο string βρεθεί αντικαθιστάτε με μία τιμή A/D μετατροπέα.

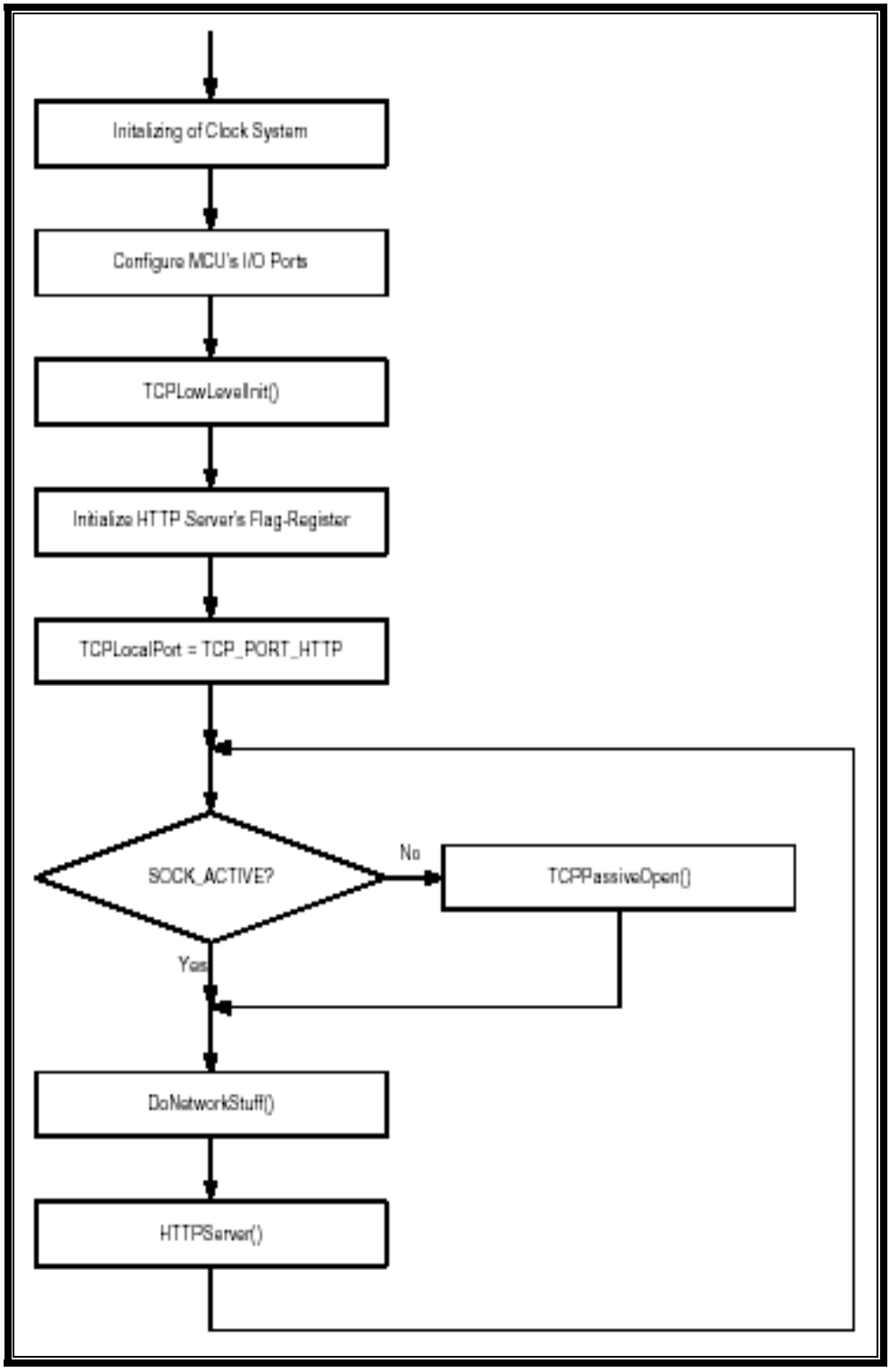
Ο HTTP Server αντικαθιστά το string AD7% με την τιμή του καναλιού 7 του A/D μετατροπέα και το ADA% με την τιμή του καναλιού 10.

```
 static void InsertDynamicValues(void)
{
 char *Key;
 char NewKey[5];
 int i;
  if (TCPTxDataCount < 4) return;              // there can't be any special string
  Key = (char *)TCP_TX_BUF;
  for (i = 0; i < TCPTxDataCount - 3; i++)
 {
  if (*Key == 'A')
   if (*(Key + 1) == 'D')
    if (*(Key + 3) == '%')
     if (*(Key + 2) == '7')                  // "AD7%"?
     {
      sprintf(NewKey, "%3u", GetAD7Val());  // εισαγωγή AD τιμή μετατροπής
      memcpy(Key, NewKey, 3);               // κανάλι 7 (P6.7)
     }
     else if (*(Key + 2) == 'A')
     {
      sprintf(NewKey, "%3u", GetTempVal()); // insert AD converter value
      memcpy(Key, NewKey, 3);               // κανάλι 10 (temp.-diode)
     }   Key++;  }}
```

Η τιμή για την αντικατάσταση του AD7% string γίνεται από τη συνάρτηση GetAD7Val( ). Το ADC12 μοντέλου του MSP430 έχει αρχικοποιηθεί έτσι ώστε να χρησιμοποιεί μία εσωτερική τιμή τάσης 2.5Volts. Αυτό υλοποιείται στο pin P6.7 του MSP430F149. η άλλη συνάρτηση που χρησιμοποιείται για A/D μετατροπές είναι η GetTempVal( ).

```
static unsigned int GetAD7Val(void)
{  ADC12MCTL0 = SREF_1 + INCH_7;              // Διάλεξε κανάλι  7, Vref+
   ADC12CTL0 |= ENC;                        // ξεκίνα μετατροπή
  ADC12CTL0 |= ADC12SC;                     // Δειγματοληψία &μετατροπή
   while (ADC12CTL0 & ADC12SC);             //  περίμενε μέχρι να τελείωσει η μετατροπή
  ADC12CTL0 &= ~ENC;                        // σταμάτα μετατροπή
  return ADC12MEM0 >> 5;  }
static unsigned int GetTempVal(void)
{  unsigned int i;
  unsigned int ADCResult;
  ADC12MCTL0 = SREF_1 + INCH_10;            // Διάλεξε κανάλι A10, Vref+
  ADC12CTL0 |= ENC;                        // ξεκίνα μετατροπή
  ADC12CTL0 |= ADC12SC;                     // Δειγματοληψία &μετατροπή
   while (ADC12CTL0 & ADC12SC);             // περίμενε μέχρι να τελείωσει η μετατροπή
   ADC12CTL0 &= ~ENC;                       //  σταμάτα μετατροπή
  ADCResult = ADC12MEM0;                    //  διάβασε την τιμή του ADC12
  for (i = 0; i < sizeof Temp_Tab; i++)     // Πάρε την θερμοκρασία από
   if (ADCResult < Temp_Tab[i])             // τον πίνακα
     break;
  return i << 1;                        // Scale value
}
```

Το κανάλι 10 είναι συνδεδεμένο εσωτερικά με μια δίοδο για την αναφορά της θερμοκρασίας. Βάζοντας την τάση αναφοράς στα 1.5 volts και κάνοντας μια πολλαπλή μετατροπή οχτώ σημείων δειγματοληψίας, μετριέται η θερμοκρασία του MCU. Χρησιμοποιώντας μια ειδική διεργασία η μέτρηση της θερμοκρασίας κυμαίνεται από τους 20 °C στους 45°C ως προς ένα ποσοστό από 0% εως 100%.

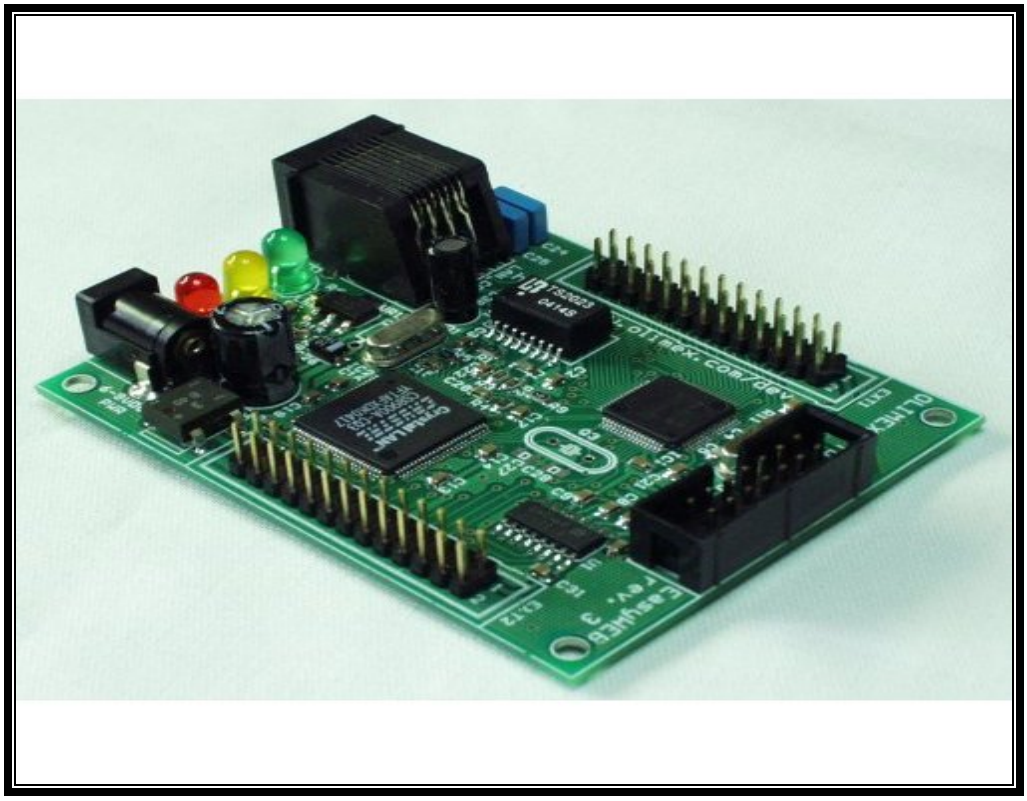


Διάγραμμα 5: Μοντέλο Web Server

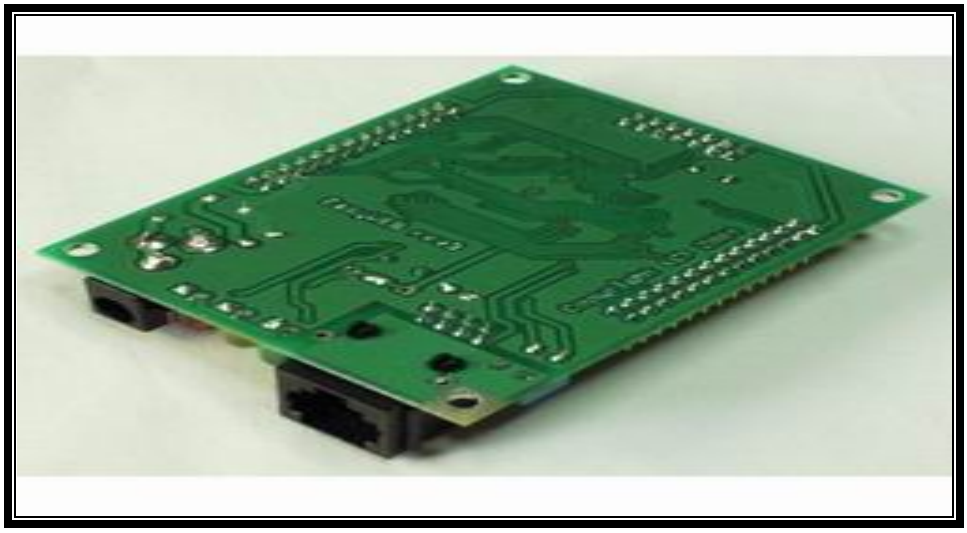
5.2 ΥΛΟΠΟΙΗΣΗ PROJECT ΕΦΑΡΜΟΓΗ HTTP SERVER

Το τελευταίο στάδιο ολοκλήρωσης του project είναι ο προγραμματισμός της πλακέτας που υλοποιείται ο HTTP SERVER. Λόγω ελλείψεων σε υλικά σημαντικών στην ολοκλήρωσης της πλακέτας θα αρκεστούμε στο να αναφέρουμε απλώς την διαδικασία του προγραμματισμού και την είσοδό της σε δίκτυο για να τρέξει η εφαρμογή του HTTP SERVER που θα περάσουμε στην Flash memory του MSP430.

Η πλακέτα εάν είχε υλοποιηθεί θα ήταν μια 2-layer πλακέτα που θα είχε την μορφή η οποία φαίνεται στην εικόνα 8 όπως την υλοποιεί η Spark Fun Electronics (www.sparkfun.com).



Εικόνα 8: Η υλοποιημένη πλακέτα του ETHERNET INTERFACE



Εικόνα 9: Κάτω όψη της πλακέτας

Αρχικά συνδέουμε την πλακέτα με τον υπολογιστή με ένα καλώδιο msp430-jtag (εικόνα 10), και με την πρίζα της ΔΕΗ με έναν 9V- μετασχηματιστή (adaptor) για την παροχή ρεύματος στα κυκλώματα (εικόνα 11). Στο τοπικό δίκτυο θα συνδεθεί με ένα καλώδιο UTP CAT-5 (RJ-45).

Εικόνα 10: Μετασχηματιστής για την παροχή ρεύματος

Εικόνα 11: msp430-jtag

Πριν περαστεί ο κώδικας στην πλακέτα μέσω του JTAG Interface πρέπει πρώτα να έχουμε φροντίσει να παραμετροποιήσουμε τον κώδικα δίνοντας του την διεύθυνση IP την διεύθυνση MAC την και την Subnet mask και την στάνταρ gateway την οποία την χρησιμοποιούμε σε περίπτωση που η IP διεύθυνση δεν είναι μέρος του υποδικτύου. Οι παραμετροποιήσεις αυτές γίνονται στα αρχεία επικεφαλίδας tcpip.h και cs8900.h για τα αρχεία tcpip.c και cs8900.c αντίστοιχα.

cs8900.h
header-file του cs8900.c

```
#define MYMAC_1         0       // ΜΟΝΑΔΙΚΗ MAC ΔΙΕΥΘΥΝΣΗ ΣΤΟ
#define MYMAC_2         1       // ΤΟΠΙΚΟ ΔΚΤΥΟ
#define MYMAC_3         2
#define MYMAC_4         3
#define MYMAC_5         4
#define MYMAC_6         5
```

tcpip.h
header-file του tcpip.c

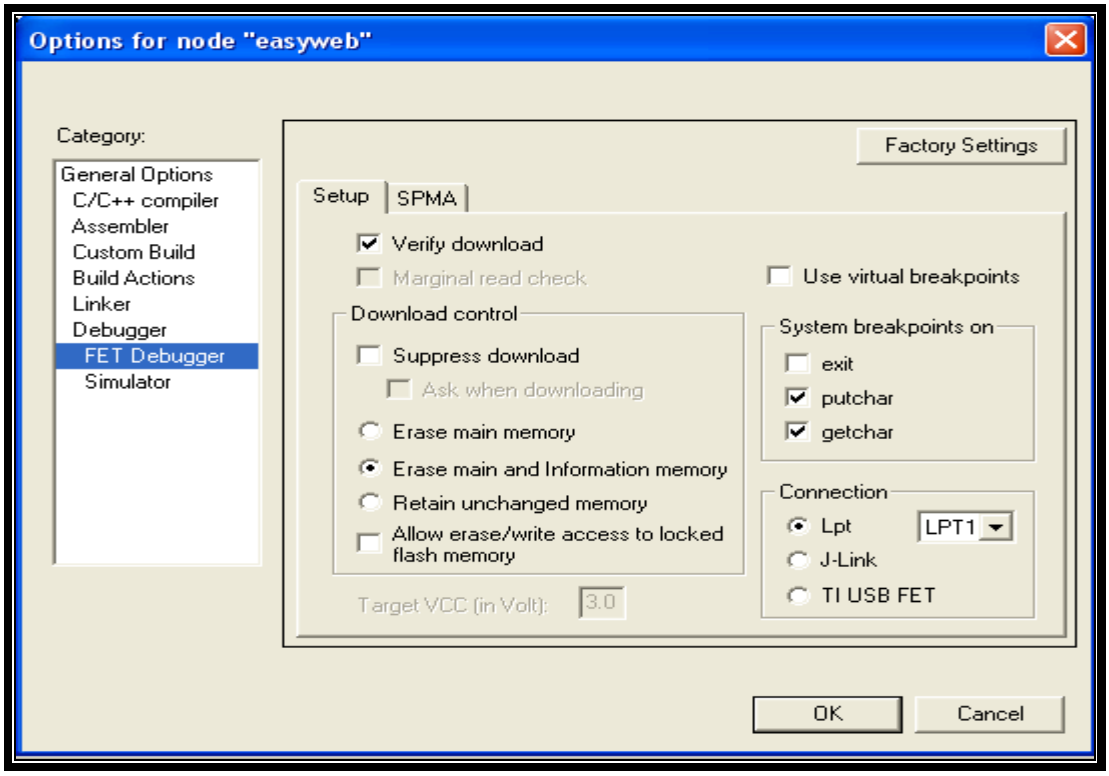
```
#define MYIP_1          192             // (IP) ΔΙΕΥΘΥΝΣΗ
#define MYIP_2          168
#define MYIP_3          0
#define MYIP_4          30

#define SUBMASK_1       255             // ΜΑΣΚΑ ΥΠΟΔΙΚΤΥΟΥ
#define SUBMASK_2       255
#define SUBMASK_3       255
#define SUBMASK_4       0

#define GWIP_1          192             // ΣΤΑΝΤΑΡ GATEWAY
#define GWIP_2          168             //
#define GWIP_3          0
#define GWIP_4          1
```

Αφού έχει γίνει η κατάλληλη συνδεσμολογία στο κύκλωμα το επόμενο βήμα είναι το 'κατέβασμα' (downloading) του κώδικα με την βοήθεια της σουίτας IAR Workbench.

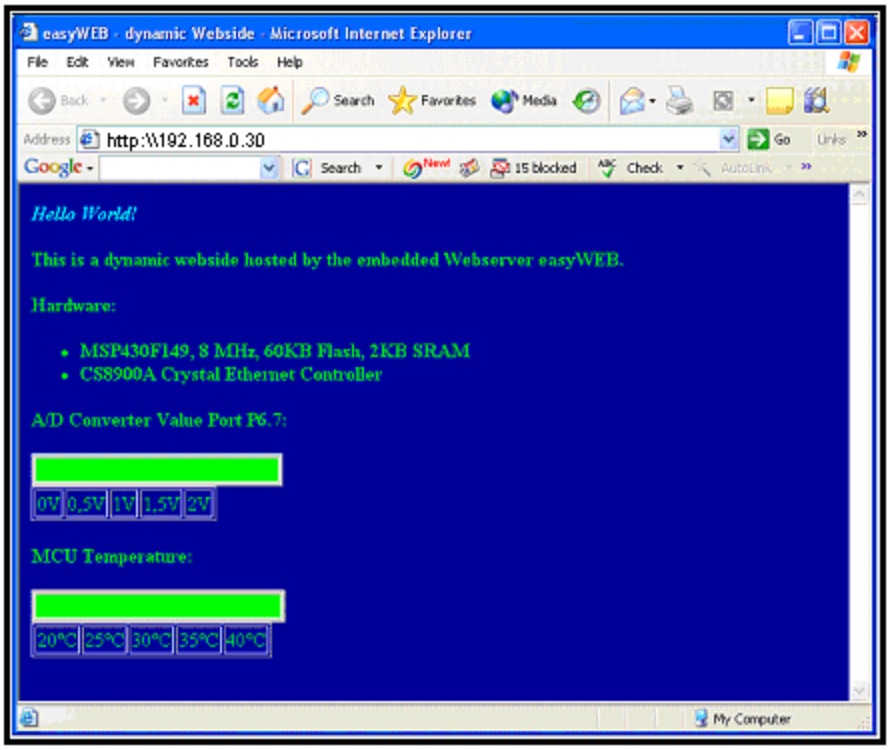
Προτού φτάσουμε στο στάδιο του Debug πρέπει να γίνουν ορισμένες ρυθμίσεις στα options του workspace. Ειδικότερα θα πρέπει να γίνουν κάποιες ρυθμίσεις στον FET Debugger (εικόνα 12) με ποιο βασική ρύθμιση την θύρα που θα χρησιμοποιηθεί για το κατέβασμα του προγράμματος. Αφού γίνουν οι ρυθμίσεις ο κώδικας είναι έτοιμος για Debug και κατέβασμα.



Εικόνα 12:Παραμετροποίηση του FET Debugger

Όταν θα είναι έτοιμη προγραμματισμένη και συνδεδεμένη η πλακέτα στο δίκτυο ο HTTP SERVER αρχικά περιμένει για μία εισερχόμενη σύνδεση. Πληκτρολογώντας στον Browser την IP διεύθυνση που του έχουμε δώσει σε έναν υπολογιστή client(η διεύθυνση που δίνεται εδώ είναι τυχαία) εγκαθιστεί σύνδεση με αυτόν και του στέλνει την ιστοσελίδα (εικόνα 13).

Κλείνοντας την ιστοσελίδα ο client τερματίζει την σύνδεση αυτή και περιμένει κάποιον άλλον client να συνδεθεί. Η ιστοσελίδα είναι δυναμική και ανανεώνεται κάθε πέντε δευτερόλεπτα δείχνοντάς μας την θερμοκρασία του επεξεργαστή και την τάση στο pin P6.7 την μέθοδο που έχει περιγραφεί παραπάνω.



Εικόνα 13: Αποθηκευμένη ιστοσελίδα

ΚΕΦΑΛΑΙΟ 6ο

6. ΜΕΛΛΟΝΤΙΚΗ ΒΕΛΤΙΩΣΗ ΣΥΣΤΗΜΑΤΟΣ

6.1 ΑΝΑΒΑΘΜΙΣΗ-ΤΡΟΠΟΠΟΙΗΣΗ ΚΥΚΛΩΜΑΤΟΣ ΠΛΑΚΕΤΑΣ

Οι τροποποιήσεις που θα μπορούν να υλοποιηθούν στο τυπωμένο κύκλωμα της πλακέτας αφορούν τον προγραμματισμό της και τη σύνδεσή της στο δίκτυο.

Αντί του RJ 45 Connector που χρησιμοποιούμε στην υλοποίηση της πλακέτας, θα μπορούσαμε να την τοποθετήσουμε σε ένα ασύρματο δίκτυο, σχεδιάζοντας και υλοποιώντας έναν πομποδέκτη Wi-Fi, συμβατό με το πρωτόκολλο επικοινωνίας 802.11b ή Bluetooth χωρίς όμως να μπορούμε να επιτύχουμε την τόσο μεγάλη απόσταση όπως στο Wi-Fi. Επιπλέον ο MSP430F149 μικροεπεξεργαστής, θα μπορούσε να αντικατασταθεί από έναν DSP chip.

6.2 ΤΡΟΠΟΠΟΙΗΣΗ ΛΟΓΙΣΜΙΚΟΥ

Στη συγκεκριμένη κατασκευή υλοποιήθηκε ένας HTTP SERVER που δημιουργούσε μία δυναμκή ιστοσελίδα. Εναλλακτικά, υπάρχει η δυνατότητα κρατώντας τον βασικό κορμό του κώδικα προγραμματισμού όσον αφορά την διεπαφή του MSP430F149 με τον CS8900A και την υλοποίηση του TCP/IP σωρού και πρωτοκόλλου, σχεδιάζοντας και προγραμματίζοντας μία άλλη δικτυακή εφαρμογή με στόχο την περαιτέρω χρηστική εκμετάλλευση της πλακέτας.

Για παράδειγμα, την υλοποίηση μίας ενσύρματης ή ασύρματης δικτυακής κάμερας με πολλές χρήσεις και όχι μόνο. Θα μπορούσε επίσης να χρησιμοποιηθεί για διαχείριση οποιασδήποτε ηλεκτρονικής online συσκευής.

Ο προγραμματισμός της επίσης θα μπορούσε να γίνει (‘περνώντας’ τον κώδικα) χρησιμοποιώντας την USB θύρα αντί του κλασσικού JTAG Interface. Εξ ορισμού η USB θύρα είναι πιο γρήγορη από την θύρα που χρησιμοποιείται σε αυτό το project.

ΠΑΡΑΡΤΗΜΑ Α΄

► ΒΑΣΙΚΟΣ ΚΩΔΙΚΑΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

Σε αυτό το κομμάτι παρατίθεται ο βασικός κώδικας για τον προγραμματισμό της πλακέτας για να χρησιμοποιηθεί στο Project: Σχεδιασμός & Κατασκευή ενός Ethernet Interface. Βασικός στόχος του κώδικα αυτού είναι η πραγματοποίηση της διεπαφής.

