ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΠΑΡΑΡΤΗΜΑ ΧΑΝΙΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΠΡΟΓΡΑΜΜΑΤΙΖΟΝΤΑΣ ΒΙΟΜΗΧΑΝΙΚΟΥΣ ΑΥΤΟΜΑΤΙΣΜΟΥΣ ΜΕ ΤΟΝ ΕΠΕΞΕΡΓΑΣΤΗ R8C/13

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΤΩΝ ΦΟΙΤΗΤΩΝ:

Βασαλάκη Κωνσταντίνου

Ποντικάκη Αντώνιου

ΧΑΝΙΑ 2007

ΑΝΩΤΑΤΟ ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΚΡΗΤΗΣ
ΠΑΡΑΡΤΗΜΑ ΧΑΝΙΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΝΙΚΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

"Προγραμματίζοντας Βιομηχανικούς Αυτοματισμούς με τον Επεξεργαστή R8C/13"

"Programming Industrial Automatisms with the Processor R8C/13"

ΤΩΝ ΣΠΟΥΔΑΣΤΩΝ:

Βασαλάκης Κωνσταντίνος 3413
kvasal@chania.teicrete.gr

Ποντικάκης Αντώνιος 3392
pontikakis2@gmail.com

Εισηγητής: Φραγκιαδάκης Νικόλαος
nfrag@chania.teicrete.gr

ΧΑΝΙΑ 2007

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ Σελ. 6

ΚΕΦΑΛΑΙΟ 1 Σελ. 8

1.1 Κατασκευή PLC ελέγχου αντλιών Σελ. 8
1.2 Κατασκευή PLC ελέγχου καμπανών Σελ. 10

ΚΕΦΑΛΑΙΟ 2 Σελ. 12

2.1 Γλώσσα προγραμματισμού C Σελ. 12
2.2 Η δομή ενός προγράμματος στην C Σελ. 12
2.3 Σχόλια στην C Σελ. 13
2.4 Βιβλιοθήκες (#include) Σελ. 13
2.5 Λέξεις κλειδιά στην C Σελ. 14
2.6 Αριθμητικά συστήματα Σελ. 14
2.7 Τύποι δεδομένων Σελ. 15
2.8 Σταθερές και μεταβλητές Σελ. 15
2.9 Πράξεις στην C Σελ. 16
2.10 Τελεστές Σελ. 16
2.11 Συντομεύσεις Σελ. 17
2.12 Λειτουργίες στην C Σελ. 17
2.13 Έλεγχος προγράμματος Σελ. 18
2.14 Μέθοδοι επανάληψης Σελ. 20
2.15 Port registers, καταχωρητές προστασίας, καταχωρητές χρονισμού Σελ. 21
2.16 Πίνακας ASCII Σελ. 21

ΚΕΦΑΛΑΙΟ 3 Σελ. 23

3.1 Αποστολή δεδομένων στο LCD Σελ. 23
3.2 Μετακίνηση του cursor Σελ. 24
3.3 Αρχικοποίηση του LCD Σελ. 24
3.4 Αποστολή ακεραίων στο LCD (0000 ~ 9999) Σελ. 25
3.5 Αποστολή κειμένου στο LCD Σελ. 25

3.6	Τοποθέτηση του cursor σε συγκεκριμένο σημείο του LCD	Σελ. 26
3.7	Τοποθέτηση του cursor σε συγκεκριμένο σημείο του LCD (4x20)	Σελ. 26
3.8	Χρονοκαθυστέρηση (Delay)	Σελ. 26
3.9	Απεικόνιση ακεραίων (00 ~ 99) στο LCD για real time clock	Σελ. 27
3.10	Σύνδεση εξωτερικού κρυστάλλου 20MHz	Σελ. 27
3.11	Εγγραφή σε LCD 4x20	Σελ. 28
3.12	Επιλογή A/D	Σελ. 28
3.13	Αρχικοποίηση εσωτερικού ταλαντωτή	Σελ. 28
3.14	Αρχικοποίηση χρονιστή επιτήρησης	Σελ. 29
3.15	Αρχικοποίηση στα port	Σελ. 30
3.16	Απεικόνιση ώρας	Σελ. 30
3.17	Απεικόνιση κέρσορα	Σελ. 32
3.18	Μετακίνηση κέρσορα	Σελ. 34
3.19	Καθαρισμός κέρσορα	Σελ. 34
3.20	Χρονοκαθυστέρηση με τον timer X	Σελ. 35
3.21	Χρονοκαθυστέρηση με τον timer Z	Σελ. 35
3.22	Χρονοκαθυστέρηση με τον timer Y	Σελ. 36
3.23	Ενεργοποίηση εξόδων	Σελ. 37
3.24	Ενδεικτικό πρόγραμμα “εσπερινού”	Σελ. 38
3.25	Πρόγραμμα μενού	Σελ. 39
3.26	Εισαγωγή χρονοκαθυστέρησης	Σελ. 40
3.27	Bell μενού	Σελ. 41
3.28	Εκτύπωση χρονομετρητή στο LCD	Σελ. 42
3.29	Ενεργοποίηση INT0	Σελ. 42
3.30	Κύριο μενού	Σελ. 43
3.31	Αποστολή δεδομένων μέσω σειριακής	Σελ. 44
3.32	Αρχικοποίηση σειριακής	Σελ. 44

ΚΕΦΑΛΑΙΟ 4 Σελ. 45

Σχηματικά πλακετών, PCB, φωτογραφίες πλακετών

- Πλακέτα εξόδων Σελ. 45
- Πλακέτα εισόδων Σελ. 49
- Πλακέτα R8C/25 και R8C/13 Σελ. 53
- Διάταξη αυτοματισμού ελέγχου καμπανών Σελ. 54

ΚΕΦΑΛΑΙΟ 5 Σελ. 57

5.1 Περιγραφή του HEX40106B Σελ. 57
5.2 Περιγραφή του MOC3041M Σελ. 59
5.3 Περιγραφή του OCP–PCT4116/X Σελ. 60
5.4 Περιγραφή του L7805 και του L7824 Σελ. 61
5.5 Περιγραφή του BT136 Σελ. 62
5.6 Περιγραφή του BT138 Σελ. 63
5.7 Περιγραφή του HFD2 Σελ. 64
5.8 Περιγραφή του IRF540 Σελ. 64
5.9 Περιγραφή του IRF610 Σελ. 65
5.10 Περιγραφή του LCD 4x20 Σελ. 66
5.11 Περιγραφή του LCD 4x20 (σειριακή επικοινωνίας) Σελ. 68
5.12 Περιγραφή του PCF8574 Σελ. 69

ΚΕΦΑΛΑΙΟ 6 Σελ. 71

6.1 Ο δίαυλος I^2C Σελ. 71
6.2 Το πρωτόκολλο RS–232C Σελ. 79

ΚΕΦΑΛΑΙΟ 7 Σελ. 81

7.1 Προεπισκόπηση επεξεργαστή Σελ. 83
7.2 Περιγραφή επεξεργαστή Σελ. 84

ΚΕΦΑΛΑΙΟ 8 Σελ. 92

- Πρόγραμμα μεταγλώττισης Σελ. 92
- Πρόγραμμα φορτώματος HEX αρχείου στον επεξεργαστή Σελ. 95

ΒΙΒΛΙΟΓΡΑΦΙΑ Σελ. 96

ΕΥΧΑΡΙΣΤΙΕΣ Σελ. 97

ΠΡΟΛΟΓΟΣ

Για να αναπτύξουμε τις εφαρμογές αυτές να ανταποκρίνονται στις απαιτήσεις τις βιομηχανίας και του εμπορίου απευθυνθήκαμε σε δύο εταιρείες. Η πρώτη εταιρεία ασχολείται με βιομηχανικούς αυτοματισμούς και ονομάζεται INSOULA με έδρα το ΒΙΟ.ΠΑ. Χανίων (Γεώργιος Παπουτσάκης τηλέφωνο 28210 80005), ο οποίος και μας κατεύθυνε στην κατασκευή της πρώτης μας εφαρμογής:

Ένα PLC το οποίο να έχει τουλάχιστον δώδεκα εισόδους και τουλάχιστον δέκα εξόδους. Οι εισόδοι να είναι 24V και οι εξόδοι να οδηγούν ρελέ 220V.

Σαν αυτοματισμό να υλοποιήσουμε ένα σύστημα το οποίο να ξεκινάει τρεις κινητήρες τριφασικούς με εκκίνηση αστέρα τριγώνου (Δ/Υ) ο καθένας.

Να έχει διαφορετικές λειτουργίες (αυτόματο / χειροκίνητο) και στο αυτόματο να ελέγχεται από ένα floater (διακόπτης στάθμης δεξαμενής).

Στη δεύτερη επιχείρηση που απευθυνθήκαμε ήταν τα χυτήρια Παπαδάκης – Παυλουδάκης (τηλέφωνο 28210 32502), για κατασκευή συστήματος PLC ελέγχου καμπανών εκκλησιών. Το αντικείμενο αυτό αν και φαίνεται απλό στην πράξη συναντήσαμε πολλές δυσκολίες διότι για να είναι εμπορεύσιμη η κατασκευή μας πρέπει να γράψουμε στον μικροεπεξεργαστή τουλάχιστον 25 προγράμματα διαφορετικών μελωδιών, με δυνατότητα χρόνου λειτουργίας με επιλογή προγράμματος και μεταβολή των χρόνων ανάμεσα στα χτυπήματα της σφύρας.

Για να ξεκινήσουμε και τις δυο κατασκευές μας υποδείχτηκαν από τους ιδιοκτήτες τα συστήματα που ήδη χρησιμοποιούν, καθώς και το αντίστοιχο κόστος αυτών.

Σκοπός μας είναι να υλοποιήσουμε συσκευές με ακόμα περισσότερες δυνατότητες από αυτές που μας υποδείχθηκαν.

Θα πρέπει όμως να αναφέρουμε στους αναγνώστες την απειρία μας όσον αφορά τις κατασκευές, λαμβάνοντας υπόψη ότι το PLC για τον έλεγχο των αντλιών ήταν μόλις η δεύτερη μας κατασκευή. Για τον λόγω αυτό θα δείτε και ατέλειες στην κατασκευή αυτή. Δεδομένου ότι η εμφάνιση και η συναρμολόγηση των πλακετών γίνεται αποκλειστικά από εμάς, χωρίς να απευθυνθούμε σε κάποια εταιρεία.

PREFACE

In order to we develop these applications they correspond in the requirements of industry and trade we were addressed in two companies. The first company deals with industrial automatisms and is named INSOULA with seat the BIO.PA. Hania (Georgjos Papoytsakis telephone 28210 80005), who also we directed in the manufacture of our first application:

A PLC that has at least twelve entries and at least ten exits. The entries are 24V and the exits they lead relay 220V.

As automatism we materialise a system which begins three engines three-phase with departure of aster of triangle (D/Y) each one.

Have different operations (automatic/hand driven) and in automatic it is checked from floater (switch of level of reservoir).

In the second enterprise where we were addressed they were the foundries Papadakis – Payloydakis (telephone 28210 32502), for manufacture of system PLC of control of bells of churches. This object even if he appears simple in the practice we met a lot of difficulties because in order to she is marketable our manufacture it should we write in the microprocessor at least 25 programs of different melodies, with possibility of time of operation with choice of program and change of years between in the blows of hammer.

In order to we begin also the two manufactures us were indicated from the householders the systems that use already, as well as the corresponding cost of these.

Aim our is to materialise appliances with still more possibilities than those that to us were indicated.

It will be supposed however we report in the readers our inexperience with regard to the manufactures, taking into consideration that PLC for the control of pumps was hardly our second manufacture. For this reason you will see also imperfections in this manufacture. Since the appearance and the assembly boards become exclusively from us, without we are addressed in some company.

Κεφάλαιο 1

1.1 Κατασκευή PLC ελέγχου αντλιών

Το πρώτο πρόβλημα που πρέπει να επιλύσουμε είναι τι επεξεργαστή θα χρησιμοποιήσουμε. Στη διάθεση μας έχουμε δυο επεξεργαστές τους οποίους γνωρίζουμε αρκετά καλά, τον R8C/13 και τον R8C/25. Εμείς προτιμήσαμε να χρησιμοποιήσουμε τον R8C/13 ο οποίος αν και είναι SMD διατίθεται από την εταιρεία GLYN μια πλακέτα με διαστάσεις DIP32. λαμβάνοντας υπόψη την απειρία μας στις κολλήσεις SMD.

Το δεύτερο πρόβλημα που πρέπει να επιλύσουμε είναι το πώς θα αξιοποιήσουμε τα διαθέσιμα pin του μικροελεγκτή, που στην περίπτωση μας είναι μόλις 32, χωρίς όλα αυτά να είναι διαθέσιμα. Για τον λόγω αυτό σκεφτήκαμε να το υλοποιήσουμε με το πρωτόκολλο I^2C της PHILIPS. Για τον λόγω αυτό επιλέξαμε τα ολοκληρωμένα PCF8574.

Το τρίτο πρόβλημα που έχουμε να αντιμετωπίσουμε είναι ότι οι είσοδοι πρέπει να είναι 24V, δεδομένου ότι ο επεξεργαστής και τα ολοκληρωμένα που χρησιμοποιήσαμε εργάζονται με 5V. Για την λύση αυτή καταφύγαμε στους οπτοζεύκτες.

Αν και φαινότανε εύκολο στην Ελλάδα ακόμα και από μεγάλες εταιρείες δεν βρίσκαμε εύκολα οπτοζεύκτες με περισσότερα του ενός καναλιού, και όπου βρήκαμε η τιμή ήταν απαγορευτική. Τελικά η αγορά έγινε μέσω internet, όπως και τα περισσότερα εξαρτήματα, με ενδεικτικό κόστος οπτοζεύκτη (χωρίς τα μεταφορικά) 0.60€. Όσο αφορά την είσοδο του οπτοζεύκτη δεν τοποθετήσαμε κάποιο κύκλωμα προστασίας (όσο αφορά πολικότητα, κλπ) διότι τροφοδοτούμε αυτές με 24V από δικό μας τροφοδοτικό, και όχι κάποια εξωτερική πηγή. Η αντίσταση που συνδέεται σε σειρά με τη οπτική δίοδο εισόδου έχει τέτοια τιμή ώστε πάνω από 16V να θεωρείται HIGH και 12V να θεωρείται LOW.

Στην συνέχεια το επόμενο πρόβλημα που συναντήσαμε είναι η μη τετραγωνική μορφή της κυματομορφής της εξόδου (από τη μεριά του transistor) του οπτοζεύκτη, αν και στην περίπτωση μας η συχνότητα αλλαγής των εισόδων είναι πολύ αργή, για τον λόγω αυτό χρησιμοποιήσαμε τον inverter HEX40106B

με Schmitt trigger, ο οποίος λόγω της υστέρησης τετραγωνίζει οποιαδήποτε ανωμαλία στην κυματομορφή εισόδου του (ακόμη και ημιτονοειδής).

Όταν έχουμε HIGH στην είσοδο το transistor του οπτοζεύκτη οδηγείται στον κόρο με αποτέλεσμα να μας δίνει LOW στην έξοδο. Στη συνέχεια το LOW στον HEX inverter (IN) έχει ως αποτέλεσμα να μας δίνει HIGH στην έξοδο αυτού το οποίο καταλήγει στα P0 ~ P7 του PCF8594 (SLAVE), και περιμένουμε να αναγνωστεί από τον επεξεργαστή, ο οποίος έχει τον ρόλο MASTER.

Για τα κυκλώματα εξόδου χρησιμοποιήσαμε ένα PCF8594 (8bit) το οποίο έχει την δυνατότητα να ελέγξει 8 εξόδους (ρελέ). Για τον έλεγχο των ρελέ χρησιμοποιήθηκαν mosfet IRF610, καθώς και το IRF540, χωρίς κάποιο κύκλωμα ιδιαίτερο στο gate αυτών λαμβανομένου υπόψη τις χαμηλές συχνότητες ενεργοποίησης αυτών.

Στη συνέχεια αξιοποιήσαμε και τα υπόλοιπα ελεύθερα pin του μικροεπεξεργαστή με τον καλύτερο δυνατό τρόπο. Το αποτέλεσμα είναι να κατασκευάσουμε ένα PLC με 16 εισόδους και 12 εξόδους. Από τις 16 εισόδους δώσαμε την δυνατότητα επιλογής 6 αναλογικών ή 6 ψηφιακών εισόδων. Ακόμα από τα pin P0.3 ~ P0.7 οδηγήσαμε ένα LCD 4x20 χαρακτήρων σε λειτουργία four bit.

Όσο αναφορά τις εξόδους εκτός από τα οκτώ ρελέ οδηγούμε και τέσσερα MOC3041 με τα αντίστοιχα triac στην έξοδο (Solid State Relay).

Ακόμα έχουμε την δυνατότητα επέκτασης χρησιμοποιώντας μονάδες εισόδων είτε εξόδων με άλλα chip PCF8594 δυνατότητας οκτώ εισόδων ή εξόδων το καθένα.

Συγκριτικά με το PLC που μας υποδείχτηκε το οποίο δεν διέθετε ούτε καν εξωτερικό LCD και λιγότερες εισόδους και εξόδους από το την κατασκευή μας. Προκαλώντας την έκπληξη του ιδιοκτήτη κατά την διάρκεια της επίδειξης αυτού.

Επιπλέον, με τη βοήθεια του σειριακού δίαυλου και την χρήση του MAX232 έχουμε μετατροπή δεδομένων από TTL σε RS232 και δυνατότητα αποστολής δεδομένων είτε σε υπολογιστή, είτε σε κινητό τηλέφωνο βάσης, για ασύρματη μεταφορά δεδομένων.

1.2 Κατασκευή PLC ελέγχου καμπανών

Σαν κατασκευή, έχοντας αποκτήσει εμπειρία από την προηγούμενη κατασκευή, δυσκολευτήκαμε λιγότερο όσο αφορά την σχεδίαση καθώς και την εμφάνιση της πλακέτας.

Η συσκευή που μας υποδείχτηκε είχε έξι ψηφιακές εισόδους και τέσσερις αντίστοιχες ψηφιακές εξόδους, οι οποίοι οδηγούν ηλεκτρονόμους, σαν ένδειξη των καταστάσεων είχε LED.

Η κατασκευή δεν είχε την απαίτηση των 24V στην είσοδο και γενικά ήταν μια απλή κατασκευή, όμως δυσκολευτήκαμε αρκετά γιατί έπρεπε να δημιουργήσουμε ένα τρόπο επιλογής προγραμμάτων, χρόνου λειτουργίας, και εσωτερικής χρονοκαθυστέρησης. Ύστερα από πολλές προσπάθειες και δοκιμές προγραμματισμού και έχοντας σαν βοήθεια τα web site τεχνικής υποστήριξης της εταιρείας RENESAS στην Ευρώπη και στην Ασία, καθώς και την αντίστοιχη βιβλιογραφία δεδομένου ότι ο προγραμματισμός έγινε σε C++, C, έφερε το καλλίτερο δυνατό αποτέλεσμα.

Έτσι η δική μας κατασκευή μας δίνει την δυνατότητα επιλογής του επιθυμητού προγράμματος μέσω μενού και κέρσορα που εμφανίζεται στην LCD 4x20 χαρακτήρων.

Σαν επιλογή επιθυμητού χρόνου λειτουργίας ο χρήστης μπορεί να επιλέξει (μπαίνοντας στο μενού) δευτερόλεπτα, λεπτά, ώρες λειτουργίας, με μέγιστη τιμή 23 ώρες, 59 λεπτά, 59 δευτερόλεπτα.

Ακόμα πετύχαμε τις χρονοκαθυστερήσης με την βοήθεια του timer x, μεταβάλλοντας τις τιμές του προδιαιρέτη και του κύριου καταχωρητή. Οι τιμές αυτές επιλέγονται από τον χρήστη μέσω μενού και μετακίνηση κέρσορα με έξι διαφορετικά Delay.

Ακόμα χρησιμοποιούμε τον timer y σαν Real Time Clock όταν το πρόγραμμα εκτελείται, και σταματώντας το πρόγραμμα όταν οι ώρες , τα λεπτά, τα δευτερόλεπτα ταυτιστούν με τα προεπιλεγμένα.

Ακόμα χρησιμοποιούμε τον timer z για να μετράει τον χρόνο κρούσης, ο οποίος είναι 0.3sec. Στην περίπτωση αυτή θα μπορούσαμε να είχαμε χρησιμοποιήσει και την ρουτίνα delay.

Ακόμα έχουμε χρησιμοποιήσει ένα counter ο οποίος μας δείχνει τον αριθμό των χτυπημάτων.

Σαν μελλοντική επέκταση, έχοντας αποκτήσει μεγάλη εμπειρία στον προγραμματισμό σε γλώσσα C, θα μπορούμε να προσθέσουμε τον timer x όταν το πρόγραμμα δεν τρέχει (πολυδιεργασίες) να μετράει real time δίνοντας έτσι στο χρήστη την δυνατότητα να προγραμματίσει εξωτερικά τον χρόνο έναρξης, καθώς και παύσης ενός προγράμματος, απαραίτητη προϋπόθεση να μην γίνει διακοπή ρεύματος, κάτι που μπορεί εύκολα να λυθεί με τη χρήση κάποιας εξωτερικής μνήμης (ιδανική μνήμη είναι η HN58X2402SI της εταιρείας RENESAS με πρωτόκολλο επικοινωνίας I^2C). Έτσι ανά τακτά χρονικά διαστήματα θα αποθηκεύονται τιμές στην μνήμη και σε περίπτωση διακοπής ρεύματος, μετά την αποκατάσταση θα ζητούνται από αυτή.

Ακόμα μια λύση για το πρόβλημα αυτό είναι η χρήση ολοκληρωμένων κυκλωμάτων όπως το DS1307 της εταιρείας DALLAS SEMICONDUCTOR, το οποίο συνδέεται με έναν εξωτερικό κρύσταλλο στα 32KHz το οποίο λειτουργεί σαν Real Time Clock με πρωτόκολλο επικοινωνίας I^2C και απαλλάσσει τον μικροελεγκτή από την χρήση κάποιου εσωτερικού timer και εξωτερικής μνήμης.

Στην περίπτωση μας κατασκευάσαμε ένα Real Time Clock που μετράει ώρες, λεπτά, δευτερόλεπτα, αλλά θα μπορούσε εύκολα να μετρήσουμε ημέρες, εβδομάδες, χρόνια.

Κεφάλαιο 2

2.1 Γλώσσα Προγραμματισμού C

Πολλοί ηλεκτρονικοί θα έχουν χρησιμοποιήσει μικροεπεξεργαστές με επιτυχία και θα έχουν γράψει προγράμματα σε assembly. Φυσικά όσο μεγαλώνει το μέγεθος και η πολυπλοκότητα ενός τέτοιου προγράμματος τόσο περισσότερο χρειαζόμαστε ένα περιβάλλον προγραμματισμού που είναι εύχρηστο και αποτελεσματικό. Μια γλώσσα προγραμματισμού όπως η C μας προσφέρει αρκετά προτερήματα στον προγραμματισμό των μικροελεγκτών, καθώς τα προγράμματα που γράφονται σε τέτοιο περιβάλλον είναι μεταβιβάσιμα. Εν' ολίγης ο σκελετός του προγράμματος μπορεί να χρησιμοποιηθεί και από άλλους μικροελεγκτές εφόσον ορίσουμε πρώτα τις θύρες και τροποποιήσουμε τις ρυθμίσεις των καταχωρητών των συναρτήσεων. Για να μπορέσετε να προγραμματίσετε στην C θα χρειαστείτε μια εισαγωγή στην γλώσσα αυτή, μερικά παραδείγματα σύνταξης, αλλά και κάποια γνώση των τεχνικών όρων.

2.2 Η δομή ενός προγράμματος στην C

Όλα τα προγράμματα στην C αποτελούνται από διάφορα μέρη όπως σχόλια, εντολές, δηλώσεις, εκφράσεις, εκχωρήσεις και λειτουργίες. Κάθε πρόγραμμα στην C θα πρέπει να έχει μια λειτουργία η οποία ονομάζεται κύρια λειτουργία και η οποία καλείται κάθε φορά που τρέχει το πρόγραμμα. Καλό θα ήταν η κύρια ρουτίνα να αποτελείται μόνο ή κυρίως από κλήσεις λειτουργιών και όχι από τον κώδικα του προγράμματος. Μια τέτοια δομή μας βοηθάει να καταλαβαίνουμε και να τροποποιούμε πιο εύκολα το πρόγραμμα μας. Η κύρια λειτουργία ορίζεται όπως οι υπόλοιπες λειτουργίες. Όλες οι εντολές και οι συναρτήσεις που ανήκουν στο main βρίσκονται ανάμεσα σε δύο αγκύλες. Οι προγραμματιστές το ονομάζουν αλλιώς και block building δημιουργία ομάδων/πακέτων.

2.3 Σχόλια στην C

Όλες οι σειρές προγραμμάτων και φράσεις οι οποίες δεν αποτελούν μέρος του προγράμματος ονομάζονται σχόλια, τα οποία αγνοούνται από τον μεταγλωττιστή και δε καταλαμβάνουν αποθηκευτικό χώρο στην μνήμη. Σκοπός των σχολίων είναι να μας βοηθούν για το τι κάνει η κάθε εντολή, ή ομάδα εντολών για μετέπειτα επεξεργασία, έτσι δεν θα είναι δύσκολο να συντηρήσουμε και να τροποποιήσουμε το πρόγραμμα. Σχόλια τα οποία δεν ξεπερνούν τη μία γραμμή ξεκινάνε με δύο διαγώνιες κάθετες γραμμές και τελειώνουν με μια διαγώνια κάθετη γραμμή. Το ερωτηματικό (;) που χρησιμοποιούσαμε στην assembly έχει διαφορετική χρήση στην C, αφού δηλώνει το τέλος μιας εντολής.

2.4 Βιβλιοθήκες (#include)

Υπάρχουν πολλές λειτουργίες και δηλώσεις οι οποίες δεν συμπεριλαμβάνονται στην C παρόλο που θα ήταν ιδιαίτερα χρήσιμες και αναγκαίες. Πολύ συχνά θα τις βρούμε κρυμμένες μέσα σε βιβλιοθήκες και θα πρέπει να τις δηλώσουμε στον μεταγλωττιστή έτσι ώστε να τις συμπεριλάβει κατά την μεταγλώττιση. Τα αρχεία αυτά ονομάζονται header files και έχουν κατάληξη (.h).

παράδειγμα:

```
#include stdio.h
```

Το παραπάνω αρχείο χρησιμοποιείται σε προγράμματα στην C που θα πρέπει να τρέξουν σε ένα PC.

2.5 Λέξεις κλειδιά στην C

Συνολικά υπάρχουν 41 λέξεις κλειδιά στην C οι οποίες είναι οι παρακάτω:

auto	else	register	union	_bool
break	enum	return	unsigned	far
case	extern	short	void	near
char	float	signed	volatile	restrict
const	for	sizeof	while	inline
continue	goto	static	_asm	
default	if	struct	_far	
do	int	switch	_near	
double	long	typedef	asm	

Όλες οι λέξεις είναι γραμμένες με μικρά γράμματα και δεν επιτρέπεται να χρησιμοποιηθούν για άλλους σκοπούς. Πολλοί μεταγλωττιστές στην C χρησιμοποιούν επιπλέον λέξεις κλειδιά έτσι ώστε να γίνεται η καλύτερη χρήση των ιδιοτήτων του μεταγλωττιστή ή μικροελεγκτή. Για τους μικροελεγκτές R8C χρησιμοποιούνται οι παραπάνω.

2.6 Αριθμητικά συστήματα

Η γλώσσα C μπορεί να λειτουργήσει με διάφορα αριθμητικά συστήματα όπως: το δυαδικό, το δεκαδικό, το οκταδικό, και το δεκαεξαδικό. Αριθμοί χωρίς κάποιο αναγνωριστικό εκλαμβάνονται ως δεκαδικοί ενώ οι αριθμοί που ξεκινάνε με 0, 0x ή 0b εκλαμβάνονται ως οκταδικοί, δεκαεξαδικοί, ή δυαδικοί αντίστοιχα. Στην γλώσσα C χρησιμοποιείται το αγγλικό σύστημα μέτρησης και επομένως η τελεία (.) λειτουργεί ως κόμμα, και το κόμμα (,) απλός χωρίζει αριθμούς. Η άνω και κάτω τελεία (:) χαρακτηρίζει μια σειρά από αριθμούς.

Βάση	Διαθέσιμοι χαρακτήρες
10	0123456789
8	01234567
16	0123456789ABCDEF
2	01

2.7 Τύποι δεδομένων

Πριν ξεκινήσει ένα πρόγραμμα θα πρέπει να γνωρίζουμε τον αποθηκευτικό χώρο που πρέπει να καταλαμβάνει μια μεταβλητή. Βασικά πρέπει να επιλέγετε ένα τύπο ο οποίος θα καταλαμβάνει όσο το δυνατόν ελάχιστη μνήμη. Τους περισσότερους σημαντικούς τύπους δεδομένων θα τους βρείτε παρακάτω:

Τύπος	Χώρος	Περιθώριο αριθμών
_bool	8	0,1
char	8	0 ~ 255
signed char	8	-128 ~ 127
int, short	16	-32768 ~ 32767
unsigned int	16	0 ~ 65535
long	32	-2147483648 ~ 21474883647
float	32	$-1.17*10^{-38} \sim 3.4*10^{-38}$

2.8 Σταθερές και μεταβλητές

Οι σταθερές είναι αριθμοί οι οποίοι δεν αλλάζουν κατά την εκτέλεση του προγράμματος. Τέτοιοι είναι οι ακέραιοι (integer), αριθμοί κινητής υποδιαστολής (float), χαρακτήρες (char) οι οποίοι πρέπει να τοποθετούνται μέσα σε (?). Οι σταθερές δηλώνονται με #define (λέξη κλειδί). Τα ονόματα των σταθερών, μεταβλητών και λειτουργιών μπορεί να είναι όποια εμείς θέλουμε αρκεί να μην περιέχουν λέξεις κλειδιά ή σύμβολα πράξεων. Γενικά χρησιμοποιούνται μόνο λατινικοί χαρακτήρες, αριθμοί και η κάτω παύλα.

