# ΑΥΤΟΝΟΜΟ ΣΥΣΤΗΜΑ ΠΡΟΓΡΑΜΜΑΤΙΖΟΜΕΝΗΣ ΜΙΚΡΟΜΕΤΡΙΚΗΣ ΜΕΤΑΤΟΠΙΣΗΣ ΜΙΑΣ ΔΙΑΣΤΑΣΗΣ



ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

**ΒΑΜΒΑΚΑ ΑΝΑΣΤΑΣΙΑ**

**Φυσικός**

Επίβλεψη:Επίκουρος Καθηγητής Ι. Ευαγγέλου

Εργαστήριο Φυσικής Υψηλών Ενεργειών

ΔΙΑΤΜΗΜΑΤΙΚΟ ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΣΤΙΣ ΣΥΓΧΡΟΝΕΣ ΗΛΕΚΤΡΟΝΙΚΕΣ ΤΕΧΝΟΛΟΓΙΕΣ

ΤΜΗΜΑ ΦΥΣΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΙΩΑΝΝΙΝΑ ΙΟΥΝΙΟΣ 2005

# Περίληψη

Το αντικείμενο της παρούσας διπλωματικής εργασίας είναι η ανάπτυξη ενός προγραμματιζόμενου συστήματος ελέγχου ενός βηματικού κινητήρα για την οδήγηση μιας τράπεζας ακριβείας. Το σύστημα αυτό πρόκειται να χρησιμοποιηθεί για την αυτοματοποίηση μιας ημιαυτόματης συσκευής μικροσυγκόλλησης του Εργαστηρίου ΦΥΕ. Ελέγχεται είτε από αυτόνομο χειριστήριο που διαθέτει, είτε από Η/Υ με κατάλληλη εφαρμογή που αναπτύχθηκε με το πακέτο λογισμικού LabView. Η επικοινωνία γίνεται μέσω της σειριακής θύρας RS-232 του Η/Υ.

Η υλοποίηση του συστήματος βασίζεται στον μικροελεγκτή AT89S8252 της εταιρείας ATMEL για το γενικό έλεγχο και στο ζεύγος των ολοκληρωμένων L297-L298 της SGT για την κίνηση της γραμμικής τράπεζας.

Η αυτοματοποίηση της συσκευής μικροσυγκόλλησης του εργαστηρίου ΦΥΕ θα συμβάλλει ουσιαστικά στην ταχύτερη και αποτελεσματικότερη συναρμολόγηση των μικρομονάδων αισθητήρων πυριτίου του ανιχνευτή Preshower του πειράματος CMS του CERN.

# Abstract

The object of the present thesis is the development of a system, which can be programmed in order to control a stepper motor for the guiding of a precision table. The system's purpose is to convert a semi-automatic micro-bonding device, being used in ΦΥΕ's laboratory, into a fully-automatic one. It is controlled either by its own autonomous control or by a computer which uses an appropriate software application developed in LabView. The communication between the system and the computer is achieved through the computer's serial port RS-232.

The system's implementation is based on the AT89S8252 microcontroller manufactured by ATMEL Company, which performs the general control, as well as on the SGT's pair of integrated circuits L297-L298 for the functioning of the linear stage.

The micro-bonding device's automatic behavior of ΦΥΕ's laboratory will drastically contribute towards a faster and more sufficient assembling of the silicon sensors' micro units of the Preshower detector of CERN's CMS experiment.

# Ευχαριστίες

Στο σημείο αυτό θα ήθελα να ευχαριστήσω τους ανθρώπους που συνέβαλαν σημαντικά στην πραγματοποίηση αυτής της εργασίας.

Τον επιβλέποντα της διπλωματικής μου εργασίας Επίκουρο Καθηγητή κ. Ι. Ευαγγέλου μέλος του Εργαστηρίου Φυσικής Υψηλών Ενεργειών του Πανεπιστήμιου Ιωαννίνων για την αμέριστη βοήθεια και καθοδήγηση καθ' όλη τη διάρκεια εκπόνησης της εργασίας μου.

Τα μέλη της εξεταστικής επιτροπής, τον Επίκουρο Καθηγητή κ. Δ. Παπαδημητρίου μέλος του Εργαστηρίου Ηλεκτρονικής και Τηλεπικοινωνιών και τον Λέκτορα κ. Ι. Παπαδόπουλο, μέλος του Εργαστηρίου Φυσικής Υψηλών ενεργειών, για τη βοήθεια που μου προσέφεραν και προπαντός για την εμπιστοσύνη που μου έδειξαν.

Τέλος ευχαριστώ θερμά τους συνάδερφους Α. Ασημίδη, Γ.Σιδηρόπουλο, Δ. Μπολανάκη, Θ. Παντελιό, Σ. Κολοβό και Φ. Φωτίου για τη βοήθεια και τις πολύτιμες συμβουλές που μου παρείχαν στην εκπλήρωση αυτής της εργασίας.

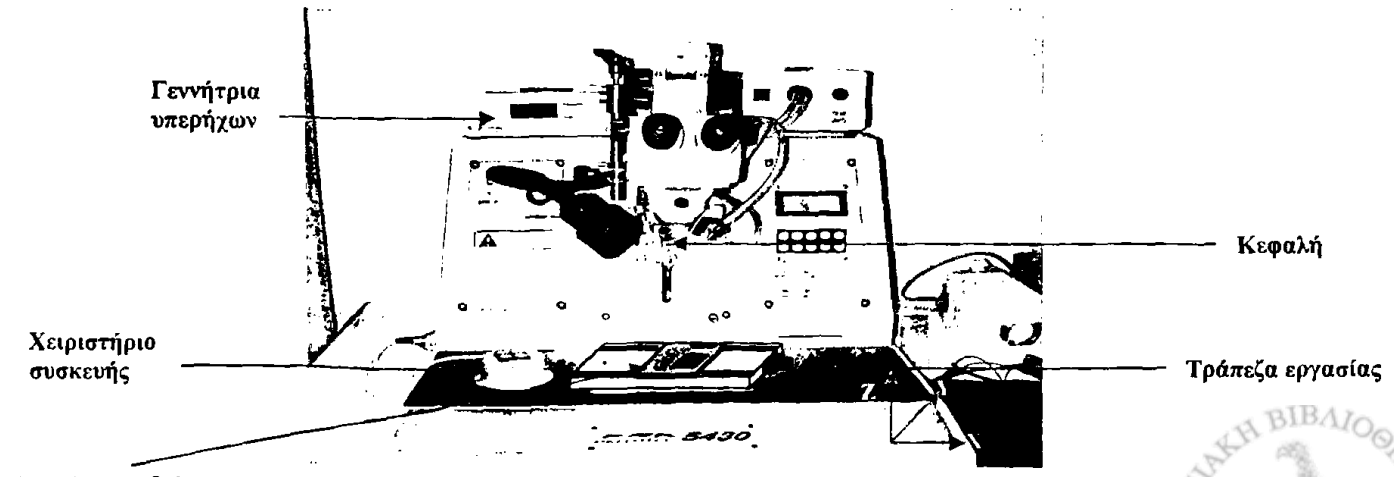
# Περιεχόμενα

# Εισαγωγή

Η παρούσα διπλωματική εργασία εκπονήθηκε στα πλαίσια του Μεταπτυχιακού Προγράμματος στις Σύγχρονες Ηλεκτρονικές Τεχνολογίες του Τμήματος Φυσικής του Πανεπιστημίου Ιωαννίνων, στις εγκαταστάσεις του Εργαστηρίου Φυσικής Υψηλών Ενεργειών.

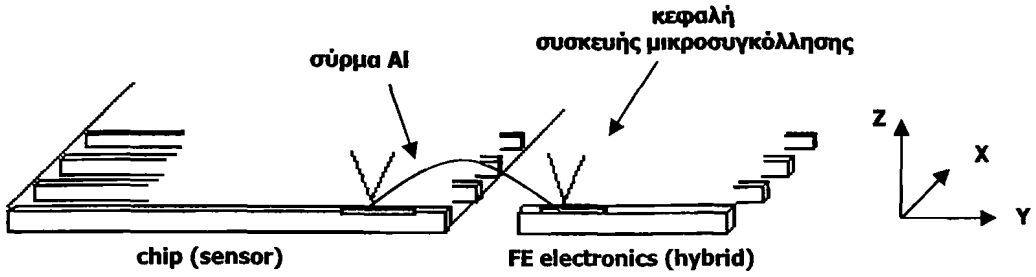
Το αντικείμενο της εργασίας αυτής είναι η ανάπτυξη και η κατασκευή ενός προγραμματιζόμενου συστήματος ελέγχου ενός βηματικού κινητήρα για την κίνηση μιας τράπεζας ακριβείας. Το σύστημα αυτό πρόκειται να χρησιμοποιηθεί για την αυτοματοποίηση της ημιαυτόματης συσκευής μικροσυγκόλλησης, τύπου 5430 της εταιρείας F&K Delvotec, του εργαστηρίου ΦΥΕ. Η συσκευή μικροσυγκόλλησης θα χρησιμοποιηθεί για την συναρμολόγηση ανιχνευτικών μικρομονάδων του επιμέρους ανιχνευτή Preshower του πειράματος CMS του CERN. Η μικροσυγκόλληση αφορά τις μικρολωρίδες των αισθητήρων πυριτίου με τα ηλεκτρονικά προενίσχυσης.

Η προγραμματιζόμενη συσκευή μικροσυγκόλλησης του εργαστηρίου ΦΥΕ, φωτογραφία της οποίας φαίνεται στο Σχήμα 1, χρησιμοποιεί σύρμα Al διαμέτρου 25μm. Η κίνηση της κεφαλής της συσκευής κατά τους άξονες Y και Z γίνεται υπό τον έλεγχο πλήρους προγραμματισμού της ίδιας συσκευής και παρέχει τη δυνατότητα απλών ή πολύπλοκων μικροσυγκολλήσεων ολοκληρωμένων κυκλωμάτων και μικροηλεκτρονικών με αξιόπιστο τρόπο. Η κίνηση της τράπεζας εργασίας κατά τους άξονες X-Y γίνεται με χειροκίνητο τρόπο με τη βοήθεια του χειριστηρίου της συσκευής. Επισημαίνεται ότι οι μικροσυγκολλήσεις γίνονται με μετακίνηση της κεφαλής κατά τον άξονα Y και φυσικά κατά τον άξονα Z για τη δημιουργία του βρόγχου του σύρματος (Σχήμα 2). Κύριος στόχος μας είναι η αυτοματοποίηση της κίνησης της τράπεζας εργασίας κατά τον άξονα X ώστε να επιτύχουμε επαναλαμβανόμενη μικροσυγκόλληση και των 32 λωρίδων του ανιχνευτή πυριτίου με τις αντίστοιχες γραμμές των ηλεκτρονικών προενίσχυσης (Σχήμα 3).




**Σχήμα 1: Φωτογραφία της συσκευής μικροσυγκόλλησης delvotec 5430**

Ας δούμε αναλυτικά με ποιόν τρόπο γίνεται η διαδικασία της μικροσυγκόλλησης. Το σύρμα Al φέρεται στη σφήνα της κεφαλής της συσκευής μικροσυγκόλλησης. Η κεφαλή, με την χρήση του χειριστηρίου που διαθέτει η συσκευή και με κίνηση της τράπεζας εργασίας στο επίπεδο X-Y, τοποθετείται ακριβώς πάνω από το πέλμα μικροσυγκόλλησης κάθε λωρίδας του ανιχνευτή και με τη χρήση υπερήχων συμπιέζεται και συγκολλείται με μηχανικό τρόπο στο σημείο αυτό. Στη συνέχεια η κεφαλή μετακινείται πάνω από το αντίστοιχο πέλμα μικροσυγκόλλησης των ηλεκτρονικών ανάγνωσης, το σύρμα Al συμπιέζεται και πάλι, οπότε επιτυγχάνεται η μικροσυγκόλληση μεταξύ των δύο τμημάτων (Σχήμα 2). Φυσικά η όλη διαδικασία απαιτεί ειδικό προγραμματισμό της συσκευής που γίνεται από έμπειρο χρήστη.




**Σχήμα 2: Διαδικασία μικροσυγκόλλησης**

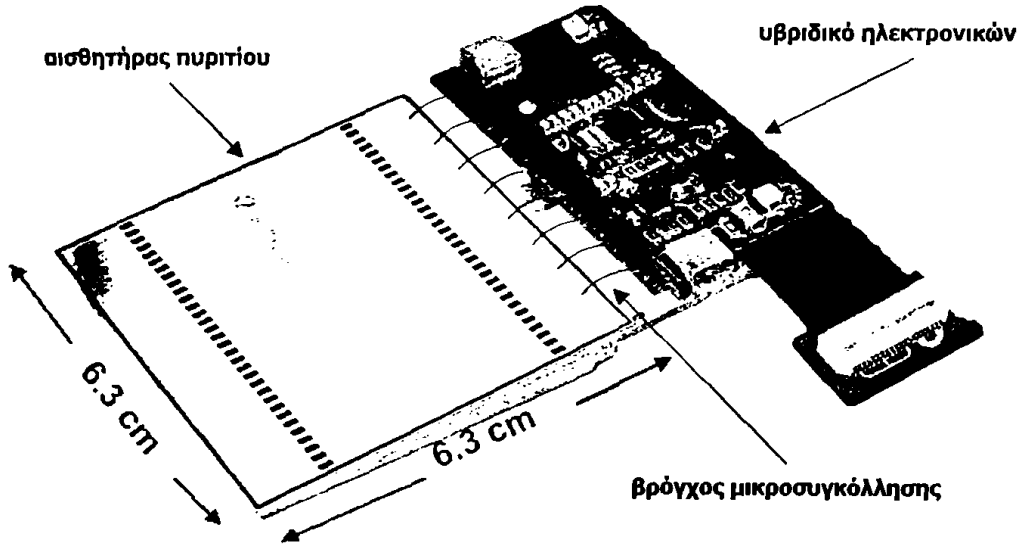
Το σύστημα της εφαρμογής μας αποτελείται από ένα βηματικό κινητήρα ο οποίος οδηγεί μια γραμμική τράπεζα ακριβείας με βήμα 1μm. Επειδή στη μικροσυγκόλληση η απόσταση μεταξύ των κέντρων των λωρίδων του ανιχνευτή είναι ακριβώς 1,9mm, ο κινητήρας θα πρέπει να κάνει 1900 βήματα, δεδομένου ότι για κάθε βήμα του κινητήρα η τράπεζα μετακινείται κατά 1μm.

Ο χρήστης επιλέγει εάν η κίνηση της τράπεζας θα ελέγχεται αυτόνομα μέσω του χειριστηρίου που διαθέτει το σύστημα, είτε από Η/Υ σε απόσταση, με κατάλληλη εφαρμογή που αναπτύχθηκε σε περιβάλλον γραφικού προγραμματισμού LabView. Μετά από 1900 βήματα του κινητήρα ή για μετατόπιση της τράπεζας κατά 1,9mm κλείνει αυτόματα ένας διακόπτης, εργασία που γινόταν πριν από το χειριστήριο της συσκευής μικροσυγκόλλησης, ώστε να ξεκινήσει η διαδικασία μικροσυγκόλλησης μεταξύ του πέλματος μικροσυγκόλλησης κάθε λωρίδας του ανιχνευτή πυριτίου και του κάθε καναλιού των ηλεκτρονικών ανάγνωσης. Πέρα από τις βασικές επιλογές και για μεγαλύτερη ευχρηστία, το σύστημα διαθέτει και επιπλέον λειτουργίες. Τόσο στην αυτόνομη όσο και στη λειτουργία μέσω του Η/Υ ο χρήστης έχει τη δυνατότητα επιλογής το σύστημα να κάνει ένα βήμα των 1,9mm κάθε φορά ή και τα 32 συνολικά, αφού 32 είναι και οι λωρίδες του αισθητήρα πυριτίου. Επιπλέον έχει τη δυνατότητα να εκτελέσει ένα ή περισσότερα δοκιμαστικά βήματα αρχικά και τα υπόλοιπα όλα μαζί, με επιλογή της αντίστοιχης λειτουργίας.

Η υλοποίηση του συστήματος οδήγησης του κινητήρα βασίζεται στον μικροελεγκτή AT89S8252 της εταιρείας ATMEL, ο οποίος ελέγχει τη λειτουργία του συστήματος έχοντας το ρόλο του master στην αυτόνομη λειτουργία και το ρόλο του

slave στη λειτουργία μέσω Η/Υ. Στη λειτουργία μέσω Η/Υ ο έλεγχος του συστήματος γίνεται με βάση τις εντολές που δέχεται ο μικροελεγκτής από την εφαρμογή που αναπτύχθηκε στο περιβάλλον προγραμματισμού LabView. Η επικοινωνία συστήματος – Η/Υ γίνεται μέσω της σειριακής θύρας RS-232 του υπολογιστή. Το κύκλωμα οδήγησης του κινητήρα αποτελείται από τα ολοκληρωμένα L297-L298 της εταιρείας SGT. Το ολοκληρωμένο L297 παράγει τα σήματα ελέγχου για τον κινητήρα, ενώ το L298 παρέχει σταθερό ρεύμα στα τυλίγματα του κινητήρα.

Η τράπεζα ακριβείας θα στερεωθεί στην οριζόντια επιφάνεια της συσκευής μικροσυγκόλλησης και θα αντικαταστήσει την τράπεζα εργασίας. Οι μικρομονάδες θα τοποθετούνται στην τράπεζα, θα στερεώνονται σε κατάλληλη θέση και θα γίνεται η μικροσυγκόλληση μεταξύ της πρώτης λωρίδας του αισθητήρα και του αντίστοιχου τμήματος των ηλεκτρονικών του ανιχνευτή. Στη συνέχεια η τράπεζα θα μετακινείται με ακρίβεια κατά 1,9mm και θα επαναλαμβάνεται η ίδια διαδικασία για όλες τις μικρολωρίδες του ανιχνευτή. Στο Σχήμα 3 που ακολουθεί φαίνεται μια μικρομονάδα του Preshower με τον αισθητήρα πυριτίου και το υβριδικό των ηλεκτρονικών προενίσχυσης. Στο ίδιο Σχήμα φαίνονται και μερικοί βρόγχοι μικροσυγκόλλησης.




**Σχήμα 3: Μικρολωριδιακός ανιχνευτής Si**

Το Κεφάλαιο 1 αυτής της εργασίας αποτελεί μια εισαγωγή στη θεωρία των βηματικών κινητήρων. Πιο αναλυτικά γίνεται αναφορά στις βασικές αρχές λειτουργίας των βηματικών κινητήρων, στα είδη αυτών και στα βασικά χαρακτηριστικά τους.

Στο δεύτερο κεφάλαιο περιγράφεται η ανάπτυξη και κατασκευή της μονάδας οδήγησης του βηματικού κινητήρα, καθώς επίσης και η ολοκλήρωση του συστήματος.

Στο επόμενο κεφάλαιο (κεφάλαιο 3) περιγράφεται το λογισμικό που αναπτύχθηκε για τον προγραμματισμό του μικροελεγκτή για την εκτέλεση των

επιμέρους λειτουργιών που απαιτούνται για την μετατόπιση της τράπεζας τόσο για την αυτόνομη λειτουργία όσο και για τη λειτουργία μέσω Η/Υ.

Το Κεφάλαιο 4 είναι αφιερωμένο στην περιγραφή της εφαρμογής που αναπτύχθηκε για τον έλεγχο του συστήματος από τον Η/Υ στο περιβάλλον προγραμματισμού LabView.

Τέλος στο Κεφάλαιο 5 αναφέρονται λεπτομέρειες για τον τρόπο κατασκευής του συστήματος καθώς επίσης και τα αποτελέσματα των δοκιμών που έγιναν μετά την ολοκλήρωσή του.

Στο τέλος της εργασίας παρατίθεται μια σειρά παραρτημάτων που έχουν ως στόχο να βοηθήσουν τον αναγνώστη στην κατανόηση του τρόπου λειτουργίας ορισμένων εξαρτημάτων που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος της εφαρμογής. Έτσι λοιπόν, στο πρώτο παράρτημα γίνεται μια πιο λεπτομερής ανάλυση των βηματικών κινητήρων. Στο δεύτερο παράρτημα αναφέρονται πληροφορίες για την αρχιτεκτονική του μικροελεγκτή AT89S8252. Στο τρίτο παράρτημα γίνεται μια εισαγωγή για τη μέθοδο προγραμματισμού της οθόνης LCD. Επίσης παρατίθενται ολοκληρωμένες υπορουτίνες, οι οποίες μπορούν να χρησιμοποιηθούν σε κάθε εφαρμογή στην οποία γίνεται χρήση οθόνης LCD. Το Παράρτημα Δ είναι ο κώδικας σε γλώσσα Assembly που φορτώσαμε στη μνήμη του μικροελεγκτή για την διεκπεραίωση της εφαρμογής μας, ενώ στο τελευταίο παράρτημα δίνονται τα τυπωμένα κυκλώματα των πλακετών που κατασκευάσαμε καθώς επίσης και φωτογραφίες του συστήματός μας.

# Κεφάλαιο 1ο
# Βηματικοί Κινητήρες

## 1.1 Ο κινητήρας της εφαρμογής

Ο τύπος του κινητήρα που χρησιμοποιείται στην εφαρμογή είναι ένας διπολικός, διφασικός κινητήρας, ιδιότητες που θα μελετήσουμε στις επόμενες παραγράφους του κεφαλαίου αυτού. Ο συγκεκριμένος τύπος κινητήρα υπερτερεί έναντι των άλλων κινητήρων της αγοράς στο σχετικά μικρό του βάρος και τις μικρές εξωτερικές διαστάσεις του.

Η γωνία βήματος του συγκεκριμένου κινητήρα είναι $\theta_s$=0,9° και ο αριθμός των βημάτων για μια πλήρη περιστροφή S= 400. Η μέγιστη τάση λειτουργίας του είναι 6V, ενώ το ρεύμα που διαρρέει τα τυλίγματά του, της τάξης των 0,8A. Ο κινητήρας έχει συνδεθεί με τον κοχλία οδήγησης τράπεζας ακριβείας τύπου LM60 τα χαρακτηριστικά της οποίας φαίνονται στον Πίνακα 1.1 και φωτογραφία αυτής στο Σχήμα 1.1. Ο κοχλίας οδήγησης της τράπεζας έχει βήμα οδήγησης 0,4mm. Αυτό μεταφράζεται σε γραμμική μετατόπιση του δρομέα της τράπεζας κατά 0,4mm για μια πλήρη περιστροφή του κοχλία. Επομένως για γωνιακή μετατόπιση 0,9° προκύπτει γραμμική μετατόπισης της τράπεζας κατά 1μm. Δηλαδή για κάθε βήμα του κινητήρα, όταν αυτός οδηγείται με οδήγηση πλήρους βήματος (full step mode), έχουμε γραμμική μετατόπιση 1μm, ενώ με οδήγηση μισού βήματος ( half step mode) πετυχαίνουμε γραμμική μετατόπιση 0,5μm. Το μήκος της διαδρομής της τράπεζας είναι συνολικά 75 mm. Εξαιτίας των δύο μηχανικών διακοπτών που υπάρχουν στα άκρα της τράπεζας για λόγους προστασίας αυτής, το μήκος της διαδρομής μειώνεται στα 68 mm.

**Technical Data**

| | |
|---|---|
| Travel: | 75mm |
| Mass: | 1,05kgr |

**Maximum traverse speed (no load)**

| | |
|---|---|
| 2-phase stepping motor | 5mm/s |

**Load characteristics**

| | |
|---|---|
| Mx (Nm) | 18 |
| My (Nm) | 8 |
| Mz (Nm) | 10 |
| Fx (N) | 70 |
| Fy (N) | 76 |
| Fz (N) | 130 |

**Resolution (calculated)**

| | |
|---|---|
| full-step | 1μm |
| half-step | 0.5μm |

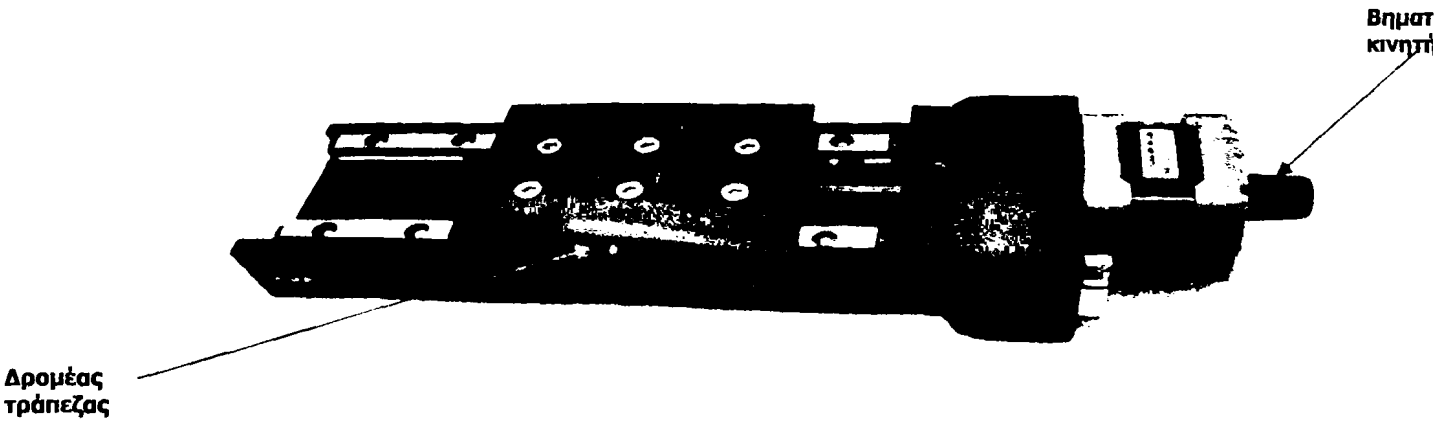
**Electrical supply**

| | |
|---|---|
| Type of connection | 9-pin Sub D-connector (male) with 1.5m connecting cable |

**Accuracy of guides (no load)**

| | |
|---|---|
| Yaw angle (arc seconds) | 50 |
| Pitch angle (arc seconds) | 50 |
| Vertical deviation (μm) | 10 |
| Lateral deviation (μm) | 10 |

**Πίνακας 1.1: Χαρακτηριστικά λειτουργίας της τράπεζας της εφαρμογής**




**Σχήμα 1.1: Η τράπεζα ακριβείας της εφαρμογής**

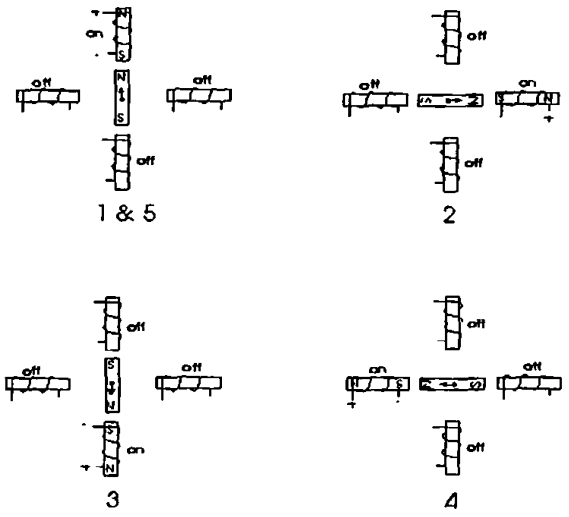
## 1.2 Ορισμός βηματικού κινητήρα

Βηματικός κινητήρας είναι ένας ηλεκτρικός κινητήρας που μετατρέπει ψηφιακά, ηλεκτρικά σήματα σε μηχανική κίνηση. Αυτό σημαίνει ότι ένα ψηφιακό σήμα χρησιμοποιείται για να οδηγήσει τον κινητήρα και κάθε φορά που αυτός λαμβάνει έναν ψηφιακό παλμό, περιστρέφεται κατά ένα καθορισμένο αριθμό μοιρών. Έτσι κάθε βήμα περιστροφής είναι η απόκρισή του σε έναν παλμό εισόδου.

Οι βηματικοί κινητήρες δεν ανταποκρίνονται απλά σε έναν παλμό ρολογιού. Έχουν στο εσωτερικό τους έναν αριθμό τυλιγμάτων, τα οποία πρέπει να ενεργοποιηθούν με τη σωστή σειρά, προτού ο άξονας του κινητήρα περιστραφεί. Αντιστρέφοντας τη σειρά ενεργοποίησης των τυλιγμάτων ο κινητήρας θα περιστραφεί αντίθετα. Επιπλέον εάν τα σήματα ελέγχου δεν σταλθούν με τη σωστή σειρά, ο κινητήρας δεν θα λειτουργήσει σωστά, με αποτέλεσμα να υπάρχει κίνδυνος καταστροφής αυτού.

## 1.3 Θεωρία λειτουργίας βηματικού κινητήρα

Οι βηματικοί κινητήρες αποτελούνται από έναν άξονα περιστροφής κατασκευασμένο από μόνιμο μαγνήτη, ο οποίος ονομάζεται ρότορας, ενώ το ακίνητο τμήμα που περιβάλλει τον κινητήρα είναι ηλεκτρομαγνήτες (ή τυλίγματα) και ονομάζεται στάτορας. Η βασική λειτουργία του κινητήρα έγκειται στο να περιστραφεί ο ρότορας κατά έναν ακριβή αριθμό μοιρών κάθε φορά που ένας παλμός στέλνεται σε αυτόν. Το Σχήμα 1.3.1 απεικονίζει μια πλήρη στροφή ενός απλού βηματικού κινητήρα.

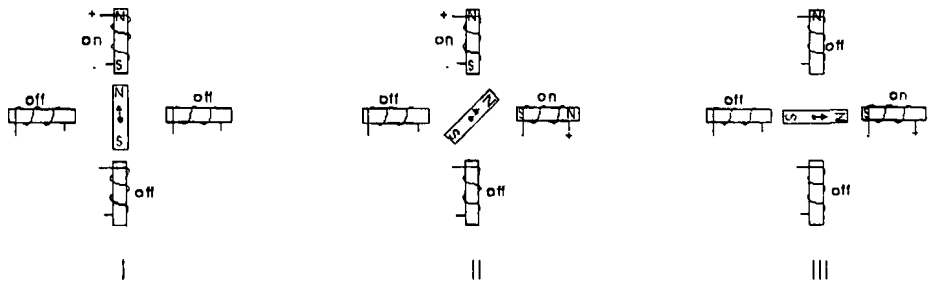



**Σχήμα 1.3.1: Μια πλήρης περιστροφή ενός απλού βηματικού κινητήρα**

Στη Θέση 1 βλέπουμε ότι ο ρότορας ξεκινά την κίνησή του από τον επάνω ηλεκτρομαγνήτη, ο οποίος διαρρέεται από ρεύμα. Για να κινηθεί ο ρότορας κατά τη

φορά κίνησης των δεικτών του ρολογιού, ο επάνω ηλεκτρομαγνήτης απενεργοποιείται (δεν διαρρέεται από ρεύμα) και ενεργοποιείται ο δεξιός ηλεκτρομαγνήτης προκαλώντας περιστροφή του ρότορα κατά 90° δεξιόστροφα, έτσι ώστε να ευθυγραμμιστεί με αυτόν (Θέση 2). Η διαδικασία αυτή επαναλαμβάνεται με τον ίδιο τρόπο για τον κάτω και αριστερό ηλεκτρομαγνήτη (Θέσεις 3 και 4 αντίστοιχα) έτσι ώστε να επιστρέψει ο ρότορας στην αρχική του θέση (Θέση 5).

Στο προηγούμενο παράδειγμα χρησιμοποιήσαμε έναν κινητήρα ανάλυσης 90°. Στην πράξη, όμως, ο κινητήρας αυτός δεν είναι αρκετά εύχρηστος. Η μέση ανάλυση των κινητήρων, δηλαδή ο αριθμός των μοιρών περιστροφής ανά παλμό, είναι αρκετά υψηλότερη στην πλειοψηφία των κινητήρων. Για παράδειγμα ένας κινητήρας με ανάλυση 5°, θα περιέστρεφε τον ρότορα 5° ανά παλμό, δηλαδή θα απαιτούσε 72 παλμούς συνολικά για να ολοκληρώσει μια πλήρη περιστροφή. Είναι εφικτό να διπλασιάσουμε την ανάλυση ενός κινητήρα χρησιμοποιώντας τη μέθοδο 'half-stepping'. Αντί να ενεργοποιούμε έναν ηλεκτρομαγνήτη κάθε φορά, με τη μέθοδο 'half-stepping' ενεργοποιούμε 2 ηλεκτρομαγνήτες ταυτόχρονα, προκαλώντας ισοδύναμη έλξη του ρότορα από τους ενεργοποιημένους ηλεκτρομαγνήτες, διπλασιάζοντας έτσι την ανάλυση. Όπως μπορούμε να δούμε στο Σχήμα 1.3.2, στην πρώτη περίπτωση μόνο ο επάνω ηλεκτρομαγνήτης διαρρέεται από ρεύμα και ο ρότορας είναι απολύτως ευθυγραμμισμένος με αυτόν. Στη δεύτερη περίπτωση ο επάνω και ο δεξιός ηλεκτρομαγνήτης είναι ενεργοποιημένοι, οπότε ο ρότορας, επειδή έλκεται και από τους δυο, ευθυγραμμίζεται ακριβώς στο μέσο αυτών. Στην τρίτη περίπτωση παραμένει ενεργοποιημένος μόνο ο δεξιός ηλεκτρομαγνήτης και ο ρότορας ευθυγραμμίζεται με αυτόν. Η παραπάνω διαδικασία επαναλαμβάνεται για μία ολόκληρη περιστροφή.




**Σχήμα 1.3.2: Μέθοδος 'half-stepping'**

Σε αντίθεση με τον κινητήρα του προηγούμενου παραδείγματος στον οποίο ο ρότορας περιστρέφεται 90° ανά βήμα, στους 'πραγματικούς' βηματικούς κινητήρες ο ρότορας και ο στάτορας αποτελούνται από περισσότερους πόλους και τυλίγματα αντίστοιχα. Αν και αυτό σημαίνει πιο πολύπλοκη διαδικασία οδήγησης, η λειτουργία τους είναι πανομοιότυπη με αυτή του κινητήρα των 90° που αναλύσαμε παραπάνω.

## 1.4 Χαρακτηριστικά λειτουργίας βηματικού κινητήρα

### 1.4.1 Βηματική γωνία

Όπως έχουμε ήδη αναφέρει ένας βηματικός κινητήρας περιστρέφεται κατά μια καθορισμένη γωνία για κάθε παλμό, η οποία ονομάζεται βηματική γωνία $\theta_s$. Μειώνοντας τη βηματική γωνία αυξάνει η ακρίβεια θέσης του ρότορα. Το παραπάνω έχει ως άμεση συνέπεια να παρουσιάζει ενδιαφέρον ο αριθμός των βημάτων για μια πλήρη περιστροφή του κινητήρα, ο οποίος συμβολίζεται με S. Η σχέση του με τη βηματική γωνία $\theta_s$ είναι η εξής:

$$\theta_s = 360/S$$

Η σχέση του αριθμού των βημάτων ανά περιστροφή (S) με τον αριθμό των πόλων του ρότορα ($N_r$) και του αριθμού των φάσεων (m) δίνεται από τις παρακάτω σχέσεις:

Για κινητήρες VR (Variable Reluctance)*

$$S = m N_r$$

Για κινητήρες PM (Permanent Magnet )** και υβριδικούς***

$$S = 2 m N_r$$

### 1.4.2 Τάση λειτουργίας

Οι βηματικοί κινητήρες έχουν μια τάση στην οποία λειτουργούν με τη μέγιστη απόδοση. Αυτή είναι συνήθως τυπωμένη πάνω στη συσκευή του κινητήρα, είτε αναγράφεται στο φυλλάδιο λειτουργίας του. Η λειτουργία του κινητήρα σε τάση μεγαλύτερη από την αναγραφόμενη είναι κάποιες φορές επιθυμητή για να διατηρήσουμε τη ροπή σε επιθυμητά επίπεδα, όμως, η κατάσταση αυτή παράγει μεγάλα ποσά θερμότητας στον κινητήρα, με αποτέλεσμα τη μείωση της διάρκειας ζωής του.

### 1.4.3 Αντίσταση

Η αντίσταση ανά τύλιγμα είναι ένα ακόμη χαρακτηριστικό του βηματικού κινητήρα. Καθορίζει τη ροή του ρεύματος σε αυτόν (Νόμος του Ohm V = I R, όπου V η τάση λειτουργίας του κινητήρα) και επηρεάζει τη μέγιστη ταχύτητα λειτουργίας του.

*, **, ***: Η ανάλυση των τύπων των βηματικών κινητήρων ακολουθεί σε επόμενη παράγραφο

### 1.4.4 Ροπή

Όπως αναφέραμε σε προηγούμενη παράγραφο, όταν τα τυλίγματα του στάτορα διεγείρονται, μετατρέπονται σε μαγνήτη με αποτέλεσμα να αναπτύσσεται μια μαγνητική ροή στους πόλους του στάτορα και μια ροπή (ροπή στρέψης) που τείνει να περιστρέψει τον ρότορα κατά τη διεύθυνση του μαγνητικού πεδίου. Η ροπή στρέψης είναι ανάλογη του αριθμού των τυλιγμάτων, των χαρακτηριστικών αυτών και του ρεύματος που διαρρέει τα τυλίγματα. Η σχέση που δίνει την ροπή στρέψης (Η) είναι η εξής :

$$H = (N*I)/L$$

όπου Ν ο αριθμός των τυλιγμάτων, Ι η ένταση του ρεύματος και L το μήκος της διαδρομής της μαγνητικής ροής η οποία εξαρτάται από τα χαρακτηριστικά των τυλιγμάτων.

Οι βηματικοί κινητήρες σχεδιάζονται ώστε να παρέχουν υψηλή ροπή στρέψης. Αυτό επιτρέπει στον κινητήρα να ξεκινά και να σταματά γρήγορα και να επιδεικνύει υψηλή ροπή επαναφοράς όταν τείνει να προκύψει μετακίνηση από τη θέση ισορροπίας, εξαιτίας της ροπής που οφείλεται στο φόρτο. Οι ορισμοί 'ροπή συγκράτησης ' και 'ροπή επαναφοράς ' που χρησιμοποιούνται συχνά σε σχέση με τη ροπή στρέψης έχουν ως εξής :

1. ***Ροπή συγκράτησης*** ορίζεται ως η μέγιστη ροπή στρέψης που είναι δυνατόν να εφαρμοστεί στον άξονα ενός διεγερμένου κινητήρα χωρίς να προκαλέσει συνεχή περιστροφή.
2. ***Ροπή επαναφοράς*** ορίζεται ως η μέγιστη ροπή στρέψης που είναι δυνατόν να εφαρμοστεί στον άξονα ενός μη διεγερμένου κινητήρα χωρίς να προκαλέσει συνεχή περιστροφή (αφορά μόνο κινητήρες με μόνιμο μαγνήτη).

### 1.4.5 Σφάλμα θέσης

Η ακρίβεια στον προσδιορισμό της θέσης του ρότορα είναι ένα από τα κύρια χαρακτηριστικά που καθορίζουν την ποιότητα ενός βηματικού κινητήρα. Η ακρίβεια ενός κινητήρα χωρίς φορτίο είναι συνάρτηση μόνο της ακρίβειας κατασκευής των διαστάσεων του ρότορα και του στάτορα. Οι έννοιες που σχετίζονται με τη θέση που σταματά ένας κινητήρας σε κίνηση έχουν ως εξής :

1. ***Θέση ακινησίας ή ισορροπίας***: Ορίζεται ως η θέση στην οποία ένας κινητήρας χωρίς φορτίο σε διέγερση σταματά.
2. ***Θέση ηρεμίας – χαλάρωσης***: Ορίζεται ως η θέση στην οποία ένας κινητήρας, που ο ρότοράς του είναι μόνιμος μαγνήτης, φτάνει σε ακινησία χωρίς διέγερση και χωρίς φορτίο.

Οι έννοιες που σχετίζονται με το σφάλμα θέσης ενός κινητήρα είναι οι παρακάτω:

1. ***Σφάλμα θέσης βήματος***: Ορίζεται ως το μέγιστο θετικό ή αρνητικό στατικό γωνιακό σφάλμα θέσης που μπορεί να προκύψει όταν ο κινητήρας έχει περιστραφεί κατά ένα βήμα από την προηγούμενη θέση ισορροπίας. Το Σφάλμα θέσης βήματος υπολογίζεται από την παρακάτω σχέση:

$$\text{Σφάλμα θέσης βήματος} = \theta_{s,\text{πειραματικό}} - \theta_{s,\text{ονομαστικό}}$$

2. ***Ακρίβεια θέσης*** : Ορίζεται ως το μέγιστο γωνιακό σφάλμα θέσης μιας θέσης ισορροπίας ως προς την ονομαστική γωνία βήματος που μπορεί να προκύψει κατά τη διάρκεια μιας πλήρους περιστροφής του ρότορα όταν κινηθεί από μια θέση αναφοράς σε θέση ισορροπίας.

3. ***Ακρίβεια θέσης υστέρησης*** : Ορίζεται ως η τιμή του σφάλματος που παίρνουμε από τη μέτρηση της ακρίβειας θέσης και ως προς τις δυο κατευθύνσεις.

### 1.4.6 Δυναμική συμπεριφορά- ταχύτητα περιστροφής

Ένα σημαντικό χαρακτηριστικό που εξετάζουμε στους βηματικούς κινητήρες είναι το πόσο γρήγορα αποκρίνονται σε ένα παλμό ή σε μια παλμοσειρά εισόδου εκτελώντας τη ζητούμενη κίνηση. Απαιτούμε από αυτούς όχι μόνο γρήγορη εκκίνηση, αλλά και ακριαία στάση. Αν η παλμοσειρά εισόδου διακοπεί για παράδειγμα ενόσω ο κινητήρας περιστρέφεται, ο κινητήρας θα σταματήσει στο σημείο που καθορίζει ο τελευταίος παλμός. Αυτά τα χαρακτηριστικά των βηματικών κινητήρων οφείλονται στην υψηλή αναλογία μεταξύ ροπής και αδράνειας του ρότορα, σε αντίθεση με τους DC ηλεκτροκινητήρες.

Η ταχύτητα με την οποία περιστρέφεται ένας βηματικός κινητήρας δίνεται συνήθως από τον αριθμό των βημάτων ανά δευτερόλεπτό. Επειδή, όμως, ο αριθμός των βημάτων ισοδυναμεί με τον αριθμό των παλμών, η ταχύτητα είναι δυνατόν να εκφραστεί με όρους συχνότητας. Επομένως ο όρος που χρησιμοποιείται συνήθως για να εκφράσει την ταχύτητα περιστροφής είναι **βηματική συχνότητα f.**

Σε καμία περίπτωση δεν πρέπει να συγχέουμε τη βηματική συχνότητα με τη **συχνότητα περιστροφής** του κινητήρα. Η συχνότητα περιστροφής μετράται σε στροφές ανά λεπτό (rev/min) και η σχέση που τη συνδέει με τη βηματική συχνότητα είναι η εξής :

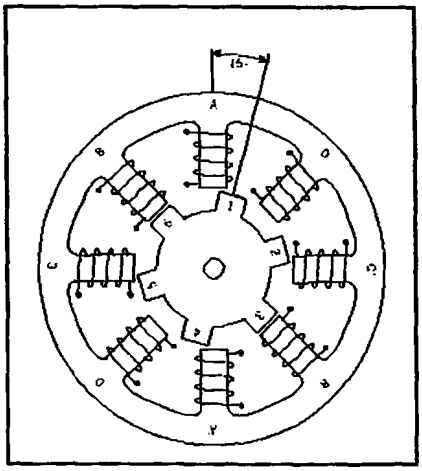
$$n = 60f/S$$

όπου n = συχνότητα περιστροφής (rev/min)
f = βηματική συχνότητα (Hz)
S = αριθμός βημάτων ανά πλήρη περιστροφή.

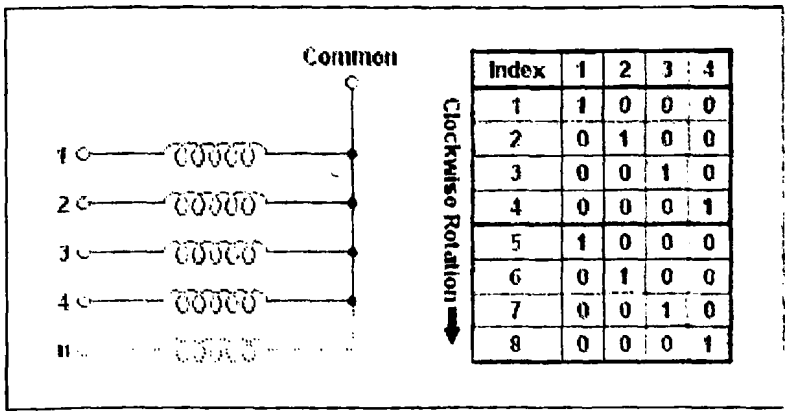
## 1.5 Ταξινόμηση βηματικών κινητήρων

### 1.5.1 Βηματικοί κινητήρες VR (variable reluctance)

Πρόκειται για το πιο κατανοητό τύπο κινητήρα τόσο από πλευράς κατασκευής όσο και λειτουργίας, γι' αυτό και έναν κινητήρα τέτοιου τύπου μελετήσαμε στο εισαγωγικό μας παράδειγμα. Ρότορας και στάτορας είναι κατασκευασμένοι από ελάσματα χάλυβα, αν και ο ρότορας δεν αποτελεί μόνιμο μαγνήτη. Το Σχήμα 1.5.1.1 απεικονίζει την τομή ενός τετραφασικού κινητήρα VR. Τα τυλίγματα του στάτορα συνδέονται όπως φαίνεται στο Σχήμα 1.5.1.2, με το ένα άκρο κοινό για όλα τα τυλίγματα. Για να περιστρέφουμε τον κινητήρα συνεχώς, αρκεί να διεγείρουμε τα τυλίγματα διαδοχικά το ένα μετά το άλλο, με τη σειρά που μας δείχνει και ο πίνακας του Σχήματος 1.5.1.2.