Ο κώδικας αυτός δημιουργήθηκε με την σουίτα IAR Embedded Workbench 4.

```
//-------------------------------------------------------------------------
// Όνομα: cs8900.c
// Func: ethernet driver για χρήση με τον LAN controller CS8900A
//-------------------------------------------------------------------------
#include "msp430x14x.h"
#include "support.h"
#include "cs8900.h"
//-------------------------------------------------------------------------
const unsigned int MyMAC[] =                    // "M1-M2-M3-M4-M5-M6"
{

  MYMAC_1 + (unsigned int)(MYMAC_2 << 8),
  MYMAC_3 + (unsigned int)(MYMAC_4 << 8),
  MYMAC_5 + (unsigned int)(MYMAC_6 << 8)
};

static const TInitSeq InitSeq[] =
{
  PP_IA, MYMAC_1 + (MYMAC_2 << 8),
  PP_IA + 2, MYMAC_3 + (MYMAC_4 << 8),
  PP_IA + 4, MYMAC_5 + (MYMAC_6 << 8),
  PP_LineCTL, SERIAL_RX_ON | SERIAL_TX_ON,      // Αρχικοποίηση του Φυσικού Επίπεδου (MAC
Διεύθυνση)
  PP_RxCTL, RX_OK_ACCEPT | RX_IA_ACCEPT | RX_BROADCAST_ACCEPT
};

//-------------------------------------------------------------------------
// αρχικοποίηση των port-pins για χρήση του LAN-controller,
//-------------------------------------------------------------------------
void Init8900(void)
{  unsigned int i;

  P3OUT = IOR | IOW;                  // επανεκκίνηση εξόδου, γραμμές ελέγχου high
  P3DIR = 0xff;                       // γραμμή ελέγχου γίνεται έξοδος
  P5OUT = 0;                          //  επανεκκίνηση εξόδου
  P5DIR = 0xff;                       // γραμμή ελέγχου γίνεται έξοδος

  DelayCycles(40000);                     // delay 10ms @ 8MHz MCLK
  DelayCycles(40000);                     // time for CS8900 POR

  Write8900(ADD_PORT, PP_SelfCTL);              // θέτουμε την register
  Write8900(DATA_PORT, POWER_ON_RESET);         // επανεκκίνηση του Ethernet-Controller
  do
  Write8900(ADD_PORT, PP_SelfST);                       // επανεκκίνηση της register
  while (!(Read8900(DATA_PORT) & INIT_DONE));    // αναμονή μέχρι επανεκκίνηση του chip

  for (i = 0; i < sizeof InitSeq / sizeof (TInitSeq); i++)
  {
    Write8900(ADD_PORT, InitSeq[i].Addr);
    Write8900(DATA_PORT, InitSeq[i].Data);
  }
}
```

```
//-------------------------------------------------------------------------
void Write8900(unsigned char Address, unsigned int Data)
{
  P5DIR = 0xff;                          // Θύρα 5 γίνεται έξοδος
  P3OUT = IOR | IOW | Address;           // θέτουμε διεύθυνση στο δίαυλο
  P5OUT = Data;
  P3OUT &= ~IOW;                         // toggle IOW-signal
  P3OUT = IOR | IOW | (Address + 1);     // και θέτουμε την επόμενη διεύθυνση στον δίαυλο
  P5OUT = Data >> 8;
  P3OUT &= ~IOW;                         // toggle IOW-signal
  P3OUT |= IOW;
}


//-------------------------------------------------------------------------
void WriteFrame8900(unsigned int Data)
{
  P5DIR = 0xff;                                  // Θύρα 5 γίνεται έξοδος
  P3OUT = IOR | IOW | TX_FRAME_PORT;             // θέτουμε διεύθυνση στο δίαυλο
  P5OUT = Data;
  P3OUT &= ~IOW;                                 // toggle IOW-signal
  P3OUT = IOR | IOW | (TX_FRAME_PORT + 1);       // και θέτουμε την επόμενη διεύθυνση στον δίαυλο
  P5OUT = Data >> 8;
  P3OUT &= ~IOW;                                 // toggle IOW-signal
  P3OUT |= IOW;
}


//-------------------------------------------------------------------------
// αντιγράφει bytes από την μνήμη του MCU στο frame port
//-------------------------------------------------------------------------
void CopyToFrame8900(void *Source, unsigned int Size)
{
  unsigned int *pSource = Source;

  P5DIR = 0xff;                                   // Θύρα 5 γίνεται έξοδος
  while (Size > 1)
  {
    P3OUT = IOR | IOW | TX_FRAME_PORT;            // θέτουμε διεύθυνση στο δίαυλο
    P5OUT = *pSource;
    P3OUT &= ~IOW;                                // toggle IOW-signal
    P3OUT = IOR | IOW | (TX_FRAME_PORT + 1);      // και θέτουμε την επόμενη διεύθυνση στον δίαυλο
    P5OUT = (*pSource++) >> 8;
    P3OUT &= ~IOW;                                // toggle IOW-signal
    P3OUT |= IOW;
    Size -= 2;
  }
    if (Size)
  {
    P3OUT = IOR | IOW | TX_FRAME_PORT;            // θέτουμε διεύθυνση στο δίαυλο
    P5OUT = *pSource;                             // write byte data bus
    P3OUT &= ~IOW;                                // toggle IOW-signal
    P3OUT |= IOW;  }}
```

```
//-------------------------------------------------------------------------
unsigned int Read8900(unsigned char Address)
{
  unsigned int ReturnValue;

  P5DIR = 0x00;                          // Θύρα 5 γίνεται είσοδος
  P3OUT = IOR | IOW | Address;           // θέτω διεύθυνση στο δίαυλο
  P3OUT &= ~IOR;                         // IOR-signal low
  ReturnValue = P5IN;
  P3OUT = IOR | IOW | (Address + 1);     // IOR high and και θέτω επόμενη διεύθυνση στο δίαυλο
  P3OUT &= ~IOR;                         // IOR-signal low
  ReturnValue |= P5IN << 8;
  P3OUT |= IOR;
  P5DIR = 0xff;                          // θύρα 5 έξοδος
    return ReturnValue;
}


//-------------------------------------------------------------------------
unsigned int ReadFrame8900(void)
{
  unsigned int ReturnValue;

  P5DIR = 0x00;                                  //  Θύρα 5 γίνεται είσοδος
  P3OUT = IOR | IOW | RX_FRAME_PORT;             // πρόσβαση στην RX_FRAME_PORT
  P3OUT &= ~IOR;                                 // IOR-signal low
  ReturnValue = P5IN;                            // λαμβάνω το 1ο byte από τον data bus (low-byte)
  P3OUT = IOR | IOW | (RX_FRAME_PORT + 1);       // IOR high και θέτω επόμενη διεύθυνση στον δίαυλο
  P3OUT &= ~IOR;                                 // IOR-signal low
  ReturnValue |= P5IN << 8;                      // λαμβάνω 2ο byte από τον data bus (high-byte)
  P3OUT |= IOR;
  P5DIR = 0xff;                                  // θύρα 5 έξοδος
    return ReturnValue;
}


//-------------------------------------------------------------------------
unsigned int ReadFrameBE8900(void)
{
  unsigned int ReturnValue;

  P5DIR = 0x00;                                  // Θύρα 5 γίνεται είσοδος
  P3OUT = IOR | IOW | RX_FRAME_PORT;             // πρόσβαση στην RX_FRAME_PORT
  P3OUT &= ~IOR;                                 // IOR-signal low
  ReturnValue = P5IN << 8;                       // λαμβάνω το 1ο byte από τον data bus (low-byte)
  P3OUT = IOR | IOW | (RX_FRAME_PORT + 1);       // IOR high και θέτω επόμενη διεύθυνση στον
δίαυλο
  P3OUT &= ~IOR;                                 // IOR-signal low
  ReturnValue |= P5IN;                           // λαμβάνω 2ο byte από τον data (low-byte)
  P3OUT |= IOR;
  P5DIR = 0xff;                                  // θύρα 5 έξοδος
    return ReturnValue;
}
```

```
//-------------------------------------------------------------------------
unsigned int ReadHB1ST8900(unsigned char Address)
{
 unsigned int ReturnValue;

 P5DIR = 0x00;                          // Θύρα 5 γίνεται είσοδος
 P3OUT = IOR | IOW | (Address + 1);     // θέτω διεύθυνση στον δίαυλο
 P3OUT &= ~IOR;                         // IOR-signal low
 ReturnValue = P5IN << 8;
 P3OUT = IOR | IOW | Address;           // IOR high και θέτω την επόμενη διεύθυνση στον bus
 P3OUT &= ~IOR;                         // IOR-signal low
 ReturnValue |= P5IN;
 P3OUT |= IOR;
 P5DIR = 0xff;                          // θύρα 5 έξοδος

 return ReturnValue;
}


//-------------------------------------------------------------------------
void CopyFromFrame8900(void *Dest, unsigned int Size)
{
 unsigned int *pDest = Dest;

 P5DIR = 0x00;                                  // Θύρα 5 γίνεται είσοδος
 while (Size > 1)
 {
  P3OUT = IOR | IOW | RX_FRAME_PORT;            // πρόσβαση στην RX_FRAME_PORT
  P3OUT &= ~IOR;                                // IOR-signal low
  *pDest = P5IN;                                // λαμβάνω το 1ο byte από τον data bus (low-byte)
  P3OUT = IOR | IOW | (RX_FRAME_PORT + 1);      // IOR high και θέτω επόμενη διεύθυνση στον bus
  P3OUT &= ~IOR;                                // IOR-signal low
  *pDest++ |= P5IN << 8;                        // λαμβάνω το 2ο byte από τον data bus (high-byte)
  P3OUT |= IOR;
  Size -= 2;  }
  if (Size)
 {
  P3OUT = IOR | IOW | RX_FRAME_PORT;            // access to RX_FRAME_PORT
  P3OUT &= ~IOR;                                // IOR-signal low
  *(unsigned char *)pDest = P5IN;               // λαμβάνω byte from data bus
  P3OUT |= IOR;                                 // IOR high
 }

 P5DIR = 0xff;                                  // θύρα 5 έξοδος
}
```

```
//---------------------------------------------------------------------------
void DummyReadFrame8900(unsigned int Size)
{
  P5DIR = 0x00;                                    // θύρα 5 είσοδος

  while (Size--)
  {
    P3OUT = IOR | IOW | RX_FRAME_PORT;          // πρόσβαση στον RX_FRAME_PORT
    P3OUT &= ~IOR;                               // IOR-signal low
  }

  P3OUT |= IOR;                                    // IOR high
  P5DIR = 0xff;                                   // θύρα 5 έξοδος
}


//---------------------------------------------------------------------------
void RequestSend(unsigned int FrameSize)
{
  Write8900(TX_CMD_PORT, TX_START_ALL_BYTES);
  Write8900(TX_LEN_PORT, FrameSize);
}
//---------------------------------------------------------------------------
// έλεγχος εάν ο CS8900A είναι έτοιμος να στείλει το πακέτο που θέλουμε
//---------------------------------------------------------------------------
unsigned int Rdy4Tx(void)
{
  Write8900(ADD_PORT, PP_BusST);
  return Read8900(DATA_PORT) & READY_FOR_TX_NOW;
}


//---------------------------------------------------------------------------
// Name: cs8900.h
// Func: header-file for cs8900.c
//---------------------------------------------------------------------------

#ifndef __CS8900_H
#define __CS8900_H

#define MYMAC_1          0              // ethernet (MAC) διεύθυνση
#define MYMAC_2          1              // (πρέπει να είναι μοναδική στο LAN!)
#define MYMAC_3          2
#define MYMAC_4          3
#define MYMAC_5          4
#define MYMAC_6          5

#define IOR              (0x40)         // CS8900's ISA-bus interface pins
#define IOW              (0x80)         // διευκρινήσεις για τον Crystal CS8900 ethernet-controller
```

```
#define PP_ChipID          (0x0000)        // offset 0h -> Corp-ID
                                           // offset 2h -> Model/Product Number
                                           // offset 3h -> Chip Revision Number

#define PP_ISAIOB          (0x0020)        // IO base address
#define PP_CS8900_ISAINT   (0x0022)        // ISA interrupt επιλογή
#define PP_CS8900_ISADMA   (0x0024)        // ISA Rec DMA channel
#define PP_ISASOF          (0x0026)        // ISA DMA offset
#define PP_DmaFrameCnt     (0x0028)        // ISA DMA Frame count
#define PP_DmaByteCnt      (0x002a)        // ISA DMA Byte count
#define PP_CS8900_ISAMemB  (0x002c)        // Memory base
#define PP_ISABootBase     (0x0030)        // Boot Prom base
#define PP_ISABootMask     (0x0034)        // Boot Prom Mask

// EEPROM data and command registers
#define PP_EECMD           (0x0040)        // NVR Interface Command register
#define PP_EEData          (0x0042)        // NVR Interface Data Register

// Configuration and control registers
#define PP_RxCFG           (0x0102)        // Rx Bus config
#define PP_RxCTL           (0x0104)        // Receive Control Register
#define PP_TxCFG           (0x0106)        // Transmit Config Register
#define PP_TxCMD           (0x0108)        // Transmit Command Register
#define PP_BufCFG          (0x010a)        // Bus configuration Register
#define PP_LineCTL         (0x0112)        // Line Config Register
#define PP_SelfCTL         (0x0114)        // Self Command Register
#define PP_BusCTL          (0x0116)        // ISA bus control Register
#define PP_TestCTL         (0x0118)        // Test Register

// Status and Event Registers
#define PP_ISQ             (0x0120)        // Interrupt Status
#define PP_RxEvent         (0x0124)        // Rx Event Register
#define PP_TxEvent         (0x0128)        // Tx Event Register
#define PP_BufEvent        (0x012c)        // Bus Event Register
#define PP_RxMiss          (0x0130)        // Receive Miss Count
#define PP_TxCol           (0x0132)        // Transmit Collision Count
#define PP_LineST          (0x0134)        // Line State Register
#define PP_SelfST          (0x0136)        // Self State register
#define PP_BusST           (0x0138)        // Bus Status
#define PP_TDR             (0x013c)        // Time Domain Reflectometry

// Initiate Transmit Registers
#define PP_TxCommand       (0x0144)        // Tx Command
#define PP_TxLength        (0x0146)        // Tx Length

// Adress Filter Registers
#define PP_LAF             (0x0150)        // Hash Table
#define PP_IA              (0x0158)        // Physical Address Register

// Frame Location
#define PP_RxStatus        (0x0400)        // Δέξου αρχή του frame
#define PP_RxLength        (0x0402)        // Δέξου μήκος frame
#define PP_RxFrame         (0x0404)        // Αποδοχή frame pointer
```

```
#define PP_TxFrame          (0x0a00)                    // Εκπομπή frame pointer

#define DEFAULTIOBASE       (0x0300

// PP_RxCFG - Receive  Configuration and Interrupt Mask bit definition - Read/write
#define SKIP_1              (0x0040)
#define RX_STREAM_ENBL      (0x0080)
#define RX_OK_ENBL          (0x0100)
#define RX_DMA_ONLY         (0x0200)
#define AUTO_RX_DMA         (0x0400)
#define BUFFER_CRC          (0x0800)
#define RX_CRC_ERROR_ENBL   (0x1000)
#define RX_RUNT_ENBL        (0x2000)
#define RX_EXTRA_DATA_ENBL  (0x4000)

// PP_RxCTL - Receive Control bit definition - Read/write
#define RX_IA_HASH_ACCEPT   (0x0040)
#define RX_PROM_ACCEPT      (0x0080)
#define RX_OK_ACCEPT        (0x0100)
#define RX_MULTCAST_ACCEPT  (0x0200)
#define RX_IA_ACCEPT        (0x0400)
#define RX_BROADCAST_ACCEPT (0x0800)
#define RX_BAD_CRC_ACCEPT   (0x1000)
#define RX_RUNT_ACCEPT      (0x2000)
#define RX_EXTRA_DATA_ACCEPT (0x4000)

// PP_TxCFG - Transmit Configuration Interrupt Mask bit definition - Read/write
#define TX_LOST_CRS_ENBL    (0x0040)
#define TX_SQE_ERROR_ENBL   (0x0080)
#define TX_OK_ENBL          (0x0100)
#define TX_LATE_COL_ENBL    (0x0200)
#define TX_JBR_ENBL         (0x0400)
#define TX_ANY_COL_ENBL     (0x0800)
#define TX_16_COL_ENBL      (0x8000)

// PP_TxCMD - Transmit Command bit definition - Read-only and
// PP_TxCommand - Write-only
#define TX_START_5_BYTES    (0x0000)
#define TX_START_381_BYTES  (0x0040)
#define TX_START_1021_BYTES (0x0080)
#define TX_START_ALL_BYTES  (0x00C0)
#define TX_FORCE            (0x0100)
#define TX_ONE_COL          (0x0200)
#define TX_NO_CRC           (0x1000)
#define TX_RUNT             (0x2000)

// PP_BufCFG - Buffer Configuration Interrupt Mask bit definition - Read/write
#define GENERATE_SW_INTERRUPT     (0x0040)
#define RX_DMA_ENBL               (0x0080)
#define READY_FOR_TX_ENBL         (0x0100)
#define TX_UNDERRUN_ENBL          (0x0200)
#define RX_MISS_ENBL              (0x0400)
#define RX_128_BYTE_ENBL          (0x0800)
```

```
#define TX_COL_COUNT_OVRFLOW_ENBL (0x1000)
#define RX_MISS_COUNT_OVRFLOW_ENBL (0x2000)
#define RX_DEST_MATCH_ENBL       (0x8000)

// PP_LineCTL - Line Control bit definition - Read/write
#define SERIAL_RX_ON       (0x0040)
#define SERIAL_TX_ON       (0x0080)
#define AUI_ONLY           (0x0100)
#define AUTO_AUI_10BASET   (0x0200)
#define MODIFIED_BACKOFF   (0x0800)
#define NO_AUTO_POLARITY   (0x1000)
#define TWO_PART_DEFDIS    (0x2000)
#define LOW_RX_SQUELCH     (0x4000)

// PP_SelfCTL - Software Self Control bit definition - Read/write
#define POWER_ON_RESET     (0x0040)
#define SW_STOP            (0x0100)
#define SLEEP_ON           (0x0200)
#define AUTO_WAKEUP        (0x0400)
#define HCB0_ENBL          (0x1000)
#define HCB1_ENBL          (0x2000)
#define HCB0               (0x4000)
#define HCB1               (0x8000)

// PP_BusCTL - ISA Bus Control bit definition - Read/write
#define RESET_RX_DMA       (0x0040)
#define MEMORY_ON          (0x0400)
#define DMA_BURST_MODE     (0x0800)
#define IO_CHANNEL_READY_ON (0x1000)
#define RX_DMA_SIZE_64K    (0x2000)
#define ENABLE_IRQ         (0x8000)

// PP_TestCTL - Test Control bit definition - Read/write
#define LINK_OFF           (0x0080)
#define ENDEC_LOOPBACK     (0x0200)
#define AUI_LOOPBACK       (0x0400)
#define BACKOFF_OFF        (0x0800)
#define FDX_8900           (0x4000)

// PP_RxEvent - Receive Event Bit definition - Read-only
#define RX_IA_HASHED       (0x0040)
#define RX_DRIBBLE         (0x0080)
#define RX_OK              (0x0100)0
#define RX_HASHED          (0x0200)
#define RX_IA              (0x0400)
#define RX_BROADCAST       (0x0800)
#define RX_CRC_ERROR       (0x1000)
#define RX_RUNT            (0x2000)
#define RX_EXTRA_DATA      (0x4000)
#define HASH_INDEX_MASK    (0xFC00)                    // Hash-Table Index Mask (6 Bit)
```

```
// PP_TxEvent - Transmit Event Bit definition - Read-only
#define TX_LOST_CRS        (0x0040)
#define TX_SQE_ERROR       (0x0080)
#define TX_OK              (0x0100)
#define TX_LATE_COL        (0x0200)
#define TX_JBR             (0x0400)
#define TX_16_COL          (0x8000)
#define TX_COL_COUNT_MASK  (0x7800)

// PP_BufEvent - Buffer Event Bit definition - Read-only
#define SW_INTERRUPT       (0x0040)
#define RX_DMA             (0x0080)
#define READY_FOR_TX       (0x0100)
#define TX_UNDERRUN        (0x0200)
#define RX_MISS            (0x0400)
#define RX_128_BYTE        (0x0800)
#define TX_COL_OVRFLW      (0x1000)
#define RX_MISS_OVRFLW     (0x2000)
#define RX_DEST_MATCH      (0x8000)

// PP_LineST - Ethernet Line Status bit definition - Read-only
#define LINK_OK            (0x0080)
#define AUI_ON             (0x0100)
#define TENBASET_ON        (0x0200)
#define POLARITY_OK        (0x1000)
#define CRS_OK             (0x4000)
// PP_SelfST - Chip Software Status bit definition
#define ACTIVE_33V         (0x0040)
#define INIT_DONE          (0x0080)
#define SI_BUSY            (0x0100)
#define EEPROM_PRESENT     (0x0200)
#define EEPROM_OK          (0x0400)
#define EL_PRESENT         (0x0800)
#define EE_SIZE_64         (0x1000)

// PP_BusST - ISA Bus Status bit definition
#define TX_BID_ERROR       (0x0080)
#define READY_FOR_TX_NOW   (0x0100)

// The following block defines the ISQ event types
#define ISQ_RX_EVENT       (0x0004)
#define ISQ_TX_EVENT       (0x0008)
#define ISQ_BUFFER_EVENT   (0x000c)
#define ISQ_RX_MISS_EVENT  (0x0010)
#define ISQ_TX_COL_EVENT   (0x0012)

#define ISQ_EVENT_MASK     (0x003f)          // ISQ mask to find out type of event

// Ports for I/O-Mode
#define RX_FRAME_PORT      (0x0000)
#define TX_FRAME_PORT      (0x0000)
#define TX_CMD_PORT        (0x0004)
#define TX_LEN_PORT        (0x0006)
```

```
#define ISQ_PORT          (0x0008)
#define ADD_PORT          (0x000a)
#define DATA_PORT         (0x000c)

#define AUTOINCREMENT     (0x8000)                  // Bit mask to set Bit-15 for autoincrement

// EEProm Commands
#define EEPROM_WRITE_EN   (0x00f0)
#define EEPROM_WRITE_DIS  (0x0000)
#define EEPROM_WRITE_CMD  (0x0100)
#define EEPROM_READ_CMD   (0x0200)

// Receive Header of each packet in receive area of memory for DMA-Mode
#define RBUF_EVENT_LOW    (0x0000)                  // Low byte of RxEvent
#define RBUF_EVENT_HIGH   (0x0001)                  // High byte of RxEvent
#define RBUF_LEN_LOW      (0x0002)                  // Length of received data - low byte
#define RBUF_LEN_HI       (0x0003)                  // Length of received data - high byte
#define RBUF_HEAD_LEN     (0x0004)                  // Length of this header

// typedefs
typedef struct                                      // struct to store CS8900's
{
  unsigned int Addr;                                // init-sequence
  unsigned int Data;
} TInitSeq;

// exported constants
extern const unsigned int MyMAC[];                  // "M1-M2-M3-M4-M5-M6"

// exported functions
void Init8900(void);
void Write8900(unsigned char Address, unsigned int Data);
void WriteFrame8900(unsigned int Data);
unsigned int Read8900(unsigned char Address);
unsigned int ReadFrame8900(void);
unsigned int ReadHB1ST8900(unsigned char Address);
unsigned int ReadFrameBE8900(void);
void CopyToFrame8900(void *Source, unsigned int Size);
void CopyFromFrame8900(void *Dest, unsigned int Size);
void DummyReadFrame8900(unsigned int Size);
void RequestSend(unsigned int FrameSize);
unsigned int Rdy4Tx(void);

#endif
```

```
//-------------------------------------------------------------------------
// Name: easyweb.c
//-------------------------------------------------------------------------
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

#include "msp430x14x.h"
#include "support.h"
#include "easyweb.h"
#include "tcpip.h"                              // easyWEB TCP/IP stack

#include "webside.c"                            // webside for our HTTP server (HTML)

static const unsigned char GetResponse[] =      // το πρώτο που στέλνει ο server στο χρήστη
{
  "HTTP/1.0 200 OK\r\n"                         // protocol ver 1.0, code 200, reason OK
  "Content-Type: text/html\r\n"                 // τύπος δεδομένων που θέλουμε να στείλουμε
  "\r\n"                                        // δείχνει το τέλος του HTTP-header
};

static unsigned char *PWebSide;                 // pointer to webside
static unsigned int HTTPBytesToSend;            // bytes που απόμειναν για αποστολή
static unsigned char HTTPStatus;                // status byte

//-------------------------------------------------------------------------
static const unsigned int Temp_Tab[] =
{
  0x064F,                           // 0C
  0x0655,
  0x065B,
  0x0660,
  0x0666,
  0x066C,                           // 5C
  0x0672,
  0x0678,
  0x067D,
  0x0683,
  0x0689,                           // 10C
  0x068F,
  0x0695,
  0x069B,
  0x06A0,
  0x06A6,                           // 15C
  0x06AC,
  0x06B2,
  0x06B8,
  0x06BD,
  0x06C3,                           // 20C
  0x06C9,
  0x06CF,
  0x06D5,
```

```
  0x06DB,
  0x06E0,                          // 25C
  0x06E6,
  0x06EC,
  0x06F2,
  0x06F8,
  0x06FD,                          // 30C
  0x0703,
  0x0709,
  0x070F,
  0x0715,
  0x071B,                          // 35C
  0x0720,
  0x0726,
  0x072C,
  0x0732,
  0x0738,                          // 40C
  0x073D,
  0x0743,
  0x0749,
  0x074F,
  0x0755,                          // 45C
  0x0FFF
};

//-------------------------------------------------------------------------
// Local function prototypes
//-------------------------------------------------------------------------
static void InitOsc(void);
static void InitPorts(void);
static void InitADC12(void);
static void HTTPServer(void);
static void InsertDynamicValues(void);
static unsigned int GetAD7Val(void);
static unsigned int GetTempVal(void);
//-------------------------------------------------------------------------
void main(void)
{
  InitOsc();
  InitPorts();
  InitADC12();

  TCPLowLevelInit();

  __enable_interrupt();                    // enable interrupts

/*
  *(unsigned char *)RemoteIP = 24;
  *((unsigned char *)RemoteIP + 1) = 8;
  *((unsigned char *)RemoteIP + 2) = 69;
  *((unsigned char *)RemoteIP + 3) = 7;

  TCPLocalPort = 2025;
```

```
  TCPRemotePort = TCP_PORT_QOTD;

  TCPActiveOpen();

  while (SocketStatus & SOCK_ACTIVE)
  {
   DoNetworkStuff();
  }
*/

  HTTPStatus = 0;

  TCPLocalPort = TCP_PORT_HTTP;

  while (1)
  {
   if (!(SocketStatus & SOCK_ACTIVE)) TCPPassiveOpen();
   DoNetworkStuff();

   HTTPServer();
  }
}

//------------------------------------------------------------------------------
static void HTTPServer(void)
{
  if (SocketStatus & SOCK_CONNECTED)            // έλεγχος αν κάποιος συνδέθηκε στο δικό μας TCP/IP
  {
   if (SocketStatus & SOCK_DATA_AVAILABLE)      // έλεγχος αν το TCP στέλνει δεδομένα
     TCPReleaseRxBuffer();

   if (SocketStatus & SOCK_TX_BUF_RELEASED)
   {
     if (!(HTTPStatus & HTTP_SEND_PAGE))
     {
       HTTPBytesToSend = sizeof(WebSide) - 1;
       PWebSide = (unsigned char *)WebSide;
     }

     if (HTTPBytesToSend > MAX_TCP_TX_DATA_SIZE)
     {
       if (!(HTTPStatus & HTTP_SEND_PAGE))
       {
         memcpy(TCP_TX_BUF, GetResponse, sizeof(GetResponse) - 1);
         memcpy(TCP_TX_BUF + sizeof(GetResponse) - 1, PWebSide,
           MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1);
         HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
         PWebSide += MAX_TCP_TX_DATA_SIZE - sizeof(GetResponse) + 1;
       }
       else
       {
         memcpy(TCP_TX_BUF, PWebSide, MAX_TCP_TX_DATA_SIZE);
         HTTPBytesToSend -= MAX_TCP_TX_DATA_SIZE;
```

```
        PWebSide += MAX_TCP_TX_DATA_SIZE;
       }

       TCPTxDataCount = MAX_TCP_TX_DATA_SIZE;   // bytes to xfer
       InsertDynamicValues();
       TCPTransmitTxBuffer();
      }
      else if (HTTPBytesToSend)
      {
       memcpy(TCP_TX_BUF, PWebSide, HTTPBytesToSend);
       TCPTxDataCount = HTTPBytesToSend;
       InsertDynamicValues();
       TCPTransmitTxBuffer();
       TCPClose();
       HTTPBytesToSend = 0;
      }

      HTTPStatus |= HTTP_SEND_PAGE;
    }
  }
  else
    HTTPStatus &= ~HTTP_SEND_PAGE;              // επανεκκίνηση της help-flag εάν δεν υπάρχει
συνδέθηκε
}
//-------------------------------------------------------------------------
// samples and returns the AD-converter value of channel 7
// (associated with Port P6.7)
//-------------------------------------------------------------------------
static unsigned int GetAD7Val(void)
{
  ADC12MCTL0 = SREF_1 + INCH_7;

  ADC12CTL0 |= ENC;
  ADC12CTL0 |= ADC12SC;
  while (ADC12CTL0 & ADC12SC);

  ADC12CTL0 &= ~ENC;

  return ADC12MEM0 >> 5;
}
```

```
//-------------------------------------------------------------------------
static unsigned int GetTempVal(void)
{
  unsigned int i;
  unsigned int ADCResult;

  ADC12MCTL0 = SREF_1 + INCH_10;              // επιλογή καναλιού A10, Vref+

  ADC12CTL0 |= ENC;
  ADC12CTL0 |= ADC12SC;

  while (ADC12CTL0 & ADC12SC);

  ADC12CTL0 &= ~ENC;

  ADCResult = ADC12MEM0;

  for (i = 0; i < sizeof Temp_Tab; i++)
    if (ADCResult < Temp_Tab[i])
      break;

  return i << 1;
}

//-------------------------------------------------------------------------
static void InsertDynamicValues(void)
{
  char *Key;
  char NewKey[5];
  int i;

  if (TCPTxDataCount < 4) return;

  Key = (char *)TCP_TX_BUF;

  for (i = 0; i < TCPTxDataCount - 3; i++)
  {
    if (*Key == 'A')
      if (*(Key + 1) == 'D')
        if (*(Key + 3) == '%')
          if (*(Key + 2) == '7')
          {
            sprintf(NewKey, "%3u", GetAD7Val());
            memcpy(Key, NewKey, 3);                  // channel 7 (P6.7)
          }
          else if (*(Key + 2) == 'A')
          {
            sprintf(NewKey, "%3u", GetTempVal());
            memcpy(Key, NewKey, 3);                  // channel 10 (temp.-diode)
          }
      Key++;
  }
}
```

```
//-------------------------------------------------------------------------
// ενεργοποιεί τον 8MHz crystal στην XT1 και την χρησιμοποιεί σαν MCLK
//-------------------------------------------------------------------------
static void InitOsc(void)
{
  WDTCTL = WDTPW + WDTHOLD;

  BCSCTL1 |= XTS;
  __bic_SR_register(OSCOFF);                    // ενεργοποίηση του XT1 oscillator

  do
  {
    IFG1 &= ~OFIFG;                             // Clear OFIFG
    DelayCycles(100);
  } while (IFG1 & OFIFG);                       // Test oscillator fault flag

  BCSCTL2 = SELM_3;                             // θέτει την XT1 σαν MCLK
}
//-------------------------------------------------------------------------
static void InitPorts(void)
{
  P1OUT = 0;                                    // ενεργοποίεί όλες τις αχρησιμοποίητες θύρες σαν
εξόδους
  P1DIR = 0xff;

  P2OUT = 0;
  P2DIR = 0xff;

  P4OUT = 0;
  P4DIR = 0xff;

  P6SEL = 0x80;                                 // χρήση της P6.7 σαν ADC module
  P6OUT = 0;
  P6DIR = 0x7f;                                 // όλες έξοδοι εκτός της P6.7
}
//-------------------------------------------------------------------------
static void InitADC12(void)
{
  ADC12CTL0 = ADC12ON + REFON + REF2_5V + SHT0_6
  ADC12CTL1 = SHS_0 + SHP + CONSEQ_0;
 }
```

```
//-------------------------------------------------------------------------
// Name: easyweb.c
// Func: header-file for easyweb.c
//-------------------------------------------------------------------------
#ifndef __EASYWEB_H
#define __EASYWEB_H
// definitions for 'HTTPStatus'
#define HTTP_SEND_PAGE          (0x01)     // help flag
#endif


//-------------------------------------------------------------------------
// Name: tcpip.c
// Func: implements the TCP/IP-stack and provides an API
//-------------------------------------------------------------------------

#include "msp430x14x.h"
#include "support.h"
#include "cs8900.h"
#include "tcpip.h"

// constants
const unsigned int MyIP[] =                         // "MYIP1.MYIP2.MYIP3.MYIP4"
{
  MYIP_1 + (unsigned int)(MYIP_2 << 8),
  MYIP_3 + (unsigned int)(MYIP_4 << 8)
};

const unsigned int SubnetMask[] =                   //
"SUBMASK1.SUBMASK2.SUBMASK3.SUBMASK4"
{
  SUBMASK_1 + (unsigned int)(SUBMASK_2 << 8),
  SUBMASK_3 + (unsigned int)(SUBMASK_4 << 8)
};

const unsigned int GatewayIP[] =                    // "GWIP1.GWIP2.GWIP3.GWIP4"
{
  GWIP_1 + (unsigned int)(GWIP_2 << 8),
  GWIP_3 + (unsigned int)(GWIP_4 << 8)
};

// variables
static TTCPStateMachine TCPStateMachine;
static TLastFrameSent LastFrameSent;

static unsigned int ISNGenHigh;
static unsigned long TCPSeqNr;
static unsigned long TCPUNASeqNr;

static unsigned long TCPAckNr;
```

```
static unsigned char TCPTimer;
static unsigned char RetryCounter;
static unsigned int TxFrame1Size;                    // bytes για αποστολή στο TxFrame1
static unsigned char TxFrame2Size;                   // bytes για αποστολή στο TxFrame2
static unsigned char TransmitControl;
static unsigned char TCPFlags;
unsigned int TCPRxDataCount;
unsigned int TCPTxDataCount;
unsigned int TCPLocalPort;                           // TCP ports
unsigned int TCPRemotePort;
unsigned int RemoteMAC[3];
unsigned int RemoteIP[2];
unsigned char SocketStatus;
// στοιχεία του νεοφερθέντος πακέτου
static unsigned int RecdFrameLength;
static unsigned int RecdFrameMAC[3];          // 48 bit MAC
static unsigned int RecdFrameIP[2];           // 32 bit IP
static unsigned int RecdIPFrameLength;        // 16 bit IP packet length

unsigned int TxFrame1Mem[(ETH_HEADER_SIZE + IP_HEADER_SIZE + TCP_HEADER_SIZE +
                  MAX_TCP_TX_DATA_SIZE + 1) >> 1];
static unsigned int TxFrame2Mem[(ETH_HEADER_SIZE + MAX_ETH_TX_DATA_SIZE + 1) >> 1];
unsigned int RxTCPBufferMem[(MAX_TCP_RX_DATA_SIZE + 1) >> 1];
//---------------------------------------------------------------------------
static void ProcessEthBroadcastFrame(void);
static void ProcessEthIAFrame(void);
static void ProcessICMPFrame(void);
static void ProcessTCPFrame(void);

// fill TX-buffers
static void PrepareARP_REQUEST(void);
static void PrepareARP_ANSWER(void);
static void PrepareICMP_ECHO_REPLY(void);

static void PrepareTCP_FRAME(unsigned long seqnr, unsigned long acknr,
  unsigned int TCPCode);
static void PrepareTCP_DATA_FRAME(void);

// general help functions
static void TCPStartRetryTimer(void);
static void TCPStartFinTimer(void);
static void TCPRestartTimer(void);
static void TCPStopTimer(void);
static void TCPHandleRetransmission(void);
static void TCPHandleTimeout(void);
static unsigned int CalcChecksum(void *Start, unsigned int Count,
  unsigned char IsTCP);
```

```
//-------------------------------------------------------------------------
// easyWEB-API function
// αρχικοποιεί τους LAN-controller, επανεκκινεί flags, ξεκινά timer-ISR
//-------------------------------------------------------------------------
void TCPLowLevelInit(void)
{
 BCSCTL1 &= ~DIVA0;                    // ACLK = XT1 / 4 = 2 MHz
 BCSCTL1 |= DIVA1;
 TACTL = ID_3 + TASSEL_1 + MC_2 + TAIE;        //.

 Init8900();
 TransmitControl = 0;
 TCPFlags = 0;
 TCPStateMachine = CLOSED;
 SocketStatus = 0; }
//-------------------------------------------------------------------------
// easyWEB-API function
//-------------------------------------------------------------------------
void TCPPassiveOpen(void)
{
 if (TCPStateMachine == CLOSED)
 {
  TCPFlags &= ~TCP_ACTIVE_OPEN;
  TCPStateMachine = LISTENING;
  SocketStatus = SOCK_ACTIVE;
 }
}


//-------------------------------------------------------------------------
// easyWEB-API function
//-------------------------------------------------------------------------
void TCPActiveOpen(void)
{
 if ((TCPStateMachine == CLOSED) || (TCPStateMachine == LISTENING))
 {
  TCPFlags |= TCP_ACTIVE_OPEN;
  TCPFlags &= ~IP_ADDR_RESOLVED;

  PrepareARP_REQUEST();
  LastFrameSent = ARP_REQUEST;
  TCPStartRetryTimer();
  SocketStatus = SOCK_ACTIVE;
 }
}
```

```
//-----------------------------------------------------------------------------
// easyWEB-API function
//-----------------------------------------------------------------------------
void TCPClose(void)
{
  switch (TCPStateMachine)
  {
    case LISTENING :
    case SYN_SENT :
      TCPStateMachine = CLOSED;
      TCPFlags = 0;
      SocketStatus = 0;
      break;
    case SYN_RECD :
    case ESTABLISHED :
      TCPFlags |= TCP_CLOSE_REQUESTED;
      break;
  }
}



//-----------------------------------------------------------------------------
// easyWEB-API function
//-----------------------------------------------------------------------------
void TCPReleaseRxBuffer(void)
{
  SocketStatus &= ~SOCK_DATA_AVAILABLE;
}



//-----------------------------------------------------------------------------
// easyWEB-API function
//-----------------------------------------------------------------------------
void TCPTransmitTxBuffer(void)
{
  if ((TCPStateMachine == ESTABLISHED) || (TCPStateMachine == CLOSE_WAIT))
    if (SocketStatus & SOCK_TX_BUF_RELEASED)
    {
      SocketStatus &= ~SOCK_TX_BUF_RELEASED;
      TCPUNASeqNr += TCPTxDataCount;

      TxFrame1Size = ETH_HEADER_SIZE + IP_HEADER_SIZE + TCP_HEADER_SIZE +
TCPTxDataCount;
      TransmitControl |= SEND_FRAME1;

      LastFrameSent = TCP_DATA_FRAME;
      TCPStartRetryTimer();
    }
}
```

```
//---------------------------------------------------------------------------
// easyWEB's 'main()'-function
//---------------------------------------------------------------------------
void DoNetworkStuff(void)
{
 unsigned int ActRxEvent;

 Write8900(ADD_PORT, PP_RxEvent);                    // point to RxEvent
 ActRxEvent = Read8900(DATA_PORT);
 if (ActRxEvent & RX_OK)
 {
  if (ActRxEvent & RX_IA) ProcessEthIAFrame();
  if (ActRxEvent & RX_BROADCAST) ProcessEthBroadcastFrame();
 }

 if (TCPFlags & TCP_TIMER_RUNNING)
  if (TCPFlags & TIMER_TYPE_RETRY)
  {
   if (TCPTimer > RETRY_TIMEOUT)
   {
    TCPRestartTimer();

    if (RetryCounter)
    {
     TCPHandleRetransmission();
     RetryCounter--;
    }
    else
    {
     TCPStopTimer();
     TCPHandleTimeout();
    }
   }
  }
  else if (TCPTimer > FIN_TIMEOUT)
  {
   TCPStateMachine = CLOSED;
   TCPFlags = 0;                                     // επανεκκινεί τα flags, σταματά την επαναποστολή
   SocketStatus &= SOCK_DATA_AVAILABLE;
  }

 switch (TCPStateMachine)
 {
  case CLOSED :
  case LISTENING :
   if (TCPFlags & TCP_ACTIVE_OPEN)
    if (TCPFlags & IP_ADDR_RESOLVED)
     if (!(TransmitControl & SEND_FRAME2))
     {
      TCPSeqNr = ((unsigned long)ISNGenHigh << 16) | TAR;
      TCPUNASeqNr = TCPSeqNr;
      TCPAckNr = 0;
```

```
          TCPUNASeqNr++;
          PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_SYN);
          LastFrameSent = TCP_SYN_FRAME;
          TCPStartRetryTimer();
          TCPStateMachine = SYN_SENT;
         }
      break;
    case SYN_RECD :
    case ESTABLISHED :
      if (TCPFlags & TCP_CLOSE_REQUESTED)
       if (!(TransmitControl & (SEND_FRAME2 | SEND_FRAME1)))
         if (TCPSeqNr == TCPUNASeqNr)
         {
          TCPUNASeqNr++;
          PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_FIN | TCP_CODE_ACK);
          LastFrameSent = TCP_FIN_FRAME;
          TCPStartRetryTimer();
          TCPStateMachine = FIN_WAIT_1;
         }
      break;
    case CLOSE_WAIT :
      if (!(TransmitControl & (SEND_FRAME2 | SEND_FRAME1)))
        if (TCPSeqNr == TCPUNASeqNr)
        {
         TCPUNASeqNr++;
         PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_FIN | TCP_CODE_ACK);  // we NEED
a retry-timeout
         LastFrameSent = TCP_FIN_FRAME;
         TCPStartRetryTimer();
         TCPStateMachine = LAST_ACK;
        }
      break;
  }

  if (TransmitControl & SEND_FRAME2)
  {
   RequestSend(TxFrame2Size);

   if (Rdy4Tx())
   {
    CopyToFrame8900((unsigned char *)TxFrame2Mem, TxFrame2Size);
   }
   else
   {
    TCPStateMachine = CLOSED;
    SocketStatus = SOCK_ERR_ETHERNET;          // δείχνει λάθος στο χρήση
    TCPFlags = 0;                               // clear all flags, stop timers etc
   }
   TransmitControl &= ~SEND_FRAME2;             // clear tx-flag
  }

  if (TransmitControl & SEND_FRAME1)
  {
```

```
   PrepareTCP_DATA_FRAME();
   RequestSend(TxFrame1Size);

   if (Rdy4Tx())
   {
    CopyToFrame8900((unsigned char *)TxFrame1Mem, TxFrame1Size);
   }
   else
   {
    TCPStateMachine = CLOSED;
    SocketStatus = SOCK_ERR_ETHERNET;        //δείχνει λάθος στο χρήστη
    TCPFlags = 0;                             // clear all flags, stop timers etc.
   }

   TransmitControl &= ~SEND_FRAME1;          // clear tx-flag
  }
}



//-----------------------------------------------------------------------------
// easyWEB internal function
// handles an incoming broadcast frame
//-----------------------------------------------------------------------------
static void ProcessEthBroadcastFrame(void)
{
 unsigned int TargetIP[2];

 // next two words MUST be read with High-Byte 1st (CS8900 AN181 Page 2)
 ReadHB1ST8900(RX_FRAME_PORT);
 RecdFrameLength = ReadHB1ST8900(RX_FRAME_PORT);          // λήψη πραγματικού μεγέθους
πακέτου

 DummyReadFrame8900(6);
 CopyFromFrame8900(&RecdFrameMAC, 6);

 if (ReadFrameBE8900() == FRAME_ARP)                      // λήψη τύπου πακέτου, έλεγχος για
ARP
   if (ReadFrameBE8900() == HARDW_ETH10)                  // Ethernet frame
    if (ReadFrameBE8900() == FRAME_IP)                    // έλεγχος πρωτοκόλλου
     if (ReadFrameBE8900() == IP_HLEN_PLEN)               // έλεγχος για HLEN, PLEN
      if (ReadFrameBE8900() == OP_ARP_REQUEST)
      {
       DummyReadFrame8900(6);
       CopyFromFrame8900(&RecdFrameIP, 4);
       DummyReadFrame8900(6);
       CopyFromFrame8900(&TargetIP, 4);
       if ((MyIP[0] == TargetIP[0]) && (MyIP[1] == TargetIP[1]))
        PrepareARP_ANSWER();                              // yes->δημιουργία ARP_ANSWER
πακέτου
      }
}
```

```
//-----------------------------------------------------------------------------
// easyWEB internal function
//-----------------------------------------------------------------------------
static void ProcessEthIAFrame(void)
{
 unsigned int TargetIP[2];
 unsigned char ProtocolType;

 // next two words MUST be read with High-Byte 1st (CS8900 AN181 Page 2)
 ReadHB1ST8900(RX_FRAME_PORT);                          // αγνόηση RxStatus Word
 RecdFrameLength = ReadHB1ST8900(RX_FRAME_PORT);        // λήψη πραγματικού μεγέθους
πακέτου

 DummyReadFrame8900(6);                                 // αγνόηση DA
 CopyFromFrame8900(&RecdFrameMAC, 6);                   // αποθήκευση SA

 switch (ReadFrameBE8900())                             // λήψη τύπου πακέτου
 {
  case FRAME_ARP :                                      // έλεγχος για ARP
   if ((TCPFlags & (TCP_ACTIVE_OPEN | IP_ADDR_RESOLVED)) == TCP_ACTIVE_OPEN)
    if (ReadFrameBE8900() == HARDW_ETH10)
     if (ReadFrameBE8900() == FRAME_IP)
      if (ReadFrameBE8900() == IP_HLEN_PLEN)
       if (ReadFrameBE8900() == OP_ARP_ANSWER)
       {
        TCPStopTimer();
        CopyFromFrame8900(&RemoteMAC, 6);
        TCPFlags |= IP_ADDR_RESOLVED;
       }
   break;
  case FRAME_IP :                                       // έλεγχος για IP-type
   if ((ReadFrameBE8900() & 0xff00 ) == IP_VER_IHL)     // IPv4, IHL=5 (20 Bytes Header)
   {                                                    // αγνόηση τύπου υπηρεσίας
    RecdIPFrameLength = ReadFrameBE8900();               // get IP frame's length
    ReadFrameBE8900();                                  // ignore identification

    if (!(ReadFrameBE8900() & (IP_FLAG_MOREFRAG | IP_FRAGOFS_MASK)))
    {
     ProtocolType = ReadFrameBE8900();                  // λήψη πρωτοκόλλου,αγνόηση
TTL
     ReadFrameBE8900();                                 // αγνόηση checksum
     RecdFrameIP[0] = ReadFrame8900();                  // λήψη source IP
     RecdFrameIP[1] = ReadFrame8900();
     TargetIP[0] = ReadFrame8900();                      // λήψη προορισμού IP
     TargetIP[1] = ReadFrame8900();


     if ((MyIP[0] == TargetIP[0]) && (MyIP[1] == TargetIP[1]))
      {
       case PROT_ICMP :
        ProcessICMPFrame();
        break;
```

```
            case PROT_TCP :
              ProcessTCPFrame();
              break;
            case PROT_UDP :
              break;
           }
        }
      break;
    }
  }
}


//-----------------------------------------------------------------------------
// easyWEB internal function
//-----------------------------------------------------------------------------
static void ProcessICMPFrame(void)
{
 unsigned int ICMPTypeAndCode;

 ICMPTypeAndCode = ReadFrameBE8900();          // λήψη τύπου μηνύματος και κώδικας
 ReadFrameBE8900();                            // αγνόηση ICMP checksum

 switch (ICMPTypeAndCode >> 8)                 // έλεγχος τύπου
 {
  case ICMP_ECHO :
    PrepareICMP_ECHO_REPLY();
    break;
 }
}



//-----------------------------------------------------------------------------
// easyWEB internal function
//-----------------------------------------------------------------------------
static void ProcessTCPFrame(void)
{
 unsigned int TCPSegSourcePort;                // segment's source port
 unsigned int TCPSegDestPort;                  // segment's destination port
 unsigned long TCPSegSeq;                      // segment's sequence number
 unsigned long TCPSegAck;                      // segment's acknowledge number
 unsigned int TCPCode;                         // TCP κώδικας και μήκος επικεφαλίδας
 unsigned char TCPHeaderSize;                  // πραγματικό μήκος επικεφαλίδας TCP
 unsigned int NrOfDataBytes;                   // πραγματικός αριθμός δεδομένων

 TCPSegSourcePort = ReadFrameBE8900();             // get ports
 TCPSegDestPort = ReadFrameBE8900();

 if (TCPSegDestPort != TCPLocalPort) return;
```

```
  TCPSegSeq = (unsigned long)ReadFrameBE8900() << 16;
  TCPSegSeq |= ReadFrameBE8900();

  TCPSegAck = (unsigned long)ReadFrameBE8900() << 16;
  TCPSegAck |= ReadFrameBE8900();

  TCPCode = ReadFrameBE8900();

  TCPHeaderSize = (TCPCode & DATA_OFS_MASK) >> 10;        // μήκος επικεφαλίδας σε Bytes
  NrOfDataBytes = RecdIPFrameLength - IP_HEADER_SIZE - TCPHeaderSize;

  if (NrOfDataBytes > MAX_TCP_RX_DATA_SIZE) return;
  if (TCPHeaderSize > TCP_HEADER_SIZE)
    DummyReadFrame8900(TCPHeaderSize - TCP_HEADER_SIZE);


 switch (TCPStateMachine)
 {                                      // RFC793
   case CLOSED :
     if (!(TCPCode & TCP_CODE_RST))
     {
       TCPRemotePort = TCPSegSourcePort;                    // λήψη remote TCP port

       RemoteMAC[0] = RecdFrameMAC[0];                      // αποθήκευση MAC and IP
       RemoteMAC[1] = RecdFrameMAC[1];                      // για μεταγενέστερη χρήση
       RemoteMAC[2] = RecdFrameMAC[2];
       RemoteIP[0] = RecdFrameIP[0];
       RemoteIP[1] = RecdFrameIP[1];

       if (TCPCode & TCP_CODE_ACK)
       {
         PrepareTCP_FRAME(TCPSegAck, 0, TCP_CODE_RST);    // TCP
       }
       else
       {
         TCPAckNr = TCPSegSeq + NrOfDataBytes;
         if (TCPCode & (TCP_CODE_SYN | TCP_CODE_FIN)) TCPAckNr++;
         PrepareTCP_FRAME(0, TCPAckNr, TCP_CODE_RST | TCP_CODE_ACK);
       }
     }
     break;
   case LISTENING :
     if (!(TCPCode & TCP_CODE_RST))
     {
       TCPRemotePort = TCPSegSourcePort;                    // λήψη remote TCP port

       RemoteMAC[0] = RecdFrameMAC[0];                      // αποθήκευση MAC and IP
       RemoteMAC[1] = RecdFrameMAC[1];
       RemoteMAC[2] = RecdFrameMAC[2];
       RemoteIP[0] = RecdFrameIP[0];
       RemoteIP[1] = RecdFrameIP[1];

       if (TCPCode & TCP_CODE_ACK)
```

```
      {
       PrepareTCP_FRAME(TCPSegAck, 0, TCP_CODE_RST);
      }
      else if (TCPCode & TCP_CODE_SYN)
      {
       // initialize global connection variables
       TCPAckNr = TCPSegSeq + 1;
       TCPSeqNr = ((unsigned long)ISNGenHigh << 16) | TAR;                    // θέτει local ISN
       TCPUNASeqNr = TCPSeqNr + 1;                                            // one byte out -> increase
by one
       PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_SYN | TCP_CODE_ACK);
       LastFrameSent = TCP_SYN_ACK_FRAME;
       TCPStartRetryTimer();
       TCPStateMachine = SYN_RECD;
      }
     }
     break;
   case SYN_SENT :

     if ((RemoteIP[0] != RecdFrameIP[0]) || (RemoteIP[1] != RecdFrameIP[1]))
       break;

         if (TCPSegSourcePort != TCPRemotePort)
       break;

     if (TCPCode & TCP_CODE_ACK)
       if (TCPSegAck != TCPUNASeqNr)               {
        if (!(TCPCode & TCP_CODE_RST))
        {
         PrepareTCP_FRAME(TCPSegAck, 0, TCP_CODE_RST);
        }
        break;                          // drop segment
       }

     if (TCPCode & TCP_CODE_RST)                   // RST??
     {
       if (TCPCode & TCP_CODE_ACK)                 // ACK είναι δεκτή, επανεκκίνηση
       {                                           // σύνδεση
        TCPStateMachine = CLOSED;
        TCPFlags = 0;                              // επανεκκίνηση όλων των flags, σταμάτημα
επαναποστολής
        SocketStatus = SOCK_ERR_CONN_RESET;
       }
       break;
     }

     if (TCPCode & TCP_CODE_SYN)               // SYN??
     {
       TCPAckNr = TCPSegSeq;
       TCPAckNr++;

       if (TCPCode & TCP_CODE_ACK)
       {
```

```
        TCPStopTimer();
        TCPSeqNr = TCPUNASeqNr;
        PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_ACK);       // ACK this ISN
        TCPStateMachine = ESTABLISHED;
        SocketStatus |= SOCK_CONNECTED;
        SocketStatus |= SOCK_TX_BUF_RELEASED;         }
       else
       {
        TCPStopTimer();
        PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_SYN | TCP_CODE_ACK);
        LastFrameSent = TCP_SYN_ACK_FRAME;              // συνέχεια αποστολής
        TCPStartRetryTimer();                           // SYN_ACK frames
        TCPStateMachine = SYN_RECD;
       }
      }
      break;
    default :
          if ((RemoteIP[0] != RecdFrameIP[0]) || (RemoteIP[1] != RecdFrameIP[1]))
      break;

     if (TCPSegSourcePort != TCPRemotePort)
      break;

     if ((TCPSegSeq < TCPAckNr) || (TCPSegSeq >= TCPAckNr + MAX_TCP_RX_DATA_SIZE))
      break;

     if (TCPCode & TCP_CODE_RST)                        // RST??
     {
      TCPStateMachine = CLOSED;                         // κλείσιμο της state machine
      TCPFlags = 0;                                     // επανεκκίνηση όλων των flags, τέλος
επαναποστολής
      SocketStatus = SOCK_ERR_CONN_RESET;               // δείχνει ένα λάθος στο χρήστη
      break;
     }

     if (TCPCode & TCP_CODE_SYN)                        // SYN??
     {
      PrepareTCP_FRAME(TCPSegAck, 0, TCP_CODE_RST);
      TCPStateMachine = CLOSED;                         // κλείσιμο σύνδεσης
      TCPFlags = 0;                                     //επανεκκίνηση όλων των flags, τέλος
επαναποστολής
      SocketStatus = SOCK_ERR_REMOTE;
      break;
     }

     if (TCPSegSeq != TCPAckNr)
     {                          // and send an ACK
      PrepareTCP_FRAME(TCPUNASeqNr, TCPAckNr, TCP_CODE_ACK);
      break;
     }

     if (!(TCPCode & TCP_CODE_ACK)) break;
```

```
    if (TCPSegAck == TCPUNASeqNr)
    {
     TCPStopTimer();
     TCPSeqNr = TCPUNASeqNr;

     switch (TCPStateMachine)                    // αλλαγή κατάστασης αν είναι αναγκαίο
     {
      case SYN_RECD :                            // ACK of our SYN?
       TCPStateMachine = ESTABLISHED;            // ο χρήστης μπορεί να στείλει δεδομένα
τώρα
       SocketStatus |= SOCK_CONNECTED;
       break;
      case ESTABLISHED :
       SocketStatus |= SOCK_TX_BUF_RELEASED;
       break;
      case FIN_WAIT_1 :
       TCPStateMachine = FIN_WAIT_2;

       TCPStartFinTimer();                       // εκκίνηση TIME_WAIT timeout
       break;
      case CLOSING :
       TCPStateMachine = TIME_WAIT;
       TCPStartFinTimer();                       // εκκίνηση TIME_WAIT timeout
       break;
      case LAST_ACK :
       TCPStateMachine = CLOSED;
       TCPFlags = 0;                             // επανεκκίνηση όλων των flags, τέλος
επαναποστολής
       SocketStatus &= SOCK_DATA_AVAILABLE;      // εκκαθάριση όλων των flags αλλά
δεδομένα διαθέσιμα
       break;
      case TIME_WAIT :
       // ACK a retransmission of remote FIN
       PrepareTCP_FRAME(TCPSegAck, TCPAckNr, TCP_CODE_ACK);
       TCPRestartTimer();                        // επανεκκίνηση TIME_WAIT timeout
       break;      }     }
    if ((TCPStateMachine == ESTABLISHED) ||
      (TCPStateMachine == FIN_WAIT_1) ||
      (TCPStateMachine == FIN_WAIT_2))
     if (NrOfDataBytes)
      if (!(SocketStatus & SOCK_DATA_AVAILABLE))
      {
       DummyReadFrame8900(6);                    //αγνόηση παραθύρου, checksum, urgent pointer
       CopyFromFrame8900(RxTCPBufferMem, NrOfDataBytes);
       TCPRxDataCount = NrOfDataBytes;
       SocketStatus |= SOCK_DATA_AVAILABLE;
       TCPAckNr += NrOfDataBytes;
       PrepareTCP_FRAME(TCPSegAck, TCPAckNr, TCP_CODE_ACK);
      }
      else
       break
    if (TCPCode & TCP_CODE_FIN)
    {
```

```
      switch (TCPStateMachine)
      {
        case SYN_RECD :
        case ESTABLISHED :
          TCPStateMachine = CLOSE_WAIT;
          break;
        case FIN_WAIT_1 :
          TCPStateMachine = CLOSING;
          SocketStatus &= ~SOCK_CONNECTED;     // TIME_WAIT
          break;
        case FIN_WAIT_2 :
          TCPStateMachine = TIME_WAIT;
          SocketStatus &= ~SOCK_CONNECTED;
          TCPStartFinTimer();                  // εκκίνηση TIME_WAIT timeout
          break;
        case TIME_WAIT :
          TCPRestartTimer();                   // επανεκκίνηση TIME_WAIT timeout
          break;
      }
      TCPAckNr++;                              // ACK remote's FIN flag

      PrepareTCP_FRAME(TCPSegAck, TCPAckNr, TCP_CODE_ACK);
    }
  }
}
```

```
//-----------------------------------------------------------------------------
// easyWEB internal function
//-----------------------------------------------------------------------------
static void PrepareARP_REQUEST(void)
{
  // Ethernet
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS) = 0xffff;
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 2) = 0xffff;
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 4) = 0xffff;
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS) = MyMAC[0];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 2) = MyMAC[1];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 4) = MyMAC[2];
  ACCESS_UINT(TxFrame2Mem, ETH_TYPE_OFS) = SWAPB(FRAME_ARP);

  // ARP
  ACCESS_UINT(TxFrame2Mem, ARP_HARDW_OFS) = SWAPB(HARDW_ETH10);
  ACCESS_UINT(TxFrame2Mem, ARP_PROT_OFS) = SWAPB(FRAME_IP);
  ACCESS_UINT(TxFrame2Mem, ARP_HLEN_PLEN_OFS) = SWAPB(IP_HLEN_PLEN);
  ACCESS_UINT(TxFrame2Mem, ARP_OPCODE_OFS) = SWAPB(OP_ARP_REQUEST);
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_HA_OFS) = MyMAC[0];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_HA_OFS + 2) = MyMAC[1];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_HA_OFS + 4) = MyMAC[2];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_IP_OFS) = MyIP[0];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_IP_OFS + 2) = MyIP[1];
```

```
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_HA_OFS) = 0;
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_HA_OFS + 2) = 0;
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_HA_OFS + 4) = 0;
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS) = 0;
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS + 2) = 0;
  if (((RemoteIP[0] ^ MyIP[0]) & SubnetMask[0]) ||
     ((RemoteIP[1] ^ MyIP[1]) & SubnetMask[1]))
  {
    ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS) = GatewayIP[0];
    ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS + 2) = GatewayIP[1];
  }
  else
  {
    ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS) = RemoteIP[0];
    ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS + 2) = RemoteIP[1];
  }

  TxFrame2Size = ETH_HEADER_SIZE + ARP_FRAME_SIZE;
  TransmitControl |= SEND_FRAME2;
}

//-----------------------------------------------------------------------------
// easyWEB internal function
//-----------------------------------------------------------------------------
static void PrepareARP_ANSWER(void)
{
  // Ethernet
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS) = RecdFrameMAC[0];
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 2) = RecdFrameMAC[1];
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 4) = RecdFrameMAC[2];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS) = MyMAC[0];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 2) = MyMAC[1];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 4) = MyMAC[2];
  ACCESS_UINT(TxFrame2Mem, ETH_TYPE_OFS) = SWAPB(FRAME_ARP);

  // ARP
  ACCESS_UINT(TxFrame2Mem, ARP_HARDW_OFS) = SWAPB(HARDW_ETH10);
  ACCESS_UINT(TxFrame2Mem, ARP_PROT_OFS) = SWAPB(FRAME_IP);
  ACCESS_UINT(TxFrame2Mem, ARP_HLEN_PLEN_OFS) = SWAPB(IP_HLEN_PLEN);
  ACCESS_UINT(TxFrame2Mem, ARP_OPCODE_OFS) = SWAPB(OP_ARP_ANSWER);
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_HA_OFS) = MyMAC[0];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_HA_OFS + 2) = MyMAC[1];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_HA_OFS + 4) = MyMAC[2];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_IP_OFS) = MyIP[0];
  ACCESS_UINT(TxFrame2Mem, ARP_SENDER_IP_OFS + 2) = MyIP[1];
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_HA_OFS) = RecdFrameMAC[0];