Μια μεταβλητή είναι ένα αντικείμενο το οποίο αποθηκεύεται στην μνήμη και μπορεί να είναι αριθμοί, γράμματα, ή σειρές γραμμάτων. Στην C όλες οι μεταβλητές πρέπει να δηλωθούν πριν χρησιμοποιηθούν. Η δήλωση θα πρέπει να τελειώνει με ένα ερωτηματικό. Οι μεταβλητές έχουν τιμές χωρίς πρόσημο.

2.9 Πράξεις στην C

Τα αριθμητικά σύμβολα που χρησιμοποιούνται στην C είναι τα ίδια με αυτά που χρησιμοποιούμε και στα κομπιουτεράκια. Το ίσον στην C έχει διαφορετική σημασία από ότι στα κοινά μαθηματικά. Η μεταβλητή αριστερά του ίσον παίρνει την τιμή της συνάρτησης που βρίσκεται στα δεξιά. Παρακάτω απεικονίζονται τα σύμβολα:

Σύμβολο	**Επεξήγηση**
+	πρόσθεση
-	αφαίρεση
*	πολλαπλασιασμός
/	διαίρεση

2.10 Τελεστές

Υπάρχουν δύο είδη τελεστών, είναι οι σχεσιακοί και οι λογικοί. Οι σχεσιακοί τελεστές χρησιμοποιούνται για να συγκρίνουν μεταβλητές. Το αποτέλεσμα της σύγκρισης είναι αληθής ή ψευδής. Παρακάτω απεικονίζονται οι σχεσιακοί τελεστές με την επεξήγηση τους:

Σύμβολο	**Επεξήγηση**	**Σύμβολο**	**Επεξήγηση**
>	μεγαλύτερο	<=	μικρότερο ή ίσο
>=	μεγαλύτερο ή ίσο	==	ίσο
<	μικρότερο	!=	διάφορο

Οι λογικοί τελεστές (AND), (OR), και (NOT) χρησιμοποιούνται για την εκτέλεση πράξεων της ψηφιακή λογικής. Παρακάτω απεικονίζονται οι τελεστές με τον πίνακα αληθείας για δυο τιμές σύγκρισης:

b	a	AND a&&b	OR a||b	NOT !a
0	0	0	0	1
0	1	0	1	0
1	0	0	1	1
1	1	1	1	0

2.11 Συντομεύσεις

Προκειμένου να αποφύγετε την πληκτρολόγηση ολόκληρης της συνάρτησης στην C, έχουν οριστεί συντομεύσεις που επιταχύνουν τη διαδικασία προγραμματισμού, αυτές απεικονίζονται παρακάτω:

Συντόμευση	Κανονική	Συντόμευση	Κανονική
a*=b	a=a*b	a--	a=a-1
a/=b	a=a/b	a<<=b	a=a<<b
a+=b	a=a+b	a>>=b	a=a>>b
a-=b	a=a-b	a&=b	a=a&b
a%=b	a=a%b	a|=b	a=a|b
a++	a=a+1	a^=b	a=a^b

2.12 Λειτουργίες στην C

Οι λειτουργίες αποτελούν την βάση της C και μπορούν να κληθούν από οποιαδήποτε σημείο της κύριας ρουτίνας άλλης λειτουργίας. Κάθε πρόγραμμα πρέπει οπωσδήποτε να έχει μια λειτουργία με την ονομασία main και η οποία θα καλείται με την εκκίνηση του προγράμματος. Οι λειτουργίες είναι αυτόνομα

μέρη προγράμματος που εκτελούν συγκεκριμένες εργασίες όπως αυτές που υπάρχουν στα κομπιουτεράκια. Ο γενικός σκελετός μιας λειτουργίας στην C έχει την μορφή:

Type function_name(type var1, type var2,…)

Μια λειτουργία χωρίς παραμέτρους εισόδου και εξόδου:

void wait_1(void)

Αν μια λειτουργία περιέχει πάνω από μια εντολή τότε αυτές θα πρέπει να ομαδοποιούνται σε ένα πακέτο που αποτελείται από αγκύλες και χωρίς ερωτηματικό στο τέλος.

Οι λειτουργίες μπορούν να κληθούν πληκτρολογώντας μόνο τα ονόματα τους και αυτό μπορεί να γίνει σε οποιοδήποτε σημείο του προγράμματος. Η λέξη κλειδί return έχει διαφορετική σημασία στην C από ότι στην assembly. Συγκεκριμένα το return δηλώνει την επιστροφή μιας τιμής και όχι το τέλος μιας ρουτίνας. Στην C επιτρέπεται και το nesting (φώλιασμα) το οποίο μπορεί μια λειτουργία να καλεί μια δεύτερη η οποία με την σειρά της θα καλεί μια Τρίτη κ.ο.κ..

2.13 Έλεγχος προγράμματος

Οι τρόποι ελέγχου είναι τρεις (if, if else, switch). Πολύ συχνά θα συναντάμε περιπτώσεις στις οποίες μια εντολή ή μια ομάδα εντολών θα πρέπει να εκτελεστούν εφόσον εκπληρωθεί κάποια προϋπόθεση. Όταν εκπληρωθεί η προϋπόθεση, η λειτουργία επιστρέφει την τιμή true η οποία μπορεί να είναι οποιοσδήποτε αριθμός εκτός του μηδέν. Το μηδέν ισοδυναμεί με false. Ο γενικός σκελετός της if είναι αυτός που απεικονίζεται παρακάτω:

if (condition)
    statement

Χρησιμοποιούμε την εντολή ***if…else*** όταν μια εντολή ή πακέτο εντολών εκτελείται, όταν εκπληρώνεται μια προϋπόθεση ενώ εκτελείται μια εντολή ή πακέτο εντολών όταν αυτή δεν εκπληρώνεται. Ο γενικός σκελετός της if…else είναι αυτός που απεικονίζεται παρακάτω:

```
if (condition)
        statement_1;
else
        statement_2;
```

Αν θέλουμε να εκτελεστεί μια σειρά εντολών, τότε θα πρέπει όταν εκπληρωθεί μια προϋπόθεση, να τις ομαδοποιήσουμε σε ένα πακέτο.

Αν έχουμε μια σχεσιακή τέλεση στην οποία έχουμε περισσότερα από ένα αποτελέσματα τότε να χρησιμοποιήσουμε την if…else καλύτερα θα είναι να χρησιμοποιήσουμε την switch/case με τις πολλαπλές επιλογές. Η γενική μορφή της παρουσιάζεται παρακάτω:

```
switch (variable)
{       case    constant_1;
                instructrion_1;
        case    constant_2;
                instructrion_2;
        case    constant_3;
                instructrion_3;
}
```

Η λειτουργία αυτή συγκρίνει το περιεχόμενο της μεταβλητής με την τιμή που έχει η σταθερά σε ξεχωριστή περίπτωση. Αν το περιεχόμενο της μεταβλητής και της σταθεράς είναι το ίδιο τότε εκτελείται η αντίστοιχη εντολή ή πακέτο εντολών.

2.14 Μέθοδοι επανάληψης

Οι μέθοδοι επανάληψης είναι τρεις (for, while, do…while). Το ***for*** το χρησιμοποιούμε όταν μέρος του προγράμματος πρέπει να εκτελεστεί πολλαπλές φορές. Ο σκελετός του φαίνεται παρακάτω:

```
for (start_value; end_value; step_size)
        instruction_1;
```

Η μεταβλητή μέτρησης αυξάνεται κατά την τιμή της step_size κάθε φορά που εκτελείται ο βρόγχος μέχρι ο έλεγχος της end_value να δώσει την τιμή true. Αν θελήσουμε να εκτελεστεί μια σειρά εντολών, τότε θα πρέπει να βάλουμε αγκύλες για να τις ομαδοποιήσουμε σε ένα πακέτο.

Ένας βρόγχος ***while*** θα χρησιμοποιηθεί όταν η εκτέλεση εντολών είναι εξαρτώμενη από μια προϋπόθεση. Η γενική του μορφή φαίνεται παρακάτω

```
while (condition)
        {instruction_1;
          instruction_2;
        }
```

Όταν καλείται ο βρόγχος while, ελέγχεται η τιμή της προϋπόθεσης και αν αυτή είναι αληθής εκτελούνται οι εντολές επανειλημμένα μέχρι το αποτέλεσμα της προϋπόθεσης να γίνει ψευδής.

Η ***do…while*** είναι ίδια με την while, με τη μόνη διαφορά ότι εδώ οι εντολές εκτελούνται τουλάχιστον μια φορά και στη συνέχεια γίνεται ο έλεγχος της προϋπόθεσης. Ο γενικός της τύπος φαίνεται παρακάτω:

```
do
{       instruction_1
        instruction_2
        instruction_3
}while(condition)
```

2.15 Port registers, καταχωρητές προστασίας, καταχωρητές χρονισμού

Η θύρα είναι ένα συγκεκριμένο κομμάτι μνήμης το οποίο επικοινωνεί με τις ακίδες του μικροελεκτή. Κατά την εκκίνηση του μικροελεκτή δέχεται εισερχόμενα δεδομένα. Η κατεύθυνση αυτή αλλάζει χρησιμοποιώντας τον καταχωρητή port direction (PD).

Οι καταχωρητές προστασίας φροντίζουν να μην σβηστούν με άλλα δεδομένα άλλοι σημαντικοί καταχωρητές.

Ο μικροελεκτής R8C διαθέτει δυο ταλαντωτές οι οποίοι λειτουργούν ως χρονιστές της CPU. Ο πρώτος είναι ο εσωτερικός ταλαντωτής του και ο άλλος ο εξωτερικός. Ο εξωτερικός ταλαντωτής είναι συνδεδεμένος με τις ακίδες (Xin και Xout). Οι καταχωρητές CM καθορίζουν το πώς παράγεται το σήμα χρονισμού του μικροελεκτή. Αυτό πραγματοποιείται επειδή μπορεί να χρησιμοποιηθεί ένας προδιαιρέτης για να ελλατώσουμε την συχνότητα του χρονιστή του επεξεργαστή. Ο ταλαντωτής χρησιμοποιείται μαζί με τους υπολοίπους καταχωρητές για να επιλέξουν τον χρονιστή και επιπλέον για να ελέγχει το σήμα του χρονιστή.

2.16 Πίνακας ASCII

Οι αλφαριθμητικοί χαρακτήρες καθώς και όλα τα σύμβολα που υπάρχουν για να δηλωθούν στην C, δηλώνονται κωδικοποιώντας τα με αριθμούς. Παρακάτω απεικονίζεται ο πίνακας ASCII με όλα τα σύμβολα, αριθμούς, γράμματα, τον κωδικό του αριθμό στο δεκαδικό, στο οκταδικό καθώς και στο δεκαεξαδικό σύστημα αρίθμησης. Τέλος απεικονίζεται και η κωδικοποίηση τους σε κώδικα HTML:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

128	Ç	144	É	161	í	177	▒	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	163	ú	179	│	195	├	211	╙	227	π	243	≤
131	â	147	ô	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
134	å	150	û	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	169	_	185	╣	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	171	½	187	╗	203	╦	219	▉	235	δ	251	√
139	ï	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	_
140	î	157	¥	173	¡	189	╜	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	174	«	190	╛	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	
143	Å	160	á	176	▒	192	└	208	╨	224	α	240	≡		

Κεφάλαιο 3

Σε αυτό το κεφάλαιο παρουσιάζονται κάποια χαρακτηριστικά παραδείγματα τα οποία τα έχουμε προγραμματίσει με τον συγκεκριμένο επεξεργαστή όπως για παράδειγμα αποστολή δεδομένων στο LCD, μετακίνηση του cursor, αρχικοποίηση του LCD, αποστολή ακεραίων στο LCD, αποστολή κειμένου στο LCD, σύνδεση εξωτερικού κρυστάλλου, αρχικοποίηση χρονιστή επιτήρησης, αρχικοποίηση εσωτερικού ταλαντωτή, αρχικοποίηση στα port, αποστολή ώρας στο LCD.

3.1 Αποστολή δεδομένων στο LCD

```
#include "tools1_1.h"
#include "sfr_r825.h"
//p0_0=d4, p0_1=d5, p0_2=d6, p0_3=d7,p1_1=rs,p1_2=E
void lcd_data(unsigned char data)
{
      delay(50);
      Port0=data;
      Port0=((Port0>>4)& 0X0f);        // shift 4 _0000 high dyte
      p1_1=1;                          //rs enable
      p1_2=1;                         //E enable
      p0=Port0;                      //D7-D4  High Byte transmit
      asm("nop");
      p1_2=0;                        // E=0
      p1_1=0;                        //rs=0
      Port0=data;
      Port0=(Port0 & 0X0f);        // Zero high byte
      p0=Port0;
      p1_1=1;                         //rs enable
      p1_2=1;                         //E enable
      asm("nop");
      asm("nop");
      p1_2=0;
      p1_1=0;
      delay (100);
}
```

3.2 Μετακίνηση του cursor

```
void lcd_ctrl(unsigned char data)
{     p1_1=0;
      Port0 = data;
      Port0=Port0 & 0xF0;       //zero low byte
      p0=((Port0>>4)&0X0f);    //4 shift to transmit the high byte
      p1_2=1;                  // E enable
      delay (10);
      p1_2=0;
      Port0=data;
      Port0=Port0 & 0x0F;    //zero high byte
      p0=Port0;
      p1_2=1;
      asm("nop");
      asm("nop");
      p1_2=0;
      delay (100);
}
```

3.3 Αρχικοποίηση του LCD

```
void init_lcd (void)
{
      delay(15000);
      lcd_ctrl(0x28);
      delay(5000);
      lcd_ctrl(0x28);
      delay(1000);
      lcd_ctrl(0x28);
      delay(1000);
      lcd_ctrl(0x0c);
      delay(1000);
      lcd_ctrl(0x01);
      delay(5000);
}
```

3.4 Αποστολή ακεραίων στο LCD (0000 ~ 9999)

```
void lcd_integer (unsigned int data)
{
      unsigned char byte;
      byte = data / 1000;
      data = data - byte * 1000;
      lcd_data(byte + 48);          //gia metatropi se ASCII (+48)
      byte = data / 100;
      data = data - byte * 100;
      lcd_data(byt + 48);
      byte = data / 10;
      data = data - byte * 10;
      lcd_data(byte + 48);
      lcd_data(data + 48);
}
```

3.5 Αποστολή κειμένου στο LCD

```
void lcd_text (char text[20])
{
      unsigned int s;
      s = 0;
      while ((!(text[s] == 0)) & (s < 24))  /* Gia ton logo oti
                                               yparxei LCD 2x24 */
      {
            lcd_data (text[s]);
            s = s + 1;
      }
}
```

3.6 Τοποθέτηση του cursor σε συγκεκριμένο σημείο του LCD

Έστω ότι θα τοποθετηθεί στο σημείο 1η γραμμή και 20η στήλη:

```
void lcd_pos (unsigned int seira, unsigned int stili)
{
      lcd_ctrl (0x80 + stili-1 + 0x40*(seira-1));
      delay(100);
}
```

3.7 Τοποθέτηση του cursor σε συγκεκριμένο σημείο του LCD (4x20)

```
void lcd_pos1 (unsigned int seira, unsigned int stili)
{
      lcd_ctrl (0x94 + stili-1 + 0x40*(seira-1));
      delay(100);
}
```

3.8 Χρονοκαθυστέρηση (Delay)

```
void delay(unsigned int del)
{
      unsigned int t;
      for (t = 0; t < del; t++)
      {}
}
```

3.9 Απεικόνιση ακεραίων (00 ~ 99) στο LCD για real time clock

```
void update_lcd (unsigned int data)
{
   unsigned char byte;
   byte=data/10;
   data=data-byte*10;
   lcd_data(byte+48);
   lcd_data(data+48);
   }
```

3.10 Σύνδεση εξωτερικού κρυστάλλου 20MHz

```
void ext_oszi(void)
{
    prc0 = 1;     /* Protect off */
    cm13 = 1;     /* Xin Xout */
    cm15 = 1;     /* XCIN-XCOUT drive capacity select bit : HIGH */
    cm05 = 0;     /* Xin on */
    cm16 = 0;     /* Main clock = No division mode */
    cm17 = 0;
    cm06 = 0;     /* CM16 and CM17 enable */
    asm("nop");   /* Waiting for stable of oscillation */
    asm("nop");  //asm gia ektelesi entolwn assembly(NOP NO OPERATION)
    asm("nop");  //enalaktika loop for synartish delay
    asm("nop");
    ocd2 = 0;    /* Main clock change */
    prc0 = 0;    /* Protect on */
}
```

3.11 Εγγραφή σε LCD 4x20

```
void lcd_pos_text(unsigned int seira, unsigned int stili, char text[20])
{ unsigned int i;
  if(seira==3 || seira==4)
     lcd_pos1(seira,stili);
  else
     lcd_pos(seira,stili);
  for(i=0; ((!(text[i] == 0)) & (i < 20));i++)
     lcddata (text[s]);
  }
```

3.12 Επιλογή A/D

```
unsigned int ad_in(unsigned char ch)
{
    adcon0 = 0x80 + ch;     //Port P0 group
    adcon1 = 0x28;          //10-bit mode
    adst = 1;               //Conversion start
    while(adst == 1);      /*Wait A/D conversion ENALAKTIKA MPOROYMAI
                       NA XRISIMOPOIHSOYMAI THN SHMAIA TOY INTERRUPT */
    return ad;              //AD value
}
```

3.13 Αρχικοποίηση εσωτερικού ταλαντωτή

```
void int_osc(void)
{
prc0 = 1;          // Look disable   BIT kataxwriti prostasias
hr00 = 1;          // high-speed on-chip oscillator on
hr1=68;            // Fall 38400 = 38610
asm("NOP");
asm("NOP");
asm("NOP");
hr01 = 1;
cm06 = 0;
cm16 = 0;
cm17 = 0;
prc0 = 0;                     // Look enable
}
```

3.14 Αρχικοποίηση χρονιστή επιτήρησης

```
#include "sfr_r825.h"         // register definition
#include "watchdog_timer.h"
void wdt_init(void)
{
    /* switch BCLK to f8 so we have a longer timeout period */
    /* unprotect Clock mode register0 so we can modify clock */
    prcr = 3;
    /* foco_s  cloc_pulse from_low speed_oscilator */
    cspro=0;
      /*Low on speed oscilator                          */
      //*******for the watch dog timer*******************
      //*******foco selsected************************//
      cm07=0;
      cm14=0;
    /* Watchdog function after it times out:0 -(default) generate
       interrupt,  1  - resets the MCU */
    pm12 = 0;
        /* 0 (default) - jumps to the watchdog interrupt routine */
        /* 1           - resets the MCU */
    /* protect PM1 */
    prcr = 0;
    /* prescaler is div by 128 watchdog
        timer period = (32,768 x 128)/(2.5MHz)=1.677sec */
    wdc7 = 1; // dia 128
    /* Start Watchdog Timer by writing any value to wdts
       register (value always resets to 0x7fff when written to) */
    wdtr = 0x00;    // prwta pragmatopoihtai midenismos
    wdtr = 0xFF;    // timi metrisis
    wdts = 0x00;    // enarxi metrisis
}
```

3.15 Αρχικοποίηση στα port

```
void init_port(void)
{
      pd1_0=0;    //input
      pd1_1=0;    //input
      pd1_2=0;    //input
      pd1_3=0;    //input
      pd1_4=0;
      pd1_5=0;    //input
      pd1_6=0;    //input
      pd1_7=0;    //input interrupt INT1
      pd3_2=0;    //input
      pd3_1=1;    //output
      pd3_0=1;
      pd3_3=0;    //input
      pd3_7=1;    //output
      pd4_5=0;    //input
}
```

3.16 Απεικόνιση ώρας

```
void init_timerx(void)
{
    txmr    = 0x00;     /* Timer mode for Timer X */
    prex    = 250-1;    /* Set Prescaler X register */
    tx      = 250-1;    /* Set Timer X register to  */
    txck0   = 0;        /* divide    32 */
    txck1   = 1;        /* f1 f2 f8 f32 */
                        /* 00 11 01 10  */
    txic=0x00;         /* Timer interrupt priority level */
    txs=0;             /* Timer X count start flag -> start */
    days=0;
    counter12=0;      //h dhlwli twn metavlitwn exei ginei stin arxi
    seconds=0;
    minutes=0;
    hours=0;
    days=0;
}

//ektypwsi tiw wras sto LCD
```

```
void update_timer()
{
  lcd_pos1(2,10);
  update_lcd(days);
  lcd_text(":");
  update_lcd(hours);
  lcd_text(":");
  update_lcd(minutes);
  lcd_text(":");
  update_lcd(seconds);
  delay(1000);
}

void update_lcd (unsigned int data)
{
   unsigned char byte;
   byte= data/10;
   data=data - byte*10;
   lcd_data(byt+48);
   lcd_data(data+48);
   }

// ROUTINA INTERRUPT TIMER X
void timerx_int()
{
   if (counter12>=10)
   {   counter12=0;
       seconds++;
   }
   else counter12++;
   if(seconds>=60)
   {   seconds=0;
       minutes++;
   }
   if(minutes>=60)
   {   minutes=0;
       hours++;}
   if(hours>=60)
   {    hours=0;
        days++;
   }
   if(days>99)
      days=0;
}
```

3.17 Απεικόνιση κέρσορα

```
void delay_cursor_check(void)
{ if(cursor1==1 || cursor==0)
     {lcd_pos(2,1);
        lcddata(0xff);
        lcd_pos1(1,1);
        lcd_text(" ");
        lcd_pos1(2,1);
        lcd_text(" ");
        lcd_pos(2,11);
        lcd_text(" ");
        lcd_pos1(1,11);
        lcd_text(" ");
        lcd_pos1(2,11);
        lcd_text(" ");
        }
    else if(cursor1==2)
      {lcd_pos1(1,1);
        lcddata(0xff);
        lcd_pos(2,1);
        lcd_text(" ");
        lcd_pos1(2,1);
        lcd_text(" ");
        lcd_pos(2,11);
        lcd_text(" ");
        lcd_pos1(1,11);
        lcd_text(" ");
        lcd_pos1(2,11);
        lcd_text(" ");
        }
     else if(cursor1==3)
       {lcd_pos1(2,1);
        lcddata(0xff);
        lcd_pos1(1,1);
        lcd_text(" ");
        lcd_pos(2,1);
        lcd_text(" ");
        lcd_pos(2,11);
        lcd_text(" ");
        lcd_pos1(1,11);
        lcd_text(" ");
        lcd_pos1(2,11);
        lcd_text(" ");
```

```
        }
      else if(cursor1==4)
        {lcd_pos(2,11);
         lcddata(0xff);
         lcd_pos1(1,1);
         lcd_text(" ");
         lcd_pos1(2,1);
         lcd_text(" ");
         lcd_pos(2,1);
         lcd_text(" ");
         lcd_pos1(1,11);
         lcd_text(" ");
         lcd_pos1(2,11);
         lcd_text(" ");
         }
      else if(cursor1==5)
         {lcd_pos1(1,11);
         lcddata(0xff);
         lcd_pos1(1,1);
         lcd_text(" ");
         lcd_pos(2,1);
         lcd_text(" ");
         lcd_pos(2,11);
         lcd_text(" ");
         lcd_pos(2,11);
         lcd_text(" ");
         lcd_pos1(2,11);
         lcd_text(" ");
         }
       else if(cursor1==6)
         {lcd_pos1(2,11);
         lcddata(0xff);
         lcd_pos1(1,1);
         lcd_text(" ");
         lcd_pos(2,1);
         lcd_text(" ");
         lcd_pos(2,11);
         lcd_text(" ");
         lcd_pos1(1,11);
         lcd_text(" ");
         lcd_pos1(2,1);
         lcd_text(" ");
         }
}
```

3.18 Μετακίνηση κέρσορα

```
void cursor1_move(void)
{if(p1_2)
    {while(p1_2){}  /*wait to be 0*/
        {if(cursor1>6)
                     cursor1=1;
         else
                   cursor1++;
         }
     }
else if(p1_1)
     {while(p1_1){} /*wait to be 0*/
        {if(cursor1<1)
                     cursor1=6;
         else
                  cursor1--;
         }
      }
}
```

3.19 Καθαρισμός κέρσορα

```
void clear_cursor(void)
{
 lcd_pos(2,1);
 lcd_text(" ");
 lcd_pos1(1,1);
 lcd_text(" ");
 lcd_pos1(2,1);
 lcd_text(" ");
 lcd_pos(2,11);
 lcd_text(" ");
 lcd_pos1(1,11);
 lcd_text(" ");
 lcd_pos1(2,11);
 lcd_text(" ");
 delay(50000);
 }
```

3.20 Χρονοκαθυστέρηση με τον timer x

```
//delay routine for timer X gives 0.001 Sec delay ->seconds_delay=1000
gives 1 Sec delay
void delay_timerx(unsigned int seconds_delay)
{
   do
    {txmod0       = 0;        /* Timer mode for Timer x */
    txmod1 = 0;         /*                         */
    txic    = 0x00;           /* Timer interrupt priority level */
      ir_txic=0x00;//zero interrupt flag
      txs = 0;            //stop counting
    prex    = data0;    /* Set Prescaler y register */
    tx      = data1;        /* Set Timer y register to  */

/*    txck0 = 0;        /* divide    1 */
/*    txck1 = 0;        /* f1 f2 f8 f32 */
                                  /* 00 11 01 10     0.1Sec overflow
(250*160/8)*20MHz=0.1Sec */
    txs           =1;         /* Timer X count start flag -> start */

      while(!ir_txic); //wait to be 1
    seconds_delay--;
      if(p0_1==0)
    break;
      }
      while(seconds_delay!=0);


      }
```

3.21 Χρονοκαθυστέρηση με τον timer z

```
void delay_timerz(unsigned int seconds_delay)
{ tzmod0    = 0;        /* Timer mode for Timer z */
    tzmod1 = 0;         /*                         */
      tzic  = 0x00;           /* Timer interrupt priority level */
      tzck0=0;
      tzck1=0;           //f1 selected
   do
    {
```

```
      ir_tzic=0x00;//zero interrupt flag
      tzs = 0;            //stop counting
    prez    = 250-1;    /* Set Prescaler z register */
    tzpr    = 80-1;         /*  Set  Timer  z  register  to  primary
register */
/*    txck0 = 0;        /* divide    1 */
/*    txck1 = 0;        /* f1 f2 f8 f32 */
                                  /* 00 11 01 10     0.1Sec overflow
(250*160/8)*20MHz=0.1Sec */
    tzs           =1;         /* Timer X count start flag -> start */

      while(!ir_tzic); //wait to be 1
    seconds_delay--;
      }
      while(seconds_delay!=0);
      }
```

3.22 Χρονομετρητής με τον timer y

```
//Timer y interrupt program*********
void timery_control(void)
 { if(timer==99)
     {timer=0;
        if(seconds1>59)
            { seconds1=0x00;
                minutes1++;
                       if(minutes1>59)
                                       {minutes1=0x00;
                                              hours1++;
                                                }
                  }
      else seconds1++;
     update_timer(hours1,minutes1,seconds1);
        if(hours1==hours && seconds1==seconds && minutes1==delay1)
           { state=OFF;
               tyic=0x00;//interupt priority level
               tys=0; //stop the counter
               seconds1=0;
               minutes1=0;
               hours1=0;
               }
       }  else timer++;
 }
```

3.23 Ενεργοποίηση εξόδων

```
void bell1_on(void)
{lcd_pos(2,6);
 lcddata(0xff);//turn this point black
 p3_0=1;
 }

 void bell1_off(void)
 { lcd_pos(2,6);
   lcd_text(" ");
   p3_0=0;
   }

 void bell2_on(void)
 { lcd_pos(2,13);
   lcddata(0xff);//turn this point black
   p3_1=1;
  }

  void bell2_off(void)
   { lcd_pos(2,13);
        lcd_text(" ");
        p3_1=0;
   }

  void bell3_on(void)
   { lcd_pos(2,20);
       lcddata(0xff);//turn this point black
      p3_2=1;
      }

  void bell3_off(void)
    { lcd_pos(2,20);
        lcd_text(" ");
        p3_2=0;
        }

  void bell4_on(void)
    {lcd_pos1(1,6);
       lcddata(0xff);//turn this point black
     p3_3=1;
```

```
      }

      void bell4_off(void)
      { lcd_pos1(1,6);
        lcd_text(" ");
        p3_3=0;
      }

  void bell5_on(void)
  {lcd_pos1(1,13);
   lcddata(0xff);//turn this point black
   p1_0=1;
   }

   void bell5_off(void)
   {lcd_pos1(1,13);
      lcd_text(" ");
      p1_0=0;
      }
```

3.24 Ενδεικτικό πρόγραμμα “εσπερινού”

```
void adam4(void)
   {int i;
    ir_tyic=0;
          tys=1;
            tyic=0x03;//piritylevel
    bell_menu();
   while(state==ON || p0_1==0){       //ON=1,  OFF=0
                      for(i=0;i<3 || p0_1==1;i++)
                                    { bell4_on();
                                       delay_timerz(400);
                                         bell4_off();
                                         delay_timerx(400);
                                     }
                                         delay_timerz(800);
                      for(i=0;i<3 || p0_1==1;i++)
                                     { bell4_on();
                                       delay_timerz(400);
                                        bell4_off();
                                        delay_timerx(400);
                                      }
                                               delay_timerx(800);
```

```
                    for(i=0;i<7 || p0_1==1;i++)
                                { bell4_on();
                                  delay_timerz(400);
                                  bell4_off();
                                  delay_timerx(400);
                                 }
                                      delay_timerx(800);
                     }
                               automatic1_display();
                               state=ON;
                     int0ic = 0x05;
  }
```

3.25 Πρόγραμμα μενού

```
void delay_menu(void)
{ clear_display();
  lcd_pos(1,1);
  lcd_text("**TIME DELAY MENU**");
  lcd_pos(2,2);
  lcd_text("DELAY 0 ");
  lcd_pos1(1,2);
  lcd_text("DELAY 1 ");
  lcd_pos1(2,2);
  lcd_text("DELAY 2 ");
  lcd_pos(2,12);
  lcd_text("DELAY 3 ");
  lcd_pos1(1,12);
  lcd_text("DELAY 4 ");
  lcd_pos1(2,12);
  lcd_text("DELAY 5 ");
  }
```