**Σχήμα 1.5.1.1: Τομή τετραφασικού κινητήρα VR**

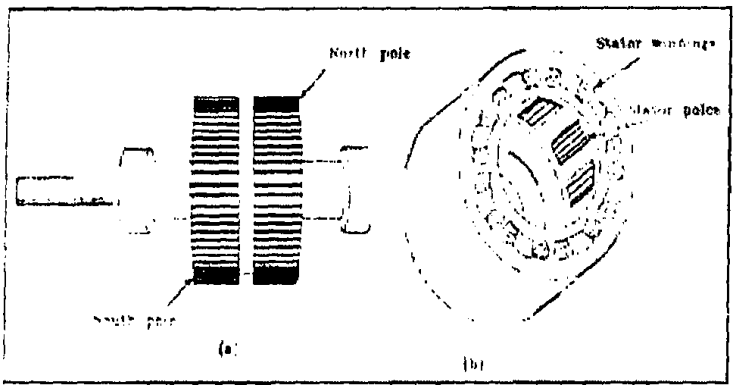



| Index | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 |
| 8 | 0 | 0 | 0 | 1 |

**Σχήμα 1.5.1.2: Συνδεσμολογία τυλιγμάτων ενός VR κινητήρα (αριστερά) και πίνακας ενεργοποίησης τυλιγμάτων (δεξιά)**

## 1.5.2 Βηματικοί κινητήρες PM ( permanent magnet )

Η ονομασία αυτού του κινητήρα οφείλεται στο γεγονός ότι ο ρότοράς του είναι μόνιμος μαγνήτης. Η αρχή λειτουργίας του οφείλεται στην αλληλεπίδραση του ρότορα με το ηλεκτρομαγνητικό πεδίο. Το Σχήμα 1.4.2.1 δείχνει τον ρότορα και τον στάτορα ενός PM κινητήρα. Οι μικροί πόλοι στην επιφάνεια του ρότορα και οι επιφάνειες των πόλων του στάτορα, αντισταθμίζονται έτσι ώστε μόνο ένας μικρός αριθμός πόλων του ρότορα να ευθυγραμμίζεται με τον διεγερμένο πόλο του στάτορα. Ο αριθμός των πόλων του ρότορα και του στάτορα καθορίζουν τη βηματική γωνιά του κινητήρα. Όσο πιο μεγάλος είναι ο αριθμός των πόλων τόσο πιο μικρή είναι η βηματική γωνία.

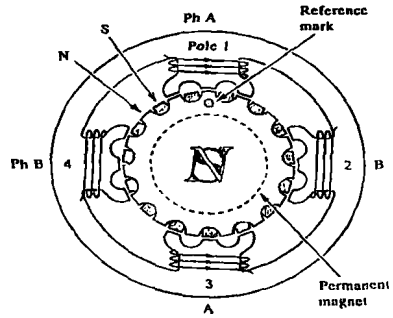



**Σχήμα 1.5.2.1: Μέρη ενός PM βηματικού κινητήρα: (a) Ρότορας, (b) Στάτορας**

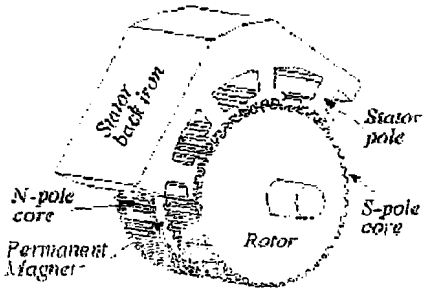
Όταν εφαρμόσουμε τάση DC σε ένα από τα τυλίγματα του στάτορα, ο ρότορας θα υπερνικήσει την προϋπάρχουσα ροπή και τείνει να ευθυγραμμιστεί με το μαγνητικό πεδίο του στάτορα. Όπως αναφέραμε και σε προηγούμενη παράγραφο, η ροπή συγκράτησης καθορίζεται από τη ροπή που απαιτείται για να περιστραφεί ο ρότορας κατά 1 πλήρες βήμα εφόσον ο στάτορας είναι ενεργοποιημένος. Σημαντικό χαρακτηριστικό ενός PM κινητήρα είναι ότι μπορεί να διατηρήσει την ροπή συγκράτησης έπ' αόριστον, όταν ο ρότορας σταματήσει να περιστρέφεται. Όταν δεν παρέχεται τάση στα τυλίγματα του στάτορα μια μικρή μαγνητική δύναμη αναπτύσσεται μεταξύ του μόνιμου μαγνήτη και του στάτορα που εμποδίζει το ρότορα να κινηθεί. Αυτή η μαγνητική δύναμη ονομάζεται 'residual' (παραμένουσα) ροπή και μπορεί να παρατηρηθεί αν περιστρέψουμε τον κινητήρα με το χέρι μας, ενώ υπολογίζεται ότι είναι περίπου ίση με το 1/10 της ροπής συγκράτησης .

### 1.5.3 Υβριδικοί βηματικοί κινητήρες

Ο υβριδικός κινητήρας είναι ο πιο εύχρηστος κινητήρας που συναντούμε αφού συνδυάζει τα καλύτερα χαρακτηριστικά των PM και VR κινητήρων, όπως της υψηλής ροπής, της μεγάλης ταχύτητας και της μικρής βηματικής γωνίας (0,9° έως 3,6°). Η ονομασία του προκύπτει από το γεγονός ότι η λειτουργία του βασίζεται στο συνδυασμό των αρχών λειτουργίας των PM και VR κινητήρων. Το Σχήμα 1.5.3.1 απεικονίζει την κάθετη τομή ενός υβριδικού, διφασικού κινητήρα, ενώ τμήμα ενός πενταφασικού κινητήρα σε τρισδιάστατη μορφή φαίνεται στο Σχήμα 1.5.3.2




**Σχήμα 1.5.3.1 : Τομή υβριδικού κινητήρα**




**Σχήμα 1.5.3.2 : Τμήμα υβριδικού κινητήρα σε τρισδιάστατη μορφή**

Η μορφή του στάτορα είναι παρόμοια με αυτή που συναντάμε στους VR κινητήρες με τη διαφορά ότι οι πόλοι του σχηματίζουν οδοντωτές προεξοχές. Ο ρότορας, όπως και στους PM κινητήρες, είναι μόνιμος μαγνήτης. Ένα σημαντικό χαρακτηριστικό του ρότορα είναι ότι στο εσωτερικό του υπάρχει ένας κυλινδρικός μαγνήτης μαγνητισμένος κατά τον άξονα συμμετρίας του ώστε να παραχθεί ένα ομογενές μαγνητικό πεδίο. Κάθε πόλος καλύπτεται από οδοντωτά καλύμματα μαλακού χάλυβα. Οι οδοντωτές προεξοχές στις περιοχές του κάθε πόλου δεν είναι ευθυγραμμισμένες, αλλά διαφέρουν μεταξύ τους κατά μισό οδόντα.

Για να κατανοήσουμε καλύτερα την λειτουργία ενός υβριδικού κινητήρα ας επιστρέψουμε στον κινητήρα του Σχήματος 1.5.3.1. Τα τυλίγματα στον πόλο 1 και 3 συνδέονται αποτελώντας τη φάση Α και βρίσκονται σε διεγερμένη κατάσταση. Οι πόλοι 2 και 4 αποτελούν αντίστοιχα τη φάση Β. Αν παρατηρήσουμε τον ρότορα θα δούμε ότι αποτελείται από εναλλασσόμενους βόρειους και νότιους πόλους με τους βόρειους εγγύτερα προς εμάς με τον τρόπο που κοιτάμε το σχήμα από τους νότιους.

Όταν οι πόλοι της φάσης Α είναι διεγερμένοι, οι οδόντες του πόλου 1 έλκουν μερικούς από τους βόρειους πόλους του ρότορα, ενώ εκείνοι του πόλου 3 ευθυγραμμίζονται με τους νότιους. Όταν το ρεύμα διοχετευτεί στη φάση Β οι πόλοι 2 και 4 διεγείρονται και ο ρότορας θα κινηθεί κατά ένα τέταρτο του οδόντα ώστε να υπάρξει ευθυγράμμιση στους πόλους 2 και 4. Στη συνέχεια ρεύμα διοχετεύεται και πάλι στη φάση Α με αντίθετη, όμως, πολικότητα. Ο ρότορας θα κινηθεί κατά ένα τέταρτο ακόμη για να φτάσουμε στην κατάσταση όπου οι πόλοι 1 και 3 του στάτορα θα ευθυγραμμιστούν με τους νότιους και βόρειους πόλους του ρότορα αντίστοιχα. Τέλος όταν διοχετευθεί και πάλι ρεύμα στη φάση Β, με αντίθετη πολικότητα, θα έχουμε μια ακόμη μετατόπιση του ρότορα κατά ένα τέταρτο του οδόντα.

## 1.6 Τρόποι διέγερσης

Όπως έχουμε ήδη αναφέρει οι βηματικοί κινητήρες περιστρέφονται με την διαδοχική διέγερση ενός ή περισσοτέρων τυλιγμάτων. Υπάρχουν τρεις τρόποι διέγερσης του κινητήρα, οι οποίοι εξαρτώνται από τον τρόπο οδήγησής του και είναι οι παρακάτω:

1. **Full step:** Αυτή η μέθοδος διέγερσης διαχωρίζεται ως εξής:

**1.1 Wave Drive (*one phase on*):** Στην περίπτωση αυτή ο κινητήρας λειτουργεί με ένα μόνο τύλιγμα διεγερμένο κάθε φορά. Μειονέκτημα αυτού του τρόπου διέγερσης είναι η ανάπτυξη μικρής ροπής και ταχύτητας περιστροφής ενώ στα πλεονεκτήματά του συμπεριλαμβάνεται η μικρή κατανάλωση ισχύος που απαιτεί το κύκλωμα οδήγησης. Στον πίνακα 1.7.1 φαίνονται τα διαδοχικά βήματα της μονοφασικής διέγερσης για ένα διφασικό κινητήρα.

**1.2 Normal Drive (*two phase on*):** Με αυτή τη μέθοδο δύο τυλίγματα (2 φάσεις) ενεργοποιούνται ταυτόχρονα, οπότε ο ρότορας μετακινείται σε μία ενδιάμεση θέση αφού έλκεται εξίσου από δύο πόλους του στάτορα με αποτέλεσμα να μην υπάρχει ευθυγράμμιση όπως συμβαίνει στην περίπτωση της μονοφασικής διέγερσης. Πλεονέκτημα αυτής της μεθόδου είναι η ανάπτυξη μεγάλης ροπής κατά τη διάρκεια κίνησης του κινητήρα καθώς επίσης και ροπής συγκράτησης όταν ο κινητήρας είναι σταματημένος. Ο παραπάνω τρόπος διέγερσης μπορεί να επιτευχθεί εάν αντιστρέψουμε τη φορά του ρεύματος στο διεγερμένο ζεύγος τυλιγμάτων. Ο πίνακας 1.7.2 δείχνει τα διεγερμένα ζεύγη ανά παλμό.

2. **Οδήγηση μισού βήματος (*half step mode*):** Πρόκειται ουσιαστικά για ένα συνδυασμό των δύο παραπάνω μεθόδων αφού επιτυγχάνεται με την ενεργοποίηση αρχικά ενός τυλίγματος, στη συνέχεια δύο, έπειτά πάλι ένα κ.ο.κ.. Η μέθοδος αυτή διπλασιάζει τον αριθμό των βημάτων που απαιτούνται για μια πλήρη περιστροφή, μειώνει δηλαδή τη βηματική γωνία στο μισό. Επειδή, όμως, ανά δεύτερο παλμό ενεργοποιείται ένα μόνο τύλιγμα, έχουμε ανάπτυξη μικρότερης ροπής απ' ότι στη μέθοδο Full step (διφασική).

3. **Microstepping:** Είναι μια καινούργια μέθοδος διέγερσης η οποία στηρίζεται στο γεγονός ότι κάθε φάση διαρρέεται από διαφορετικό ρεύμα κάθε χρονική στιγμή. Το Σχήμα 1.7.2 δείχνει την κυματοσυνάρτηση του ρεύματος για κάθε φάση. Παρατηρούμε ότι οι κυματομορφές έχουν ημιτονοειδή μορφή με διαφορά φάσης 90° η μία από την άλλη. Με αυτό τον τρόπο διέγερσης η βηματική γωνία διαιρείται σε άλλες μικρότερες γωνιές με αποτέλεσμα να έχουμε μεγαλύτερη ακρίβεια θέσης και μεγαλύτερη ανάλυση του κινητήρα.

Clockwise Rotation

| Index | 1a | 1b | 2a | 2b |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 | 0 |
| 7 | 0 | 0 | 1 | 0 |
| 8 | 0 | 0 | 0 | 1 |

**Πίνακας 1.7.1: Wave Drive Sequence**

Clockwise Rotation

| Index | 1a | 1b | 2a | 2b |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 1 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 1 | 0 |
| 4 | 0 | 0 | 1 | 1 |
| 5 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | 0 | 0 |
| 7 | 0 | 1 | 1 | 0 |
| 8 | 0 | 0 | 1 | 1 |

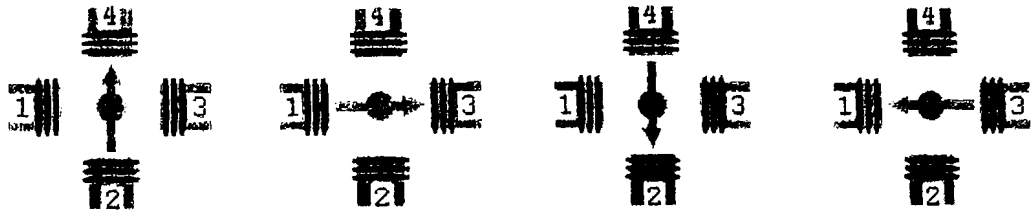
**Πίνακας 1.7.2: Normal Drive (two phase on) Sequence**

Clockwise Rotation

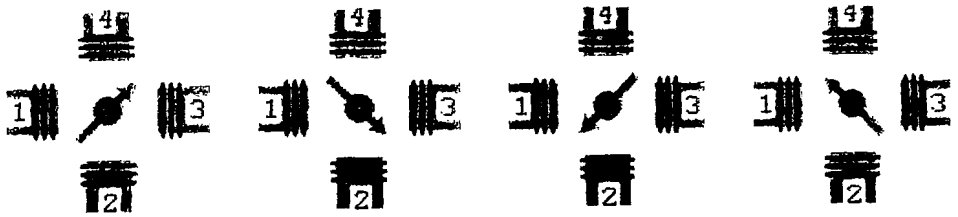
| Index | 1a | 1b | 2a | 2b |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 1 |
| 9 | 1 | 0 | 0 | 0 |
| 10 | 1 | 1 | 0 | 0 |
| 11 | 0 | 1 | 0 | 0 |
| 12 | 0 | 1 | 1 | 0 |
| 13 | 0 | 0 | 1 | 0 |
| 14 | 0 | 0 | 1 | 1 |
| 15 | 0 | 0 | 0 | 1 |
| 16 | 1 | 0 | 0 | 1 |

**Πίνακας 1.7.3: Half Step Sequence**

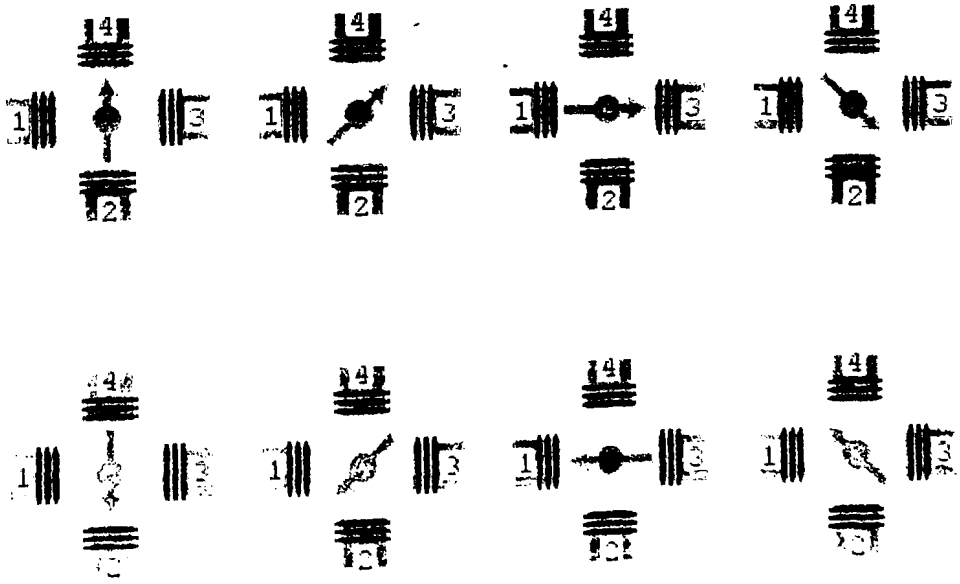
Το Σχήμα 1.7.1 που ακολουθεί είναι μια απεικόνιση των τρόπων διέγερσης ενός διφασικού κινητήρα. Τα τυλίγματα 4, 3, 2, 1 αντιστοιχούν στα τυλίγματα 1a, 1b, 2a και 2b των πινάκων 1.7.1, 1.7.2, 1.7.3.



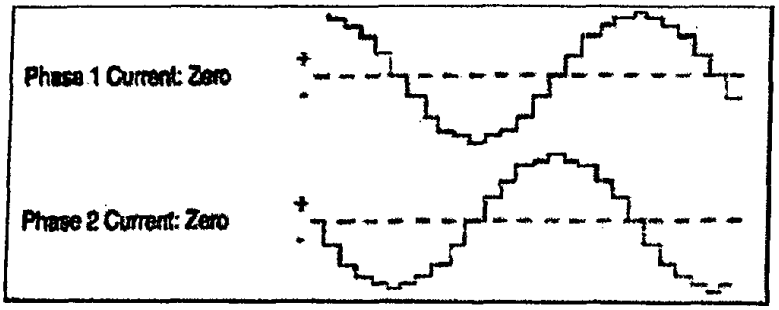
**Σχήμα 1.7.1.α : Wave drive sequence (one phase on)**



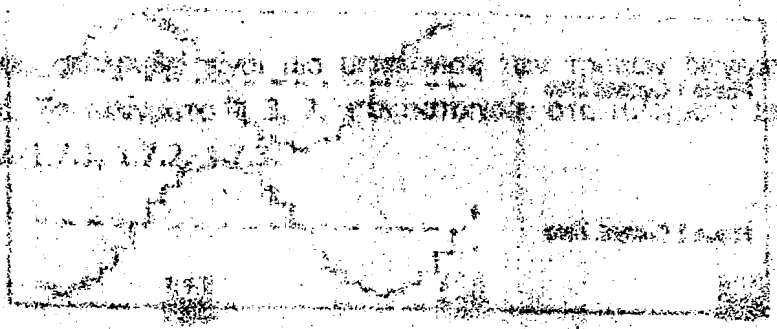
**Σχήμα 1.7.1.β : Normal drive sequence (two phase on)**



**Σχήμα1.7.1.γ : Half step sequence**




**Σχήμα 1.7.2: Κυματοσυνάρτηση ρεύματος στη μέθοδο διέγερσης microstepping**



# Κεφάλαιο 2ο

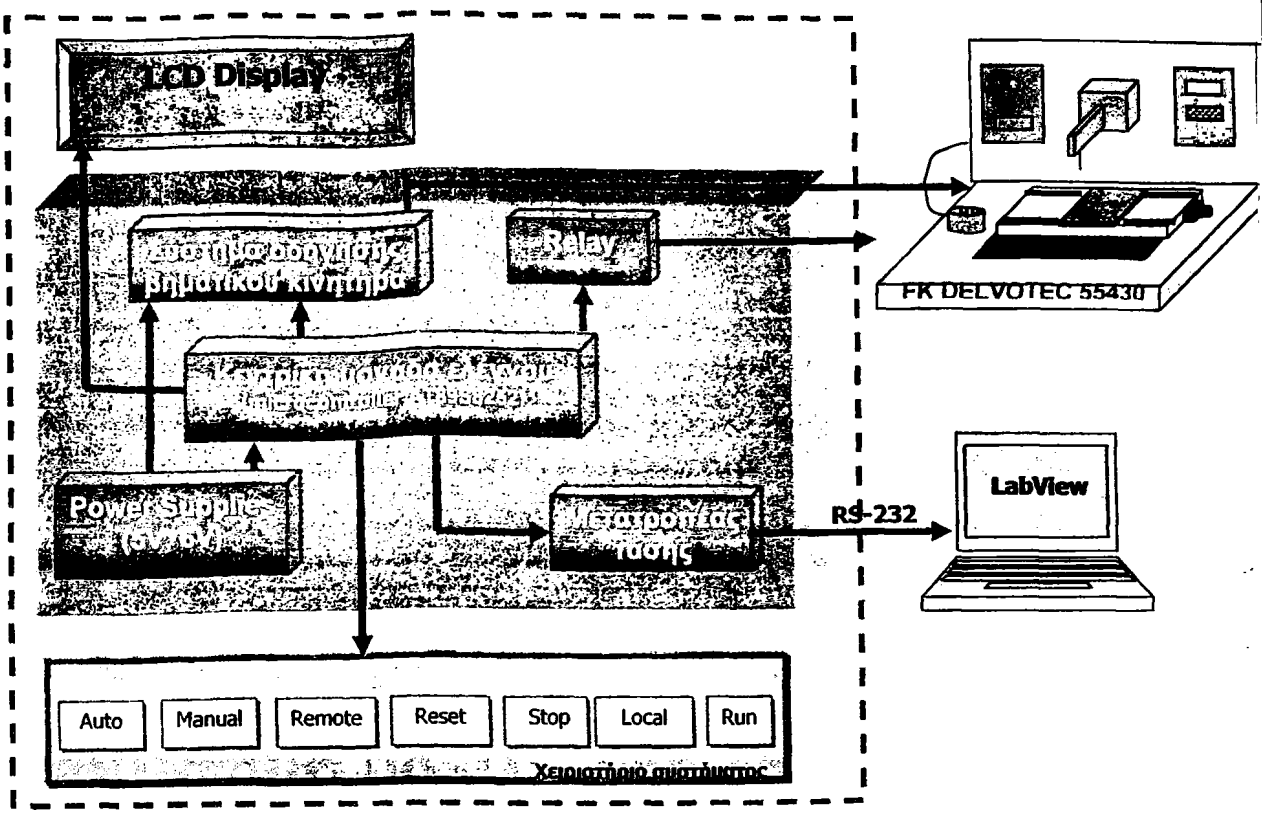
# Υλοποίηση υλικού συστήματος

## 2.1 Εισαγωγή

Όπως αναφέρθηκε και στην εισαγωγή το αντικείμενο αυτής της εργασίας ήταν η ανάπτυξη και η κατασκευή ενός προγραμματιζόμενου συστήματος ελέγχου ενός βηματικού κινητήρα για την κίνηση τράπεζας ακριβείας η οποία ελέγχεται είτε από το χειριστήριο του συστήματος ή από κατάλληλη εφαρμογή σε Η/Υ που αναπτύχθηκε σε περιβάλλον προγραμματισμού LabView.

Το σύστημα οδήγησης του κινητήρα ελέγχεται είτε από μικροελεγκτή που ενσωματώνεται σε αυτό ή από προσωπικό Η/Υ, ενώ περιλαμβάνει εξωτερικό χειριστήριο για την επιλογή των λειτουργιών εκτέλεσης της τράπεζας, καθώς επίσης και μια LCD οθόνη. Ο ρόλος της οθόνης είναι βοηθητικός, αφού σε αυτή θα εμφανίζονται μηνύματα που θα ενημερώνουν τον χρήστη για την τρέχουσα λειτουργία της τράπεζας και θα τον καθοδηγούν για την επόμενη ενέργεια.

Στο λογικό διάγραμμα του Σχήματος 2.1 που ακολουθεί, φαίνεται η αρχιτεκτονική του συστήματος.

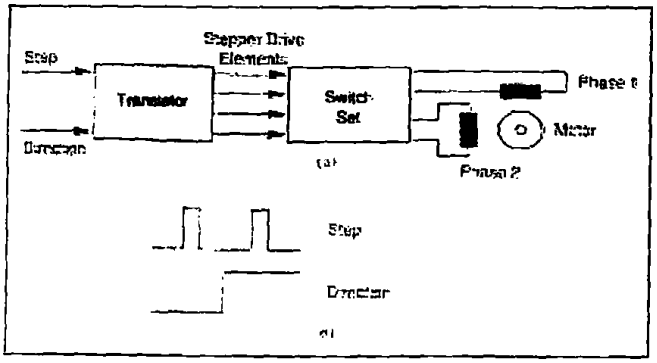



**Σχήμα 2.1: Λογικό διάγραμμα του συστήματος οδήγησης της τράπεζας ακριβείας της εφαρμογής**

Στις επόμενες παραγράφους αυτού του κεφαλαίου παρουσιάζεται αναλυτικά κάθε τμήμα του συστήματος οδήγησης του κινητήρα.

## 2.2 Ένα γενικό σύστημα οδήγησης βηματικών κινητήρων

Ένα απλό σύστημα οδήγησης βηματικού κινητήρα αποτελείται από ένα μεταφραστή (translator) και μία διάταξη διακοπτών (Switch Set). Ο μεταφραστής λαμβάνει ένα σήμα για τον αριθμό των βημάτων που θα εκτελέσει ο κινητήρας και ένα σήμα για τη διεύθυνση περιστροφής του και στέλνει 4 ανεξάρτητα σήματα ελέγχου στη βαθμίδα ισχύος του κινητήρα (Switch Set), το οποίο με τη σειρά του στέλνει σήματα ισχύος σε καθένα από τα τυλίγματα (phase1 και phase2) του κινητήρα. Οι αρχές λειτουργίας του μεταφραστή είναι ίδιες για όλες τις μεθόδους οδήγησης, αν και στη μέθοδο μικροβηματισμού (microstepping) είναι λίγο πιο σύνθετες. Το Σχήμα 2.2.1 είναι το διάγραμμα ενός τυπικού συστήματος οδήγησης. Στο ίδιο σχήμα φαίνεται και το σήμα για τον αριθμό των παλμών που δεν είναι παρά μια απλή παλμοσειρά τετραγωνικών παλμών (ένας παλμός για κάθε βήμα του ρότορα). Το σήμα για τη διεύθυνση περιστροφής είναι ένας παλμός συνεχούς τάσης είτε θετικής ή αρνητικής, ανάλογα με τη φορά περιστροφής που έχει επιλέξει ο χρήστης. Ο translator λαμβάνει τα σήματα αυτά από τη μονάδα ελέγχου, η οποία δεν υπάρχει στο σχήμα και συνήθως είναι ένας ηλεκτρονικός υπολογιστής ή ένας μικροελεγκτής όπως γίνεται και στην παρούσα εργασία.



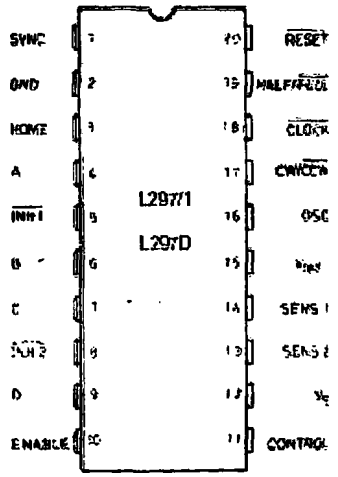

**Σχήμα 2.2.1: Ένα τυπικό κύκλωμα οδήγησης βηματικού κινητήρα**

## 2.3 Το κύκλωμα οδήγησης

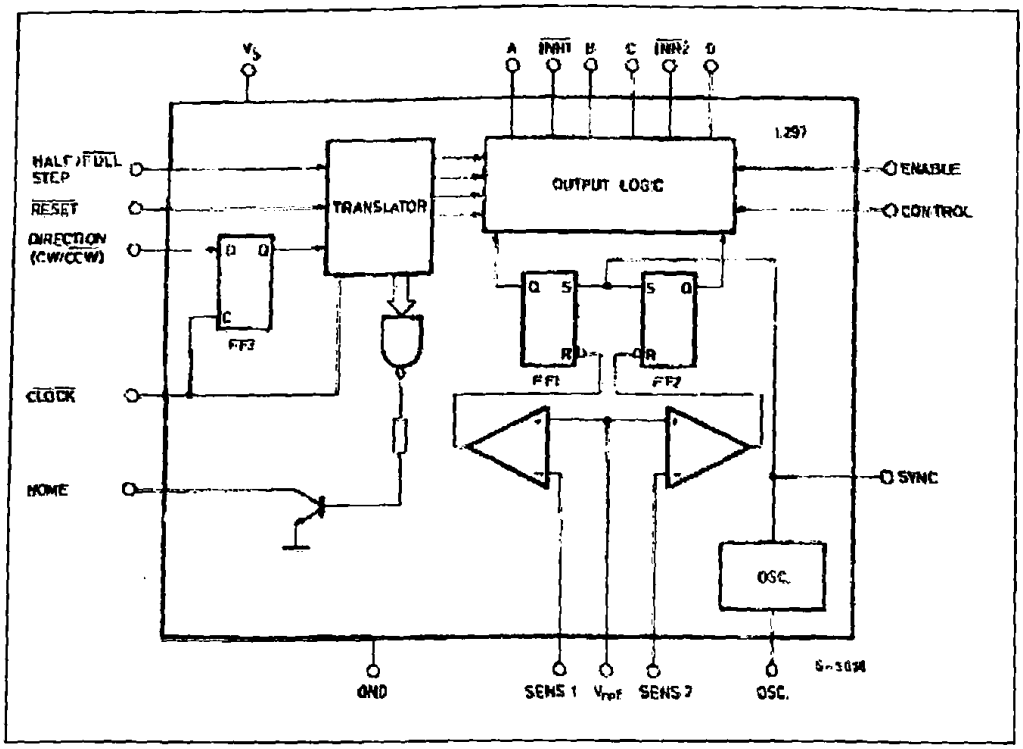
Το κύκλωμα οδήγησης που προτείνουμε σε αυτή την εφαρμογή είναι ταυτόχρονα απλό και ευέλικτο. Βασίζεται στο ζεύγος των ολοκληρωμένων κυκλωμάτων L297-L298 της SGS των οποίων τις αρχές λειτουργίας θα μελετήσουμε στις επόμενες παραγράφους.

### 2.3.1 Το ολοκληρωμένο κύκλωμα L297

Το ολοκληρωμένο αυτό κύκλωμα παράγει σήματα ελέγχου για την οδήγηση διφασικών, διπολικών και τετραφασικών, μονοπολικών κινητήρων και δίνει τη δυνατότητα επιλογής της φοράς περιστροφής σε πλήρη βηματισμό ή ημιβηματισμό του κινητήρα (full step ή half step). Ο προγραμματισμός του γίνεται μέσω συμβατών εισόδων TTL. Το κάθε πλήρες βήμα ή ημιβήμα εκτελείται ταυτόχρονα με το αρνητικό μέτωπο του σήματος που εφαρμόζεται στην είσοδο χρονισμού του (CLOCK). Στο Σχήμα 2.3.1.1 φαίνεται η τοπολογία των ακροδεκτών του και στο επόμενο Σχήμα 2.3.1.2 το γενικό διάγραμμά του.




**Σχήμα 2.3.1.1 : Η τοπολογία των ακροδεκτών του ολοκληρωμένου κυκλώματος L297**




**Σχήμα 2.3.1.2: Το γενικό διάγραμμα του L297**

Το ολοκληρωμένο L297 έχει σχεδιαστεί για εφαρμογές σε συνεργασία με οδηγούς γέφυρας, συστοιχία τρανζίστορ Darlington, ή διακριτών κυκλωμάτων οδήγησης. Δέχεται σήματα ρολογιού, διεύθυνσης περιστροφής και μεθόδου οδήγησης από τον ελεγκτή του συνολικού συστήματος, ο οποίος συνήθως είναι ένας μικροελεγκτής. Παράγει δε σήματα ελέγχου για τη βαθμίδα ισχύος.

Τα κυριότερα στοιχεία του είναι ένας μεταφραστής (translator) που παράγει τα διαδοχικά σήματα των φάσεων και ένα κύκλωμα αποκοπής PWM (Pulse Width Modulation), το οποίο ελέγχει την ένταση του ρεύματος στα τυλίγματα. Ο μεταφραστής παράγει τρεις διαφορετικές σειρές σημάτων εξόδου οι οποίες εξαρτώνται από την κατάσταση της εισόδου Half/Full, για την επίτευξη των τριών μεθόδων διέγερσης που μελετήθηκαν στο προηγούμενο κεφάλαιο.

Ένας εσωτερικός ταλαντωτής διεγείρει ένα δισταθή πολυδονητή (FF1 και FF2) στην αρχή κάθε περιόδου ταλάντωσης και μόνο όταν τα τυλίγματα του στάτορα είναι συνδεδεμένα στην τάση τροφοδοσίας. Λόγω της αυτεπαγωγής των τυλιγμάτων το ρεύμα εξόδου θα αυξάνεται αρχικά με γραμμικό τρόπο, με αποτέλεσμα την εμφάνιση μιας γραμμικά αυξανόμενης τάσης στις αντιστάσεις R1 και R2 που θα συνδέονται στις εισόδους ελέγχου SENS1 και SENS2 και παίζουν το ρόλο αισθητήρων ρεύματος. Όταν η τάση αυτή γίνει ίση με προκαθορισμένη (από τον χρήστη) μέγιστη τιμή $V_{REF}$ δύο εσωτερικοί συγκριτές μηδενίζουν τους δισταθείς πολυδονητές και το ρεύμα του στάτορα μηδενίζεται. Με τον τρόπο αυτό μπορούμε να καθορίσουμε το μέγιστο ρεύμα που διαρρέει τα τυλίγματα του στάτορα. Την ίδια στιγμή ένα δικτύωμα διόδων εξωτερικά του ολοκληρωμένου θα ελαττώνει το εξ' επαγωγής πεδίο του στάτορα. Το πεδίο του στάτορα ελαττώνεται πολύ γρήγορα, επειδή το δικτύωμα των

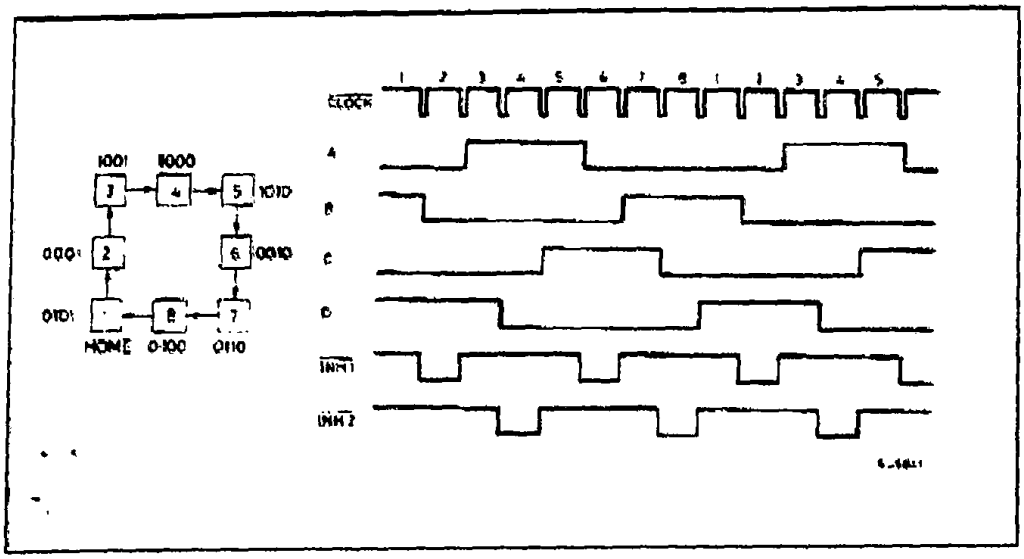
διόδων άγει εκείνη τη χρονική στιγμή αφού η στιγμιαία τάση στα τυλίγματα είναι ελαφρώς μεγαλύτερη από την τάση τροφοδοσίας.

Επίσης από το ολοκληρωμένο παράγονται και δύο σήματα inhibit τα οποία συνδέονται στις δύο εισόδους ενεργοποίησης του ολοκληρωμένου L298, και χρησιμοποιούνται για τη γρήγορη εξασθένηση του ρεύματος όταν τα τυλίγματα απενεργοποιούνται. Στην περίπτωση που το ολοκληρωμένο χρησιμοποιείται για την οδήγηση μονοπολικού κινητήρα, το κύκλωμα αποκοπής λειτουργεί σε αυτές τις γραμμές. Η είσοδος CONTROL, ανάλογα με την τιμή που θα πάρει, καθορίζει αν το κύκλωμα αποκοπής θα λειτουργήσει στις γραμμές φάσεων ABCD (τιμή '1') ή στις γραμμές inhibit INH1 και INH2 (τιμή '0').

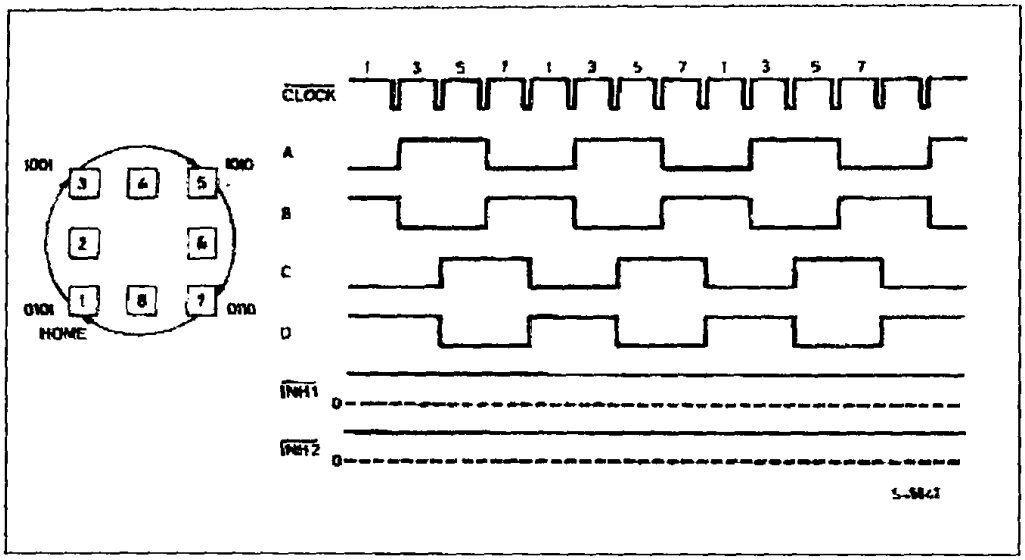
Όταν απαιτείται ο έλεγχος πολλαπλών κυκλωμάτων οδήγησης και πολλών κινητήρων με ένα μόνο σύστημα ελέγχου, είναι προφανές ότι είναι απαραίτητος ο συγχρονισμός των ταλαντωτών όλων των L297. Κάτι τέτοιο είναι αρκετά απλό στην υλοποίηση, καθώς απαιτείται μόνο ένα κύκλωμα σταθεράς χρόνου σε ένα μόνο κύκλωμα οδήγησης. Η έξοδός του SYNC θα χρονίζει και τις αντίστοιχες εισόδους των υπολοίπων κυκλωμάτων οδήγησης.

Η είσοδος ENABLE χρησιμοποιείται για την ενεργοποίηση ή μη του κινητήρα. Όταν η είσοδος ενεργοποίησης βρίσκεται σε χαμηλή λογική στάθμη, ο κινητήρας δεν οδηγείται και ο ρότορας μπορεί να περιστρέφεται ελεύθερα (από κάποιο εξωτερικό αίτιο). Η εφαρμογή χαμηλής λογικής στάθμης στην είσοδο RESET επαναφέρει τον translator στην αρχική του κατάσταση με ABCD = 0101.

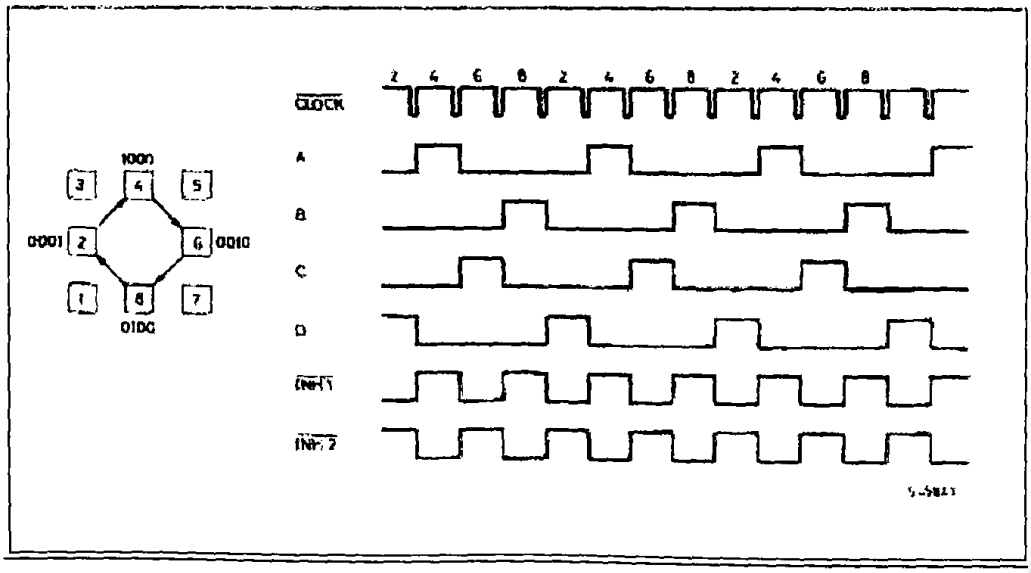
Στα σχήματα 2.3.1.3, 2.3.1.4 και 2.3.1.5 δίνονται οι διαδοχικές καταστάσεις και οι κυματομορφές εξόδου για καθεμιά από τις τρεις μεθόδους οδήγησης. Όπως έχουμε ήδη αναφέρει ο translator κάνει ένα βήμα κατά τη μετάβαση από λογικό μηδέν σε λογικό ένα του σήματος CLOCK. Η είσοδος CW/CCW μας δίνει τη δυνατότητα επιλογής της φοράς περιστροφής του κινητήρα. Εφαρμογή χαμηλής λογικής στάθμης στην παραπάνω είσοδο περιστρέφει τον ρότορα του κινητήρα κατά την φορά κίνησης των δεικτών του ρολογιού( δεξιόστροφα). Ενώ εφαρμογή θετικής λογικής στάθμης περιστρέφει τον ρότορα αντίθετα από τη φορά κίνησης των δεικτών του ρολογιού (αριστερόστροφα). Οδήγηση μισού βήματος επιτυγχάνεται με εφαρμογή λογικού ένα στην είσοδο HALF/FULL. Κανονική οδήγηση επιτυγχάνεται με εφαρμογή λογικού μηδέν στην παραπάνω είσοδο, όταν ο translator είναι σε μια από τις περιττές καταστάσεις (1,3,5,7). Αντίστοιχα οδήγηση κύματος (wave drive mode) επιτυγχάνεται με εφαρμογή λογικού ένα στην είσοδο HALF/FULL, όταν ο translator είναι σε μια από τις άρτιες καταστάσεις (2,4,6,8). [9]




**Σχήμα 2.3.1.3: Half Step Mode**




**Σχήμα 2.3.1.4: Normal Drive Mode**




**Σχήμα 2.3.1.5: Wave Drive Mode**

### 2.3.2 Το ολοκληρωμένο κύκλωμα L298

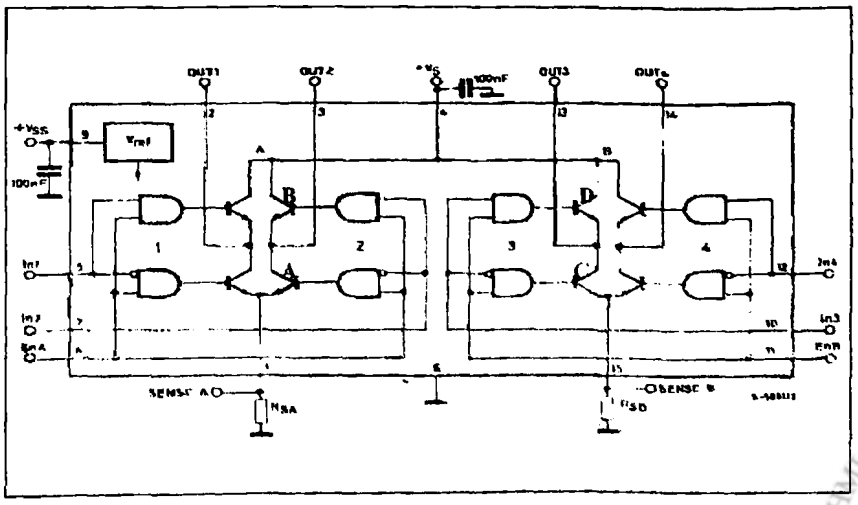
Πρόκειται για ένα μονολιθικό ολοκληρωμένο κύκλωμα το οποίο αποτελείται από δυο γέφυρες ισχύος. Είναι σχεδιασμένο να δέχεται στην είσοδό του λογικά επίπεδα τάσης TTL και να οδηγεί επαγωγικά φορτία όπως πηνία, relays, DC κινητήρες και βηματικούς κινητήρες. Το γενικό διάγραμμά του φαίνεται στο Σχήμα 2.3.2.1.

Στο ολοκληρωμένο παρέχονται δυο είσοδοι ENABLE (EnA και EnB) που του επιτρέπουν να ενεργοποιηθεί και να απενεργοποιηθεί ανεξάρτητα από τις εισόδους των σημάτων οδήγησης. Οι εκπομποί των τρανζίστορ A και C των δύο γεφυρών συνδέονται μεταξύ τους και ο αντίστοιχος ακροδέκτης χρησιμοποιείται για τη σύνδεση μιας εξωτερικής αντίστασης ανίχνευσης ρεύματος.