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_HA_OFS + 2) = RecdFrameMAC[1];
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_HA_OFS + 4) = RecdFrameMAC[2];
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS) = RecdFrameIP[0];
  ACCESS_UINT(TxFrame2Mem, ARP_TARGET_IP_OFS + 2) = RecdFrameIP[1];
  TxFrame2Size = ETH_HEADER_SIZE + ARP_FRAME_SIZE;
  TransmitControl |= SEND_FRAME2;}
```

```
//-------------------------------------------------------------------------
// easyWEB internal function
//-------------------------------------------------------------------------
static void PrepareICMP_ECHO_REPLY(void)
{
 unsigned int ICMPDataCount;

 if (RecdIPFrameLength > MAX_ETH_TX_DATA_SIZE)
  ICMPDataCount = MAX_ETH_TX_DATA_SIZE - IP_HEADER_SIZE - ICMP_HEADER_SIZE;
 else
  ICMPDataCount = RecdIPFrameLength - IP_HEADER_SIZE - ICMP_HEADER_SIZE;

 // Ethernet
 ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS) = RecdFrameMAC[0];
 ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 2) = RecdFrameMAC[1];
 ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 4) = RecdFrameMAC[2];
 ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS) = MyMAC[0];
 ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 2) = MyMAC[1];
 ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 4) = MyMAC[2];
 ACCESS_UINT(TxFrame2Mem, ETH_TYPE_OFS) = SWAPB(FRAME_IP);

 // IP
 ACCESS_UINT(TxFrame2Mem, IP_VER_IHL_TOS_OFS) = SWAPB(IP_VER_IHL);
 ACCESS_UINT(TxFrame2Mem, IP_TOTAL_LENGTH_OFS) =
  __swap_bytes(IP_HEADER_SIZE + ICMP_HEADER_SIZE + ICMPDataCount);
 ACCESS_UINT(TxFrame2Mem, IP_IDENT_OFS) = 0;
 ACCESS_UINT(TxFrame2Mem, IP_FLAGS_FRAG_OFS) = 0;
 ACCESS_UINT(TxFrame2Mem, IP_TTL_PROT_OFS) = SWAPB((DEFAULT_TTL << 8) |
PROT_ICMP);
 ACCESS_UINT(TxFrame2Mem, IP_HEAD_CHKSUM_OFS) = 0;
 ACCESS_UINT(TxFrame2Mem, IP_SOURCE_OFS) = MyIP[0];
 ACCESS_UINT(TxFrame2Mem, IP_SOURCE_OFS + 2) = MyIP[1];
 ACCESS_UINT(TxFrame2Mem, IP_DESTINATION_OFS) = RecdFrameIP[0];
 ACCESS_UINT(TxFrame2Mem, IP_DESTINATION_OFS + 2) = RecdFrameIP[1];
 ACCESS_UINT(TxFrame2Mem, IP_HEAD_CHKSUM_OFS) =
  CalcChecksum((unsigned char *)TxFrame2Mem + IP_VER_IHL_TOS_OFS, IP_HEADER_SIZE, 0);

 // ICMP
 ACCESS_UINT(TxFrame2Mem, ICMP_TYPE_CODE_OFS) = SWAPB(ICMP_ECHO_REPLY << 8);
 ACCESS_UINT(TxFrame2Mem, ICMP_CHKSUM_OFS) = 0;          // αρχικοποίηση του checksum
πεδίου
 CopyFromFrame8900((unsigned char *)TxFrame2Mem + ICMP_DATA_OFS, ICMPDataCount);
ACCESS_UINT(TxFrame2Mem, ICMP_CHKSUM_OFS) =
   CalcChecksum((unsigned char *)TxFrame2Mem + IP_DATA_OFS, ICMPDataCount +
    ICMP_HEADER_SIZE, 0);

 TxFrame2Size = ETH_HEADER_SIZE + IP_HEADER_SIZE + ICMP_HEADER_SIZE +
ICMPDataCount;
 TransmitControl |= SEND_FRAME2;
}
```

```
//---------------------------------------------------------------------------
// easyWEB internal function
//---------------------------------------------------------------------------
static void PrepareTCP_FRAME(unsigned long seqnr, unsigned long acknr,
  unsigned int TCPCode)
{
  // Ethernet
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS) = RemoteMAC[0];
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 2) = RemoteMAC[1];
  ACCESS_UINT(TxFrame2Mem, ETH_DA_OFS + 4) = RemoteMAC[2];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS) = MyMAC[0];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 2) = MyMAC[1];
  ACCESS_UINT(TxFrame2Mem, ETH_SA_OFS + 4) = MyMAC[2];
  ACCESS_UINT(TxFrame2Mem, ETH_TYPE_OFS) = SWAPB(FRAME_IP);

  // IP
  ACCESS_UINT(TxFrame2Mem, IP_VER_IHL_TOS_OFS) = SWAPB(IP_VER_IHL);

  if (TCPCode & TCP_CODE_SYN)
    ACCESS_UINT(TxFrame2Mem, IP_TOTAL_LENGTH_OFS) =
      SWAPB(IP_HEADER_SIZE + TCP_HEADER_SIZE + TCP_OPT_MSS_SIZE);
  else
    ACCESS_UINT(TxFrame2Mem, IP_TOTAL_LENGTH_OFS) =
      SWAPB(IP_HEADER_SIZE + TCP_HEADER_SIZE);

  ACCESS_UINT(TxFrame2Mem, IP_IDENT_OFS) = 0;
  ACCESS_UINT(TxFrame2Mem, IP_FLAGS_FRAG_OFS) = 0;
  ACCESS_UINT(TxFrame2Mem, IP_TTL_PROT_OFS) = SWAPB((DEFAULT_TTL << 8) |
PROT_TCP);
  ACCESS_UINT(TxFrame2Mem, IP_HEAD_CHKSUM_OFS) = 0;
  ACCESS_UINT(TxFrame2Mem, IP_SOURCE_OFS) = MyIP[0];
  ACCESS_UINT(TxFrame2Mem, IP_SOURCE_OFS + 2) = MyIP[1];
  ACCESS_UINT(TxFrame2Mem, IP_DESTINATION_OFS) = RemoteIP[0];
  ACCESS_UINT(TxFrame2Mem, IP_DESTINATION_OFS + 2) = RemoteIP[1];
  ACCESS_UINT(TxFrame2Mem, IP_HEAD_CHKSUM_OFS) =
    CalcChecksum((unsigned char *)TxFrame2Mem + IP_VER_IHL_TOS_OFS,
      IP_HEADER_SIZE, 0);
   // TCP
  ACCESS_UINT(TxFrame2Mem, TCP_SRCPORT_OFS) = __swap_bytes(TCPLocalPort);
  ACCESS_UINT(TxFrame2Mem, TCP_DESTPORT_OFS) = __swap_bytes(TCPRemotePort);
  WriteDWBE((unsigned char *)TxFrame2Mem + TCP_SEQNR_OFS, seqnr);
  WriteDWBE((unsigned char *)TxFrame2Mem + TCP_ACKNR_OFS, acknr);
  ACCESS_UINT(TxFrame2Mem, TCP_WINDOW_OFS) = SWAPB(MAX_TCP_RX_DATA_SIZE);
                                                    // Bytes δεδομένα για αποδοχή
 ACCESS_UINT(TxFrame2Mem, TCP_CHKSUM_OFS) = 0;      // αρχικοποίηση του checksum
  ACCESS_UINT(TxFrame2Mem, TCP_URGENT_OFS) = 0;
  if (TCPCode & TCP_CODE_SYN)                {
    ACCESS_UINT(TxFrame2Mem, TCP_DATA_CODE_OFS) = SWAPB(0x6000 | TCPCode);
                                                    // TCP μήκος επικεφαλίδας = 24
    ACCESS_UINT(TxFrame2Mem, TCP_DATA_OFS) = SWAPB(TCP_OPT_MSS);
    ACCESS_UINT(TxFrame2Mem, TCP_DATA_OFS + 2) = SWAPB(MAX_TCP_RX_DATA_SIZE);
ACCESS_UINT(TxFrame2Mem, TCP_CHKSUM_OFS) =
```

```
    CalcChecksum((unsigned char *)TxFrame2Mem + TCP_SRCPORT_OFS,
      TCP_HEADER_SIZE + TCP_OPT_MSS_SIZE, 1);
   TxFrame2Size = ETH_HEADER_SIZE + IP_HEADER_SIZE + TCP_HEADER_SIZE +
    TCP_OPT_MSS_SIZE;
  }
  else
  {
   ACCESS_UINT(TxFrame2Mem, TCP_DATA_CODE_OFS) = SWAPB(0x5000 | TCPCode);
                                                          // TCP μήκος επικεφαλίδας = 20
   ACCESS_UINT(TxFrame2Mem, TCP_CHKSUM_OFS) =
    CalcChecksum((unsigned char *)TxFrame2Mem + TCP_SRCPORT_OFS,
      TCP_HEADER_SIZE, 1);
   TxFrame2Size = ETH_HEADER_SIZE + IP_HEADER_SIZE + TCP_HEADER_SIZE;
  }
  TransmitControl |= SEND_FRAME2;}


//---------------------------------------------------------------------------
// easyWEB internal function
//---------------------------------------------------------------------------
static void PrepareTCP_DATA_FRAME(void)
{
  // Ethernet
  ACCESS_UINT(TxFrame1Mem, ETH_DA_OFS) = RemoteMAC[0];
  ACCESS_UINT(TxFrame1Mem, ETH_DA_OFS + 2) = RemoteMAC[1];
  ACCESS_UINT(TxFrame1Mem, ETH_DA_OFS + 4) = RemoteMAC[2];
  ACCESS_UINT(TxFrame1Mem, ETH_SA_OFS) = MyMAC[0];
  ACCESS_UINT(TxFrame1Mem, ETH_SA_OFS + 2) = MyMAC[1];
  ACCESS_UINT(TxFrame1Mem, ETH_SA_OFS + 4) = MyMAC[2];
  ACCESS_UINT(TxFrame1Mem, ETH_TYPE_OFS) = SWAPB(FRAME_IP);

  // IP
  ACCESS_UINT(TxFrame1Mem, IP_VER_IHL_TOS_OFS) = SWAPB(IP_VER_IHL);
  ACCESS_UINT(TxFrame1Mem, IP_TOTAL_LENGTH_OFS) =
    __swap_bytes(IP_HEADER_SIZE + TCP_HEADER_SIZE + TCPTxDataCount);
  ACCESS_UINT(TxFrame1Mem, IP_IDENT_OFS) = 0;
  ACCESS_UINT(TxFrame1Mem, IP_FLAGS_FRAG_OFS) = 0;
  ACCESS_UINT(TxFrame1Mem, IP_TTL_PROT_OFS) = SWAPB((DEFAULT_TTL << 8) |
PROT_TCP);
  ACCESS_UINT(TxFrame1Mem, IP_HEAD_CHKSUM_OFS) = 0;
  ACCESS_UINT(TxFrame1Mem, IP_SOURCE_OFS) = MyIP[0];
  ACCESS_UINT(TxFrame1Mem, IP_SOURCE_OFS + 2) = MyIP[1];
  ACCESS_UINT(TxFrame1Mem, IP_DESTINATION_OFS) = RemoteIP[0];
  ACCESS_UINT(TxFrame1Mem, IP_DESTINATION_OFS + 2) = RemoteIP[1];
  ACCESS_UINT(TxFrame1Mem, IP_HEAD_CHKSUM_OFS) =
    CalcChecksum((unsigned char *)TxFrame1Mem + IP_VER_IHL_TOS_OFS,
      IP_HEADER_SIZE, 0);
```

```
 // TCP
  ACCESS_UINT(TxFrame1Mem, TCP_SRCPORT_OFS) = __swap_bytes(TCPLocalPort);
  ACCESS_UINT(TxFrame1Mem, TCP_DESTPORT_OFS) = __swap_bytes(TCPRemotePort);

  WriteDWBE((unsigned char *)TxFrame1Mem + TCP_SEQNR_OFS, TCPSeqNr);
  WriteDWBE((unsigned char *)TxFrame1Mem + TCP_ACKNR_OFS, TCPAckNr);

  ACCESS_UINT(TxFrame1Mem, TCP_DATA_CODE_OFS) = SWAPB(0x5000 | TCP_CODE_ACK);
  ACCESS_UINT(TxFrame1Mem, TCP_WINDOW_OFS) = SWAPB(MAX_TCP_RX_DATA_SIZE);
ACCESS_UINT(TxFrame1Mem, TCP_CHKSUM_OFS) = 0;
  ACCESS_UINT(TxFrame1Mem, TCP_URGENT_OFS) = 0;
  ACCESS_UINT(TxFrame1Mem, TCP_CHKSUM_OFS) =
    CalcChecksum((unsigned char *)TxFrame1Mem + TCP_SRCPORT_OFS,
      TCP_HEADER_SIZE + TCPTxDataCount, 1);
}



//---------------------------------------------------------------------------
// easyWEB internal function
//---------------------------------------------------------------------------
static unsigned int CalcChecksum(void *Start, unsigned int Count,
  unsigned char IsTCP)
{
  unsigned long Sum = 0;
  unsigned int *pStart = Start;

  if (IsTCP)
  {
    Sum += MyIP[0];                         //εισαγωγή και της ψευδο-επικεφαλίδας
    Sum += MyIP[1];
    Sum += RemoteIP[0];
    Sum += RemoteIP[1];
    Sum += __swap_bytes(Count);             // μήκος επικεφαλίδας TCP και μήκος δεδομένων
    Sum += SWAPB(PROT_TCP);
  }

  while (Count > 1)
  {
    Sum += *pStart++;
    Count -= 2;
  }

  if (Count)
    Sum += *(unsigned char *)pStart;

  while (Sum >> 16)
    Sum = (Sum & 0xFFFF) + (Sum >> 16);

  return ~Sum;
}
```

```
//-----------------------------------------------------------------------------
// easyWEB internal function
//-----------------------------------------------------------------------------
static void TCPStartRetryTimer(void)
{
  TCPTimer = 0;
  RetryCounter = MAX_RETRYS;
  TCPFlags |= TCP_TIMER_RUNNING;
  TCPFlags |= TIMER_TYPE_RETRY;
}


//-----------------------------------------------------------------------------
// easyWEB internal function
//-----------------------------------------------------------------------------
static void TCPStartFinTimer(void)
{
  TCPTimer = 0;
  TCPFlags |= TCP_TIMER_RUNNING;
  TCPFlags &= ~TIMER_TYPE_RETRY;
}

//-----------------------------------------------------------------------------
// easyWEB internal function
// επανεκκίνηση του timer
//-----------------------------------------------------------------------------
static void TCPRestartTimer(void)
{
  TCPTimer = 0;
}

//-----------------------------------------------------------------------------
// easyWEB internal function
// σταμάτημα του timer
//-----------------------------------------------------------------------------
static void TCPStopTimer(void)
{
  TCPFlags &= ~TCP_TIMER_RUNNING;
}
```

```
//-------------------------------------------------------------------------
// easyWEB internal function
//-------------------------------------------------------------------------
static void TCPHandleRetransmission(void)
{
 switch (LastFrameSent)
 {
  case ARP_REQUEST :
   PrepareARP_REQUEST();
   break;
  case TCP_SYN_FRAME :
   PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_SYN);
   break;
  case TCP_SYN_ACK_FRAME :
   PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_SYN | TCP_CODE_ACK);
   break;
  case TCP_FIN_FRAME :
   PrepareTCP_FRAME(TCPSeqNr, TCPAckNr, TCP_CODE_FIN | TCP_CODE_ACK);
   break;
  case TCP_DATA_FRAME :
   TransmitControl |= SEND_FRAME1;
   break;
 }
}


//-------------------------------------------------------------------------
// easyWEB internal function
//-------------------------------------------------------------------------
static void TCPHandleTimeout(void)
{
 TCPStateMachine = CLOSED;

 if ((TCPFlags & (TCP_ACTIVE_OPEN | IP_ADDR_RESOLVED)) == TCP_ACTIVE_OPEN)
  SocketStatus = SOCK_ERR_ARP_TIMEOUT;        //δείχνει στο χρήστη ένα λάθος
 else
  SocketStatus = SOCK_ERR_TCP_TIMEOUT;

 TCPFlags = 0;                                          // καθαρίζει όλα τα flags
}

//-------------------------------------------------------------------------
// easyWEB internal function
//-------------------------------------------------------------------------
#pragma vector = TIMERA1_VECTOR
__interrupt void TCPClockHandler(void)
{
 if (TAIV == 10)
 {
  ISNGenHigh++;
  TCPTimer++;   }}
```

```
//-------------------------------------------------------------------------
// Όνομα: tcpip.h
// Func: header-file for tcpip.c
//-------------------------------------------------------------------------

#ifndef __TCPIP_H
#define __TCPIP_H

#include "msp430x14x.h"

// easyWEB-stack definitions
#define MYIP_1          192             // H IP διεύθυνση
#define MYIP_2          168
#define MYIP_3          0
#define MYIP_4          30

#define SUBMASK_1       255             // subnet mask
#define SUBMASK_2       255
#define SUBMASK_3       255
#define SUBMASK_4       0

#define GWIP_1          192             // standard gateway
#define GWIP_2          168
#define GWIP_3          0
#define GWIP_4          1

#define RETRY_TIMEOUT   8
#define FIN_TIMEOUT     2

#define MAX_RETRYS      4

#define MAX_TCP_TX_DATA_SIZE 768
#define MAX_TCP_RX_DATA_SIZE 536
#define MAX_ETH_TX_DATA_SIZE 60
#define DEFAULT_TTL     64

// Ethernet network layer definitions
#define ETH_DA_OFS      0               // Destination MAC address (48 Bit)
#define ETH_SA_OFS      6               // Source MAC address (48 Bit)
#define ETH_TYPE_OFS    12              // Type field (16 Bit)
#define ETH_DATA_OFS    14              // Δεδομένα πακέτου
#define ETH_HEADER_SIZE 14

#define FRAME_ARP       (0x0806)        // Τύπος Πακέτου
#define FRAME_IP        (0x0800)
```

```
// IPv4 layer definitions
#define IP_VER_IHL_TOS_OFS   (ETH_DATA_OFS + 0)  // Version, Header Length, Type of Service
#define IP_TOTAL_LENGTH_OFS  (ETH_DATA_OFS + 2)  // IP Frame's Total Length
#define IP_IDENT_OFS         (ETH_DATA_OFS + 4)  // Identifying Value
#define IP_FLAGS_FRAG_OFS    (ETH_DATA_OFS + 6)  // Flags and Fragment Offset
#define IP_TTL_PROT_OFS      (ETH_DATA_OFS + 8)  // Frame's Time to Live, Protocol
#define IP_HEAD_CHKSUM_OFS   (ETH_DATA_OFS + 10) // IP Frame's Header Checksum
#define IP_SOURCE_OFS        (ETH_DATA_OFS + 12) // Source Address (32 Bit)
#define IP_DESTINATION_OFS   (ETH_DATA_OFS + 16) // Destination Address (32 Bit)
#define IP_DATA_OFS          (ETH_DATA_OFS + 20) // Frame Data (if no options)
#define IP_HEADER_SIZE       20                  // w/o options

#define IP_VER_IHL           (0x4500)            // IPv4, Header Length = 5x32 bit
#define IP_TOS_D             (0x0010)            // TOS low delay
#define IP_TOS_T             (0x0008)            // TOS high throughput
#define IP_TOS_R             (0x0004)

#define IP_FLAG_DONTFRAG     (0x4000)
#define IP_FLAG_MOREFRAG     (0x2000)
#define IP_FRAGOFS_MASK      (0x1fff)

#define PROT_ICMP            1                   // Internet Control Message Protocol
#define PROT_TCP             6                   // Transmission Control Protocol
#define PROT_UDP             17                  // User Datagram Protocol

// ARP definitions
#define ARP_HARDW_OFS        (ETH_DATA_OFS + 0)  // Hardware address type
#define ARP_PROT_OFS         (ETH_DATA_OFS + 2)  // Protocol
#define ARP_HLEN_PLEN_OFS    (ETH_DATA_OFS + 4)
#define ARP_OPCODE_OFS       (ETH_DATA_OFS + 6)
#define ARP_SENDER_HA_OFS    (ETH_DATA_OFS + 8)
#define ARP_SENDER_IP_OFS    (ETH_DATA_OFS + 14) // IP address αποστολέα
#define ARP_TARGET_HA_OFS    (ETH_DATA_OFS + 18)
#define ARP_TARGET_IP_OFS    (ETH_DATA_OFS + 24)
#define ARP_FRAME_SIZE       28

#define HARDW_ETH10          1
#define IP_HLEN_PLEN         (0x0604)            // MAC = 6 byte long, IP = 4 byte long
#define OP_ARP_REQUEST       1                   #define OP_ARP_ANSWER        2

// ICMP definitions
#define ICMP_TYPE_CODE_OFS   (IP_DATA_OFS + 0)   // τύπος μηνύματος
#define ICMP_CHKSUM_OFS      (IP_DATA_OFS + 2)
#define ICMP_DATA_OFS        (IP_DATA_OFS + 4)
#define ICMP_HEADER_SIZE     4

#define ICMP_ECHO            8
#define ICMP_ECHO_REPLY      0
```

```
// TCP layer definitions
#define TCP_SRCPORT_OFS      (IP_DATA_OFS + 0)         // Source Port (16 bit)
#define TCP_DESTPORT_OFS     (IP_DATA_OFS + 2)         // Destination Port (16 bit)
#define TCP_SEQNR_OFS        (IP_DATA_OFS + 4)         // Sequence Number (32 bit)
#define TCP_ACKNR_OFS        (IP_DATA_OFS + 8)         // Acknowledge Number (32 bit)
#define TCP_DATA_CODE_OFS    (IP_DATA_OFS + 12)   // Data Offset and Control Bits (16 bit)
#define TCP_WINDOW_OFS       (IP_DATA_OFS + 14)        // Window Size (16 bit)
#define TCP_CHKSUM_OFS       (IP_DATA_OFS + 16)        // Checksum Field (16 bit)
#define TCP_URGENT_OFS       (IP_DATA_OFS + 18)        // Urgent Pointer (16 bit)
#define TCP_DATA_OFS         (IP_DATA_OFS + 20)        // Frame Data (if no options)
#define TCP_HEADER_SIZE      20

#define DATA_OFS_MASK        (0xf000)

#define TCP_CODE_FIN         (0x0001)
#define TCP_CODE_SYN         (0x0002)
#define TCP_CODE_RST         (0x0004)
#define TCP_CODE_PSH         (0x0008)
#define TCP_CODE_ACK         (0x0010)
#define TCP_CODE_URG         (0x0020)

#define TCP_OPT_MSS          (0x0204)          )
#define TCP_OPT_MSS_SIZE     4

// define some TCP standard-ports, useful for testing...
#define TCP_PORT_ECHO        7
#define TCP_PORT_DISCARD     9
#define TCP_PORT_DAYTIME     13
#define TCP_PORT_QOTD        17
#define TCP_PORT_CHARGEN     19
#define TCP_PORT_HTTP        80                        // word wide web HTTP

#define ACCESS_UINT(Base, Offset)   (*(unsigned int *)((unsigned char *)&(Base) + (Offset)))

#define SWAPB(Word)              ((unsigned int)((Word) << 8) | ((Word) >> 8))

// typedefs
typedef enum
{
  CLOSED,
  LISTENING,
  SYN_SENT,
  SYN_RECD,
  ESTABLISHED,
  FIN_WAIT_1,
  FIN_WAIT_2,
  CLOSE_WAIT,
  CLOSING,
  LAST_ACK,
  TIME_WAIT
} TTCPStateMachine;
```

```
typedef enum
{
  ARP_REQUEST,
  TCP_SYN_FRAME,
  TCP_SYN_ACK_FRAME,
  TCP_FIN_FRAME,
  TCP_DATA_FRAME
} TLastFrameSent;

// definitions for 'TransmitControl'
#define SEND_FRAME1               (0x01)
#define SEND_FRAME2               (0x02)

// definitions for 'TCPFlags'
#define TCP_ACTIVE_OPEN           (0x01)        // easyWEB θα αρχικοποιήση μία σύνδεση
#define IP_ADDR_RESOLVED          (0x02)
#define TCP_TIMER_RUNNING         (0x04)
#define TIMER_TYPE_RETRY          (0x08)
#define TCP_CLOSE_REQUESTED       (0x10)

// definitions for 'SocketStatus'
#define SOCK_ACTIVE               (0x01)
#define SOCK_CONNECTED            (0x02)        // ο χρήστης μπορεί να κάνει λήψη και αποστολή
δεδομένων
#define SOCK_DATA_AVAILABLE       (0x04)   // νέα δεδομένα διαθέσιμα
#define SOCK_TX_BUF_RELEASED      (0x08)   // ο χρήστη μπορεί να γεμίσει το buffer
#define SOCK_ERROR_MASK           (0xf0)
#define SOCK_ERR_OK               (0x00)
#define SOCK_ERR_ARP_TIMEOUT      (0x10)
#define SOCK_ERR_TCP_TIMEOUT      (0x20)
#define SOCK_ERR_CONN_RESET       (0x30)   // η σύνδεση δέχτηκε επανεκκίνηση από άλλο TCP
#define SOCK_ERR_REMOTE           (0x40)
#define SOCK_ERR_ETHERNET         (0x50)

// exported functions
// easyWEB-API functions
void TCPLowLevelInit(void);
void TCPPassiveOpen(void);                // αναμονή για σύνδεση
void TCPActiveOpen(void);                 // άνοιγμα σύνδεσης
void TCPClose(void);                      // κλείσιμο σύνδεσης
void TCPReleaseRxBuffer(void);
void TCPTransmitTxBuffer(void);
void DoNetworkStuff(void);

// exported constants
extern const unsigned int MyIP[];         // τοπική διεύθυνση IP
extern const unsigned int SubnetMask[];   // subnet mask (outbount connections)
extern const unsigned int GatewayIP[];    // gateway IP addr (outbount connections)

// exported variables
// easyWEB-API global vars and flags
extern unsigned char SocketStatus;        // API status variable
extern unsigned int TCPLocalPort;         // TCP ports
```

```
extern unsigned int TCPRemotePort;
extern unsigned int RemoteMAC[3];          // MAC διεύθυνση του TCP-session
extern unsigned int RemoteIP[2];           // IP address of current TCP-session
extern unsigned int TCPRxDataCount;
extern unsigned int TCPTxDataCount;
extern unsigned int TxFrame1Mem[];
extern unsigned int RxTCPBufferMem[];          // δεδομένα ληφθέντος TCP segment

// easyWEB-API TCP data buffer-pointers
#define TCP_TX_BUF     ((unsigned char *)TxFrame1Mem + ETH_HEADER_SIZE + \
                IP_HEADER_SIZE + TCP_HEADER_SIZE)
#define TCP_RX_BUF     ((unsigned char *)RxTCPBufferMem)

#endif

/*************************************************************
MSP430 microcontroller.
Easyweb Test
Αρχικοποίηση MSP430F149IPMG4
*************************************************************/

#ifndef __msp430x14x
#define __msp430x14x

/************************************************************
* STANDARD BITS
************************************************************/

#define BIT0            0x0001
#define BIT1            0x0002
#define BIT2            0x0004
#define BIT3            0x0008
#define BIT4            0x0010
#define BIT5            0x0020
#define BIT6            0x0040
#define BIT7            0x0080
#define BIT8            0x0100
#define BIT9            0x0200
#define BITA            0x0400
#define BITB            0x0800
#define BITC            0x1000
#define BITD            0x2000
#define BITE            0x4000
#define BITF            0x8000

/************************************************************
* STATUS REGISTER BITS
************************************************************/

#define C               0x0001
#define Z               0x0002
#define N               0x0004
```

```
#define V               0x0100
#define GIE             0x0008
#define CPUOFF          0x0010
#define OSCOFF          0x0020
#define SCG0            0x0040
#define SCG1            0x0080

/* Low Power Modes coded with Bits 4-7 in SR */

#ifndef __IAR_SYSTEMS_ICC /* Begin #defines for assembler */
#define LPM0            CPUOFF
#define LPM1            SCG0+CPUOFF
#define LPM2            SCG1+CPUOFF
#define LPM3            SCG1+SCG0+CPUOFF
#define LPM4            SCG1+SCG0+OSCOFF+CPUOFF
/* End #defines for assembler */

#else /* Begin #defines for C */
#define LPM0_bits       CPUOFF
#define LPM1_bits       SCG0+CPUOFF
#define LPM2_bits       SCG1+CPUOFF
#define LPM3_bits       SCG1+SCG0+CPUOFF
#define LPM4_bits       SCG1+SCG0+OSCOFF+CPUOFF

#include "In430.h"

#define LPM0     _BIS_SR(LPM0_bits) /* Enter Low Power Mode 0 */
#define LPM0_EXIT _BIC_SR(LPM0_bits) /* Exit Low Power Mode 0 */
#define LPM1     _BIS_SR(LPM1_bits) /* Enter Low Power Mode 1 */
#define LPM1_EXIT _BIC_SR(LPM1_bits) /* Exit Low Power Mode 1 */
#define LPM2     _BIS_SR(LPM2_bits) /* Enter Low Power Mode 2 */
#define LPM2_EXIT _BIC_SR(LPM2_bits) /* Exit Low Power Mode 2 */
#define LPM3     _BIS_SR(LPM3_bits) /* Enter Low Power Mode 3 */
#define LPM3_EXIT _BIC_SR(LPM3_bits) /* Exit Low Power Mode 3 */
#define LPM4     _BIS_SR(LPM4_bits) /* Enter Low Power Mode 4 */
#define LPM4_EXIT _BIC_SR(LPM4_bits) /* Exit Low Power Mode 4 */
#endif /* End #defines for C */
```

```
/************************************************************
* PERIPHERAL FILE MAP
************************************************************/
/************************************************************
* SPECIAL FUNCTION REGISTER ADDRESSES + CONTROL BITS
************************************************************/

#define IE1_            0x0000  /* Interrupt Enable 1 */
sfrb    IE1           = IE1_;
#define WDTIE            0x01
#define OFIE             0x02
#define NMIIE            0x10
#define ACCVIE           0x20
#define URXIE0           0x40
#define UTXIE0           0x80

#define IFG1_           0x0002  /* Interrupt Flag 1 */
sfrb    IFG1          = IFG1_;
#define WDTIFG           0x01
#define OFIFG            0x02
#define NMIIFG           0x10
#define URXIFG0          0x40
#define UTXIFG0          0x80

#define ME1_            0x0004  /* Module Enable 1 */
sfrb    ME1           = ME1_;
#define URXE0            0x40
#define USPIE0           0x40
#define UTXE0            0x80

#define IE2_            0x0001  /* Interrupt Enable 2 */
sfrb    IE2           = IE2_;
#define URXIE1           0x10
#define UTXIE1           0x20

#define IFG2_           0x0003  /* Interrupt Flag 2 */
sfrb    IFG2          = IFG2_;
#define URXIFG1          0x10
#define UTXIFG1          0x20

#define ME2_            0x0005  /* Module Enable 2 */
sfrb    ME2           = ME2_;
#define URXE1            0x10
#define USPIE1           0x10
#define UTXE1            0x20
```

```
/************************************************************
* WATCHDOG TIMER
************************************************************/

#define WDTCTL_           0x0120  /* Watchdog Timer Control */
sfrw    WDTCTL          = WDTCTL_;
/* The bit names have been prefixed with "WDT" */
#define WDTIS0            0x0001
#define WDTIS1            0x0002
#define WDTSSEL           0x0004
#define WDTCNTCL          0x0008
#define WDTTMSEL          0x0010
#define WDTNMI            0x0020
#define WDTNMIES          0x0040
#define WDTHOLD           0x0080

#define WDTPW             0x5A00

/* WDT-interval times [1ms] coded with Bits 0-2 */
/* WDT is clocked by fMCLK (assumed 1MHz) */
#define WDT_MDLY_32       WDTPW+WDTTMSEL+WDTCNTCL                 /* 32ms interval
(default) */
#define WDT_MDLY_8        WDTPW+WDTTMSEL+WDTCNTCL+WDTIS0          /* 8ms     " */
#define WDT_MDLY_0_5      WDTPW+WDTTMSEL+WDTCNTCL+WDTIS1          /* 0.5ms   "
*/
#define WDT_MDLY_0_064    WDTPW+WDTTMSEL+WDTCNTCL+WDTIS1+WDTIS0   /*
0.064ms " */
/* WDT is clocked by fACLK (assumed 32KHz) */
#define WDT_ADLY_1000     WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL         /* 1000ms
" */
#define WDT_ADLY_250      WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL+WDTIS0  /*
250ms   " */
#define WDT_ADLY_16       WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL+WDTIS1  /*
16ms    " */
#define WDT_ADLY_1_9      WDTPW+WDTTMSEL+WDTCNTCL+WDTSSEL+WDTIS1+WDTIS0
/* 1.9ms   " */
/* Watchdog mode -> reset after expired time */
/* WDT is clocked by fMCLK (assumed 1MHz) */
#define WDT_MRST_32       WDTPW+WDTCNTCL                  /* 32ms interval (default) */
#define WDT_MRST_8        WDTPW+WDTCNTCL+WDTIS0           /* 8ms     " */
#define WDT_MRST_0_5      WDTPW+WDTCNTCL+WDTIS1           /* 0.5ms   " */
#define WDT_MRST_0_064    WDTPW+WDTCNTCL+WDTIS1+WDTIS0    /* 0.064ms " */
/* WDT is clocked by fACLK (assumed 32KHz) */
#define WDT_ARST_1000     WDTPW+WDTCNTCL+WDTSSEL          /* 1000ms  " */
#define WDT_ARST_250      WDTPW+WDTCNTCL+WDTSSEL+WDTIS0   /* 250ms   " */
#define WDT_ARST_16       WDTPW+WDTCNTCL+WDTSSEL+WDTIS1   /* 16ms    " */
#define WDT_ARST_1_9      WDTPW+WDTCNTCL+WDTSSEL+WDTIS1+WDTIS0    /* 1.9ms
" */

/* INTERRUPT CONTROL */
/* These two bits are defined in the Special Function Registers */
/* #define WDTIE             0x01 */
/* #define WDTIFG            0x01 */
```

```
/************************************************************
* HARDWARE MULTIPLIER
************************************************************/

#define MPY_            0x0130  /* Multiply Unsigned/Operand 1 */
sfrw    MPY           = MPY_;
#define MPYS_           0x0132  /* Multiply Signed/OPerand 1 */
sfrw    MPYS          = MPYS_;
#define MAC_            0x0134  /* Multiply Unsigned and Accumulate/Operand 1 */
sfrw    MAC           = MAC_;
#define MACS_           0x0136  /* Multiply Signed and Accumulate/Operand 1 */
sfrw    MACS          = MACS_;
#define OP2_            0x0138  /* Operand 2 */
sfrw    OP2           = OP2_;
#define RESLO_          0x013A  /* Result Low Word */
sfrw    RESLO         = RESLO_;
#define RESHI_          0x013C  /* Result High Word */
sfrw    RESHI         = RESHI_;
#define SUMEXT_         0x013E  /* Sum Extend */
const sfrw SUMEXT       = SUMEXT_;

/************************************************************
* DIGITAL I/O Port1/2
************************************************************/
#define P1IN_           0x0020  /* Port 1 Input */
const sfrb P1IN         = P1IN_;
#define P1OUT_          0x0021  /* Port 1 Output */
sfrb    P1OUT         = P1OUT_;
#define P1DIR_          0x0022  /* Port 1 Direction */
sfrb    P1DIR         = P1DIR_;
#define P1IFG_          0x0023  /* Port 1 Interrupt Flag */
sfrb    P1IFG         = P1IFG_;
#define P1IES_          0x0024  /* Port 1 Interrupt Edge Select */
sfrb    P1IES         = P1IES_;
#define P1IE_           0x0025  /* Port 1 Interrupt Enable */
sfrb    P1IE          = P1IE_;
#define P1SEL_          0x0026  /* Port 1 Selection */
sfrb    P1SEL         = P1SEL_;

#define P2IN_           0x0028  /* Port 2 Input */
const sfrb P2IN         = P2IN_;
#define P2OUT_          0x0029  /* Port 2 Output */
sfrb    P2OUT         = P2OUT_;
#define P2DIR_          0x002A  /* Port 2 Direction */
sfrb    P2DIR         = P2DIR_;
#define P2IFG_          0x002B  /* Port 2 Interrupt Flag */
sfrb    P2IFG         = P2IFG_;
#define P2IES_          0x002C  /* Port 2 Interrupt Edge Select */
sfrb    P2IES         = P2IES_;
#define P2IE_           0x002D  /* Port 2 Interrupt Enable */
sfrb    P2IE          = P2IE_;
#define P2SEL_          0x002E  /* Port 2 Selection */
sfrb    P2SEL         = P2SEL_;
```

```
/************************************************************
* DIGITAL I/O Port3/4
************************************************************/

#define P3IN_            0x0018  /* Port 3 Input */
const sfrb P3IN         = P3IN_;
#define P3OUT_           0x0019  /* Port 3 Output */
sfrb    P3OUT           = P3OUT_;
#define P3DIR_           0x001A  /* Port 3 Direction */
sfrb    P3DIR           = P3DIR_;
#define P3SEL_           0x001B  /* Port 3 Selection */
sfrb    P3SEL           = P3SEL_;

#define P4IN_            0x001C  /* Port 4 Input */
const sfrb P4IN         = P4IN_;
#define P4OUT_           0x001D  /* Port 4 Output */
sfrb    P4OUT           = P4OUT_;
#define P4DIR_           0x001E  /* Port 4 Direction */
sfrb    P4DIR           = P4DIR_;
#define P4SEL_           0x001F  /* Port 4 Selection */
sfrb    P4SEL           = P4SEL_;

/************************************************************
* DIGITAL I/O Port5/6
************************************************************/

#define P5IN_            0x0030  /* Port 5 Input */
const sfrb P5IN         = P5IN_;
#define P5OUT_           0x0031  /* Port 5 Output */
sfrb    P5OUT           = P5OUT_;
#define P5DIR_           0x0032  /* Port 5 Direction */
sfrb    P5DIR           = P5DIR_;
#define P5SEL_           0x0033  /* Port 5 Selection */
sfrb    P5SEL           = P5SEL_;

#define P6IN_            0x0034  /* Port 6 Input */
const sfrb P6IN         = P6IN_;
#define P6OUT_           0x0035  /* Port 6 Output */
sfrb    P6OUT           = P6OUT_;
#define P6DIR_           0x0036  /* Port 6 Direction */
sfrb    P6DIR           = P6DIR_;
#define P6SEL_           0x0037  /* Port 6 Selection */
sfrb    P6SEL           = P6SEL_;
```

```
/************************************************************
* USART
************************************************************/

#define PENA            0x80      /* UCTL */
#define PEV             0x40
#define SPB             0x20      /* to distinguish from stackpointer SP */
#define CHAR            0x10
#define LISTEN          0x08
#define SYNC            0x04
#define MM              0x02
#define SWRST           0x01

#define CKPH            0x80      /* UTCTL */
#define CKPL            0x40
#define SSEL1           0x20
#define SSEL0           0x10
#define URXSE           0x08
#define TXWAKE          0x04
#define STC             0x02
#define TXEPT           0x01

#define FE              0x80      /* URCTL */
#define PE              0x40
#define OE              0x20
#define BRK             0x10
#define URXEIE          0x08
#define URXWIE          0x04
#define RXWAKE          0x02
#define RXERR           0x01

/************************************************************
* UART 0/1
************************************************************/

#define U0CTL_          0x0070  /* UART 0 Control */
sfrb    U0CTL           = U0CTL_;
#define U0TCTL_         0x0071  /* UART 0 Transmit Control */
sfrb    U0TCTL          = U0TCTL_;
#define U0RCTL_         0x0072  /* UART 0 Receive Control */
sfrb    U0RCTL          = U0RCTL_;
#define U0MCTL_         0x0073  /* UART 0 Modulation Control */
sfrb    U0MCTL          = U0MCTL_;
#define U0BR0_          0x0074  /* UART 0 Baud Rate 0 */
sfrb    U0BR0           = U0BR0_;
#define U0BR1_          0x0075  /* UART 0 Baud Rate 1 */
sfrb    U0BR1           = U0BR1_;
#define U0RXBUF_        0x0076  /* UART 0 Receive Buffer */
const sfrb U0RXBUF      = U0RXBUF_;
#define U0TXBUF_        0x0077  /* UART 0 Transmit Buffer */
sfrb    U0TXBUF         = U0TXBUF_;

#define U1CTL_          0x0078  /* UART 1 Control */
```

```
sfrb    U1CTL           = U1CTL_;
#define U1TCTL_          0x0079  /* UART 1 Transmit Control */
sfrb    U1TCTL          = U1TCTL_;
#define U1RCTL_          0x007A  /* UART 1 Receive Control */
sfrb    U1RCTL          = U1RCTL_;
#define U1MCTL_          0x007B  /* UART 1 Modulation Control */
sfrb    U1MCTL          = U1MCTL_;
#define U1BR0_           0x007C  /* UART 1 Baud Rate 0 */
sfrb    U1BR0           = U1BR0_;
#define U1BR1_           0x007D  /* UART 1 Baud Rate 1 */
sfrb    U1BR1           = U1BR1_;
#define U1RXBUF_         0x007E  /* UART 1 Receive Buffer */
const sfrb U1RXBUF      = U1RXBUF_;
#define U1TXBUF_         0x007F  /* UART 1 Transmit Buffer */
sfrb    U1TXBUF         = U1TXBUF_;

/* Alternate register names */

#define UCTL0_           0x0070  /* UART 0 Control */
sfrb    UCTL0           = UCTL0_;
#define UTCTL0_          0x0071  /* UART 0 Transmit Control */
sfrb    UTCTL0          = UTCTL0_;
#define URCTL0_          0x0072  /* UART 0 Receive Control */
sfrb    URCTL0          = URCTL0_;
#define UMCTL0_          0x0073  /* UART 0 Modulation Control */
sfrb    UMCTL0          = UMCTL0_;
#define UBR00_           0x0074  /* UART 0 Baud Rate 0 */
sfrb    UBR00           = UBR00_;
#define UBR10_           0x0075  /* UART 0 Baud Rate 1 */
sfrb    UBR10           = UBR10_;
#define RXBUF0_          0x0076  /* UART 0 Receive Buffer */
const sfrb RXBUF0       = RXBUF0_;
#define TXBUF0_          0x0077  /* UART 0 Transmit Buffer */
sfrb    TXBUF0          = TXBUF0_;

#define UCTL1_           0x0078  /* UART 1 Control */
sfrb    UCTL1           = UCTL1_;
#define UTCTL1_          0x0079  /* UART 1 Transmit Control */
sfrb    UTCTL1          = UTCTL1_;
#define URCTL1_          0x007A  /* UART 1 Receive Control */
sfrb    URCTL1          = URCTL1_;
#define UMCTL1_          0x007B  /* UART 1 Modulation Control */
sfrb    UMCTL1          = UMCTL1_;
#define UBR01_           0x007C  /* UART 1 Baud Rate 0 */
sfrb    UBR01           = UBR01_;
#define UBR11_           0x007D  /* UART 1 Baud Rate 1 */
sfrb    UBR11           = UBR11_;
#define RXBUF1_          0x007E  /* UART 1 Receive Buffer */
const sfrb RXBUF1       = RXBUF1_;
#define TXBUF1_          0x007F  /* UART 1 Transmit Buffer */
sfrb    TXBUF1          = TXBUF1_;

#define UCTL_0_          0x0070  /* UART 0 Control */
```

```
sfrb    UCTL_0          = UCTL_0_;
#define UTCTL_0_          0x0071  /* UART 0 Transmit Control */
sfrb    UTCTL_0         = UTCTL_0_;
#define URCTL_0_          0x0072  /* UART 0 Receive Control */
sfrb    URCTL_0         = URCTL_0_;
#define UMCTL_0_          0x0073  /* UART 0 Modulation Control */
sfrb    UMCTL_0         = UMCTL_0_;
#define UBR0_0_           0x0074  /* UART 0 Baud Rate 0 */
sfrb    UBR0_0          = UBR0_0_;
#define UBR1_0_           0x0075  /* UART 0 Baud Rate 1 */
sfrb    UBR1_0          = UBR1_0_;
#define RXBUF_0_          0x0076  /* UART 0 Receive Buffer */
const sfrb RXBUF_0      = RXBUF_0_;
#define TXBUF_0_          0x0077  /* UART 0 Transmit Buffer */
sfrb    TXBUF_0         = TXBUF_0_;

#define UCTL_1_           0x0078  /* UART 1 Control */
sfrb    UCTL_1          = UCTL_1_;
#define UTCTL_1_          0x0079  /* UART 1 Transmit Control */
sfrb    UTCTL_1         = UTCTL_1_;
#define URCTL_1_          0x007A  /* UART 1 Receive Control */
sfrb    URCTL_1         = URCTL_1_;
#define UMCTL_1_          0x007B  /* UART 1 Modulation Control */
sfrb    UMCTL_1         = UMCTL_1_;
#define UBR0_1_           0x007C  /* UART 1 Baud Rate 0 */
sfrb    UBR0_1          = UBR0_1_;
#define UBR1_1_           0x007D  /* UART 1 Baud Rate 1 */
sfrb    UBR1_1          = UBR1_1_;
#define RXBUF_1_          0x007E  /* UART 1 Receive Buffer */
const sfrb RXBUF_1      = RXBUF_1_;
#define TXBUF_1_          0x007F  /* UART 1 Transmit Buffer */
sfrb    TXBUF_1         = TXBUF_1_;

/************************************************************
* Timer A
************************************************************/

#define TAIV_             0x012E  /* Timer A Interrupt Vector Word */
sfrw    TAIV            = TAIV_;
#define TACTL_            0x0160  /* Timer A Control */
sfrw    TACTL           = TACTL_;
#define CCTL0_            0x0162  /* Timer A Capture/Compare Control 0 */
sfrw    CCTL0           = CCTL0_;
#define CCTL1_            0x0164  /* Timer A Capture/Compare Control 1 */
sfrw    CCTL1           = CCTL1_;
#define CCTL2_            0x0166  /* Timer A Capture/Compare Control 2 */
sfrw    CCTL2           = CCTL2_;
#define TAR_              0x0170  /* Timer A */
sfrw    TAR             = TAR_;
#define CCR0_             0x0172  /* Timer A Capture/Compare 0 */
sfrw    CCR0            = CCR0_;
#define CCR1_             0x0174  /* Timer A Capture/Compare 1 */
sfrw    CCR1            = CCR1_;
```

```
#define CCR2_            0x0176  /* Timer A Capture/Compare 2 */
sfrw    CCR2           = CCR2_;

#define TASSEL2          0x0400       /* to distinguish from UART SSELx */
#define TASSEL1          0x0200
#define TASSEL0          0x0100
#define ID1             0x0080
#define ID0             0x0040
#define MC1              0x0020
#define MC0              0x0010
#define TACLR             0x0004
#define TAIE             0x0002
#define TAIFG             0x0001

#define MC_0             00*0x10
#define MC_1             01*0x10
#define MC_2             02*0x10
#define MC_3             03*0x10
#define ID_0            00*0x40
#define ID_1            01*0x40
#define ID_2            02*0x40
#define ID_3            03*0x40
#define TASSEL_0         00*0x100
#define TASSEL_1         01*0x100
#define TASSEL_2         02*0x100
#define TASSEL_3         03*0x100

#define CM1              0x8000
#define CM0              0x4000
#define CCIS1            0x2000
#define CCIS0            0x1000
#define SCS             0x0800
#define SCCI             0x0400
#define CAP             0x0100
#define OUTMOD2          0x0080
#define OUTMOD1          0x0040
#define OUTMOD0          0x0020
#define CCIE             0x0010
#define CCI             0x0008
#define OUT              0x0004
#define COV              0x0002
#define CCIFG             0x0001

#define OUTMOD_0         00*0x20
#define OUTMOD_1         01*0x20
#define OUTMOD_2         02*0x20
#define OUTMOD_3         03*0x20
#define OUTMOD_4         04*0x20
#define OUTMOD_5         05*0x20
#define OUTMOD_6         06*0x20
#define OUTMOD_7         07*0x20
#define CCIS_0          00*0x1000
#define CCIS_1          01*0x1000
```

```
#define CCIS_2          02*0x1000
#define CCIS_3          03*0x1000
#define CM_0            00*0x4000
#define CM_1            01*0x4000
#define CM_2            02*0x4000
#define CM_3            03*0x4000

/************************************************************
* Timer B
************************************************************/

#define TBIV_           0x011E  /* Timer B Interrupt Vector Word */
sfrw    TBIV          = TBIV_;
#define TBCTL_          0x0180  /* Timer B Control */
sfrw    TBCTL         = TBCTL_;
#define TBCCTL0_        0x0182  /* Timer B Capture/Compare Control 0 */
sfrw    TBCCTL0       = TBCCTL0_;
#define TBCCTL1_        0x0184  /* Timer B Capture/Compare Control 1 */
sfrw    TBCCTL1       = TBCCTL1_;
#define TBCCTL2_        0x0186  /* Timer B Capture/Compare Control 2 */
sfrw    TBCCTL2       = TBCCTL2_;
#define TBCCTL3_        0x0188  /* Timer B Capture/Compare Control 3 */
sfrw    TBCCTL3       = TBCCTL3_;
#define TBCCTL4_        0x018A  /* Timer B Capture/Compare Control 4 */
sfrw    TBCCTL4       = TBCCTL4_;
#define TBCCTL5_        0x018C  /* Timer B Capture/Compare Control 5 */
sfrw    TBCCTL5       = TBCCTL5_;
#define TBCCTL6_        0x018E  /* Timer B Capture/Compare Control 6 */
sfrw    TBCCTL6       = TBCCTL6_;
#define TBR_            0x0190  /* Timer B */
sfrw    TBR           = TBR_;
#define TBCCR0_         0x0192  /* Timer B Capture/Compare 0 */
sfrw    TBCCR0        = TBCCR0_;
#define TBCCR1_         0x0194  /* Timer B Capture/Compare 1 */
sfrw    TBCCR1        = TBCCR1_;
#define TBCCR2_         0x0196  /* Timer B Capture/Compare 2 */
sfrw    TBCCR2        = TBCCR2_;
#define TBCCR3_         0x0198  /* Timer B Capture/Compare 3 */
sfrw    TBCCR3        = TBCCR3_;
#define TBCCR4_         0x019A  /* Timer B Capture/Compare 4 */
sfrw    TBCCR4        = TBCCR4_;
#define TBCCR5_         0x019C  /* Timer B Capture/Compare 5 */
sfrw    TBCCR5        = TBCCR5_;
#define TBCCR6_         0x019E  /* Timer B Capture/Compare 6 */
sfrw    TBCCR6        = TBCCR6_;

#define SHR1            0x4000
#define SHR0            0x2000
#define CNTL1           0x1000
#define CNTL0           0x0800
#define TBSSEL2         0x0400
#define TBSSEL1         0x0200
#define TBSSEL0         0x0100
```

```
#define TBCLR            0x0004
#define TBIE            0x0002
#define TBIFG            0x0001

#define TBSSEL_0         00*0x0100
#define TBSSEL_1         01*0x0100
#define TBSSEL_2         02*0x0100
#define TBSSEL_3         03*0x0100
#define CNTL_0           00*0x0800
#define CNTL_1           01*0x0800
#define CNTL_2           02*0x0800
#define CNTL_3           03*0x0800
#define SHR_0            00*0x2000
#define SHR_1            01*0x2000
#define SHR_2            02*0x2000
#define SHR_3            03*0x2000

#define SLSHR1           0x0400
#define SLSHR0           0x0200

#define SLSHR_0          00*0x0200
#define SLSHR_1          01*0x0200
#define SLSHR_2          02*0x0200
#define SLSHR_3          03*0x0200

/************************************************************
* Basic Clock Module
************************************************************/

#define DCOCTL_          0x0056  /* DCO Clock Frequency Control */
sfrb    DCOCTL         = DCOCTL_;
#define BCSCTL1_         0x0057  /* Basic Clock System Control 1 */
sfrb    BCSCTL1        = BCSCTL1_;
#define BCSCTL2_         0x0058  /* Basic Clock System Control 2 */
sfrb    BCSCTL2        = BCSCTL2_;

#define MOD0             0x01
#define MOD1             0x02
#define MOD2             0x04
#define MOD3             0x08
#define MOD4             0x10
#define DCO0             0x20
#define DCO1             0x40
#define DCO2             0x80

#define RSEL0            0x01
#define RSEL1            0x02
#define RSEL2            0x04
#define XT5V             0x08
#define DIVA0            0x10
#define DIVA1            0x20
#define XTS             0x40
#define XTOFF            0x80
```

```
#define DCOR            0x01
#define DIVS0           0x02
#define DIVS1           0x04
#define SELS            0x08
#define DIVM0           0x10
#define DIVM1           0x20
#define SELM0           0x40
#define SELM1           0x80

/************************************************************
* Flash Memory
************************************************************/

#define FCTL1_          0x0128  /* FLASH Control 1 */
sfrw    FCTL1         = FCTL1_;
#define FCTL2_          0x012A  /* FLASH Control 2 */
sfrw    FCTL2         = FCTL2_;
#define FCTL3_          0x012C  /* FLASH Control 3 */
sfrw    FCTL3         = FCTL3_;

#define FRKEY           0x9600
#define FWKEY           0xA500
#define FXKEY           0x3300      /* for use with XOR instruction */

#define ERASE           0x0002
#define MERAS           0x0004
#define WRT             0x0040
#define SEGWRT          0x0080

#define FN0             0x0001
#define FN1             0x0002
#define FN2             0x0004
#define FN3             0x0008
#define FN4             0x0010
#define FN5             0x0020
#define FSSEL0          0x0040      /* to distinguish from UART SSELx */
#define FSSEL1          0x0080

#define BUSY            0x0001
#define KEYV            0x0002
#define ACCVIFG         0x0004
#define WAIT            0x0008
#define LOCK            0x0010
#define EMEX            0x0020
```

```
/************************************************************
* Comparator A
************************************************************/

#define CACTL1_          0x0059  /* Comparator A Control 1 */
sfrb    CACTL1         = CACTL1_;
#define CACTL2_          0x005A  /* Comparator A Control 2 */
sfrb    CACTL2         = CACTL2_;
#define CAPD_            0x005B  /* Comparator A Port Disable */
sfrb    CAPD           = CAPD_;

#define CAIFG            0x01
#define CAIE             0x02
#define CAIES            0x04
#define CAON             0x08
#define CAREF0           0x10
#define CAREF1           0x20
#define CARSEL           0x40
#define CAEX             0x80
#define CAOUT            0x01
#define CAF              0x02
#define P2CA0            0x04
#define P2CA1            0x08
#define CACTL24          0x10
#define CACTL25          0x20
#define CACTL26          0x40
#define CACTL27          0x80

#define CAPD0            0x01
#define CAPD1            0x02
#define CAPD2            0x04
#define CAPD3            0x08
#define CAPD4            0x10
#define CAPD5            0x20
#define CAPD6            0x40
#define CAPD7            0x80

/ADC12/
#define ADC12CTL0_       0x01A0  /* ADC12 Control 0 */
sfrw    ADC12CTL0      = ADC12CTL0_;
#define ADC12CTL1_       0x01A2  /* ADC12 Control 1 */
sfrw    ADC12CTL1      = ADC12CTL1_;
#define ADC12IFG_        0x01A4  /* ADC12 Interrupt Flag */
sfrw    ADC12IFG       = ADC12IFG_;
#define ADC12IE_         0x01A6  /* ADC12 Interrupt Enable */
sfrw    ADC12IE        = ADC12IE_;
#define ADC12IV_         0x01A8  /* ADC12 Interrupt Vector Word */
sfrw    ADC12IV        = ADC12IV_;

#define ADC12MEM_        0x0140  /* ADC12 Conversion Memory */
#ifndef __IAR_SYSTEMS_ICC
#define ADC12MEM         ADC12MEM_ /* ADC12 Conversion Memory (for assembler) */
#else
```

```
#define ADC12MEM          ((int*) ADC12MEM_) /* ADC12 Conversion Memory (for C) */
#endif
#define ADC12MEM0_         ADC12MEM_ /* ADC12 Conversion Memory 0 */
sfrw    ADC12MEM0       = ADC12MEM0_;
#define ADC12MEM1_         0x0142  /* ADC12 Conversion Memory 1 */
sfrw    ADC12MEM1       = ADC12MEM1_;
#define ADC12MEM2_         0x0144  /* ADC12 Conversion Memory 2 */
sfrw    ADC12MEM2       = ADC12MEM2_;
#define ADC12MEM3_         0x0146  /* ADC12 Conversion Memory 3 */
sfrw    ADC12MEM3       = ADC12MEM3_;
#define ADC12MEM4_         0x0148  /* ADC12 Conversion Memory 4 */
sfrw    ADC12MEM4       = ADC12MEM4_;
#define ADC12MEM5_         0x014A  /* ADC12 Conversion Memory 5 */
sfrw    ADC12MEM5       = ADC12MEM5_;
#define ADC12MEM6_         0x014C  /* ADC12 Conversion Memory 6 */
sfrw    ADC12MEM6       = ADC12MEM6_;
#define ADC12MEM7_         0x014E  /* ADC12 Conversion Memory 7 */
sfrw    ADC12MEM7       = ADC12MEM7_;
#define ADC12MEM8_         0x0150  /* ADC12 Conversion Memory 8 */
sfrw    ADC12MEM8       = ADC12MEM8_;
#define ADC12MEM9_         0x0152  /* ADC12 Conversion Memory 9 */
sfrw    ADC12MEM9       = ADC12MEM9_;
#define ADC12MEM10_        0x0154  /* ADC12 Conversion Memory 10 */
sfrw    ADC12MEM10       = ADC12MEM10_;
#define ADC12MEM11_        0x0156  /* ADC12 Conversion Memory 11 */
sfrw    ADC12MEM11       = ADC12MEM11_;
#define ADC12MEM12_        0x0158  /* ADC12 Conversion Memory 12 */
sfrw    ADC12MEM12       = ADC12MEM12_;
#define ADC12MEM13_        0x015A  /* ADC12 Conversion Memory 13 */
sfrw    ADC12MEM13       = ADC12MEM13_;
#define ADC12MEM14_        0x015C  /* ADC12 Conversion Memory 14 */
sfrw    ADC12MEM14       = ADC12MEM14_;
#define ADC12MEM15_        0x015E  /* ADC12 Conversion Memory 15 */
sfrw    ADC12MEM15       = ADC12MEM15_;

#define ADC12MCTL_         0x0080  /* ADC12 Memory Control */
#ifndef __IAR_SYSTEMS_ICC
#define ADC12MCTL          ADC12MCTL_ /* ADC12 Memory Control (for assembler) */
#else
#define ADC12MCTL          ((char*) ADC12MCTL_) /* ADC12 Memory Control (for C) */
#endif
#define ADC12MCTL0_        ADC12MCTL_ /* ADC12 Memory Control 0 */
sfrb    ADC12MCTL0      = ADC12MCTL0_;
#define ADC12MCTL1_        0x0081  /* ADC12 Memory Control 1 */
sfrb    ADC12MCTL1      = ADC12MCTL1_;
#define ADC12MCTL2_        0x0082  /* ADC12 Memory Control 2 */
sfrb    ADC12MCTL2      = ADC12MCTL2_;
#define ADC12MCTL3_        0x0083  /* ADC12 Memory Control 3 */
sfrb    ADC12MCTL3      = ADC12MCTL3_;
#define ADC12MCTL4_        0x0084  /* ADC12 Memory Control 4 */
sfrb    ADC12MCTL4      = ADC12MCTL4_;
#define ADC12MCTL5_        0x0085  /* ADC12 Memory Control 5 */
sfrb    ADC12MCTL5      = ADC12MCTL5_;
```

```
#define ADC12MCTL6_       0x0086  /* ADC12 Memory Control 6 */
sfrb    ADC12MCTL6      = ADC12MCTL6_;
#define ADC12MCTL7_       0x0087  /* ADC12 Memory Control 7 */
sfrb    ADC12MCTL7      = ADC12MCTL7_;
#define ADC12MCTL8_       0x0088  /* ADC12 Memory Control 8 */
sfrb    ADC12MCTL8      = ADC12MCTL8_;
#define ADC12MCTL9_       0x0089  /* ADC12 Memory Control 9 */
sfrb    ADC12MCTL9      = ADC12MCTL9_;
#define ADC12MCTL10_      0x008A  /* ADC12 Memory Control 10 */
sfrb    ADC12MCTL10     = ADC12MCTL10_;
#define ADC12MCTL11_      0x008B  /* ADC12 Memory Control 11 */
sfrb    ADC12MCTL11     = ADC12MCTL11_;
#define ADC12MCTL12_      0x008C  /* ADC12 Memory Control 12 */
sfrb    ADC12MCTL12     = ADC12MCTL12_;
#define ADC12MCTL13_      0x008D  /* ADC12 Memory Control 13 */
sfrb    ADC12MCTL13     = ADC12MCTL13_;
#define ADC12MCTL14_      0x008E  /* ADC12 Memory Control 14 */
sfrb    ADC12MCTL14     = ADC12MCTL14_;
#define ADC12MCTL15_      0x008F  /* ADC12 Memory Control 15 */
sfrb    ADC12MCTL15     = ADC12MCTL15_;

#define ADC12SC          0x001 /* ADC12CTL0 */
#define ENC             0x002
#define ADC12TOVIE       0x004
#define ADC12OVIE        0x008
#define ADC12ON          0x010
#define REFON            0x020
#define REF2_5V          0x040
#define MSH             0x080

#define SHT0_0           00*0x100
#define SHT0_1           01*0x100
#define SHT0_2           02*0x100
#define SHT0_3           03*0x100
#define SHT0_4           04*0x100
#define SHT0_5           05*0x100
#define SHT0_6           06*0x100
#define SHT0_7           07*0x100
#define SHT0_8           08*0x100
#define SHT0_9           09*0x100
#define SHT0_10          10*0x100
#define SHT0_11          11*0x100
#define SHT0_12          12*0x100
#define SHT0_13          13*0x100
#define SHT0_14          14*0x100
#define SHT0_15          15*0x100

#define SHT1_0           00*0x1000
#define SHT1_1           01*0x1000
#define SHT1_2           02*0x1000
#define SHT1_3           03*0x1000
#define SHT1_4           04*0x1000
#define SHT1_5           05*0x1000
```

```
#define SHT1_6          06*0x1000
#define SHT1_7          07*0x1000
#define SHT1_8          08*0x1000
#define SHT1_9          09*0x1000
#define SHT1_10          10*0x1000
#define SHT1_11          11*0x1000
#define SHT1_12          12*0x1000
#define SHT1_13          13*0x1000
#define SHT1_14          14*0x1000
#define SHT1_15          15*0x1000

#define ADC12BUSY         0x0001 /* ADC12CTL1 */
#define CONSEQ_0         00*2
#define CONSEQ_1         01*2
#define CONSEQ_2         02*2
#define CONSEQ_3         03*2
#define ADC12SSEL_0       00*8
#define ADC12SSEL_1       01*8
#define ADC12SSEL_2       02*8
#define ADC12SSEL_3       03*8
#define ADC12DIV_0       00*0x20
#define ADC12DIV_1       01*0x20
#define ADC12DIV_2       02*0x20
#define ADC12DIV_3       03*0x20
#define ADC12DIV_4       04*0x20
#define ADC12DIV_5       05*0x20
#define ADC12DIV_6       06*0x20
#define ADC12DIV_7       07*0x20
#define ISSH            0x0100
#define SHP             0x0200
#define SHS_0            00*0x400
#define SHS_1            01*0x400
#define SHS_2            02*0x400
#define SHS_3            03*0x400

#define CSTARTADD_0      00*0x1000
#define CSTARTADD_1      01*0x1000
#define CSTARTADD_2      02*0x1000
#define CSTARTADD_3      03*0x1000
#define CSTARTADD_4      04*0x1000
#define CSTARTADD_5      05*0x1000
#define CSTARTADD_6      06*0x1000
#define CSTARTADD_7      07*0x1000
#define CSTARTADD_8      08*0x1000
#define CSTARTADD_9      09*0x1000
#define CSTARTADD_10      10*0x1000
#define CSTARTADD_11      11*0x1000
#define CSTARTADD_12      12*0x1000
#define CSTARTADD_13      13*0x1000
#define CSTARTADD_14      14*0x1000
#define CSTARTADD_15      15*0x1000

#define INCH_0           00 /* ADC12CTLx */
```

```
#define INCH_1          01
#define INCH_2          02
#define INCH_3          03
#define INCH_4          04
#define INCH_5          05
#define INCH_6          06
#define INCH_7          07
#define INCH_8          08
#define INCH_9          09
#define INCH_10          10
#define INCH_11          11
#define INCH_12          12
#define INCH_13          13
#define INCH_14          14
#define INCH_15          15
#define SREF_0          00*0x10
#define SREF_1          01*0x10
#define SREF_2          02*0x10
#define SREF_3          03*0x10
#define SREF_4          04*0x10
#define SREF_5          05*0x10
#define SREF_6          06*0x10
#define SREF_7          07*0x10
#define EOS            0x80

/Interrupt Vectors (offset from 0xFFE0)/
#define PORT2_VECTOR        1 * 2  /* 0xFFE2 Port 2 */
#define UART1TX_VECTOR      2 * 2  /* 0xFFE4 UART 1 Transmit */
#define UART1RX_VECTOR      3 * 2  /* 0xFFE6 UART 1 Receive */
#define PORT1_VECTOR        4 * 2  /* 0xFFE8 Port 1 */
#define TIMERA1_VECTOR      5 * 2  /* 0xFFEA Timer A CC1-2, TA */
#define TIMERA0_VECTOR      6 * 2  /* 0xFFEC Timer A CC0 */
#define ADC_VECTOR          7 * 2  /* 0xFFEE ADC */
#define UART0TX_VECTOR      8 * 2  /* 0xFFF0 UART 0 Transmit */
#define UART0RX_VECTOR      9 * 2  /* 0xFFF2 UART 0 Receive */
#define WDT_VECTOR          10 * 2 /* 0xFFF4 Watchdog Timer */
#define COMPARATORA_VECTOR  11 * 2 /* 0xFFF6 Comparator A */
#define TIMERB1_VECTOR      12 * 2 /* 0xFFF8 Timer B 1-7 */
#define TIMERB0_VECTOR      13 * 2 /* 0xFFFA Timer B 0 */
#define NMI_VECTOR          14 * 2 /* 0xFFFC Non-maskable */
#define RESET_VECTOR        15 * 2 /* 0xFFFE Reset [Highest Priority] */
/End of Modules/
#endif /* #ifndef __msp430x14x */
```

```
//-------------------------------------------------------------------------------
// Όνομα: support.c
//-------------------------------------------------------------------------------
#ifndef __SUPPORT_H
#define __SUPPORT_H

extern void DelayCycles(unsigned int Count);
extern void WriteDWBE(unsigned char *Add, unsigned long Data);
#endif
```