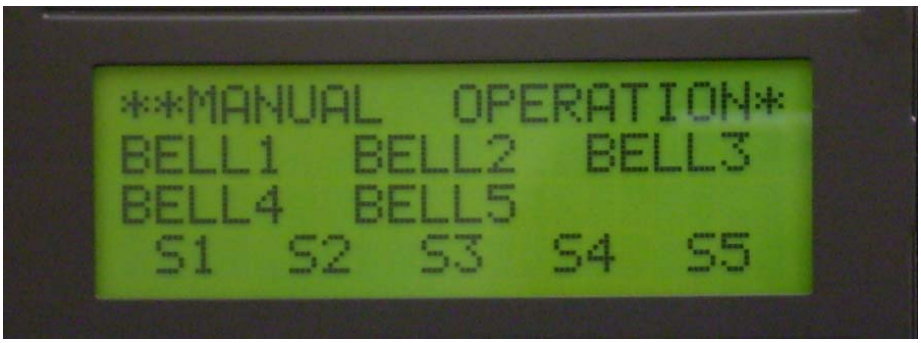
3.26 Εισαγωγή χρονοκαθυστέρησης

```
void time_program_select(void)
{

 if(p1_2)
           { while(p1_2){}//wait to depress the switch
           if(seconds>50)
                      {if(minutes>59)
                                {minutes=0;
                                     hours++;}
                     else
                                 minutes++;
                              seconds=0;
                     update_timer(hours,minutes,seconds);
                               }

         else  {
                       seconds=seconds+10;
                  update_timer(hours,minutes,seconds);
                          }
           }
 else if(p1_1)
              {while(p1_1){}
                     if(seconds==0)
                         {if(minutes<0)
                                   {minutes=59;
                                        hours--;
                            update_timer(hours,minutes,seconds);
                                        }

                            else
                                   minutes=minutes-1;
                           seconds=50;
                           lcd_pos1(2,1);
                  update_timer(hours,minutes,seconds);
                          }
                 else
                    {seconds=seconds-10;
                  update_timer(hours,minutes,seconds);
                          }
                 }
else if(!p0_1)
              {while(!p0_1);
```

```
                    delay(5000);
                   if(hours==23)
                      {hours=0;
                         update_timer(hours,minutes,seconds);
                         }
             else
                       {hours++;
                          update_timer(hours,minutes,seconds);
                          }
        }
}
```

3.27 Bell μενού

```
   //*************BELL MENU***************
  void bell_menu(void)
  {  lcd_pos(1,1);
     Lcd_text("**MANUAL  OPERATION*");
    lcd_pos(2,1);
      lcd_text("BELL1  BELL2  BELL3  ");
      lcd_pos1(1,1);
      lcd_text("BELL4  BELL5         ");
      lcd_pos1(2,1);
      lcd_text(" S1  S2  S3  S4  S5  ");
  }
```



3.28 Εκτύπωση χρονομετρητή στο LCD

```
void update_timer(int hours,int minutes,int seconds)
{lcd_pos1(2,1);
update_lcd(hours);
lcd_text(":");
update_lcd(minutes);
lcd_text(":");
update_lcd(seconds);
lcd_text("               ");
}
```

3.29 Ενεργοποίηση INT0

```
void int0_enable(void)
{int0f0=0;
 int0f1=1;  //filter enable f8 sample
int0ic = 0x05;
int0en = 1;
asm( "\tFSET      I");        /* Enable interrupt */
}
```

3.30 Κύριο μενού

```
void lcd_menu(void)
{ clear_display(); //clear display
  clear_display(); //clear display
  lcd_pos(1,1);
  lcd_text("*****MAIN MENU******");
  lcd_pos(2,1);
  lcd_text(" MANUALL");
  lcd_pos1(1,1);
  lcd_text(" PROGRAM 1");
  lcd_pos1(2,1);
  lcd_text(" PROGRAM 2");
  lcd_pos(2,12);
  lcd_text("PROGRAM 3");
  lcd_pos1(1,12);
  lcd_text("PROGRAM 4");
  lcd_pos1(2,12);
  lcd_text("PROGRAM 5");
  }
```

3.31 Αποστολή δεδομένων μέσω σειριακής

```
void send_data(unsigned char data)
{
        u0tbl = data;
        te_u0c1 = 1;          /* Transmission enabled */
    flag_energ = 1;          /* Set transmission flag */
        while(ir_s0tic)      /* Wait till transmission complete*/
        ir_s0tic = 0;       /* Clear of the serial transmission flag */
    flag_energ = 0;        /* Transmission completed */
    te_u0c1 = 0;          /* Transmission disabled */
}
```

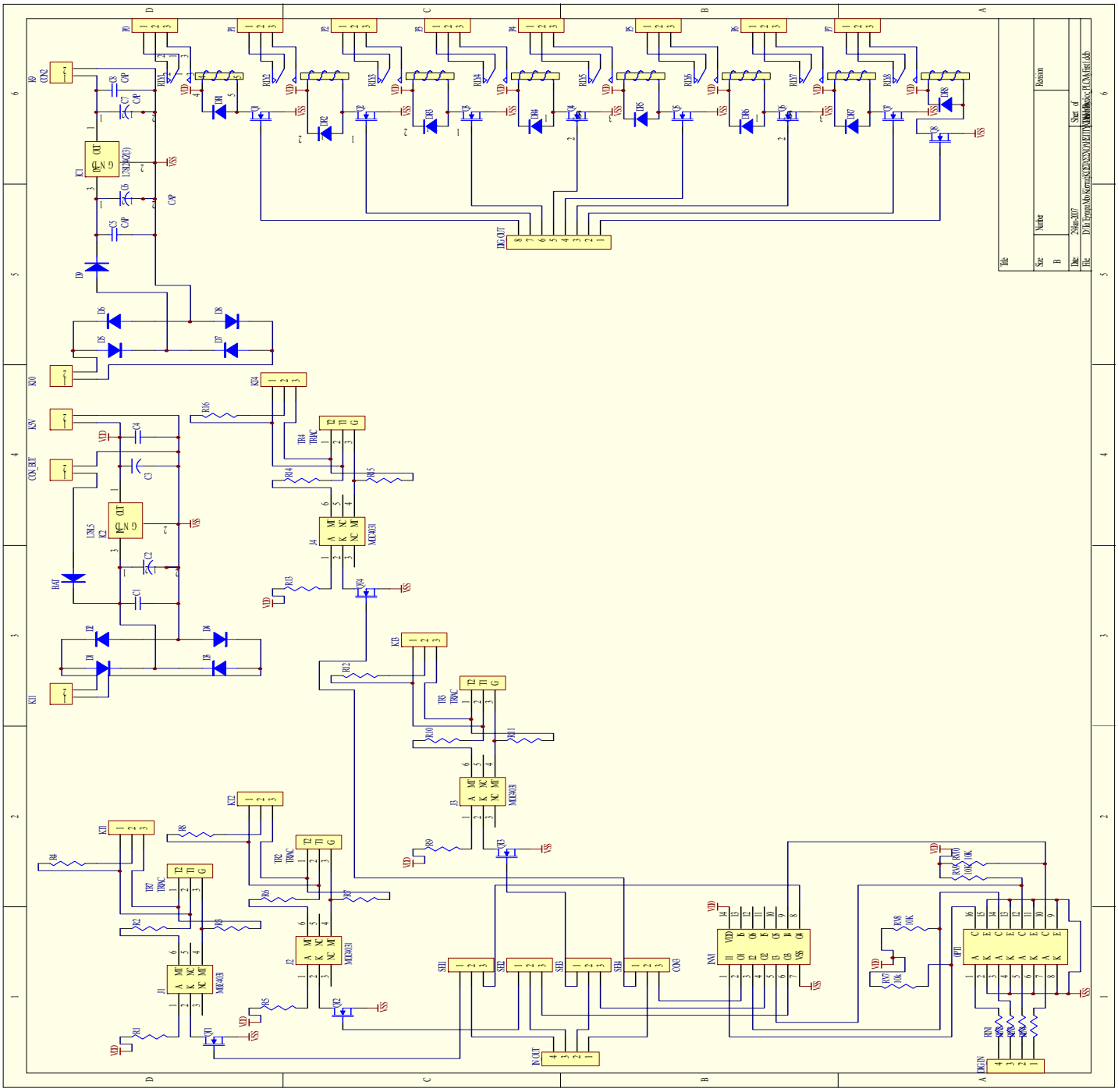
3.32 Αρχικοποίηση σειριακής

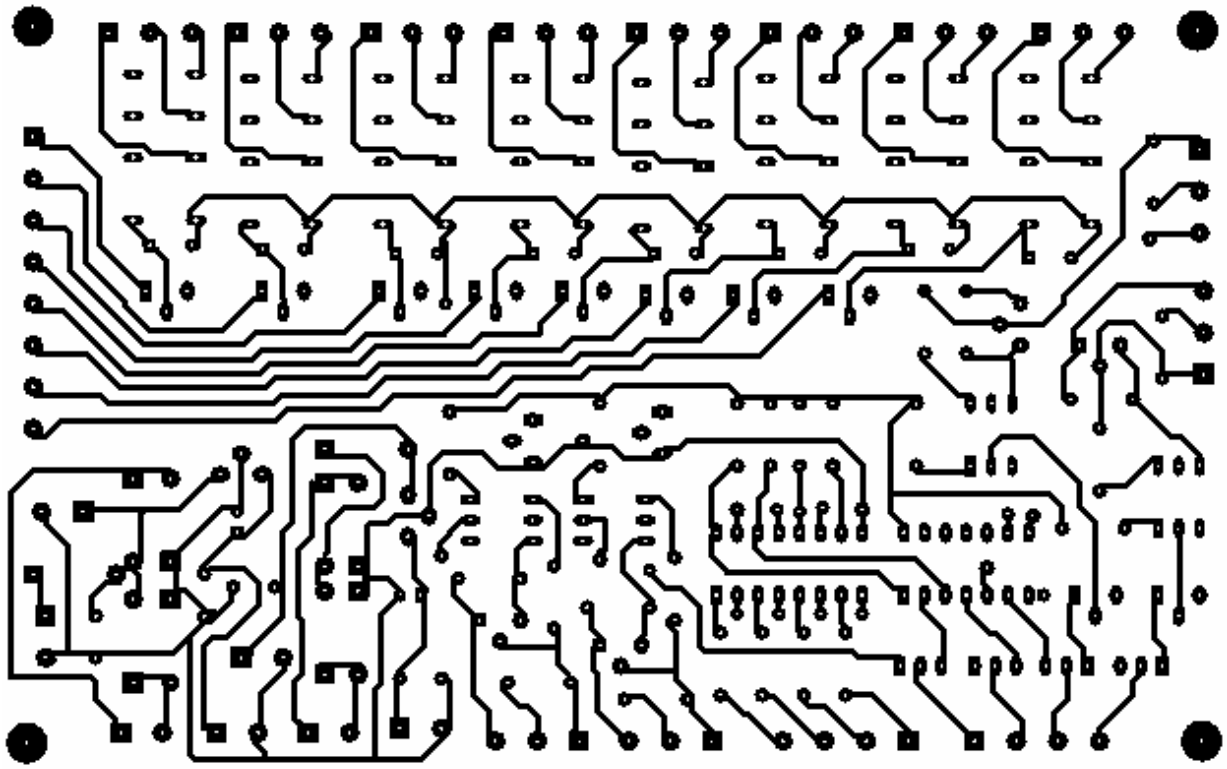
```
void init_uart0(void)
{
/* Setting port registers */
    p1 = p1 | 0x10;            /* TxD0 port off */

/* Setting port direction registers */
    pd1 = pd1 | 0x10;         /* TxD0 port direction = output */
    pd1 = pd1 & 0xdf;         /* RxD0 port direction = input */

    txck0 = 1;                /* Timer X count source = f8 */
    txck1 = 0;

/* Clock synchronous serial I/O setting */
    u0mr = 0x25;  /* Setting UART0 transmit/receive mode register */
/* b2-b0:Serial I/O mode select bit */
/* (101:Transfer data 8 bits long) */
/* b3:Internal/external clock select bit (0:Internal clock) */
/* b4:Stop bit length select bit (0:One stop bit) */
/* b5:Odd/even parity select bit (1:Even parity bit) */
/* b6:Parity enable bit (1:Parity enable) */
/* b7:Reserved bit (Must always be set to "0") */
   u0c0 = 0x00;/* Setting UART0 transmit/receive control register 0 */
/* b1-b0:BRG count source select bit (00:f1 is selected) */
/* b2:Reserved bit (Must always be set to "0" */
/* b3:Transmit register empty flag (Write disabled) */
/* b4:Nothing is assigned */
/* b5:Data output select bit (0:TxDi pin is CMOS output) */
/* b6:CLK polarity select bit */
/* (Must always be "0" in UART mode */
/* b7:Transfer format select bit */
/* (Must always be "0" in UART mode */
/* b7:Nothing is assigned */
    u0rrm = 0;      /* Continuous receive mode disabled */
    u0brg = 104-1;  /* Setting UART0 baud rate generator */
                    /* (Approx. 9600bps @16MHz,f1) */
    re_u0c1 = 1;    /* Reception enabled */
}
```

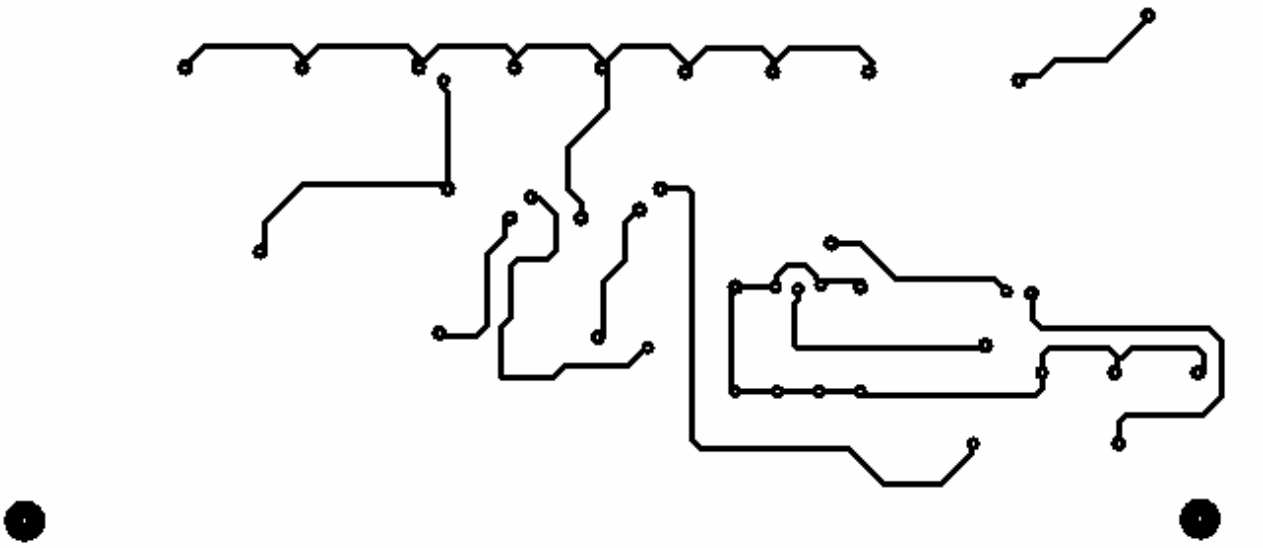
Κεφάλαιο 4

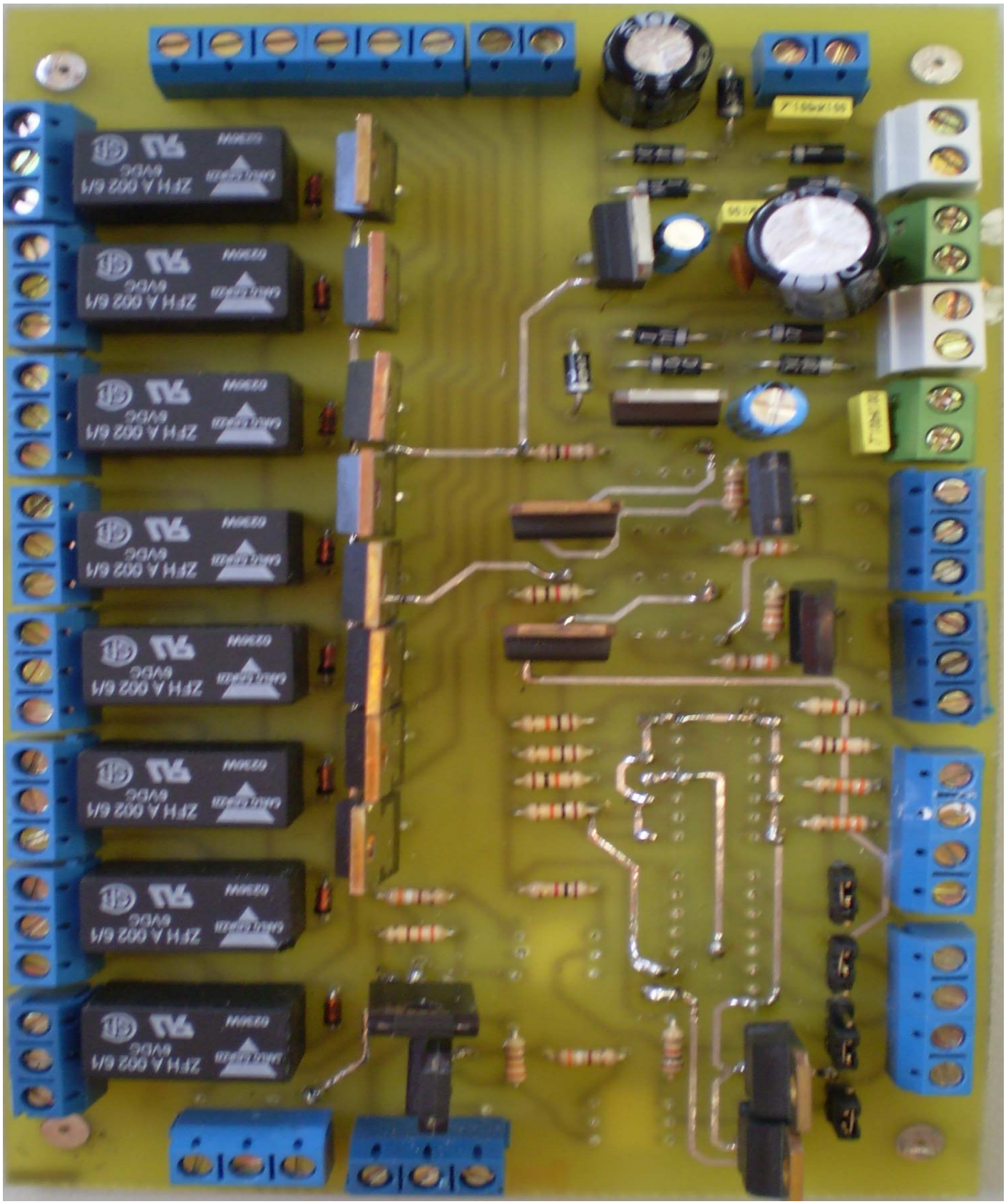
Σχηματικά πλακετών (PCB)

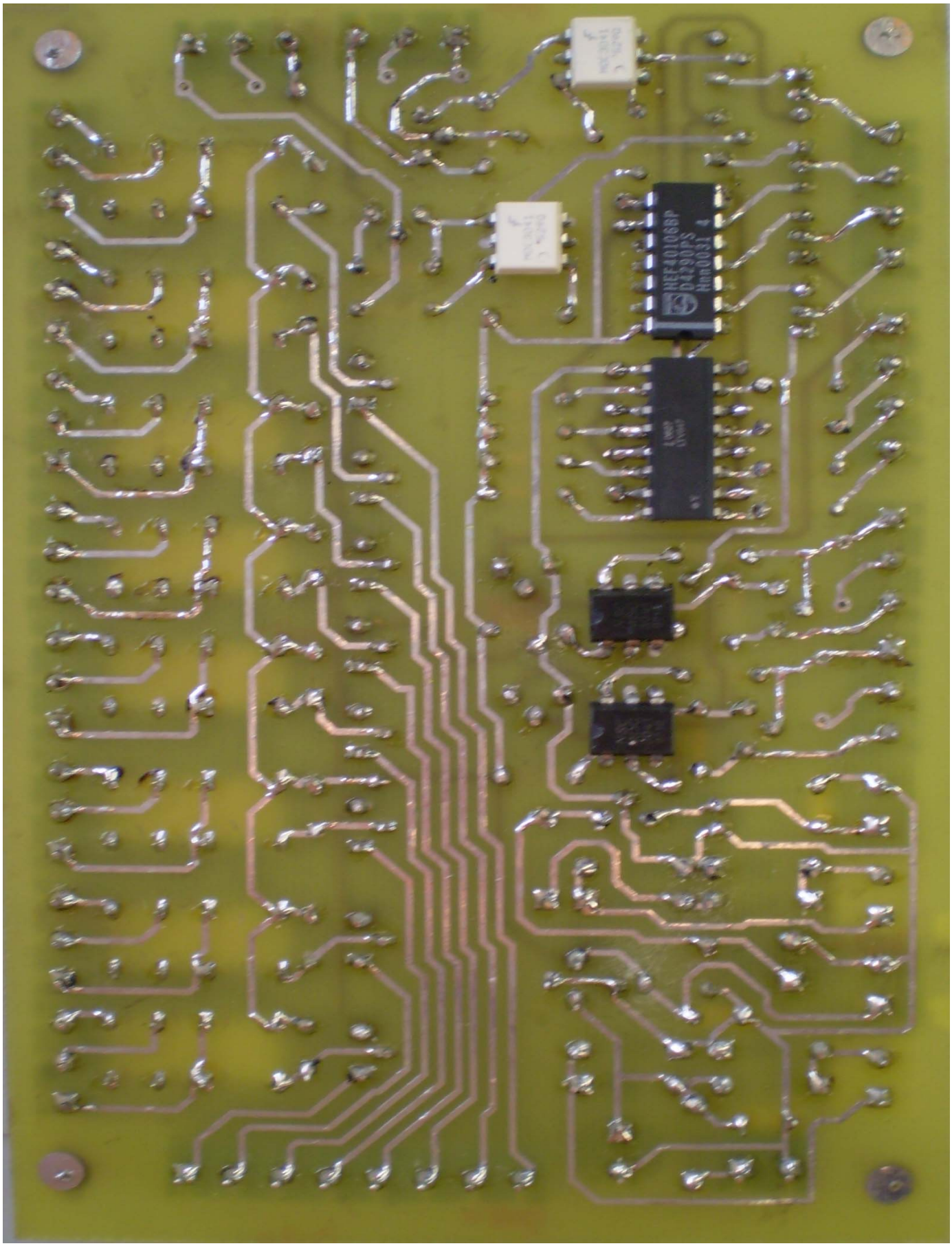
Παρακάτω απεικονίζεται η ***πλακέτα εξόδων*** (τροφοδοτική).