Το L298 περιέχει δύο βαθμίδες εξόδου (A, B) καθεμιά από τις οποίες είναι ένα ανεξάρτητο κύκλωμα γέφυρας που μπορεί να οδηγήσει ένα επαγωγικό φορτίο με κοινό ή διαφορικό τρόπο. Το ρεύμα που διαρρέει το φορτίο εξέρχεται από το κύκλωμα της γέφυρας και από το ολοκληρωμένο στην έξοδο Sense (SenseA και SenseB). Η σύνδεση δυο εξωτερικών αντιστάσεων RSA και RSB στις εξόδους SenseA και SenseB μας επιτρέπει να υπολογίσουμε την ένταση του ρεύματος .

Καθεμία από τις γέφυρες οδηγείται από τέσσερις λογικές πύλες, οι είσοδοι των οποίων είναι οι In1, In2, EnA και In3, In4, EnB. Οι είσοδοι En θέτουν την κατάσταση της γέφυρας όταν είναι σε λογικό ένα. Αντίθετα ένα λογικό μηδέν στην είσοδο En αναστέλλει την ενεργή λειτουργία της γέφυρας. Όλες οι είσοδοι είναι συμβατές με σήματα TTL.

Για να λειτουργήσει η διπλή γέφυρα ισχύος του L298 χρειάζεται κάποιον ελεγκτή όπως είναι το L297. Η γέφυρα μπορεί να παρέχει διαρκώς ρεύμα έντασης 2A, ενώ στιγμιαία μπορεί να δώσει μέχρι και 3A. Η μέγιστη επιτρεπτή τάση εισόδου ανέρχεται σε 36V. Το ρεύμα που διαρρέει τις δυο γέφυρες προκαλεί πτώση τάσης στις αντιστάσεις που είναι συνδεδεμένες στις εξόδους SenseA και SenseB. Το ολοκληρωμένο L297 συγκρίνει την πτώση τάσης στις αντιστάσεις με τη $V_{REF}$ . Όταν οι δύο τάσεις εξισωθούν το ρεύμα παίρνει τη μέγιστη τιμή του.



**Σχήμα 2.3.2.1: Το γενικό διάγραμμα του ολοκληρωμένου κυκλώματος L298**

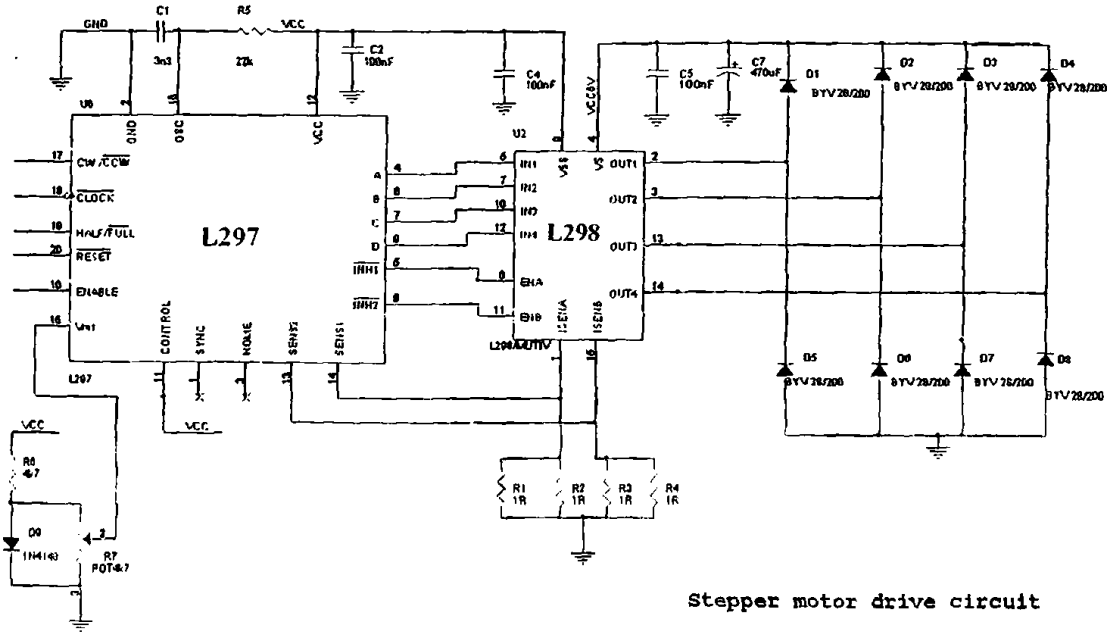
### 2.3.3 Το κύκλωμα οδήγησης του κινητήρα της εφαρμογής μας

Με βάση τις αρχές που περιγράφηκαν στις δυο προηγούμενες παραγράφους σχεδιάστηκε το κύκλωμα του σχήματος 2.3.3.1 για την οδήγηση του κινητήρα μας. Είναι στην ουσία ένα κύκλωμα η λειτουργία του οποίου βασίζεται στα χαρακτηριστικά των ολοκληρωμένων L297-L298.

Τροφοδοτώντας κατάλληλα τις εισόδους του L297 με τη χρήση ενός μικροελεγκτή, μπορούμε να οδηγήσουμε τον κινητήρα μας με έναν από τους τρεις τρόπους οδήγησης [κύματος (wave drive), κανονικού βήματος (full step) και μισού βήματος (half step)]. Οι γραμμές των φάσεων A,B,C,D οδηγούνται απευθείας στις εισόδους των πυλών ελέγχου των κυκλωμάτων γέφυρας του L298, In1, In2, In3 και In4, ενώ οι έξοδοι INH1 και INH2 οδηγούνται στις εισόδους EnA και EnB.

Για να πετύχουμε μέγιστο ρεύμα οδήγησης ίσο με 0,8A, απαραίτητο για την οδήγηση του κινητήρα μας, θα πρέπει η τάση στα άκρα των αντιστάσεων R1, R2 και R3, R4 να είναι ίση με τη $V_{REF.}$ Αυτό γίνεται εφικτό εάν τροφοδοτήσουμε τη $V_{REF}$, μέσω ενός διαιρέτη τάσης, με τάση 0,8A*0,5Ω=0,4V.

Το δικτύωμα των διόδων που προστατεύει τα τυλίγματα του στάτορα από φαινόμενα αυτεπαγωγής, έχει χρόνο απόκρισης t≤200ns και αντέχει μέση τιμή ρεύματος έως και 2A και τάσης 200V. Τέλος οι πυκνωτές των 100nF παρέχουν προστασία από το θόρυβο στα ολοκληρωμένα L297 και L298, ενώ για R=22kΩ και C=3,3nF η συχνότητα του κυκλώματος αποκοπής είναι f=1/0,69RC=1,996kHz.



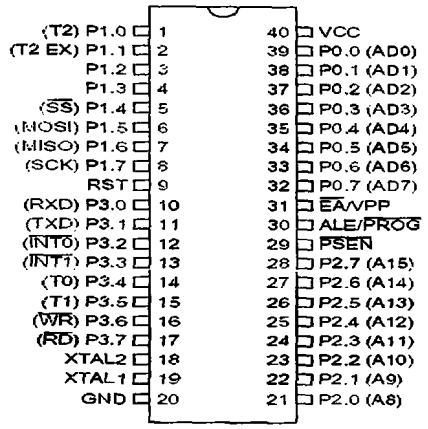

**Σχήμα 2.3.3.1: Το κύκλωμα οδήγησης του κινητήρα της εφαρμογής**

## 2.4 Ο μικροελεγκτής AT89S8252

Όλες οι λειτουργίες του συστήματός μας ελέγχονται από τον μικροελεγκτή που παίζει επιπλέον και το ρόλο της μονάδας ελέγχου του κυκλώματος οδήγησης του κινητήρα μας. Στη συγκεκριμένη περίπτωση ο μικροελεγκτής που χρησιμοποιήθηκε είναι ο AT89S8252 της εταιρείας ATMEL.

### 2.4.1 Γενικά

Ο μικροελεγκτής AT89S8252 είναι ένας 8-bit μικροελεγκτής η τοπολογία του οποίου φαίνεται στο Σχήμα 2.4.1.1.




**Σχήμα 2.4.1.1: Η τοπολογία των ακροδεκτών του μικροελεγκτή AT89S8252**

Μερικά από τα κύρια χαρακτηριστικά του μικροελεγκτή είναι τα παρακάτω:

- Εσωτερική μνήμη RAM 256 Bytes
- Επαναπρογραμματιζόμενη εσωτερική μνήμη Flash 8Kbytes
- Μνήμη EPROM 2kBytes
- 4 θύρες I/O με 8 προγραμματιζόμενους ακροδέκτες η καθεμία
- 2 16-bit καταχωρητές τους Program Counter και Data Pjointer
- 3 16-bit χρονιστές / μετρητές
- 9 πηγές παραγωγής interrupt
- Σειριακή μεταφορά δέκτη / πομπού full duplex
- Προγραμματιζόμενο watchdog timer
- Ταλαντωτή και κρύσταλλο ρολογιού

Είναι σημαντικό να σημειωθεί ότι μερικοί ακροδέκτες μπορεί να χρησιμοποιηθούν για περισσότερες από μια λειτουργίες, οι οποίες, όμως, δεν μπορούν να υλοποιηθούν ταυτόχρονα. Για παράδειγμα ο ακροδέκτης P3.0 της θύρας P3 μπορεί να χρησιμοποιηθεί ως ακροδέκτης γενικής χρήσης I/O ή ως είσοδος RXD

στον καταχωρητή του σειριακού δέκτη δεδομένων, SBUF. Ο προγραμματιστής (χρήστης) αποφασίζει για ποια από τις δύο λειτουργίες θα χρησιμοποιηθεί ο ακροδέκτης και ανάλογα σχεδιάζει το hardware και γράφει το πρόγραμμα.

Η καρδιά του μικροελεγκτή είναι ένα κύκλωμα που παράγει παλμούς με τους οποίους συγχρονίζονται οι εσωτερικές του λειτουργίες. Οι ακροδέκτες XTAL1 και XTAL2 προορίζονται για τη σύνδεση ενός κυκλώματος ταλαντωτή. Ο ταλαντωτής που αποτελείται από έναν κρύσταλλο, 2 πυκνωτές και έναν αντιστροφέα μέσα στο ολοκληρωμένο, παράγει μια παλμοσειρά στη συχνότητα του κρυστάλλου.

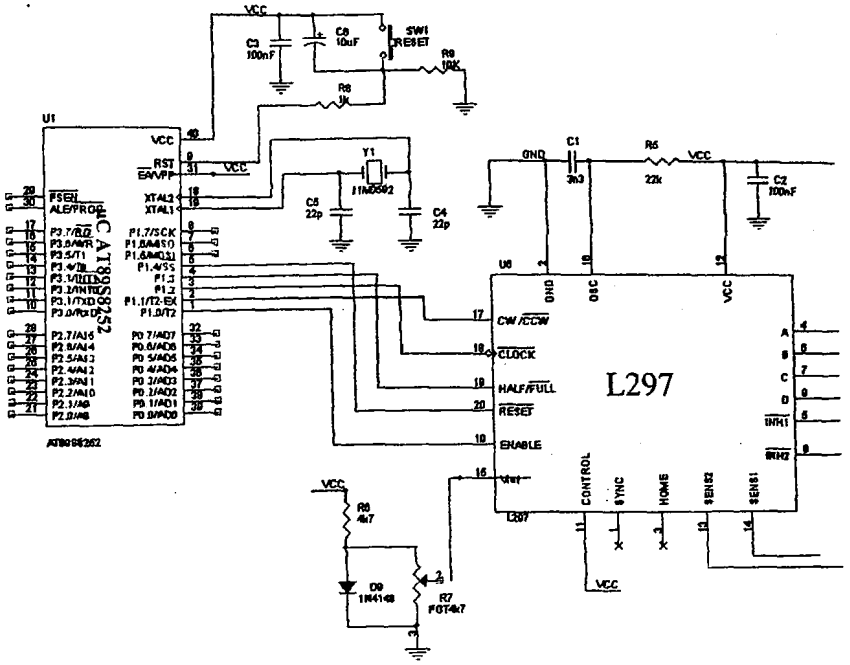
Κώδικας μπορεί να προσπελαστεί αποκλειστικά στην εξωτερική μνήμη, εάν γίνει σύνδεση του ακροδέκτη της εξωτερικής προσπέλασης EA στη γείωση. Σε αντίθετη περίπτωση ο EA πρέπει να είναι πάντα συνδεδεμένος σε λογικό ένα ώστε ο μικροελεγκτής να εκτελεί τον κώδικα από την εσωτερική μνήμη. [12]

### 2.4.2 Η λειτουργία του μικροελεγκτή ως μονάδα ελέγχου

Η συμπεριφορά του μικροελεγκτή στο σύστημά μας καθορίζεται από κατάλληλο κώδικα προγραμματισμού, ο οποίος φορτώνεται στη μνήμη του. Με την εφαρμογή της τροφοδοσίας ή το πάτημα του διακόπτη RESET ο κώδικάς εκτελείται από την αρχή.

Το σύστημα οδήγησης του βηματικού κινητήρα συνδέεται στη θύρα P1 του μικροελεγκτή. Ανάλογα με τις εντολές που λαμβάνει από τον κώδικα θέτει τους ακροδέκτες της παραπάνω θύρας στην ανάλογη κατάσταση ορίζοντας έτσι τις παραμέτρους κίνησης που επιθυμεί ο χρήστης. Έτσι αν μικροελεγκτής λάβει για παράδειγμα την εντολή να κινήσει τον κινητήρα προς τα αριστερά, με οδήγηση πλήρους βήματος, τότε εκτελεί τις ακόλουθες ενέργειες: α) θέτει αρχικά τον ακροδέκτη P1.0 σε λογικό ένα για να ενεργοποιήσει τον κινητήρα, β) θέτει τον ακροδέκτη P1.1 ο οποίος συνδέεται με τον ακροδέκτη CW/CCW του ολοκληρωμένου L297 σε λογικό ένα γ) θέτει τον ακροδέκτη P1.3 που συνδέεται με τον ακροδέκτη HALF/FULL σε λογικό μηδέν ενώ δ) ταυτόχρονα παράγει μια σειρά παλμών στον ακροδέκτη P1.2 που αντιστοιχεί στο CLOCK του L297.

Το κύκλωμα RC (R5=22kΩ, C1=3,3nF) παράγει τους παλμούς που αντιστοιχούν στο CLOCK του L297, ενώ οι πυκνωτές C3 και C2 μεγέθους 100nF φιλτράρουν την τάση τροφοδοσίας από το θόρυβο. Το Σχήμα 2.4.2.1 που ακολουθεί, δείχνει τη σύνδεση του μικροελεγκτή με το σύστημα οδήγησης του κινητήρα και πιο συγκεκριμένα με το ολοκληρωμένο L297.

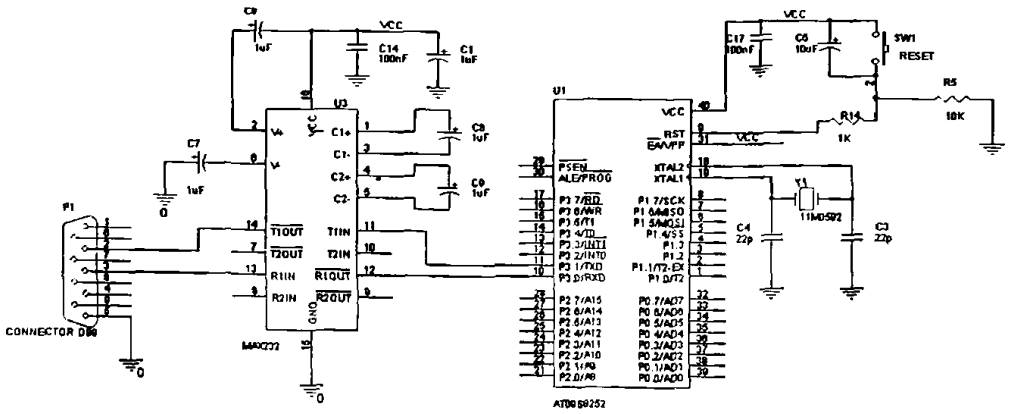



**Σχήμα 2.4.2.1: Σύνδεση μονάδας ελέγχου (μC) - συστήματος οδήγησης**

## 2.5 Η Βαθμίδα διεπαφής με την RS-232

Σημαντικός είναι ο ρόλος του μικροελεγκτή για την επικοινωνία με το PC, όπου ο χρήστης μέσω κατάλληλης εφαρμογής ελέγχει πλήρως τη λειτουργία του συστήματος όπως αναλύεται στο Κεφάλαιο 4. Η επικοινωνία με τον υπολογιστή γίνεται μέσω μιας συνηθισμένης σειριακής θύρας RS-232. Η θύρα RS-232 καταλήγει σε ένα συνδετήρα τύπου D με 9 ακροδέκτες. Ανάμεσα στο PC και στη σειριακή παρεμβάλλεται το ολοκληρωμένο MAX 232 της εταιρείας MAXIM. Το ολοκληρωμένο αυτό λειτουργεί ως μεταφραστής επιπέδων μεταξύ των επιπέδων TTL της σειριακής θύρας του μικροελεγκτή και των επιπέδων τάσης σύμφωνα με τα οποία λειτουργεί η θύρα RS-232, μετατρέπει δηλαδή τα σήματα 0...5V από τον μικροελεγκτή σε συμμετρικές τάσεις ±12,5V και αντίστροφα. Γι' αυτό το λόγο το MAX-232 χρησιμοποιεί έναν ενσωματωμένο ανυψωτή τάσης. Ο ανυψωτής τάσης χρησιμοποιεί τους πυκνωτές C6, C7, C8 και C9 της τάξης του 1uF για να παράγει τη συμμετρική τάση ±12,5V από την τάση τροφοδοσίας των 5V. Οι πυκνωτές C14 των 100nF και C1 του 1uF χρησιμοποιούνται για το φιλτράρισμα της τάσης τροφοδοσίας από το θόρυβο που τυχόν παρεμβάλλουν τα υπόλοιπα υποκυκλώματα του συστήματος.

Τα δεδομένα που στέλνει ο μικροελεγκτής στον Η/Υ εκπέμπονται από τον ακροδέκτη TXD (P3.1) ο οποίος συνδέεται με τον ακροδέκτη T1IN του ολοκληρωμένου MAX232. Αφού μετατραπούν σε λογικά επίπεδα RS-232, μεταφέρονται στον Η/Υ μέσω του ακροδέκτη T1OYT. Ομοίως ο Η/Υ στέλνει τα δεδομένα στον μικροελεγκτή μέσω του ακροδέκτη R1IN και αφού μετατραπούν σε σήματα TTL, μέσω του ακροδέκτη R1OYT του MAX232 αποστέλλονται στον μικροελεγκτή και λαμβάνονται από τον ακροδέκτη RXD ( P3.0). Το Σχήμα 2.5.1 δείχνει τον τρόπο σύνδεσης του μικροελεγκτή με το PC.



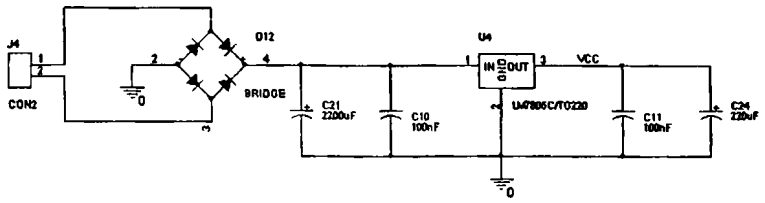

**Σχήμα 2.5.1: Σύνδεση μC – PC**

## 2.6 Τροφοδοτικά

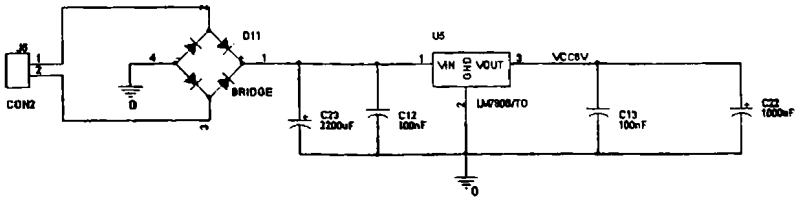
Η συσκευή λειτουργεί με δύο διαφορετικές τάσεις τροφοδοσίας: +5V λογική τάση ( σήματα TTL) και +6V για το κύκλωμα οδήγησης του κινητήρα. Οι δύο τάσεις τροφοδοσίας παρέχονται από δύο διαφορετικά και ανεξάρτητα μεταξύ τους τροφοδοτικά για λόγους μείωσης του θορύβου. Τα Σχήματα 2.6.1 και 2.6.2 απεικονίζουν τα κυκλώματα των τροφοδοτικών των +5V και +6V αντίστοιχα.

Για τη λειτουργία των τροφοδοτικών χρησιμοποιείται μετασχηματιστής των 9V και 1A, ο οποίος τροφοδοτεί τις γέφυρες ανόρθωσης μέσω των συνδετήρων J4 και J5. Μέσω της γέφυρας ανόρθωσης παίρνουμε ανορθωμένη τάση ίση με 12,7V, η οποία εξομαλύνεται από τους ηλεκτρολυτικούς πυκνωτές των 2200μF, ενώ παράλληλα προς αυτούς συνδέονται πυκνωτές 100nF που αποκόπτουν τον θόρυβο που παρεμβάλλει η τροφοδοσία.

Η τάση των 12,7V τροφοδοτεί τους ευρέως χρησιμοποιούμενους σταθεροποιητές τάσης LM7805 και LM7806 και στην έξοδό τους παίρνουμε τις σταθεροποιημένες τάσεις των 5V και 6V αντίστοιχα. Όπως προτείνει ο κατασκευαστής, στην έξοδο των σταθεροποιητών έχουμε τοποθετήσει τους ανάλογους ηλεκτρολυτικούς πυκνωτές για τη σωστή λειτουργία αυτών.




**Σχήμα 2.6.1: Τροφοδοτικό 5V**




**Σχήμα 2.6.2: Τροφοδοτικό 6V**

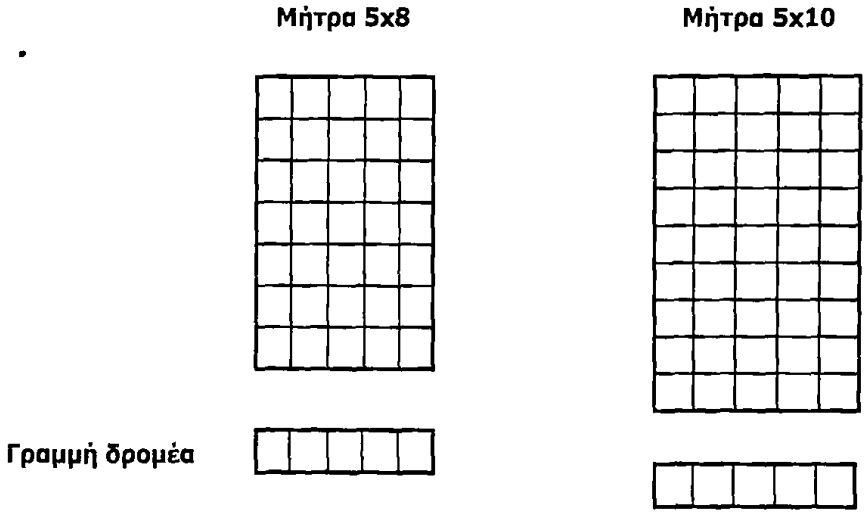
## 2.7 Η οθόνη υγρών κρυστάλλων (Liquid Crystal Display)

Όπως αναφέρθηκε και στην εισαγωγή αυτής της εργασίας το σύστημα θα αποτελείται και από μια οθόνη LCD 2 γραμμών και 16 χαρακτήρων ανά γραμμή, η οποία θα μας δίνει πληροφορίες για τη κατάσταση στην οποία βρίσκεται το σύστημά μας όπως π.χ. το όνομα της λειτουργίας που εκτελείται κάθε φορά, το σύνολο των βημάτων που έχουν εκτελεστεί, ενώ θα εμφανίζει και τα ανάλογα μηνύματα για να μας ενημερώνει για τη θέση της τράπεζας, ή σε περίπτωση σφάλματος. [3],[12]

### 2.7.1 Η λειτουργία της οθόνης LCD

Τα τελευταία χρόνια ένας μεγάλος αριθμός ηλεκτρονικών εφαρμογών απαιτούν μονάδες απεικόνισης με περισσότερες δυνατότητες από τους ενδείκτες των 7 τομέων και γι' αυτό χρησιμοποιούν τις 'έξυπνες' οθόνες υγρών κρυστάλλων LCD. Η ικανότητα τους να εμφανίζουν όχι μόνο αριθμούς, αλλά γράμματα, σύμβολα ακόμη και λέξεις τις έχει κάνει πιο εύχρηστες από τα υπόλοιπα εξαρτήματα παρόμοιας λειτουργίας.

Ο κάθε χαρακτήρας στην οθόνη δημιουργείται από μια μήτρα από τελείες (dot matrix). Η μήτρα αποτελείται από 5 στήλες και 8 (5x8) ή 10 (5x10) γραμμές από τελείες (Σχήμα 2.7.1.1). Η $8^η$ και η $10^η$ γραμμή της κάθε μήτρας εξυπηρετούν την εμφάνιση του δρομέα (cursor) στην οθόνη.




**Σχήμα 2.7.1.1: Μήτρα χαρακτήρων**

Πίσω από κάθε οθόνη υγρών κρυστάλλων βρίσκεται ένας ελεγκτής που ανάλογα με την εντολή που θα δεχθεί θα εκτελέσει και την αντίστοιχη λειτουργία. Ο ελεγκτής της οθόνης της εφαρμογής μας είναι το ολοκληρωμένο κύκλωμα HD44780 της εταιρίας Hitachi ο οποίος έχει καθιερωθεί και είναι σε θέση να ελέγχει την

απεικόνιση σε οθόνες υγρών κρυστάλλων. Ο ελεγκτής HD44780 παρέχει τρεις γραμμές ελέγχου όπως επίσης και 4 ή 8 γραμμές που αποτελούν τον δίαυλο δεδομένων.

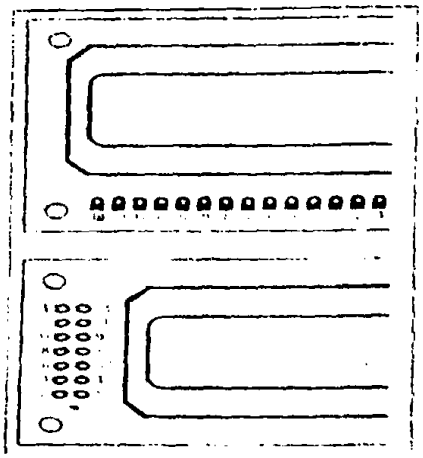
Η πλειοψηφία των οθονών έχει καθορισμένη τοπολογία 14 ακροδεκτών, 8 από τους οποίους αποτελούν τον δίαυλο δεδομένων, με 3 γραμμές ελέγχου και 2 γραμμές ισχύος που τοποθετούνται σε 2 σειρές των 7 ακροδεκτών η καθεμιά ή όλοι μαζί σε μια απλή σειρά. Οι δυο διαφορετικές συνδέσεις φαίνονται στο Σχήμα 2.7.1.2. Η οθόνη της εφαρμογής μας ανήκει στην πρώτη κατηγόρια. Στον πίνακα 2.7.1 δίνεται η ονομασία και λειτουργία κάθε ακροδέκτη. Οι ακροδέκτες 1 και 2 είναι οι γραμμές τροφοδοσίας Vss και Vdd. Ο ακροδέκτης Vdd συνδέεται σε θετική τάση, ενώ ο Vss στη γείωση. Η λειτουργία του ακροδέκτη Vee είναι να μεταβάλλει το contrast της οθόνης, ενώ η τιμή της καθορίζεται από ένα διαιρέτη τάσης που συνδέεται σε αυτόν. Οι τρεις γραμμές έλεγχου αναφέρονται ως EN, RS και RW και η λειτουργία τους αναλύεται παρακάτω:

Η γραμμή **EN** καλείται **'ENABLE'**. Αυτή η γραμμή ελέγχου χρησιμοποιείται για να ενημερώσει την οθόνη ότι στέλνουμε δεδομένα. Για να στείλουμε δεδομένα στην οθόνη το πρόγραμμα πρέπει να θέσει αρχικά τη γραμμή αυτή σε λογικό ένα και έπειτα να θέσει τις άλλες 2 γραμμές έλεγχου και να μεταφέρει τα δεδομένα στον δίαυλο δεδομένων. Όταν οι υπόλοιπες γραμμές έλεγχου είναι απολύτως έτοιμες θέτουμε το EN ξανά σε λογικό μηδέν. Η μετάβαση από λογικό ένα σε λογικό μηδέν ενημερώνει το ολοκληρωμένο HD44780 να πάρει τα δεδομένα που βρέθηκαν στις άλλες γραμμές και στον δίαυλο δεδομένων (data bus) και να τα μεταχειριστεί σαν σχόλια.

Η γραμμή **RS** είναι η **"REGISTER SELECT"** γραμμή. Όταν RS=0 τα δεδομένα μεταχειρίζονται ως σχόλια ή ως μια ειδική εντολή (όπως καθαρισμός οθόνης, καθορισμός της θέσης του δρομέα...). Όταν RS=1, τα δεδομένα που στέλνονται είναι κείμενο το οποίο θα εμφανιστεί στην οθόνη.

Η γραμμή **RW** είναι η γραμμή **READ/WRITE**. Όταν RW=0 οι πληροφορίες στον δίαυλο γράφονται στην οθόνη. Όταν RW=1 το πρόγραμμα διαβάζει την οθόνη. Μόνο μια εντολή (Get LCD status) είναι σχόλιο ανάγνωσης. Όλα τα υπόλοιπα είναι για έγγραφη έτσι το RW πρέπει να είναι πάντα 0.

Τελειώνοντας αν ο δίαυλος αποτελείται από τέσσερις ή οχτώ γραμμές εξαρτάται από τον τρόπο λειτουργίας που επιλέχτηκε από το χρηστή. Στην περίπτωση ενός 8-bit διαύλου οι γραμμές αναφέρονται ως D0,..D7.
Περισσότερες πληροφορίες για τον προγραμματισμό της LCD οοθόνης υπάρχουν στο Παράρτημα ' Ο προγραμματισμός της LCD οθόνης '.



**Σχήμα 2.7.1.2: Τοπολογία ακροδεκτών των δυο βασικών τύπων οθόνης LCD**

| Pin No | Όνομα | Λειτουργία |
|---|---|---|
| 1 | Vss | Ground |
| 2 | Vdd | +ve Supply (5V) |
| 3 | Vee | Contrast (0-5V) |
| 4 | RS | Register Select |
| 5 | R/W | Read/Write |
| 6 | EN | Enable |
| 7 | D0 | Data bit 0(LSB) |
| 8 | D1 | Data bit 1 |
| 9 | D2 | Data bit 2 |
| 10 | D3 | Data bit 3 |
| 11 | D4 | Data bit 4 |
| 12 | D5 | Data bit 5 |
| 13 | D6 | Data bit 6 |
| 14 | D7 | Data bit 7(MSB) |

**Πίνακας 2.7.1: Ονομασία και λειτουργία ακροδεκτών της LCD οθόνης**

Στην παρούσα εφαρμογή ο δίαυλος δεδομένων της οθόνης συνδέεται στη θύρα P0 του μικροελεγκτή, ενώ οι τρεις γραμμές ελέγχου RS, R/W και EN στους ακροδέκτες P2.0, P2.1 και P2.2 αντίστοιχα.

## 2.8 Το τελικό κύκλωμα

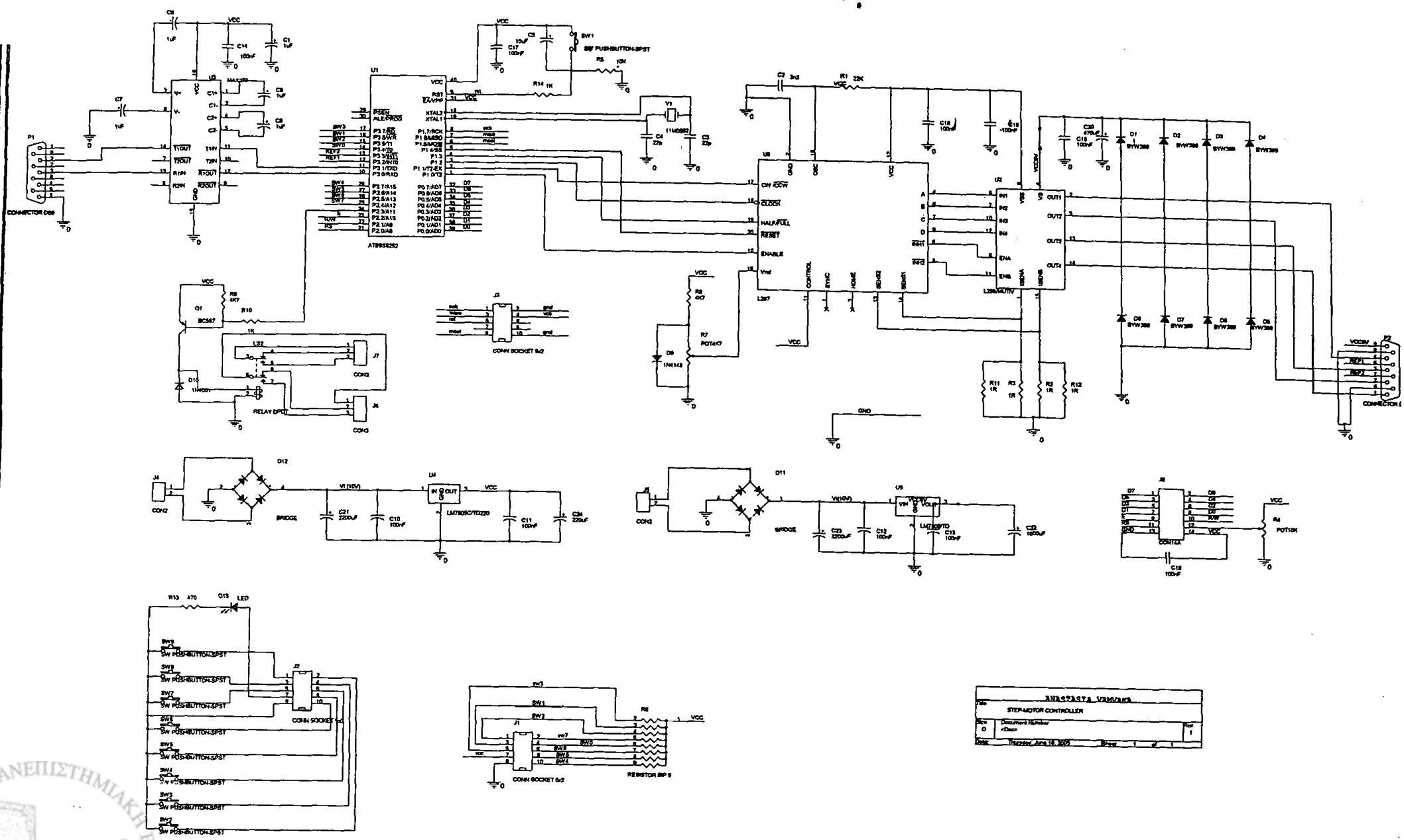
Συνδέοντας τα υποκυκλώματα των προηγούμενων παραγραφών προκύπτει το τελικό κύκλωμα το σχεδιαστικό του οποίου φαίνεται του Σχήματος 2.8.1. Η σχεδίασή του έγινε με χρήση του προγράμματος ORCAD 9.1.

Όπως έχουμε ήδη αναφέρει στην εισαγωγή το σύστημα έχει την δυνατότητα να εκτελεί συγκεκριμένες λειτουργίες, που θα μελετηθούν στο επόμενο κεφάλαιο. Η επιλογή αυτών των λειτουργιών γίνεται από ένα σύνολο διακοπτών που ονομάζουμε 'χειριστήριο'. Το χειριστήριο έχει κατασκευαστεί εξωτερικά από το συνολικό σύστημα και συνδέεται στους ακροδέκτες P3.4-P3.7 και P2.4-P2.7 των θυρών P3 και P2 του μικροελεγκτή. Με το πάτημα οποιουδήποτε διακόπτη, ο αντίστοιχος ακροδέκτης του μικροελεγκτή αλλάζει κατάσταση, μεταβαίνοντας από λογικό '1' σε λογικό '0', οπότε ο μικροελεγκτής ενημερώνεται και εκτελεί την λειτουργία που έχουμε ορίσει σε κάθε διακόπτη.

Ο σκοπός του συστήματός μας είναι ο έλεγχος της εξωτερικής συσκευής μικροσυγκόλλησης, bonder. Αυτό επιτυγχάνεται με τη χρησιμοποίηση ενός relay. Επειδή, όμως, ο μικροελεγκτής δεν μπορεί να παρέχει το ρεύμα που χρειάζεται το relay, έχουμε τοποθετήσει ένα τρανζίστορ, η βάση του οποίου συνδέεται, μέσω μιας αντίστασης, στον ακροδέκτη P2.3 του μικροελεγκτή. Ταυτόχρονα ο συλλέκτης του τρανζίστορ συνδέεται με το πηνίο του relay. Το τρανζίστορ έχει τη δυνατότητα να παρέχει το ρεύμα που απαιτείται για το κλείσιμο του relay. Μια δίοδος έχει τοποθετηθεί παράλληλα στο relay γιατί κατά την απενεργοποίηση του το πηνίο παράγει ανάστροφη τάση η οποία μπορεί να καταστρέψει το τρανζίστορ. Στην ουσία ο ρόλος της διόδου είναι να παρέχει προστασία στο τρανζίστορ από επαγόμενες τάσεις.

Για την εύρεση των τερματικών θέσεων της τράπεζας και για αποφυγή καταστροφής της, η τράπεζα διαθέτει δυο μηχανικούς διακόπτες τους REF1 και REF2. Οι διακόπτες αυτοί είναι συνδεδεμένοι στους ακροδέκτες P3.2 και P3.3 του μικροελεγκτή. Όταν η τράπεζα φτάνει στο δεξιό ή αριστερό άκρο της διαδρομής της, με το κλείσιμο των διακοπτών REF1 και REF2 οι αντίστοιχοι ακροδέκτες αλλάζουν κατάσταση μεταβαίνοντας από λογικό '1' σε λογικό '0'. Με τη μετάβαση αυτή ο μικροελεγκτής ενημερώνεται και μέσω του προγράμματος σταματά οποιαδήποτε κίνηση της τράπεζας και απενεργοποιείται ο κινητήρας.

Τέλος όπως αναφέρθηκε και στην προηγούμενη παράγραφο στο σύστημα υπάρχει μία LCD οθόνη που μας δίνει πληροφορίες για την κατάσταση του συστήματος. Η οθόνη συνδέεται στο σύστημα μέσω του κονέκτορα J6. Μέσω ενός ποτενσιόμετρου που συνδέεται στον ακροδέκτη VIE της LCD (ακροδέκτης 12 του κονέκτορα J6) ο χρήστης καθορίζει το contrast της οθόνης.




**Σχήμα 2.8.1: Το τελικό κύκλωμα**

# Κεφάλαιο 3ο

## Το λογισμικό του συστήματος

### 3.1 Εισαγωγή

Ο προγραμματισμός της λειτουργίας του μικροελεγκτή έγινε με βάση το σύνολο των εντολών της οικογένειας των μικροελεγκτών 8051 με το οποίο είναι απόλυτα συμβατός ο AT89S8252. Αυτό περιλαμβάνει εντολές μεταφοράς δεδομένων, εντολές αριθμητικών και λογικών πράξεων και εντολές μεταφοράς της ροής του προγράμματος.

### 3.2 Γενική περιγραφή του κώδικα προγραμματισμού

Για την περιγραφή του κώδικα της παρούσας εφαρμογής χρησιμοποιήθηκε ο editor και ο assembler μVision της Keil Software που παρέχει εργαλεία για την διόρθωση και την υπόδειξη λαθών. Προσομοίωση της λειτουργίας του κώδικα έγινε με τον simulator dScope - 251/51 της ίδιας εταιρείας.

Όπως αναφέρθηκε και στην εισαγωγή ο χρήστης έχει την δυνατότητα να επιλέξει ανάμεσα σε δύο τρόπους λειτουργίας του συστήματος: της αυτόνομης (LOCAL) όπου όλες τις εργασίες τις εκτελεί ο μικροελεγκτής ανάλογα με την εντολή που θα λάβει με το πάτημα οποιουδήποτε διακόπτη από το χειριστήριο, και της λειτουργίας μέσω υπολογιστή (REMOTE), με αποστολή χαρακτήρων μέσω της σειριακής RS-232. Στη δεύτερη περίπτωση ο υπολογιστής έχει το ρόλο του master και ανάλογα με την εντολή (ένα byte) που θα στείλει στον μικροελεγκτή, αυτός θα εκτελέσει και την αντίστοιχη εργασία.

Για την υλοποίηση της σειριακής επικοινωνίας (ο AT89S8252 έχει τρεις διαφορετικούς τρόπους λειτουργίας ) είναι απαραίτητο να τεθούν οι κατάλληλες τιμές στους καταχωρητές ειδικής λειτουργίας SFRs που είναι επιφορτισμένοι με τον έλεγχο της λειτουργίας του χρονιστή Timer 1 και της σειριακής θύρας. Έτσι φορτώνεται στον καταχωρητή TMOD η κατάλληλη τιμή (20h) ώστε η σειριακή λειτουργία να γίνει με βάση το δεύτερο τρόπο (ασύγχρονη σειριακή επικοινωνία). Για να επιτευχθεί επικοινωνία με ρυθμό μετάδοσης (baud rate) 1200bps φορτώνεται στον καταχωρητή TH1 η τιμή 0d0h. Η ταχύτητα επικοινωνίας είναι παράμετρος και της συχνότητας του εξωτερικού κρυστάλλου, που ρυθμίζει και τον χρόνο του κύκλου μηχανής του μικροελεγκτή και προκύπτει από την παρακάτω σχέση:

$$\textbf{Baud Rate} = \frac{2^{SMOD} * (OscillatorFrequency)}{384 * (265 - TH1)} \qquad [1]$$

Ο κώδικας αρχικοποιεί και την τιμή του καταχωρητή που ελέγχει τα interrupts. Όπως αναφέραμε στο προηγούμενο κεφάλαιο, ο AT89S8252 έχει εννέα πηγές παραγωγής interrupts από τις οποίες οι τέσσερις ελέγχουν τη σειριακή θύρα. Για να εκμεταλλευτούμε τα τελευταία πρέπει να θέσουμε στους καταχωρητές SCON και PCON τις κατάλληλες τιμές (050h και 80h αντίστοιχα). Όλα τα παραπάνω γίνονται στην υπορουτίνα 'INIT' του κώδικα.

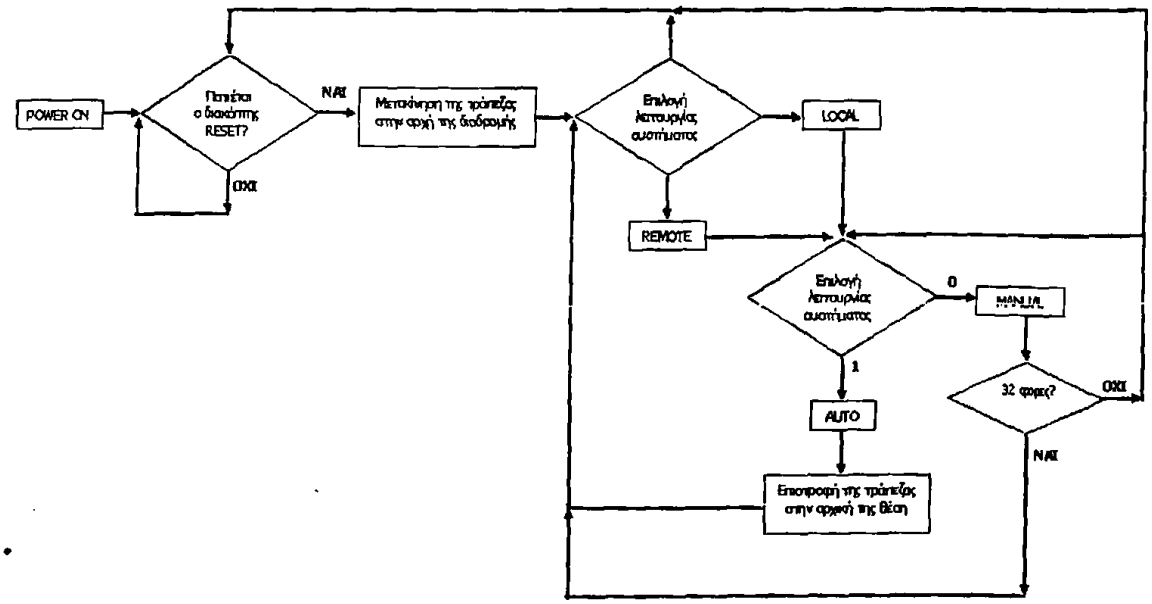
Με την αρχικοποίηση των παραπάνω καταχωρητών, ο μικροελεγκτής θέτει στους ακροδέκτες P1.0 έως P1.4 που συνδέονται με το ολοκληρωμένο L297 τις κατάλληλες τιμές που καθορίζουν τις παραμέτρους κίνησης της τράπεζας. Έτσι λοιπόν, για την ενεργοποίηση του κινητήρα θέτει τον ακροδέκτη P1.0, που συνδέεται με την είσοδο Enable, σε λογικό ένα. Επειδή θέλουμε ο κινητήρας μας να κάνει ολόκληρα βήματα, θέτουμε τον ακροδέκτη P1.3( Half/Full) στην τιμή μηδέν. Με την ανάλογη τιμή στον ακροδέκτη P1.1 επιλέγουμε τη διεύθυνση κίνησης της τράπεζας. Αφού έχουμε θέσει τους ακροδέκτες του L297 στις τιμές που καθορίζουν την κίνηση της τράπεζας, αναλαμβάνει μια άλλη υπορουτίνα, η 'DOSTEP', να παράγει παλμούς ρολογιού στον ακροδέκτη P1.2 που συνδέεται με τον ακροδέκτη CLOCK του L297.