Support.s43

```
Exported Symbols
;-------------------------------------------------------------------------------
        PUBLIC  DelayCycles
        PUBLIC  WriteDWBE
;-------------------------------------------------------------------------------
; Implementation
;-------------------------------------------------------------------------------
        RSEG    CODE
;-------------------------------------------------------------------------------
; DelayCycles
;
; Delays the program flow by the given number of cycles (R12). The parameter
; MUST be >= 20 (decimal).
;
; IN:  R12
;-------------------------------------------------------------------------------
DelayCycles                             ; call #DelayCycles (5 cyc)
        push    R12                     ; Save R12 (3 cyc)
        sub.w   #17,R12                 ; Adjust cycle count (2 cyc)
        clrc                        ; Clear carry bit (1 cyc)
        rrc.w   R12                     ; Divide by 2 (1 cyc)
        rra.w   R12                     ; Divide by 2 (1 cyc)
Delay2      nop                         ; (1 cyc)
        dec.w   R12                     ; (1 cyc)
        jnz     Delay2                  ; (2 cyc)
        pop     R12                     ; Restore R12 (2 cyc)
        ret                         ; (2 cyc)
;-------------------------------------------------------------------------------
; WriteDWBE
;
; Writes a DWORD to the given memory location in big-endian format. The
; memory address MUST be word-aligned.
;
; IN:  R12        Address
;      R14        Lower Word
;      R15        Upper Word
;-------------------------------------------------------------------------------
WriteDWBE
        swpb    R14                     ; Swap bytes in lower word
        swpb    R15                     ; Swap bytes in upper word
        mov.w   R15,0(R12)              ; Write 1st word to memory
```

```
        mov.w   R14,2(R12)              ; Write 2nd word to memory
        swpb    R14                     ; Restore R14
        swpb    R15                     ; Restore R15
        ret

        END
```