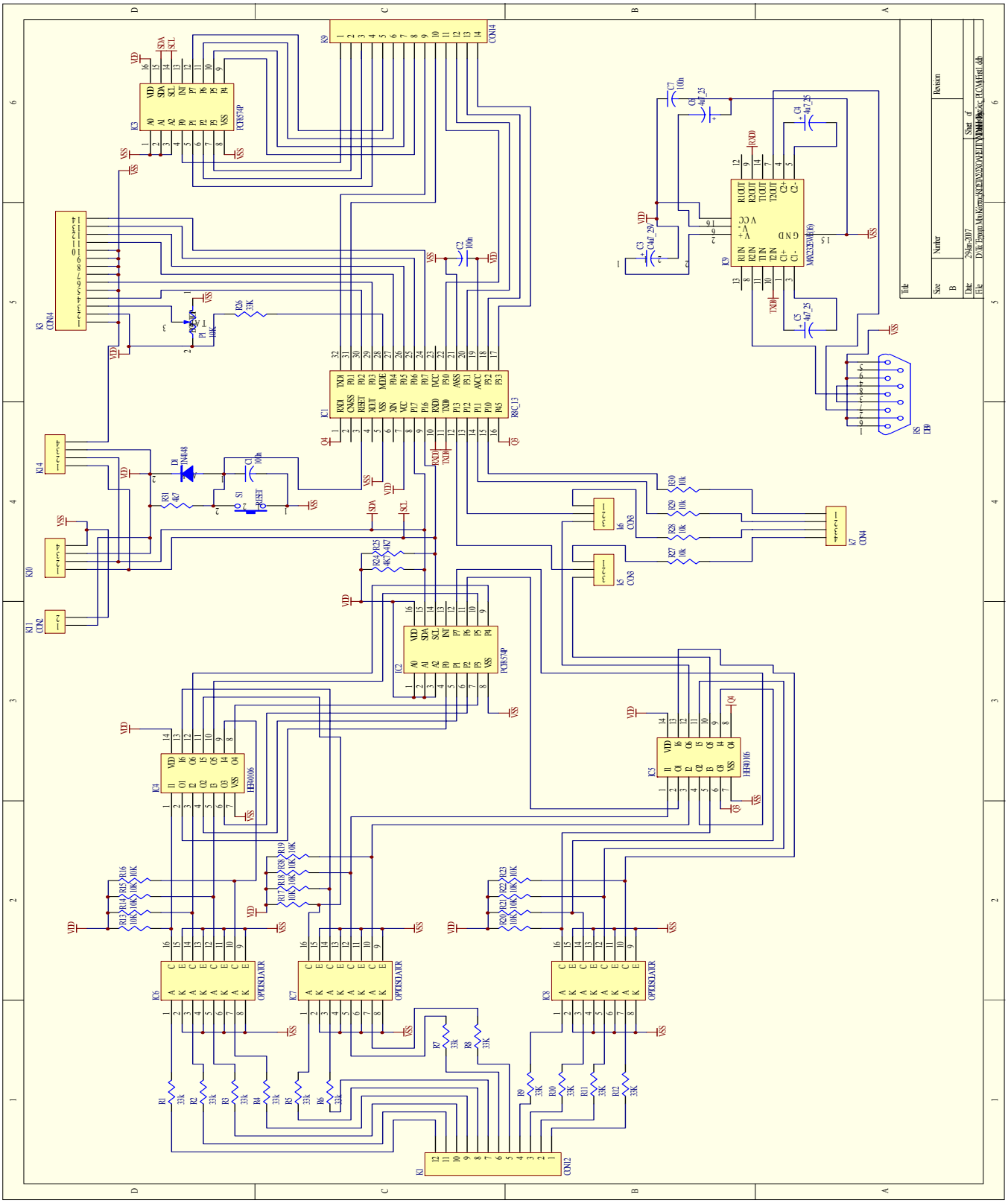


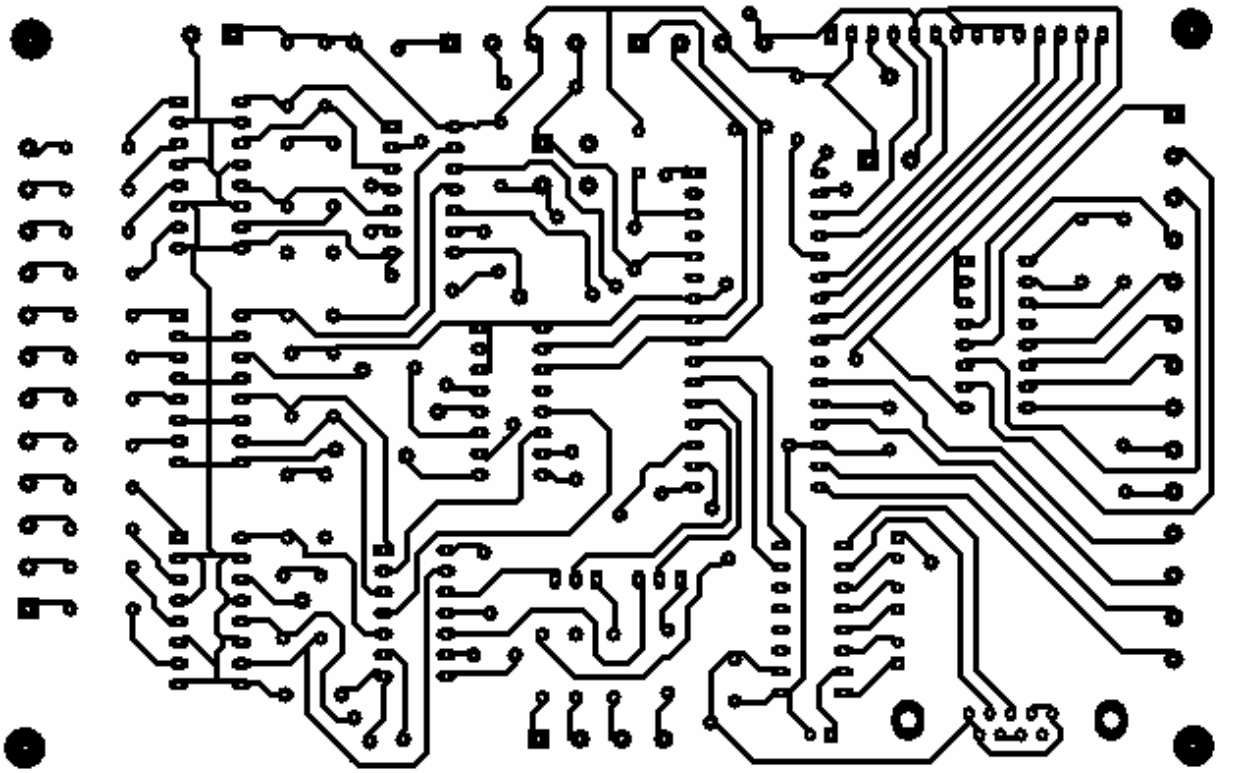


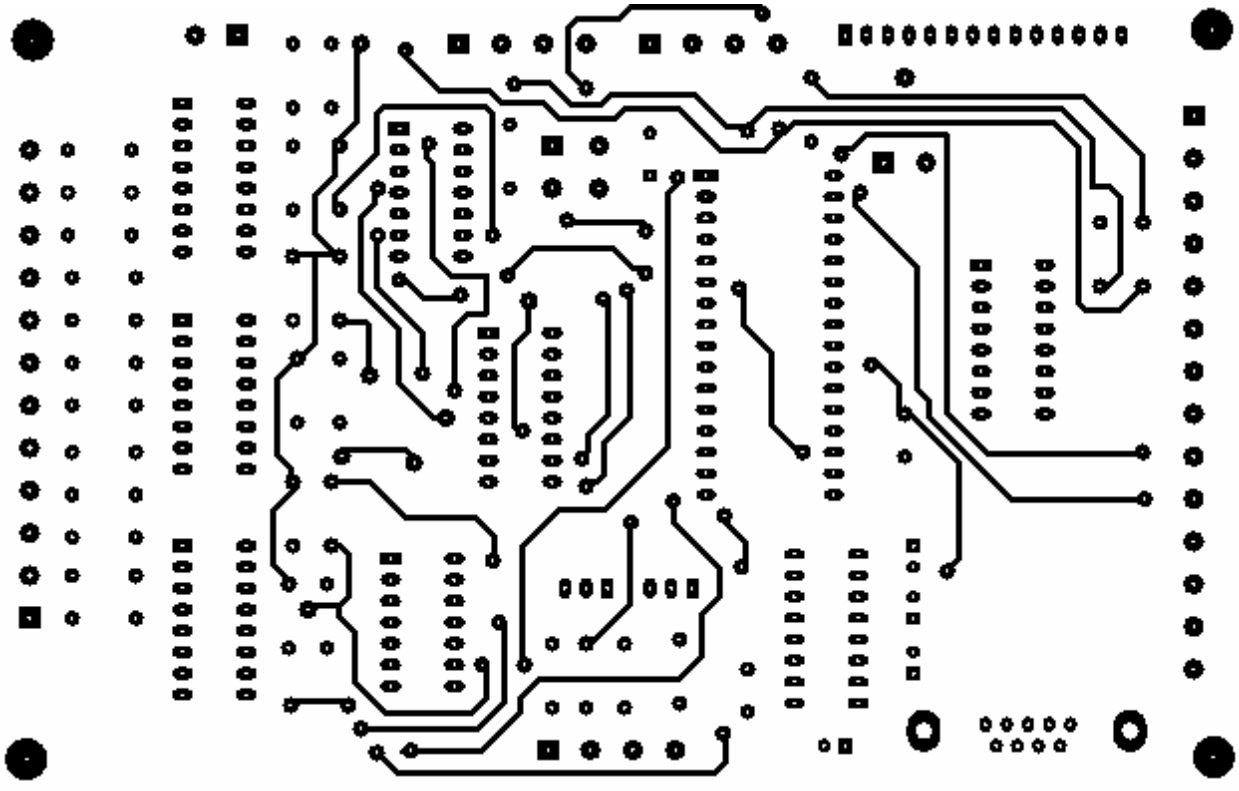


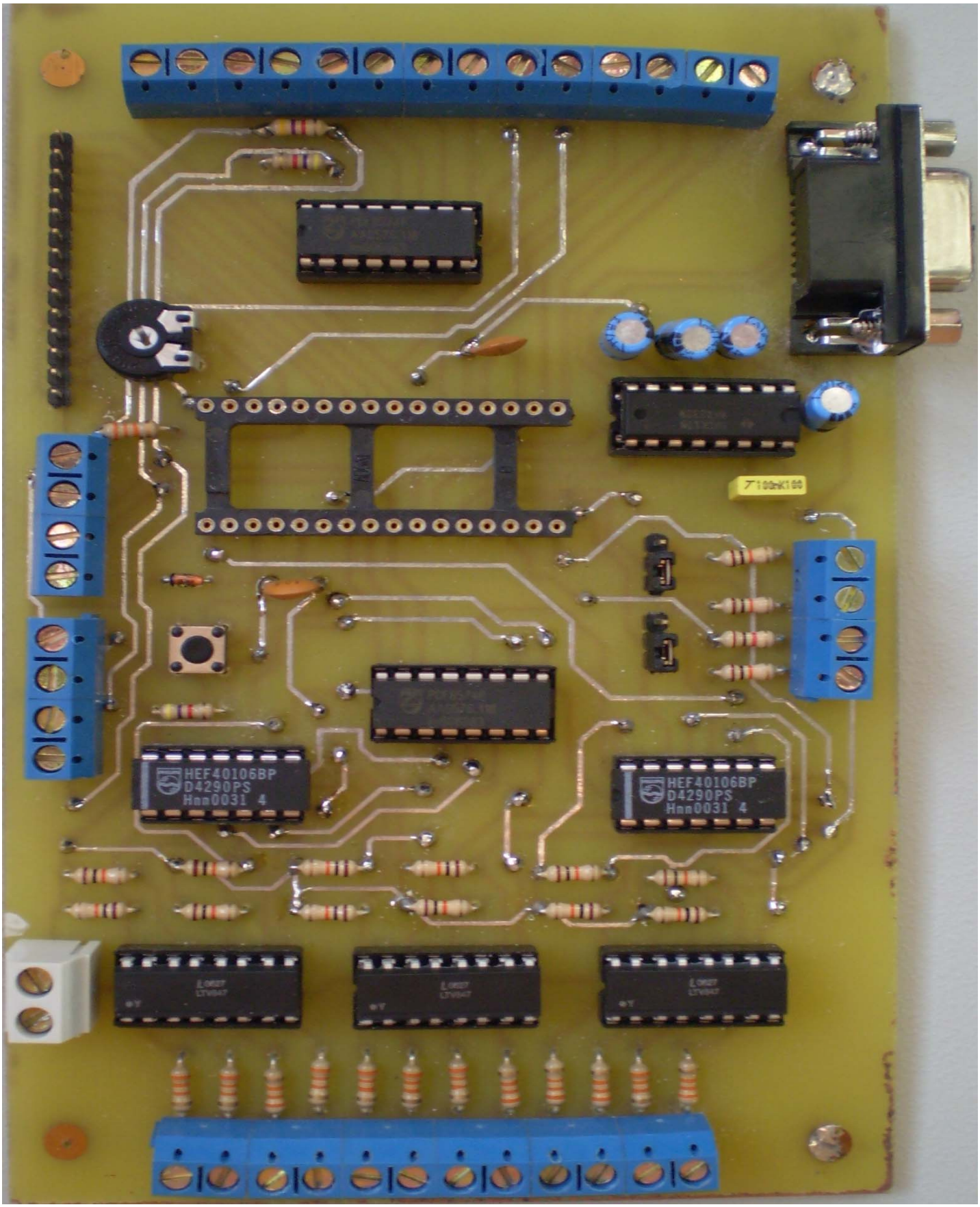

HEF40106BP
D4390PS
Hnn0031

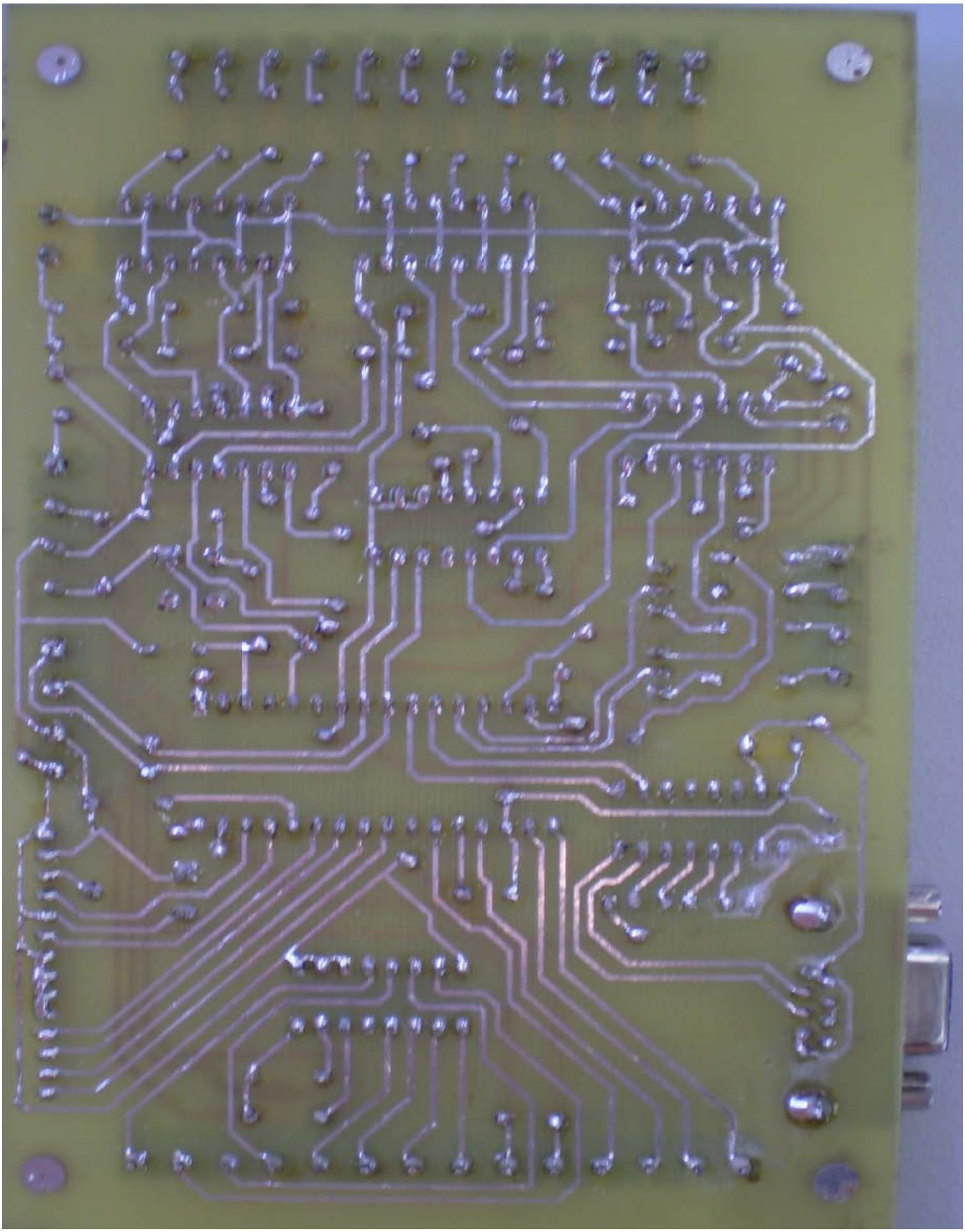
Παρακάτω απεικονίζεται η ***πλακέτα εισόδων*** (κύρια πλακέτα μικροεπεξεργαστή)



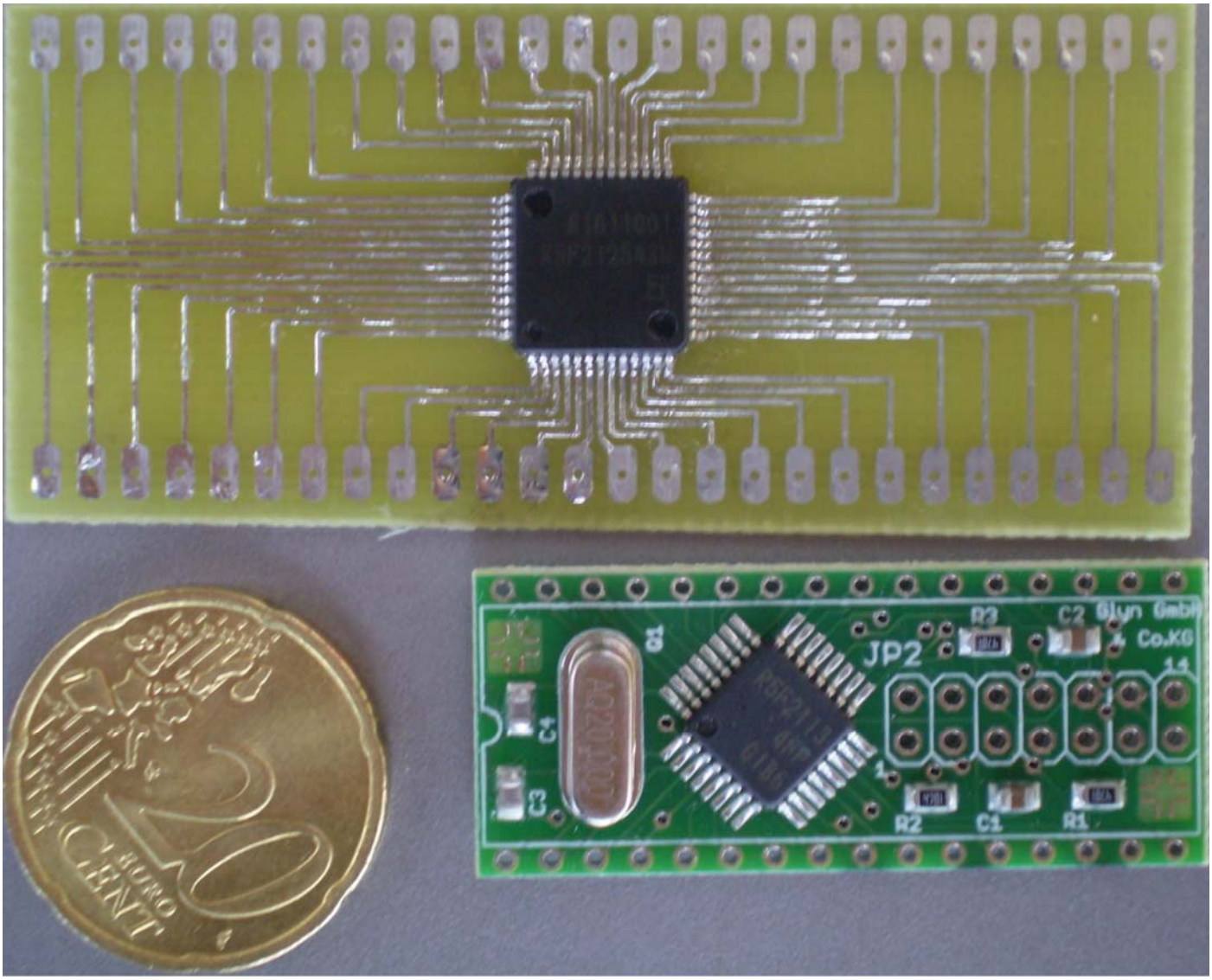





HEF40106BP
D4290PS
Hnn0031 4
HEF40106BP
D4290PS
Hnn0031 4

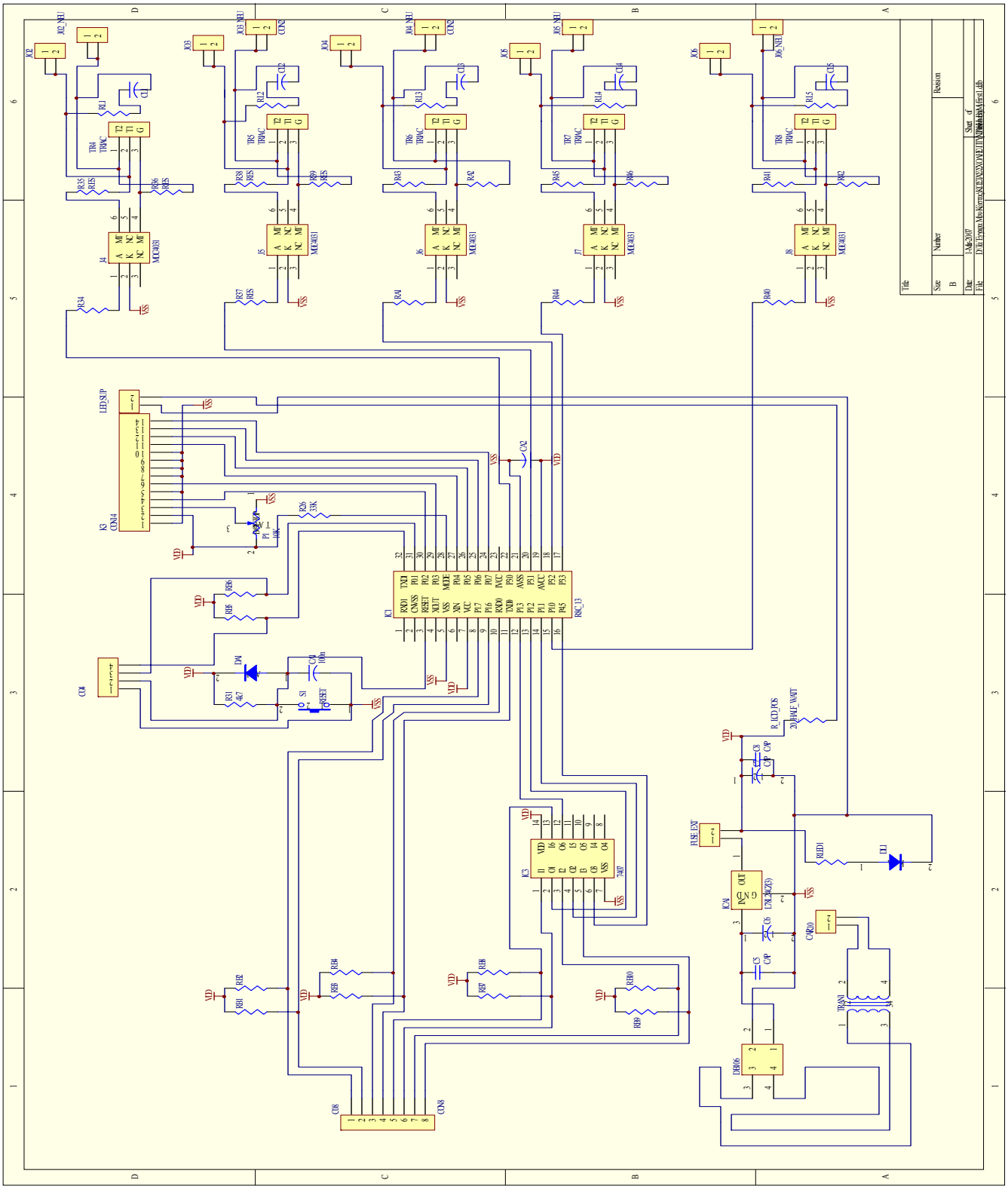


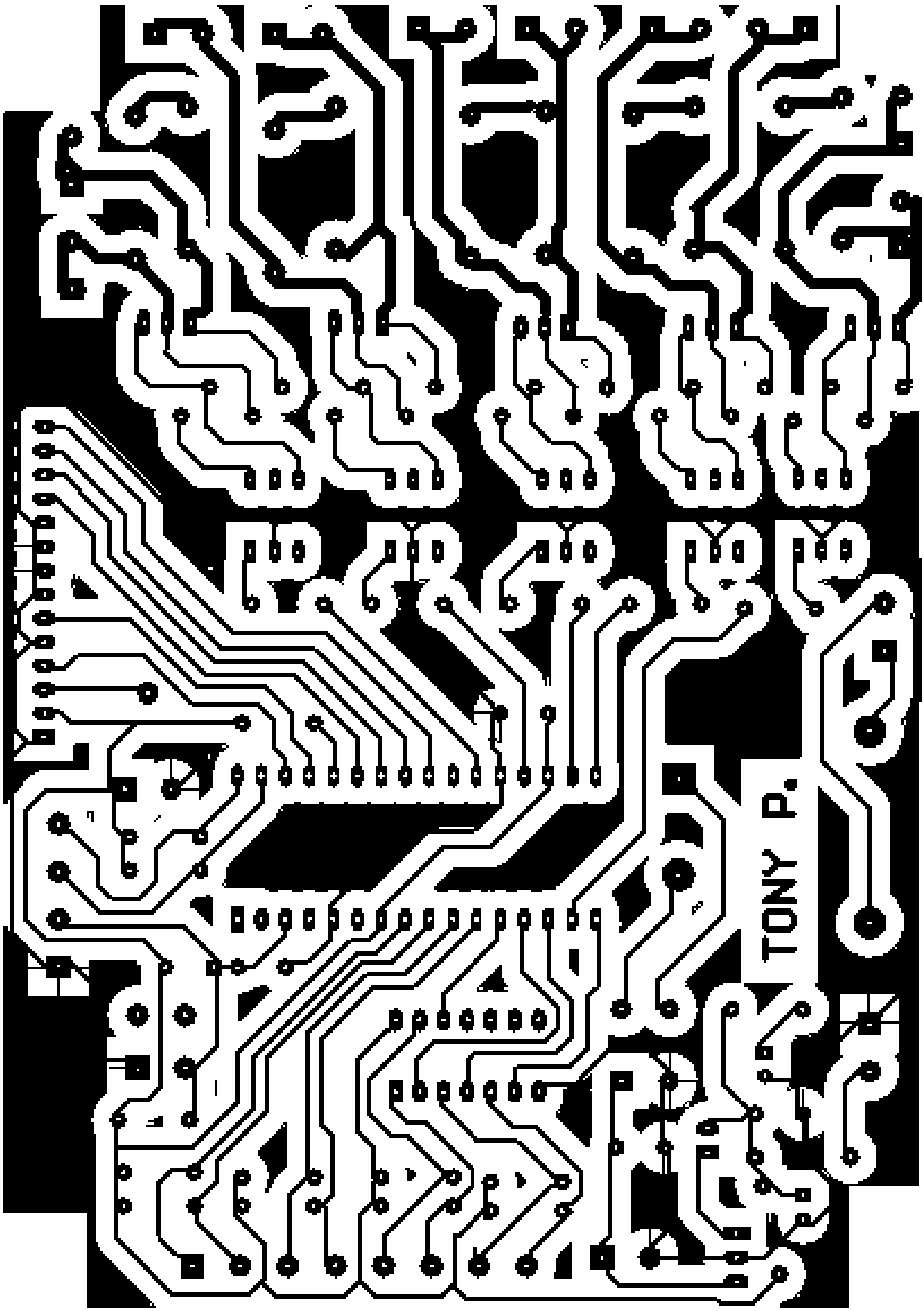
Παρακάτω απεικονίζεται η πλακέτα μικροελεγκτή ***R8C/25 και R8C/13***.



Η μικρή πλακέτα κατασκευάζεται από την εταιρεία GLYN και πωλείται από τους τοπικούς αντιπροσώπους της. Περιλαμβάνει την θύρα προγραμματισμού E8. καθώς και τον εξωτερικό κρύσταλλο 20MHz. Η αντίστοιχη πάνω πλακέτα με τον επεξεργαστή R8C/25 κατασκευάστηκε από εμάς.

Παρακάτω απεικονίζεται το σχηματικό, το pcb, καθώς και η όλη κατασκευή όπου θα χρησιμοποιηθεί σε ***διάταξη αυτόματου ελέγχου καμπανών***. Η κατασκευή σχεδιάστηκε για λογαριασμό των χυτηρίων Χανίων Παπαδάκης – Παυλουδάκης "Εφαρμογές Αυτοματισμού Καμπανών Εκκλησιών".




TONY P.

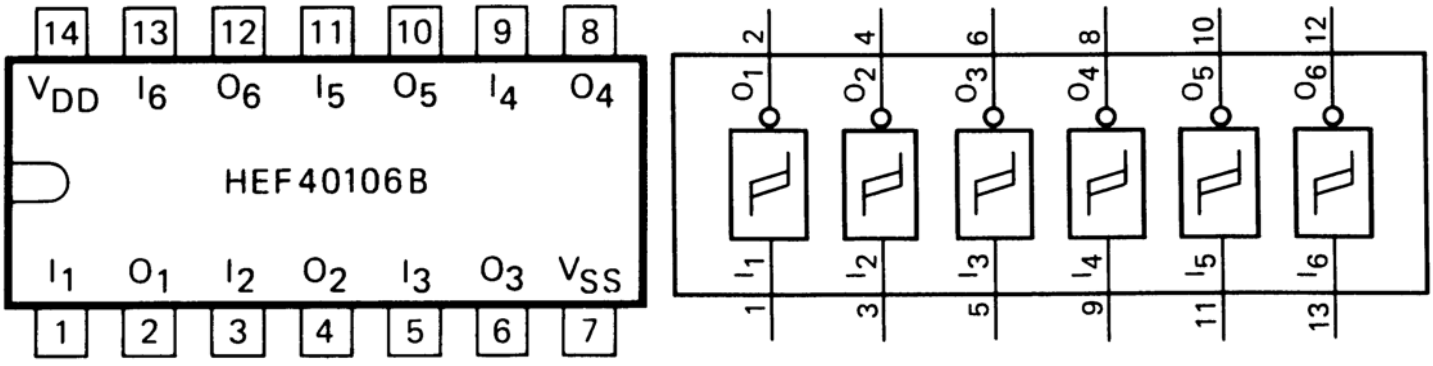

**MANUAL OPERATION*
BELL1 BELL2 BELL3
BELL4 BELL5
S1 S2 S3 S4 S5

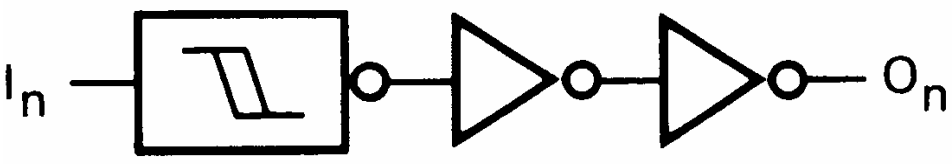
Κεφάλαιο 5

Σε αυτό το κεφάλαιο παρουσιάζονται τα κύρια χαρακτηριστικά των ολοκληρωμένων εξαρτημάτων (I.C.) που χρησιμοποιούνται σε όλες μας τις πλακέτες.

5.1 Περιγραφή του HEX40106B

Παρακάτω απεικονίζεται το ολοκληρωμένο HEX40106B σε εσωτερικό block διάγραμμα, το λογικό block διάγραμμα για καθέναν inverter, καθώς και σε διάγραμμα αρίθμησης των pin.



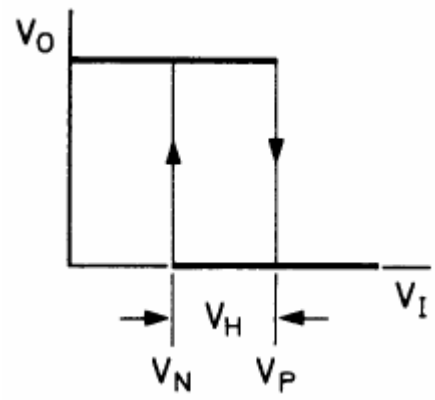


Κάθε κύκλωμα του HEF40106B λειτουργεί ως αναστροφέας δρώντας ως Schmitt–trigger. Οι διακόπτες Schmitt–trigger διαθέτουν διαφορετικά σημεία λήψης για τα θετικά και τα αρνητικά σήματα εισόδου αντίστοιχα. Η διαφορά μεταξύ της θετικής τάσης εισαγωγής (V_P) και της αρνητικής τάσης εισαγωγής (V_N) ορίζεται ως τάση υστέρησης (V_H).

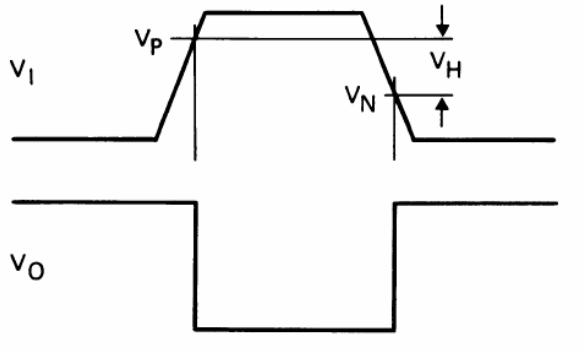
Αυτή η συσκευή μπορεί να χρησιμοποιηθεί για τα ενισχυμένα σήματα θορύβου ή στα "τετραγωνικά " αργά μεταβαλλόμενα κυματοειδή σήματα.

Στον παρακάτω πίνακα φαίνονται τα κύρια χαρακτηριστικά του για V_{SS}=0V, και θερμοκρασία περιβάλλοντος ίση με 25 ^{o}C:

	V_{DD} (V)	SYMBOL	MIN (V)	TYPICAL (V)	MAX (V)
Τάση Υστέρησης	5	V_H	0.5	0.8	
Επίπεδο μεταβολής για θετική τάση εισόδου	5	V_P	2	3	3.5
Επίπεδο μεταβολής για αρνητική τάση εισόδου	5	V_N	1.5	2.2	3



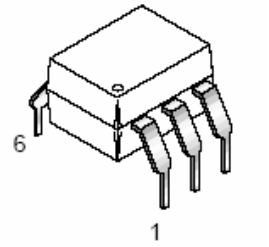
Χαρακτηριστική μεταφοράς

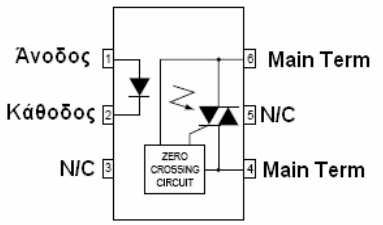


Καθορισμός V_P, V_N και V_H, όπου το V_N και V_P είναι από 30% έως 70% της κυματομορφής

5.2 Περιγραφή του MOC3041M

Παρακάτω απεικονίζεται το ολοκληρωμένο MOC3041M σε εσωτερικό block διάγραμμα, καθώς και σε διάγραμμα αρίθμησης των pin.

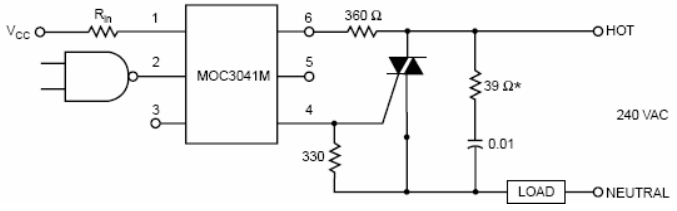




Το MOC3041M αποτελείται από μια δίοδο υπέρυθρης εκπομπής η οποία συνδέεται οπτικά με ένα μονολιθικό πυρίτιο ανιχνευτή ο οποίος εκτελεί τη λειτουργία ενός μηδενικού περάσματος τάσης (Zero Voltage Crossing). Είναι σχεδιασμένο για τη χρήση με triac σε λογικά συστήματα τα οποία τροφοδοτούνται από γραμμές με 115VAC, όπως από τηλέγραφους, τηλεοράσεις, ρελέ στερεάς κατάστασης, σε βιομηχανικούς ελέγχους, εκτυπωτές, μηχανές, σωληνοειδής και καταναλωτικές συσκευές, κ.λ.π.. Εφαρμογές που βρίσκει τόπο είναι σε ελέγχους σωληνοειδών βαλβίδων, ελέγχους φωτισμού, στατικούς διακόπτες δύναμης, κινήσεις μηχανών εναλλασσόμενου ρεύματος, ελέγχους θερμοκρασίας, εκκινητές μηχανών εναλλασσόμενου ρεύματος, ρελέ στερεάς κατάστασης. Διατρέχεται από τα παρακάτω χαρακτηριστικά γνωρίσματα:

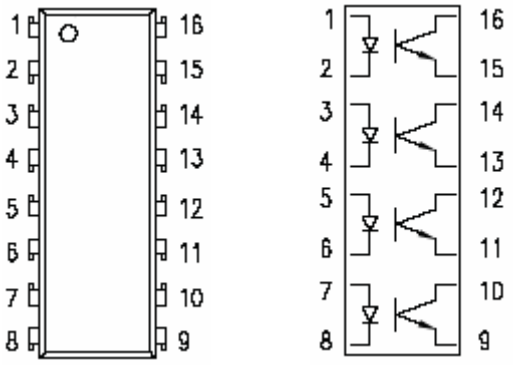
- Απλοποιεί τον λογικό έλεγχο σε ισχύς 115VAC,
- Μηδενικό πέρασμα τάσης (Zero Voltage Crossing),
- dV/dT ισούται με 2000V/μS,
- V_{DE} αναγνωρίζεται κατόπιν επιλογής.