Για τη σωστή λειτουργία της οθόνης LCD ακολουθεί μια υπορουτίνα που στην ουσία κάνει αρχικοποίηση. Η λογική αρχικοποίησης της οθόνης δίνεται αναλυτικά στο Παράρτημα "Ο προγραμματισμός της LCD οθόνης". Για να είμαστε βέβαιοι ότι δεν εμφανίζονται χαρακτήρες στην οθόνη από την αρχικοποίηση κάνουμε καθαρισμό αυτής, καλώντας την υπορουτίνα "CLEAR_LCD". Για το επόμενο χρονικό διάστημα των 5sec εμφανίζεται στην οθόνη το μήνυμα 'POWER ON' για να ενημερώσει τον χρήστη ότι το σύστημα έχει τεθεί σε λειτουργία.

Αναφέραμε στην εισαγωγή ότι η επιλογή του τρόπου λειτουργίας του συστήματος αλλά και κάθε ενέργειας αυτού γίνεται αρχικά μόνο από το χειριστήριο. Έτσι ο μικροελεγκτής ελέγχει αρχικά αν έχει πατηθεί κάποιος από τους διακόπτες και ανάλογα εκτελεί την αντίστοιχη εργασία. Αν λοιπόν πατηθεί ο διακόπτης RST, τότε

γίνεται αρχικοποίηση στην κίνηση της τράπεζας , ενώ με το πάτημα των διακοπτών LOCAL και REMOTE επιλέγουμε την αυτόνομη λειτουργία του συστήματος ή τη λειτουργία μέσω ηλεκτρονικού υπολογιστή. Ο διακόπτης λειτουργίας RUN και οι MANUAL, AUTO χρησιμοποιούνται μόνο στην περίπτωση της αυτόνομης λειτουργίας (LOCAL), αφού στην περίπτωση της REMOTE λειτουργίας η επιλογή των επόμενων εργασιών γίνεται με αποστολή χαρακτήρων μέσω της σειριακής θύρας. Ένα γενικό διάγραμμα ροής του κώδικα φαίνεται στο Σχήμα 3.2.1, ενώ ολόκληρος ο κώδικας παρατίθεται προς μελέτη στο Παράρτημα Γ.

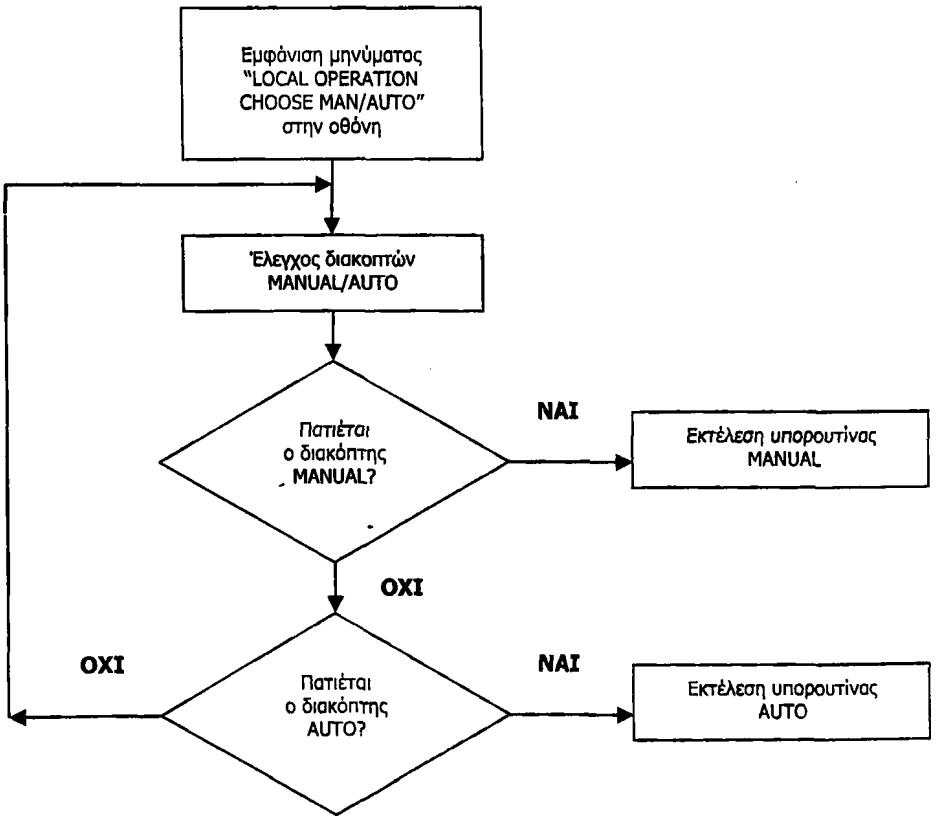
Στις παραγράφους που ακολουθούν αναλύονται χωριστά οι λειτουργίες LOCAL και REMOTE. Για την καλύτερη κατανόηση του κώδικα, στο τέλος κάθε παραγράφου ακολουθεί το αντίστοιχο διάγραμμα ροής κάθε υπορουτίνας.




**Σχήμα 3.2.1: Το γενικό διάγραμμα ροής του κώδικα**

## 3.3 Αυτόνομη λειτουργία (LOCAL OPERATION)

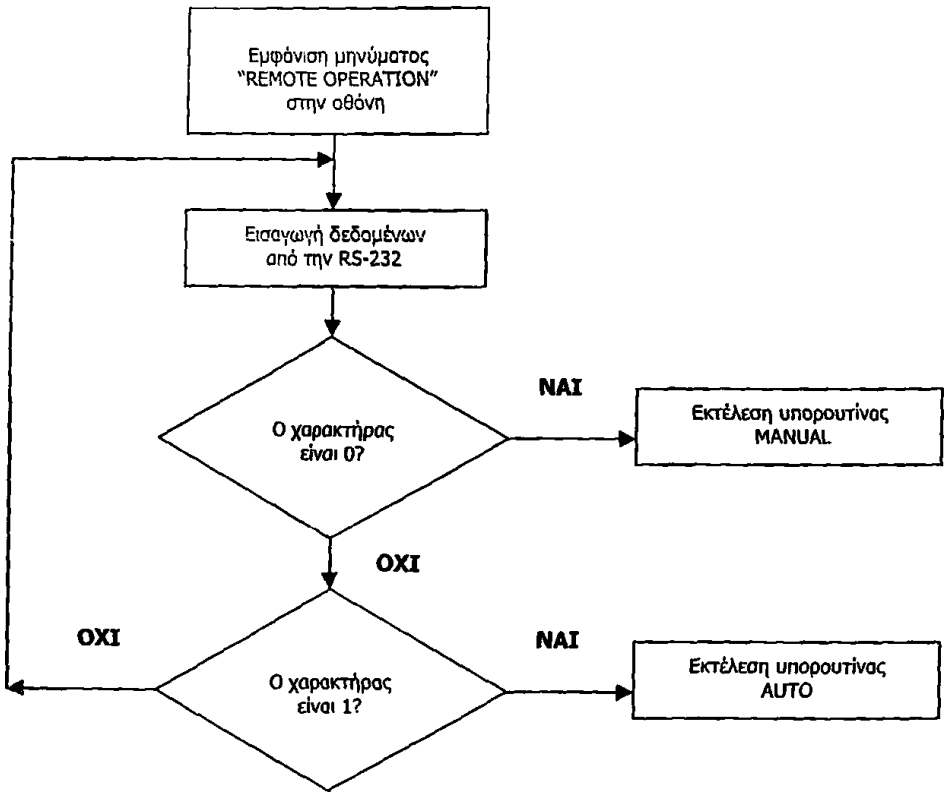
Με το πάτημα του διακόπτη LOCAL ο ακροδέκτης P2.5 του μικροελεγκτή, αλλάζει κατάσταση και ενεργοποιείται η εκτέλεση της υπορουτίνας 'LOCATE' από την αντίστοιχη θέση μνήμης. Το μήνυμα "LOCAL OPERATION" εμφανίζεται στην πρώτη γραμμή της οθόνης, ενώ η δεύτερη γραμμή με το μήνυμα "CHOOSE MAN/AUTO" προτρέπει τον χρήστη να επιλέξει μεταξύ των λειτουργιών MANUAL και AUTO, οι οποίες αναλύονται σε επόμενες παραγράφους. Παράλληλα φορτώνεται στον καταχωρητή R5 η τιμή 32d, αριθμός που δηλώνει το σύνολο των βημάτων που πρέπει να εκτελεστούν. Στη συνέχεια γίνεται έλεγχος αν έχει πατηθεί κάποιος από τους διακόπτες MANUAL ή AUTO οπότε η ροή του προγράμματος μεταφέρεται στην αντίστοιχη υπορουτίνα. Η ανάλυση αυτών γίνεται σε επόμενη παράγραφο. Το Σχήμα 3.3.1 αποτελεί το διάγραμμα ροής της υπορουτίνας 'LOCAL'.




**Σχήμα 3.3.1: Το διάγραμμα ροής της υπορουτίνας 'LOCAL'**

## 3.4 Λειτουργία μέσω Η/Υ (REMOTE OPERATION)

Όπως έχουμε ήδη αναφέρει στη λειτουργία REMOTE η επιλογή των επιμέρους λειτουργιών γίνεται με αποστολή χαρακτήρων από τον υπολογιστή στον μικροελεγκτή και συγκεκριμένα μέσα από την εφαρμογή του προγράμματος LabView η οποία θα αναλυθεί στο Κεφάλαιο 4. Όταν λοιπόν ο μικροελεγκτής λάβει τον χαρακτήρα '0' εκτελεί την υπορουτίνα MANUAL, ενώ με τη λήψη του χαρακτήρα '1' εκτελείται η υπορουτίνα AUTO. Σε περίπτωση λήψης κάποιου άλλου χαρακτήρα, εκτός από τους δυο προαναφερθέντες, ο μικροελεγκτής τον αγνοεί και περιμένει έως ότου λάβει το σωστό χαρακτήρα. Όπως και στην υπορουτίνα 'LOCAL' έτσι και εδώ φορτώνεται στον καταχωρητή R5 η τιμή 32d για να δηλώσει το σύνολο των βημάτων που θα εκτελεστούν. Το διάγραμμα ροής της υπορουτίνας 'REMOTE' απεικονίζεται στο Σχήμα 3.4.1 που ακολουθεί.




**Σχήμα 3.4.1: Το διάγραμμα ροής της υπορουτίνας 'REMOTE'**

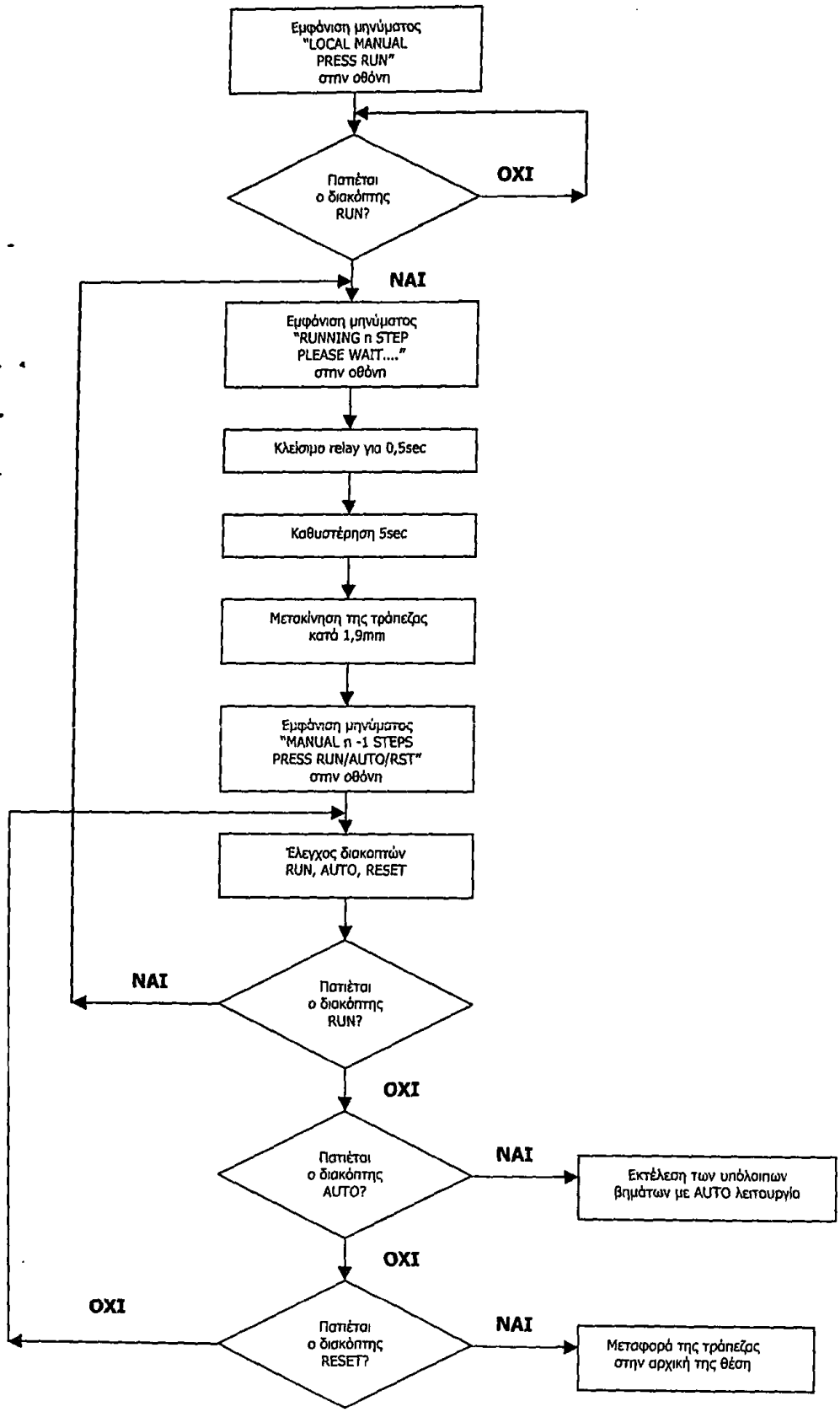
## 3.5 Λειτουργίες MANUAL και AUTO

Το σύστημα μας έχει τη δυνατότητα, με το πάτημα του διακόπτη λειτουργίας RUN να εκτελεί ένα βήμα (MANUAL) ή τριάντα δυο βήματα των 1,9mm συνολικά (AUTO), ανάλογα με τον τρόπο που έχει επιλεχθεί μέσα από τις υπορουτίνες 'LOCATE' και 'REMOTE'. Μέσα στον κώδικα έχει γίνει, επιπλέον, πρόβλεψη για συνδυασμό αυτών των τρόπων λειτουργίας, δηλαδή ένας αριθμός βημάτων να γίνεται με την υπορουτίνα AUTO και τα υπόλοιπα ένα-ένα χωριστά με χρήση της λειτουργίας MANUAL. Είναι δηλαδή εφικτό οποιαδήποτε στιγμή ο χρήστης να περάσει από λειτουργία MANUAL σε AUTO.

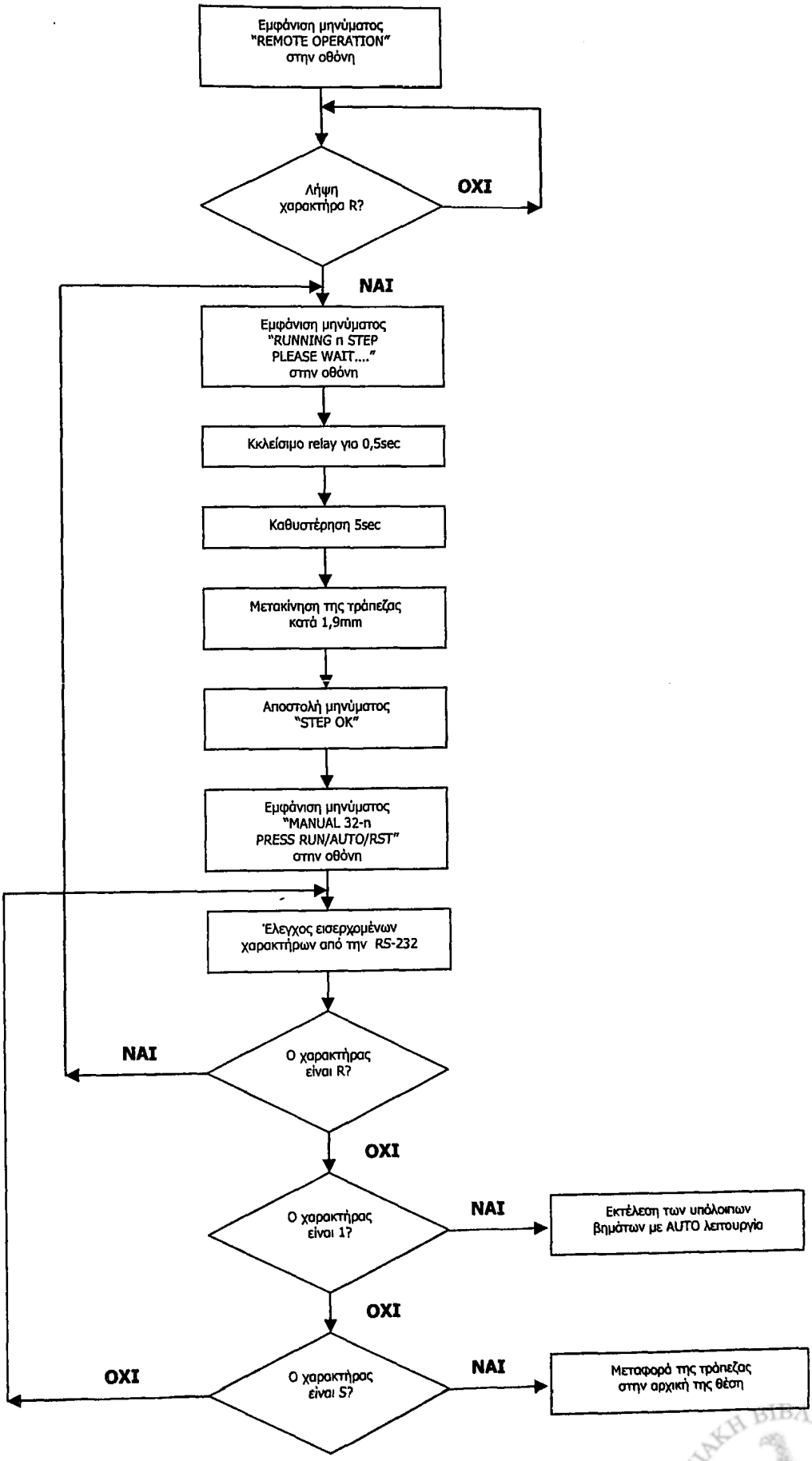
### 3.5.1 Λειτουργία MANUAL

Εάν υποθέσουμε ότι έχουμε επιλέξει την αυτόνομη λειτουργία και έχουμε πατήσει τον διακόπτη MANUAL για να κάνει το σύστημα μας ένα δοκιμαστικό βήμα. Μόλις πατηθεί ο διακόπτης εμφανίζεται το μήνυμα "LOCAL/MANUAL" στην πρώτη γραμμή της οθόνης, ενώ το μήνυμα "PRESS RUN" στη δεύτερη γραμμή, καθοδηγεί το χρήστη να πατήσει το διακόπτη λειτουργίας RUN. Με το κλείσιμο του παραπάνω διακόπτη εμφανίζεται το μήνυμα "RUNNING n STEP" "PLEASE WAIT...", όπου n ο αριθμός του τρέχοντος βήματος. Στη συνέχεια κλείνει το relay για 0,5s και αφού εκτελεστεί η προβλεπόμενη καθυστέρηση των 5sec η τράπεζα μετακινείται κατά 1,9mm ( υπορουτίνα 'METAKINISI') όποτε μειώνεται κατά ένα και ο αριθμός των εναπομενόντων προς εκτέλεση βημάτων. Μετά την εκτέλεση του δοκιμαστικού βήματος το μήνυμα "MANUAL n-1 STEPS" "PRESS RUN/AUTO/RST" δίνει στον χρήστη τη δυνατότητα επιλογής ανάμεσα στους διακόπτες RUN, AUTO ή RESET για την εκτέλεση του επόμενου βήματος, ή των υπολοίπων βημάτων μέσω της AUTO λειτουργίας, ενώ έχει την δυνατότητα να μεταφέρει την τράπεζα στην αρχική της θέση με το πάτημα του διακόπτη RESET.

Η ίδια διαδικασία επαναλαμβάνεται και στην περίπτωση της REMOTE λειτουργίας, με τη διαφορά ότι αντί για το πάτημα του διακόπτη RUN ο μικροελεγκτής θα περιμένει να λάβει τον χαρακτήρα 'R' για την εκτέλεση του επόμενου βήματος, τον χαρακτήρα '1' για να περάσει από MANUAL λειτουργία σε AUTO και τον χαρακτήρα 'S' για μεταφορά της τράπεζας στην αρχική της θέση. Πρέπει να σημειώσουμε ότι στη λειτουργία REMOTE, μετά από κάθε βήμα, ο μικροελεγκτής στέλνει στον υπολογιστή το μήνυμα 'STEP OK' (υπορουτίνα SEND_STEPOK). Το διάγραμμα ροής της υπορουτινας MANUAL τόσο σε LOCAL όσο και σε REMOTE λειτουργία απεικονίζεται στα Σχήματα 3.5.1.1 και 3.5.1.2



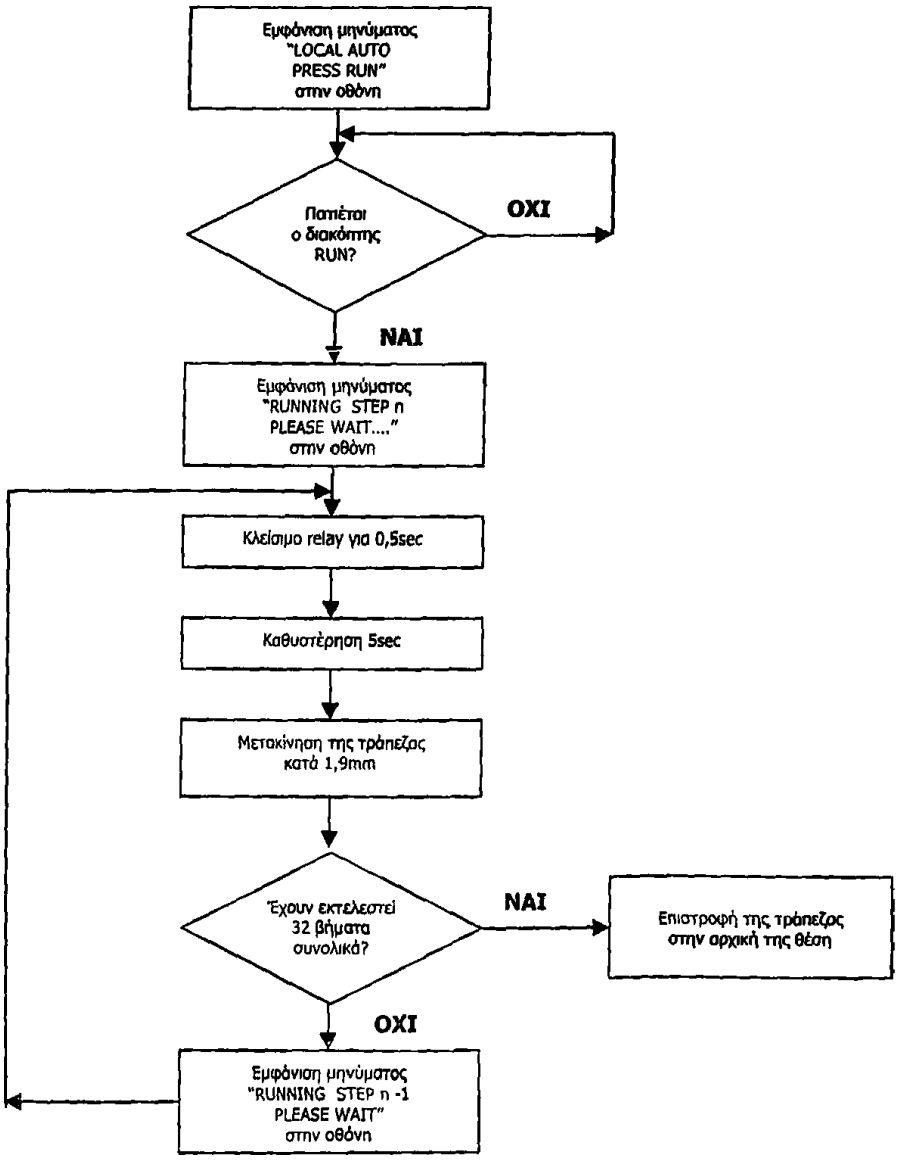

**Σχήμα 3.5.1.1: Το διάγραμμα ροής της υπορουτίνας MANUAL σε λειτουργία LOCAL**



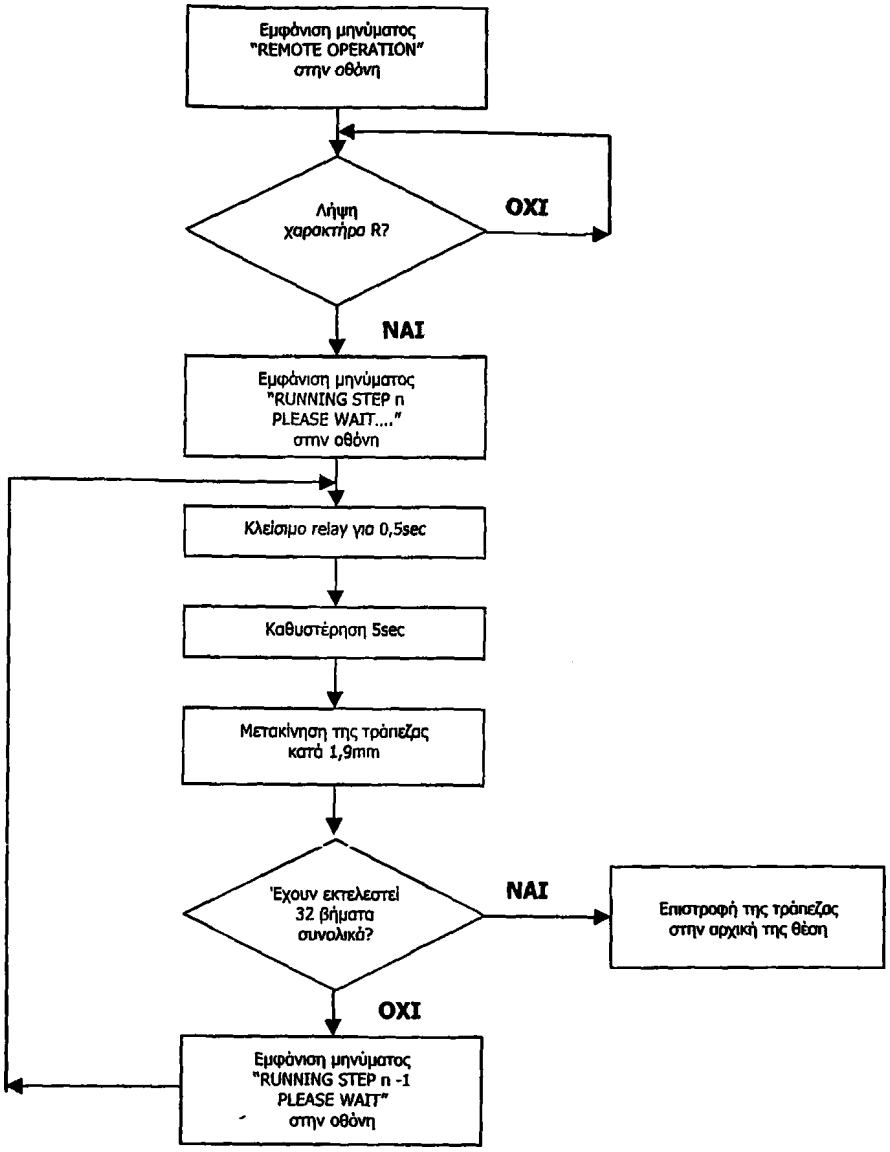

**Σχήμα 3.5.1.2.: Το διάγραμμα ροής της υπορουτίνας MANUAL σε λειτουργία REMOTE**

### 3.5.2. Λειτουργία AUTO

Η υπορουτίνα AUTO εκτελεί ακριβώς την ίδια λειτουργία με την υπορουτίνα MANUAL, αλλά τριάντα δυο φορές. Στην περίπτωση που έχουμε επιλέξει τη λειτουργία REMOTE, μετά την εκτέλεση και των 32 βημάτων ο μικροελεγκτής στέλνει στον υπολογιστή το μήνυμα 'OK!' (υπορουτίνα SEND_OK), ενώ ο ακροδέκτης P1.1 τίθεται στην τιμή 0, ώστε η τράπεζα να αλλάξει φορά κίνησης και να επιστρέψει στο σημείο από το οποίο ξεκίνησε, πριν το πάτημα του διακόπτη λειτουργίας RUN, με το κάλεσμα της υπορουτίνας 'EPISTROFH'. Στα Σχήματα 3.5.2.1 και 3.5.2.2 που ακολουθούν παρουσιάζεται το διάγραμμα ροής της τρέχουσας υπορουτινας.




**Σχήμα 3.5.2.1: Το διάγραμμα ροής της υπορουτίνας AUTO σε λειτουργία LOCAL**




**Σχήμα 3.5.2.2: Το διάγραμμα ροής της υπορουτίνας AUTO σε λειτουργία REMOTE**

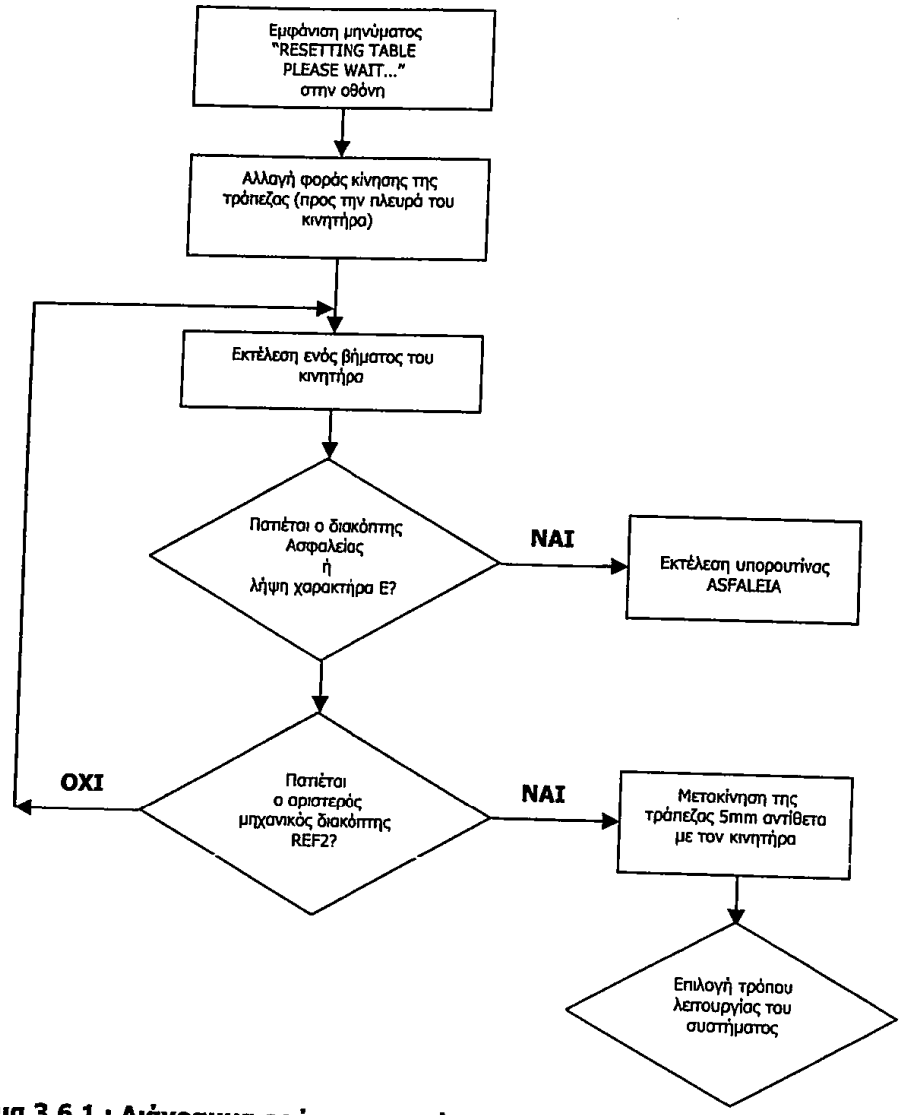
Η μετακίνηση που πρέπει να κάνει η τράπεζα κάθε φορά πρέπει να είναι ακριβώς 1,9mm. Έχουμε επιλέξει ο κινητήρας μας να κάνει πλήρη βήματα και από τα χαρακτηριστικά λειτουργίας του γνωρίζουμε ότι ένα πλήρες βήμα μετακινεί την τράπεζα κατά 1μm. Επομένως για μετατόπιση 1,9mm απαιτούνται συνολικά 1900 βήματα. Η υπορουτίνα 'METAKINISI' που καλείται σε όλους τους τρόπους λειτουργίας παρέχει αυτή τη μετατόπιση. Για λόγους ασφάλειας μετά από κάθε βήμα του κινητήρα (του 1μm) γίνεται έλεγχος για το αν έχει πατηθεί ο διακόπτης ασφάλειας (LOCAL) ή έχει σταλθεί ο χαρακτήρας 'E' (REMOTE), όποτε καλείται η υπορουτίνα 'ASFALEIA' στην οποία ο κινητήρας απενεργοποιείται με μηδενισμό του ακροδέκτη P1.0 (είσοδος ENABLE του L297) και το μήνυμα 'SYSTEM HALTED' εμφανίζεται στην οθόνη. Στο τέλος εκτέλεσης της υπορουτίνας 'ASFALEIA' η ροή του προγράμματος μεταφέρεται στην υπορουτίνα 'CHOOSE' όπου γίνεται η επιλογή του τρόπου λειτουργίας του συστήματος.

Για αποφυγή καταστροφής της τράπεζας, στην περίπτωση που φτάσει στη δεξιά τερματική θέση της διαδρομής της, μετά από κάθε βήμα του κινητήρα, εκτός

από τον διακόπτη ασφαλείας, ελέγχεται αν έχει κλείσει ο οριακός μηχανικός διακόπτης REF1. Η οθόνη μας ενημερώνει για τη θέση της τράπεζας με την εμφάνιση του μηνύματος 'REF1' σε αυτή, ενώ σταματά οποιαδήποτε κίνηση με απενεργοποίηση του κινητήρα. Όλα τα παραπάνω γίνονται στην υπορουτίνα 'SWRIGHT'. Στη συνέχεια γίνεται αρχικοποίηση στην κίνηση της τράπεζας με την υπορουτίνα 'RESETTING_TABLE' που αναλύεται στην επόμενη παράγραφο.

## 3.6 Αρχικοποίηση στη κίνηση της τράπεζας

Όταν η τράπεζα φτάσει στο δεξιότερο άκρο της διαδρομής της και κλείσει ο διακόπτης REF1 σταματά οποιαδήποτε κίνηση αυτής και απενεργοποιείται ο κινητήρας. Η τράπεζα θα πρέπει να επιστρέψει στην αρχική της θέση. Το σημείο αυτό έχει επιλεχθεί κατά βούληση να είναι 5mm μετά τον αριστερό μηχανικό οριακό διακόπτη REF2. Η αρχικοποίηση της κίνησης της τράπεζας γίνεται με το κλείσιμο του διακόπτη RST του χειριστηρίου της συσκευής στην αυτόνομη λειτουργία ( δεν πρέπει να συγχέεται με το διακόπτη που κάνει reset στο πρόγραμμα που είναι φορτωμένο στη μνήμη του μικροελεγκτη) ή με τη λήψη του χαρακτήρα 'S' στη λειτουργία μέσω υπολογιστή. Κατά τη διάρκεια αυτής της διαδικασίας, η τράπεζα κινείται προς τα αριστερά έως ότου συναντήσει τον διακόπτη REF2, αλλάζει φορά κίνησης και αφού μετακινηθεί κατά 5mm προς τα δεξιά σταματά, ενώ η οθόνη μας πληροφορεί για τη εργασία που εκτελείται. Στη συνέχεια, η ροή του προγράμματος μεταφέρεται στην υπορουτίνα 'CHOOSE' όπου ο χρήστης επιλέγει τον τρόπο λειτουργίας του συστήματος. Παράλληλα του δίνεται η δυνατότητα να κάνει αρχικοποίηση στην κίνηση της τράπεζας εάν βέβαια απαιτείται. Και στην υπορουτίνα αρχικοποίησης πάντα μετά από κάθε βήμα του κινητήρα ελέγχεται ο διακόπτης ασφάλειας. Δυνατότητα αρχικοποίησης της τράπεζας έχουμε και μετά από κάθε βήμα των 1,9mm που κάνει αυτή σε λειτουργία MANUAL, καθώς επίσης και στην έναρξη της εφαρμογής μας, πριν ακόμη επιλέξουμε τρόπο λειτουργίας του συστήματος. Το διάγραμμα ροής αυτής της υπορουτίνας δίνεται στο Σχήμα 3.6.1.

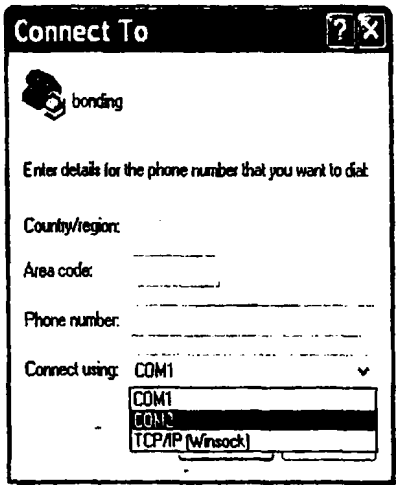



**Σχήμα 3.6.1.: Διάγραμμα ροής υπορουτίνας 'RESETTING_TABLE'.**

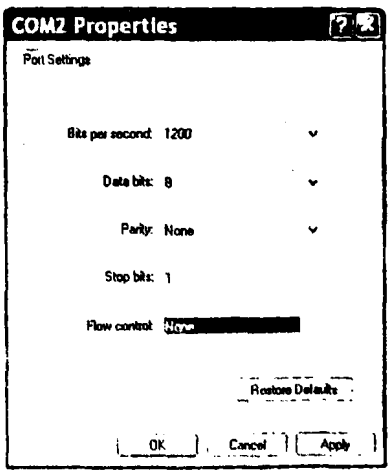
## 3.7 Έλεγχος του κώδικα μέσω Hyper Terminal

Για τον έλεγχο του κώδικα σε λειτουργία REMOTE χρησιμοποιήσαμε ένα πρόγραμμα προσομοίωσης ( π.χ. Hyper Terminal), του οποίου η κύρια εργασία είναι η επικοινωνία μέσω της σειριακής θύρας.

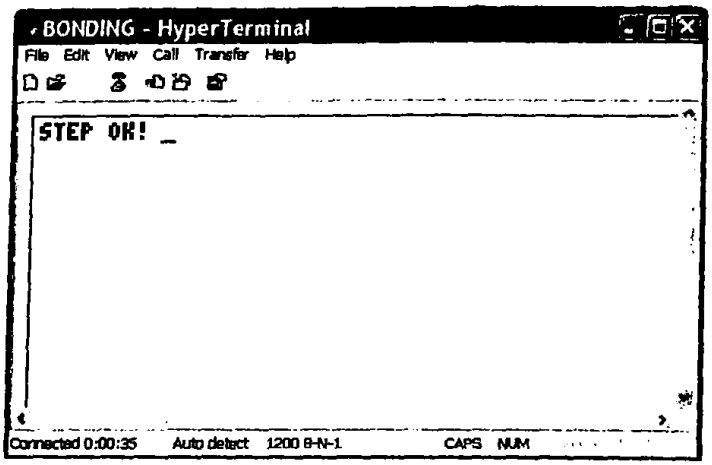
Για να συνεργαστεί η συσκευή με το Hyper Terminal ο χρήστης δεν έχει παρά να ξεκινήσει το πρόγραμμα και να επιλέξει απευθείας επικοινωνία με τη σειριακή θύρα Com (Com1 Com2) του υπολογιστή στην οποία έχει συνδεθεί εξωτερικά η συσκευή (Εικόνα 3.7.1). Στη συνέχεια, στα χαρακτηριστικά σύνδεσης πρέπει να επιλεγεί ο ρυθμός μεταφοράς δεδομένων (1200bps), το μήκος δεδομένων 8-bit χωρίς χρήση ισοτιμίας και χωρίς χρήση ελέγχου ροής των δεδομένων, όπως παρουσιάζεται στην εικόνα 3.7.2. Στιγμιότυπο της λειτουργίας του προγράμματος Hyper Terminal αποτελεί η εικόνα 3.7.3.




**Εικόνα 3.7.1: Επιλογή σειριακής θύρας επικοινωνίας**




**Εικόνα 3.7.2: Επιλογή χαρακτηριστικών σύνδεσης συστήματος Η/Υ μέσω του προγράμματος Hyper Terminal**




**Εικόνα 3.7.3: Στιγμιότυπο λειτουργίας του προγράμματος Hyper Terminal**

# Κεφάλαιο 4ο

# Η Διεπαφή LabView

## 4.1 Εισαγωγή

Κάθε σύστημα ελεγχόμενο από Η/Υ απαιτεί ένα λογισμικό υποστήριξης από το οποίο θα πραγματοποιούνται οι ρυθμίσεις του συστήματος. Για την ανάπτυξη της παρούσας εφαρμογής χρησιμοποιήθηκε το πακέτο λογισμικού LabView της εταιρείας National Instruments.

Το LabView (Laboratory Virtual Instruments Electronic Workbench) είναι ένα περιβάλλον γραφικού προγραμματισμού, ιδιαίτερα δημοφιλές στον χώρο των ηλεκτρικών μετρήσεων, που βασίζεται στη γλώσσα γραφικού προγραμματισμού "G". Αντί για γραμμές κώδικα χρησιμοποιούνται έτοιμες συναρτήσεις υπό μορφή αντικειμένων και η διασύνδεση μεταξύ τους ορίζεται με τη βοήθεια αγωγών, με αποτέλεσμα το πρόγραμμα να μοιάζει περισσότερο σαν ένα πρόγραμμα βαθμίδων. Τα προγράμματα LabView καλούνται "εικονικά όργανα"( Virtual instruments:VI) γιατί η εμφάνιση και η λειτουργία τους μιμείται τα πραγματικά όργανα.

Το τμήμα με το οποίο αλληλεπιδρά ο χρήστης ονομάζεται πρόσοψη (front panel). Σε αυτό παρουσιάζονται περιστροφικά κουμπιά, διακόπτες, οθόνες και άλλα στοιχεία ελέγχου και ενδείξεων. Ο χρήστης εισάγει τα δεδομένα χρησιμοποιώντας το πληκτρολόγιο ή το ποντίκι και στη συνέχεια βλέπει τα αποτελέσματα στην οθόνη.

Τα εικονικά όργανα δέχονται οδηγίες από ένα block diagram που σχεδιάζεται σε γλώσσα γραφικού προγραμματισμού G. Το block diagram παρέχει μια γραφική λύση στο πρόβλημα προγραμματισμού. Στην ουσία παρέχει τον πηγαίο κώδικα για το εικονικό όργανο. [5],[6]

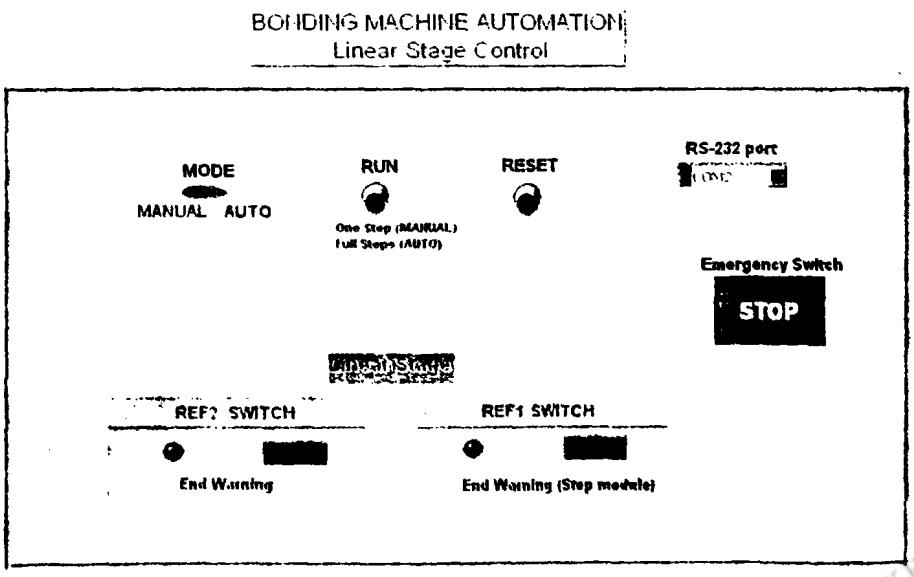
Στην πρόσοψη (front panel) του εικονικού οργάνου της εφαρμογής διακρίνονται τα εξής πεδία εισαγωγής δεδομένων:

- RS-232 port: πεδίο επιλογής σειριακής θύρας επικοινωνίας(COM1,COM2 κτλ)
- MODE: πεδίο επιλογής λειτουργίας (MANUAL, AUTO) του συστήματος
- RUN: διακόπτης λειτουργίας
- RESET: διακόπτης αρχικοποίησης της τράπεζας
- EMERGENCY SWITCH: διακόπτης ασφαλείας

Τα πεδία εξόδου είναι τα εξής :

- REF1 SWITCH: δείκτης μηχανικού διακόπτη REF1 της τράπεζας και μήνυμα ανάγνωσης ανάλογα με την κατάσταση στην οποία αυτός βρίσκεται. Και οι δυο δείκτες ενεργοποιούνται με το κλείσιμο του συγκεκριμένου διακόπτη.
- REF2 SWITCH: δείκτης μηχανικού διακόπτη REF1 της τράπεζας και μήνυμα ανάγνωσης. Η λειτουργία τους είναι ανάλογη με αυτή των δεικτών του διακόπτη REF1.