;---

Webside.c

;---

```
const unsigned char WebSide[] =
{
 "<html>\r\n"
 "<head>\r\n"
 "<meta http-equiv=\"refresh\" content=\"5\">\r\n"
 "<title>easyWEB - dynamic Webside</title>\r\n"
 "</head>\r\n"
 "\r\n"
 "<body bgcolor=\"#3030A0\" text=\"#FFFF00\">\r\n"
 "<p><b><font color=\"#FFFFFF\" size=\"6\"><i>Hello World!</i></font></b></p>\r\n"
 "\r\n"
 "<p><b>This is a dynamic webside hosted by the embedded Webserver</b> <b>easyWEB.</b></p>\r\n"
 "<p><b>Hardware:</b></p>\r\n"
 "<ul>\r\n"
 "<li><b>MSP430F149, 8 MHz, 60KB Flash, 2KB SRAM</b></li>\r\n"
 "<li><b>CS8900A Crystal Ethernet Controller</b></li>\r\n"
 "</ul>\r\n"
 "\r\n"
 "<p><b>A/D Converter Value Port P6.7:</b></p>\r\n"
 "\r\n"
 "<table bgcolor=\"#ff0000\" border=\"5\" cellpadding=\"0\" cellspacing=\"0\" width=\"500\">\r\n"
 "<tr>\r\n"
 "<td>\r\n"
 "<table width=\"AD7%\" border=\"0\" cellpadding=\"0\" cellspacing=\"0\">\r\n"
 "<tr><td bgcolor=\"#00ff00\"> </td></tr>\r\n"
 "</table>\r\n"
 "</td>\r\n"
 "</tr>\r\n"
 "</table>\r\n"
 "\r\n"
 "<table border=\"0\" width=\"500\">\r\n"
 "<tr>\r\n"
 "<td width=\"20%\">0V</td>\r\n"
 "<td width=\"20%\">0.4V</td>\r\n"
 "<td width=\"20%\">0.7V</td>\r\n"
 "<td width=\"20%\">1.1V</td>\r\n"
 "<td width=\"20%\">1.5V</td>\r\n"
 "</tr>\r\n"
 "</table>\r\n"
 "\r\n"
 "<p><b>MCU Temperature:</b></p>\r\n"
 "\r\n"
 "<table bgcolor=\"#ff0000\" border=\"5\" cellpadding=\"0\" cellspacing=\"0\" width=\"500\">\r\n"
```

```
  "<tr>\r\n"
  "<td>\r\n"
  "<table width=\"ADA%\" border=\"0\" cellpadding=\"0\" cellspacing=\"0\">\r\n"
  "<tr><td bgcolor=\"#00ff00\"> </td></tr> \r\n"
  "</table>\r\n"
  "</td>\r\n"
  "</tr>\r\n"
  "</table>\r\n"
  "\r\n"
  "<table border=\"0\" width=\"500\">\r\n"
  "<tr>\r\n"
  "<td width=\"20%\">0°C</td>\r\n"
  "<td width=\"20%\">10°C</td>\r\n"
  "<td width=\"20%\">20°C</td>\r\n"
  "<td width=\"20%\">30°C</td>\r\n"
  "<td width=\"20%\">40°C</td>\r\n"
  "</tr>\r\n"
  "</table>\r\n"
  "</body>\r\n"
  "</html>\r\n"
  "\r\n"
};
```

ΠΑΡΑΡΤΗΜΑ Β

► ΣΟΥΙΤΑ ΣΧΕΔΙΑΣΜΟΥ ΠΛΑΚΕΤΑΣ

Β.1 Διαδικασία Δημιουργίας Schematic στο Orcad

Β.1.1 Εισαγωγή στο Orcad

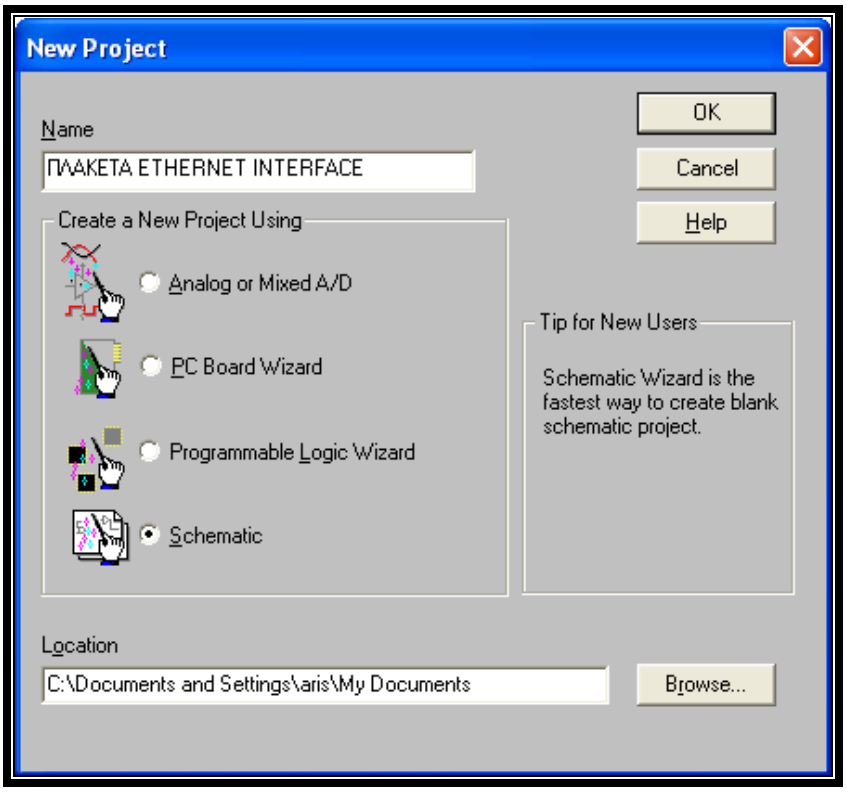
Το πρόγραμμα αυτό είναι μία σουίτα από εργαλεία δημιουργημένο από την Cadence, για το σχεδιασμό και το layout πλακετών κυκλωμάτων. Χρησιμοποιήθηκε η έκδοση Orcad suite 10.3. Το κύκλωμα που σχεδιάστηκε φαίνεται στο παρακάτω σχήμα(schematic).