Το παρακάτω κύκλωμα είναι χαρακτηριστικό για την χρήση μετατροπής hot γραμμών. Το φορτίο μπορεί να συνδεθεί είτε στην ουδέτερη, είτε στην hot γραμμή. Η αντίσταση 39Ω και ο πυκνωτής 0.01μF είναι για την επίπληξη του triac, και μπορεί να μην είναι απαραίτητα ανάλογα με το φορτίο που χρησιμοποιούμε και ιδιαίτερα με το triac.



5.3 Περιγραφή του OCP–PCT4116/X

Παρακάτω απεικονίζεται το ολοκληρωμένο OCP–PCT4116/X σε εσωτερικό block διάγραμμα, καθώς και σε διάγραμμα αρίθμησης των pin.



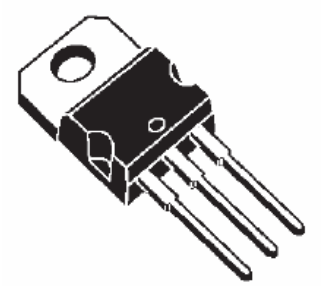
1,3,5,7 → Άνοδος
2,4,6,8 → Κάθοδος
9,11,13,15 → Εκπομπός
10,12,14,16 → Συλλέκτης

Στον παρακάτω πίνακα φαίνονται τα κύρια χαρακτηριστικά του για V_{SS}=0V, και θερμοκρασία περιβάλλοντος ίση με 25 ^{o}C:

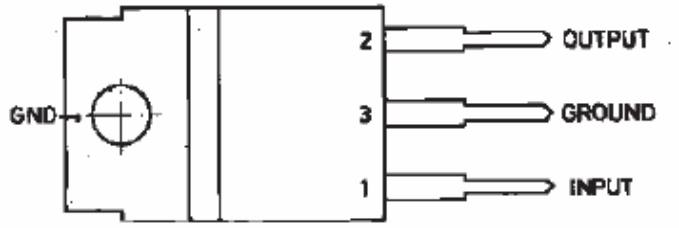
	Σύμβολο	**Όρος**	**MIN**	**TYPICAL**	**MAX**
Τάση εισόδου	V_F	I_f=20mA	---	1.2V	1.4V
Κορφή τάσης εισόδου	V_{FW}	I_{FW}=0.5A	---	---	3.5V
Ρεύμα αντιστροφής	I_R	V_R=4V	---	---	10μA
Συχνότητα αποκοπής	f_C	V_{CE}=5V, I_C=2mA, R_L=100Ω	---	80KHz	---
Χρόνος απόκρισης (Άνοδος)	t_r	V_{CE}=5V, I_C=2mA, R_L=100Ω	---	5μsec	20μsec
Χρόνος απόκρισης (Κάθοδος)	t_f	V_{CE}=5V, I_C=2mA, R_L=100Ω	---	4μsec	20μsec

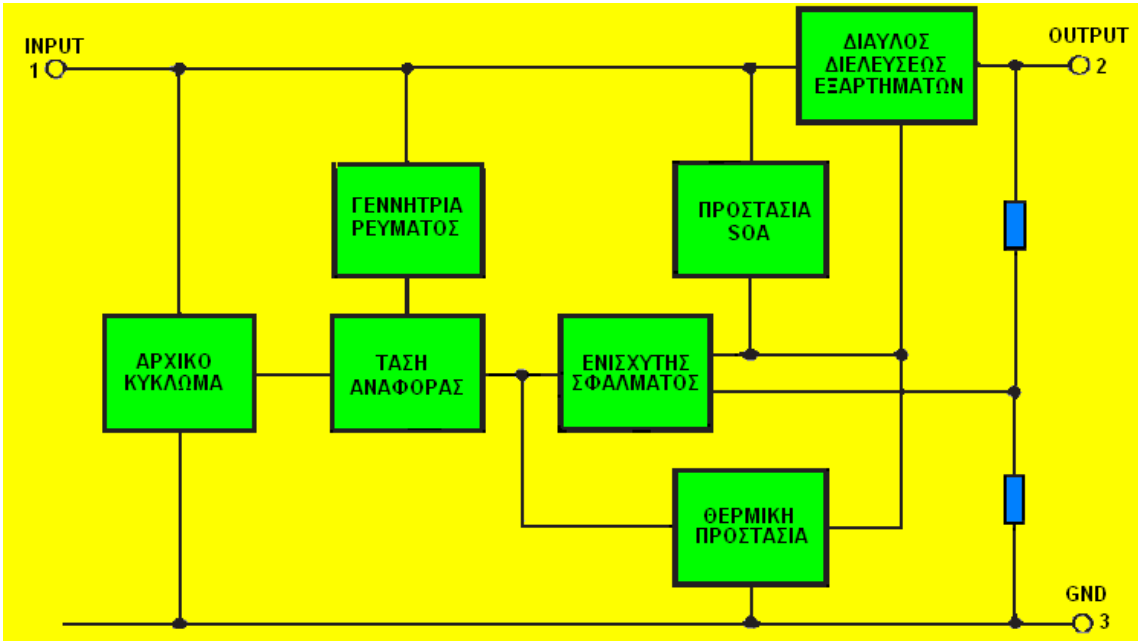
5.4 Περιγραφή του L7805 και του L7824

Παρακάτω απεικονίζεται τα ολοκληρωμένα L7805, L7824 σε εσωτερικό block διάγραμμα, καθώς και σε διάγραμμα αρίθμησης των pin.



TO-220

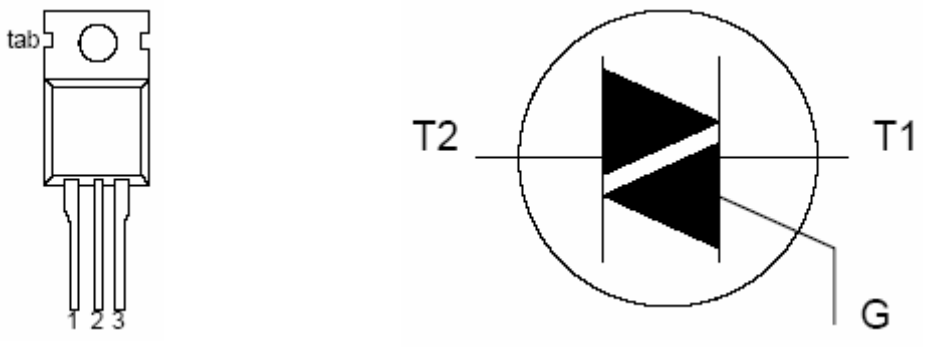




ΤΕΧΝΙΚΑ ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ		
Ρεύμα παραγωγής	:	Έως 1.5A
Τάσεις παραγωγής	:	5, 5.2, 6, 8, 8.5, 9, 12, 15, 18, 24 V
Θερμική προστασία υπερφόρτωσης		
Προστασία μετάβασης παραγωγής SOA		

5.5 Περιγραφή του BT136

Παρακάτω απεικονίζεται το ολοκληρωμένο BT136, το οποίο είναι ένα triac, σε εσωτερικό block διάγραμμα, καθώς και σε διάγραμμα αρίθμησης των pin.



PIN	DESCRIPTION
1	main terminal 1
2	main terminal 2
3	gate
tab	main terminal 2

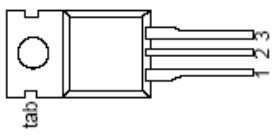
Το συγκεκριμένο triac έχει ρεύμα λειτουργίας στα 4A, με στιγμιαίο ρεύμα (peak) 4A. Το triac είναι ένας ημιαγωγός πέντε στρωμάτων (N,P,N,P,N), τριών ακροδεκτών, οποίος μπορεί να μεταβεί από την κατάσταση 0FF στην κατάσταση ON και με τις δύο πολικότητες μιας εναλλασσόμενης τάσης τροφοδότησης. Ισοδυναμεί με δυο thyristors συνδεσμολογημένα αντιπαράλληλα με τρεις ακροδέκτες. Οι δυο ακροδέκτες (T1,T2) διαρρέονται από το ελεγχόμενο ρεύμα ,και ο τρίτος (πύλη) χρησιμοποιείται για το "σκανδαλισμό" του, δηλαδή τον έλεγχο της αγωγιμότητας. Η δομή του, η ισοδυναμία του με 2 thyristors. Το triac προορίζεται για χρήση σε εφαρμογές οι οποίες απαιτούν υψηλό αμφίδρομο έλεγχο, και εμποδίζουν την υψηλή θερμοκρασία στο κύκλωμα.

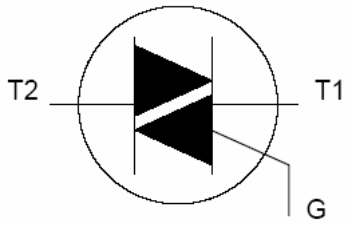
Οι χαρακτηριστικές εφαρμογές που το διαπρέπουν περιλαμβάνουν τον έλεγχο μηχανών, το βιομηχανικό και εσωτερικό φωτισμό, τη θέρμανση, καθώς και την στατική μετατροπή.

	Τα κυριότερα χαρακτηριστικά λειτουργίας ενός triac είναι τα εξής
1.	Αμφίπλευρη αγωγιμότητα.
2.	Σκανδαλισμός με παλμούς πύλης θετικής ή αρνητικής πολικότητας ανεξάρτητα από την πολικότητα της τάσης που εφαρμόζεται μεταξύ T1,T2.
3.	Σε κατάσταση αγωγής του triac η πύλη του δεν ασκεί έλεγχο.
4.	Η αγωγιμότητα σταματά όταν το ρεύμα που διαρρέει τους T1,T2 γίνει μικρότερο από το ρεύμα συγκράτησης (holding current).
5.	Χρόνος αποκοπής της τάξης των μs.

5.6 Περιγραφή του BT138

Παρακάτω απεικονίζεται το ολοκληρωμένο BT138, το οποίο είναι ένα triac, σε εσωτερικό block διάγραμμα, καθώς και σε διάγραμμα αρίθμησης των pin.





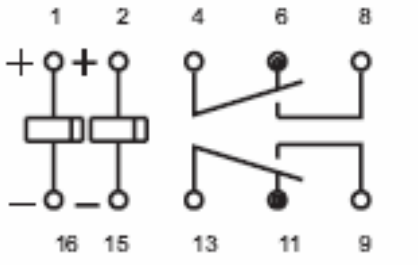
PIN	DESCRIPTION
1	main terminal 1
2	main terminal 2
3	gate
tab	main terminal 2

Το συγκεκριμένο triac έχει ρεύμα λειτουργίας στα 12A, με στιγμιαίο ρεύμα (peak) 95A. Τα υπόλοιπα χαρακτηριστικά του είναι ακριβώς τα ίδια με το BT136.

5.7 Περιγραφή του HFD2

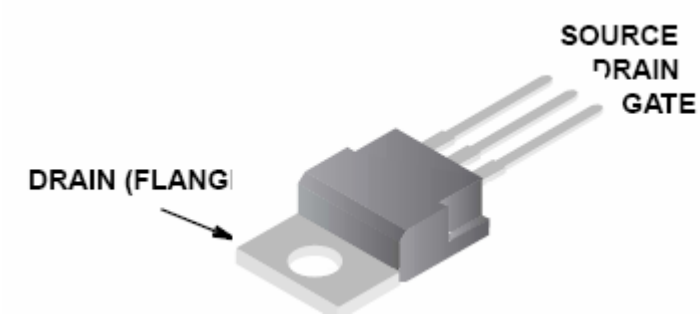
Παρακάτω απεικονίζεται το ολοκληρωμένο HFD2, το οποίο είναι ένα ρελέ, σε εσωτερικό block διάγραμμα, καθώς και σε φωτογραφία του.

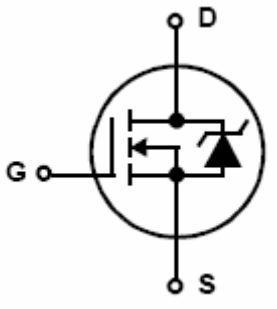
Στο διπλανό σχήμα απεικονίζεται το ρελέ το οποίο λειτουργεί στα 2A, 30V DC, με μέγιστη κατανάλωση ισχύος 60W, έχει υψηλή ευαισθησία της τάξεως των 150mW, κυκλοφορεί σε συσκευασία φιλική προς το περιβάλλον (RoHS υποχωρητικό), οι διαστάσεις του είναι 20.2 x 10.0 x 10.6 , έχει χρόνο μανδάλωσης 3msec, καθώς και χρόνο επανεκκίνησης. Στη διπλανή φωτογραφία απεικονίζεται το εσωτερικό block διάγραμμα του ρελέ, καθώς και η αρίθμηση των pin του.



5.8 Περιγραφή του IRF540

Παρακάτω απεικονίζεται το ολοκληρωμένο IRF540, το οποίο είναι ένα mosfet, σε εσωτερικό block διάγραμμα, αρίθμησης των pin του, καθώς και σε φωτογραφία του.





Το συγκεκριμένο mosfet έχει τάση λειτουργίας στα 100V με 28A, έχει αντίσταση R_{DS}=0.077Ω, ταχύτητα switching κάποια νανοδευτερόλεπτα, έχει γραμμική χαρακτηριστική μεταφοράς.

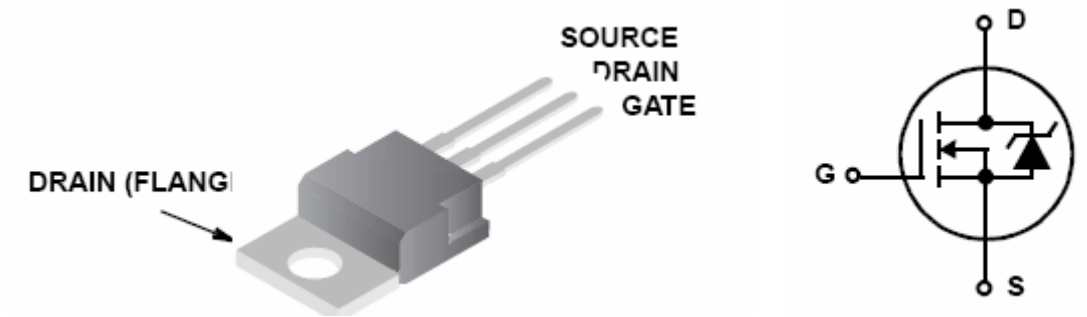
Οι αυξανόμενες απαιτήσεις από τις ηλεκτρονικές συσκευές ισχύος για υψηλότερη συχνότητα λειτουργίας σε εφαρμογές όπως η αλλαγή κατάστασης σε ισχύς τροφοδότησης, έχει οδηγήσει στην ανάπτυξη των MOSFET ισχύος. Τα MOSFET ισχύος λειτουργούν με μικρές απώλειες και απαιτούν πολύ μικρότερες τιμές ρεύματος πύλης από ότι απαιτεί η βάση ενός ισοδύναμου transistor για να διατηρηθεί στην κατάσταση αγωγής.

Σε χαμηλότερες τάσεις η αντίσταση στην κατάσταση αγωγής (on state) ενός MOSFET ισχύος είναι μικρότερη από αυτή ενός ισοδύναμου transistor κυμαινόμενη από 0,05 έως 0,25Ω για μια συσκευή 100V και 2-8 Ω για μια συσκευή 1000V.

Οι εφαρμογές των MOSFET ισχύος δεν είναι πολύ εξαπλωμένη από το γεγονός ότι η τεχνολογία που χρησιμοποιείται για την κατασκευή τους είναι σήμερα περισσότερο ακριβή απ' ότι αυτή των άλλων συσκευών ηλεκτρονικών ισχύος. Χρησιμοποιούνται όμως ιδιαίτερα στην κατασκευή inverters choppers και παλμοτροφοδοτικών.

5.9 Περιγραφή του IRF610

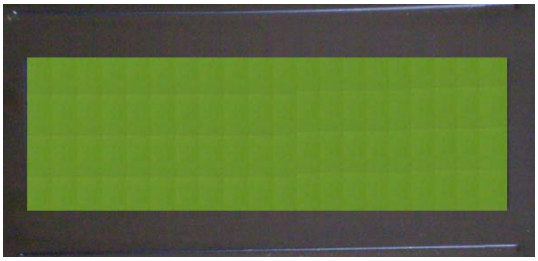
Παρακάτω απεικονίζεται το ολοκληρωμένο IRF610, το οποίο είναι ένα mosfet, σε εσωτερικό block διάγραμμα, αρίθμησης των pin του, καθώς και σε φωτογραφία του.

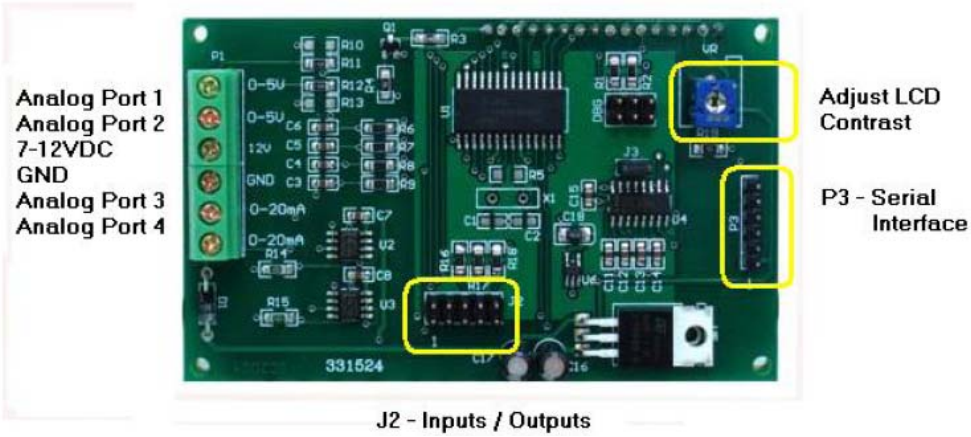


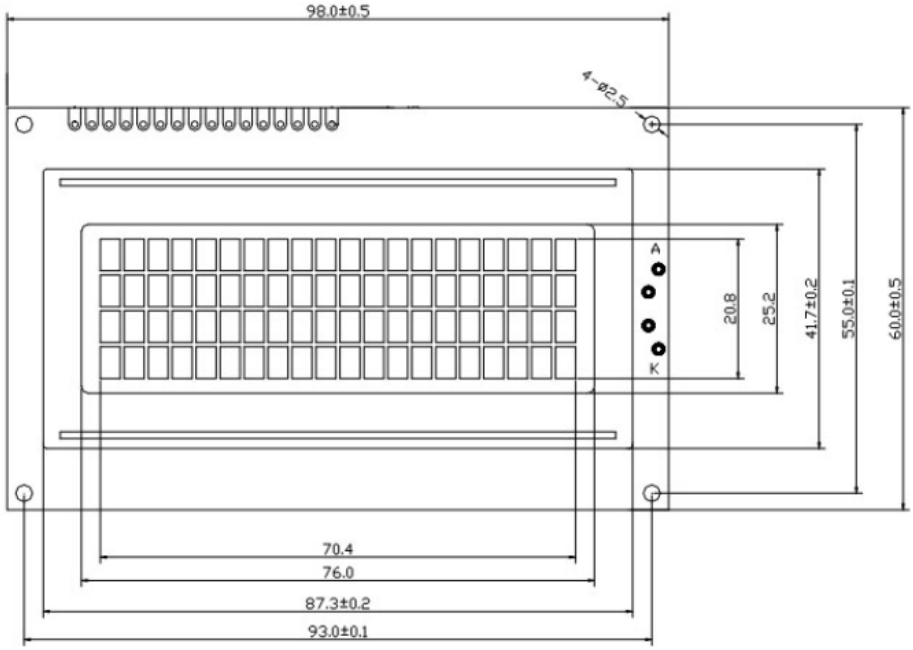
Το συγκεκριμένο mosfet έχει τάση λειτουργίας στα 200V με 3.3A, έχει αντίσταση R_{DS}=1.5Ω, ταχύτητα switching κάποια νανοδευτερόλεπτα, έχει γραμμική χαρακτηριστική μεταφοράς.

5.10 Περιγραφή του LCD 4x20

Παρακάτω απεικονίζεται το LCD 4x20, σε εσωτερικό block διάγραμμα, καθώς και σε φωτογραφία του.





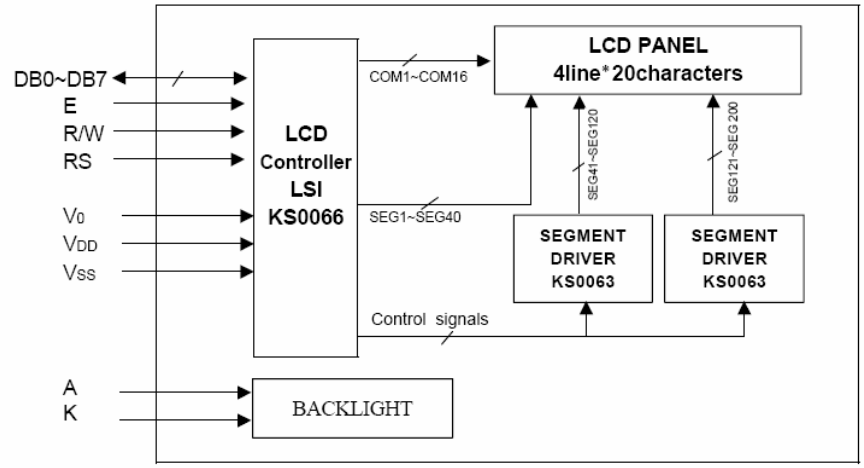


DISPLAY POSITION

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
FIRST LINE	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13
SECOND LINE	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
THIRD LINE	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23	24	25	26	27
FOURTH LINE	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67

DD RAM ADDRESS

Παρακάτω απεικονίζεται το εσωτερικό μπλοκ διάγραμμα του LCD 4x20.

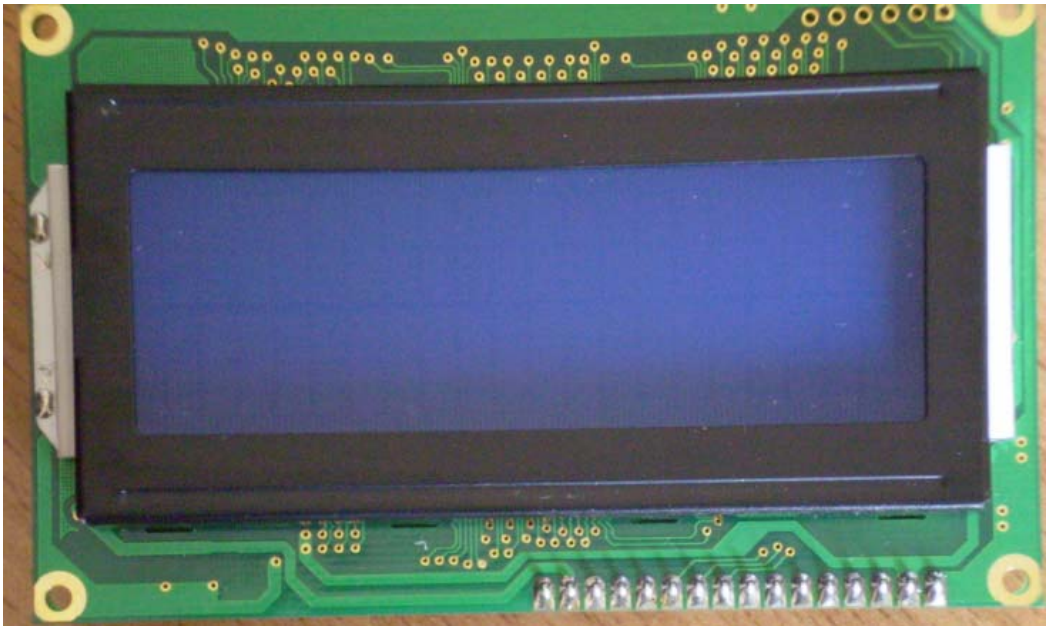


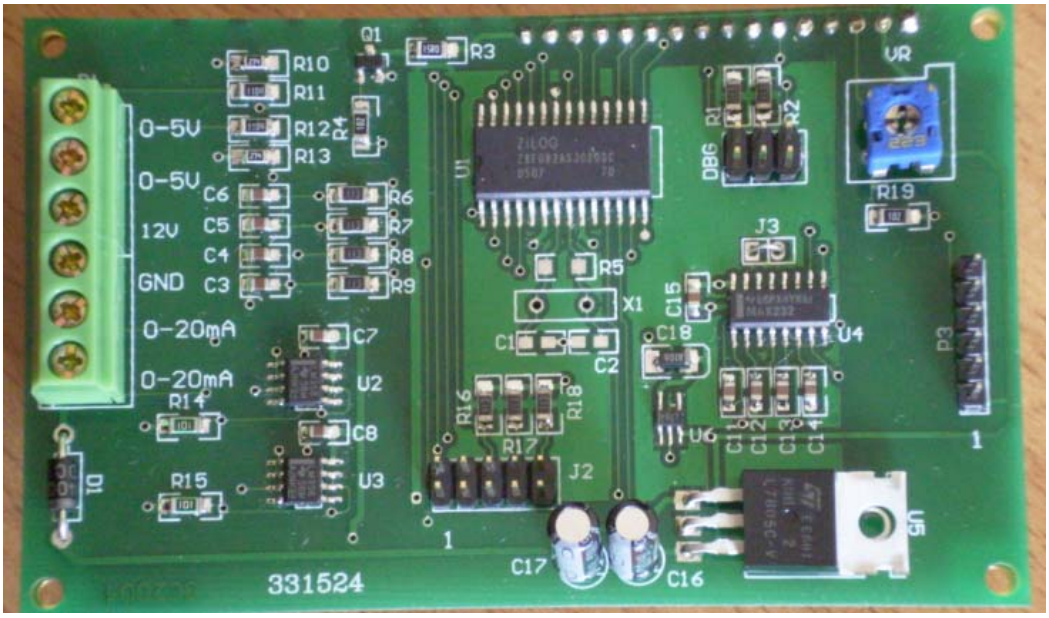
Παρακάτω απεικονίζονται τα pin του LCD σε τι αντιστοιχίζουν.

Pin No.	Symbol	Function
1	VSS	Ground terminal of module
2	VDD	Supply terminal of module +5 V
3	V0	Power Supply for Liquid crystal Drive
4	RS	Register Select RS = 0… Instruction Register RS = 1… Data Register
5	R/W	Read / Write R/W = 1 (Read) R/W = 0 (Write)
6	E	Enable
7	DB0	Bi-directional Data Bus, Data Transfer is performed once , thru DB0~DB7 , in the case of interface data . Length is 8-bits; and twice , thru DB4~DB7 in the case of interface data length is 4-bits . Upper four bits first then lower four bits .
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED - (K)	Please also refer to 6.1 PCB drawing and description .
16	LED + (A)	Please also refer to 6.1 PCB drawing and description .

5.11 Περιγραφή του LCD 4x20 (σειριακής επικοινωνίας)

Παρακάτω απεικονίζεται το LCD 4x20 (σειριακής επικοινωνίας), σε φωτογραφία του. Το συγκεκριμένο LCD το δοκιμάσαμε αλλά τελευταία στιγμή αλλάξαμε γνώμη και τοποθετήσαμε στην κατασκευή μας το LCD όπου αναφέρεται στην παράγραφο 5.10.