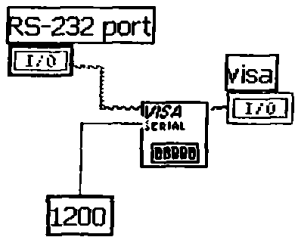
Υπάρχει επίσης και ένα επιπλέον πεδίο εξόδου, μη ορατό από το χρήστη, το οποίο χρησιμοποιείται για αποσφαλμάτωση του προγράμματος (debugging). Το πεδίο αυτό είναι το "error out" που δηλώνει το είδος και τον κωδικό του σφάλματος. Πρόσβαση στο πεδίο αυτό έχουμε εάν μεταφερθούμε στο block diagram του εικονικού οργάνου και με πάτημα του δεξιού πλήκτρου του ποντικιού στο αντίστοιχο σύμβολο, επιλέξουμε 'Show Indicator'.




**Εικόνα 4.1.1: Η πρόσοψη (front panel) ελέγχου του συστήματός μας**

## 4.2 Το διάγραμμα λειτουργίας της εφαρμογής

Η σειριακή θύρα που χρησιμοποιείται για την επικοινωνία δηλώνεται στο πεδίο "RS-232 port". Πρώτα γίνεται αρχικοποίηση των παραμέτρων της θύρας. Στην προκειμένη περίπτωση, επιλέχθηκαν οι εξ ορισμού παράμετροι (μήκος δεδομένων 8 bit, χωρίς ισοτιμία και χωρίς έλεγχο ροής των δεδομένων) εκτός από τον ρυθμό μετάδοσης τον οποίο επιλέξαμε να είναι 1200bps. Η υλοποίηση αυτής της λογικής έχει δημιουργηθεί στον κώδικα LabView με τη μορφή που παρουσιάζεται παρακάτω.

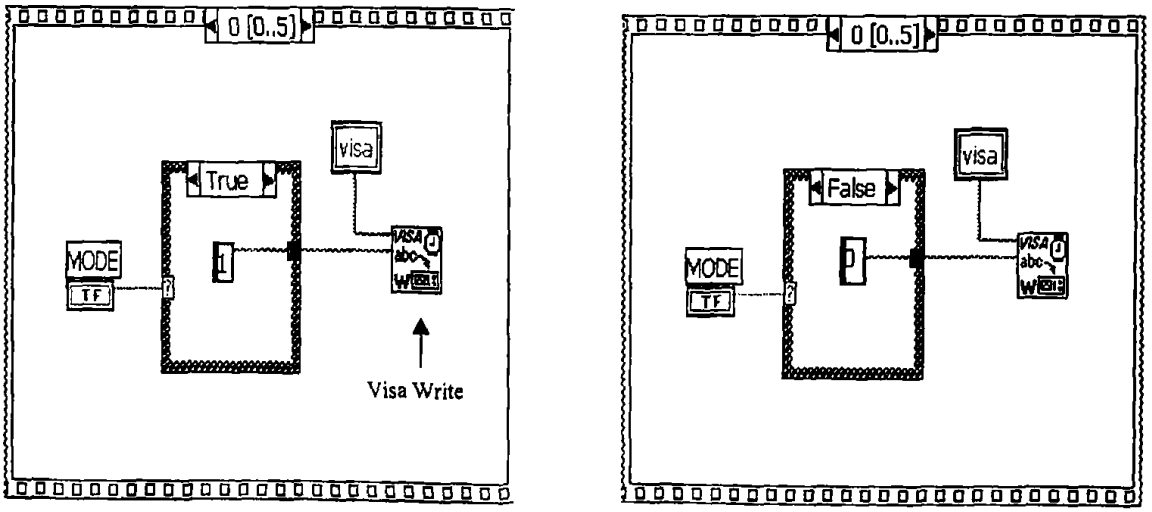



**Σχήμα 4.2.1: Το διάγραμμα λειτουργίας αρχικοποίησης της θύρας επικοινωνίας με τον Η/Υ**

Στη συνέχεια εκτελείται μια ακολουθία (που δηλώνεται με το εργαλείο "Sequence Structure") τα βήματα της οποίας είναι τα ακόλουθα:

### Βήμα 1: Επιλογή τρόπου λειτουργίας του συστήματος

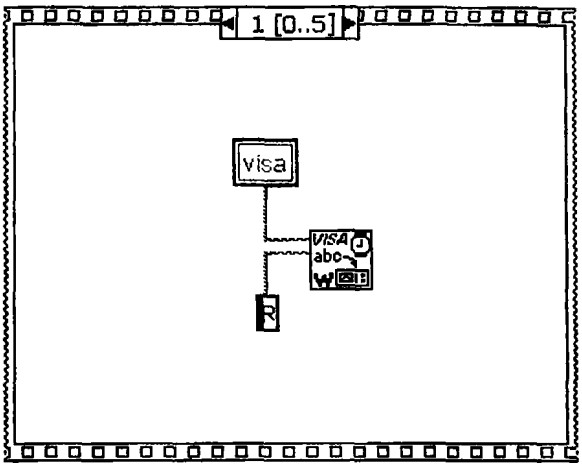
Ο τρόπος λειτουργίας του συστήματος επιλέγεται από τον διακόπτη **MODE.** Από το διακόπτη αυτό, 'Slide Switch' όπως καλείται, ο χρήστης έχει τη δυνατότητα επιλογής ανάμεσα σε MANUAL ή AUTO. Όπως έχει ήδη αναφερθεί στο προηγούμενο κεφάλαιο για την επιλογή αυτή αποστέλλονται δυο χαρακτήρες οι '0' και '1'. Για την ανάγνωση των χαρακτήρων γίνεται χρήση της συνάρτησης "Visa Write".




**Σχήμα 4.2.2: Το διάγραμμα λειτουργίας του Vi για το βήμα 1**

### Βήμα 2: Έλεγχος διακόπτη λειτουργίας RUN

Στο δεύτερο βήμα της συνάρτησης "Sequence Structure" και πάλι με χρήση της συνάρτησης "Visa Write" γίνεται ανάγνωση του χαρακτήρα 'R'.



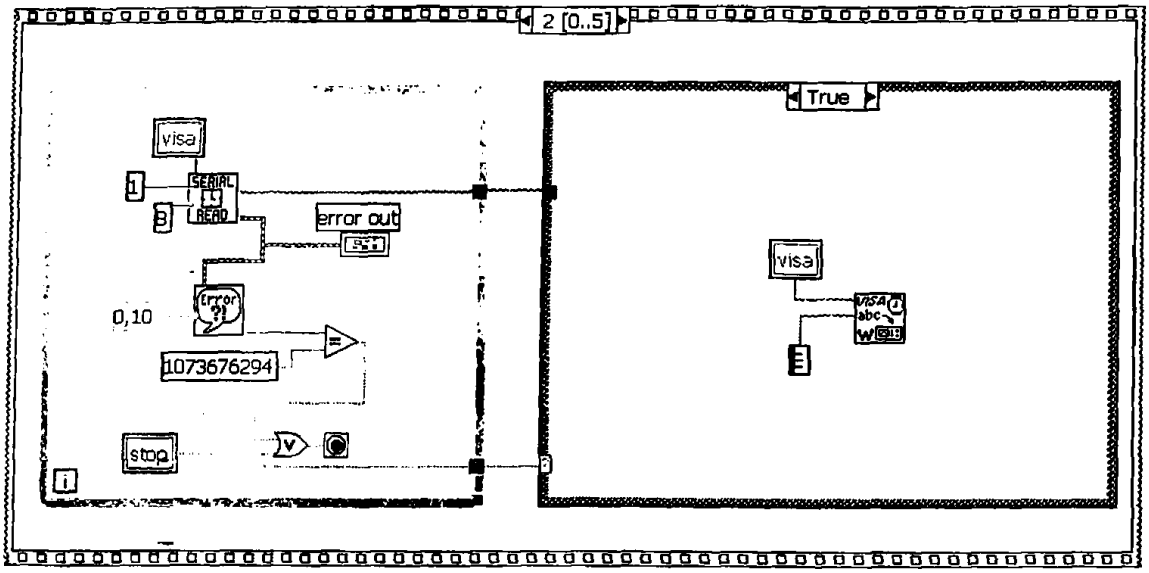

**Σχήμα 4.2.3: Το διάγραμμα λειτουργίας του Vi για το βήμα 2**

### Βήμα 3: Έλεγχος διακόπτη ασφαλείας και μηχανικού οριακού διακόπτη REF1

Αναφέραμε στο προηγούμενο κεφάλαιο, στη μελέτη του κώδικα, ότι, πάντα μετά από κάθε βήμα του κινητήρα γίνεται έλεγχος :

- αν έχει πατηθεί ο διακόπτης ασφαλείας
- έχει κλείσει ο διακόπτης REF1 της τράπεζας.

Για να επιτευχθεί αυτό στον κώδικα LabView, στο τρίτο βήμα της συνάρτησης "Sequence Structure" έχει χρησιμοποιηθεί μία συνάρτηση "While loop" η οποία σταματά να εκτελείται μόνο όταν πατηθεί ο διακόπτης ασφαλείας, ή ο υπολογιστής δεχθεί από τον μικροελεγκτή, μέσω της σειριακής, ένα μήνυμα ως ένδειξη ότι έχει πατηθεί ο διακόπτης REF1 της τράπεζας. Εάν πράγματι συμβεί ένα από τα παραπάνω, θα εκτελεστεί η συνάρτηση "Case Structure" στο δεξιό τμήμα του παραθύρου, στην είσοδο της οποίας έχει συνδεθεί ο διακόπτης ασφαλείας. Στην περίπτωση που έχει πατηθεί ο διακόπτης θα έχει γίνει ανάγνωση του χαρακτήρα 'E' (Σχήμα 4.2.4).

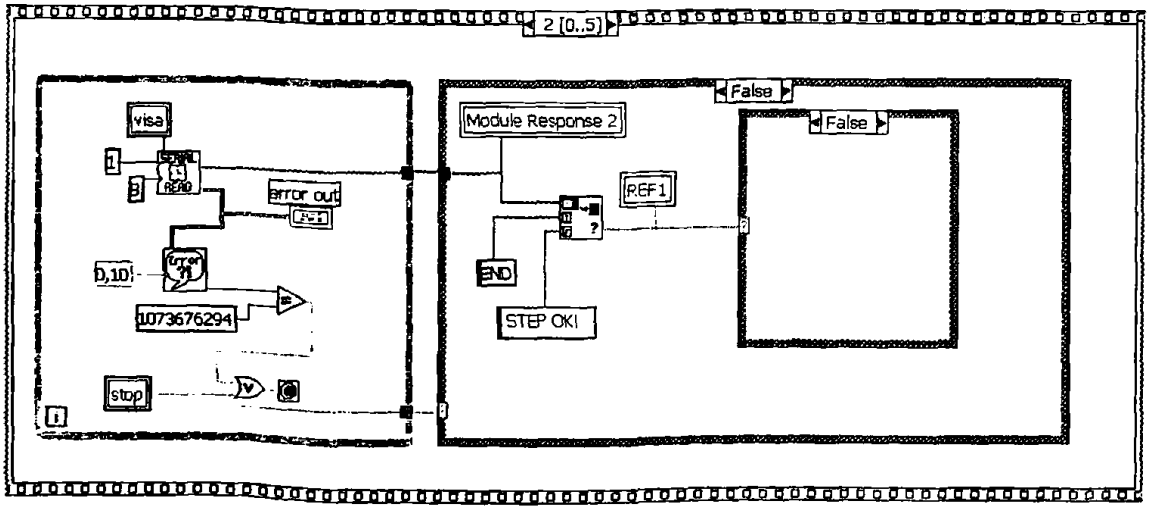



**Σχήμα 4.2.4: Το διάγραμμα ροής του Vi στην περίπτωση που έχει πατηθεί ο διακόπτης ασφαλείας**

Σε περίπτωση, που έχει κλείσει ο διακόπτης REF1, μέσω της συνάρτησης "Match True/False string":

- θα εμφανιστεί στην οθόνη του "REF1 SWITCH", στο front panel, το μήνυμα "END" και
- θα ανάψει ο δείκτης REF1 που συνδέεται στην έξοδο Selection της συνάρτησης "Match True/False string".

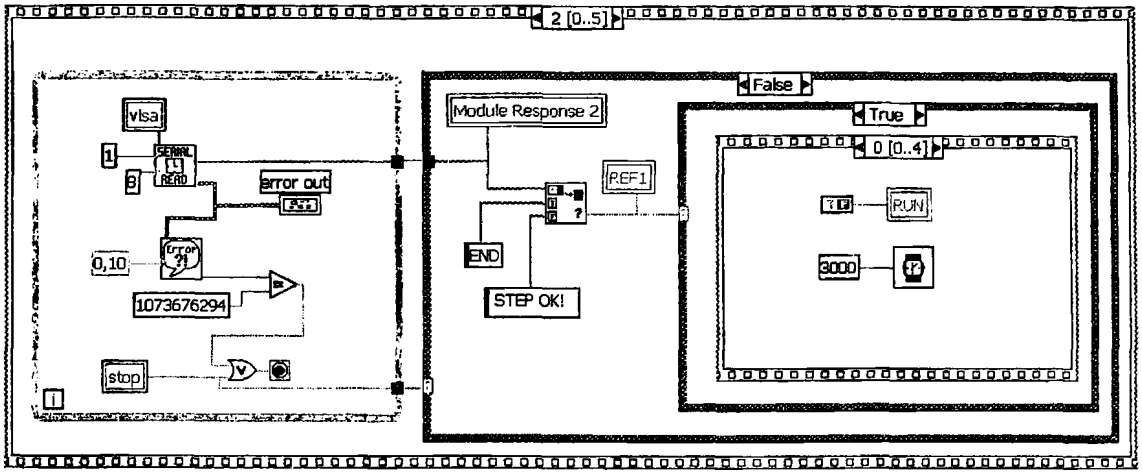
Εάν δεν συμβεί καμία από τις δυο παραπάνω περιπτώσεις, κάθε φορά που εκτελείται ένα βήμα των 1,9mm, στην ίδια οθόνη θα εμφανίζεται το μήνυμα "STEP OK!" και ο κώδικας θα συνεχιστεί με το τέταρτο βήμα της συνάρτησης "Sequence Loop"(Σχήμα 4.2.5)




**Σχήμα 4.2.5: Το διάγραμμα ροής του VI για το βήμα 3**

Ας υποθέσουμε ότι η τράπεζα κατά την κίνησή της έχει φτάσει στο δεξιό άκρο και έχει κλείσει ο διακόπτης REF1. Στην περίπτωση αυτή, όπως φαίνεται και στο Σχήμα 4.2.4, θα εκτελεστεί μια δεύτερη συνάρτηση "Case Structure" στην είσοδο της

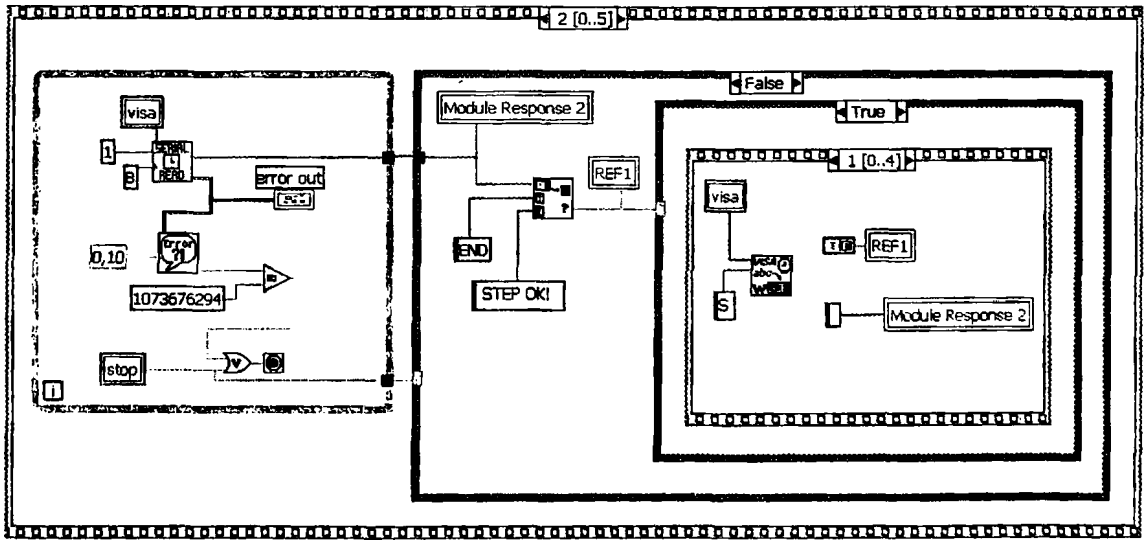
οποίας συνδέεται ο δείκτης REF1. Αν πράγματι ανάψει ο δείκτης, τότε θα εκτελεστεί η συνάρτηση "Sequence Loop" που εσωκλείεται σε αυτή (Σχήμα 4.2.6). Στο πρώτο βήμα της συνάρτησης γίνεται αρχικοποίηση του διακόπτη λειτουργίας RUN, για επόμενη χρήση, ενώ η αναμονή των 3sec που παρεμβάλλεται είναι ο χρόνος που ο δείκτης REF1 παραμένει φωτεινός και εμφανίζεται στην οθόνη το μήνυμα "END".




**Σχήμα 5.2.6: Το διάγραμμα ροής του VI στην περίπτωση που έχει κλείσει ο διακόπτης REF1.**

Στο δεύτερο βήμα προκειμένου να γίνει αρχικοποίηση της τράπεζας:

- αποστέλλεται ο χαρακτήρας "S"
- σβήνει ο δείκτης REF1 και καθαρίζει η αντίστοιχη οθόνη (Σχήμα 4.2.7).



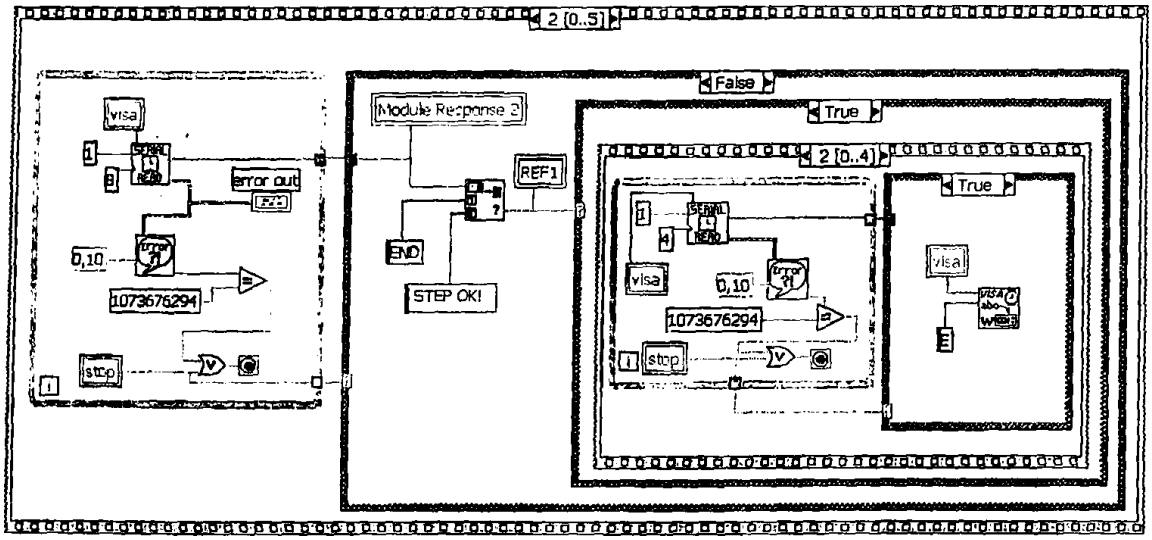

**Σχήμα 4.2.7: Το διάγραμμα ροής του VI για την αρχικοποίηση της τράπεζας**

Στο τρίτο βήμα, και πάλι με χρήση της συνάρτησης "While Loop", ελέγχεται:

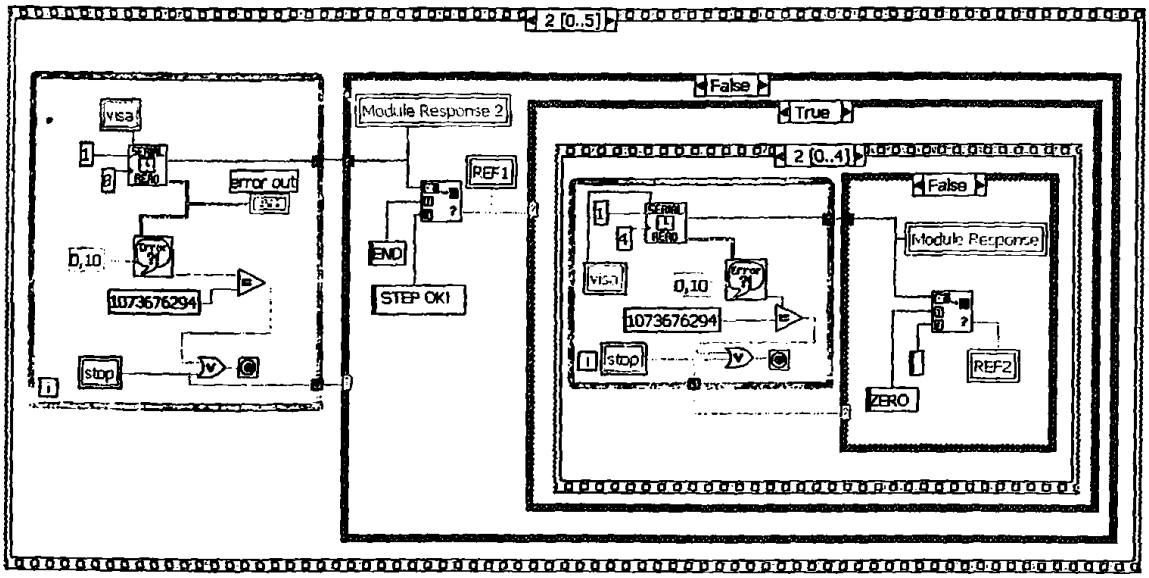
- εάν έχει πατηθεί ο διακόπτης ασφαλείας κατά τη διάρκεια της αρχικοποίησης
- ή έχει σταλεί κάποιο μήνυμα στον Η/Υ ως ένδειξη της θέσης της τράπεζας.

Εάν κάποια από τις δύο παραπάνω περιπτώσεις συμβεί, τότε η συνάρτηση "While Loop" σταματά να εκτελείται. Η συνάρτηση "Case Structure", στο δεξιό τμήμα του

παραθύρου, εκτελεί ακριβώς την ίδια λειτουργία με αυτή που αναλύσαμε παραπάνω. Έτσι λοιπόν, εάν πράγματι έχει πατηθεί ο διακόπτης ασφαλείας, θα έχουμε αποστολή του χαρακτήρα "E" (Σχήμα 4.2.8), ενώ εάν έχει κλείσει ο αριστερός, μηχανικός διακόπτης REF2, στην αντίστοιχη οθόνη στο front panel, εμφανίζεται το μήνυμα "ZERO" και ανάβει ο δείκτης REF2 (Σχήμα 4.2.9).

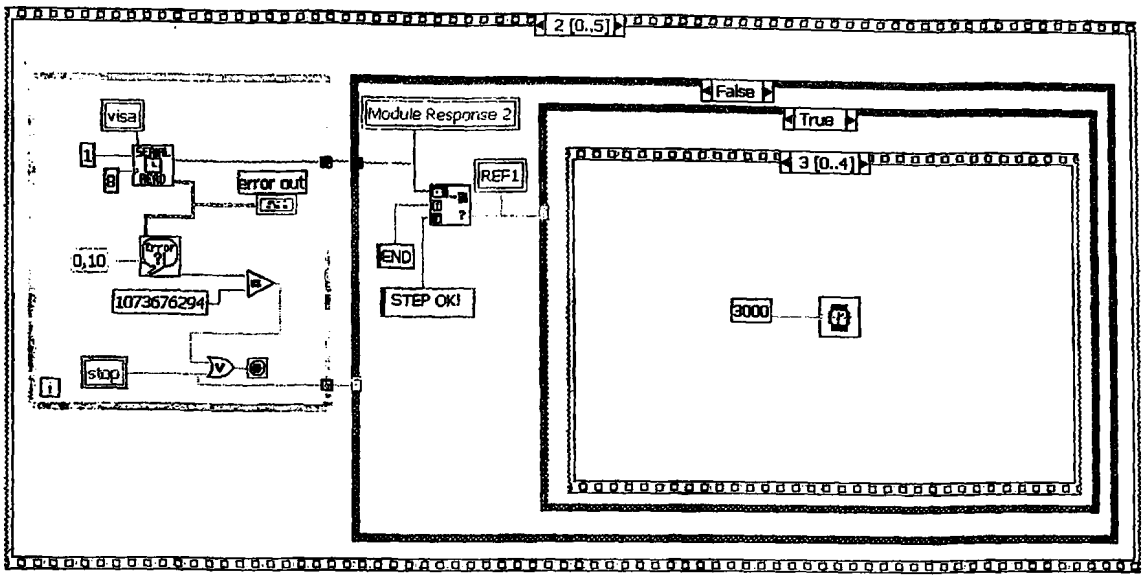



**Σχήμα 4.2.8: Έλεγχος του διακόπτη ασφαλείας κατά τη διάρκεια της αρχικοποίησης της τράπεζας**



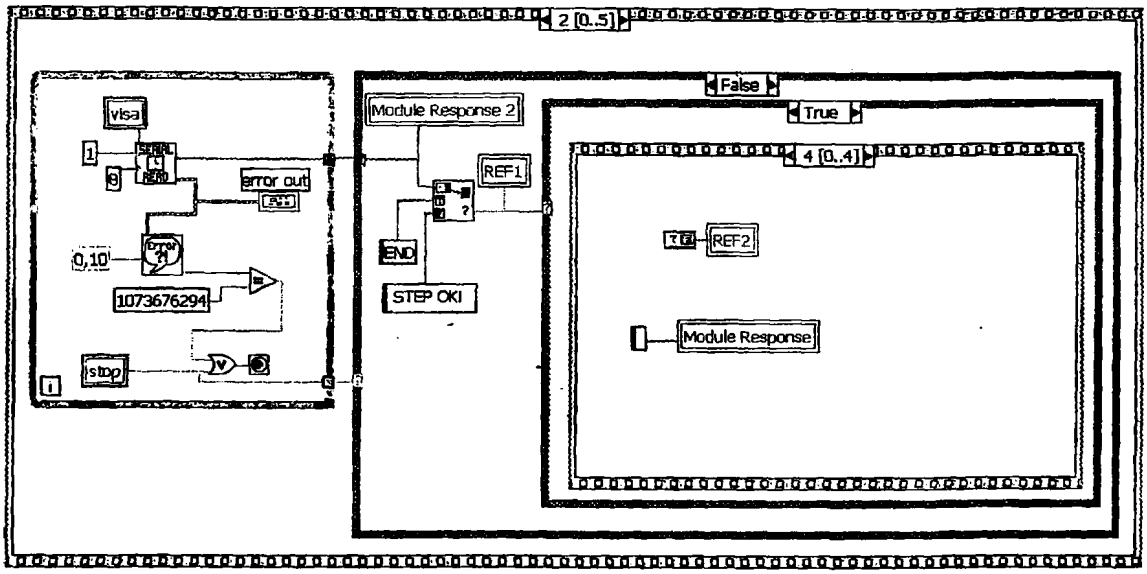

**Σχήμα 4.2.9: Το διάγραμμα ροής του VI στην περίπτωση που έχει κλείσει ο διακόπτης REF2.**

Επειδή το κλείσιμο του διακόπτη REF2 είναι στιγμιαίο, στο τέταρτο βήμα της συνάρτησης "Sequence Loop" παρεμβάλλεται χρόνος αναμονής 3sec, χρόνος που ο δείκτης θα παραμείνει φωτεινός και το μήνυμα "ZERO" θα εμφανίζεται στην οθόνη (Σχήμα 4.2.10).




**Σχήμα 4.2.10: Αναμονή 3000msec**

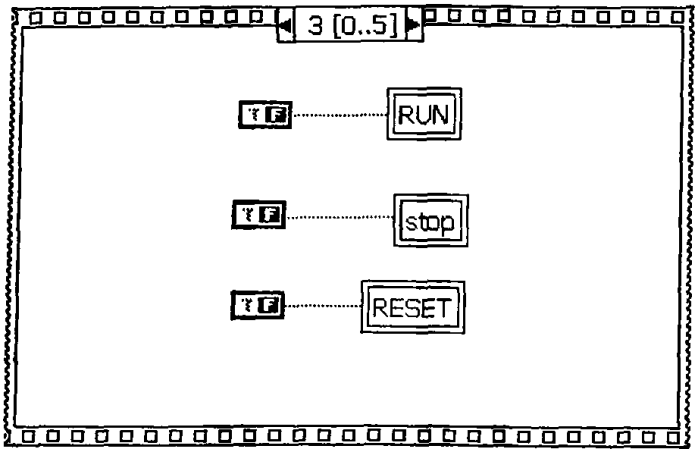
Στο τελευταίο βήμα της συνάρτησης γίνεται καθαρισμός της οθόνης που εμφανίζεται το μήνυμα "ZERO" και σβήσιμο του δείκτη REF2 (Σχήμα 4.2.11).




**Σχήμα 4.2.11: Καθαρισμός της οθόνης και σβήσιμο του δείκτη REF2**

## Βήμα 4: Αρχικοποίηση διακοπτών

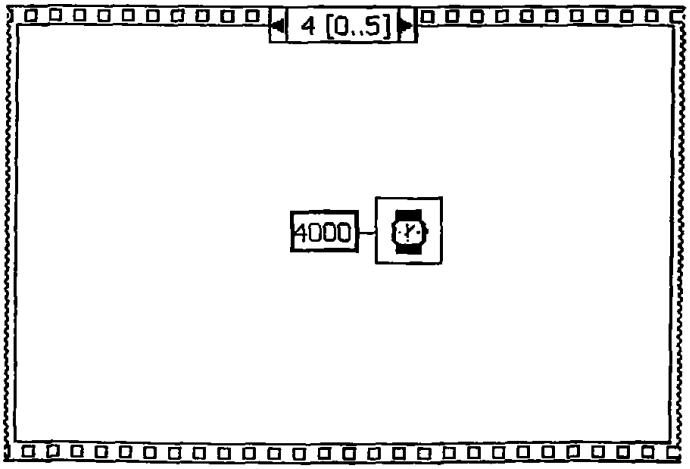
Όταν βρισκόμαστε σε λειτουργία MANUAL, έχει ήδη αναφερθεί αρκετές φορές, ότι για την εκτέλεση ενός βήματος απαιτείται το πάτημα του διακόπτη λειτουργίας RUN. Για να γίνει αυτό εφικτό στον κώδικα LabView πρέπει μετά από κάθε βήμα να γίνεται αρχικοποίηση στον διακόπτη RUN, ώστε να επιστρέφει στην αρχική του κατάσταση. Η υλοποίηση αυτής της λειτουργίας γίνεται στο τέταρτο βήμα της "Sequence Structure". Στο ίδιο βήμα γίνεται επίσης αρχικοποίηση του διακόπτη ασφαλείας και του διακόπτη RESET σε περίπτωση που έχουν πατηθεί κατά την εκτέλεση προηγούμενου βήματος (Σχήμα 4.2.12).




**Σχήμα 4.2.12: Αρχικοποίηση διακοπτών**

### Βήμα 5: Αναμονή 4000 msec

Το πέμπτο βήμα δεν είναι παρά μια καθυστέρηση 4sec απαραίτητη για την εκτέλεση του επόμενου βήματος. Πιο αναλυτικά μέσω του πέμπτου βήματος, δίνουμε στον μικροελεγκτή το χρόνο να διαβάσει τους επόμενους χαρακτήρες που θα σταλούν με το πάτημα κάποιου διακόπτη από την πρόσοψη του εικονικού οργάνου της εφαρμογής μας. (Σχήμα 4.2.13).

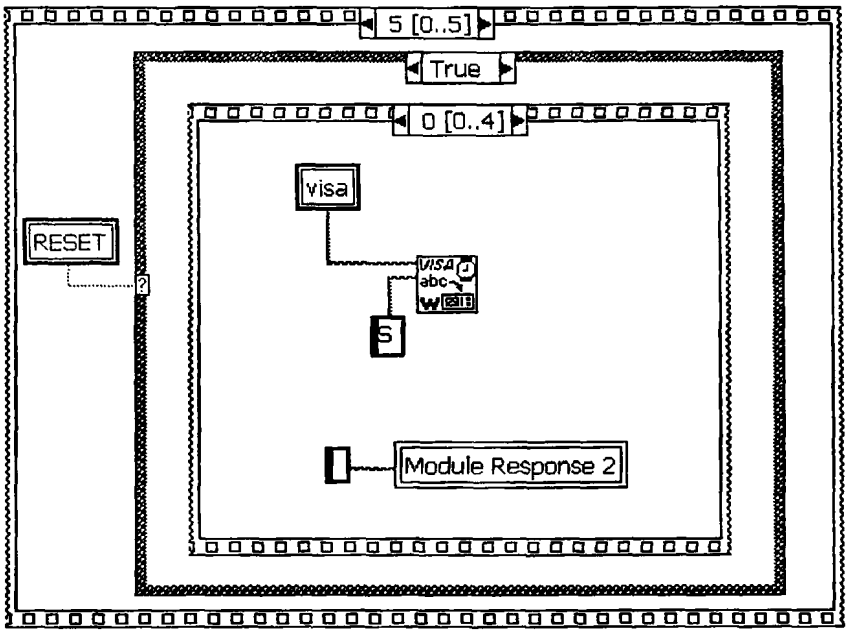



**Σχήμα 4.2.13: Αναμονή 4000msec**

### Βήμα 6: Έλεγχος διακοπτών RESET, MODE

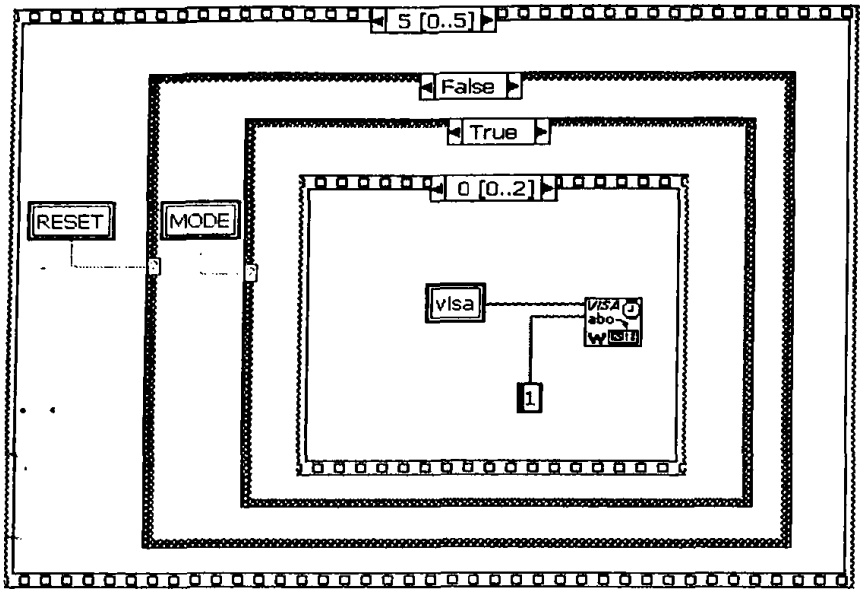
Στη λειτουργία MANUAL, πάντα μετά την εκτέλεση ενός βήματος, δίνεται στο χρήστη η δυνατότητα επιλογής της επόμενης λειτουργίας που θα εκτελεστεί, ανάλογα με το διακόπτη που αυτός θα πατήσει. Η επιλογή αυτή γίνεται εφικτή μέσω του βήματος 6.

Ο διακόπτης RESET συνδέεται στην είσοδο μιας συνάρτησης "Case Structure". Εάν αυτός πατηθεί τότε γίνεται αρχικοποίηση στην κίνηση της τράπεζας με τη βοήθεια της συνάρτησης "Sequence Structure" που περικλείεται σε αυτή, τα βήματα της οποίας έχουν ήδη αναλυθεί στο Βήμα 2 (Σχήματα 4.2.7 έως 4.2.11).

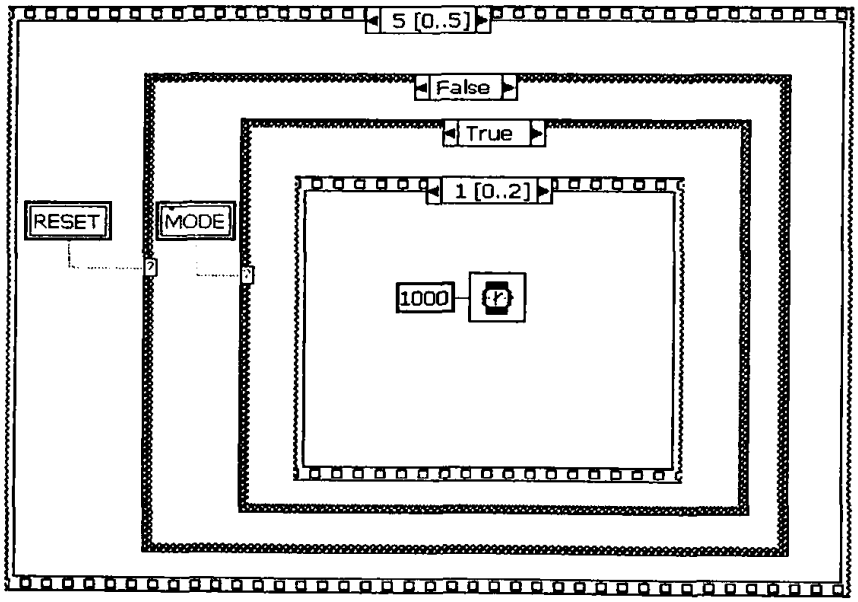



**Σχήμα 4.2.14: Αποστολή χαρακτήρα "S" για την αρχικοποίηση της τράπεζας**

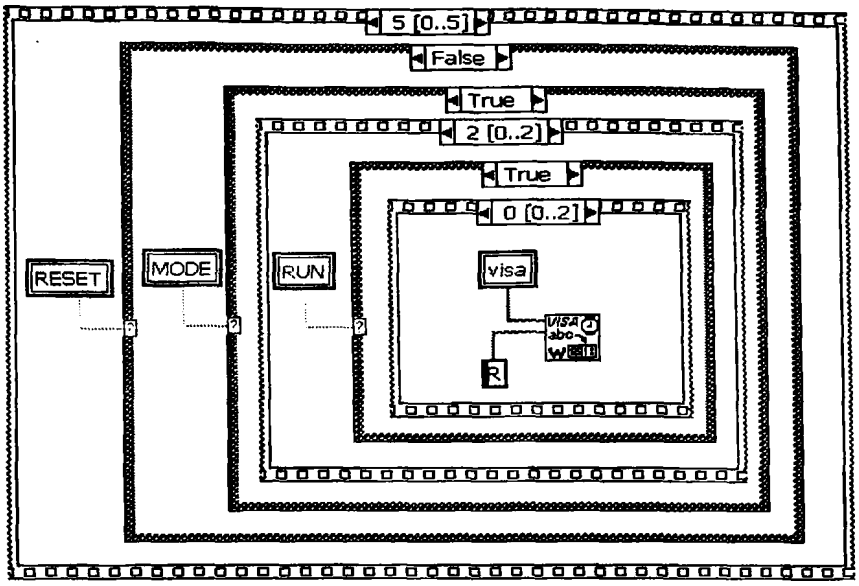
Στην περίπτωση που δεν έχει πατηθεί ο διακόπτης RESET, ελέγχεται εάν έχει αλλάξει κατάσταση ο διακόπτης MODE στο front panel (MANUAL → AUTO), με την σύνδεση και αυτού του διακόπτη στην είσοδο μιας συνάρτησης "Case Structure". Εάν βρίσκεται στη θέση AUTO, τότε έχει αποσταλεί ο χαρακτήρας "1" (Σχήμα 4.2.15) και μετά από χρόνο 1sec γίνεται έλεγχος του διακόπτη λειτουργίας RUN με τη σύνδεση αυτού στην είσοδο μιας "Case Structure" συνάρτησης. Εάν πράγματι έχει πατηθεί ο διακόπτης λειτουργίας, αποστέλλεται ο χαρακτήρας "R" (Σχήμα 4.2.17), οπότε η ροή του προγράμματος μεταφέρεται στο βήμα 3. Σε αντίθετη περίπτωση, ο κώδικας εκτελείται από την αρχή. Θα πρέπει να σημειωθεί ότι ο χρόνος αναμονής του 1sec (Σχήμα 4.2.16) σε αυτό το σημείο του κώδικα, είναι απαραίτητος ώστε να διαβάσει η εφαρμογή τον χαρακτήρα "R".




**Σχήμα 4.2.15: Αποστόλη του χαρακτήρα "1" για αλλαγή λειτουργίας από MANUAL σε AUTO**




**Σχήμα 4.2.16: Αναμονή 1000msec**




**Σχήμα 4.2.17: Αποστολή χαρακτήρα "R" για την εκτέλεση των βημάτων μέσω της λειτουργίας AUTO**

Ολόκληρη η λογική του κώδικα LabView εσωκλείεται μέσα σε μία συνάρτηση "While Loop" ούτως ώστε να εκτελείται διαρκώς (ατέρμων βρόχος). Επίσης στον κώδικα αυτόν υπάρχει μια επιπρόσθετη "Case Structure", στην είσοδο της οποίας συνδέεται ο διακόπτης λειτουργίας RUN που καθορίζει αν θα εκτελεστεί η εφαρμογή ή όχι ανάλογα με την κατάσταση του προαναφερθέντος διακόπτη. Έξω από τη συνάρτηση "While Loop", εκτός από την αρχικοποίηση της θύρας που αναφέραμε στην αρχή της παραγράφου, γίνεται αρχικοποίηση σε όλους τους διακόπτες και δείκτες που χρησιμοποιούνται στην εφαρμογή.

# Κεφάλαιο 5ο

# Κατασκευή και έλεγχος του συστήματος

## 5.1 Κατασκευή τυπωμένων κυκλωμάτων

Με τη βοήθεια του λογισμικού πακέτου Orcad Layout και με βάση το σχηματικό διάγραμμα που παρουσιάστηκε στο δεύτερο κεφάλαιο, σχεδιάστηκαν τα τυπωμένα κυκλώματα PCB της κύριας πλακέτας και του χειριστηρίου του συστήματος τα οποία παρατίθενται στο Παράρτημα Ε.

Ιδιαίτερη προσοχή δόθηκε στο πάχος των γραμμών τροφοδοσίας, στις γειώσεις και στις γραμμές του κυκλώματος ισχύος (μεταξύ των εξόδων του ολοκληρωμένου κυκλώματος L298 και του κινητήρα) γιατί πρέπει να αντέξουν ρεύμα έντασης 1Α.

Το επόμενο στάδιο ήταν η εμφάνιση των τυπωμένων κυκλωμάτων με τη φωτοχημική μέθοδο στις αντίστοιχες εγκαταστάσεις του Εργαστηρίου ΦΥΕ όπου παρέχονται τα κατάλληλα εργαλεία και τηρούνται οι απαραίτητες συνθήκες ασφαλείας.

Επόμενη φάση πριν την ολοκλήρωση, ήταν το τρύπημα των πλακετών, η συναρμολόγηση των εξαρτημάτων, ο οπτικός έλεγχος των κολλήσεων και ο ηλεκτρικός έλεγχος των συνδέσεων.

Φωτογραφία της κύριας πλακέτας μετά την ολοκλήρωση της κατασκευής της δίνεται στο παράρτημα Ε.

## 5.2 Έλεγχος του συστήματος

Μετά την κατασκευή των πλακετών ακολούθησε ο έλεγχος του συστήματος για την διαπίστωση της καλής λειτουργίας του.

Αρχικά μετρήθηκε με το πολύμετρο εάν όλοι οι ακροδέκτες τροφοδοσίας των ολοκληρωμένων συνδέονται με τις εξόδους των σταθεροποιητών τάσεων.

Στην συνέχεια έγινε ο προγραμματισμός του μικροελεγκτή AT89S8252 και η τοποθέτησή του πάνω στην κύρια πλακέτα. Επόμενο βήμα ήταν η σύνδεση της οθόνης LCD και της πλακέτας του χειριστηρίου στους ακροδέκτες J6 και J1 αντίστοιχα της κύριας πλακέτας, ενώ ο Η/Υ και ο βηματικός κινητήρας συνδέθηκαν στους ακροδέκτες P2 και P1.