Τα κύρια συστατικά του σχεδίου είναι:

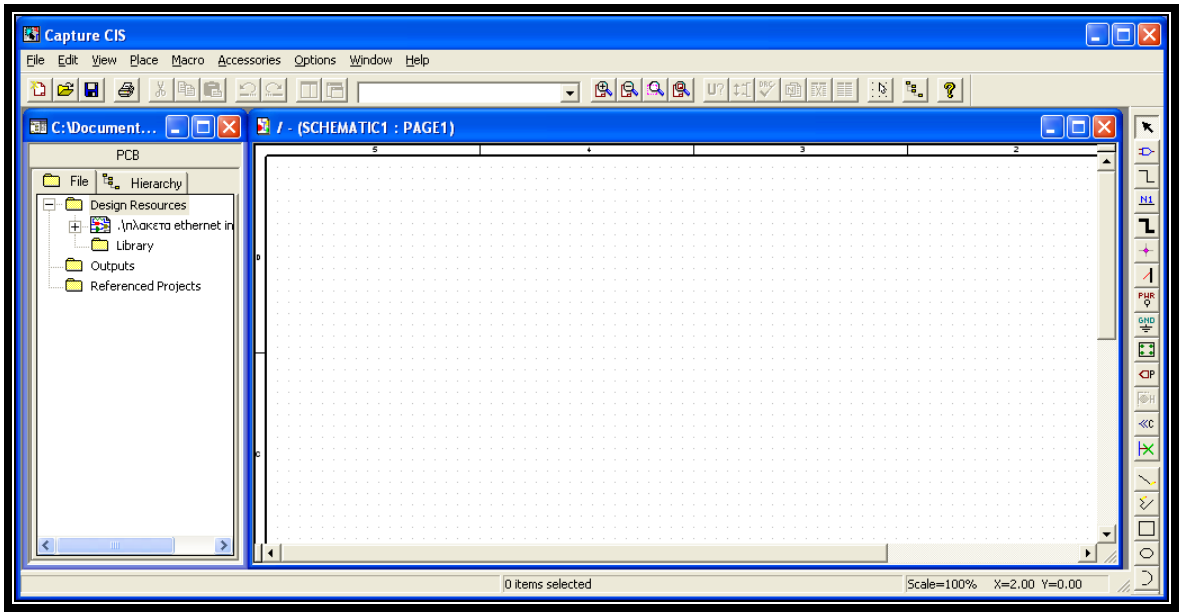
- Ο microcontroller MSP430F149IPMG4
- Ο ISA Ethernet Controller CS8900A.
- Ο Transformer E2023 (pulse Engineering).
- Ο RJ45 LAN Controller.
- Τρεις Headers (X3 (14-pin), X4 (14-pin), X6 (26(pin)).

Για τον σχεδιασμό του schematic της πλακέτας χρησιμοποιήθηκε το εργαλείο Capture της σουίτας.

Για την δημιουργία ενός νέου project επιλέγουμε στο μενού File→New→Project όπου θα μας προβάλει το παρακάτω παράθυρο διαλόγου (Dialog Box).

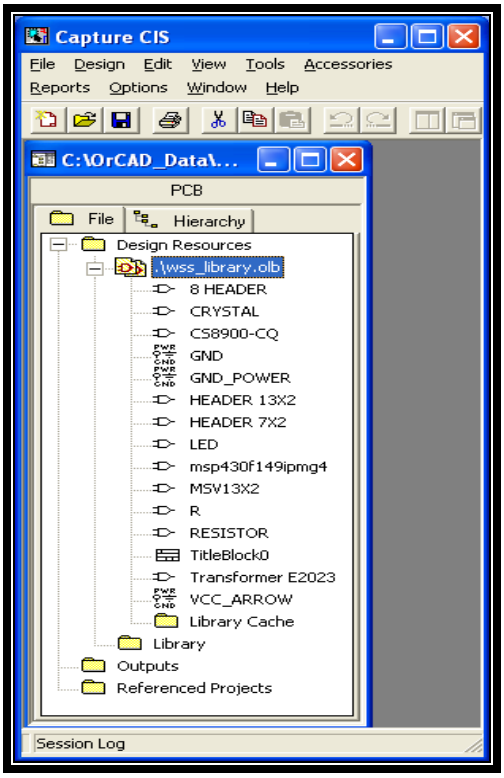


Εικόνα 1: Dialog Box.



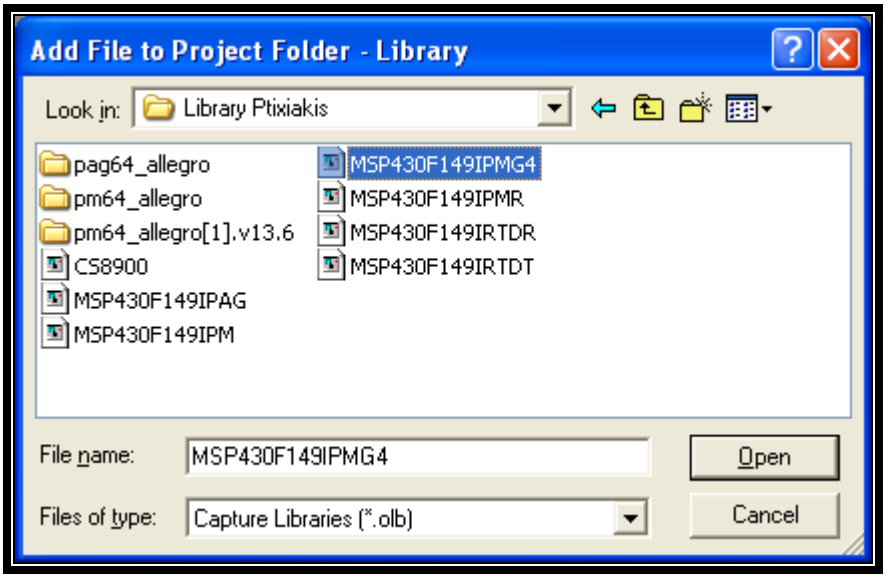
Εικόνα 2: Νέα άδεια σελίδα Schematic & παράθυρο του Project: Πλακέτα Ethernet Interface.

Εν συνεχεία θα δημιουργήσουμε μία βιβλιοθήκη με τα μέρη που θα χρησιμοποιηθούν στο Schematic. Για να επιτύχουμε αυτό (εικόνα 3) κάνουμε κλικ στο File→New→Library



Εικόνα 3: Βιβλιοθήκη Schematic.

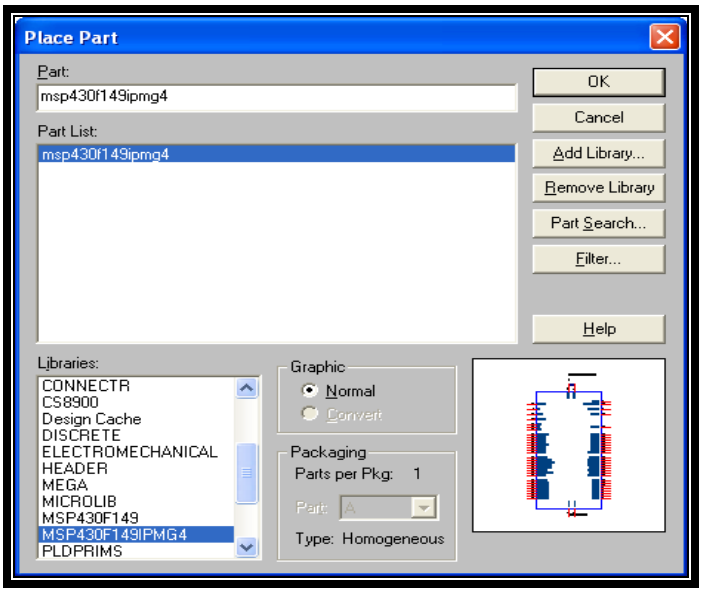
Με δεξί κλικ στον φάκελο Library επιλέγουμε add File όπου εισάγουμε από ένα αρχείο τα Schematic των MSP430F149IPMG4 και CS8900A (εικόνα 4).



Εικόνα 4: Εισαγωγή αρχείων σε βιβλιοθήκη

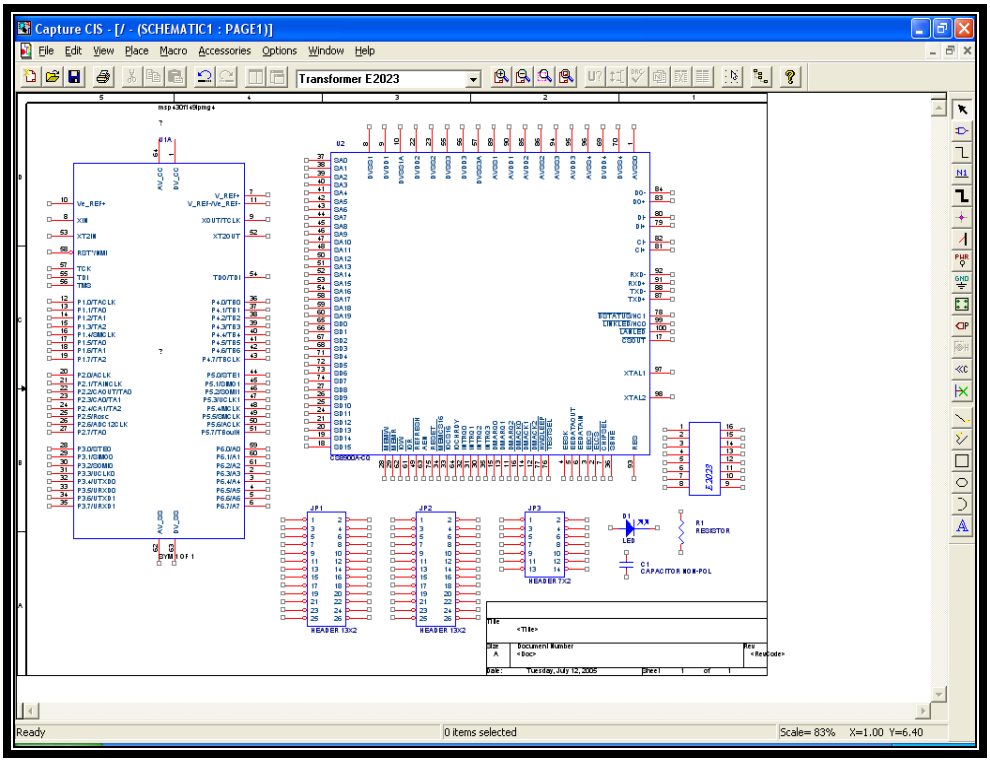
Β.1.2 Τοποθέτηση και Σύνδεση Μερών Πλακέτας

Ανοίγοντας την αρχική σελίδα του Schematic κάνουμε κλικ στο εικονίδιο place part στο ToolBar στο δεξί μέρος του Schematic. Στην συνέχεια εμφανίζεται ένα Dialog Box, όπως φαίνεται στην παρακάτω εικόνα.



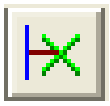
Εικόνα 5: Εισαγωγή μερών πλακέτας.

Επιλέγοντας τα μέρη που θα χρησιμοποιηθούν πατώντας στο παραπάνω Dialog Box Ok εισάγονται τα μέρη αυτά στο αριστερό μέρος της σελίδας του Schematic όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 6:Βασικά μέρη κυκλώματος της πλακέτας

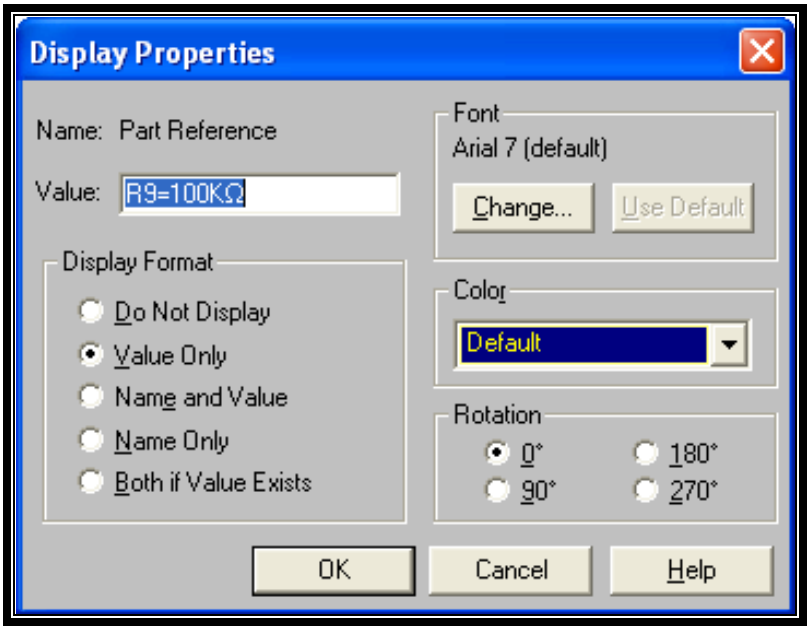
Για να δημιουργηθούν οι ηλεκτρικές ενώσεις μεταξύ των μερών θα χρειαστεί να τοποθετηθούν ηλεκτρικά εικονικά σύρματα. Από την παρακάτω εικόνα είναι το πρώτο αριστερά και στο Schematic βρίσκεται πάνω στο ToolBar σαν κουμπί συντόμευσης. Το μεσαίο στην εικόνα είναι για την δημιουργία ενός δίαυλου και το τελευταίο για Bus entry, όταν πρέπει κάποιο σύρμα να ενωθεί με ένα δίαυλο.



Τέλος, λόγου του ότι μπορεί να υπάρξουν κάποια pins που δεν χρησιμοποιούνται, θα χρησιμοποιηθεί το εικονίδιο δεξιά στο ToolBar για την ακύρωση αυτών των pins.

Για την ευκολία του χρήστη αρχικά θα πρέπει να δηλωθούν τα ονόματα, όχι μόνο των βασικών πακέτων (packages) του κάθε μέρους, αλλά και τα pins που τυχόν υπάρχουν σε αυτά. Όσον αφορά την σωστή μετέπειτα λειτουργία του Layout θα πρέπει να δηλωθούν τα Footprints των μερών της πλακέτας.

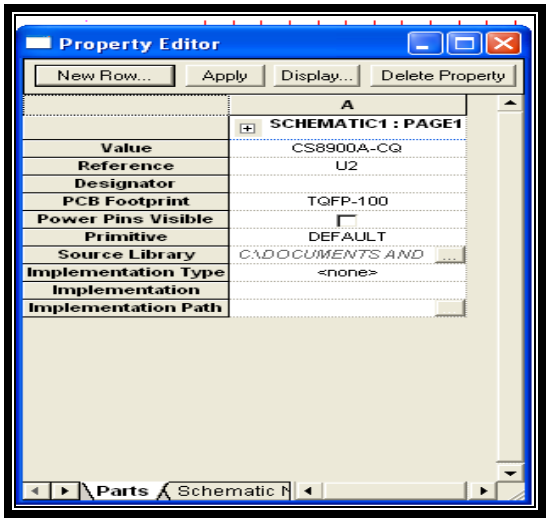
Για την εισαγωγή των τιμών (values) στους πυκνωτές και τις αντιστάσεις κάνουμε διπλό κλικ πάνω στο όνομα του κάθε μέρους, όπου εμφανίζεται στη συνέχεια το παρακάτω Dialog Box (εικόνα 7).



Εικόνα 7: Display Properties

Για τις διαδικασίες αυτές κάνουμε τα εξής:

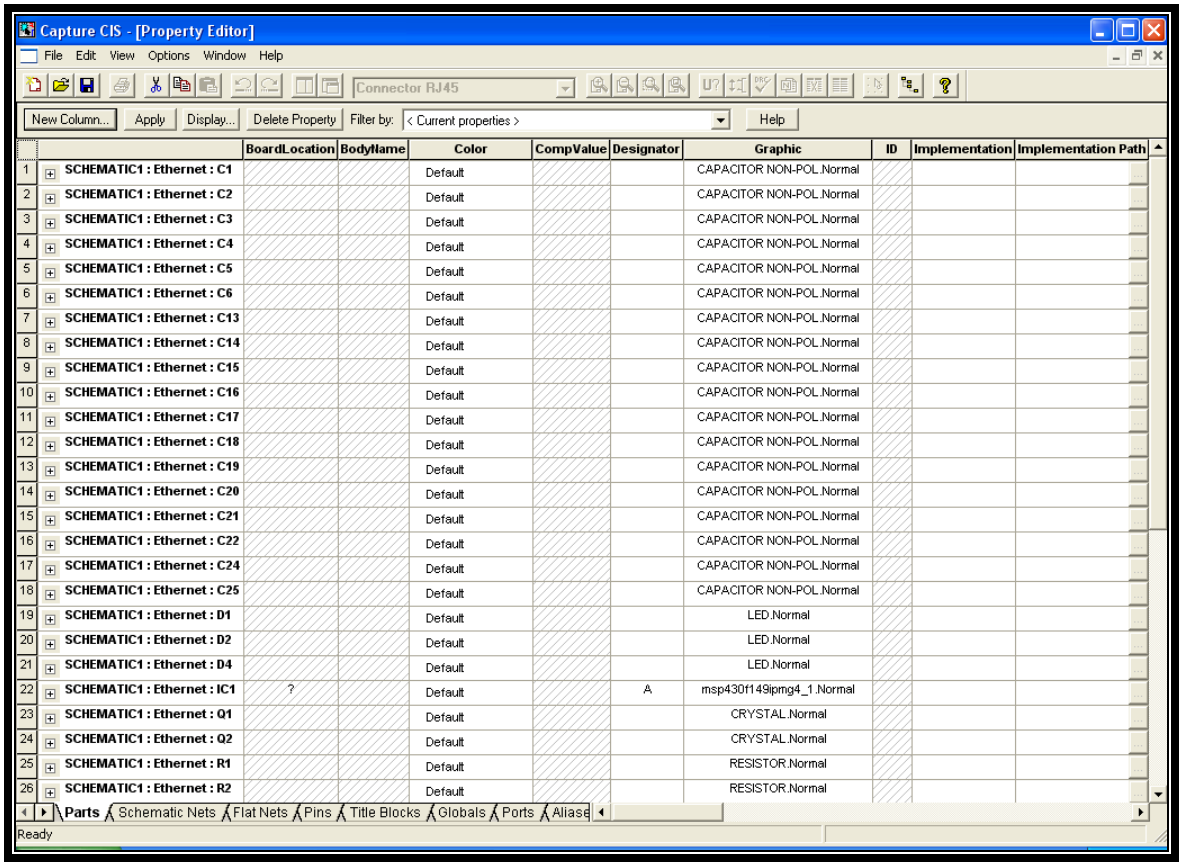
Με διπλό κλικ πάνω στο κάθε μέρος ανοίγει ένας Property Editor, όπως φαίνεται στην παρακάτω εικόνα



Εικόνα 8:Property Editor

B.1.3 Ορίζοντας Footprints στα Μέρη του Schematic

Σε αυτό το κομμάτι του Orcad θα πρέπει να ορίσουμε και να αντιστοιχήσουμε τα συγκεκριμένα footprints στο κάθε μέρος ώστε να μπορούμε να χρησιμοποιήσουμε το Layout Plus. Πατώντας ctrl-A επιλέγουμε όλα τα μέρη από το schematic. Στη συνέχεια πατώντας ctrl-E εμφανίζεται το dialog box του property editor.



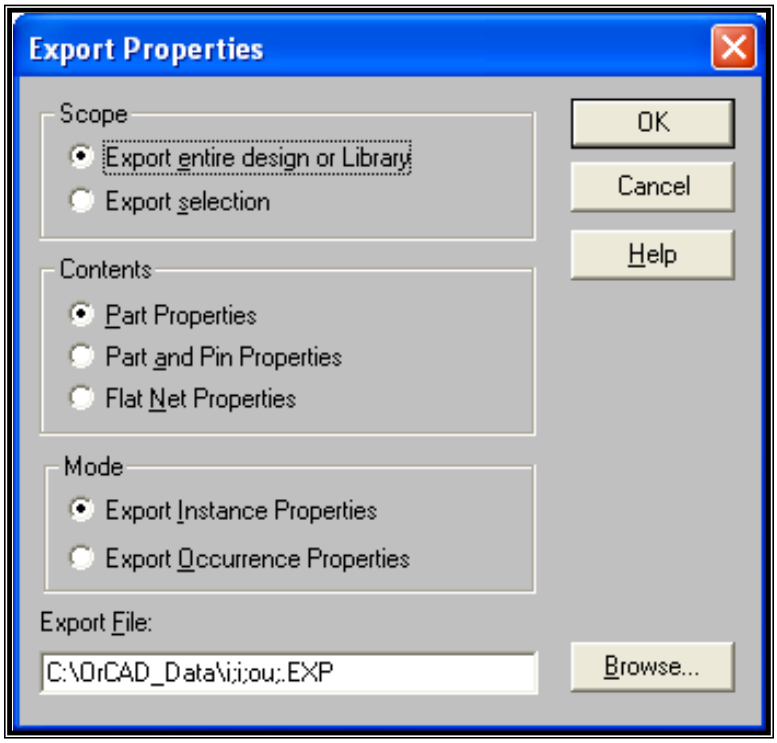
		BoardLocation	BodyName	Color	CompValue	Designator	Graphic	ID	Implementation	Implementation Path
1	SCHEMATIC1 : Ethernet : C1			Default			CAPACITOR NON-POL.Normal			
2	SCHEMATIC1 : Ethernet : C2			Default			CAPACITOR NON-POL.Normal			
3	SCHEMATIC1 : Ethernet : C3			Default			CAPACITOR NON-POL.Normal			
4	SCHEMATIC1 : Ethernet : C4			Default			CAPACITOR NON-POL.Normal			
5	SCHEMATIC1 : Ethernet : C5			Default			CAPACITOR NON-POL.Normal			
6	SCHEMATIC1 : Ethernet : C6			Default			CAPACITOR NON-POL.Normal			
7	SCHEMATIC1 : Ethernet : C13			Default			CAPACITOR NON-POL.Normal			
8	SCHEMATIC1 : Ethernet : C14			Default			CAPACITOR NON-POL.Normal			
9	SCHEMATIC1 : Ethernet : C15			Default			CAPACITOR NON-POL.Normal			
10	SCHEMATIC1 : Ethernet : C16			Default			CAPACITOR NON-POL.Normal			
11	SCHEMATIC1 : Ethernet : C17			Default			CAPACITOR NON-POL.Normal			
12	SCHEMATIC1 : Ethernet : C18			Default			CAPACITOR NON-POL.Normal			
13	SCHEMATIC1 : Ethernet : C19			Default			CAPACITOR NON-POL.Normal			
14	SCHEMATIC1 : Ethernet : C20			Default			CAPACITOR NON-POL.Normal			
15	SCHEMATIC1 : Ethernet : C21			Default			CAPACITOR NON-POL.Normal			
16	SCHEMATIC1 : Ethernet : C22			Default			CAPACITOR NON-POL.Normal			
17	SCHEMATIC1 : Ethernet : C24			Default			CAPACITOR NON-POL.Normal			
18	SCHEMATIC1 : Ethernet : C25			Default			CAPACITOR NON-POL.Normal			
19	SCHEMATIC1 : Ethernet : D1			Default			LED.Normal			
20	SCHEMATIC1 : Ethernet : D2			Default			LED.Normal			
21	SCHEMATIC1 : Ethernet : D4			Default			LED.Normal			
22	SCHEMATIC1 : Ethernet : IC1	?		Default		A	msp430f149ipmg4_1.Normal			
23	SCHEMATIC1 : Ethernet : Q1			Default			CRYSTAL.Normal			
24	SCHEMATIC1 : Ethernet : Q2			Default			CRYSTAL.Normal			
25	SCHEMATIC1 : Ethernet : R1			Default			RESISTOR.Normal			
26	SCHEMATIC1 : Ethernet : R2			Default			RESISTOR.Normal			

Εικόνα 9: Property Editor

Αφού θέσουμε στο πλαίσιο filter by το orcad layout κλείνουμε το property editor και μπορούμε να κάνουμε εξαγωγή της λίστας αυτής.

B.1.4 Εξαγωγή Χαρακτηριστικών από το Schematic

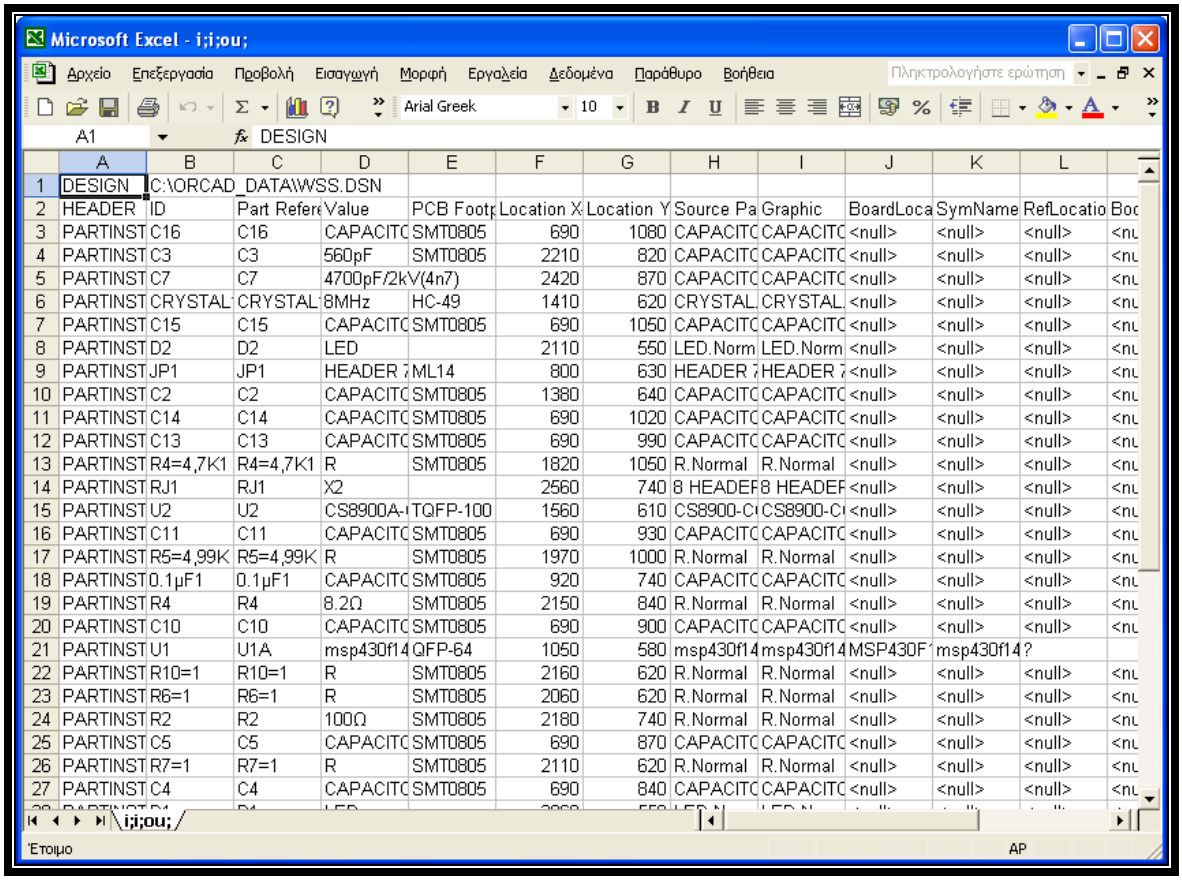
Για να πετύχουμε την εξαγωγή χαρακτηριστικών, αφού πρώτα έχουμε επιλέξει το αρχείο με την κατάληξη *.dsn κάνουμε κλικ στο Tools→Export Properties και θα μας εμφανιστεί το παρακάτω dialog box.



Εικόνα 10: Εξαγωγή Ιδιοτήτων

Αφήνουμε όλες τις παραμέτρους όπως είναι προεπιλεγμένες και κάνουμε κλικ στο OK. Τώρα μπορούμε να επεξεργαστούμε αυτό το αρχείο και να το επαναεισάγουμε στο Capture.

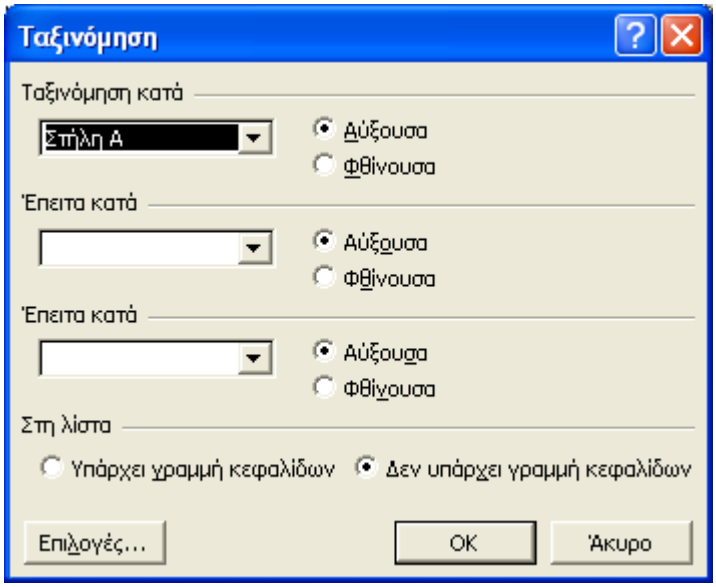
Για να μπορέσουμε να επεξεργαστούμε το αρχείο αυτό καλύτερα θα χρειαστεί να το ανοίξουμε με Microsoft Excel. Αφού το ανοίξουμε το αρχείο, θα εμφανίζει όπως παρακάτω:



	A	B	C	D	E	F	G	H	I	J	K	L	
1	DESIGN	C:\ORCAD_DATA\WSS.DSN											
2	HEADER	ID	Part Refere	Value	PCB Footp	Location X	Location Y	Source Pa	Graphic	BoardLoca	SymName	RefLocatio	Boc
3	PARTINST	C16	C16	CAPACITC	SMT0805	690	1080	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
4	PARTINST	C3	C3	560pF	SMT0805	2210	820	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
5	PARTINST	C7	C7	4700pF/2kV(4n7)		2420	870	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
6	PARTINST	CRYSTAL	CRYSTAL	8MHz	HC-49	1410	620	CRYSTAL	CRYSTAL	<null>	<null>	<null>	<nu
7	PARTINST	C15	C15	CAPACITC	SMT0805	690	1050	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
8	PARTINST	D2	D2	LED		2110	550	LED.Norm	LED.Norm	<null>	<null>	<null>	<nu
9	PARTINST	JP1	JP1	HEADER 7	ML14	800	630	HEADER 7	HEADER 7	<null>	<null>	<null>	<nu
10	PARTINST	C2	C2	CAPACITC	SMT0805	1380	640	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
11	PARTINST	C14	C14	CAPACITC	SMT0805	690	1020	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
12	PARTINST	C13	C13	CAPACITC	SMT0805	690	990	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
13	PARTINST	R4=4,7K1	R4=4,7K1	R	SMT0805	1820	1050	R.Normal	R.Normal	<null>	<null>	<null>	<nu
14	PARTINST	RJ1	RJ1	X2		2560	740	8 HEADEF	8 HEADEF	<null>	<null>	<null>	<nu
15	PARTINST	U2	U2	CS8900A-I	TQFP-100	1560	610	CS8900-CI	CS8900-CI	<null>	<null>	<null>	<nu
16	PARTINST	C11	C11	CAPACITC	SMT0805	690	930	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
17	PARTINST	R5=4,99K	R5=4,99K	R	SMT0805	1970	1000	R.Normal	R.Normal	<null>	<null>	<null>	<nu
18	PARTINST	0.1μF1	0.1μF1	CAPACITC	SMT0805	920	740	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
19	PARTINST	R4	R4	8.2Ω	SMT0805	2150	840	R.Normal	R.Normal	<null>	<null>	<null>	<nu
20	PARTINST	C10	C10	CAPACITC	SMT0805	690	900	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
21	PARTINST	U1	U1A	msp430f14	QFP-64	1050	580	msp430f14	msp430f14	MSP430F	msp430f14	?	
22	PARTINST	R10=1	R10=1	R	SMT0805	2160	620	R.Normal	R.Normal	<null>	<null>	<null>	<nu
23	PARTINST	R6=1	R6=1	R	SMT0805	2060	620	R.Normal	R.Normal	<null>	<null>	<null>	<nu
24	PARTINST	R2	R2	100Ω	SMT0805	2180	740	R.Normal	R.Normal	<null>	<null>	<null>	<nu
25	PARTINST	C5	C5	CAPACITC	SMT0805	690	870	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu
26	PARTINST	R7=1	R7=1	R	SMT0805	2110	620	R.Normal	R.Normal	<null>	<null>	<null>	<nu
27	PARTINST	C4	C4	CAPACITC	SMT0805	690	840	CAPACITC	CAPACITC	<null>	<null>	<null>	<nu

Εικόνα 11: Εμφανίζοντας τη δημιουργημένη λίστα από τα χαρακτηριστικά στο Excel

Επιλέγουμε όλα τα κελιά και κάνουμε κλικ στο Δεδομένα→Ταξινόμηση και εμφανίζεται το Dialog Box της ταξινόμησης.