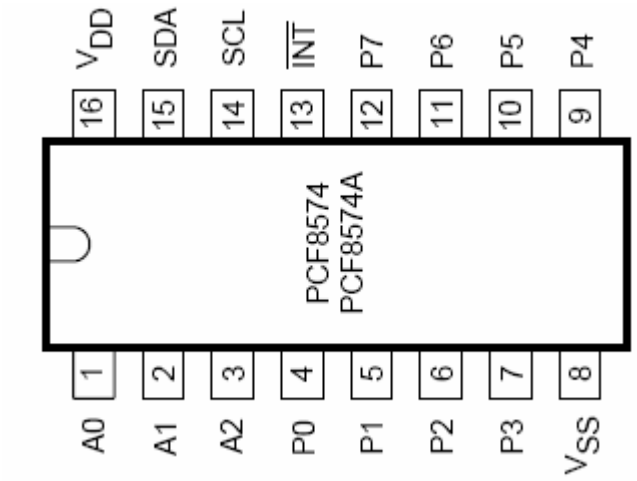




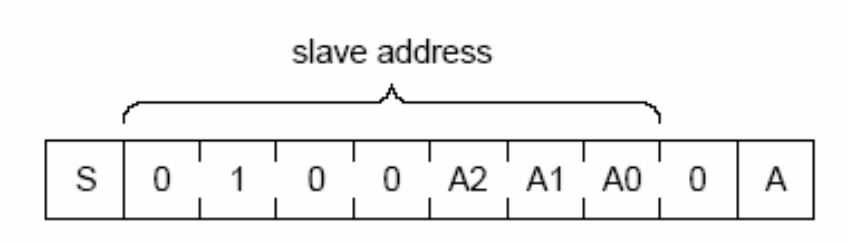
Σε αυτό το σημείο παραθέτουμε ένα απόσπασμα από τον κώδικα του συγκεκριμένου LCD:

```
send_data(254);         /* turn on underline cursor */
      send_data(1);

      send_data('L');   /* Display LCD ON on LCD */
      send_data('C');
      send_data('D');
      send_data(' ');
      send_data('O');
      send_data('N');
```

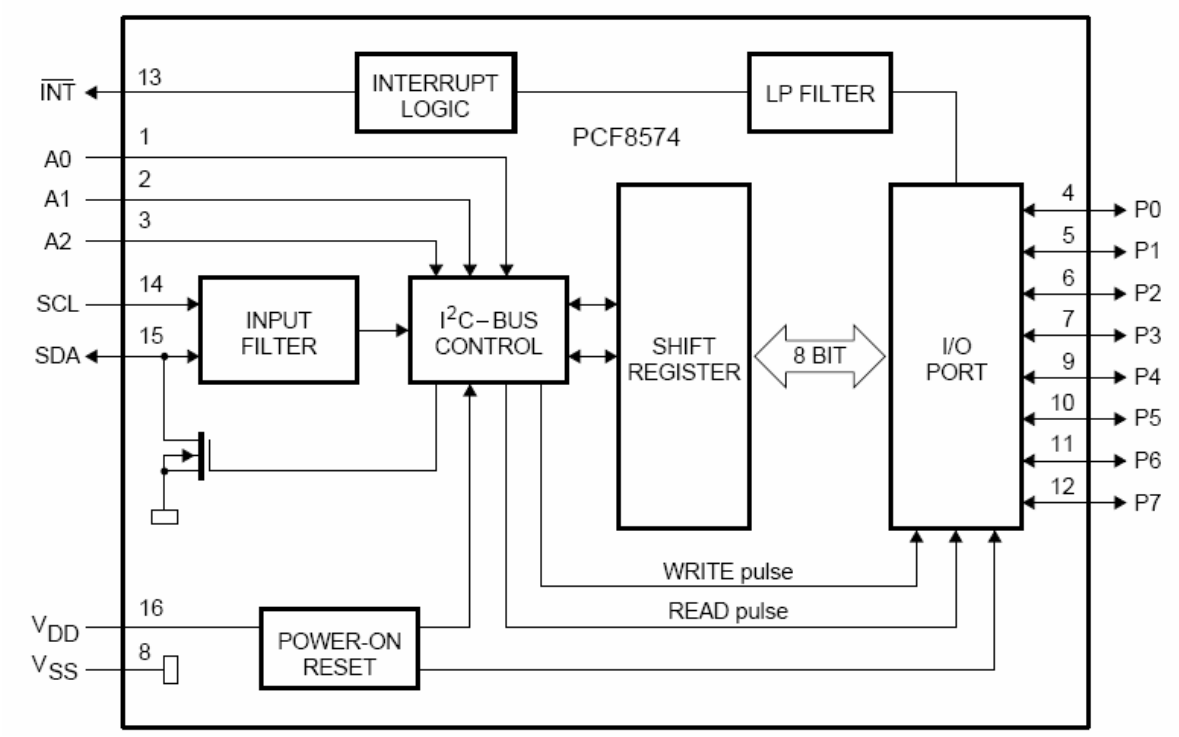
5.12 Περιγραφή του PCF8574



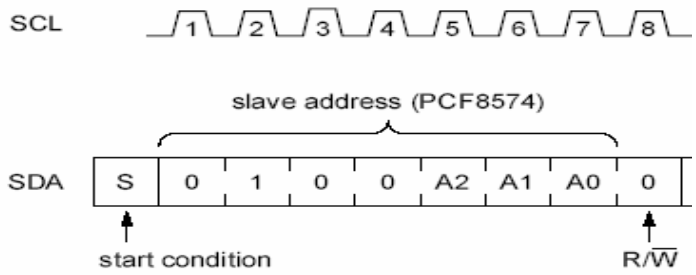
Στην διπλανή φωτογραφία απεικονίζεται το ολοκληρωμένο PCF8574, το οποίο είναι ένας 8bit αποσυμπιεστής με δίαυλο επικοινωνίας I^2C. Παρακάτω απεικονίζεται μια φωτογραφία της flag του ολοκληρωμένου λειτουργώντας ως slave.



Παρακάτω απεικονίζεται το εσωτερικό μπλοκ διάγραμμά του. Βλέποντας το παρατηρούμε ότι οι πόρτες P0 ~ P7 μπορούν να λειτουργήσουν είτε ως είσοδοι, είτε ως έξοδοι. Ανάλογα και την τιμή όπου έχουν οι Accumulator κάνει και την αντίστοιχη λειτουργία, επικοινωνώντας μέσω του δίαυλου I^2C.



Παρακάτω απεικονίζεται μια φωτογραφία με το πώς λειτουργούν τα SCL, SDAστο ολοκληρωμένο.



Κεφάλαιο 6

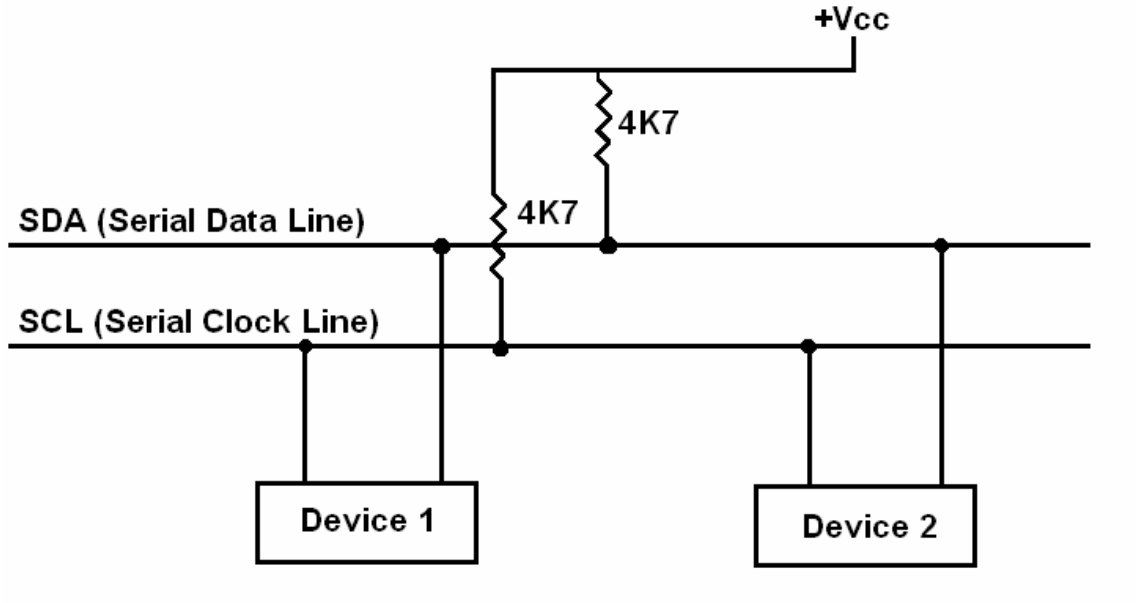
Σε αυτό το κεφάλαιο παρουσιάζονται τα πρωτόκολλα επικοινωνίας I^2C, καθώς και το RS–232.

6.1 Ο δίαυλος I^2C

Τα σημερινά ηλεκτρονικά συστήματα συνήθως περιλαμβάνουν τουλάχιστον έναν μικροελεγκτή και περιφερειακές συσκευές, όπως μνήμες και κυκλώματα I/O. Το ζητούμενο είναι ένας εύκολος και φθηνός τρόπος διασύνδεσης αυτών των κυκλωμάτων, σύμφωνα με ένα διαδεδομένο πρότυπο, που εξασφαλίζει τη μεταξύ τους επικοινωνία και την επεκτασιμότητα του συστήματος.

Με βάση τα παραπάνω έχει προταθεί και υλοποιηθεί από την εταιρεία Philips ένας πρότυπος σειριακός δίαυλος επικοινωνίας, που υποστηρίζεται πλέον από πολλά ολοκληρωμένα κυκλώματα, κάθε τεχνολογίας κατασκευής (CMOS, NMOS, bipolar). Ο δίαυλος αυτός ονομάζεται I^2C και στηρίζεται στην αρχιτεκτονική δύο συρμάτων, με τη βοήθεια των οποίων τα διάφορα ολοκληρωμένα κυκλώματα ανταλλάσσουν σειριακά δεδομένα και σήματα συγχρονισμού. Ταυτόχρονα, το πρότυπο αυτό δημιουργεί ένα πρωτόκολλο επικοινωνίας, για την αποφυγή συγκρούσεων κατά την ανταλλαγή δεδομένων.

Τα δύο σύρματα στα οποία στηρίζεται η αρχιτεκτονική του διαύλου I^2C ονομάζονται Serial Data (SDA) ή *γραμμή δεδομένων* και Serial Clock (SCL) ή *γραμμή χρονισμού*. Οι δύο αυτές γραμμές μεταφέρουν πληροφορίες ανάμεσα στις συσκευές που ενώνονται με τον δίαυλο, όπως αυτές απεικονίζονται στο παρακάτω σχήμα.



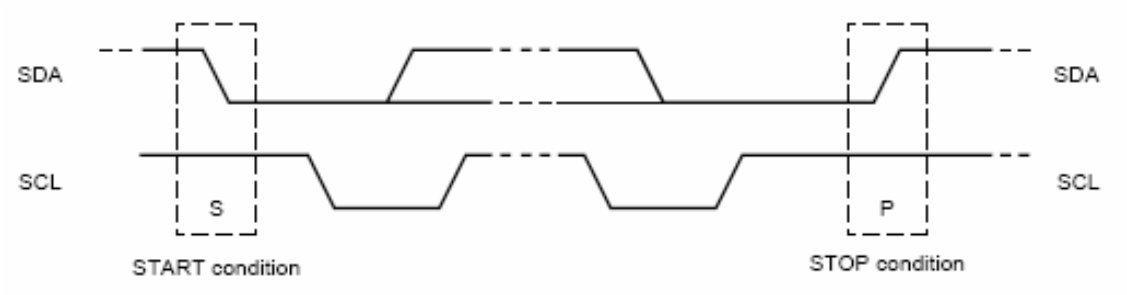
Κάθε συσκευή αναγνωρίζεται από μια μοναδική διεύθυνση και μπορεί να λειτουργήσει ως πομπός ή ως δέκτης δεδομένων. Για παράδειγμα, μια μνήμη μπορεί να λαμβάνει ή να στέλνει δεδομένα μέσω του διαύλου, το ίδιο κι ένας μικροελεγκτής, ενώ μια συσκευή οδήγησης LCD μπορεί μόνο να λαμβάνει.

Οι δύο γραμμές οδηγούνται στην τάση τροφοδοσίας μέσω αντιστάσεων pull–up (αντιστάσεων ανύψωσης σε τάση). Άρα, όταν οι γραμμές δεδομένων και χρονισμού δεν είναι ενεργές ευρίσκονται και οι δύο σε υψηλή στάθμη. Τότε λέμε ότι ο δίαυλος είναι ελεύθερος. Ας σημειωθεί ότι η λογική κατάσταση των γραμμών κρίνεται ως προς τη γείωση και συνεπώς η γραμμή που μεταφέρει το δυναμικό της γης μπορεί να θεωρηθεί ότι είναι η Τρίτη γραμμή του διαύλου.

Μια άλλη διάκριση ανάμεσα στις διατάξεις που ενώνονται με τον δίαυλο είναι σε Master (κύρια διάταξη) και σε Slave (εξαρτώμενη διάταξη). Χαρακτηρίζουμε ως Master κάθε ολοκληρωμένο κύκλωμα που έχει την πρωτοβουλία στη διακίνηση δεδομένων και αποστέλλει τους παλμούς συγχρονισμού προς τα υπόλοιπα κυκλώματα. Κάθε διάταξη που αποκρίνεται στην κύρια διάταξη και παίρνει μέρος στην επικοινωνία θεωρείται εξαρτώμενη.

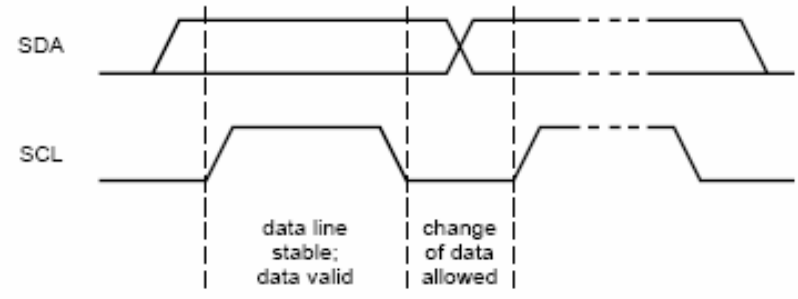
Η μεταφορά των δεδομένων ανάμεσα στις συσκευές του διαύλου γίνεται με τη βοήθεια των παλμών συγχρονισμού που παράγει το κύριο (Master) ολοκληρωμένο κύκλωμα στη γραμμή SCL. Κατά τη διάρκεια που ο παλμός του ωρολογιακού σήματος ευρίσκεται σε υψηλή στάθμη (HIGH), τα δεδομένα στη γραμμή SDA δεν πρέπει να αλλάζουν. Τα δεδομένα μπορούν να αλλάξουν στη

γραμμή SDA όταν ο παλμός του ωρολογιακού σήματος οδηγείται σε χαμηλή στάθμη (LOW), όπως ακριβώς απεικονίζεται στο παρακάτω σχήμα.



Η αρχή της αποστολής δεδομένων μέσω της γραμμής SDA γίνεται με τη **συνθήκη εκκίνησης** (START). Η συνθήκη αυτή δημιουργείται με μια μετάβαση από υψηλή σε χαμηλή στάθμη στη γραμμή SDA, όταν το σήμα του ωρολογιακού σήματος είναι σε λογική κατάσταση HIGH.

Μια άλλη, μοναδική συνθήκη κατά τη μεταφορά δεδομένων είναι η **συνθήκη τερματισμού** (STOP). Αυτή δημιουργείται όταν η κύρια διάταξη (Master) προκαλεί μετάβαση της γραμμής SDA από χαμηλή στάθμη σε υψηλή στάθμη, ενώ το σήμα του ωρολογιακού σήματος ευρίσκεται στη λογική κατάσταση LOW. Παρακάτω απεικονίζονται οι συνθήκες START και STOP όπως ακριβώς έγινε η περιγραφή τους παραπάνω.



Τα δεδομένα μεταφέρονται στο δίαυλο ομαδοποιημένα κατά bytes (δηλαδή ομάδες των 8 bits). Ανάμεσα στα bytes που μεταφέρονται από τον πομπό προς τον δέκτη, ο δέκτης παράγει έναν **παλμό επιβεβαίωσης** ACK (acknowledge). Κατά τον παλμό ACK ο δέκτης μεταφέρει τη γραμμή SDA σε χαμηλή λογική κατάσταση, κατά τον παλμό του ωρολογιακού σήματος που

ακολουθεί μετά από ένα πλήρες byte. Με τον παλμό ACK ο δέκτης πληροφορεί τον πομπό ότι έχει λάβει το προηγούμενο byte.

Μια διαδοχή από μεταφερόμενα bytes τερματίζεται όταν η κύρια διάταξη (Master) δημιουργήσει μια συνθήκη STOP, δηλαδή προκαλέσει τη μεταβολή της γραμμής SDA από 0 σε 1, ενώ το ωρολογιακό σήμα ευρίσκεται σε υψηλή λογική στάθμη. Παρακάτω παρουσιάζονται οι ρουτίνες START, STOP, αποστολής χαρακτήρων, καθώς και διαβάσματος.

```
/*   #define SDA  p1_7
     #define SCL  p1_6   */
void i2c_start()
{
      SDA = 0;
      delay(5);
      SCL = 0;
      delay(5);
}

void i2c_stop()
{
      SDA = 0;
      delay(5);
      SCL = 1;
      delay(5);
      SDA = 1;
      delay(5);    //45KHz
}

void i2c_send(unsigned char b)
{
      unsigned char mask;
      mask = 0x80;        //10000000
      do
      {
            if(b & mask) SDA = 1;
            else SDA = 0;
            SCL = 1;
            delay(5);
            SCL = 0;
            delay(5);
            mask = mask/2;    //0x40 1000000  45Khz
      }
```

```
      while(mask > 0);
      SDA = 1;
      SCL = 1;
      delay(5);
      SCL = 0;
      delay(5);
}
unsigned char i2c_read(void)
{
      unsigned char mask,data;
      data = 0x00;
      mask = 0x80;
      SDA = 1;
      do
      {
            pd1_7 = 0;        // Port P1_7 input(port direction)
            SCL = 1;
            delay(5);
            if(SDA ==1)data = data | mask;
            SCL = 0;
            delay(5);
            mask = mask/2;
            pd1_7 = 1;        // Port P1_7 output(port direction)
      }
      while(mask > 0);            //SDA = 0;
      delay(5);
      SCL = 1;
      delay(5);
      SCL = 0;
      delay(5);
      return data;
}

// 79=01001111 W/R A0=1, A1=1, A2=1
unsigned char input_check (unsigned int adress)
{   unsigned char re;
    i2c_start();
    i2c_send(adress); //adress=79
    re = i2c_read();
    i2c_stop();
    return re;

}
```

```
// 64 OR 0X40 R/W=0 A0=0, A1=0, A2=0
void output_control (unsigned char adress, unsigned char buffer)  //
{                                                          //output control
    i2c_start();
    i2c_send(adress);//adress=64
    i2c_send(buffer);
    i2c_stop();
}
/*         adress=64=1000000        */
          /* energize P0=1*/
unsigned char relay0_on(unsigned int adress)
{
  buffer=buffer | 0x01;
  output_control(adress,buffer);
  return ON;
  }
          /* energize P1=1*/
unsigned char relay1_on(unsigned int adress)
{
  buffer=buffer | 0x02;
  output_control(adress,buffer);
  return ON;
  }
          /* energize P2=1*/
unsigned char relay2_on(unsigned int adress)
{
  buffer=buffer | 0x04;
  output_control(adress,buffer);
  return ON;
  }
        /* energize P3=1*/
unsigned char relay3_on(unsigned int adress)
{
  buffer=buffer | 0x08;
  output_control(adress,buffer);
  return ON;
  }
       /* energize P4=1*/
unsigned char relay4_on(unsigned int adress)
{
  buffer=buffer | 0x010;
  output_control(adress,buffer);
  return ON;
  }
```

```
       /* energize P5=1*/
unsigned char relay5_on(unsigned int adress)
{
  buffer=buffer | 0x20;
  output_control(adress,buffer);
  return ON;
  }


       /* energize P6=1*/
unsigned char relay6_on(unsigned int adress)
{
  buffer=buffer | 0x40;
  output_control(adress,buffer);
  return ON;
  }
       /* energize P7=1*/
unsigned char relay7_on(unsigned int adress)
{
  buffer=buffer | 0x80;
  output_control(adress,buffer);
  return ON;
  }
      /* energize P0=0*/
unsigned char relay0_off(unsigned int adress)
{
  buffer=buffer & 0xfe;
  output_control(adress,buffer);
  return OFF;
  }
      /* energize P1=0*/
unsigned char  relay1_off(unsigned int adress)
{
  buffer=buffer & 0xfd;
  output_control(adress,buffer);
  return OFF;
  }
       /*  energize P2=0*/
unsigned char relay2_off(unsigned int adress)
{
  buffer=buffer & 0xfb;
  output_control(adress,buffer);
  return OFF;
  }
```

```
      /*   energize P3=0*/
unsigned char relay3_off(unsigned int adress)
{
  buffer=buffer & 0xf7;
  output_control(adress,buffer);
  return OFF;
  }
       /*  energize P4=0*/
unsigned char relay4_off(unsigned int adress)
{
  buffer=buffer & 0xef;
  output_control(adress,buffer);
  return OFF;
  }
       /*    energize P5=0*/
unsigned char relay5_off(unsigned int adress)
{
  buffer=buffer & 0xdf;
  output_control(adress,buffer);
  return OFF;
  }
      /*    energize P6=0*/
unsigned char relay6_off(unsigned int adress)
{
  buffer=buffer & 0xbf;
  output_control(adress,buffer);
  return OFF;
  }
      /*    energize P6=0*/
unsigned char relay7_off(unsigned int adress)
{
  buffer=buffer & 0x7f;
  output_control(adress,buffer);
  return OFF;
  }
```

6.2 Το πρωτόκολλο RS–232C

Το πρωτόκολλο RS–232C επιτρέπει την ασύγχρονη σειριακή επικοινωνία ανάμεσα σε δύο συσκευές. Εάν επιθυμούμε να συνδέσουμε περισσότερες από δύο συσκευές σε έναν υπολογιστή, χρειαζόμαστε περισσότερες από μια σειριακές θύρες. Υπάρχουν, βέβαια, και άλλα σειριακά πρωτόκολλα, όπως το πρωτόκολλο σύγχρονης επικοινωνίας I^2C, που επιτρέπουν τη διασύνδεση πολλών συσκευών σε ένα σειριακό κύκλωμα.

Το πρωτόκολλο RS-232C χρησιμοποιεί αρνητική ψηφιακή λογική και μεγάλες στάθμες, ώστε να επιτρέπει τη διάδοση του σήματος σε μεγάλες αποστάσεις χωρίς απώλειες. Αυτά έχουν σαν αποτέλεσμα οι τάσεις του πρωτοκόλλου RS-232C να μην είναι συμβατές με τις στάθμες TTL.

Τα επίπεδα τάσεων του πρωτοκόλλου RS-232C, σύμφωνα με τις προδιαγραφές που θέσπισε η ένωση E.I.A., είναι τα εξής:

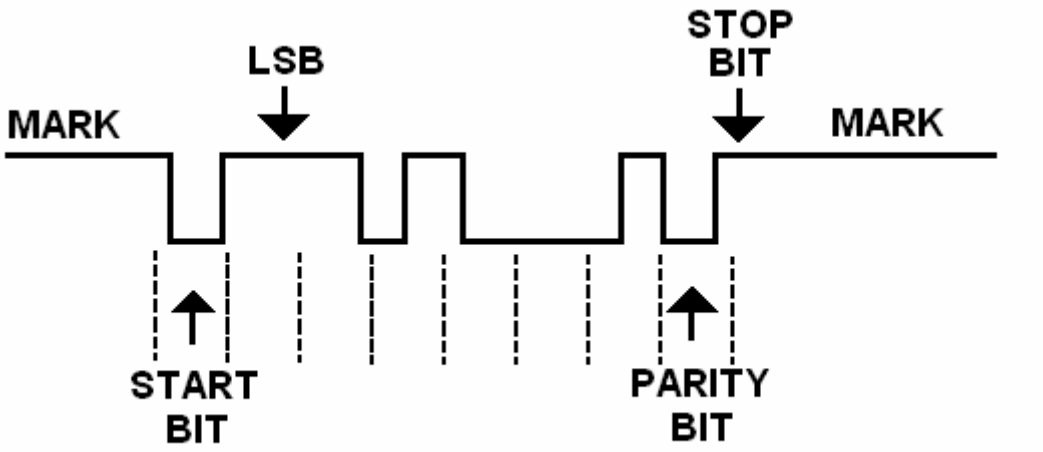
ΠΡΟΔΙΑΓΡΑΦΕΣ ΠΡΩΤΟΚΟΛΛΟΥ RS-232C

1. Το λογικό 0, που λέγεται και **SPACE**, βρίσκεται μεταξύ +3V και +25V (στην πράξη λαμβάνονται και εκπέμπονται από +5V έως και +15V).
2. Το λογικό 1, που λέγεται και **MARK**, βρίσκεται μεταξύ -3V και -25V (στην πράξη λαμβάνονται και εκπέμπονται από -5V έως και -15V).
3. Η περιοχή από -3V έως και +3V δεν αντιπροσωπεύει καθορισμένη λογική στάθμη.
4. Κανένας από τους ακροδέκτες της σειριακής θύρας δεν μπορεί να δεχτεί δυναμικό μεγαλύτερο από 25V σε σχέση με την πηγή.
5. Το μέγιστο ρεύμα δεν μπορεί να ξεπερνά τα 500mA.

Αν και σύμφωνα με το πρωτόκολλο ο μέγιστος ρυθμός μετάδοσης (baud rate) δεν ξεπερνά τα 19.2 Kbps, οι σημερινές ταχύτητες μπορεί να είναι σαφώς μεγαλύτερες.

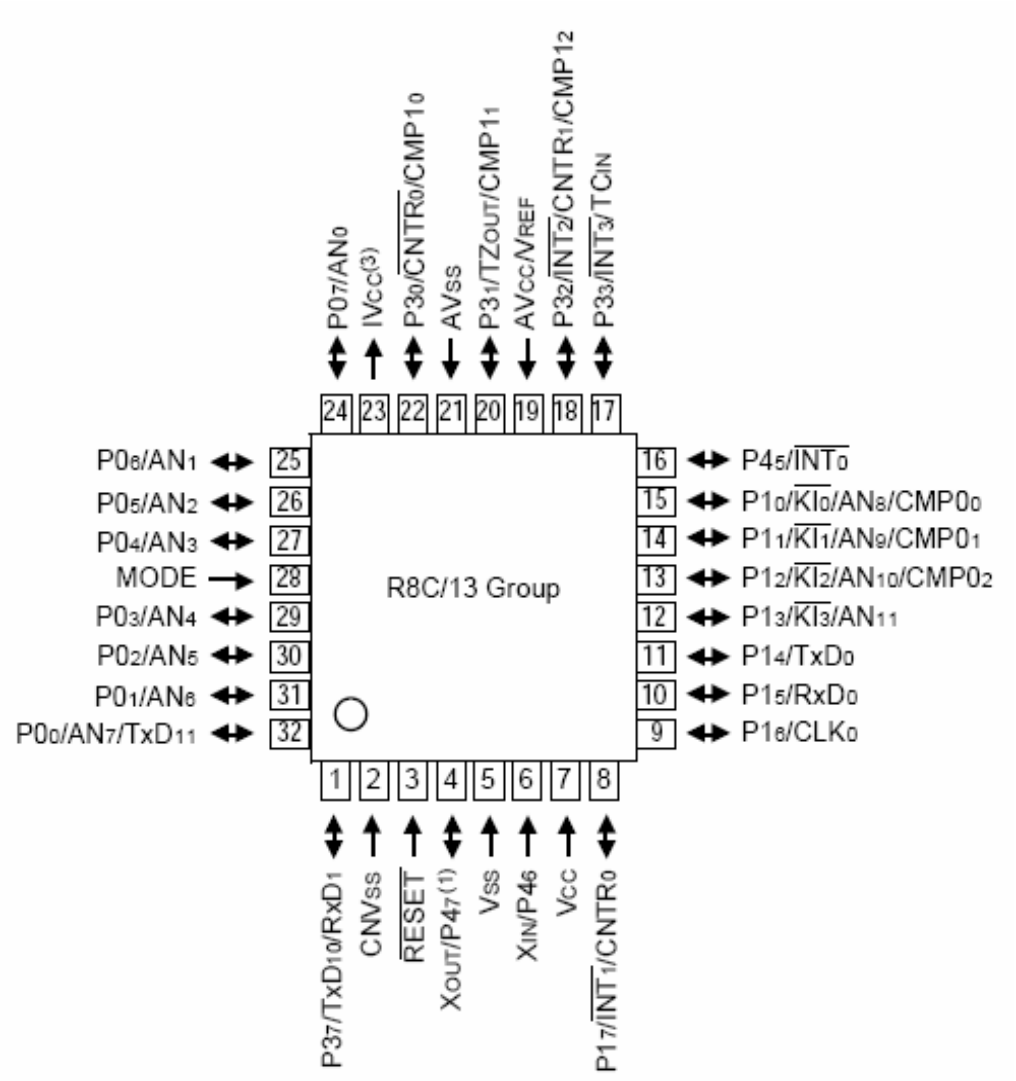
Με χρήση του πρωτοκόλλου RS-232C, ένα τερματικό χαρακτήρων (ASCII terminal) μπορεί να αποστείλει μέσω μιας γραμμής επικοινωνίας δεδομένα, σύμφωνα με τους κανόνες της ασύγχρονης σειριακής μετάδοσης που περιγράψαμε στην παραπάνω παράγραφο.

Όταν η γραμμή είναι ανενεργή, ευρίσκεται σε συνθήκη MARK, δηλαδή -12V περίπου, που αντιστοιχούν σε λογικό 1. Η γραμμή ενεργοποιείται με τη συνθήκη SPACE (λογικό 0 ή +12V). Ακολουθεί η μετάδοση επτά ή οκτώ bits για τον αποστελλόμενο χαρακτήρα, ένα προαιρετικό bit άρτιας ή περιττής ισοτιμίας (parity) και ένα ή δύο STOP bits (συνθήκη MARK), που σηματοδοτούν το τέλος του χαρακτήρα, όπως ακριβώς απεικονίζεται και στο παρακάτω σχήμα.