Επόμενο βήμα ήταν η παροχή τάσης στο κύκλωμα με τη χρήση ενός τροφοδοτικού συνεχούς ρεύματος. Με την παροχή τάσης μετρήθηκαν με το πολύμετρο εκ νέου οι τιμές των τάσεων στους ακροδέκτες τροφοδοσίας των ολοκληρωμένων και στις εξόδους των σταθεροποιητών τάσης και βρέθηκαν όλες σωστές (δηλαδή μέσα στα όρια λειτουργίας των ολοκληρωμένων). Τέλος ρυθμίστηκε η τάση του ποτενσιόμετρου R4 που συνδέεται στον ακροδέκτη Vee της οθόνης LCD, ώστε να επιτευχθεί η καλύτερη δυνατή αντίθεση εικόνας. Με τον ίδιο τρόπο ρυθμίστηκε και η τάση του ποτενσιόμετρου R7 που συνδέεται στον ακροδέκτη $V_{ref}$ του ολοκληρωμένου κυκλώματος L297, ώστε να είναι ίση με την τάση στα άκρα των αντιστάσεων R1, R2 , R3 και R4.

Στο επόμενο στάδιο ελέγχθηκε η λειτουργία του συστήματος με χρήση του χειριστηρίου του. Πιο αναλυτικά, επιλέχθηκε τόσο η χειροκίνητη λειτουργία όσο και η αυτόνομη, αλλά και συνδυασμός των δύο λειτουργιών. Δηλαδή στην αρχή μέσω της χειροκίνητης λειτουργίας έγιναν κάποια δοκιμαστικά βήματα και τα υπόλοιπα όλα μαζί, με αλλαγή της λειτουργίας σε αυτόματη.

Μετά τον έλεγχο της αυτόνομης λειτουργίας, σειρά είχε η λειτουργία μέσω του Η/Υ. Σε πρώτη φάση έγινε χρήση του προγράμματος προσομοίωσης τερματικού, Hyper Terminal, του οποίου κυρία εργασία είναι η επικοινωνία μέσω της σειριακής θύρας. Στόχος μας ήταν ο έλεγχος της καλής επικοινωνίας του υπολογιστή με το μικροελεγκτή, αλλά και η ανίχνευση λαθών στον κώδικα όσον αφορά την αποστολή και τη λήψη χαρακτήρων. Έτσι λοιπόν με τη χρήση του παραπάνω προγράμματος σειριακής επικοινωνίας έγινε απευθείας επικοινωνία με τον μικροελεγκτή και παρατηρήθηκε η σωστή απόκριση του. Ακολούθως έγινε έλεγχος της λειτουργίας μέσω του Η/Υ, με την εφαρμογή που αναπτύχθηκε στο περιβάλλον προγραμματισμού LabView. Και με αυτή την εφαρμογή διαπιστώθηκε η σωστή λειτουργία του συστήματος. Το σύστημα τέθηκε σε λειτουργία, χωρίς παύση, για 6 ώρες δύο συνεχείς ημέρες (6 ώρες κάθε μέρα), ώστε να διαπιστωθεί εάν θα υπάρξουν τυχόν δυσλειτουργίες και βλάβες κατά την χρησιμοποίησή του για μεγάλο χρονικό διάστημα. Διαπιστώθηκαν λοιπόν τα παρακάτω:

1. Η καλή λειτουργία του συστήματος.
2. Η σωστή επικοινωνία του μικροελεγκτή με τον υπολογιστή.

3. Ο σωστός προγραμματισμός του μικροελεγκτή.

Κατά τη διάρκεια των δοκιμών, παρατηρήθηκε ότι το ολοκληρωμένο κύκλωμα L298 και οι σταθεροποιητές τάσης LM7805 και LM7806 θερμαίνονταν αρκετά. Προκείμενου να μειωθεί η θερμοκρασία τους, στερεώθηκαν σε αυτά ψήκτρες για την απορρόφηση της θερμότητας που εκπέμπουν.

## 5.3 Ολοκλήρωση του συστήματος

Προκείμενου η συσκευή να μην είναι ευάλωτη σε δυσμενείς συνθήκες και εύκολη στη χρήση τοποθετήθηκε στο εσωτερικό ενός πλαστικού κουτιού. Στη θέση του τροφοδοτικού συνδέθηκε ένας μετασχηματιστής 9V και 1A, έτσι ώστε το σύστημα, κάθε χρονική στιγμή να τροφοδοτείται με εναλλασσόμενη τάση 220V. Μετά τη σύνδεση του μετασχηματιστή, με ένα πολύμετρο μετρήθηκαν οι τάσεις στις εξόδους των γεφυρών ανόρθωσης και των σταθεροποιητών τάσης, και διαπιστώθηκε οτι είναι όλες σωστές.

Με τη χρήση του προγράμματος CorelDraw σχεδιάστηκε η πρόσοψη και οι πλαϊνές επιφάνειες του κουτιού. Με τη βοήθεια του σχηματικού χαράχτηκαν τα σημεία τοποθέτησης του χειριστηρίου της συσκευής, της οθόνης LCD του διακόπτη ασφαλείας, της ένδειξης μικροσυγκόλλησης και των ακροδεκτών συνδέσεις των υπόλοιπων στοιχείων του συστήματος (Η/Υ, βηματικός κινητήρας, συσκευής μικροσυγκόλλησης, καλώδιο τροφοδοσίας). Με ιδιαίτερη προσοχή κόπηκαν και αφαιρέθηκαν τα τμήματα του πλαστικού και στη θέση του συγκολλήθηκαν όλα τα παραπάνω στοιχεία. Φωτογραφίες των ακροδεκτών, αλλά και της πρόσοψης του κουτιού, παρατίθενται στο Παράρτημα Ε.

Στη συνέχεια τοποθετήθηκε στο εσωτερικό του κουτιού η κεντρική πλακέτα μαζί με το μετασχηματιστή και έγιναν οι απαραίτητες συνδέσεις αυτής με την οθόνη LCD, το χειριστήριο και την ένδειξη μικροσυγκόλλησης.

Για την αποφυγή υπερβολικής θέρμανσης του εσωτερικού του κουτιού και την πιθανή αλλοίωση των καλωδίων σύνδεσης από την εκπομπή θερμότητας του ολοκληρωμένου L298 και των σταθεροποιητών τάσης, τοποθετήθηκε στη μια πλαϊνή όψη του κουτιού ένας μικρός ανέμιστήρας. Ο ρόλος του είναι η απομάκρυνση του θερμού αέρα από τον εσωτερικό χώρο από ένα σύνολο τρυπών που έγιναν πάνω στον κουτί. Για την ανακύκλωση του αέρα, τρύπες έγιναν και στην πλευρά του κουτιού ακριβώς απέναντι από τον ανεμιστήρα.

Με το κλείσιμο του κουτιού, το σύστημα τέθηκε ξανά σε λειτουργία για χρονικό διάστημα 5 ωρών. Επιβεβαιώθηκε με αυτόν τον τρόπο η σωστή λειτουργία του.

# Βιβλιογραφία-Αναφορές


[1] The 8051 Microcontroller
Architecture, Programming and Applications-Kenneth J. Ayala

[2] Προγραμματίζοντας τον μικροελεγκτή 8051-Myke Pedro
Εκδόσεις Τζιόλα

[3] Σημειώσεις για το εργαστήριο « Μικροϋπολογιστές ΙΙ, Μέρος Ι»
Τσαγκουριάς Νικόλαος

[4] LabView 6i User Manual
National Instruments

[5] LabView 6i Student Edition
National Instruments

[6] LabView For Everyone, Second Edition - Jeffrey Travis
National Instruments

[7] Ανάπτυξη συστήματος ελέγχου κινητής τράπεζας ακριβείας για τον έλεγχο ποιότητας μικρολωριδιακών ανιχνευτών πυριτίου.- Α.Ασημίδης

[8] Microchip Technology Inc.
AN907 Stepping Motors Fundamentals, 2004

[9] SGS-Thomson Microelectronics
AN470 The L297 Stepper Motor Controller

**[10]** Stepping Motor Tutorial by Dr. Douglas W. Jones (http://www.cs.uiova.edu/~jones/step/#introduction). Copyright 1995

**[11]** Drive Circuits Basics, Industrial Circuits Applications Notes (http://library.solarbotics.net/pdflib/pdf/drive.pdf)

**[12]** http://www.8052.com

**[13]** http://www.eio.com/jasstep.htm

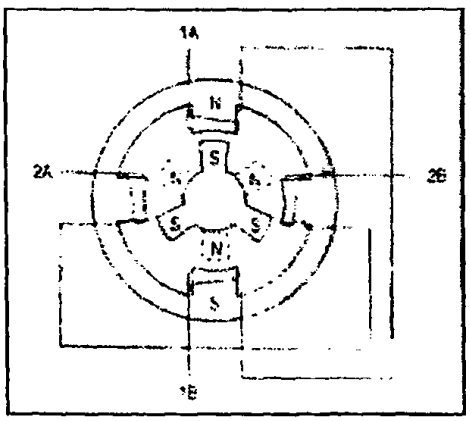
**[14]** http://www.anaheimautomation.com/intro.htm

# Παράρτημα Α

## Μελέτη βηματικών κινητήρων

### Α.1 Ανάλυση λειτουργίας βηματικών κινητήρων

Στο Σχήμα Α.1.1 απεικονίζεται η κάθετη τομή ενός σύγχρονου βηματικού κινητήρα. Ο στάτορας έχει 4 προεξέχοντες πόλους (ακίνητα τυλίγματα), ενώ ο ρότορας αποτελείται από τρεις μαγνήτες (6 πόλοι) και στις δύο περιπτώσεις κατασκευασμένοι από μαλακό χάλυβα. Δυο ζεύγη καλωδίων έχουν τοποθετηθεί στους πόλους του στάτορα. Κάθε ζεύγος συνδέει 2 τυλίγματα σε σειρά. Κάθε ζεύγος τυλιγμάτων αποκαλείται φάση, που σημαίνει ότι ο κινητήρας του παραδείγματος είναι ένας διφασικός κινητήρας.

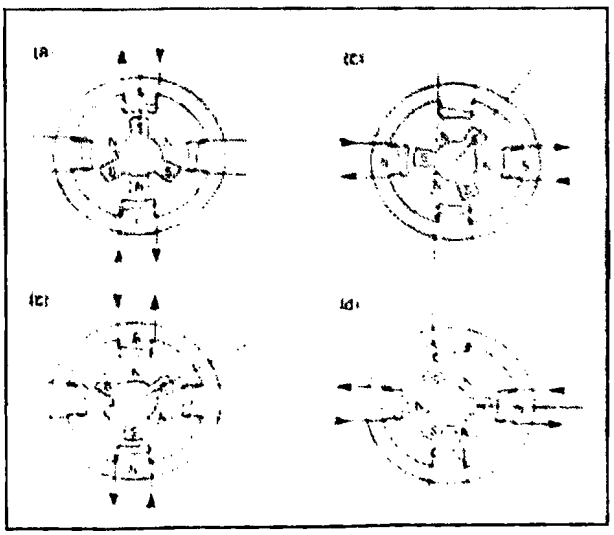



**Σχήμα Α.1.1: Διάγραμμα που δείχνει τη θέση των 6 πόλων του ρότορα και των 4 πόλων του στάτορα ενός τυπικού βηματικού κινητήρα.**

Όταν από μια πηγή ισχύος παρέχεται ρεύμα μόνο σε ένα ζεύγος τυλιγμάτων του στάτορα, τότε το συγκεκριμένο ζεύγος, αν χρησιμοποιήσουμε την τεχνική ορολογία λέμε ότι «διεγείρεται», δηλαδή μετατρέπεται σε μαγνήτη. Ένα από τα τυλίγματα θα αποτελεί το βόρειο πόλο του μαγνήτη και το αντιδιαμετρικό του το

νότιο. Στο Σχήμα Α.1.2.α βλέπουμε ότι το επάνω και κάτω τύλιγμα του στάτορα διαρρέονται από ρεύμα και μετατρέπονται σε μαγνήτη, με το επάνω τύλιγμα να αποτελεί το βόρειο πόλο του μαγνήτη και το κάτω το νότιο πόλο αντίστοιχα. Αυτό θα προκαλέσει την κίνηση του ρότορα έτσι ώστε ο βόρειος πόλος του στάτορα να ευθυγραμμιστεί με τον πλησιέστερο νότιο πόλο του ρότορα, ενώ ο βόρειος πόλος του θα ευθυγραμμιστεί με το νότιο πόλο του στάτορα. Μια βοηθητική γραμμή τοποθετείται στον νότιο πόλο του ρότορα για να μπορούμε να παρακολουθήσουμε την κίνησή του, καθώς το ρεύμα διαρρέει κάθε τύλιγμα του στάτορα χωριστά. Στο Σχήμα Α.1.2.b το επάνω και κάτω τύλιγμα αποδιεγείρονται, ενώ ρεύμα διαρρέει τα πλευρικά τυλίγματα. Το τύλιγμα στη δεξιά πλευρά αποτελεί το νότιο πόλο του μαγνήτη και το αντίστοιχο τύλιγμα της αριστερής πλευράς το βόρειο πόλο. Σε αυτή την κατάσταση ο πόλος του ρότορα που θα πρέπει να ευθυγραμμιστεί με το μαγνήτη του στάτορα είναι ο επόμενος κατά τη φορά κίνησης των δεικτών του ρολογιού σε σχέση με αυτόν του Σχήματος Α.1.2.α. Αυτό σημαίνει ότι ο ρότορας θα πρέπει να περιστραφεί κατά 30° δεξιόστροφα για να ευθυγραμμιστούν οι πόλοι του με τους πόλους του στάτορα. Στο Σχήμα Α.1.2.c το βόρειο και το νότιο τύλιγμα διεγείρονται ξανά με αντίθετη, όμως, πολικότητα. Αυτή η αλλαγή θα περιστρέψει το ρότορα ακόμη 30°. Πρέπει να παρατηρήσουμε ότι ο ρότορας έχει μετακινηθεί κατά 3 βήματα από την αρχική του θέση. Στο Σχήμα Α1.2.d επαναλαμβάνεται η ίδια διαδικασία με αυτή του σχήματος Α.1.2.b μόνο που τα τυλίγματα του στάτορα έχουν αντίθετη πολικότητα. Ο ρότορας έχει κάνει 4 βήματα συνολικά, που σημαίνει ότι έχει περιστραφεί κατά 120°. Επομένως απαιτούνται 12 βήματα των 30° για μια πλήρη περιστροφή του ρότορα.

Ένας άλλος τρόπος για να υπολογίσουμε τον αριθμό των μοιρών που περιστρέφεται ο ρότορας όταν δέχθεί έναν παλμό είναι να διαιρέσουμε τον αριθμό των μοιρών μιας πλήρους περιστροφής (360°) με τον αριθμό των πόλων (βόρειοι και νότιοι) του ρότορα. Έτσι για τον κινητήρα του παραδείγματος οι 360° διαιρούνται με το 12 για να δώσουν 30°.



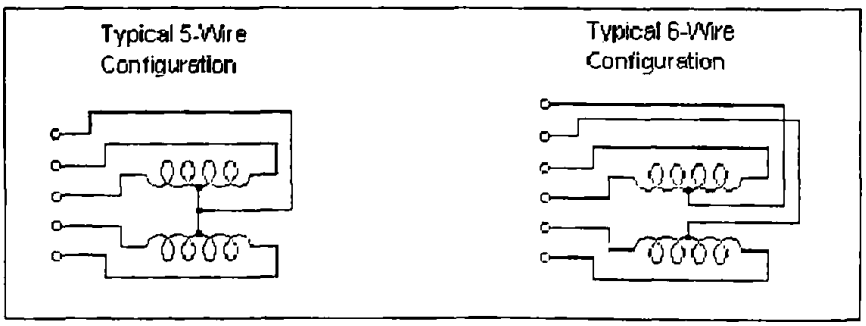
**Σχήμα Α.1.2: Κίνηση του ρότορα όταν τα τυλίγματα του στάτορα διαρρέονται ακολουθιακά από ρεύμα.**

## Α.2 Μονοπολικοί – Διπολικοί κινητήρες

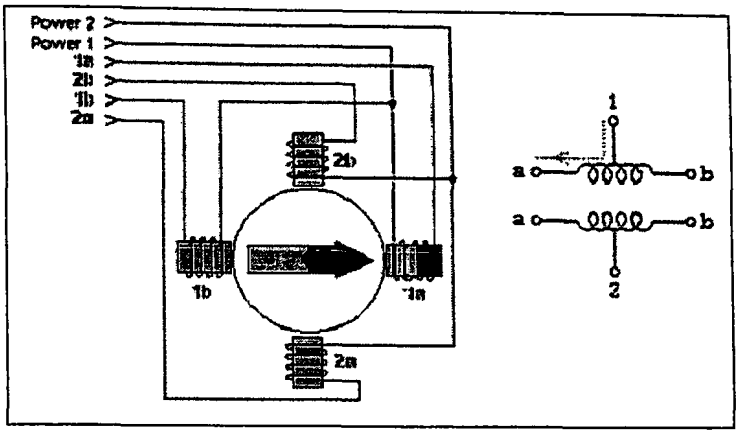
Οι βηματικοί κινητήρες είναι γενικά διαθέσιμοι σε 2 εκδόσεις: στους μονοπολικούς με 5 ή 6 καλώδια, οι οποίοι απαιτούν μόνο μια πηγή τροφοδοσίας και στους διπολικούς με 4 καλώδια που απαιτούν δυο πηγές τροφοδοσίας, είτε μια της οποίας η πολικότητα ελέγχεται από διακόπτη. Η μηχανική κατασκευή και των δύο τύπων είναι ακριβώς η ίδια. Οι διαφορές τους σχετίζονται με τον τρόπο που τα άκρα των τυλιγμάτων βγαίνουν εκτός κινητήρα και στον τρόπο αντιστροφής της πολικότητας του ρεύματος, δηλαδή στο κύκλωμα οδήγησης τους. Στις επόμενες παραγράφους θα αναλύσουμε κάθε τύπο κινητήρα ξεχωριστά. [8]

### Α.2.1 Μονοπολικοί κινητήρες

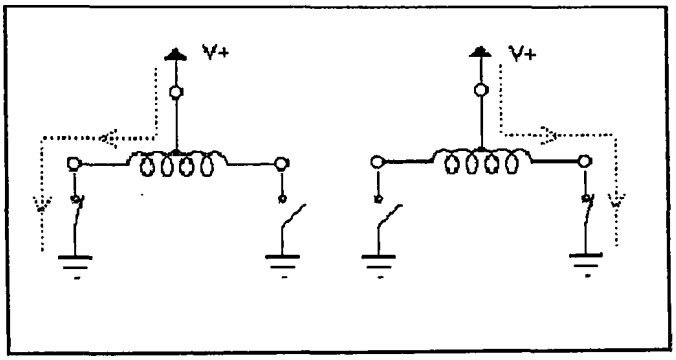
Οι μονοπολικοί κινητήρες που συναντώνται στους PM και υβριδικούς κινητήρες αποτελούνται από δύο τυλίγματα με ένα 'center tap' καλώδιο σε κάθε τύλιγμα. Τα 'center taps' είτε βρίσκονται στο εσωτερικό του κινητήρα ως δύο ξεχωριστά καλώδια, ή συνδέονται στο εσωτερικό και οδηγούνται εκτός ως ένα καλώδιο με αποτέλεσμα να υπάρχουν συνολικά 6 καλώδια (1η περίπτωση) ή 5 (2η περίπτωση). Το Σχήμα Α.2.1.1 δείχνει τον τρόπο σύνδεσης των 6 και 5 καλωδίων αντίστοιχα. Το καλώδιο 'center tap' συνδέεται σε θετική τροφοδοσία, ενώ τα άκρα των τυλιγμάτων γειώνονται εναλλάξ έτσι ώστε να αντιστρέφεται η διεύθυνση του μαγνητικού πεδίου που αναπτύσσεται σε αυτά. Η διεύθυνση ροής του ρεύματος στα τυλίγματα καθορίζει ποιοι πόλοι του ρότορα υφίστανται έλξη από τους διεγερμένους πόλους του στάτορα. Πρέπει να αναφέρουμε ότι η διεύθυνση του ρεύματος εξαρτάται από ποιο μισό τμήμα του τυλίγματος ενεργοποιείται κάθε φορά. Το Σχήμα Α.2.1.2 αποτελεί ένα ιδεατό μοντέλο ενός μονοπολικού κινητήρα. Τα καλώδια Power1 και Power2 είναι τα 'center taps', ενώ το βέλος απεικονίζει το ρότορα. Το Σχήμα Α.2.1.3 δείχνει την διεύθυνση ροής του ρεύματος για αντίθετες πολικότητες, ενώ το Σχήμα Α.2.1.4 δείχνει τη σειρά με την οποία πρέπει να διεγείρονται τα τμήματα των τυλιγμάτων για να λειτουργεί σωστά ο κινητήρας.



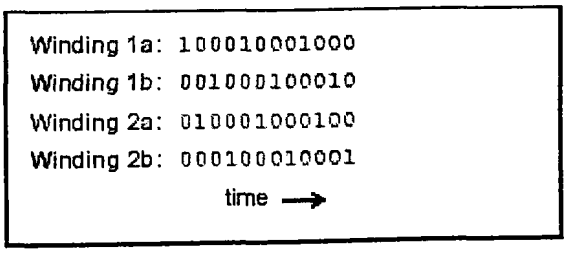

**Σχήμα Α.2.1.1 : Συνδεσμολογία των 5 και 6 καλωδίων ενός μονοπολικού κινητήρα**




**Σχήμα Α.2.1.2: Ιδεατό μοντέλο ενός μονοπολικού κινητήρα**




**Σχήμα Α.2.1.3: Διεύθυνση ρεύματος σε ένα τύλιγμα με αντίθετη πολικότητα**

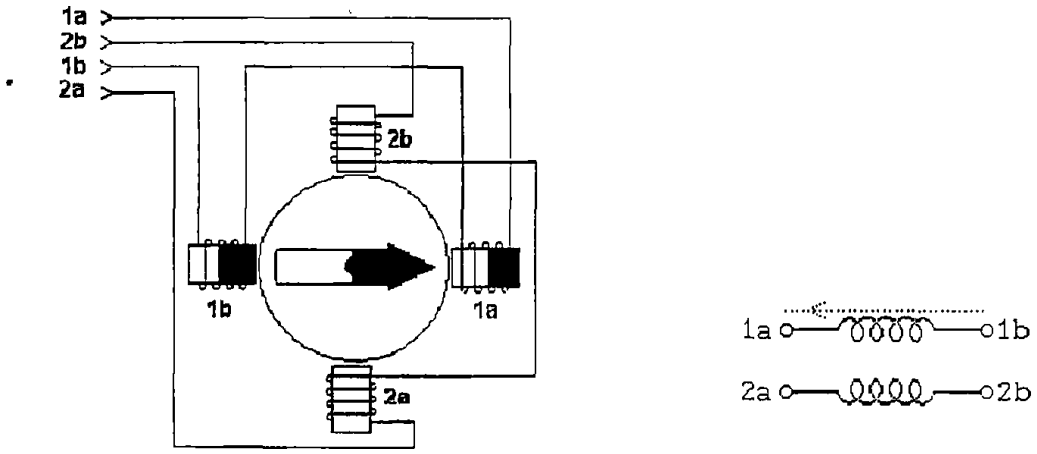



**Σχήμα Α.2.1.4: Σειρά διέγερσης των τυλιγμάτων του κινητήρα του σχήματος Α.2.1.2**

## Α.2.2 Διπολικοί κινητήρες

Οι διπολικοί κινητήρες αποτελούνται από δυο τυλίγματα ανά φάση όπως και οι μονοπολικοί, αλλά έχουν συνολικά 4 καλώδια. Σε αντίθεση με τους μονοπολικούς, οι διπολικοί κινητήρες δεν έχουν center tap καλώδιο. Το Σχήμα Α.2.2.1 αποτελεί ένα ιδεατό μοντέλο ενός διπολικού κινητήρα. Το πλεονέκτημα της μη ύπαρξης του center tap καλωδίου είναι ότι το ρεύμα διαρρέει ολόκληρο το τύλιγμα κάθε φορά αντί για το μισό όπως συμβαίνει στους μονοπολικούς κινητήρες, με αποτέλεσμα να αναπτύσσουν μεγαλύτερη ροπή από τους μονοπολικούς κινητήρες ίδιου μεγέθους. Μειονεκτούν έναντι των πρώτων στο ότι απαιτούν πιο σύνθετο κύκλωμα οδήγησης.

Το ρεύμα που διαρρέει τα τυλίγματα ενός διπολικού κινητήρα είναι αμφίπλευρο. Για να συμβαίνει αυτό, όμως, πρέπει να αλλάξουμε την πολικότητα στα άκρα των τυλιγμάτων. Όπως δείχνει το Σχήμα Α.2.2.1 το ρεύμα θα ρέει από αριστερά προς τα δεξιά στο τύλιγμα 1, όταν το άκρο 1a έχει θετική πολικότητα και το 1b αρνητική. Η ροή του ρεύματος θα αλλάξει όταν αντιστραφεί η πολικότητα των άκρων. Ο πίνακας του Σχήματος Α.2.2.2 δείχνει τη διαδοχική πολικότητα των τυλιγμάτων για τη σωστή λειτουργία του κινητήρα. Το κύκλωμα οδήγησης γνωστό ως 'κύκλωμα γέφυρας-Η' χρησιμοποιείται για να αλλάξει την πολικότητα στα άκρα. Επειδή, όπως αναφέραμε και παραπάνω, ο διπολικός κινητήρας αποτελείται από 2 τυλίγματα ανά φάση απαιτούνται δύο κυκλώματα γέφυρας-Η.




**Σχήμα Α.2.2.1: Ιδεατό μοντέλο ενός διπολικού κινητήρα**

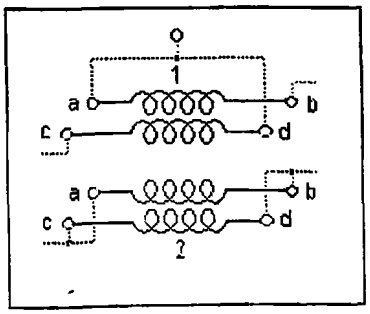
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Terminal 1a: | + | 0 | - | 0 | + | 0 | - | 0 | + | 0 | - | 0 |
| Terminal 1b: | - | 0 | + | 0 | - | 0 | + | 0 | - | 0 | + | 0 |
| Terminal 2a: | 0 | + | 0 | - | 0 | + | 0 | - | 0 | + | 0 | - |
| Terminal 2b: | 0 | - | 0 | + | 0 | - | 0 | + | 0 | - | 0 | + |
| time —> | | | | | | | | | | | | |

**Σχήμα Α.2.2.2: Σειρά πολικότητας των τυλιγμάτων του κινητήρα**

### Α.2.3 Διπλονηματικοί κινητήρες

Οι κινητήρες με διπλονηματικά τυλίγματα είναι πανομοιότυποι με τους διπολικούς με μια εξαίρεση: κάθε τύλιγμα αποτελείται από δύο πηνία τα οποία περιστρέφονται παράλληλα μεταξύ τους, με αποτέλεσμα οι διπλονηματικοί κινητήρες να έχουν οκτώ καλώδια αντί για τέσσερα πού αναλογούν στους διπολικούς.

Οι διπλονηματικοί κινητήρες είναι δυνατόν να οδηγηθούν είτε ως διπολικοί, ή ως μονοπολικοί κινητήρες. Για να χρησιμοποιηθεί ένας διπλονηματικός κινητήρας ως μονοπολικός τα δύο καλώδια κάθε τυλίγματος συνδέονται σε σειρά μεταξύ τους και το σημείο σύνδεσης χρησιμοποιείται ως center tap. Το τύλιγμα 1 του σχήματος Α.2.3.1 απεικονίζει πως συνδέονται τα καλώδια των τυλιγμάτων για να λειτουργήσει ο κινητήρας ως μονοπολικός. Για να χρησιμοποιήσουμε τον κινητήρα αυτού του είδους ως διπολικό τα δύο καλώδια κάθε τυλίγματος συνδέονται είτε παράλληλα ή σειριακά. Το τύλιγμα 2 του σχήματος Α.2.3.1 δείχνει την παράλληλη σύνδεση. Η παράλληλη σύνδεση επιτρέπει τη λειτουργία με υψηλό ρεύμα, ενώ η σειριακή σύνδεση επιτρέπει τη λειτουργία σε υψηλή τάση.



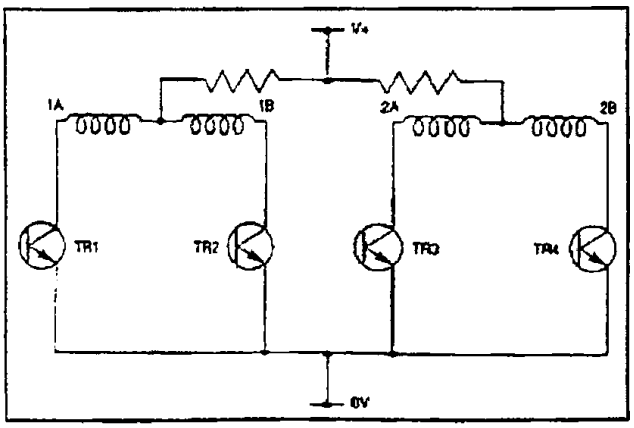

**Σχήμα Α.2.3.1: Διπλονηματικά τυλίγματα**

## Α.3 Κυκλώματα οδήγησης

Σε αυτή την παράγραφο θα ασχοληθούμε με τη βαθμίδα οδήγησης των βηματικών κινητήρων. Η λειτουργία της εστιάζεται στον έλεγχο της ροής του ρεύματος στα τυλίγματα του κινητήρα, ενώ παράλληλα ελέγχει και τη διεύθυνση ροής του. Συνδέεται άμεσα με τα τυλίγματα και την τροφοδοσία ισχύος, ενώ ελέγχεται από ένα ψηφιακό σύστημα, τον 'μεταφραστή' (translator), ο οποίος καθορίζει πότε και ποιοι διακόπτες αυτού θα είναι ανοικτοί ή κλειστοί. Παρακάτω αναλύονται τα πιο απλά κυκλώματα που απαιτούνται για την οδήγηση μονοπολικού κινητήρα αλλά και πιο σύνθετα που χρησιμοποιούνται για τους διπολικούς κινητήρες, όπως αυτός της εφαρμογής μας.

### Α.3.1 Μονοπολικό κύκλωμα οδήγησης (unipolar drive)

Αποτελεί το πιο απλό κύκλωμα οδήγησης για έναν βηματικό κινητήρα. Στο Σχήμα Α.3.1.1 μπορούμε να δούμε ότι το κύκλωμα οδήγησης αποτελείται από δύο τρανζίστορ τα οποία συνδέονται σε σειρά με τα τυλίγματα κάθε φάσης. Τα τρανζίστορ παίζουν το ρόλο των διακοπτών που καθορίζουν αν κάποιο τύλιγμα διαρρέεται από ρεύμα ανάλογα με την κατάστασή τους αν είναι, δηλαδή, κλειστοί ή ανοικτοί. Το κοινό άκρο των τυλιγμάτων 'center tap' συνδέεται σε θετική τάση, ενώ η αρνητική τάση συνδέεται στον εκπομπό κάθε τρανζίστορ. Ονομάζεται μονοπολικό κύκλωμα οδήγησης γιατί το ρεύμα ρέει προς τη μία κατεύθυνση κάθε φορά. Πιο αναλυτικά στέλνει το ρεύμα στα τυλίγματα μέσω του center tap καλωδίου και εναλλάξ γειώνει το άλλο άκρο τους. Το κύκλωμα απαιτεί ο κινητήρας, κάποιες φορές, να λειτουργεί σε τάση μεγαλύτερη από την τάση λειτουργίας του (όταν κλείνουμε την τροφοδοσία σε κάποιο από τα τρανζίστορ). Αυτό προκαλεί αύξηση στην τιμή του ρεύματος, ο περιορισμός του οποίου επιτυγχάνεται με τις αντιστάσεις που συνδέονται μεταξύ του κοινού άκρου των τυλιγμάτων και της θετικής τάσης.



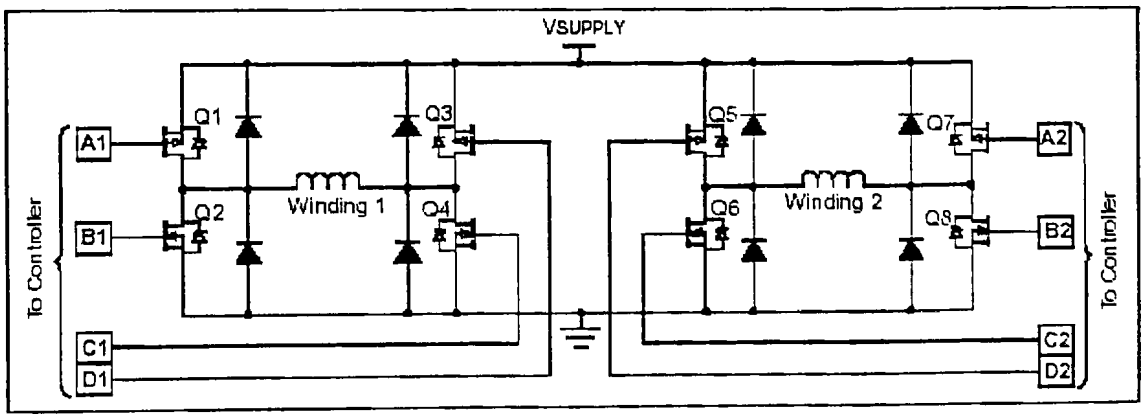

**Σχήμα Α.3.1.1: Μονοπολικό κύκλωμα οδήγησης**

### Α.3.2 Κύκλωμα οδήγησης με γέφυρα

Η οδήγηση των διπολικών κινητήρων είναι πιο σύνθετη από αυτή των μονοπολικών γιατί δεν έχουν center tap καλώδιο στα τυλίγματά τους. Έτσι για να αντιστρέψουμε τη διεύθυνση του πεδίου που αναπτύσσεται στα τυλίγματα αρκεί να αντιστρέψουμε τη διεύθυνση ροής του ρεύματος. Το κύκλωμα που απαιτείται για την οδήγηση διπολικού κινητήρα είναι το κύκλωμα οδήγησης με γέφυρα το οποίο απεικονίζεται στο Σχήμα Α.3.2.1. Όπως βλέπουμε κάθε τύλιγμα συνδέεται με τέσσερα τρανζίστορ. Οι δίοδοι που συνδέονται παράλληλα με τα τρανζίστορ έχουν ως αποστολή την καταστολή των οξειών κορυφών που εμφανίζονται στη τάση εξόδου στο κύκλωμα ισχύος όταν ανοίγει οποιοδήποτε τρανζίστορ. Η ύπαρξη τεσσάρων τρανζίστορ στο βασικό κύκλωμα γέφυρας προσφέρει 16 πιθανούς τρόπους λειτουργίας. Οι δύο πιο εύχρηστοι τρόποι δίνονται παρακάτω:

**Forward mode:** Άγουν τα τρανζίστορ Q1 και Q4
**Reserve mode:** Άγουν τα τρανζίστορ Q2 και Q3

Οι παραπάνω τρόποι λειτουργίας επιτρέπουν στο ρεύμα να πάει από την τροφοδοσία στη γείωση μέσω των τυλιγμάτων. Το κύκλωμα οδήγησης με γέφυρα βρίσκει εφαρμογή όχι μόνο στους βηματικούς, αλλά και στους DC κινητήρες καθώς επίσης και σε περιπτώσεις που απαιτείται αντιστροφή της πολικότητας του ρεύματος. Σε σύγκριση με τη μονοπολική οδήγηση, στην οποία το ρεύμα διαρρέει το τύλιγμα μόνο κατά τη μία κατεύθυνση είναι δυνατό να επιτευχθεί έως και 40% αύξηση της ροπής στρέψης. (Επειδή στη μονοπολική οδήγηση το ρεύμα καταναλώνεται στη τύλιγμα και στις εξωτερικές αντιστάσεις η διπολική οδήγηση πλεονεκτεί ως προς αυτό το σημείο.)




**Σχήμα Α.3.2.1: Κύκλωμα οδήγησης με γέφυρα**

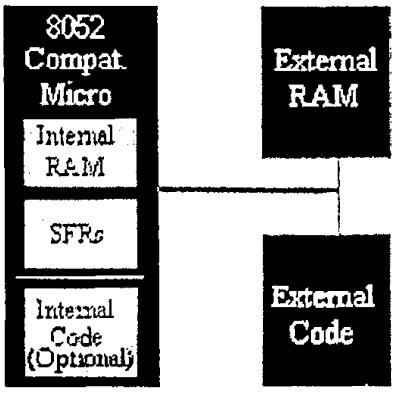
# Παράρτημα Β

# Η αρχιτεκτονική του μικροελεγκτή AT89S8252

## Β.1 Τύποι μνήμης [12]

Ο μικροελεγκτής AT89S8252 έχει τρία βασικά είδη μνήμης τα οποία απεικονίζονται στο Σχήμα Β1.1 και είναι τα παρακάτω:

- Εσωτερική μνήμη (on-chip memory)
- Εξωτερική μνήμη κώδικα (external code memory)
- Εξωτερική μνήμη RAM



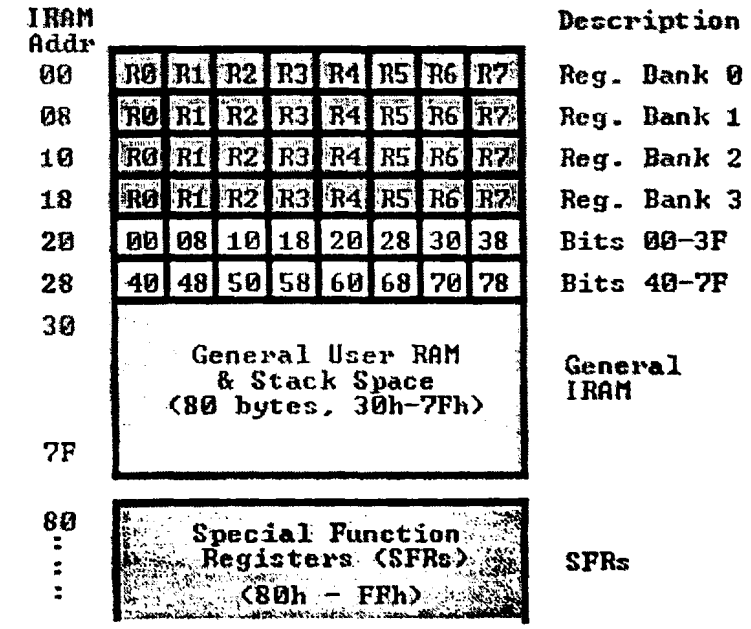

**Σχήμα Β.1.1: Τύποι μνήμης για τον μικροελεγκτή AT89S8252**

Η **εσωτερική μνήμη (on-chip memory)** αναφέρεται σε κάθε είδους μνήμη (RAM, ROM, EPROM) που περιέχεται στον μικροελεγκτή.

Η **εσωτερική μνήμη RAM** έχει μέγεθος 256 bytes και είναι οργανωμένη σε τρεις διακεκριμένες περιοχές όπως φαινεται στο Σχήμα Β.1.2:

1. 32 bytes από τη διεύθυνση $00_h$ μέχρι $1F_h$ που απαρτίζουν 32 λειτουργικούς καταχωρητές, οργανωμένους σε 4 τράπεζες (Register bank 0,1,2,3) με 8 καταχωρητές (R0...R7) η καθεμία. Τα bits RS0 και RS1 του καταχωρητή PSW καθορίζουν ποιά τράπεζα χρησιμοποιείται κάθε φορά.
2. Η περιοχή μνήμης που αποτελείται από 16 bytes και η διευθυνσιοδότησή τους γίνεται με bits. Η διευθυνσιοδότηση με bits είναι χρήσιμη όπου χρειάζεται ένα bit για ένα γεγονός.
3. Τέλος η περιοχή της μνήμης από $30_h$ εως $7F_h$ η οποία διευθυνσιοδοτείται με bytes και είναι γενικής χρήσης.

Τα πρώτα 128 bytes στην εσωτερική μνήμη RAM καταλαμβάνουν ένα παράλληλο χώρο με τον χώρο που καταλαμβάνουν οι SFRs. Έχουν τις ίδιες διευθύνσεις αλλά διαφορετική φυσική περιοχή μνήμης.




**Σχήμα Β.1.2: Οργάνωση εσωτερικής μνήμης RAM**

Η **μνήμη κώδικα (Internal Code memory)** είναι η μνήμη στην οποία είναι αποθηκευμένος ο κώδικας που θα τρέξει στον μικροελεγκτή. Το μέγεθός της είναι 8k και μπορεί να είναι μνήμη ROM είτε EPROM ή επαναπρογραμματιζόμενη Flash μνήμη.

Η **εξωτερική μνήμη RAM (External RAM)** όπως δηλώνει και το όνομά της είναι μνήμη τυχαίας προσπέλασης η οποία βρίσκεται off-chip. Επειδή ακριβώς είναι off-chip, δεν είναι τόσο ευέλικτη ως προς την πρόσβαση και τόσο γρήγορη όπως η εσωτερική μνήμη RAM. Για παράδειγμα για να αυξήσουμε κατά 1 την τιμή ενός byte που βρίσκεται αποθηκευμένο σε μια διεύθυνση μνήμης στην εσωτερική μνήμη RAM, απαιτείται μια εντολή και ένας κύκλος εντολής, ενώ για την ίδια εργασία στην εξωτερική μνήμη RAM απαιτούνται 4 εντολές και 7 κύκλοι εντολών. Στην περίπτωση αυτή η εξωτερική μνήμη είναι 7 φορές πιο αργή από την εσωτερική.

Η **εξωτερική μνήμη κώδικα (External Code Memory)** εκτός από την εξωτερική μνήμη RAM που μελετήσαμε παραπάνω, μπορεί να συνεργαστεί με εξωτερική επαναπρογραμματιζόμενη μνήμη EPROM 64k.

## B.2 Special Function Registers

Οι λειτουργίες του μικροελεγκτή που δεν χρησιμοποιούν την εσωτερική μνήμη RAM (από $00_h$-$7F_h$) γίνεται από μια ομάδα ειδικών εσωτερικών καταχωρητών που ο καθένας ονομάζεται **special function register** και διευθυνσιοδοτούνται με διευθύνσεις από $80_h$-$FF_h$. [12]

Κάθε SFR έχει μία διεύθυνση και ένα όνομα. Δεν είναι όλες οι διευθύνσεις από $80_h$-$FF_h$ διαθέσιμες για ειδικούς καταχωρητές και φυσικά η προσπάθεια προσπέλασης μιας κενής θέσης θα προκαλέσει διακοπή του προγράμματος. Οι SFR φαίνονται στον επόμενο πίνακα Π-1 με τα ονόματα και τις αντίστοιχες διευθύνσεις τους.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 0F8H | | | | | | | | 0FFH |
| 0F0H | B<br>00000000 | | | | | | | 0F7H |
| 0E8H | | | | | | | | 0EFH |
| 0E0H | ACC<br>00000000 | | | | | | | 0E7H |
| 0D8H | | | | | | | | 0DFH |
| 0D0H | PSW<br>00000000 | | | | | SRCR<br>000001XX | | 0D7H |
| 0C8H | T2CON<br>00000000 | T2MOD<br>XXXXXX00 | RCAP2L<br>00000000 | RCAP2H<br>00000000 | TL2<br>00000000 | TH2<br>00000000 | | 0CFH |
| 0C0H | | | | | | | | 0C7H |
| 0B8H | IP<br>XX000000 | | | | | | | 0BFH |
| 0B0H | P3<br>11111111 | | | | | | | 0B7H |
| 0A8H | IE<br>0X000000 | | SRSR<br>00XXXXXX | | | | | 0AFH |
| 0A0H | P2<br>11111111 | | | | | | | 0A7H |
| 98H | SCON<br>00000000 | SBUF<br>XXXXXXXX | | | | | | 9FH |
| 90H | P1<br>11111111 | | | | | | WMCON<br>00000010 | 97H |
| 88H | TCON<br>00000000 | TMOD<br>00000000 | TL0<br>00000000 | TL1<br>00000000 | TH0<br>00000000 | TH1<br>00000000 | | 8FH |
| 80H | P0<br>11111111 | SP<br>00000111 | DP0L<br>00000000 | DP0H<br>00000000 | DP1L<br>00000000 | DP1H<br>00000000 | SPDR<br>XXXXXXXX | PCON<br>0XXX0000 | 87H |

**Πίνακας Π-1: Χάρτης με τους SFRs του μικροελεγκτή AT89S8252**

Στη συνέχεια ακολουθεί ανάλυση όλων των SFRs του μικροελεγκτή AT89S8252.

**P0 (Port P0, Address 80h):** Η θύρα P0 μπορεί να χρησιμοποιηθεί ως Input/Output, ή όταν χρησιμοποιούνται μαζί ως δίαυλος διπλής κατεύθυνσης low order byte μνήμης και δεδομένων για την εξωτερική μνήμη. Κάθε bit αυτού του SFR ανταποκρίνεται σε ένα από τα pins της θύρας. Όταν ένα pin πρόκειται να χρησιμοποιηθεί ως είσοδος, το "1" πρέπει να γραφεί σε αυτό. Αντίστοιχος προγραμματισμός γίνεται με τη χρήση ενός pin ως έξοδος, με την εγγραφή του "0" σε αυτό.

**SP (Stack Pointer, Address 81h):** Η τιμή που παίρνει ο Stack Pointer δηλώνει από ποιά διεύθυνση της εσωτερικής μνήμης RAM θα διαβαστεί η επόμενη τιμή που θα πάρουμε από την μνήμη Stack. Εάν αποθηκεύσουμε μια τιμή στη Stack μνήμη, η τιμή αυτή θα αποθηκευτεί στη διεύθυνση SP+1. Δηλαδή ο SP αυξάνει την τιμή του κατά 1 πρίν την αποθήκευση. Ανάλογα μειώνεται κατά 1 μετά την ανάκτηση δεδομένων.