Εικόνα 12: Dialog Box Ταξινόμησης.

Κάνουμε ταξινόμηση σύμφωνα με την τιμή και το Part Reference και κάνουμε κλικ στο OK.

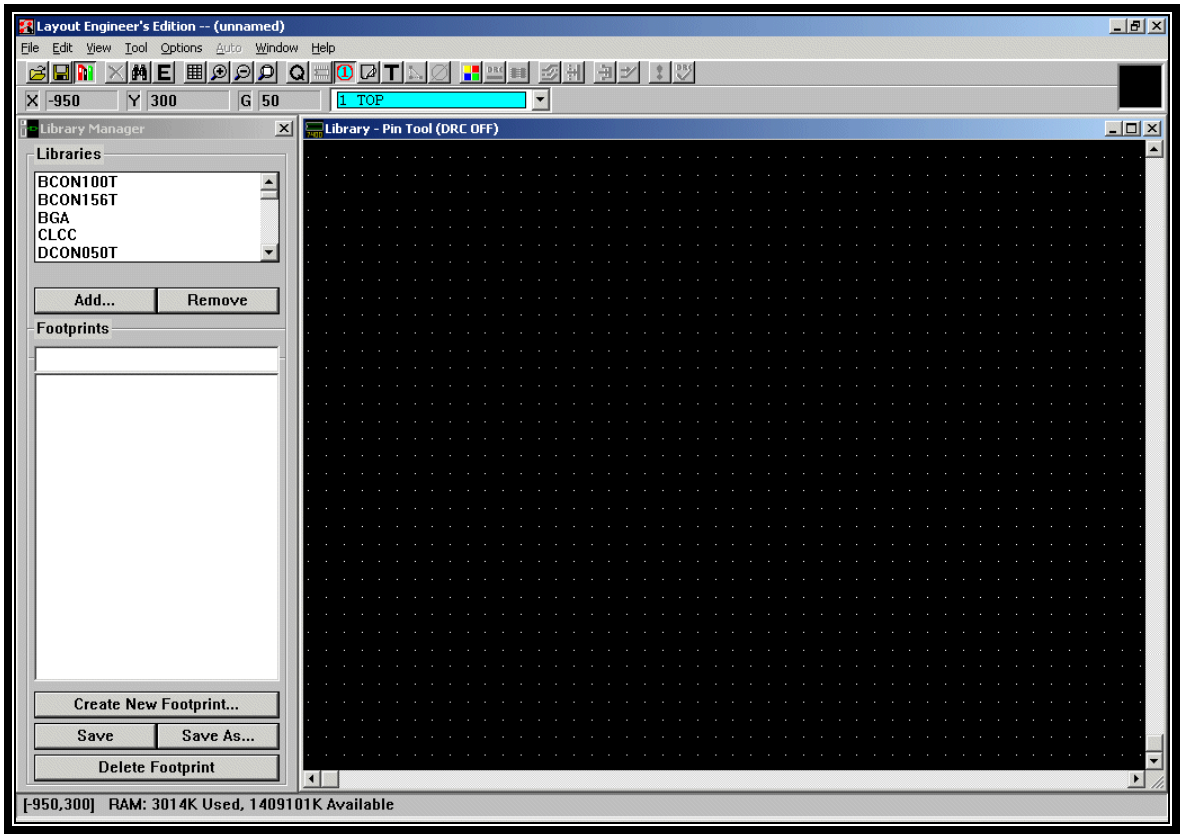
Το αρχείο τώρα μπορεί να επαναεισαχθεί στο Orcad. Στο Capture επιλέγουμε Tools→Import Properties και κάνουμε κλικ στο OK.

B.2 Διαδικασία Δημιουργίας Layout στο Orcad

Πριν ξεκινήσουμε την διαδικασία δημιουργίας της πλακέτας στο Layout θα πρέπει να δημιουργήσουμε τα footprints των μερών της πλακέτας .

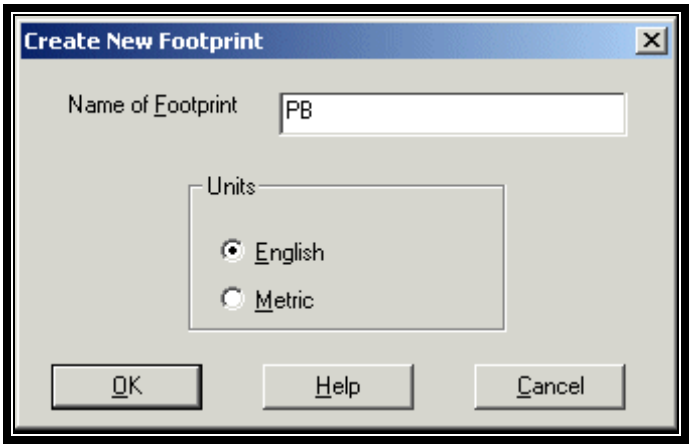
Όλα τα footprints που θα δημιουργηθούν θα περιλαμβάνονται στην ίδια βιβλιοθήκη(library) που θα χρησιμοποιήσουμε για να τα αποθηκεύσουμε.

Ξεκινούμε το Layout Plus και ανοίγουμε το Library Manager κάνοντας κλίκ στο **Tools→Library Manager** και εμφανίζεται η παρακάτω επιφάνεια εργασίας (εικόνα 13)



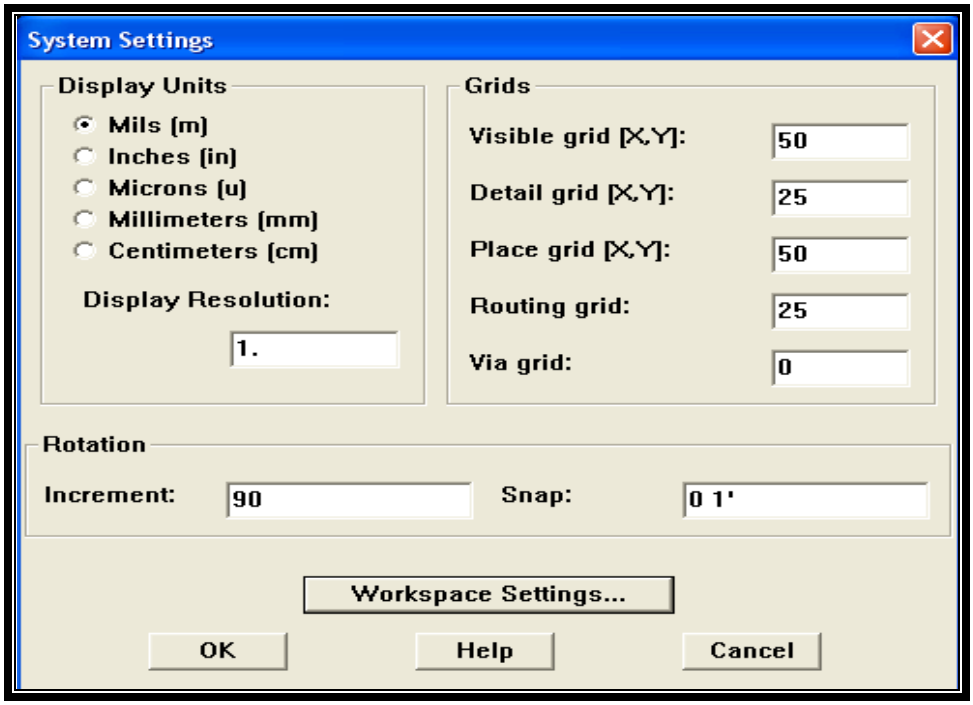
Εκόνα 13: Library Manager

Στα αριστερά μπορούμε να δούμε ότι είδη υπάρχουν αρκετές έτοιμες βιβλιοθήκες που μπορούμε να χρησιμοποιήσουμε. Στον **Library Manager** κάνουμε κλίκ στο **Create New Footprint** και εμφανίζεται το παρακάτω παράθυρο διαλόγου (εικόνα 14) όπου ονομάζουμε το **Footprint.**



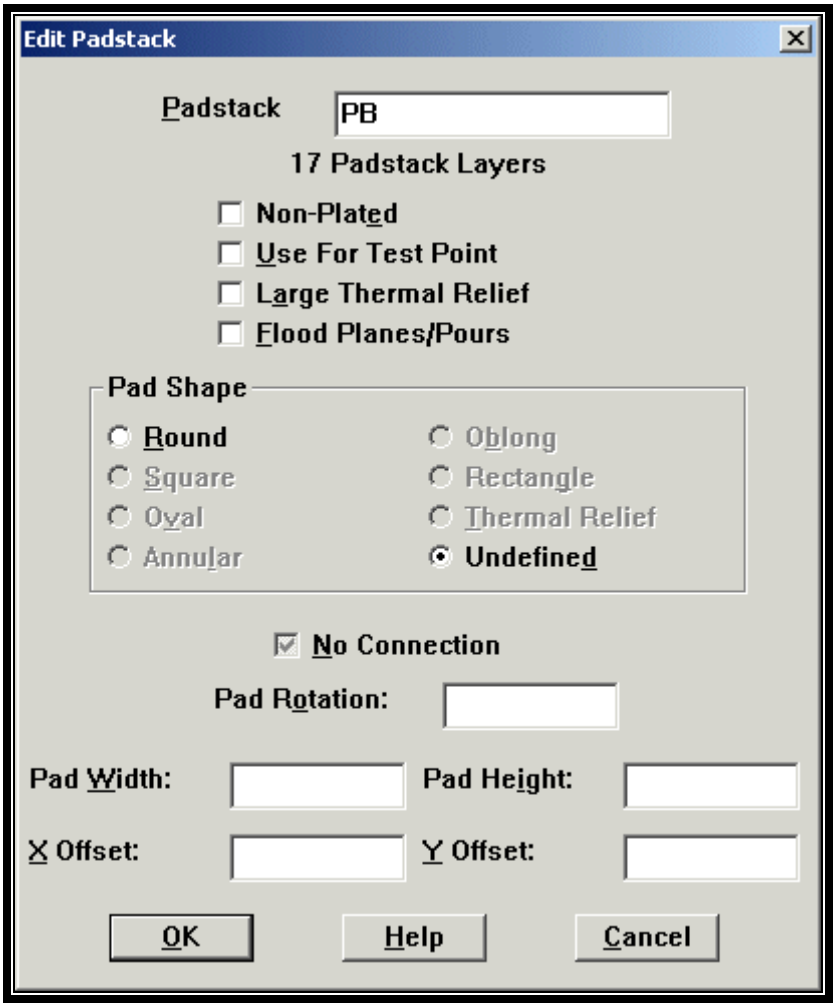
Εικόνα 14: Δημιουργώντας ένα νέο Footprint

Κάνουμε κλίκ **Options→System Settings** για να αλλάξουμε settings του συστήματος όπως το σύστημα μέτρησης (εικόνα 15)



Εικόνα 15: Αλλάζοντας βασικές Ρυθμίσεις

Το μέρος της πλακέτας που δημιουργούμε δηλαδή το SM/R_0805 έχει δύο pin αλλα χρειάζεται να αρχικοποιήσουμε μόνο το ένα αφού είναι ίδια. Ανοίγουμε τα padstacks spreadsheet και επιλέγουμε να αλλάξουμε το T1 που χρησιμοποιείται ηδη από το πρώτο pin. Επιλέγουμε το T1 και εμφανίζεται το παράθυρο διαλόγου (εικόνα 16) για όλα τα layers στα padstacks.



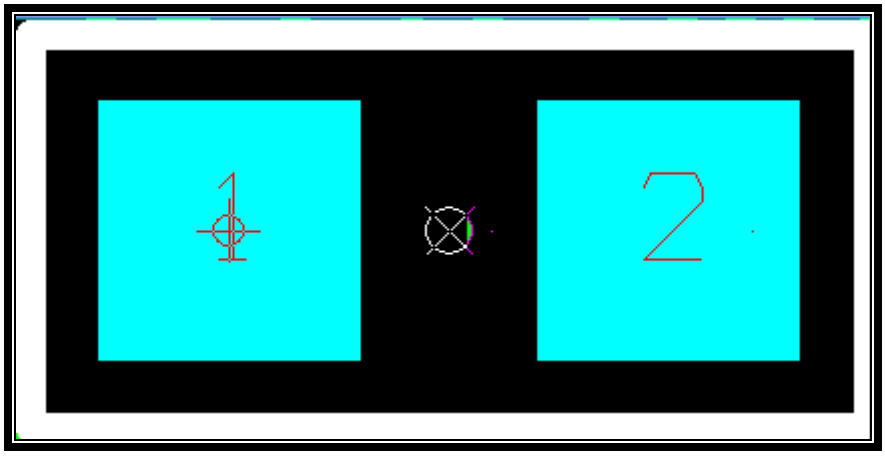
Εικόνα 16: Παράθυρο διαλόγου για όλα τα layers στα padstacks

Padstacks

Padstack or Layer Name	Pad Shape	Pad Width	Pad Height	X Offset
SPARE3	Round	22	22	0
SM.llb_pad5_1				
TOP	Square	50	50	0
BOTTOM	Undefined	0	0	0
PLANE	Undefined	0	0	0
INNER	Undefined	0	0	0
SMTOP	Square	50	50	0
SMBOT	Undefined	0	0	0
SPTOP	Square	50	50	0
SPBOT	Undefined	0	0	0
SSTOP	Undefined	0	0	0
SSBOT	Undefined	0	0	0
ASYTOP	Square	50	50	0
ASYBOT	Undefined	0	0	0
DRLDWG	Undefined	0	0	0
DRILL	Undefined	0	0	0
COMMENT LAYER	Undefined	0	0	0
SPARE2	Undefined	0	0	0
SPARE3	Undefined	0	0	0

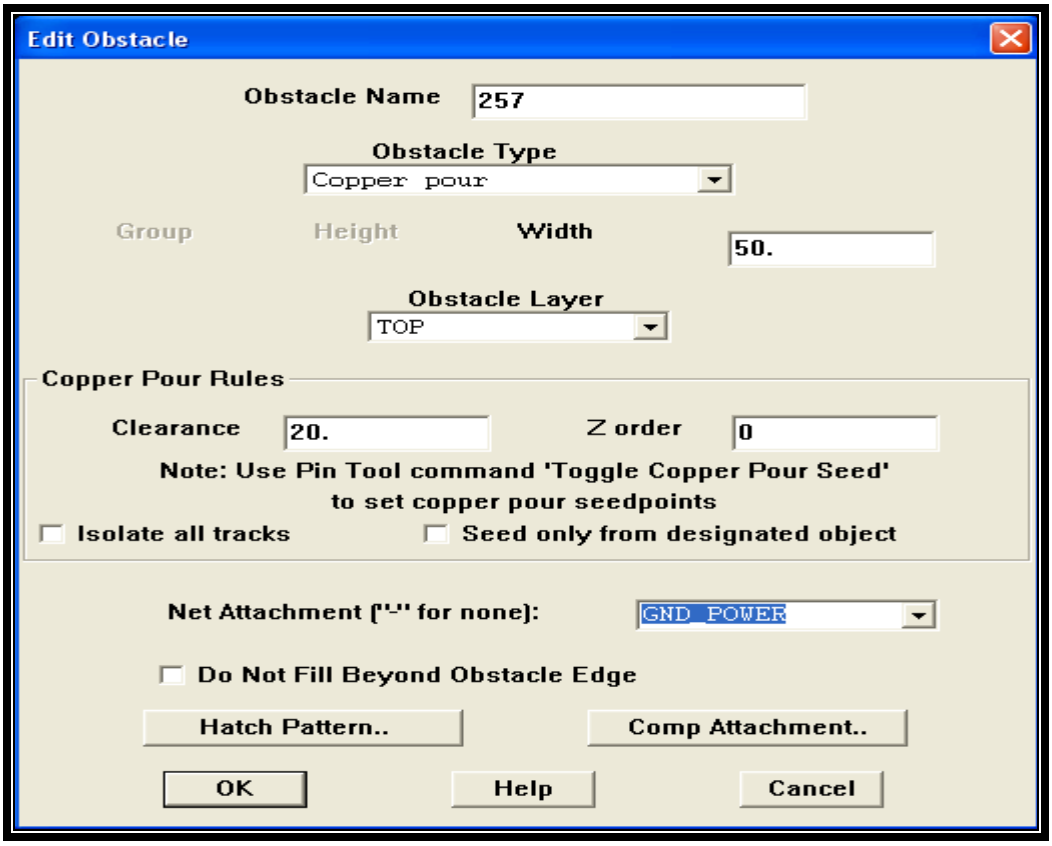
Εικόνα 17: Padstacks

Τελικά η τελική μορφή του μέρους SM/R_0805 μετα και από άλλες διάφορες παραμετροποιήσεις θα έχει την παρακάτω μορφή (εικόνα 18)

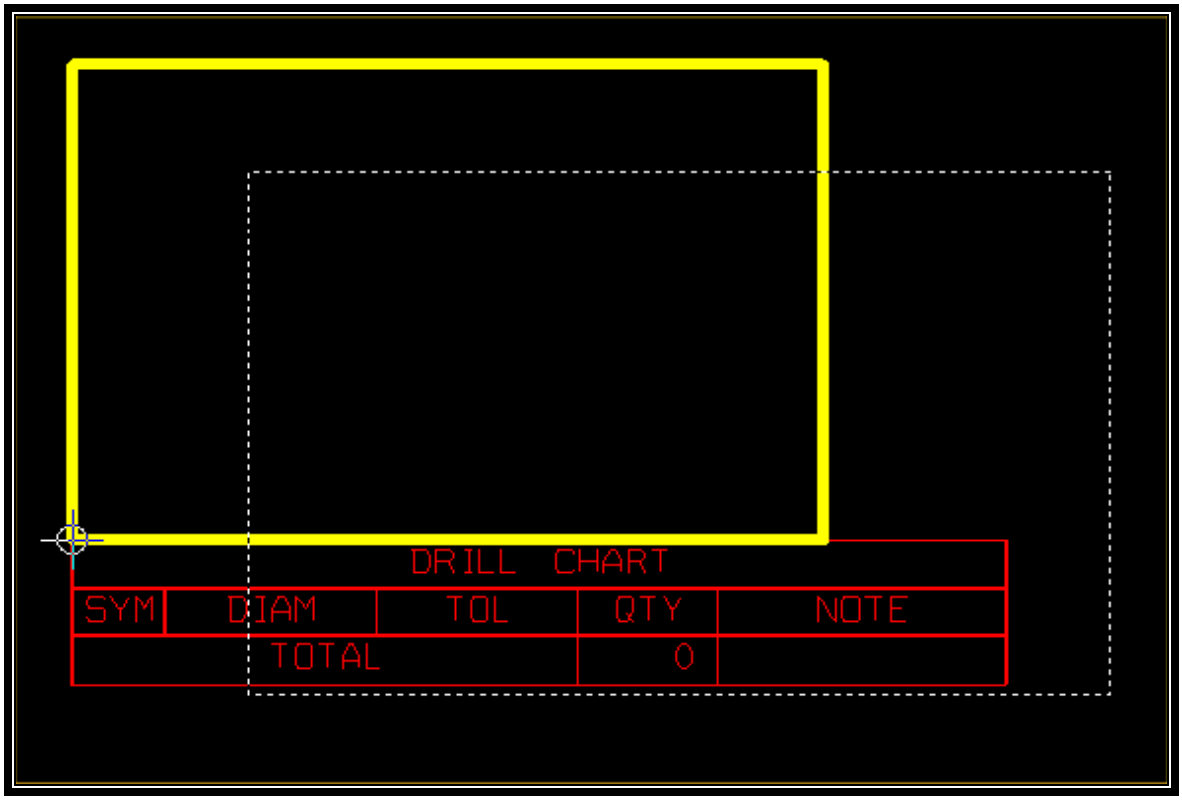


Εικόνα 18: Τελική μορφή μέρους SM/R

Πρίν ξεκινήσουμε να τοποθετούμε τα μέρη της πλακέτας θα πρέπει να φτιάξουμε το περίγραμμα της πλακέτας. Για να δημιουργήσουμε το περίγραμμα επιλέγουμε το **Obstacle Tool**, κάνουμε δεξί κλίκ στο New και επιλέγουμε Properties (εικόνα 19)



Εικόνα 19: Ιδιότητες Περιγράμματος



Εικόνα 20: Περίγραμμα

Στην συνέχεια αλλάζουμε το layer stackup. Αυτός ο πίνακας έχει όλα τα layers που χρησιμοποιούνται στο σχήμα. (εικόνα 21)

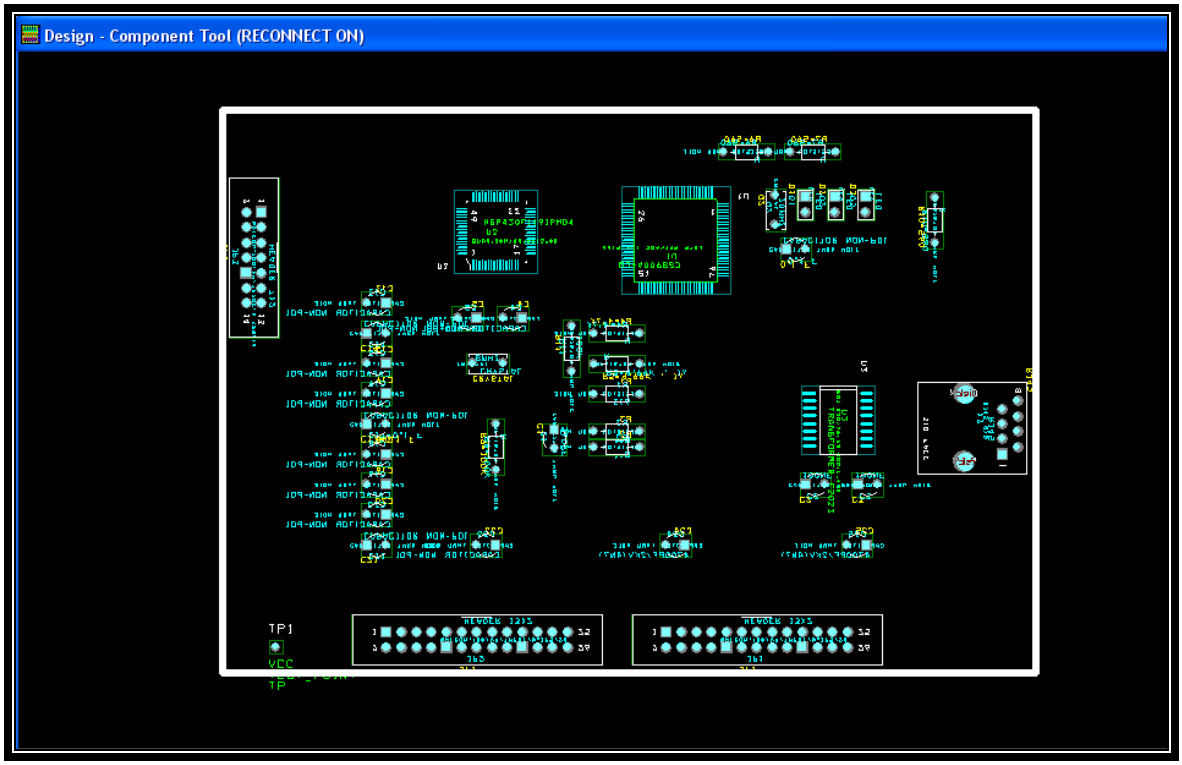
Layers

Layer Name	Layer Hotkey	Layer NickName	Layer Type	Mirror Layer
TOP	1	TOP	Routing	BOTTOM
BOTTOM	2	BOT	Routing	TOP
GND	3	GND	Plane	(None)
POWER	4	PWR	Plane	(None)
INNER1	5	IN1	Routing	(None)
INNER2	6	IN2	Routing	(None)
INNER3	7	IN3	Unused	(None)
INNER4	8	IN4	Unused	(None)
INNER5	9	IN5	Unused	(None)
INNER6	Ctrl + 0	IN6	Unused	(None)
INNER7	Ctrl + 1	IN7	Unused	(None)
INNER8	Ctrl + 2	IN8	Unused	(None)
INNER9	Ctrl + 3	IN9	Unused	(None)
INNER10	Ctrl + 4	I10	Unused	(None)
INNER11	Ctrl + 5	I11	Unused	(None)
INNER12	Ctrl + 6	I12	Unused	(None)
SMTOP	Ctrl + 7	SMT	Doc	SMBOT
SMBOT	Ctrl + 8	SMB	Doc	SMTOP

Εικόνα 21: Layer Stackup

Β.3 Τοποθετώντας Μέρη & Συνδέσεις της Πλακέτας

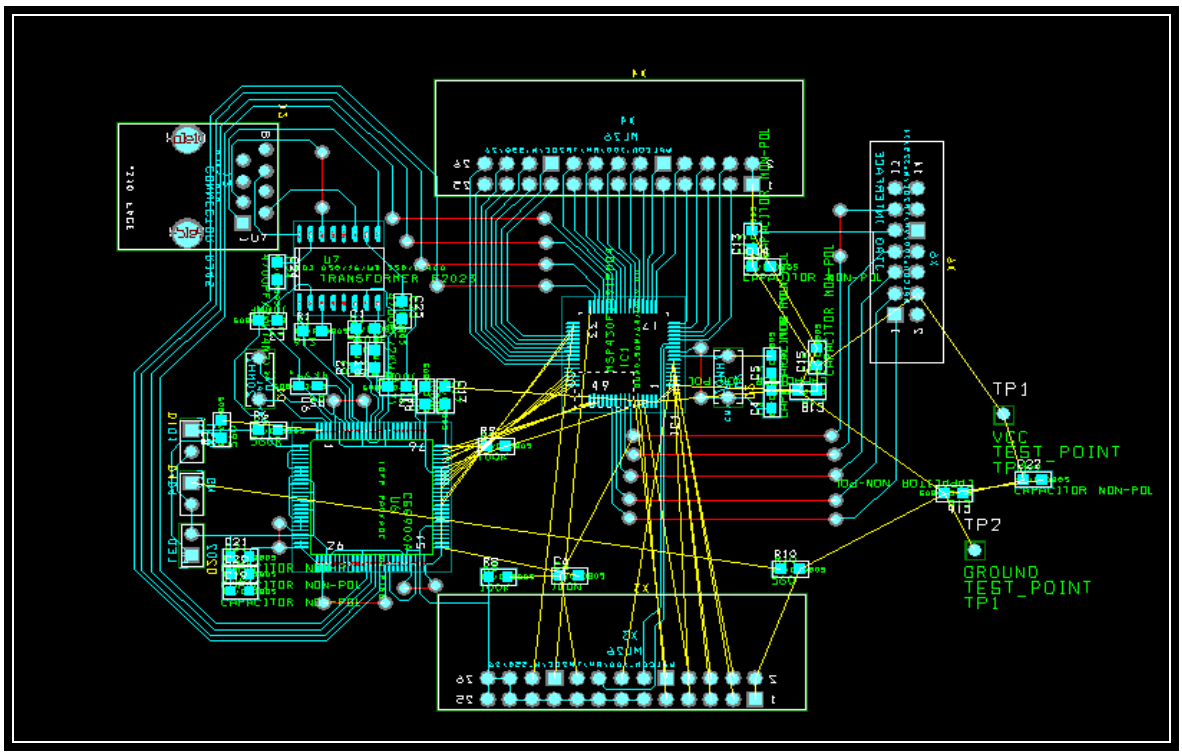
Στο σημείο αυτό είμαστε έτοιμοι να τοποθετήσουμε τα μέρη της πλακέτας, κάνοντας κλίκ στο Component Tool όπως φαίνεται στο εικονίδιο αριστερά, τοποθετούμε τα μέρη του σχεδίου μας. (εικόνα 22)



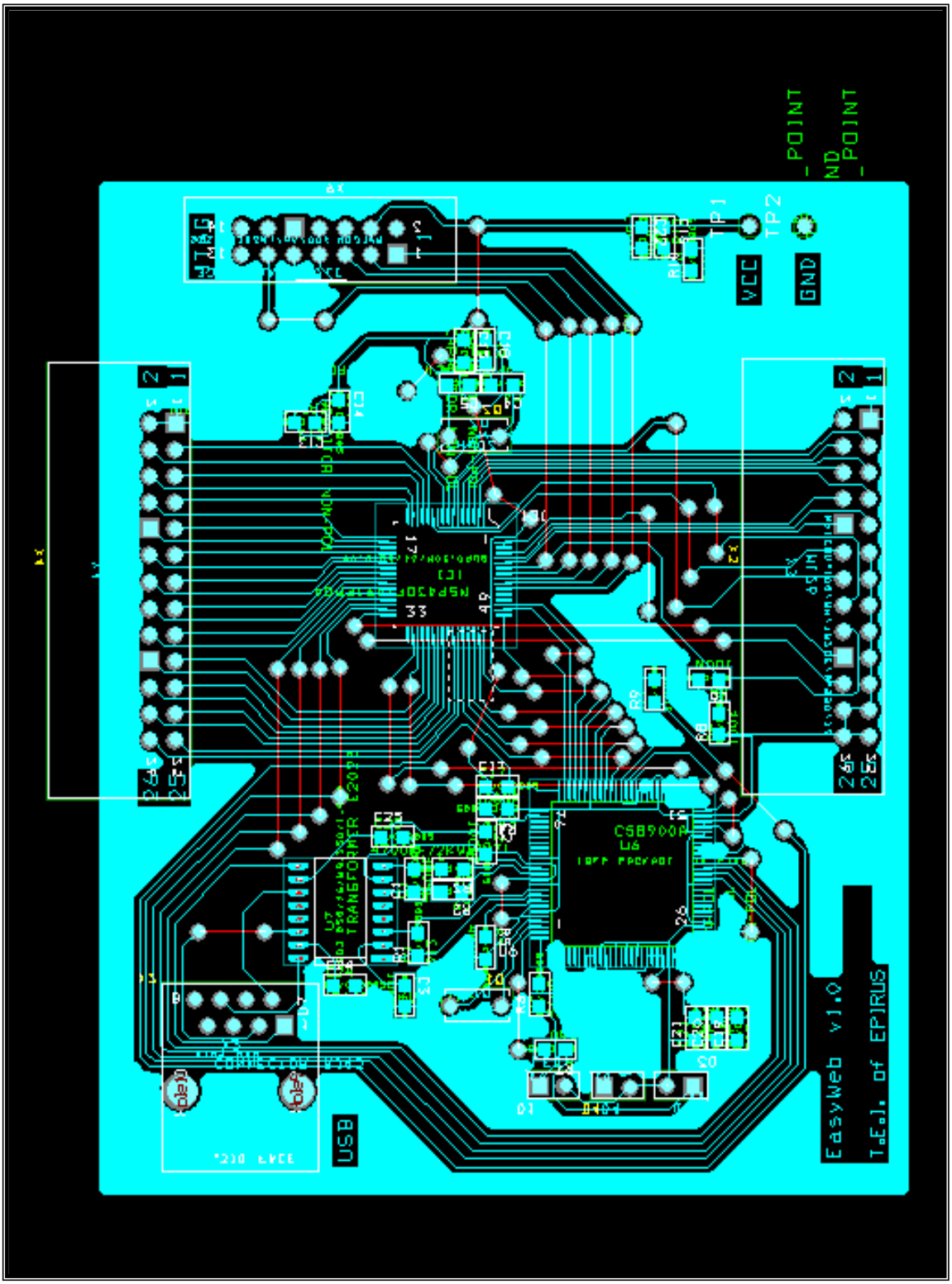
Εικόνα 22: Τοποθετημένα μέρη του σχεδίου

Για να κάνουμε τις συνδέσεις και τα Nets μεταξύ των μερών της πλακέτας κάνουμε κλίκ στο εικονίδιο Conection Toll.