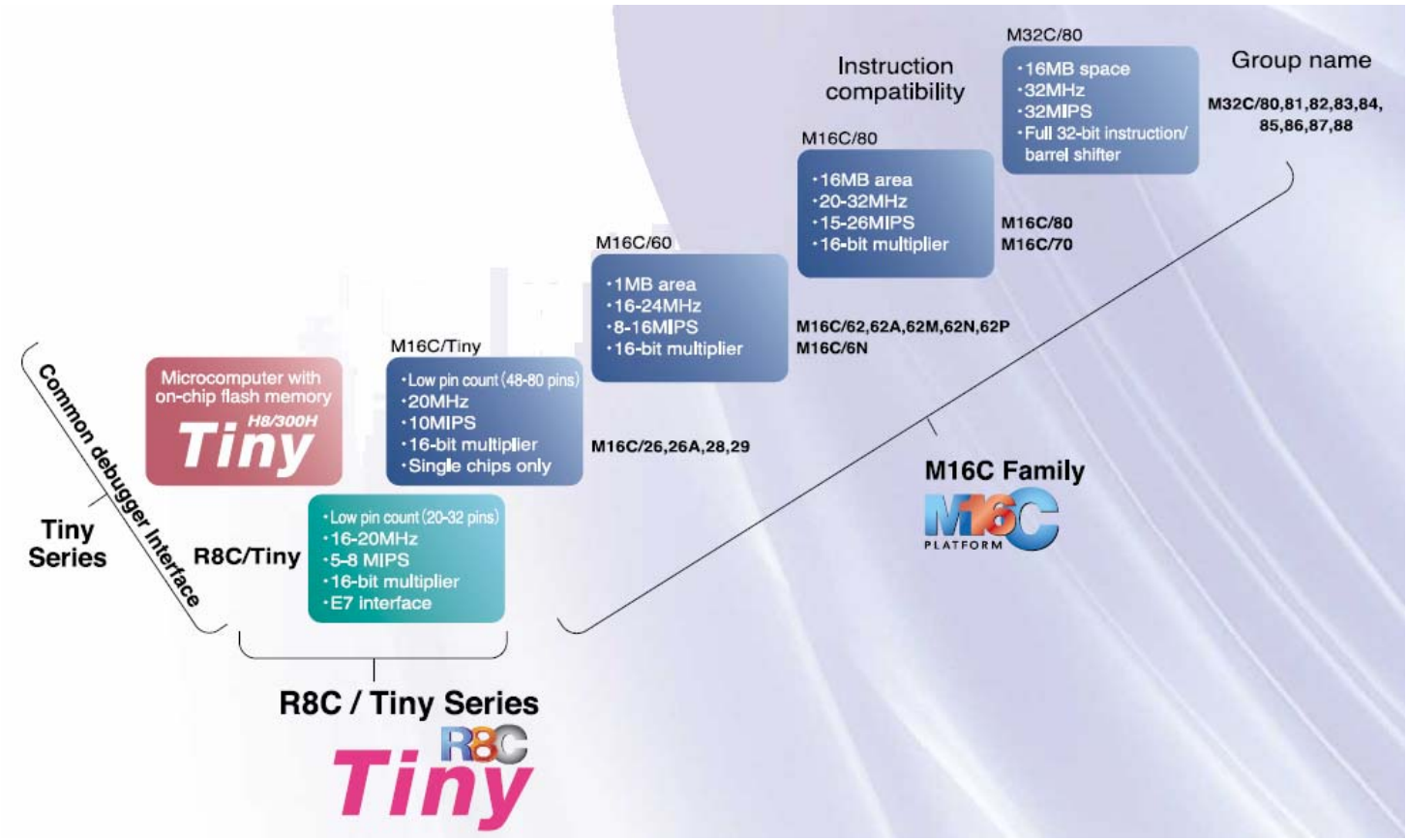


Κεφάλαιο 7

Σε αυτό το κεφάλαιο παρουσιάζεται μια γενική περιγραφή του επεξεργαστή R8C/13, ο οποίος είναι αυτός που απεικονίζεται παρακάτω. Παρατηρούμε ότι ο κάθε ακροδέκτης έχει περισσότερες από μια δυνατότητες, αυτό επιτυγχάνεται ενεργοποιώντας τα αντίστοιχα bit στο πρόγραμμα.

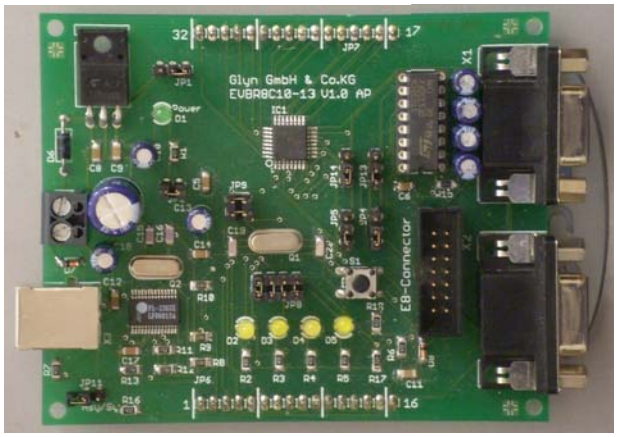


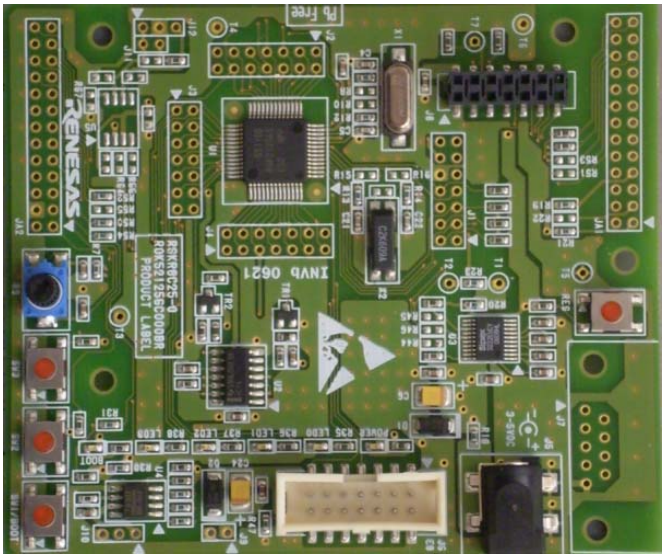
Παρακάτω απεικονίζεται μια φωτογραφία με μια αναδρομή στους επεξεργαστές της εταιρείας RENESAS (πηγή φωτογραφίας από κατάλογο της εταιρείας Σεπτέμβριος 2004).

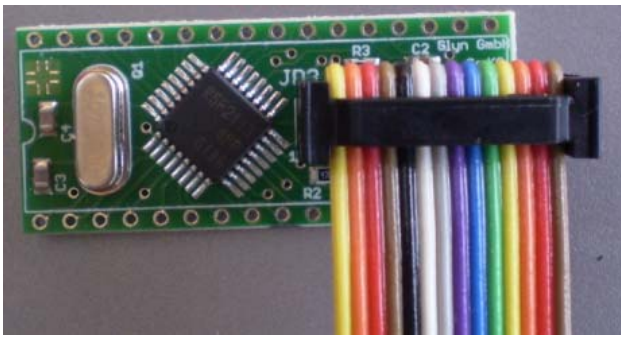


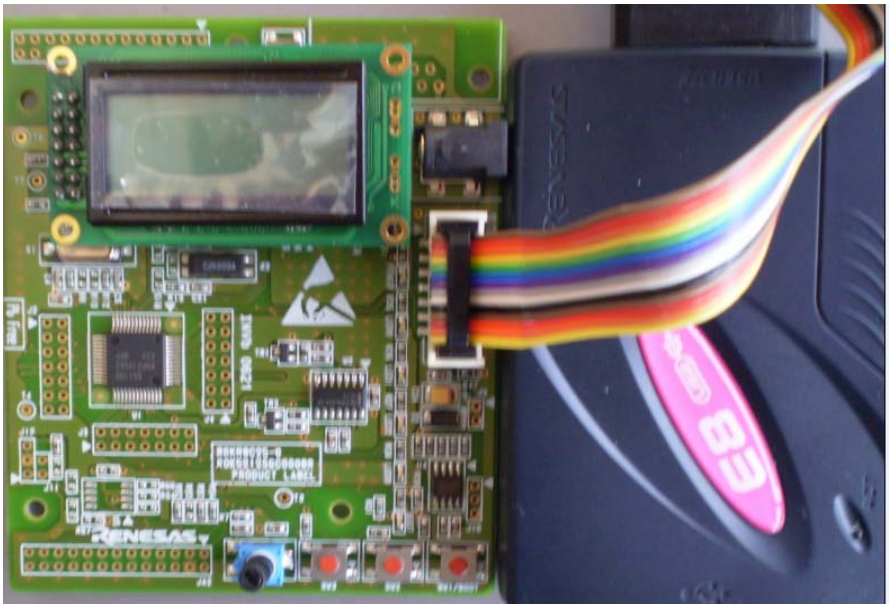
7.1 Προεπισκόπηση επεξεργαστή

Ο επεξεργαστής R8C/13 είναι 32pin, 16bit και ανήκει στην οικογένεια M16C. Κατασκευάζεται από την Ιαπωνική εταιρεία RENESAS τεχνολογίας CMOS. Ποικίλες είναι οι εφαρμογές που βρίσκει βάση ο συγκεκριμένος επεξεργαστής όπως για παράδειγμα ηλεκτρικές οικιακές συσκευές, εξοπλισμοί γραφείων (αισθητήρια, ασφάλειες), βιομηχανικούς εξοπλισμούς, ήχος, κλπ.. Παρακάτω απεικονίζονται οι πλακέτες με τον επεξεργαστή, τον debugger, το αναπτυξιακό του επεξεργαστή, καθώς και η μεταξύ τους σύνδεση.



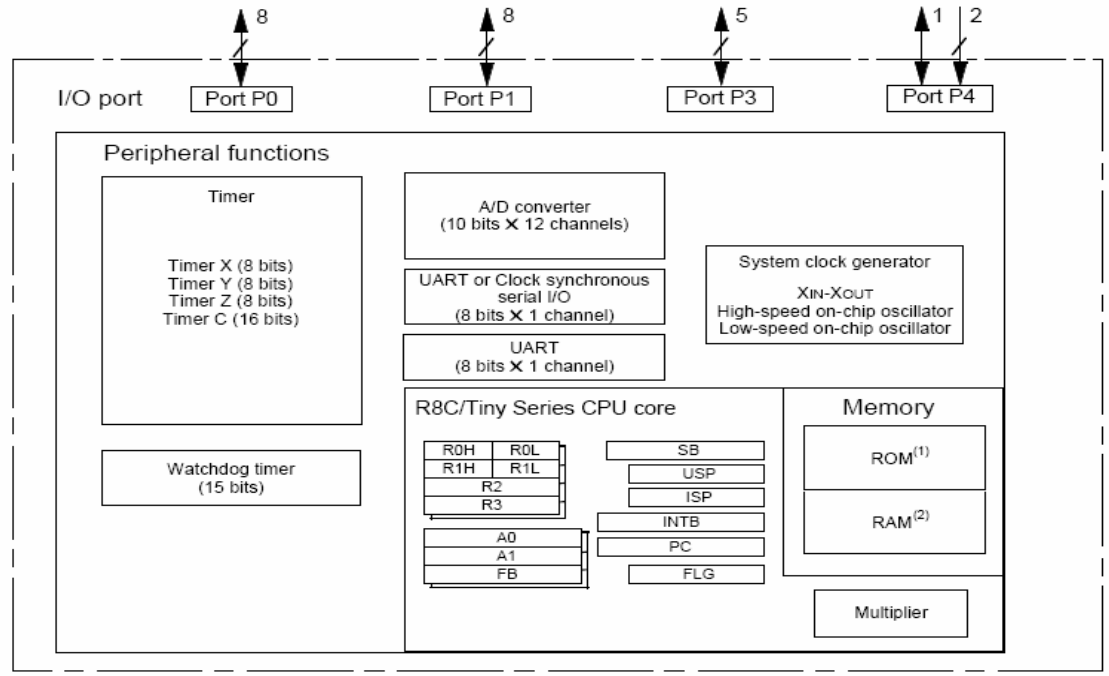






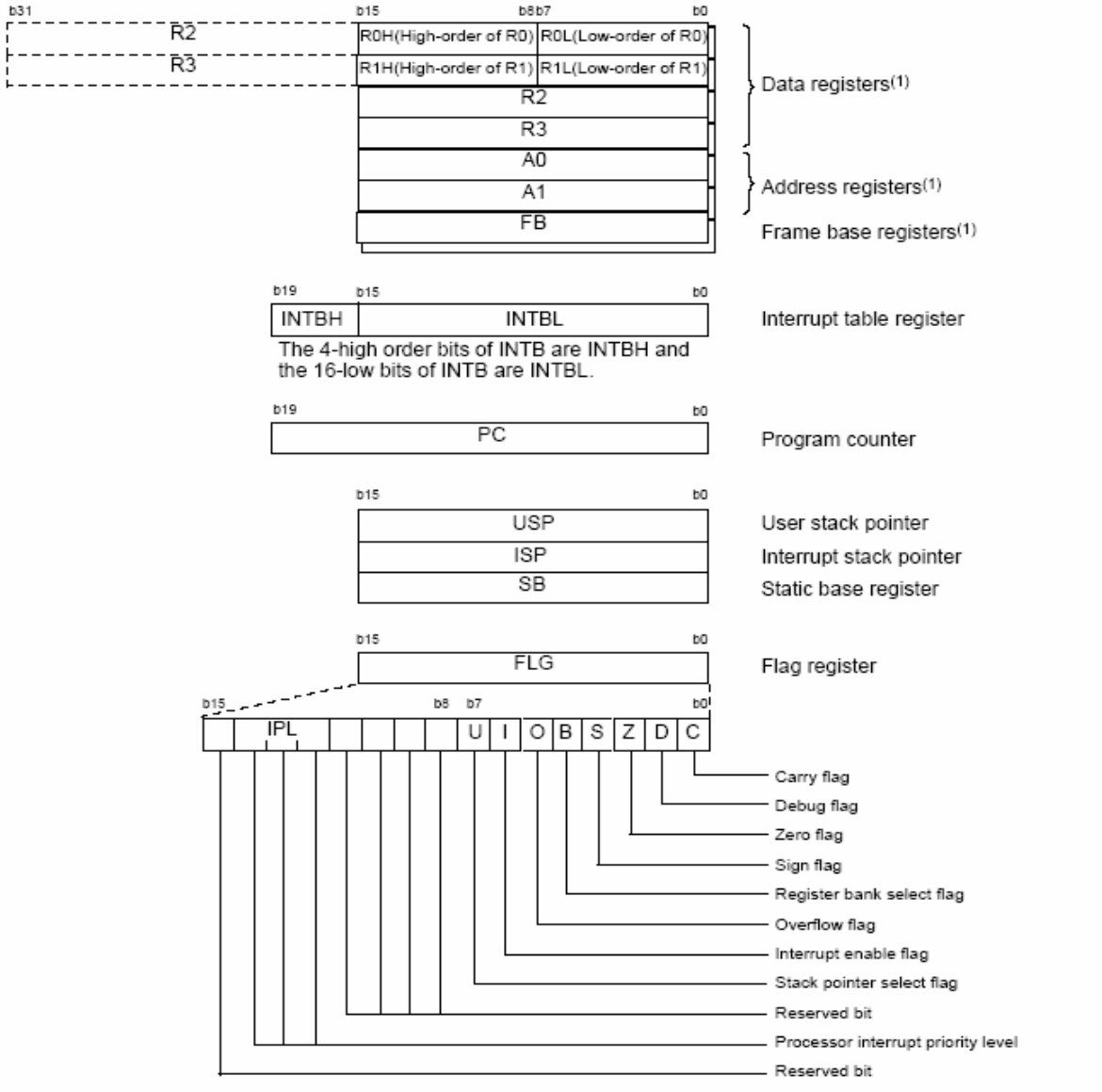
7.2 Περιγραφή επεξεργαστή

Παρακάτω απεικονίζεται το εσωτερικό μπλοκ διάγραμμα του επεξεργαστή.



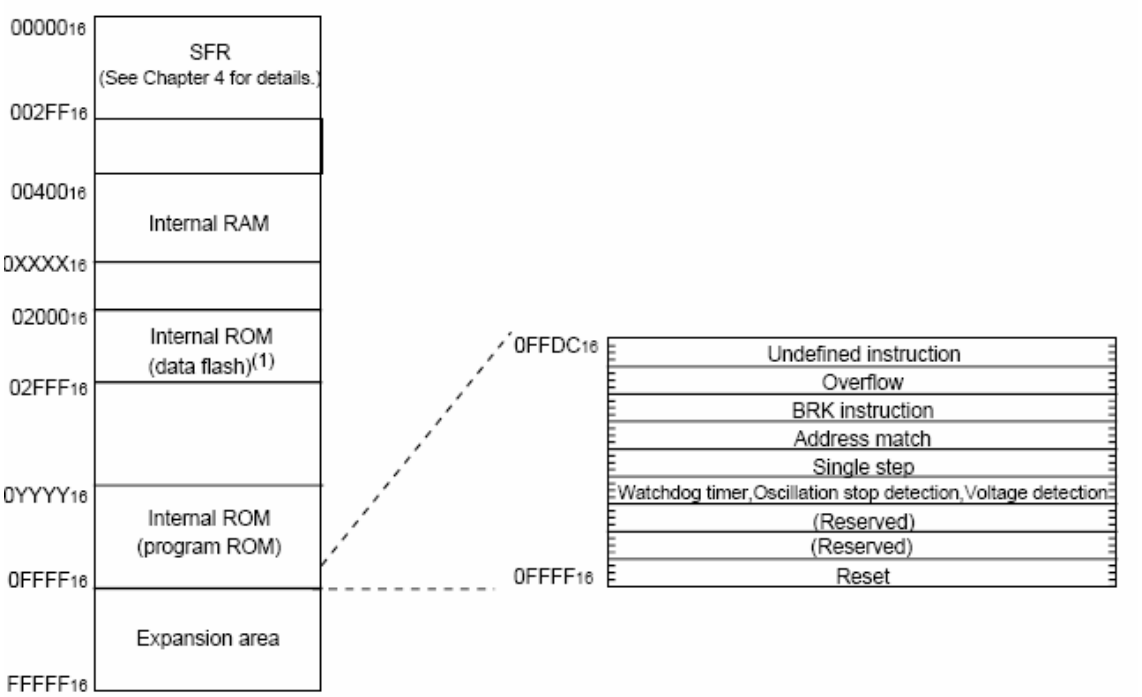
Από το παραπάνω μπλοκ διάγραμμα παρατηρούμε τους χρονιστές (TIMER) , τον A/D (αναλογικό σε ψηφιακό μετατροπέα), τους καταχωρητές (R0, R1, R2, R3), την μνήμη (Memory), τους δίαυλους επικοινωνίας (Ports),

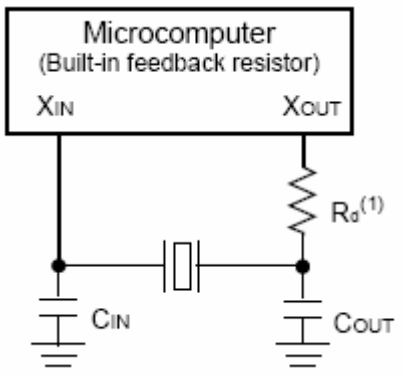
καθώς και τον εσωτερικό ταλαντωτή (System Clock Generator). Επίσης οι πόρτες εισόδου/εξόδου προγραμματίζοντας ’τες ανάλογα λειτουργούν είτε σαν είσοδοι, είτε σαν έξοδοι, αντίστοιχα. Οι καταχωρητές R0, R1, R2, R3 είναι 16bit, και έχουν την δυνατότητα να συνδεθούν μεταξύ τους ανά ζευγάρια και να γίνουν 32bit καταχωρητές. Τα ζευγάρια αυτά είναι τα R2R0, και R3R1.



Ο εσωτερικός ταλαντωτής είναι διπλός, ο ένας είναι 8MHz, και ο άλλος είναι 125KHz. Ο κρύσταλλος των 8MHz είναι για την λειτουργία του, ενώ ο κρύσταλλος των 125KHz ενεργοποιείται αυτόματα μετά από επανεκκίνηση (RESET) για να έχει ομαλή εκκίνηση. Έχει την δυνατότητα ο επεξεργαστής να χρησιμοποιήσει και εξωτερικό ταλαντωτή της τάξεως των 20MHz.

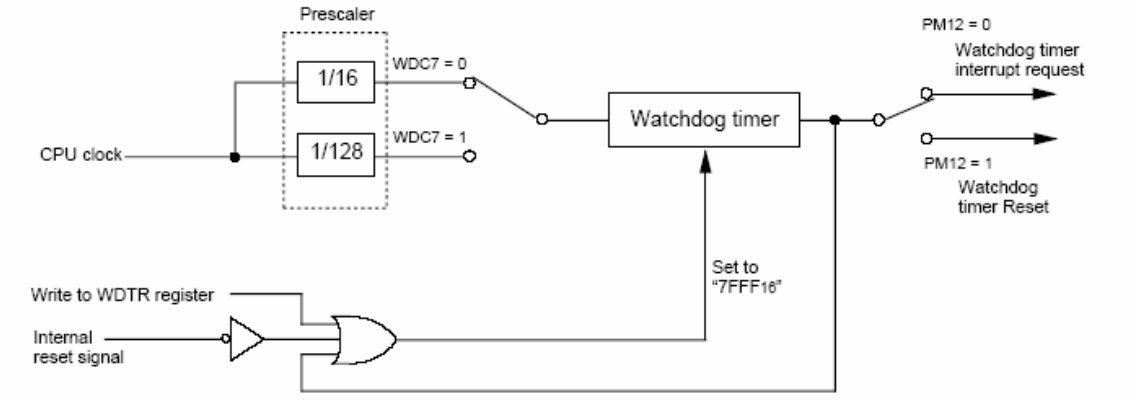
Παρακάτω απεικονίζεται η περιοχή της μνήμης του επεξεργαστή η οποία έχει μέγεθος 1MByte, και αριθμεί από την διεύθυνση 00000_{16} έως την $FFFFF_{16}$.





Στην διπλανή φωτογραφία απεικονίζεται ο τρόπος με τον οποίο πρέπει να τοποθετηθεί εξωτερικά ένας κρύσταλλος. Η συνδεσμολογία του κρυστάλλου γίνεται στα άκρα Xin, και Xout. Η αντίσταση είναι για την ανάδραση η οποία αποσυνδέεται κατά την λειτουργία κράτησης μειώνοντας έτσι την κατανάλωση ισχύος μέσα στο ολοκληρωμένο. Όταν εφαρμοστεί ένας εξωτερικός παλμός τότε το κύριο ρολόι δεν έχει την δυνατότητα να τεθεί εκτός λειτουργίας. Όταν ενεργοποιείται αυτή η λειτουργία τότε όλα τα ρολόγια, συμπεριλαμβανομένου και του κύριου, τίθενται εκτός λειτουργίας.

Παρακάτω απεικονίζεται ο χρονιστής επιτήρησης εξωτερικού κρυστάλλου, ο οποίος ανιχνεύει πότε το πρόγραμμα τίθεται εκτός λειτουργίας, με αποτέλεσμα η χρήση του να είναι αναγκαία, ακόμα και από τον κατασκευαστή, για να έχουμε καλύτερη αξιοπιστία στο πρόγραμμα. Ο χρονιστής επιτήρησης είναι ένας 16bit απαριθμητής ο οποίος μετράει προς τα κάτω, ως παλμό χρησιμοποιεί τον ίδιο με την CPU αλλά χρησιμοποιεί και έναν προδιαιρέτη (δια 16, δια 128) ανάλογα με το πιο bit είναι ενεργοποιημένο εκείνη την στιγμή.

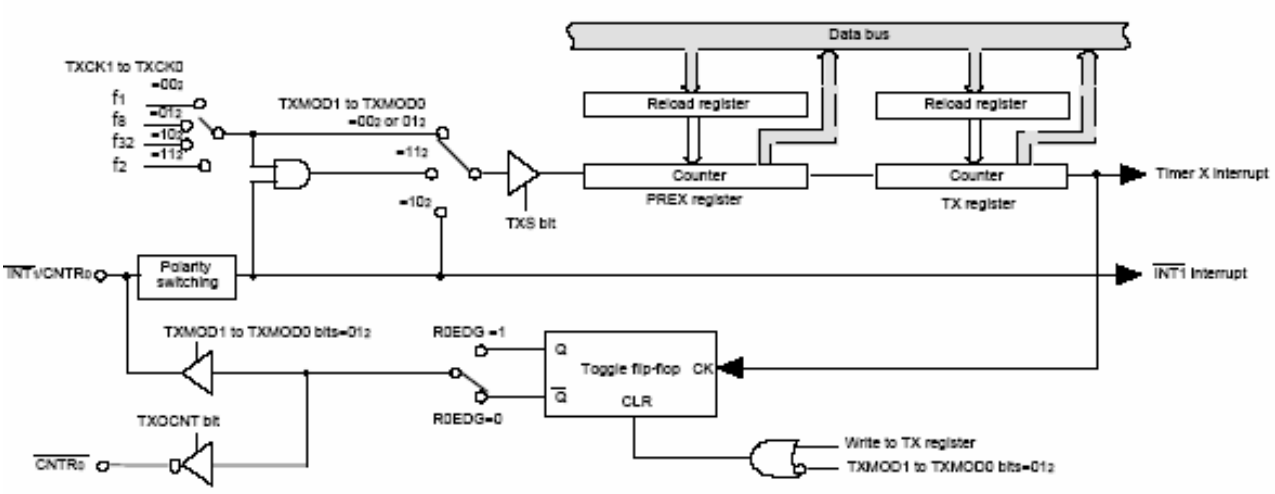


Η περίοδος του χρονιστή υπολογίζεται από τον παρακάτω τύπο:

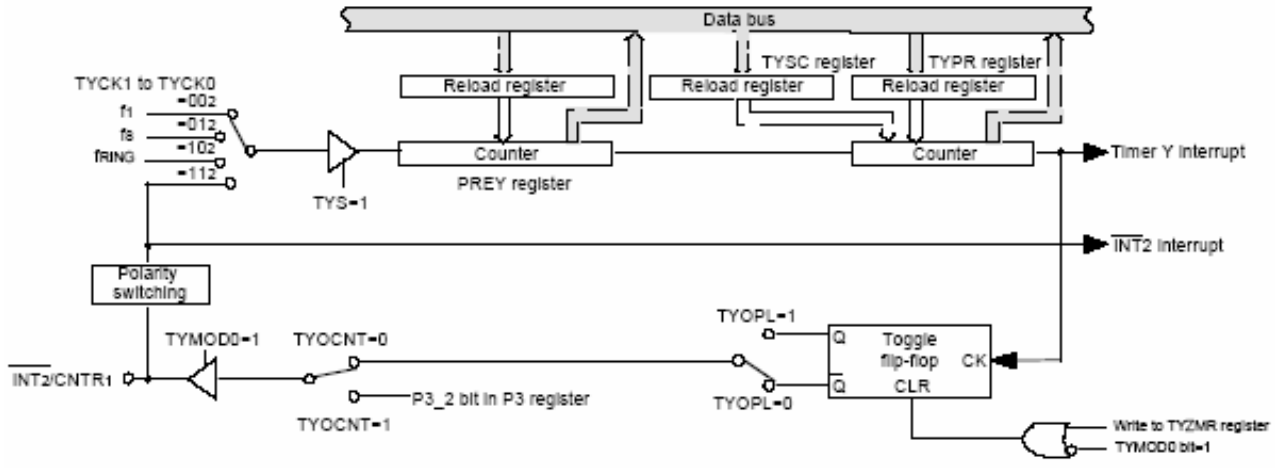
$$T = \frac{προδιαιρέτης(16\_ή\_128)x32768}{ρολόι\_CPU}$$

Σχετικά με τους χρονιστές που περιλαμβάνονται στον επεξεργαστή είναι τέσσερις (4). Οι τρεις (3) από αυτούς (timer x, timer y, timer z) είναι 8bit και έχει ο καθένας και έναν 8bit προδιαιρέτη, ενώ ο timer c είναι 16bit και κάνει σύλληψη εισόδου και σύγκριση εξόδου.

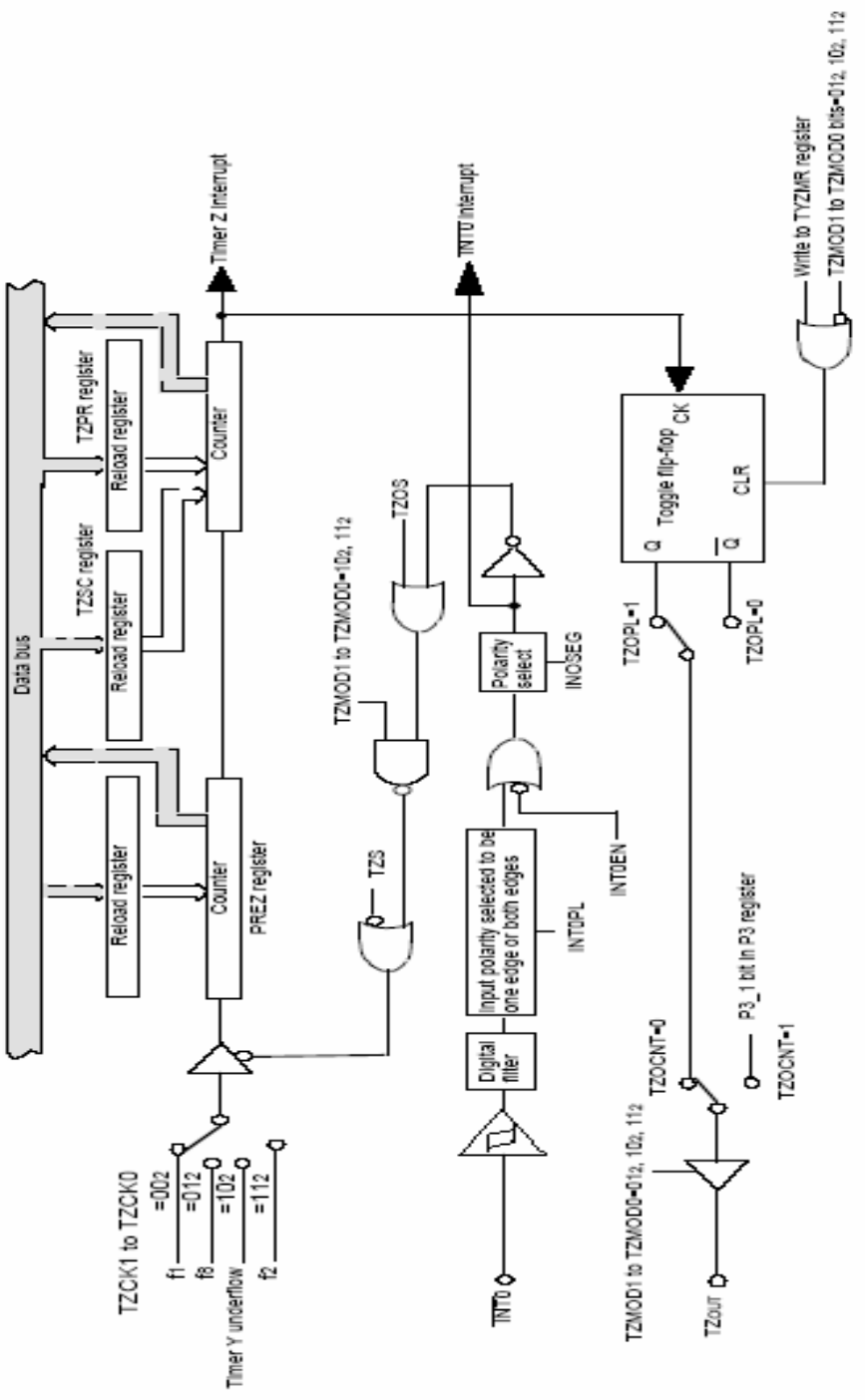
Αναλυτικότερα ο ***timer x*** έχει την δυνατότητα να λειτουργήσει ως χρονιστής, να παράγει παλμούς στην έξοδο, να μετράει τους εξοτερικούς παλμούς, να μετράει το εύρος των παλμών, και τέλος να μετράει την περίοδο των παλμών. Παρακάτω απεικονίζεται το εσωτερικό μπλοκ διάγραμμα του timer x.