**DPL/DPH (Data Pointer low/high, Addresses 82h/83h, 84h/85h):** Οι SFRs DPL και DPH αναπαριστούν στην ουσία τους δύο 16-bit καταχωρητές DPTR (Data Pointer) και χρησιμοποιούνται για να αποθηκεύσουν διευθύνσεις μνήμης για εσωτερική και εξωτερική ανάκτηση κώδικα και εξωτερική ανάκτηση δεδομένων.

**PCON (Power Control, Addresses 87h):** Ο Power Control SFR χρησιμοποιείται για την επιλογή του mode λειτουργίας του μικροελεγκτή. Ένα από τα bit του καταχωρητή χρησιμοποιείται για το διπλασσιασμό του baud rate για την επικοινωνία του μικροελεγκτή με τον υπολογιστή μέσω της σειριακής θύρας.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SMOD | * | * | * | GF1 | GF0 | PD | IDL |

| BIT | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| SMOD | Bit καθορισμού σειριακού ρυθμού μετάδοσης δεδομένων. Εάν το bit τεθεί "1" και χρησιμοποιούμε τον Timer1 (modes 1,2,3) διπλασσιάζεται ο ρυθμός μετάδοσης. Εάν τεθεί "0" τότε χρησιμοποιείται ο ρυθμός μετάδοσης του Timer1. |
| BITS 6-4 | Δεν χρησιμοποιούνται |
| GF1 | Bit 1 σημαίας γενικής χρήσης |
| GF0 | Bit 0 σημαίας γενικής χρήσης |
| PD | Power down bit |
| IDL | Idle mode bit |

**Πίνακας Π-2: Ο SFR PCON**

**TCON (Timer Control, Address 88h):** Ο Timer Control SFR χρησιμοποιείται για τον προσδιορισμό λειτουργίας των Timer 1 και Timer 0 του μικροελεγκτή, ενώ κάποια από τα bit του μας ενημερώνουν για την ενεργοποίηση ή μη των εξωτερικών interrupts.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TF1 | TR1 | TF0 | TR0 | IE1 | IT1 | IE0 | IT0 |

| BIT | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| TF1 | Σημαία υπερχείλησης του Timer 1 |
| TR1 | Bit ελέγχου της κατάστασης του Timer 1 |
| TF0 | Σημαία υπερχείλησης του Timer 0 |
| TR0 | Bit ελέγχου της κατάστασης του Timer 0 |
| IE1 | Σημαία εξωτερικου interrupt 1 |
| IT1 | Bit ελέγχου του σήματος ενεργοποίησης του εξωτερικού interrupt 1 |
| IE0 | Σημαία εξωτερικου interrupt 0 |
| IT0 | Bit ελέγχου του σήματος ενεργοποίησης του εξωτερικού interrupt 0 |

**Πίνακας Π-3: Ο SFR TCON**

**TMOD (Timer Mode, Address 89h):** Ο Timer Mode δηλώνει τον τρόπο λειτουργίας των Timer 1 και Timer 0. Χρησιμοποιώντας αυτόν τον καταχωρητή επιλέγουμε εάν ο Timer θα χρησιμοποιηθεί ως 16-bit, 13-bit, 8-bit ή ως δύο timers χωριστά.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| GATE | C/T | M1 | M0 | GATE | C/T | M1 | M0 |
| Timer1 | | | | Timer0 | | | |

| BIT | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| GATE | Όταν το bit αυτό είναι 1, ο Timer θα τρέχει μόνο όταν INT1(P3.3)=1. Εάν το bit είναι 0 τότε θα τρέχει ανεξάρτητα από την κατάσταση του INT1. |
| C/T | Bit επιλογης λειτουργίας του Timer ως χρονομετρητης [timer("0")] ή καταμετρητης[counter("1")]. |
| M1 | Timer/Counter mode select bit 1. |
| M0 | Timer/Counter mode select bit 0. |

**Πίνακας Π-4: Ο SFR TMOD**

Τα bits M1 και M0 καθορίζουν εάν ο Timer θα χρησιμοποιηθεί ως 13-bit, 16-bit, 8-bit ή ως δύο ανεξάρτητοι 8-bit καταχωρητές. Ο τρόπος επιλογής δίνεται στον παρακάτω πίνακα.

| M1 | M0 | MODE |
|---|---|---|
| **0** | **0** | **13-bit Timer** |
| **0** | **1** | **16-bit Timer** |
| **1** | **0** | **8-bit Timer** |
| **1** | **1** | **two separate 8-bit Timers** |

**TL0/TH0 (Timer0 low/high, Addresses 8Ah/8Ch):** Οι δύο SFRs μαζί αποτελούν του Timer 0. Η ακριβής συμπεριφορά τους εξαρτάται από το πως ο Timer 0 έχει επιλεχθεί, μέσω του SFR TMOD, να λειτουργεί.

**TL1/TH1 (Timer1 low/high, Addresses 8Bh/8Dh):** Οι δύο SFRs μαζί αντιπροσωπεύουν τον Timer 1. Η λειτουργία τους είναι πανομοιότυπη με αυτή των TL0 και TH0.

**P1 (Port 1, Address 90h):** Τα pins της θύρας P1 μπορούν να χρησιμοποιηθούν ως είσοδος/έξοδος ανάλογα με την τιμή που έχουν ( "1" και "0' αντίστοιχα)

**SCON (Serial Control, Address 98h):** Ο Serial Control SFR ελέγχει τη λειτουργία της σειριακής θύρας του μικροελεγκτή. Πιο αναλυτικά μέσω αυτού επιλέγουμε το ρυθμό μετάδοσης δεδομένων (baud rate), είτε ενεργοποιούμε τη σειριακή ώστε να λάβει δεδομένα. Εκτός αυτού ο Serial Control καταχωρητής περιέχει σημαίες (flags), οι οποίες τίθενται όταν ένα byte ληφθεί ή αποσταλλεί επιτυχώς.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SM0 | SM1 | SM2 | REN | TB8 | RB8 | TI | RI |

| ΒΙΤ | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| SM0 | Serial Port mode bit 0. |
| SM1 | Serial Port mode bit 1. |
| SM2 | Multiprocessor communications bit. Τίθεται από τον χρήστη ώστε να είναι εφικτή η επικοικωνία σε modes λειτουργίας 2 και 3. Όταν SM2=1 παράγεται ένα interrupt με την προϋπόθεση οτι το 9° bit των δεδομένων που έχουν ληφθεί είναι "1" Εάν το bit αυτό είναι "0" τότε δεν παράγεται κανένα interrupt. Τίθεται επίσης στην τιμή "0" όταν βρισκόμαστε σε λειτουργία mode 0. |
| REN | Receiver Enable. Αυτό το bit πρέπει να είναι πάντα "1" για να είναι εφικτή η λήψη χαρακτήρων. |
| TB8 | Transmit bit 8. Το 9$^{th}$ bit που αποστέλλεται σε mode λειτουργίας 2 και 3. |
| RB8 | Receive bit 8. Το 9$^{th}$ bit που λαμβάνεται σε mode λειτουργίας 2 και 3. |
| TI | Σημαία αποστολής χαρακτήρων. Τίθεται "1" όταν αποσταλεί επιτυχώς ένα byte. |
| RI | Σημαία λήψης χαρακτήρων. Τίθεται "1" όταν ληφθεί επιτυχώς ένα byte. |

**Πίνακας Π-5: Ο SFR SCON**

**SBUF (Serial Buffer, Address 99h):** Ο Serial Buffer SFR χρησιμοποιείται για την αποστολή και λήψη δεδομένων μέσω της σειριακής θύρας. Οποιαδήποτε τιμή γραφτεί στον SBUF θα σταλθεί εκτός του μικροελεγκτή μέσω της γραμμής TXD. Αντίστοιχα κάθε τιμή που λαμβάνει ο μικροελεγκτής μέσω του γραμμής RXD μεταφέρεται στον κώδικα μέσω του SBUF. Με άλλα λόγια ο SBUF χρησιμοποιείται ως έξοδος όταν γράφονται δεδομένα στον μικροελεγκτή και ως είσοδος για ανάγνωση δεδομένων.

**P2 (Port 2, Address A0h):** Η θύρα P2 μπορεί να χρησιμοποιηθεί ως Input/Output παρόμοια με την P1. Η επιπλέον λειτούργια της είναι να παρέχει το high-order byte διεύθυνσης της εξωτερικής μνήμης, σε συνδιασμό με το low-order byte που παρέχει η P1.

**IE (Interrupt Enable, Address A8h):** Ο SFR Interrupt Enable χρησιμοποιείται για την ενεργοποίηση και απενεργοποίηση των interrupts. Τα πρώτα εφτά bits του SFR ενεργοποιούν/απενεργοποιούν συγκεκριμένα interrupts, ενώ το όγδοο bit όλα. Εάν το όγδοο bit του IE είναι "0" όλα τα interrupts απενεργοποιούνται ανεξάρτητα από το αν κάποιο έχει ενεργοποιηθεί από κάποιο χαμηλότερο bit.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| EA | Reserved | Reserved | ES | ET1 | EX1 | ET0 | EX0 |

| BIT | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| EA | Ενεργοποίηση/απενεργοποίηση όλων των interrupts |
| BIT 6 | Δεν χρησιμοποιείται(διαθέσιμο για μελλοντική χρήση) |
| BIT 5 | Δεν χρησιμοποιείται(διαθέσιμο για μελλοντική χρήση) |
| ES | Ενεργοποίηση interrupt σειριακής θύρας |
| ET1 | Ενεργοποίηση interrupt υπερχείλησης του Timer 1 |
| EX1 | Ενεργοποίηση εξωτερικού interrupt 1 |
| ET0 | Ενεργοποίηση interrupt υπερχείλησης του Timer 0 |
| EX0 | Ενεργοποίηση εξωτερικού interrupt 0 |

**Πίνακας Π-6: Ο SFR IE**

**P3 (Port 3, Address B0h):** Η θύρα P3 μπορεί να χρησιμοποιηθεί ως Input/Output παρόμοια με την P1. Σε αντίθεση με τις θύρες P0 και P2 που μπορούν να έχουν λειτουργίες με εξωτερική διευθυνσιοδότηση και να αλλάξουν και τα 8 pins όταν είναι σε εναλλακτική λειτουργία, κάθε pin της θύρας P3 μπορεί να προγραμματιστεί ξεχωριστά για χρήση I/O ή εναλλακτική λειτουργία.

**IP (Interrupt Priority, Address B8h):** Ο SFR Interrupt Priority θέτει προτεραιότητα σε interrupts που πιθανόν να συμβούν ταυτόχρονα. Η προτεραιότητα είναι high (1) ή low (0). Τα interrupts με high προτεραιότητα διακόπτουν την εκτέλεση αυτών με low. Εάν, όμως, εκτελείται το serial interrupt, κανένα άλλο interrupt δεν μπορεί να το διακόψει γιατί είναι το υψηλότερο σε προτεραιότητα.

| + 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| * | * | Reserved | PS | PT1 | PX1 | PT0 | PX0 |

| BIT | Περιγραφή Λειτουργίας |
|---|---|
| * | Δεν χρησιμοποιείται |
| * | Δεν χρησιμοποιείται |
| Bit 5 | Δεν χρησιμοποιείται(διαθέσιμο για μελλοντική χρήση) |
| PS | Προτεραιότητα στο interrupt της σειριακής θύρας |
| PT1 | Προτεραιότητα στο interrupt του Timer 1 |
| PX1 | Προτεραιότητα στο εξωτερικό interrupt 1 |
| PT0 | Προτεραιότητα στο interrupt του Timer 0 |
| PX0 | Προτεραιότητα στο εξωτερικό interrupt 0 |

**Πίνακας Π-7: Ο SFR IP**

**T2CON (Timer 2 Control, Address C8h):** Ο Timer 2 είναι ένας 16-bit χρονομετρητής (timer)/καταμετρητής (counter). Ο τρόπος λειτουργίας του επιλέγεται από το bit C/T2 του SFR T2CON. Ο Timer 2 έχει τρείς διαφορετικές modes λειτουργίας: ανίχνευσης γεγονότος (capture), αυξανόμενης ή μειώμενης καταμέτρησης (up or down counter) και γεννήτριας του ρυθμού μετάδοσης γεγονότων (baud rate generator), οι οποίες επιλέγονται από bits του T2CON. Ο Timer 2 επιλέγεται ως γεννήτρια ρυθμού μετάδοσης θέτοντας τα bits TCLK ή RLCK του T2CON. Στη λειτουργία του ως καταμετρητής, ο καταχωρητής αυξάνεται κατά 1 στη μετάβαση από λογικό "1" σε λογικό "0"στον αντίστοιχο εξωτερικό ακροδέκτη T2EX. Για να χρησιμοποιήσουμε τον Timer 2 ως ανιχνευτή γεγονότος αρκεί να θέσουμε τους ακροδέκτες EXEN2 και CP/RL2 σε λογικό "1".

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| TF2 | EXF2 | RCLK | TCLK | EXEN2 | TR2 | C/T2 | CP/RL2 |

| ΒΙΤ | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| TF2 | Σημαία υπερχείλισης για τον Timer2.Τίθεται "1"όταν έχουμε υπερχείλιση του Timer2, ενώ μηδενίζεται όταν γίνει κλήση της αντίστοιχης ρουτίνας του timer.Δεν παίρνει την τιμή "1" εάν κάποιο από τα bits RCLK ή TCLK γίνει "1". |
| EXF2 | Εξωτερική σημαία για τον Timer2. Τίθεται "1" όταν έχουμε ανίχνευση ή αποθήκευση τιμής κατά τη μετάβαση 1-0 στο bit T2EX(P1.1), αλλά μόνο όταν το bit EXEN2 είναι "1". |
| RCLK | Timer 2 Receive Clock. Όταν τίθεται στην τιμή "1", ο Timer 2 χρησιμοποιείται για τον καθορισμό του ρυθμού λήψης δεδομένων μέσω σειριακής θύρας, ενώ όταν είναι "0" χρησιμοποιείται ο Timer 1. |
| TCLK | Timer 2 Transmit Clock. Όταν τίθεται στην τιμή "1", ο Timer 2 χρησιμοποιείται για τον καθορισμό του ρυθμού αποστολής δεδομένων μέσω της σειριακής θύρας, ενώ όταν είναι "0" χρησιμοποιείται ο Timer 1. |
| EXEN2 | Timer2 External Enable. Όταν τίθεται στην τιμή "1", η μετάβαση 1-0 στο bit T2EX(P1.1) θα προκαλέσει ανίχνευση ή αποθήκευση κάποιας τιμής. |
| TR2 | Timer 2 Run. When set, Timer2 will be turned on. Otherwise, it is turned off. |
| C/T2 | Bit επιλογής λειτουργίας του Timer 2. Όταν είναι "0" χρησιμοποιείται ως χρονομετρητής, ενώ όταν έχει την τιμή "1" χρησιμοποιείται ως καταμετρητής εξωτερικών γεγονότων. |
| CP/RL2 | Capture or Reload select for Timer 2. Εάν η τιμή αυτού του bit είναι "0", έχουμε φόρτωση τιμής κατά την υπερχείληση του Timer 2, ή τη μετάβαση 1-0 στον ακροδέκτη T2EX EXEN2=1. Εάν είναι "1" ο Timer 2 χρησιμοποιείται ως ανιχνευτής γεγονότων κατα τη μετάβαση 1-0 στον ακροδέκτηT2EX έαν EXEN2=1 |

**Πίνακας Π-8: Ο SFR TCON2**

**TL2/TH2 (Timer2 low/high, Addresses CCh/CFh ):** Ο Timer 2 αποτελείται από δύο 8-bit καταχωρητές τους TL2 και TH2. Στη λειτουργία του σαν χρονομετρητής ο καταχωρητής TL2 αυξάνεται σε κάθε κύκλο μηχανής. Επειδή ο κάθε κύκλος μηχανής

αποτελείται από 12 περιόδους του ταλαντωτή, ο ρυθμός καταμέτρησης είναι το 1/12 της συχνότητας του κρυστάλλου.

**RCAP2L/RCAP2H (Timer 2 Capture Low/High, Addresses CAh/CBh):** Οι δύο SFRs αναπαριστούν τον καταχωρητή ανίχνευσης για τον Timer 2. Χρησιμοποιούνται για την αποθήκευση της τιμής του Timer 2, ή για την ανίχνευση της τιμής αυτού. Η ακριβής λειτουργία τους εξαρτάται από την τιμή που «φορτώνεται» στον καταχωρητή T2CON.

**T2MOD (Timer 2 Mode, Address C9h):** Ο καταχωρητής T2MOD καθορίζει τον τρόπο λειτουργίας του Timer 2.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| - | - | - | - | - | - | T2OE | DCEN |

| BIT | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| Bits 2-7 | Δεν χρησιμοποιούνται. Διαθέσιμα για μελλοντική χρήση |
| T2OE | Timer 2 Output Enable bit |
| DCEN | Όταν DCEN=1 ο Timer 2 χρησιμοποιείται ως καταμετρητής (up or down counter). |

**PSW (Program Status Word, Address D0h):** Ο PSW χρησιμοποιείται για την αποθήκευση αποτελεσμάτων συγκεκριμένων εντολών προγράμματος. Περιλαμβάνει carry flag (C), auxiliary carry (AC), overflow (OV) και parity flag. Επιπλέον περιέχει και τις register select flags για την επιλογή της τράπεζας καταχωρητών (register bank) που θέλουμε να χρησιμοποιήσουμε.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| CY | AC | F0 | RS1 | RS0 | OV | Reserved | P |

| BIT | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| CY | Carry Flag |
| AC | Auxiliary carry flag |
| F0 | User flag 0 |
| RS1 | Register bank select bit 1 |
| RS0 | Register bank select bit 0 |
| OV | Overflow flag |
| Bit 1 | Not used (reserved for future) |
| P | Parity Flag |

**Πίνακας Π-9: Ο SFR PSW**

**ACC (Accumulator, Address E0h):** Ο καταχωρητής Acc είναι ο πιο ευέλικτος από τους SFRs και χρησιμοποιείται σε πολλές διεργασίες όπως μαθηματικές πράξεις, χειρισμούς λογικής Boole και μεταφορά δεδομένων μεταξύ του 8052 και της εξωτερικής μνήμης.

**B (B Register, Address 0Fh):** Ο καταχωρητής B χρησιμοποιείται σε πολ/μούς και διαιρέσεις και δεν έχει άλλη λειτουργία εκτός από την αποθήκευση δεδομένων.

**WMCON (Watchdog and Memory Control Register, Address 96h):** Ο καταχωρητής WMCON περιέχει bits ελέγχου του Watchdog Timer. Οι ακροδέκτες EEMEN και EEMWE χρησιμοποιούνται για την επιλογή της μνήμης δεδομένων EEPROM μήκους 2k, ενώ ο DPS επιλέγει έναν από τους δύο DPTR καταχωρητές που είναι διαθέσιμοι.

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| PS2 | PS1 | PS0 | EEMWE | EEMEN | DPS | WDTRST | WDTEN |

| BIT | ΠΕΡΙΓΡΑΦΗ ΛΕΙΤΟΥΡΓΙΑΣ |
|---|---|
| PS2<br>PS1<br>PS0 | Prescaler bits for the Watchdog Timer. When all three bits are set to "0" the watchdog timer has a nominal period of 16ms.When all three bits are set to "1" the nominal period is 2048ms |
| EEMWE | EEPROM Data Memory Write Enable Bit. Set this bit "1" before initiating byte write to on-chip EEPROM with the MOVX instruction. User software should set this bit "0" after EEPROM write is completed |
| EEMEN | Internal EEPROM Access Enable. When EEMEN=1 the MOVX instruction with DPTR will acess on-chip EEPROM instead of external data memory. When EEMEN=0, MOVX with DPTR accesses external data memory. |
| DPS | Data Pointer Register Select. DPS=0 selects the first bank of Data Pointer Register,DP0, and DPS=1 selects the second bank DP1 |
| WDTRST<br>RDY/BSY | Watchdog Timer Reset and EEPROM Ready/Busy Flag. Each time this bit is set to "1" by user software a pulse is generated to reset the watchdog timer. The WDTRST bit is then automatically reset to "0" in the next instruction cycle. The WDTRST bit is Write-Only. This bit also serves as the RDY/BSY flag in a Read-Only mode during EEPROM write.RDY/BSY=1 means that the EEPROM is ready to be programmed. While programming operations are being executed, the RDY/BSY bit equals "0" and is automatically reset to "1" when programming is completed |
| WDTEN | Watchdog Timer Enable Bit. WDTEN=1 enables the watchdog timer and WDTEN=0 disables the watchdog timer. |

**Πίνακας Π-10: Ο SFR WMCON**

**SPI Registers (SPCR, SPDR, SPSR Registers, Addresses D5h, 86h, AAh):** Οι τρεις καταχωρητές χρησιμοποιούνται για τον έλεγχο του SPI (Serial Peripheral Interface).

# Παράρτημα Γ

# Ο προγραμματισμός της LCD οθόνης

## Γ.1 Ο ελεγκτής HD44780

Υπεύθυνος για την λειτουργία της οθόνης της εφαρμογής μας, αλλά και της πλειοψηφίας των οθονών είναι ο ελεγκτής HD44780.Ο ελεγκτής της οθόνης διαθέτει τρία είδη μνήμης και τέσσερις βασικούς καταχωρητές.

Τα είδη μνήμης είναι:

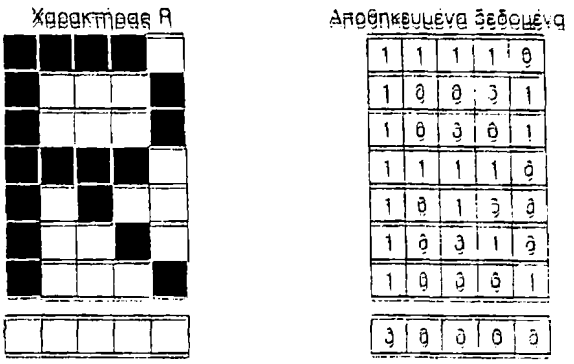
- Η μνήμη ROM της γεννήτριας χαρακτήρων (Character Generator ROM ή CG ROM) που αποτελείται από 9920 bits.
- Η μνήμη RAM της γεννήτριας χαρακτήρων (Character Generator RAM ή CG RAM) που αποτελείται από 512 bits.
- Η μνήμη RAM των δεδομένων απεικόνισης (Display Data RAM ή DD RAM) που αποτελείται από 80x8 bits.

Οι τέσσερις καταχωρητές είναι:

- Ο καταχωρητής εντολών (Instruction Register ή IR) μεγέθους 8 bits.
- Ο καταχωρητής δεδομένων (Data Register ή DR) μεγέθους 8 bits.
- Ο μετρητής διευθύνσεων (Address Counter ή AC) μεγέθους 7 bits.
- Η σημαία απασχολημένου (Busy Flag ή BF) μεγέθους 1 bit.

Στη μνήμη ROM της γεννήτριας χαρακτήρων (**CG ROM**) είναι αποθηκευμένα μόνιμα τα δεδομένα για τους χαρακτήρες που μπορεί να απεικονίσει η οθόνη. Για τον κάθε χαρακτήρα έχουν αποθηκευτεί τα δεδομένα της μήτρας έτσι ώστε να είναι

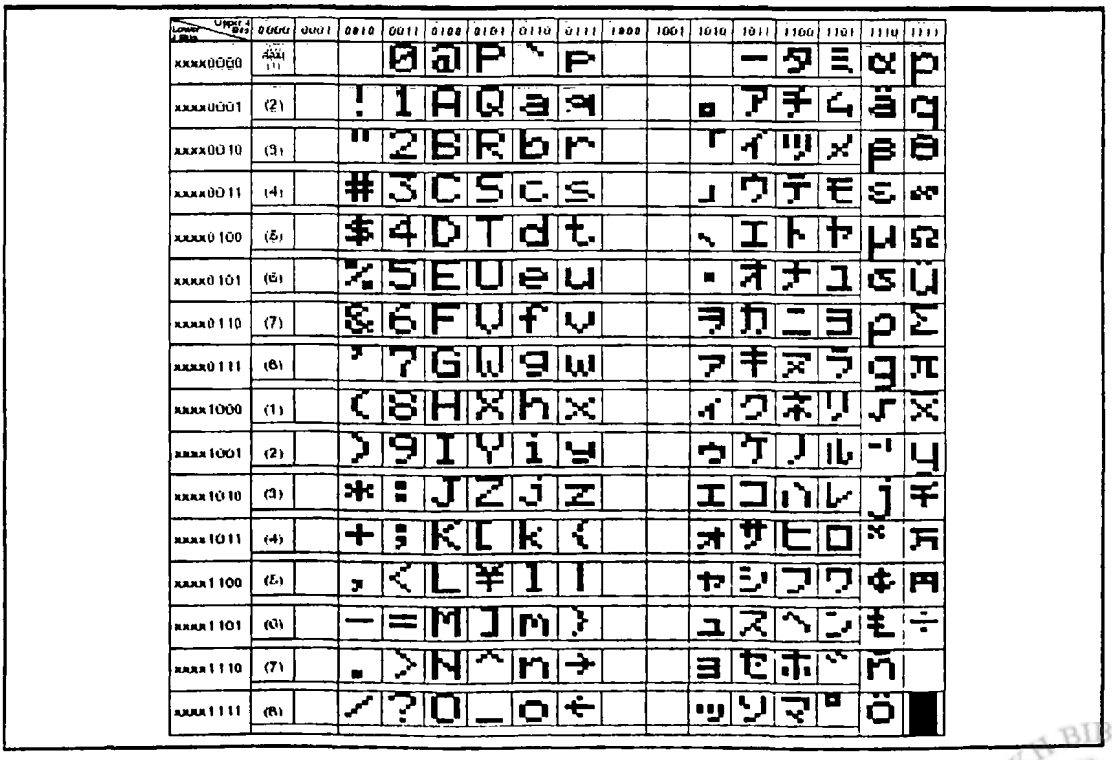
δυνατή η εμφάνισή του. Στο Σχήμα Γ1.1 εμφανίζονται τα δεδομένα που είναι αποθηκευμένα ώστε να είναι δυνατή η απεικόνιση του χαρακτήρα R.




**Σχήμα Γ1.1: Δεδομένα στη CG ROM για τον χαρακτήρα R**

Στη μνήμη είναι αποθηκευμένα τα δεδομένα για 208 χαρακτήρες της μήτρας 5x8 τελειών και για 32 χαρακτήρες της μήτρας 5x10 τελειών. Ο πίνακας Π1 εμφανίζει τους χαρακτήρες που μπορεί να απεικονίσει η οθόνη. Στις δυο τελευταίες στήλες είναι αποθηκευμένοι χαρακτήρες μήτρας 5x10 τελειών.

Ο κάθε χαρακτήρας έχει ένα ξεχωριστό κωδικό ο οποίος ονομάζεται **κωδικός χαρακτήρα** και είναι η ταυτότητά του. Ο κωδικός του χαρακτήρα (στη δεκαεξαδική του μορφή) συμπεραίνεται από την σειρά και τη στήλη που έχει ο χαρακτήρας στον πίνακα (σειρά:στήλη$_{16}$). Για την απεικόνιση ενός χαρακτήρα στην οθόνη θα πρέπει ο αντίστοιχος κωδικός να αποθηκευτεί στην κατάλληλη θέση μνήμης απεικόνισης DD RAM, λειτουργία που θα περιγραφεί στη συνέχεια.

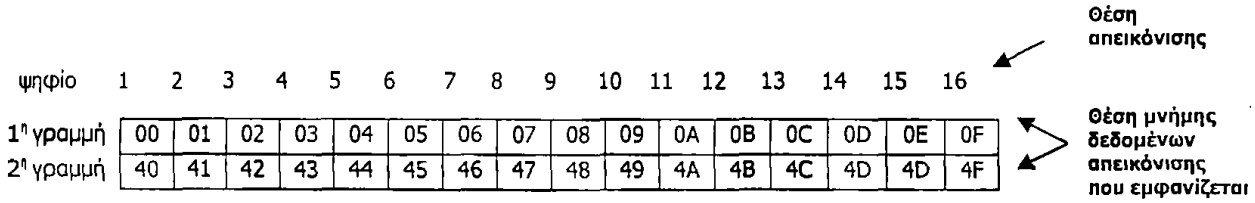


**Πίνακας Π1: Χάρτης κωδικών χαρακτήρων**

Στη μνήμη RAM της γεννήτριας χαρακτήρων (**CG RAM**) είναι δυνατή η δημιουργία μέχρι 8 νέων χαρακτήρων στη μήτρα των 5x8 τελειών ή 4 χαρακτήρων στην μήτρα των 5x10 τελειών. Επομένως, εάν οι χαρακτήρες που είναι σχεδιασμένοι από τον κατασκευαστή δεν καλύπτουν τις τρέχουσες ανάγκες, είναι δυνατή η δημιουργία νέων χαρακτήρων (π.χ. χαρακτήρων του ελληνικού αλφαβήτου που δεν συμπεριλαμβάνονται στον χάρτη των κωδικών χαρακτήρων).

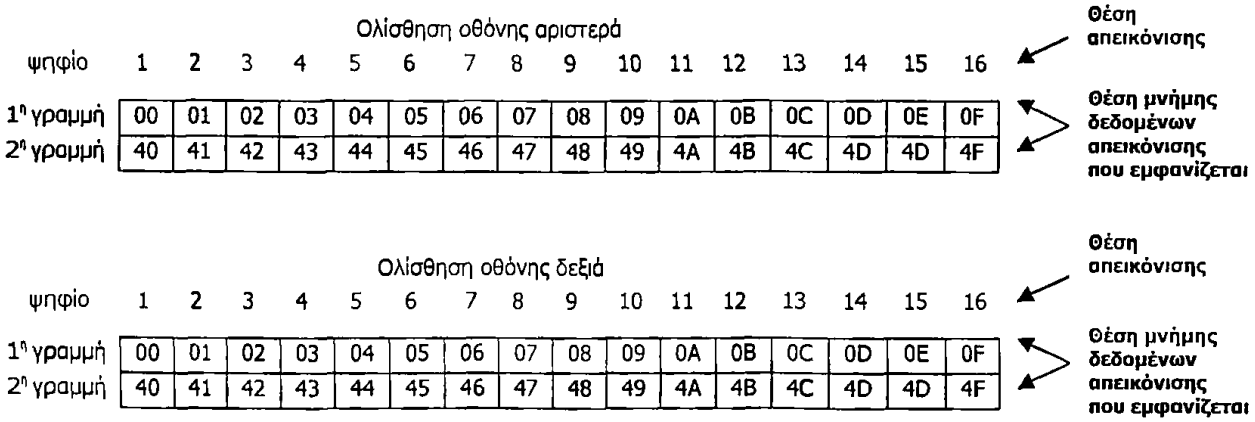
Στην μνήμη RAM των δεδομένων απεικόνισης (**DD RAM**) αποθηκεύονται οι κωδικοί των χαρακτήρων που εμφανίζονται στην οθόνη. Το μέγεθος της μνήμης είναι 80 bytes (80x8 bits). Η περιοχή της μνήμης που δεν χρησιμοποιείται για απεικόνιση μπορεί να χρησιμοποιηθεί ως μνήμη RAM γενικού τύπου.

Οι δύο σειρές της οθόνης δεν απεικονίζουν συνεχόμενες διευθύνσεις μνήμης RAM των δεδομένων απεικόνισης. Το Σχήμα Γ1.2 δίνει τη σχέση μεταξύ των θέσεων απεικόνισης στην οθόνη και των θέσεων της μνήμης δεδομένων απεικόνισης που εμφανίζονται στην οθόνη μετά την αρχικοποίηση αυτής.




**Σχήμα Γ1.2: Σχέση θέσεων απεικόνισης και θέσεων μνήμης DD RAM**

Με τις εντολές ολίσθησης που διαθέτει η οθόνη είναι δυνατόν να επιλεγεί για εμφάνιση μια άλλη περιοχή της μνήμης RAM των δεδομένων απεικόνισης. Στο Σχήμα Γ1.3 εμφανίζεται η περιοχή απεικόνισης στην οθόνη μετά από ολίσθηση κατα μία θέση αριστερά ή κατά μία θέση δεξιά από την αρχική θέση.




**Σχήμα Γ1.3: Περιοχή απεικόνισης μετά από ολίσθηση**

Ο ελεγκτής της οθόνης έχει δύο 8-bits καταχωρητές τον καταχωρητή εντολών IR (Instruction Register) και τον καταχωρητή δεδομένων DR (Data Register). Επιλέγονται από το σήμα RS.

Στον **καταχωρητή IR** αποθηκεύονται οι κωδικοί των εντολών όπως για παράδειγμα η εντολή καθαρισμού της οθόνης, η εντολή ολίσθησης του δρομέα, η θέση μνήμης των δεδομένων απεικόνισης DD RAM και η θέση της μνήμης RAM της γεννήτριας χαρακτήρων CG RAM. Ο καταχωρητής μπορεί να εγγραφεί από τον μικροελεγκτή, αλλά δεν μπορεί να αναγνωσθεί.

Στον **καταχωρητή DR** αποθηκεύονται προσωρινά τα δεδομένα προς εγγραφή στην μνήμη απεικόνισης DD RAM ή της CG RAM, ή που διαβάζονται από την DD RAM ή την CG RAM. Τα δεδομένα που γράφονται στον DR από τον μικροελεγκτή εγγράφονται αυτόματα στην DD RAM ή την CG RAM με μία εσωτερική λειτουργία. Ο καταχωρητής DR χρησιμοποιείται επίσης για την αποθήκευση των δεδομένων κατά την ανάγνωση από την DD RAM ή την CG RAM. Όταν η πληροφορία της διεύθυνσης εγγράφεται στον καταχωρητή IR, τα δεδομένα διαβάζονται στον DR από την DD RAM ή την CG RAM με μία εσωτερική λειτουργία. Η διαδικασία της μεταφοράς στον μικροελεγκτή ολοκληρώνεται με την ανάγνωση από τον μικροελεγκτή του καταχωρητή DR. Εφόσον ο μικροελεγκτής διαβάσει τον DR, τα δεδομένα της επόμενης διεύθυνσης της DD RAM ή της CG RAM μεταφέρονται στον DR για την επόμενη ανάγνωση από τον μικροελεγκτή.

Ο **μετρητής διευθύνσεων AC** προσδιορίζει τις διευθύνσεις στις DD RAM και CG RAM. Όταν μία εντολή για διεύθυνση εγγράφεται στον καταχωρητή IR, τότε η πληροφορία της διεύθυνσης μεταφέρεται από τον IR στον AC. Η επιλογή είτε της DD RAM, είτε της CG RAM προσδιορίζεται συγχρόνως από την εντολή. Μετά την εγγραφή στην (ή την ανάγνωση από την) DD RAM ή την CG RAM, η τιμή του AC αυξάνεται κατά 1 (ή μειώνεται κατά 1). Τα περιεχόμενα του καταχωρητή AC εξάγονται στους ακροδέκτες DB0-DB6 όταν το σήμα RS=0 και το σήμα R/W=1.

Όταν η τιμή της σημαίας απασχολημένου, **Busy Flag**, είναι 1 ο ελεγκτής εργάζεται εσωτερικά και η επόμενη εντολή δεν γίνεται αποδεκτή. Η σημαία απασχολημένου εξάγεται στον ακροδέκτη DB7 εφόσον RS=0 και R/W=1.

Ο πίνακας Π2 περιγράφει συνοπτικά τον τρόπο επιλογής των παραπάνω καταχωρητών με βάση τις τιμές των σημάτων RS και R/W. [3]

| RS | R/W | Λειτουργία |
|---|---|---|
| 0 | 0 | Εγγραφή των κωδικών των εντολών στον IR |
| 0 | 1 | Ανάγνωση σημαίας απασχολημένου (DB7) και μετρητή διευθύνσεων (DB0- DB6) |
| 1 | 0 | Εγγραφή δεδομένων από τον DR στην DD RAM ή την CG RAM |
| 1 | 1 | Ανάγνωση δεδομένων στον DR από την DD RAM ή την CG RAM |

**Πίνακας Π2: Επιλογή καταχωρητών**

## Γ.2 Οι εντολές προγραμματισμού της LCD οθόνης

Όπως έχει ειπωθεί ο ελεγκτής HD44780 είναι υπεύθυνος για τη λειτουργία της οθόνης LCD. Η λειτουργία της βασίζεται σε μια σειρά εντολών που δέχεται ο ελεγκτής. Ο πίνακας Π1 που ακολουθεί παρέχει πληροφορίες για τις εντολές αυτές. Η εκτέλεση κάθε εντολής γίνεται σε ορισμένο χρόνο ο οποίος εμφανίζεται στην τελευταία στήλη του πίνακα. Εάν η σημαία απασχολημένου (Busy Flag) δεν λαμβάνεται υπόψη στον έλεγχο περάτωσης της εκτέλεσης της εντολής τότε θα πρέπει να υπάρξει, μετά την αποστολή της εντολής στον ελεγκτή, μία καθυστέρηση ίση ή μεγαλύτερη της αναγραφόμενης. [3]

| Εντολή | Κωδικός | | | | | | | | | | Περιγραφή λειτουργίας | Χρόνος εκτέλεσης |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 | | |
| ;αθαρισμός θόνης | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Καθαρίζει την οθόνη και μεταφέρει τον δρομέα στην αρχική διεύθυνση (διεύθυνση 0) | 1.64ms |
| πιστροφή ρομέα στην αρχή | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | * | Επιστρέφει τον δρομέα στην αρχική θέση. Σε περίπτωση ολίσθησης της οθόνης υπάρχει επιστροφή ενώ τα δεδομένα της DD RAM δεν μεταβάλλονται | 1.64ms |
| ρισμός ιτάστασης σαγωγής | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | Α/Μ | ΟΛ | Ορίζει την κατεύθυνση κίνησης του δρομέα Α/Μ και αν η οθόνη θα ολισθαίνει ΟΛ κατά την εγγραφή ή την ανάγνωση δεδομένων | 40μs |
| λεγχος τεικόνισης | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ΑΟ | ΑΔ | ΘΔ | Ενεργοποιεί την απεικόνιση ΑΟ της οθόνης , του δρομέα ΑΔ και το αναβόσβημα της θέσης του δρομέα ΘΔ | 40μs |
| λίσθηση ιομέα/οθόνης | 0 | 0 | 0 | 0 | 0 | 1 | Ο/Δ | Δ/Α | * | * | Μετακινεί τον δρομέα και ολισθαίνει την οθόνη Ο/Δ χωρίς την μεταβολή των δεδομένων της DD RAM δεξιά ή αριστερά Δ/Α | 40μs |
| ιισμός ιτουργίας | 0 | 0 | 0 | 0 | 1 | ΜΔ | ΓΡ | ΜΜ | * | * | Ορισμός μήκους λέξης δεδομένων ΜΔ, αριθμού απεικονιζόμενων γραμμών ΓΡ και του μεγέθους της μήτρας ΜΜ | 40μs |
| ιισμός εύθυνσης ήμης RAM ννήτριας ρακτήρων | 0 | 0 | 0 | 1 | Διεύθυνση μνήμης RAM γεννήτριας χαρακτήρων $AC_{CG\ RAM}$ | | | | | | Ορίζει τη διεύθυνση μνήμης CG RAM για την αποστολή ή λήψη δεδομένων | 40μs |
| ιισμός εύθυνσης ήμης RAM εικόνισης ρακτήρων | 0 | 0 | 1 | Διεύθυνση μνήμης RAM των δεδομένων απεικόνισης $AC_{DD\ RAM}$ | | | | | | | Ορίζει τη διεύθυνση μνήμης DD RAM για την αποστολή ή λήψη δεδομένων | 40μs |
| νάγνωση μαίας ασχολημένου ή ιμετρητή ευθύνσεων | 0 | ΣΑ | Μετρητής διευθύνσεων AC | | | | | | | | Ανάγνωση της σημαίας απασχολημένου ΣΑ που δείχνει την εσωτερική λειτουργία της μονάδας και του καταχωρητή διευθύνσεων AC | 40μs |
| γραφή δομένων στην ; RAM ή ην DD RAM | 1 | 0 | Δεδομένα προς εγγραφή | | | | | | | | Εγγραφή δεδομένων στη μνήμη RAM των δεδομένων απεικόνισης ή στην μνήμη RAM της γεννήτριας χαρακτήρων | 40μs |
| νάγνωση δομένων από ν CG RAM ή ν DD RAM | 1 | 1 | Δεδομένα προς ανάγνωση | | | | | | | | Ανάγνωση δεδομένων από την μνήμη RAM των δεδομένων απεικόνισης ή από την μνήμη RAM της γεννήτριας χαρακτήρων | 40μs |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Α/Μ | = | 1 | Αύξηση | Δ/Α | = | 1 | Ολίσθηση δεξιά |
| Α/Μ | = | 0 | Μείωση | Δ/Α | = | 0 | Ολίσθηση αριστερά |
| ΟΛ | = | 1 | Ολίσθηση οθόνης | ΜΔ | = | 1 | Μήκος δεδομένων 8-bits |
| ΑΟ | = | 1 | Απεικόνιση στην οθόνη | ΜΔ | = | 0 | Μήκος δεδομένων 4-bits |
| ΑΔ | = | 1 | Απεικόνιση δρομέα | ΓΡ | = | 1 | Εμφάνιση 2 γραμμών |
| ΘΔ | = | 1 | Αναβόσβημα θέσης δρομέα | ΓΡ | = | 0 | Εμφάνιση 1 γραμμής |
| Ο/Δ | = | 1 | Ολίσθηση οθόνης | ΜΜ | = | 1 | Μέγεθος μήτρας 5x10 τελείες |
| Ο/Δ | = | 0 | Μετακίνηση δρομέα | ΜΜ | = | 0 | Μέγεθος μήτρας 5x7 τελείες |

**Πίνακας Π3: Ακολουθία εντολών για την LCD οθόνη**

## Γ.3 Χρησιμοποιώντας τη γραμμή ελέγχου EN

Όπως έχει ειπωθεί η γραμμή ελέγχού EN χρησιμοποιείται για να ενημερωθεί η οθόνη οτι ο χρήστης είναι έτοιμος να εκτελέσει μια εντολή και ότι έχει προετοιμάσει τον δίαυλο δεδομένων (data bus) και τις άλλες γραμμές ελέγχου (control lines) να δεχθούν δεδομένα. Πρέπει να πούμε οτι η γραμμή EN πρέπει να αυξάνεται/μειώνεται πριν/μετά από κάθε εντολή που στέλνεται στην LCD ανεξάρτητα από το αν η εντολή είναι read/write, σχόλιο/κείμενο. Εν συντομία πρέπει πάντα να χειριζόμαστε την EN όταν επικοινωνούμε με την οθόνη. Η EN είναι ένας τρόπος για να καταλάβει η οθόνη οτι "μιλάμε" σε αυτή. Εάν δεν αυξήσουμε/μειώσουμε την τιμή της γραμμής αυτής η οθόνη δεν θα καταλάβει οτι αναφερόμαστε σε αυτή για τις υπόλοιπες γραμμές ελέγχου.

Πριν αλληλεπιδράσουμε με οποιονδήποτε τρόπο με την LCD οθόνη θα πρέπει EN=1. Αυτό γίνεται με την εντολή:

**SETB EN**

Αμέσως μόλις τελειώσουμε τις έντολές με τις υπόλοιπες γραμμές ελέγχου θα μηδενίσουμε ξανά τη γραμμή EN.

**CLR EN**

Η LCD λαμβάνει και εκτελεί τις εντολές/σχόλια αμέσως μόλις EN=0. Εάν, όμως, ποτέ δεν μηδενίσουμε την EN οι εντολές δεν θα εκτελεστούν. Επιπρόσθετα όταν EN=0 και η LCD εκτελεί τις εντολές, απαιτείται ένα χρονικό διάστημα για την εκτέλεση κάθε εντολής. Ο χρόνος αυτός εξαρτάται από τη εντολή και από την ταχύτητα του κρυστάλλου που είναι συνδεδεμένος στην είσοδο του ταλαντωτή του επεξεργαστή HD44780. [12]

## Γ.4 Ελέγχοντας τη σημαία απασχολημένου (Busy flag) της LCD οθόνης

Όπως αναφέραμε στην προηγούμενη παράγραφο απαιτείται ένα χρονικό διάστημα για να εκτελέσει η οθόνη καθε εντολή. Η καθυστέρηση ποικίλλει εξαρτώμενη από τη συχνότητα του κρυστάλλου που λειτουργεί ο επεξεργαστής HD44780, όπως επίσης και από την εντολή που εκτελείται.

Ενώ είναι πολύ πιθανό να γράψουμε έναν κώδικα ο οποίος θα περιμένει για ένα συγκεκριμένο χρονικό διάστημα που θα επιτρέψει στην οθόνη να εκτελέσει την εντολή, η μέθοδος του "waiting" δεν είναι τόσο ευέλικτη. Εάν η συχνότητα του κρυστάλλου μεταβληθεί, ο κώδικας θα πρέπει να τροποποιηθεί. Επιπλέον εάν η LCD αλλαχθεί με μία άλλη, παρόλο που το HD44780 είναι συμβατό, θα απαιτείται περισσότερος χρόνος για την εκτέλεση των εντολών και το πρόγραμμα δεν θα δουλέψει έως ότου να τροποποιηθεί κατάλληλα.

Μια πιο ευέλικτη μέθοδος προγραμματισμού είναι η χρησιμοποίηση της εντολής "Ανάγνωση σημαίας απασχολημένου", η οποία καθορίζει εάν η τελευταία εντολή που στάλθηκε στην οθόνη έχει εκτελεστεί ή όχι. Η εντολή "Ανάγνωση σημαίας απασχολημένου" μας επιστρέφει δύο bits από πληροφορίες. Η πληροφορία η οποία είναι χρήσιμη σε εμάς βρίσκεται αποθηκευμένη στο bit DB7 του διαύλου δεδομένων. Γενικά όταν χρησιμοποιούμε την παραπάνω εντολή η LCD αμέσως θα αυξήσει το DB7 εάν αυτή είναι ακόμη απασχολημένη με την εκτέλεση της προηγούμενης εντολής, ή θα το μειώσει για να υποδείξει οτί δεν είναι πλέον απασχολημένη. Στο σημείο αυτό μπορούμε ελεύθερα να στείλουμε την επόμενη εντολή στην οθόνη.