Στην συνέχεια κάνουμε την δρομολόγηση (route) των συνδέσεων επιλέγοντας ένα από τα δύο tools τα Edit Segment Mode Add/Edit Route Mode. Πρίν όμως από αυτά πρέπει να ανοίξουμε το nets spreadsheet όπου επιλέγουμε όλα τα nets του σχήματος.



Εικόνα 23: Τα μέρη του σχεδίου με τις συνδέσεις



Εικόνα 24: Ολοκληρωμένο σχέδιο της πλακέτας

ΠΑΡΑΡΤΗΜΑ Γ΄

Γ. Η ΣΟΥΙΤΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

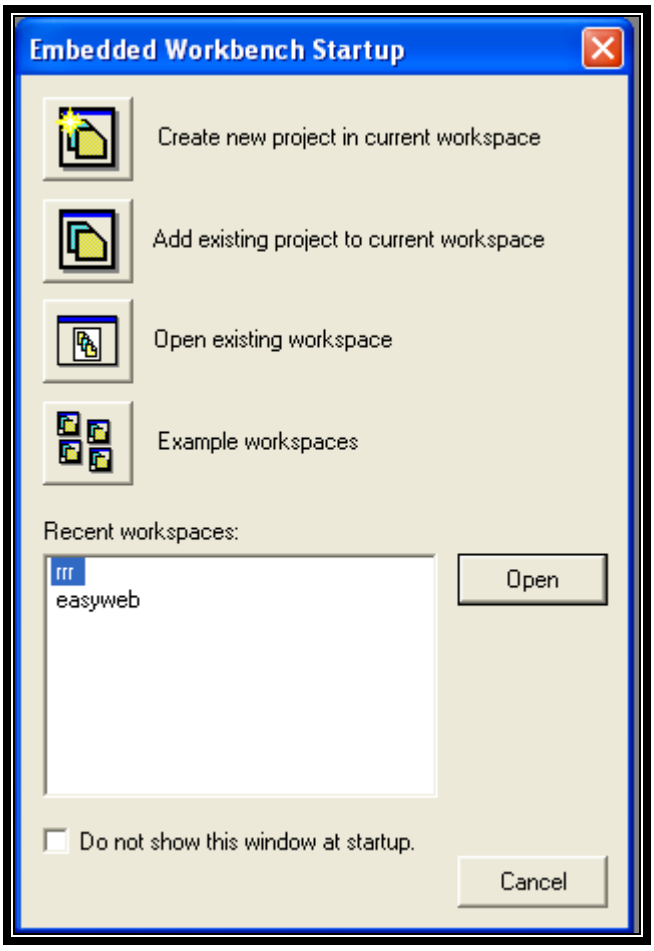
Γ.1 IAR EMBEDDED WORKBENCH Ver.4

► Δημιουργία Project με Κώδικα Προγραμματισμού

Γ.1.1 Χρήση Σουίτας Προγραμματισμού: IAR Embedded Workbench Version 4

Για τη ολοκλήρωση του Project: Ethernet Interface χρησιμοποιήθηκε η σουίτα προγραμματισμού IAR Embedded Workbench Ver. 4

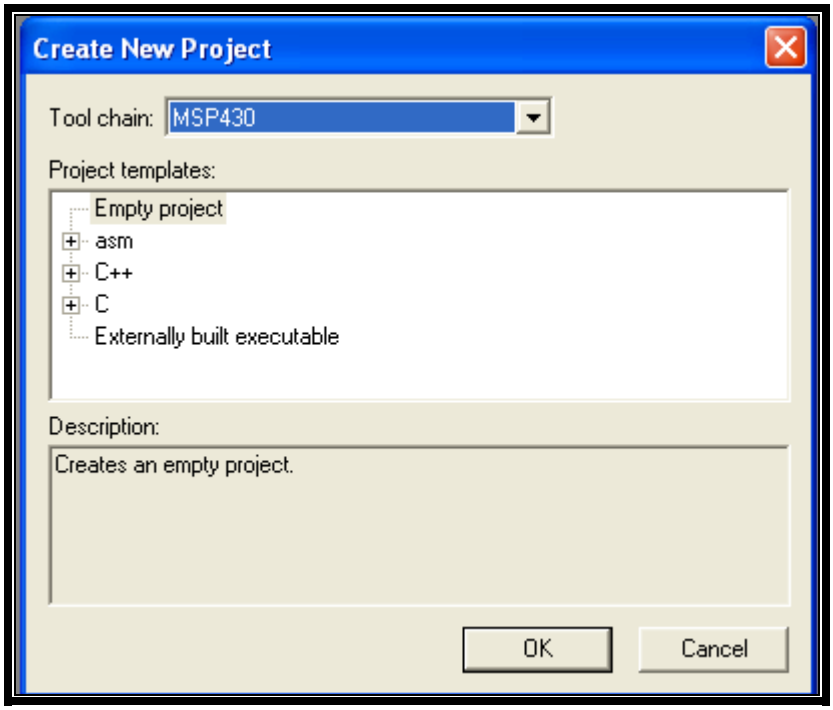
Ακολουθήθηκαν τα παρακάτω βήματα: Αρχικά όταν ανοίξουμε το πρόγραμμα θα εμφανιστεί το παρακάτω dialog box (εικόνα 1)



Εικόνα 1: Αρχικό παράθυρο

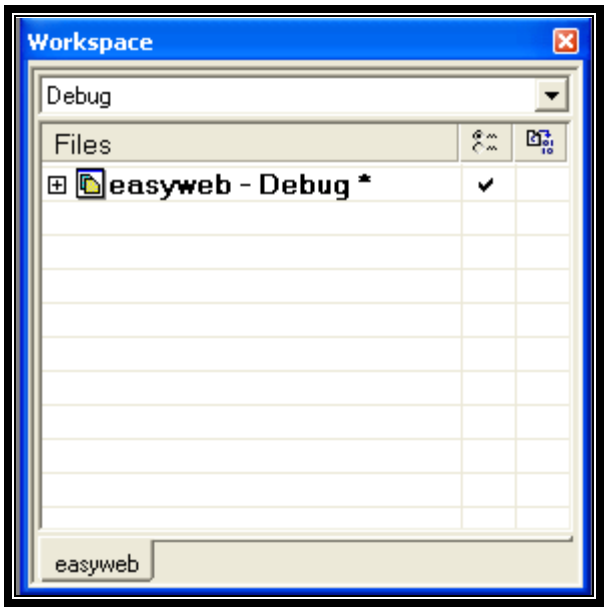
Στη συνέχεια, πρέπει να δημιουργήσουμε ένα καινούριο Project στον ήδη υπάρχοντα workspace με βάση τα παρακάτω:

Για να δημιουργήσουμε το Project κάνουμε κλικ στο Create new project in current workspace και εμφανίζεται το παρακάτω dialog box (εικόνα 2).



Εικόνα 2: Δημιουργία Νέου Project.

Το Project θα εμφανιστεί στο παράθυρο του Workspace (εικόνα 3)

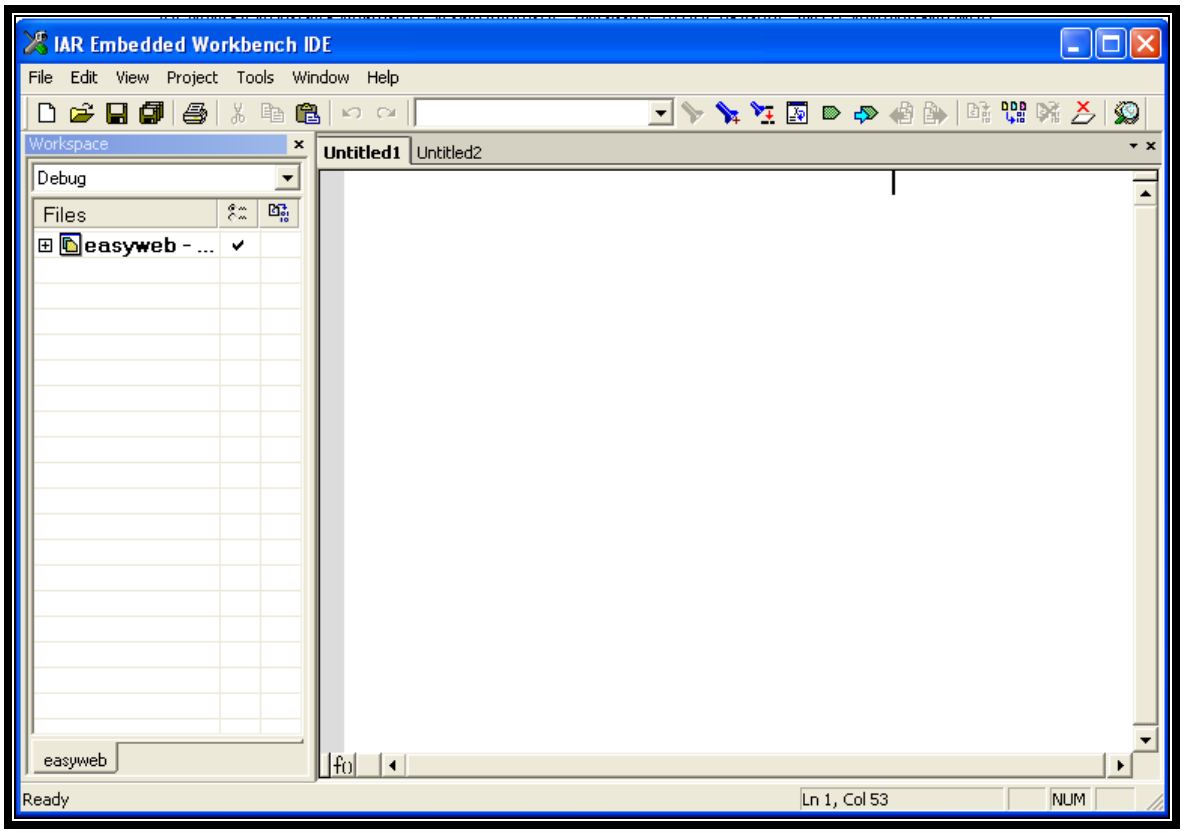


Εικόνα 3: Παράθυρο Workspace

‘Εξ ορισμού το πρόγραμμα δημιουργεί δύο build configurations: Debug και Release. Ο αστερίσκος στο όνομα του Project μας δείχνει ότι υπάρχουν αλλαγές που δεν έχουν αποθηκευτεί.

Ένα Project αρχείο (με ακρωνύμιο ewp) έχει δημιουργηθεί στον κατάλογο Projects. Αυτό το αρχείο περιέχει χρήσιμες πληροφορίες, σχετικές με το Project, όπως παραδείγματος χάριν Build options.

Για να δημιουργήσουμε ένα αρχείο κάνουμε κλικ File→New File και και είμαστε έτοιμοι να γράψουμε κώδικα.(εικόνα 4).



Εικόνα 4: Δημιουργία Κώδικα Αρχείου

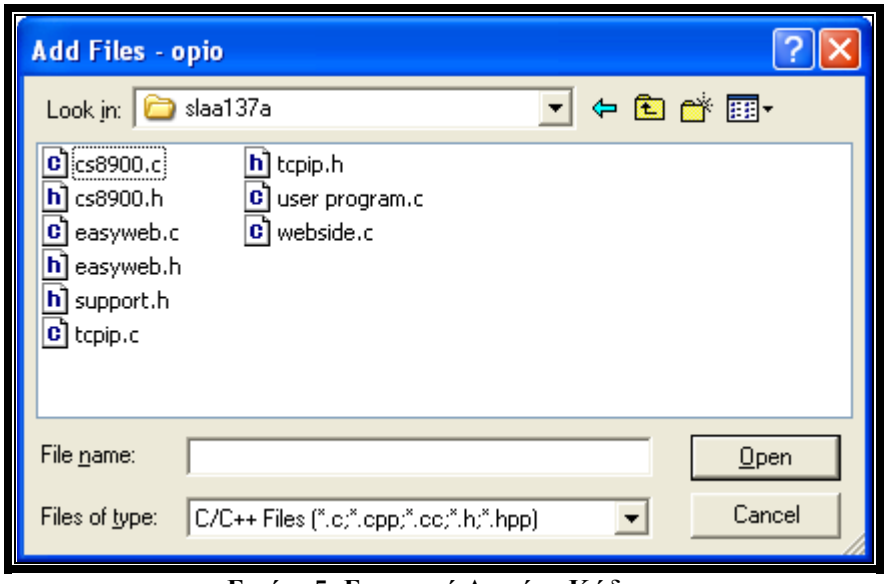
Γ.1.2 Εισάγωντας αρχεία στο Project

Το Project χρησιμοποιεί τα ακόλουθα αρχεία:

- **Cs8900a.c**
- **Cs8900.h**
- **Tcp/ip.c**
- **Tcp/ip.h**
- **Easyweb.c**
- **Easyweb.h**
- **User program.c**
- **Support.h**
- **Support.s43**

Επιλέγουμε από την μπάρα Project→Add Files για να ανοίξει ένα στάνταρ dialog box, στο οποίο θα επιλέξουμε τα αρχεία που θα χρησιμοποιήσουμε στο Project.

Επιλέγουμε ένα αρχείο την φορά και πατάμε Open.



Εικόνα 5: Εισαγωγή Αρχείων Κώδικα

Γ.1.3 Ορίζοντας τις Μεταβλητές (options) του Project

Σε αυτό το σημείο θα ορίσουμε τις μεταβλητές του Project. Αρχικά θα καθοριστούν οι γενικές παράμετροι οι οποίες αφορούν όλη τη διαδικασία Build Configuration.

Στην αρχή επιλέγουμε το όνομα του Project, που έχει εμφανιστεί στο παράθυρο Workspace , κάνουμε κλικ Project→Options και μας εμφανίζει το παρακάτω dialog box (εικόνα 6). Στην συνέχεια επιλέγουμε στο εικονίδιο Target Device:MSP430f149.

Page	Option/Setting
Target	Device: msp430F149
Output	Executable
Library	IAR CLIB (C library)

Εικόνα 6: Settings Προγράμματος

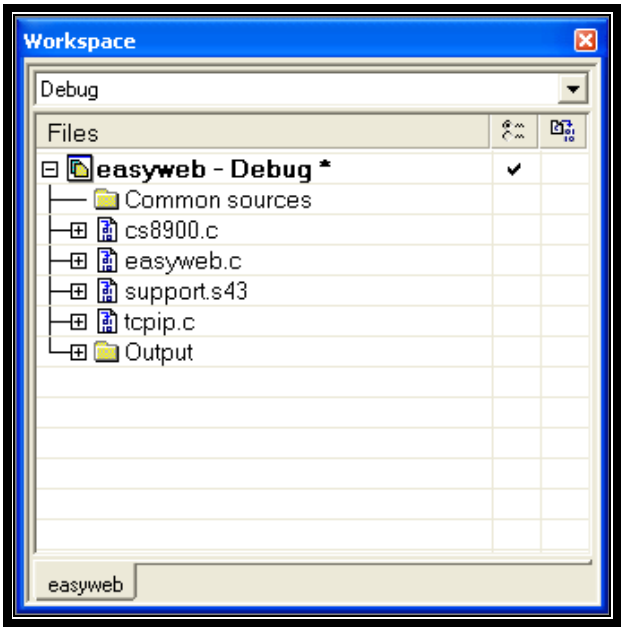
Θα πρέπει τα παραπάνω χαρακτηριστικά να έχουν αυτές τις τιμές.

Στην συνέχεια θα αρχικοποιήσουμε τις μεταβλητές του Compiler επιλέγοντας C/C++ compiler από την λίστα Category για να δούμε τα Options του Compiler (εικόνα 7)

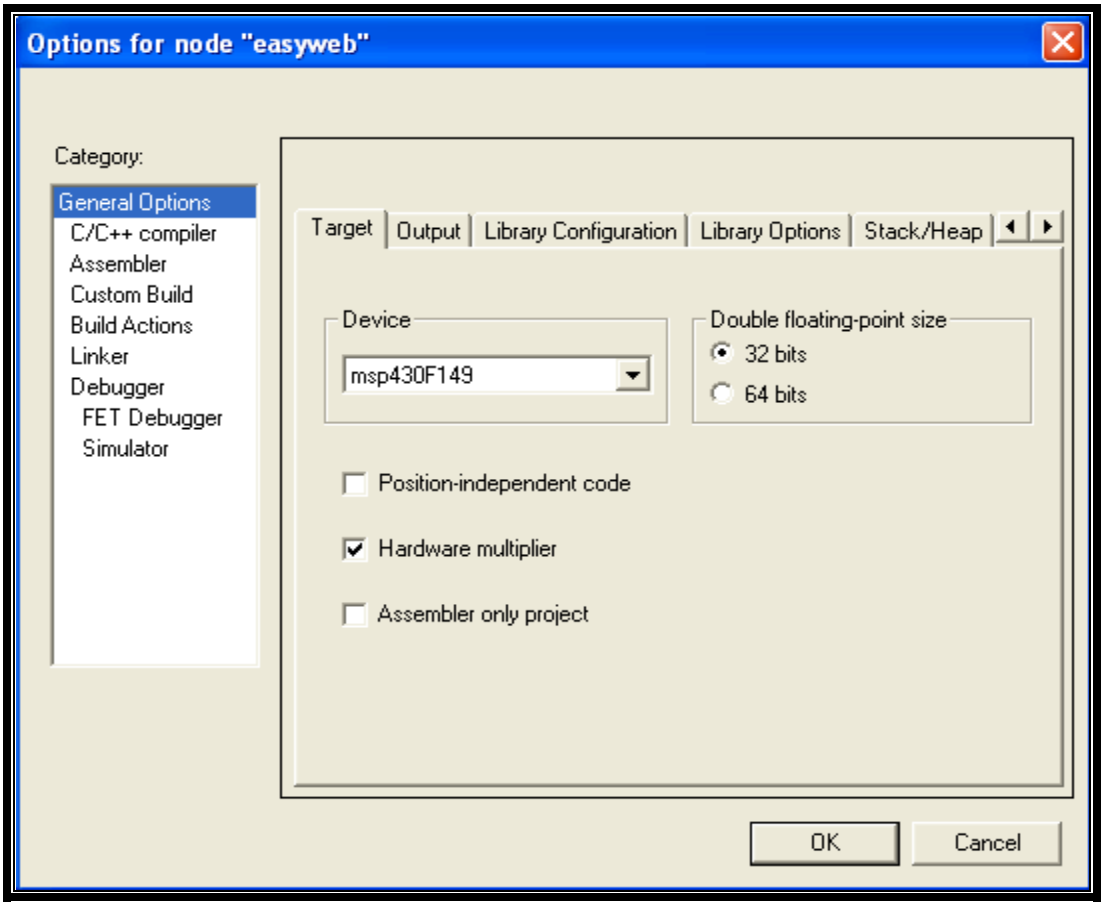
Page	Setting
Code	Optimizations, Size: None
Output	Generate debug info
List	Output list file Assembler mnemonics

Εικόνα 7: Settings Προγράμματος

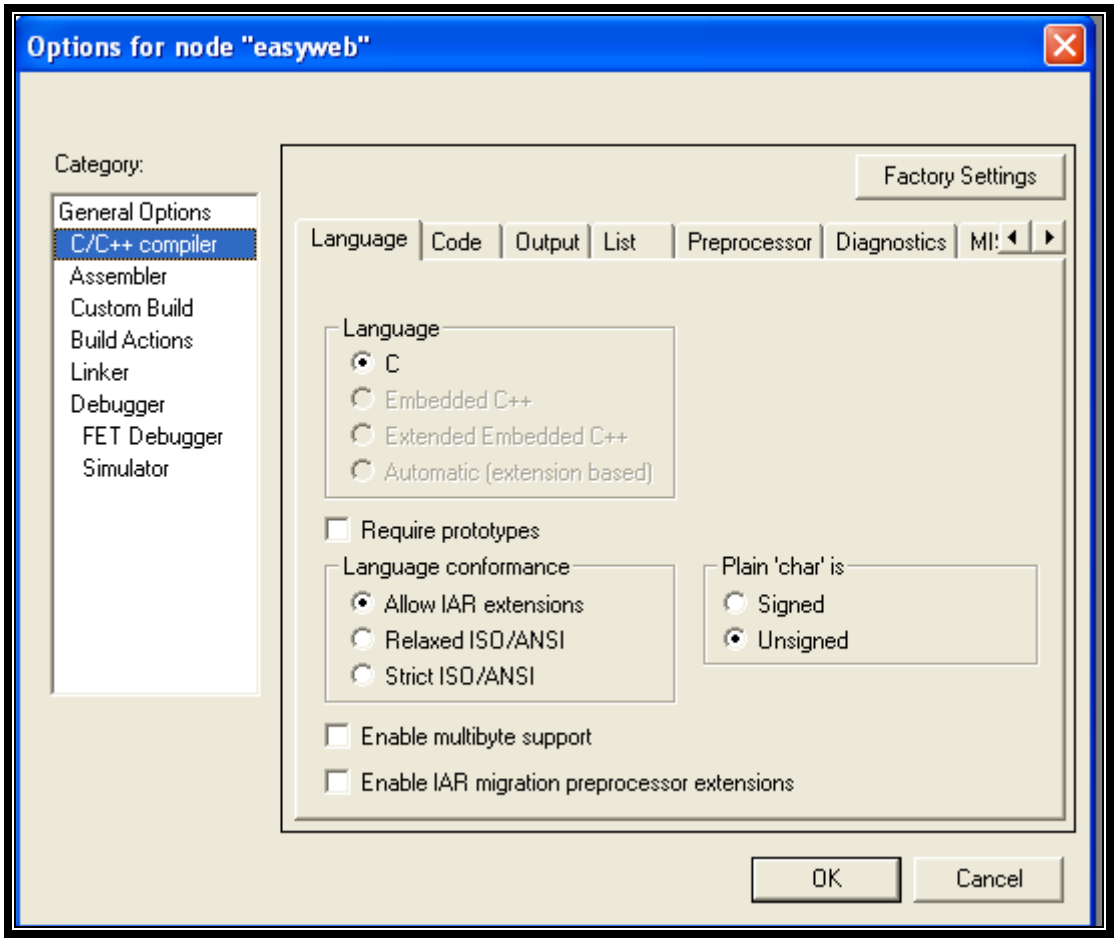
Θα πρέπει τα παραπάνω χαρακτηριστικά να έχουν αυτές τις τιμές. Τέλος πατάμε OK για να γίνει αρχικοποίηση των χαρακτηριστικών.



Εικόνα 8: Project Workspace



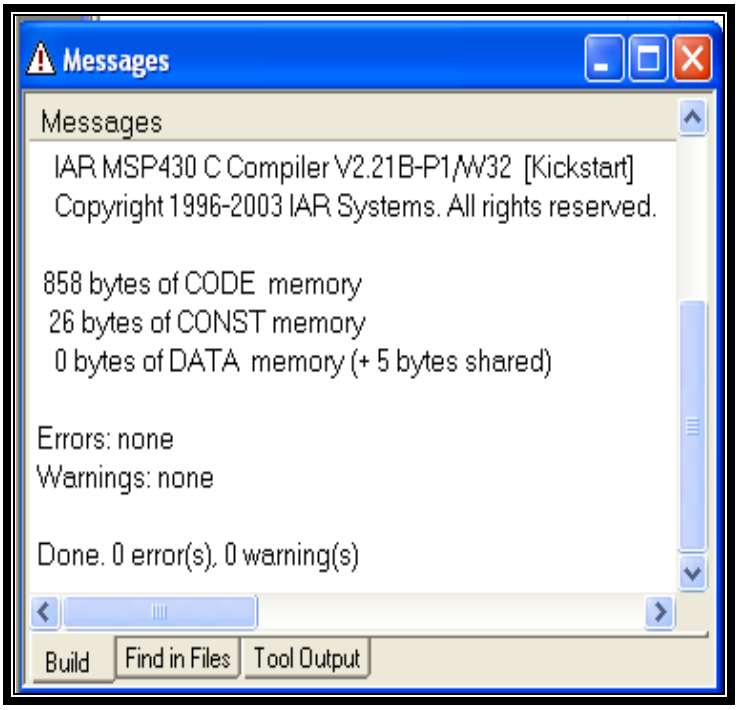
Εικόνα 9: Γενικά Χαρακτηριστικά του Project



Εικόνα 10: Αρχικοποιώντας τις μεταβλητές του Compiler

Γ.2 Μεταγλώττιση (Compiling) και Σύνδεση των Αρχείων Κώδικα

Για το Compiling των αρχείων κώδικα επιλέγουμε το αρχείο από το παράθυρο Workspace, κάνουμε δεξί κλικ επάνω σε αυτό και επιλέγουμε Compile. Αφού ολοκληρωθεί η διαδικασία του Compiling το message box θα μας εμφανίσει αν υπάρχουν λάθη ή παραλείψεις στον κώδικά μας (εικόνα 11).

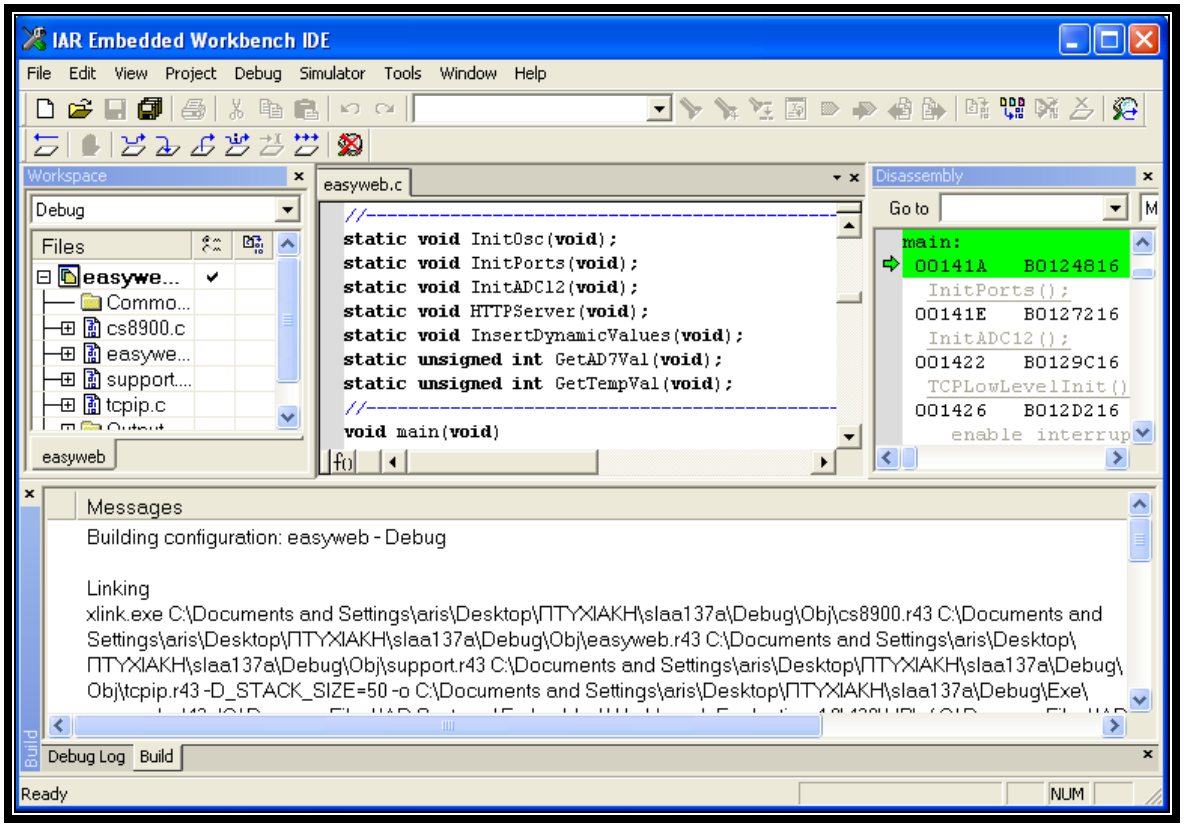


Εικόνα 11: Εικονίδιο μηνυμάτων (Message Box)

Αυτή η διαδικασία επαναλαμβάνεται για όλα τα αρχεία κώδικα ένα προς ένα.

Για την εξομοίωση σε αυτό το Project και για τον έλεγχο λαθών του κώδικα κάνουμε δεξί κλικ πάνω στο όνομά του και επιλέγουμε Rebuild All.

Στην συνέχεια το project είναι έτοιμο για debug. Κάνοντας κλίκ στο Project → Debug μεταφερόμαστε στο session του debug (εικόνα 12).



Εικόνα 12: Debug session

Τέλος κάνοντας κλικ στο Debug→Go ξεκινά η διαδικασία.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] TEXAS INSTRUMENTS MSP430 Internet Connectivity [slaa137]

[2] CS8900A Product Data Sheet Cirrus™ Logic Inc.

[3] ΔΙΚΤΥΑ ΥΠΟΛΟΓΙΣΤΩΝ ANDREW S. TANENBAUM 3η ΕΚΔΟΣΗ

[4] CS8900A Ethernet Controller Technical Reference Manual (ano083)

[5] Orcad documentation (Starting Layout in OrCAD)

[6]LAN ISOLATION TRANSFORMER CATALOG. PULSE INC. [EC 100]

[7] PULSE JACK 1x1 TAB-UP RJ45 Pulse Inc. [J402]

[8] TEXAS INSTRUMENTS MECHANICAL DATA SHEET [ipmg4]

[9] TEXAS INSTRUMENTS MSP430 Family Mixed-Signal Microcontroller Application Reports, Author: Lutz Bierl [SLAA024]

[10] TEXAS INSTRUMENTS MSP430x1xx Family User's Guide [slaa049e]

[11] TEXAS INSTRUMENTS MSP430x13x, MSP430x14x, MSP430x14x1 MIXED SIGNAL MICROCONTROLLER [slaa272f]

[12] MSP430 IAR EmbeddedWorkbench® ID_User Guide for Texas Instruments' MSP430 Microcontroller Family

[13] MSP430 IAR C/C++ CompilerReference Guidefor Texas Instruments' MSP430 Microcontroller Family

[14] KEIL Software_easyweb:Tiny TCP/IP stack & WebServer/application note APTN_164

[15] www.sparkfun.com

[16] www.iar.com

[17] www.ti.com

[18] www.gidforums.com

[19] www.sparkfun.com/cgi-bin/phpbb/