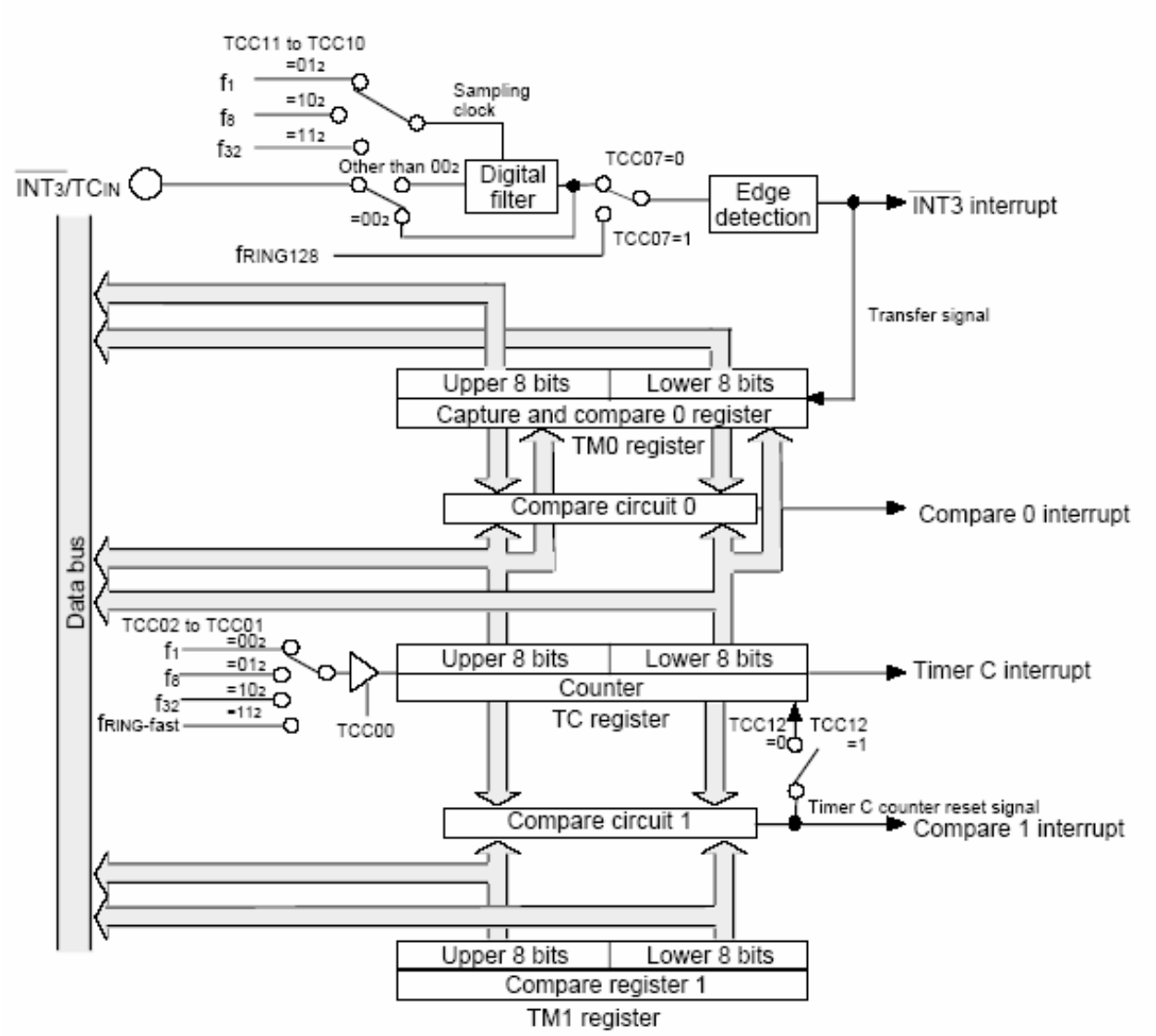
Αναλυτικότερα ο ***timer y*** έχει δύο καταχωρητές, έναν πρωτεύων καθώς και έναν δευτερεύων, ακόμα έχει την δυνατότητα να λειτουργήσει ως χρονιστής, και να δημιουργήσει προγραμματιζόμενη κυματομορφή στην έξοδο. Παρακάτω απεικονίζεται το εσωτερικό μπλοκ διάγραμμα του timer y.



Αναλυτικότερα ο ***timer z*** έχει δύο καταχωρητές, έναν πρωτεύων καθώς και έναν δευτερεύων, ακόμα έχει την δυνατότητα να λειτουργήσει ως χρονιστής, ως προγραμματιζόμενη κυματομορφή στην έξοδο, ως προγραμματιζόμενη κυματομορφή στην έξοδο με έναν μόνο παλμό, καθώς και ως προγραμματιζόμενη κυματομορφή στην έξοδο με έναν μόνο παλμό αλλά με εμφάνισή του μετά από κάποιον χρόνο. Παρακάτω απεικονίζεται το εσωτερικό μπλοκ διάγραμμα του timer z.

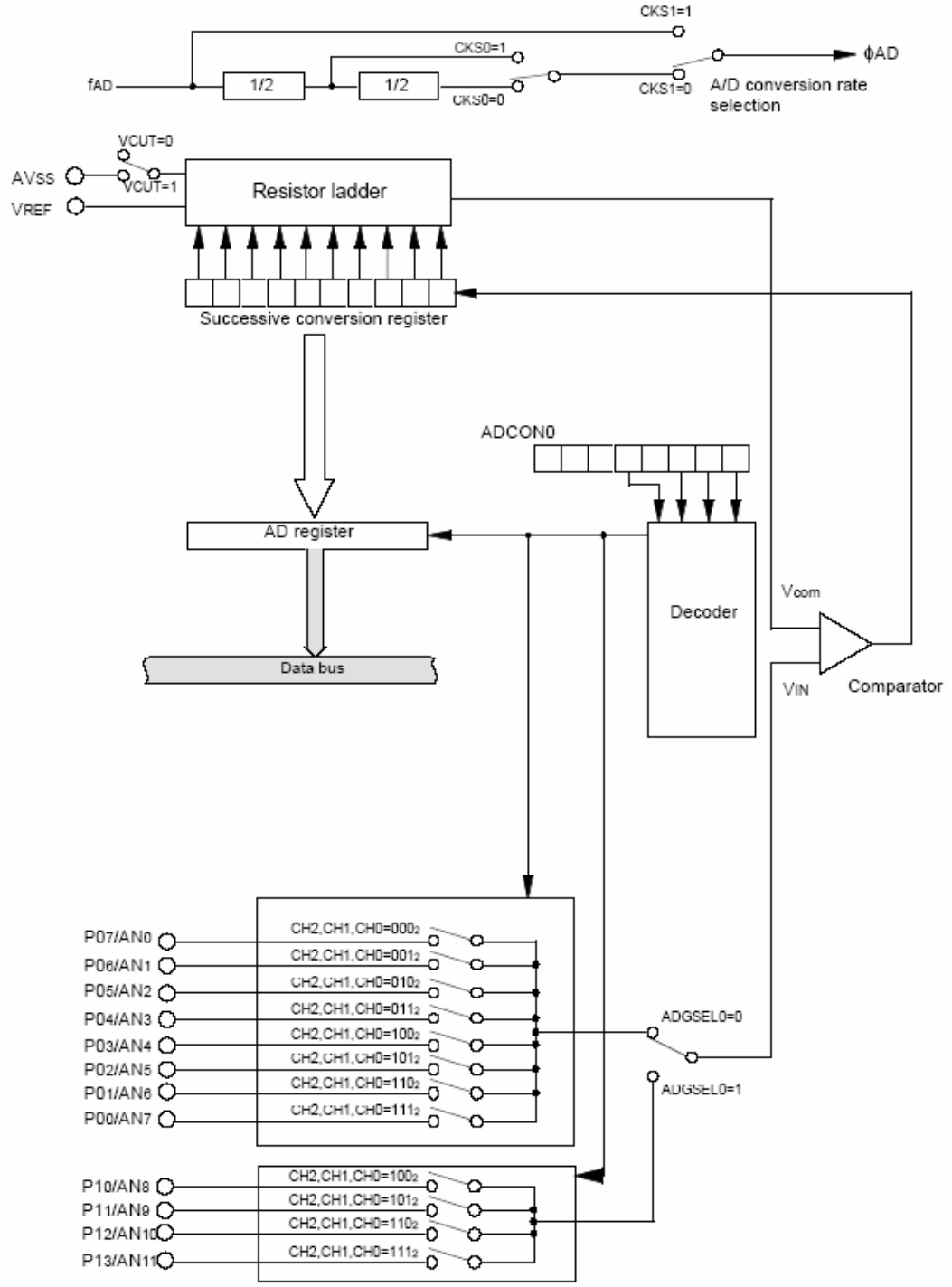


Παρακάτω απεικονίζεται ο ***timer c*** ο οποίος είναι 16bit και έχει την δυνατότητα να λειτουργήσει για σύλληψη της εισόδου, και για σύγκριση της εξόδου. Ο timer c δεν έχει χρησιμοποιηθεί στην κατασκευή μας.



Όσον αφορά τον μετατροπέα A/D, αυτός αποτελείται από ένα διαδοχικό κύκλωμα μετατροπέων προσέγγισης 10bit με έναν ενισχυτή χωρητικής σύζευξης. Οι αναλογικές είσοδοι μοιράζονται τα pin $P0_0$ $P0_7$ και $P1_0$ $P1_3$. Επομένως, όταν χρησιμοποιούνται αυτά τα pin, πρέπει να σιγουρευτούμε ότι τα αντίστοιχα bits τίθενται "0". Όταν δεν χρησιμοποιείται ο μετατροπέας A/D τότε θέτουμε το bit του V_{CUT} ίσο με "0", έτσι ώστε κανένα ρεύμα να μην διαρρεύσει από το pin του V_{REF} στη βαθμίδα με τις αντιστάσεις, όπου βοηθούν στην κατανάλωση ισχύος του chip. Όταν ο μετατροπέας είναι 8bit τότε οι βαθμίδες που δημιουργούνται

είναι 2^8=256 βαθμίδες, ενώ όταν ο μετατροπέας είναι 10bit τότε οι βαθμίδες που δημιουργούνται είναι 2^{10}=1024 βαθμίδες. Παρακάτω απεικονίζεται το εσωτερικό μπλοκ διάγραμμα του μετατροπέα A/D.

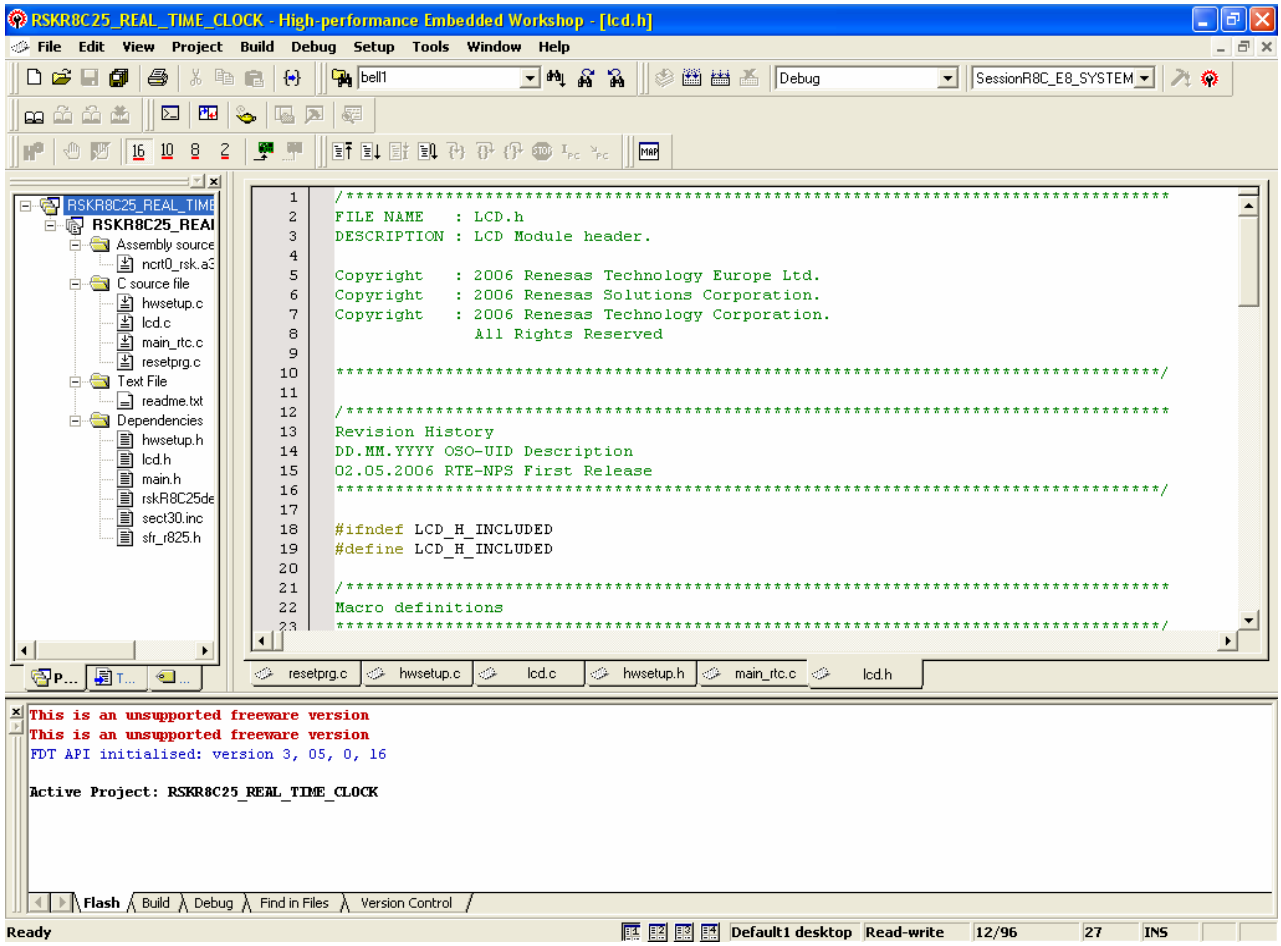


Ο μετατροπέας A/D έχει την ικανότητα να λειτουργήσει με τις τρεις επακόλουθες μεθόδους πρώτον την συνεχή (repeat mode), δεύτερον την επιλεκτική (one shot mode), και τρίτον την sample and hold.

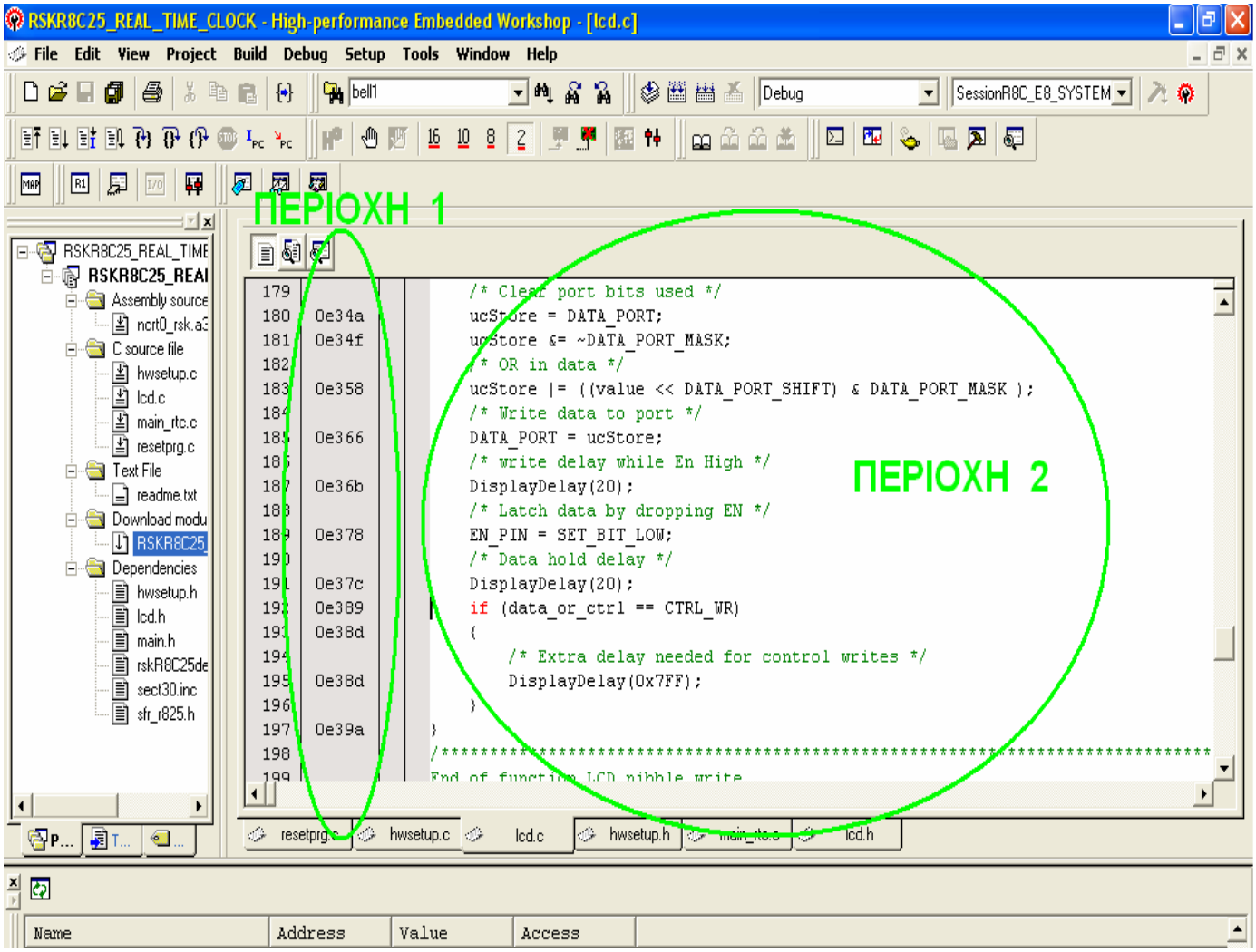
Κεφάλαιο 8

Σε αυτό το κεφάλαιο παρουσιάζεται το ***πρόγραμμα μεταγλώττισης***, όπου ανοίξαμε τον κώδικα στον υπολογιστή για να τον περάσουμε αργότερα στον επεξεργαστή. Το πρόγραμμα όπου χρησιμοποιήσαμε είναι το HEW της RENESAS. Ο compiler είναι ο NC30 και ανήκει στην εταιρεία GNU.

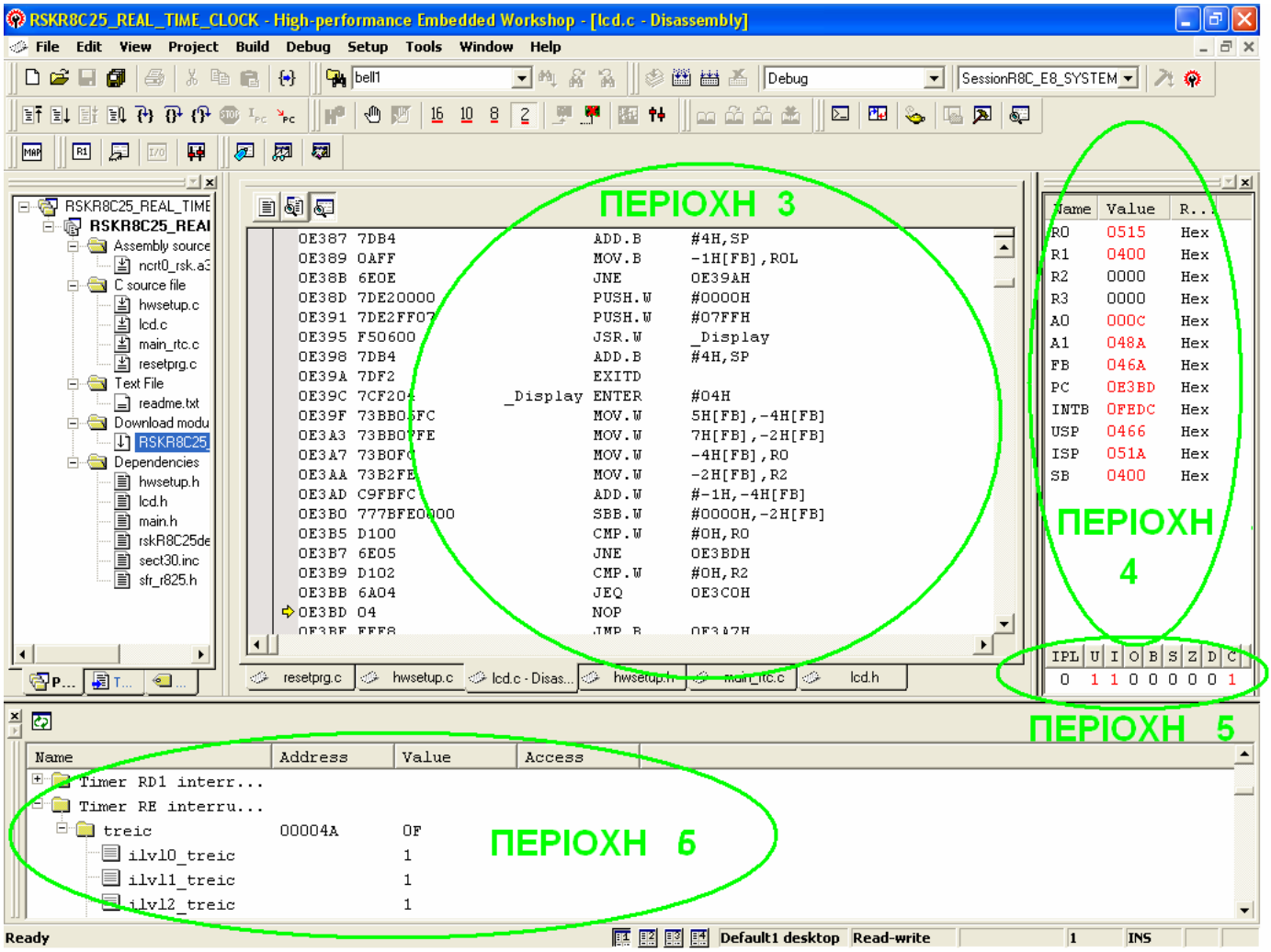
Το γραφικό του περιβάλλον του προγράμματος το καθιστά αρκετά εύκολο στη χρήση του.



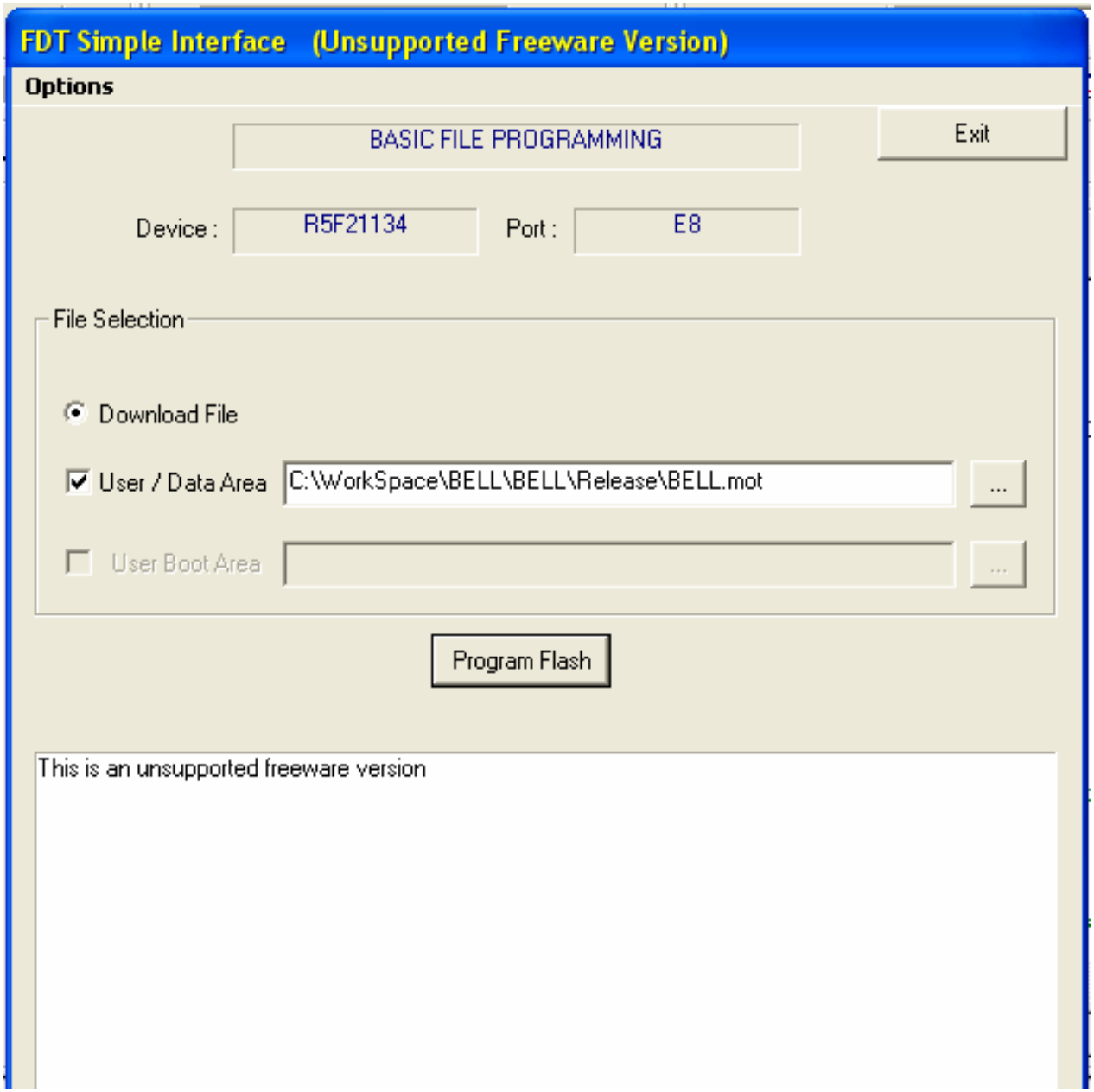
Παρακάτω φαίνονται στην **περιοχή 1** η περιοχή μνήμης όπου καταλαμβάνεται από την αντίστοιχη γραμμή του κώδικα, καθώς και στην **περιοχή 2** όπου απεικονίζεται ο κώδικας σε γλώσσα προγραμματισμού C.



Παρακάτω φαίνονται στην **περιοχή 3** όλες οι εντολές του κώδικα μας σε γλώσσα προγραμματισμού assembly, στην **περιοχή 4** όλους τους καταχωρητές με την τιμή όπου έχουν την συγκεκριμένη στιγμή, στην **περιοχή 5** τα flag του επεξεργαστή, και τέλος στην **περιοχή 6** όλους τους καταχωρητές όπου μπορούν να χρησιμοποιηθούν.



Παρακάτω παρουσιάζεται το ***πρόγραμμα φορτώματος του HEX αρχείου στον επεξεργαστή***, όπου περάσαμε τον κώδικα από τον υπολογιστή στον επεξεργαστή.



ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Ι. Καλόμοιρος – Σ. Μπουλταδάκης – Ι. Πέταλας, "Έλεγχος κυκλωμάτων και μετρήσεων με Η/Υ", εκδόσεις ΤΖΙΟΛΑ έκδοση 2002, ISBN 960-8050-67-7.

2. H. Tan – T.B. D'Orazio, "C για μηχανικούς", εκδόσεις ΤΖΙΟΛΑ έκδοση 2000, ISBN 960-8050-33-2.

3. Sartaj Sahni, "Δομές δεδομένων αλγόριθμοι και εφαρμογές C++", εκδόσεις ΤΖΙΟΛΑ έκδοση 2004, ISBN 960-418-030-4.

4. Β. Σεραφίδης, "C για αρχάριους", εκδόσεις ΚΛΕΙΔΑΡΙΘΜΟΣ έκδοση 1995 με ανατύπωση το 2002, ISBN 960-209-268-8.

5. Σημειώσεις "Ηλεκτρονικών Ισχύος" Α. Μανίτης, Χανιά 2003.

6. Manual επεξεργαστή R8C/13 RENESAS Technology.

7. Διαφημιστικό φυλλάδιο RENESAS Σεπτεμβρίου 2004.

8. www.google.com εύρεση των επακόλουθων datasheet (HEX40106B, MOC3041M, OCP-PCT4116/X, L7805, L7824, BT136, BT138, HFD2, IRF540, IRF610, LCD 4x20, PCF8574).

9. www.lookuptables.com εύρεση των δυο πινάκων του κώδικα ASCII.

ΕΥΧΑΡΙΣΤΙΕΣ

Σε αυτό το σημείο θα θέλαμε να ευχαριστήσουμε θερμά για την πολύτιμη βοήθειά τους, τους παρακάτω αναφερόμενους. Κατά αλφαβητική σειρά είναι:

1. Brigitte Wilson, display-electronik.
2. Burkhard Kainka, ak mobul bus.
3. Gerkard H. Schalk, Philps.
4. Marco Jury, Glyn.
5. Michael Belke, Glyn.
6. Pokorny Alexander, Glyn.

Εύρεση των tools όπου χρησιμοποιήσαμε.

1. tools.support.eu@renesas.com (EUROPE)
2. csc@renesas.com (JAPAN)

Site όπου χρησιμοποιήθηκαν στην εργασία.

1. www.ak-modul-bus.de
2. www.m16c.de
3. www.glyn.de
4. www.renesas.com
5. www.renesasinteractive.com
6. www.kpitgnutools.com
7. www.display-elektronik.de
8. www.elektor.de
9. www.futurlec.com
10. www.ak.digikey.com