Επειδή χρησιμοποιούμε αυτή την εντολή κάθε φορά που στέλνουμε ένα νέο σχόλιο στην οθόνη, είναι χρήσιμο να γράψουμε μία υπορουτίνα που θα ελέγχει συνεχώς την κατάσταση του bit DB7.

```
WAIT_LCD:
        SETB EN                 ; Start LCD command
        CLR RS                  ; It is a command
        SETB RW                 ; It is a read command
        MOV DATA,#0FFh          ; Set all pins to FF initially
        MOV A,DATA              ; Read the return value
        JB ACC.7, WAIT_LCD      ; If bit DB7=high, LCD is still busy
        CLR EN                  ; Finish command
        CLR RW                  ; Turn off RW for future return
        RET
```

Το κάλεσμα της παραπάνω υπορουτίνας δίνει στην οθόνη τον χρόνο που απαιτείται για να εκτελέσει τις εντολές και επιπλέον κάνει το πρόγραμμά μας συμβατό με οποιαδήποτε LCD οθόνη ανεξάρτητα από το πόσο γρήγορη ή αργή είναι. [12]

## Γ.5 Αρχικοποίηση της οθόνης

Πριν χρησιμοποιήσουμε την LCD οθόνη πρέπει να κάνουμε αρχικοποίηση. Αυτό γίνεται εφικτό στέλνοντας στην οθόνη μια σειρά από εντολές αρχικοποίησης.[12]

Η πρώτη εντολή που στέλνουμε πρέπει να λεει στην LCD εάν θα επικοινωνήσουμε με έναν 4-bit ή 8-bit δίαυλο δεδομένων. Επίσης επιλέγουμε 5x8 character font (μήτρα 5x8 τελειών). Αυτές οι δύο εκδοχές επιλέγονται στέλνοντας στην οθόνη το σχόλιο "38h" ως εντολή. Υπενθυμίζουμε οτι η γραμμή ελέγχου RS πρέπει να είναι μηδέν όταν στέλνουμε ένα σχόλιο ως εντολή στην οθόνη. Για να στείλουμε το σχόλιο "38h" στην LCD εκτελούμε τις παρακάτω εντολές για τον μικροελεγκτή AT89S8252.

```
SETB EN
CLR RS
MOV DATA, #38h
CLR EN
LCALL WAIT_LCD
```

Το σχόλιο "38h" είναι στην πραγματικότητα το άθροισμα ενός αριθμού εκδοχών των bits. Η εντολή που πρέπει να στείλουμε είναι στην ουσία το σχόλιο"20h" (Ορισμός λειτουργίας). Παρ' όλα αυτά προσθέτουμε σε αυτό τις τιμές "10h" για να υποδείξουμε 8-bit αριθμό (8-bit δίαυλο δεδομένων) και "08h" για να υποδείξουμε οτι η οθόνη έχει δύο γραμμές.

Έχουμε στείλει το πρώτο byte για τη διαδικασία της αρχικοποίησης. Το δεύτερο byte είναι η εντολή "0Eh". Επιπλέον πρέπει να επαναλάβουμε τον παραπάνω κώδικα μαζί με την εντολή αυτή. Το δεύτερο κομμάτι κώδικα είναι:

```
SETB EN
CLR RS
MOV DATA, #0Eh
CLR EN
LCALL WAIT_LCD
```

Το σχόλιο "0Eh" είναι η εντολή "08h"+"04h" για να ανοίξει η LCD. Σε αυτό προστίθεται το σχόλιο "02h" για να ανοίξει ο δρομέας.

Το τελευταίο byte που πρέπει να στείλουμε χρησιμοποιείται για τη διαμόρφωση των διαφόρων λειτουργικών παραμέτρων της LCD. Αρκεί, λοιπόν, να στείλουμε την τιμή "06h".

```
SETB EN
CLR RS
```

```
MOV DATA, #06h
CLR EN
LCALL WAIT_LCD
```

Το σχόλιο "06h" είναι η εντολή "04h"+"02h" που υποδεικνύει οτι κάθε φορά που στέλνουμε έναν χαρακτήρα στην οθόνη, η θέση του δρομέα μετακινείται αυτόματα μια θέση προς τα δεξιά. Ολόκληρος ο κώδικας για την αρχικοποίηση της οθόνης ακολουθεί παρακάτω:

```
INIT_LCD:
        SETB EN
        CLR RS
        MOV DATA, #38h
        CLR EN
        LCALL WAIT_LCD
        SETB EN
        CLR RS
        MOV DATA, #0Eh
        CLR EN
        LCALL WAIT_LCD
        SETB EN
        CLR RS
        MOV DATA, #06h
        CLR EN
        LCALL WAIT_LCD
        RET
```

## Γ.6 Καθαρισμός της οθόνης

Όταν η LCD αρχικοποιηθεί, η επιφάνειά της θα πρέπει αυτόματα να καθαριστεί από τον επεξεργαστή HD44780. Παρ' όλα αυτά καλό θα ήταν να καθαρίσουμε την οθόνη ως πρώτη λειτουργία μετά την αρχικοποίηση της. [12]

Ένα σχόλιο υπάρχει για να επιτευχθεί αυτή η λειτουργία. Χωρίς έκπληξη αυτό το σχόλιο είναι το "01h". Αφού ο καθαρισμός της οθόνης είναι μια λειτουργία που θα κάνουμε περισσότερες από μια φορές, είναι καλή ιδέα να φτιάξουμε μια υπορουτίνα που θα εκτελεί αυτή τη λειτουργία :

```
CLEAR_LCD:
        SETB EN
        CLR RS
```

```
MOV DATA, #06h
CLR EN
LCALL WAIT_LCD
RET
```

## Γ.7 Εγγραφή κειμένου στην οθόνη

Η εγγραφή κειμένου στην οθόνη είναι αυτό που στην ουσία θέλουμε να κάνουμε. Ακολουθεί μια υπορουτίνα που κάνει αυτή την εργασία:

```
WRITE_TEXT:
        SETB EN
        SETB RS
        MOV DATA, A
        CLR EN
        LCALL WAIT_LCD
```

Τώρα που έχουμε γράψει όλες τις απαραίτητες υπορουτίνες, είναι αρκετά εύκολο να εμφανίσουμε κείμενο στην οθόνη. Ο κώδικας που ακολουθεί, εμφανίζει στην οθόνη το κείμενο "HALLO WORLD".

```
LCALL INIT_LCD
LCALL CLEAR_LCD
MOV A, # 'H'
LCALL WRITE_TEXT
MOV A, # 'E'
LCALL WRITE_TEXT
MOV A, # 'L'
LCALL WRITE_TEXT
MOV A, # 'L'
LCALL WRITE_TEXT
MOV A, # 'O'
LCALL WRITE_TEXT
MOV A, # 'W'
LCALL WRITE_TEXT
MOV A, # 'O'
LCALL WRITE_TEXT
MOV A, # 'R'
LCALL WRITE_TEXT
MOV A, # 'L'
LCALL WRITE_TEXT
MOV A, # 'D'
LCALL WRITE_TEXT
RET
```

## Γ.8 Θέση του δρομέα

Το προηγούμενο παράδειγμα 'HELLO WORLD' είναι απλό από την άποψη οτι γράφει το κείμενο στην πάνω αριστερή γωνία της οθόνης. Τι συμβαίνει, όμως, όταν θέλουμε να εμφανίσουμε την λέξη 'HELLO' στην πάνω αριστερή γωνία και τη λέξη 'WORLD' στη δεύτερη γραμμή στον 10ο χαρακτήρα; Αυτό ακούγεται απλό και είναι απλό, αν και απαιτείται περισσότερη κατανόηση του σχεδιασμού της LCD.

Το ολοκληρωμένο HD44780 περιέχει μνήμη η οποία είναι προσαρμοσμένη στην οθόνη. Όλο το κείμενο που γράφουμε στο HD44780 αποθηκεύεται στη μνήμη. Ο επεξεργαστής διαβάζει αυτόματα τη μνήμη στην οθόνη για να εμφανιστεί το κείμενο σε αυτή. Η μνήμη μπορεί να απεικονιστεί με τον ακόλουθο "memory map":

| Display | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Line 1 | 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 0A | 0B | 0C | 0D | 0E | 0F | 10 | 11 | 12 | 13 | 14 | 15 | ... |
| Line 2 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 4A | 4B | 4C | 4D | 4E | 4F | 50 | 51 | 52 | 53 | 54 | 55 | ... |

**Σχήμα Γ8.1: Σχέση θέσεων απεικόνισης και θέσεων μνήμης DD RAM**

Στον παραπάνω πίνακα παρατηρούμε οτι η οθόνη μετρά 2 γραμμές με 16 χαρακτήρες ανά γραμμή. Ο αριθμός σε κάθε κουτί είναι η διεύθυνση μνήμης η οποία ανταποκρίνεται σε κάθε θέση της οθόνης. Ο πρώτος χαρακτήρας στην πάνω αριστερή γωνία είναι η διεύθυνση 00h. Ο επόμενος χαρακτήρας είναι στη διεύθυνση 01h κ.τ.λ. Αυτό συνεχίζεται έως ότου φτάσουμε στον 16ο χαρακτήρα της πρώτης γραμμής, ο οποίος βρίσκεται στη διεύθυνση 0Fh. Όμως ο πρώτος χαρακτήρας της δεύτερης γραμμής, όπως φαίνεται και από τον πίνακα, είναι στη διεύθυνση 40h. Αυτό σημαίνει οτι αν γράψουμε ένα χαρακτήρα στην τελευταία θέση της πρώτης γραμμής και στη συνέχεια γράψουμε ένα δεύτερο χαρακτήρα, ο δεύτερος χαρακτήρας δεν θα εμφανιστεί στη δεύτερη γραμμή της οθόνης. Αυτό συμβαίνει γιατί ο δεύτερος χαρακτήρας αυτόματα θα γραφεί στη διεύθυνση 10h, ενώ η δεύτερη γραμμή ξεκινά από τη διεύθυνση 40h. Έτσι χρειαζόμαστε ένα σχόλιο που θα σταλθεί στη LCD και θα λεει ποιά θα είναι η θέση του δρομέα στη δεύτερη γραμμή. Η "Set Cursor Position" εντολή είναι η "80h". Σε αυτή πρέπει να προσθέσουμε τη διεύθυνση της θέσης όπου επιθυμούμε την θέση του δρομέα. Στο παράδειγμά μας είπαμε οτι θέλαμε να εμφανίσουμε τη λέξη 'WORLD' στη δεύτερη γραμμή, στον 10ο χαρακτήρα.

Αναφερόμενοι ξανά στον πίνακα με τις διευθύνσεις μνήμης βλέπουμε οτι η θέση του 10ου χαρακτήρα της δεύτερης γραμμής είναι η διεύθυνση 4Ah. Πριν λοιπόν γράψουμε την λέξη 'WORLD' στην LCD, πρέπει να στείλουμε την εντολή "80h"+ "4Ah". Έτσι στέλνοντας το σχόλιο "CAh" στην LCD, ο δρομέας θα τοποθετηθεί στη δεύτερη γραμμή, στο 10ο χαρακτήρα. [12]

# Παράρτημα Δ

## Ο Κώδικας σε γλώσσα Assembly για τον μικροελεγκτή AT89S8252

```
;*******************************************************************************
;                                                                              ;
; Stepper Motor Driving Routine                                                ;
; for Linear Stage control                                                     ;
;                                                                              ;
*******************************************************************************
; Author: A.Vamvaka                                                            ;
; HEP-Lab UOI 2005                                                             ;
*******************************************************************************

; Motor control:
                RESET           BIT     P1.4
                ENABLE                  P1.0
                CLOCK                   P1.2
                HALF/FULL               P1.3
                CW/CCW                  P1.1

; Terminal Switches:
                Left Terminal Switsch   BIT     P3.3
                Right Terminal Switch           P3.2

; Control bits:
                Resetting_Table BIT     P2.7
                Local                   P2.5
                Remote                  P3.7
                Manual                  P3.6
                Auto                    P3.5
                Run                     P3.4
                Emergency Stop          P2.6

; LCD Display Bits:
                Enable          BIT     P2.2
                Register Select         P2.0
                Read/Write              P2.1
                Data bits               P0.0-P0.7


        .org 0000h

;************************************** INITIALIZATION ***************************************

INIT:           MOV SP,#30h             ; Initialize Stack Pointer
                MOV TMOD,#20h           ; Set Timer for Mode2
                MOV TH1,#0d0h           ;baud rate 1200bps
                MOV SCON,#50h
                ORL PCON,#80h
                MOV TCON,#40h           ; Run Timer1

;************************************** Motor Control bits ***********************************

                SETB P1.4
                CLR P1.3                ; Choose full steps
                SETB P1.0               ; Enable stepper motor

                LCALL INIT_LCD          ; Initialize LCD Display
                LCALL CLEAR_LCD         ; Clear LCD Display
                LCALL WRITE_POWERON     ; Show "POWER ON" message
                LCALL DELAY5s           ; for 5 seconds
START:          LCALL WRITE_PRESSRST    ; Show "PLEASE
                                        ;       PRESS RST" message
CHECK_RST:      JNB P2.7, RESETTING_TABLE ; Loop until P2.7=0
                SJMP CHECK_RST

;************************************** Emergency stop condition *****************************

ASFALEIA:       LCALL DELAY50ms
                ACALL WRITE_ASFALEIA    ; Show "SYSTEM HALTED
                                        ;  PRESS RST" message
                CLR P1.0                ; Disable stepper motor
                LJMP CHECK_RST
                RET
```

```
;********************************** Choose system operation ************************************

CHOOSE:         MOV P3,#11111111b
                MOV P2,#11111011b
                MOV A,P2
                ANL A,#11101000b
                MOV P2,A
                LCALL WRITE_CHOOSE          ; Show "PLEASE CHOOSE
                                            ;       LOCAL/REMOTE/RST" message
CHOOSE1:        JNB P2.5, LOCATE            ; If bit P2.5=0 choose LOCAL operation
                JNB P3.7, REMOTE            ; If bit P3.7=0 choose REMOTE operation
                JNB P2.7, RESETTING_TABLE   ; If bit P2.7=0 choose RESETTING_TABLE operation
                SJMP CHOOSE1

;************************************ RESETTING_TABLE OPERATION (Local operation) ***************************

RESETTING_TABLE: LCALL DELAY50ms
                 LCALL WRITE_RESETTINGTABLE ; Show "RESETTING TABLE
                                            ;       PLEASE WAIT..."
                 SETB P1.0                  ; Turn stepper motor on
                 CLR P1.1                   ; Stepper motor is rotated to right
REF2:            JNB P3.3, GORIGHT          ; Check Left Terminal Switch
                 ACALL DOSTEP               ; Stepper motor makes one step
                 JNB P2.6, ASFALEIA         ; Check Emergency Stop Switch
                 SJMP REF2
GORIGHT:         SETB P1.1                  ; Stepper motor is rotated to left
                 LCALL METAKINISI5          ; Move table for 5mm
                 SJMP CHOOSE
                 RET

;************************************ LOCAL OPERATION ***************************************

LOCATE:         LCALL DELAY50ms
                LCALL WRITE_LOCAL           ; Show "LOCAL OPERATION
                                            ;       CHOOSE MAN/AUTO" message
                MOV R5,#32d
                JNB P3.6, MANUAL            ; If P3.6=0 choose MANUAL operation
                JNB P3.5, AUTO              ; If P3.5=0 choose AUTO operation
                SJMP LOCATE

;************************************ REMOTE OPERATION ***************************************

REMOTE:         LCALL WRITE_REMOTE          ; Show "REMOTE OPERATION" message
                MOV R5,#32d
FIRSTCHR:       LCALL GETCHR                ; Get character
                CJNE A,#'0',NEXTCHR         ; Check if ASCII character is '0'
                LJMP MANUALr                ; Then switch to MANUAL mode
NEXTCHR:        CJNE A,#'1',FIRSTCHR        ; Check if ASCII character is '1'
                LJMP AUTOr                  ; Then switch to AUTO mode

;************************************ MANUAL MODE (Local operation) ***************************

MANUAL:         LCALL DELAY50ms
                LCALL WRITE_MODE0           ; Show "LOCAL MANUAL
                                            ;       PRESS RUN" message
                SETB P1.0                   ; Enable stepper motor
                CJNE R5,#32d,CONTINUE
                SJMP LOOP10
CONTINUE:       LCALL WRITE_PRESSRUNORRSTORAUTO ; Show "MANUAL ** STEP
                                            ;       RUN/AUTO/RST" message
LOOP10:         JNB P2.7, RESETTING_TABLE   ; Check RST switch
                JNB P3.5, AUTO              ; Check AUTO switch
                JB P3.4, LOOP10             ; Check RUN switch
                LCALL DELAY50ms
                SETB P1.1
                LCALL YPOROYTINA1           ; Show the number of step which is execudte

                LCALL MODE
                DEC R5
                CJNE R5,#00d,ENDROYT
                LJMP START
ENDROYT:        JNB P3.4,ENDROYT            ; Check RUN switch
```

```
                LCALL DELAY50ms
                SJMP CONTINUE
                RET

;************************************ AUTO MODE (Local operation) ********************************

AUTO:           LCALL DELAY50ms
                LCALL WRITE_MODE1       ; Show "LOCAL AUTO
                                        ;       PRESS RUN" message
                SETB P1.0               ; Enable stepper motor
LOOP:           JB P3.4,LOOP            ; Check RUN switch
                LCALL DELAY50ms
                SETB P1.1               ; Stepper motor rotates to left
COUNTER32:      LCALL YPOROYTINA1       ; Show the number of step which
                                        ; is executed
                LCALL MODE
                LCALL DELAY500ms
                LCALL DELAY500ms
                DJNZ R5, COUNTER32
                LCALL EPISTROFH         ; Table makes 32 steps to the opposite direction
                RET

;*************************************************************************************************

MODE:           CLR P2.3                ; Turn relay off
                LCALL DELAY500ms        ; 0,5 sec of delay
                SETB P2.3               ; Turn relay on
                LCALL DELAY5s           ; 5 sec of delay
                LCALL METAKINISI        ; Table makes one 1,875mm-step
                RET

;************************************** 5 mm movement ***************************************

METAKINISI5:    MOV R1,#88h
                MOV R2,#13h
COUNTER5000:    LCALL DOSTEP            ; Stepper motor makes one step
                JNB P2.6, ASFALEIA1     ; Check Emergency Stop Switch
                JNB RI,NEXT5            : Check if a character is received
                SJMP GETCHR5            ; Get character
NEXT5:          MOV A,R1
                CLR C
                SUBB A,#01d
                MOV R1,A
                MOV A,R2
                SUBB A,#00d
                MOV R2,A
                CJNE R2,#00d, COUNTER5000
                CJNE R1,#00d, COUNTER5000
                RET

;************************************** Get character 'E' ***********************************
GETCHR5:        MOV A,SBUF              ; Get character
                ;ANL A,#7fh             ;  Mask off 8th bit
                CLR RI                  ; Clear serial status bit
                CJNE A,#'E', NEXT5      ; Compare character to 'E'
                LJMP ASFALEIA1          ; Emergency Stop condition
                RET

;********************************** Emergency Stop Operation ************************************

ASFALEIA1:      LCALL DELAY50ms
                ACALL WRITE_ASFALEIA    ; Show "SYSTEM HALTED
                                        ;       PRESS RST" message
                CLR P1.0                ; Disable stepper motor
                LJMP CHECK_RST
                RET

;******************************** 1,9mm movement (Local operation) ******************************

METAKINISI:     MOV R1,#6Ch
                MOV R2,#07h
COUNTER1900:    LCALL DOSTEP            ; Stepper motor makes one step
```

```
                JNB P3.2, SWRIGHT          ; Check right terminal switch
                JNB P2.6, ASFALEIA1        ; Check Emergency Stop Switch
                MOV A,R1
                CLR C
                SUBB A,#01d
                MOV R1,A
                MOV A,R2
                SUBB A,#00d
                MOV R2,A
                CJNE R2,#00d,COUNTER1900
                CJNE R1,#00d,COUNTER1900
                RET

;********** Table has reached right terminal switch (Local Operation) **********

SWRIGHT:        LCALL DELAY50ms
                LCALL WRITE_SWRIGHT        ; Show "REF1
                                           ;       PRESS RST" message
                CLR P1.0                   ; Disable stepper motor
HIER:           JB P2.7, HIER              ; Check RST switch
                LJMP RESETTING_TABLE       ; Then execute RESETTING_TABLE operation
                RET

;********** Table moves to initial position (Local operation) **********

EPISTROFH:      LCALL DELAY5s              ; 5 sec of delay
                LCALL WRITE_INITPOSITION   ; Show "MOVING TO
                                           ;       INIT POSITION" message
                CLR P1.1                   ; Stepper motor rotates to right
                MOV R5,#32d
LOOP32:         LCALL METAKINISI           ; Table makes one 1,875mm-step
                DJNZ R5, LOOP32
                LJMP CHOOSE
                RET

;********** Stepper motor makes one step **********

DOSTEP:         CLR P1.2
                LCALL DELAY1ms             ; 1ms of delay
                SETB P1.2
                LCALL DELAY1ms
                RET

;********** 1ms of delay **********

DELAY1ms:       MOV R7,#153d
TIMER:          JNB RI,NEXT10              ; Check if a character is received
                SJMP GETCHR10              ; Get character
NEXT10:         JNB P2.6, ASFALEIA1        ; Check Emergency Stop switch
                DJNZ R7,TIMER
                NOP
                RET

;********** 50ms of delay **********

DELAY50ms:      MOV R3,#50d
LOOP2:          ACALL DELAY1ms
                DJNZ R3,LOOP2
                RET

;********** 500ms of delay **********

DELAY500ms:     MOV R4,#10d
LOOP3:          ACALL DELAY50ms
                DJNZ R4, LOOP3
                RET
;********** 5 sec of delay **********

DELAY5s:        MOV R0,#10d
LOOP4:          ACALL DELAY500ms
                DJNZ R0, LOOP4
                RET
```

```
;************************************** Get character 'E' ***************************************

GETCHR10:       MOV A,SBUF              ; Get character
                ;ANL A,#7fh             ; Mask off 8th bit
                CLR RI                  ; Clear serial status bit
                CJNE A,#'E', NEXT10     ; Check if ASCII character is 'E'
                LJMP ASFALEIA2          ; Emergency Stop condition
                RET

;*********************************AUTO MODE (Remote operation) ********************************

AUTOr:          LCALL WRITE_AUTOr       ; Show "REMOTE AUTO
                                        ;       PRESS RUN" message
                SETB P1.0               ; Enable Stepper motor
THERE:          LCALL GETCHR            ; Get character
                CJNE A,#'R',THERE       ; Check if ASCII character is 'R'
                SETB P1.1
COUNTER32r:     LCALL YPOROYTINA1       ; Show the number of step which is executed
                LCALL MODEr
                LCALL DELAY500ms
                LCALL DELAY500ms
                DJNZ R5, COUNTER32r
                LCALL SEND_OK           ; Send message "OK"
                LCALL EPISTROFHr        ; Table makes 32 steps to the opposite direction
                RET

;********************************** MANUAL MODE (Remote operation) ********************************

MANUALr:        LCALL WRITE_MANUALr     ; Show "REMOTE MANUAL
                                        ;       PRESS RUN" message
                SETB P1.0               ; Enable stepper motor
                CJNE R5,#32d,CONTINUEr
                SJMP LOOP60
CONTINUEr:      LCALL WRITE_PRESSRUNORRSTORAUTO  ; Show "MANUAL ** STEP
                                                 ;       RUN/AUTO/RST" message
LOOP60:         LCALL GETCHR            ; Get character
                CJNE A,#'R',LOOP70      ; Check if ASCII character is 'R'
                SJMP NEXT_STEP
LOOP70:         CJNE A,#'S',LOOP80      ; Check if ASCII character is 'S'
                LJMP RESETTING_TABLEr   ; Then execute RESETTING_TABLE operation
LOOP80:         CJNE A,#'1',LOOP60      ; Check if ASCII character is '1'
                SJMP AUTOr              ; Then switch to AUTO mode
NEXT_STEP:      LCALL DELAY50ms
                SETB P1.1
                LCALL YPOROYTINA1       ; Show the number of step which is executed
                LCALL MODEr
                DEC R5
                LCALL SEND_STEPOK       ; Send message "STEP OK"
                LCALL DELAY500ms
                LCALL DELAY500ms
                MOV A,R5
                JZ EPISTROFHr
                SJMP CONTINUEr
                RET

;************************** Table moves to initial position (Remote operation) ************************

EPISTROFHr:     LCALL DELAY5s
                LCALL WRITE_INITPOSITION
                CLR P1.1                ; Stepper motor rotates to right
                MOV R5,#32d
LOOP32r:        LCALL METAKINISIr       ; Table makes one 1,875mm-step
                DJNZ R5, LOOP32r
                LJMP CHOOSE
                RET
;*************************************************************************************************

MODEr:          CLR P2.3                ; Turn relay off
                LCALL DELAY500ms
                SETB P2.3               ; Turn relay on
                LCALL DELAY5s           ; 5 sec of delay
```

```
                LCALL METAKINISIr          ; Table makes one 1,875mm-step
                RET

;****************************** 1,9mm movement (Remote Operation) ******************************

METAKINISIr:    MOV R1,#6Ch
                MOV R2,#07h
COUNTER1900r:   LCALL DOSTEP
                JNB P2.6,ASFALEIA2         ; Check Emergency Stop switch
                JNB P3.2,SWRIGHTr          ; Check Right Terminal switch
                JNB RI,NEXT1               ; Check if a character is received
                SJMP GETCHR1               ; Get character
NEXT1:          MOV A,R1
                CLR C
                SUBB A,#01d
                MOV R1,A
                MOV A,R2
                SUBB A,#00d
                MOV R2,A
                CJNE R2,#00d, COUNTER1900r
                CJNE R1,#00d, COUNTER1900r
                RET

;************************************** Get character 'E' **************************************

GETCHR1:        MOV A,SBUF                 ; Get character
                ;ANL A,#7fh                ; Mask off 8th bit
                CLR RI                     ; Clear serial status bit
                CJNE A,#'E', NEXT1         ; Check if ASCII character is 'E'
                LJMP ASFALEIA2             ; Emergency Stop condition
                RET

;********************** Table has reached right terminal switch (Remote operation) **********************

SWRIGHTr:       LCALL DELAY50ms
                LCALL WRITE_SWRIGHTr       ; Show "REF1" message
                LCALL SEND_END             ; Send message "END"
                CLR P1.0                   ; Disable stepper motor
LOOP50:         LCALL GETCHR               ; Get character
                CJNE A,#'S', LOOP50        ; Check if ASCII character is 'S'
                LCALL CLEAR_LCD            ; Clear LCD Display
                LJMP RESETTING_TABLEr      ; Execute RESETTING_TABLE operation
                RET

;************************** RESETTING_TABLE OPERATION (Remote operation) **************************

RESETTING_TABLEr: LCALL DELAY50ms
                LCALL WRITE_RESETTINGTABLE ; Show "RESETTING TABLE
                                           ;       PLEASE WAIT..." message
                SETB P1.0                  ; Enable stepper motor
                CLR P1.1                   ; Stepper motor rotates to right
REF2r:          JNB P3.3, GORIGHTr         ; Check left terminal switch
                ACALL DOSTEP               ; Stepper motor makes one step
                JNB P2.6, ASFALEIA2        ; Check Emergency Stop switch
                JNB RI,NEXT2               ; Check if a character received
                SJMP GETCHR2               ; Get character
NEXT2:          SJMP  REF2r
GORIGHTr:       LCALL SEND_ZERO            ; Send message "ZERO"
                SETB P1.1                  ; Stepper motor rotates to right
                LCALL METAKINISI5          ; Table moves 5mm
                LJMP CHOOSE
                RET

GETCHR2:        MOV A,SBUF                 ; Get character
                ;ANL A,#7fh                ; Mask off 8th bit
                CLR RI                     ; Clear serial status bit
                CJNE A,#'E', NEXT2         ; Check if ASCII character is 'E'
                LJMP ASFALEIA2             ; Emergency Stop condition
                RET

ASFALEIA2:      LCALL DELAY50ms
                LCALL WRITE_ASFALEIA2      ; Show "SYSTEM HALTED" message
```

```
                    CLR P1.0                    ; Disable stepper motor
                    LCALL DELAY5s
                    LCALL WRITE_CHOOSE          ; Show "PLEASE CHOOSE:
                                                ;       LOCAL/REMOTE/RST" message
                    LJMP CHOOSE1
                    RET

;************************************** Get character ***********************************

GETCHR:             JNB RI, GETCHR              ; Wait until character received
                     MOV A,SBUF                 ; Get character
                    ;ANL A,#7Fh                 ; Mask off 8th bit
                     CLR RI                     ; Clear serial status bit
                     RET

;************************************** Send character ***********************************

SENDCHR:             CLR TI
                     MOV SBUF,A                 ; Initiate Xmission
XMITLOOP:           JNB TI,XMITLOOP             ; Wait until character sent
                     RET

;***************************** Show the number of step which is executed ******************

YPOROYTINA1:        LCALL CLEAR_LCD
                    MOV R1,#00d                 ;R1=TENS
                    MOV R2,#00d                 ;R2=UNITS
                    MOV R6,#00d
                    MOV A,#00d
                    MOV A,R5
                    MOV R6,A
                    MOV A,R6
LOOP200:            CLR C
                    SUBB A,#10d
                    JC STOREUNITS1
                    MOV R6,A
                    INC R1
                    SJMP LOOP200

STOREUNITS1:        MOV A,R6
                    MOV R2,A
HEXTOASCIIa:        MOV A,R1                    ;R1=TENS
                    ADD A,#30h
                    MOV R1,A
                    MOV A,R2                    ;R2=UNITS
                    ADD A,#30h
                    MOV R2,A
                    LCALL WRITE_MODE1A
                    RET

;********************** Write text "RUNNING STEP **// PLEASE WAIT..." to the LCD **********

WRITE_MODE1A:       LCALL INIT_LCD
                    LCALL CLEAR_LCD
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'U'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                     MOV A,#'I'
                     LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    MOV A,#'G'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
```

```
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,R1
                LCALL WRITE_TEXT
                MOV A,R2
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
                MOV P0,#0C0h
                CLR P2.2
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'W'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'I'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'.'
                LCALL WRITE_TEXT
                MOV A,#'.'
                LCALL WRITE_TEXT
                MOV A,#'.'
                LCALL WRITE_TEXT
                RET

;********************************** Write text "POWER ON" to the LCD **********************************

WRITE_POWERON:  LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'W'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                RET

;************************** Write text "PLEASE // PRESS RST" to the LCD **************************
```

```
WRITE_PRESSRST:     LCALL INIT_LCD
                    LCALL CLEAR_LCD
                    LCALL WRITE_TEXT
                    MOV A,#'P'
                    LCALL WRITE_TEXT
                    MOV A,#'L'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'A'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    SETB P2.2
                    CLR P2.0
                    MOV P0,#0C0h
                    CLR P2.2
                    LCALL WRITE_TEXT
                    MOV A,#'P'
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    RET

;********************** Write text "RESETTING TABLE// PLEASE WAIT..." to the LCD **********************

WRITE_RESETTINGTABLE: LCALL INIT_LCD
                      LCALL CLEAR_LCD
                      LCALL WRITE_TEXT
                      MOV A,#'R'
                      LCALL WRITE_TEXT
                      MOV A,#'E'
                      LCALL WRITE_TEXT
                      MOV A,#'S'
                      LCALL WRITE_TEXT
                      MOV A,#'E'
                      LCALL WRITE_TEXT
                      MOV A,#'T'
                      LCALL WRITE_TEXT
                      MOV A,#'T'
                      LCALL WRITE_TEXT
                      MOV A,#'I'
                      LCALL WRITE_TEXT
                      MOV A,#'N'
                      LCALL WRITE_TEXT
                      MOV A,#'G'
                      LCALL WRITE_TEXT
                      MOV A,#' '
                      LCALL WRITE_TEXT
                      MOV A,#'T'
                      LCALL WRITE_TEXT
                      MOV A,#'A'
                      LCALL WRITE_TEXT
                      MOV A,#'B'
                      LCALL WRITE_TEXT
```

```
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
                MOV P0,#0C0h
                CLR P2.2
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'W'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'I'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'.'
                LCALL WRITE_TEXT
                MOV A,#'.'
                LCALL WRITE_TEXT
                MOV A,#'.'
                LCALL WRITE_TEXT
                RET

;************************ Write text "SYSTEM HALTED// PRESS RST" to the LCD ************************

WRITE_ASFALEIA:  LCALL INIT_LCD
                LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'Y'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'M'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'H'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'D'
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
```

```
                    MOV P0,#0C0h
                    CLR P2.2
                    LCALL WRITE_TEXT
                    MOV A,#'P'
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    RET

;********************* Write text "PLEASE CHOOSE// LOCAL/ REMOTE/RST" to the LCD *********************

WRITE_CHOOSE:       LCALL INIT_LCD
                    LCALL CLEAR_LCD
                    LCALL WRITE_TEXT
                    MOV A,#'P'
                    LCALL WRITE_TEXT
                    MOV A,#'L'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'A'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'C'
                    LCALL WRITE_TEXT
                    MOV A,#'H'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#':'
                    LCALL WRITE_TEXT
                    SETB P2.2
                    CLR P2.0
                    MOV P0,#0C0h
                    CLR P2.2
                    LCALL WRITE_TEXT
                    MOV A,#'L'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'C'
                    LCALL WRITE_TEXT
                    MOV A,#'A'
                    LCALL WRITE_TEXT
                    MOV A,#'L'
                    LCALL WRITE_TEXT
                    MOV A,#'/'
```

```
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'M'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'/'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                RET

;******************** Write text "LOCAL OPERATION// CHOOSE MAN/AUTO" to the LCD **********************

WRITE_LOCAL:    LCALL INIT_LCD
                LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'C'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'I'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
                MOV P0,#0C0h
                CLR P2.2
                LCALL WRITE_TEXT
                MOV A,#'C'
                LCALL WRITE_TEXT
                MOV A,#'H'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'S'
```

```
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#':'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'M'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                MOV A,#'/'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'U'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                RET

;***************************** Write text "REMOTE OPERATION" to the LCD *****************************

WRITE_REMOTE:   LCALL INIT_LCD
                LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'M'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT'
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'I'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                RET


;************************** Write text "LOCAL MANUAL// PRESS RUN" to the LCD ************************

WRITE_MODE0:    LCALL INIT_LCD
                 LCALL CLEAR_LCD
                 LCALL WRITE_TEXT
                 MOV A,#'L'
```

```
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'C'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'M'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                MOV A,#'U'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
                MOV P0,#0C0h
                CLR P2.2
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'U'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                RET

;************************** Write text "LOCAL AUTO// PRESS RUN" to the LCD **************************

WRITE_MODE1:    LCALL INIT_LCD
                LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#'O'
                LCALL WRITE_TEXT
                MOV A,#'C'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'U'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'O'
```

```
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
                MOV P0,#0C0h
                CLR P2.2
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'U'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                RET

;****************************** Write text "REF1// PRESS RST" to the LCD ******************************

WRITE_SWRIGHT:  LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'F'
                LCALL WRITE_TEXT
                MOV A,#'1'
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
                MOV P0,#0C0h
                CLR P2.2
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                RET


;****************************** Write text "MOVING TO//INIT POSITION" to the LCD ******************************

WRITE_INITPOSITION:  LCALL INIT_LCD
                     LCALL CLEAR_LCD
                     LCALL WRITE_TEXT
                     MOV A,#'M'
```

```
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'V'
                    LCALL WRITE_TEXT
                    MOV A,#'I'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    MOV A,#'G'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    SETB P2.2
                    CLR P2.0
                    MOV P0,#0C0h
                    CLR P2.2
                    LCALL WRITE_TEXT
                    MOV A,#'I'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    MOV A,#'I'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'P'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'I'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#'I'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    RET

;*****************************************************************************************************

WRITE_PRESSRUNORRSTORAUTO: LCALL INIT_LCD
                           LCALL CLEAR_LCD
                           LCALL YPOROYTINA2
                           RET

YPOROYTINA2:               LCALL CLEAR_LCD
                           MOV R1,#00d              ;R1=TENS
                           MOV R2,#00d              ;R2=UNITS
                           MOV R6,#00d
                           MOV A,#00d
                           MOV A,R5
                           MOV R6,A
                           MOV A,R6
LOOP300:                   CLR C
                    SUBB A,#10d
                    JC STOREUNITS2
                    MOV R6,A
                    INC R1
                    SJMP LOOP300
```

```
STOREUNITS2:        MOV A,R6
                    MOV R2,A
HEXTOASCIIb:        MOV A,R1                    ;R1=TENS
                    ADD A,#30h
                    MOV R1,A
                    MOV A,R2   ;R2=UNITS
                    ADD A,#30h
                    MOV R2,A
                    LCALL WRITE_MODE0A
                    RET

;********************** Write text "MANUAL ** STEP//RUN/AUTO/RST?" to the LCD **********************

WRITE_MODE0A:       LCALL INIT_LCD
                    LCALL CLEAR_LCD
                    LCALL WRITE_TEXT
                    MOV A,#'M'
                    LCALL WRITE_TEXT
                    MOV A,#'A'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    MOV A,#'U'
                    LCALL WRITE_TEXT
                    MOV A,#'A'
                    LCALL WRITE_TEXT
                    MOV A,#'L'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,R1
                    LCALL WRITE_TEXT
                    MOV A,R2
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'P'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    SETB P2.2
                    CLR P2.0
                    MOV P0,#0C0h
                    CLR P2.2
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'U'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    MOV A,#'/'
                    LCALL WRITE_TEXT
                    MOV A,#'A'
                    LCALL WRITE_TEXT
                    MOV A,#'U'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'/'
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
```

```
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#'?'
                    LCALL WRITE_TEXT
                    RET

;************************* Write text "REMOTE AUTO//PRESS RUN" to the LCD ***************************

WRITE_AUTOr:        LCALL INIT_LCD
                    LCALL CLEAR_LCD
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'M'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'A'
                    LCALL WRITE_TEXT
                    MOV A,#'U'
                    LCALL WRITE_TEXT
                    MOV A,#'T'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
                    SETB P2.2
                    CLR P2.0
                    MOV P0,#0C0h
                    CLR P2.2
                    LCALL WRITE_TEXT
                    MOV A,#'P'
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#'S'
                    LCALL WRITE_TEXT
                    MOV A,#' '
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'U'
                    LCALL WRITE_TEXT
                    MOV A,#'N'
                    LCALL WRITE_TEXT
                    RET

;************************* Write text "REMOTE MANUAL//PRESS RUN" to the LCD ************************

WRITE_MANUALr:      LCALL INIT_LCD
                    LCALL CLEAR_LCD
                    LCALL WRITE_TEXT
                    MOV A,#'R'
                    LCALL WRITE_TEXT
                    MOV A,#'E'
                    LCALL WRITE_TEXT
                    MOV A,#'M'
                    LCALL WRITE_TEXT
                    MOV A,#'O'
                    LCALL WRITE_TEXT
```

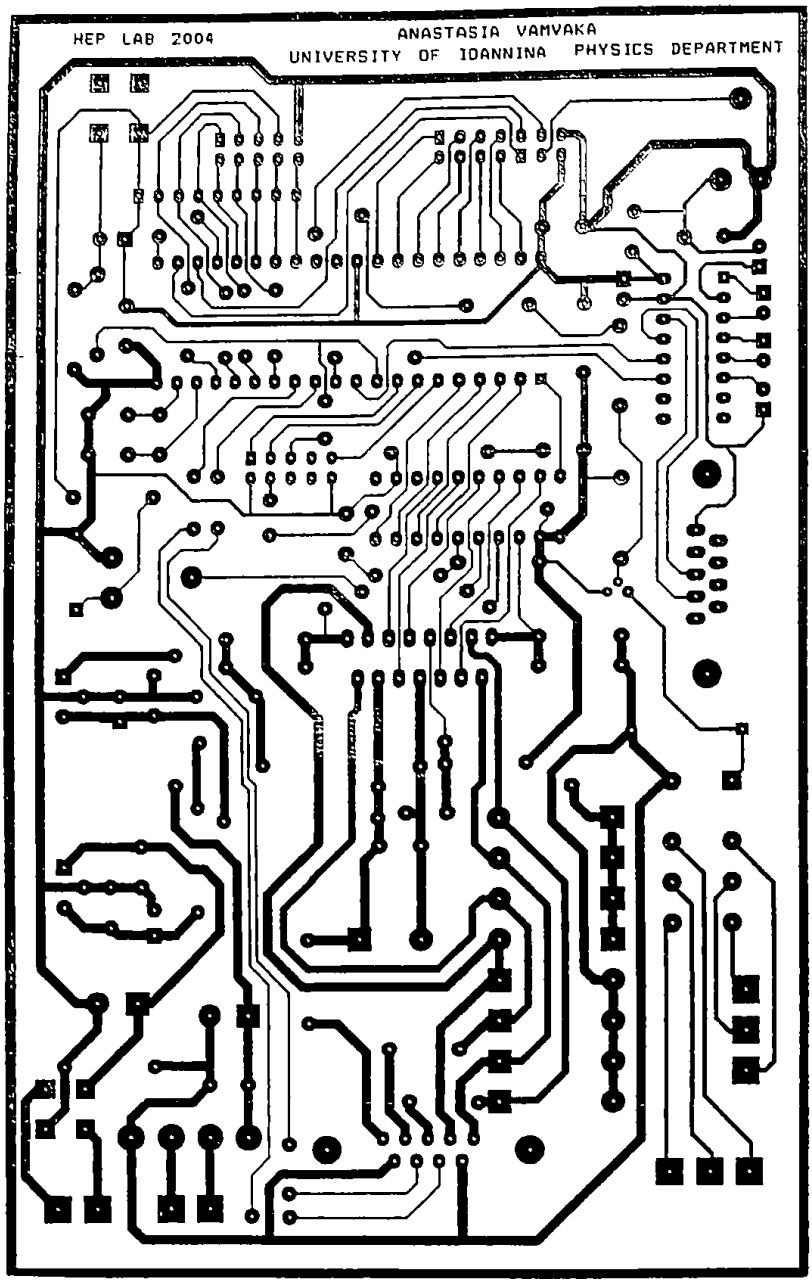
```
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'M'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                MOV A,#'U'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                SETB P2.2
                CLR P2.0
                MOV P0,#0C0h
                CLR P2.2
                LCALL WRITE_TEXT
                MOV A,#'P'
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'U'
                LCALL WRITE_TEXT
                MOV A,#'N'
                LCALL WRITE_TEXT
                RET

;********************************** Write text "REF1" to the LCD **********************************

WRITE_SWRIGHTr:  LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'R'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'F'
                LCALL WRITE_TEXT
                MOV A,#'1'
                LCALL WRITE_TEXT
                RET

;******************************** Write text "SYSTEM HALTED" to the LCD ********************************

WRITE_ASFALEIA2:  LCALL INIT_LCD
                LCALL CLEAR_LCD
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'Y'
                LCALL WRITE_TEXT
                MOV A,#'S'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
```

```
                LCALL WRITE_TEXT
                MOV A,#'M'
                LCALL WRITE_TEXT
                MOV A,#' '
                LCALL WRITE_TEXT
                MOV A,#'H'
                LCALL WRITE_TEXT
                MOV A,#'A'
                LCALL WRITE_TEXT
                MOV A,#'L'
                LCALL WRITE_TEXT
                MOV A,#'T'
                LCALL WRITE_TEXT
                MOV A,#'E'
                LCALL WRITE_TEXT
                MOV A,#'D'
                LCALL WRITE_TEXT
                RET

;************************************ Initializing the LCD Display ************************************

INIT_LCD:       SETB P2.2
                CLR P2.0
                MOV P0,#38h
                CLR P2.2
                LCALL WAIT_LCD
                SETB P2.2
                CLR P2.0
                MOV P0,#0Eh
                CLR P2.2
                LCALL WAIT_LCD
                SETB P2.2
                CLR P2.0
                MOV P0,#06h
                LCALL WAIT_LCD
                RET

;*****************************************************************************************************

WAIT_LCD:       SETB P2.2
                CLR P2.0
                SETB P2.1
                MOV P0,#0FFh
                MOV A,P0
                JB ACC.7, WAIT_LCD
                CLR P2.2
                CLR P2.1
                RET

;************************************Writing text to the LCD ************************************

WRITE_TEXT:     SETB P2.2
                SETB P2.0
                MOV P0,A
                CLR P2.2
                LCALL WAIT_LCD
                RET

;************************************ Clearing the LCD Display ************************************

CLEAR_LCD:      SETB P2.2
                CLR P2.0
                MOV P0,#01h
                CLR P2.2
                RET

;************************************ "Send message "ZERO" ************************************

SEND_ZERO:      MOV A,#5AH              ;Z
                LCALL SENDCHR
                MOV A,#45H              ;E
```

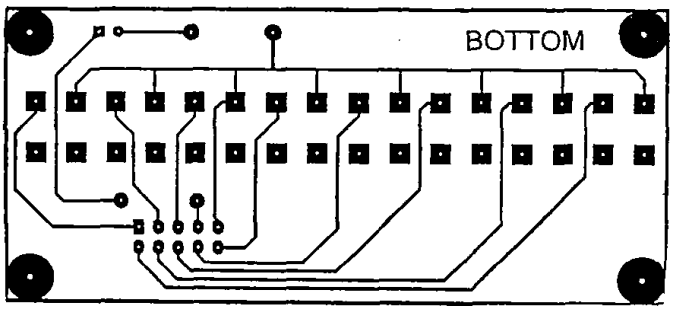
```
                    LCALL SENDCHR
                    MOV A,#52H                  ;R
                    LCALL SENDCHR
                    MOV A,#4FH                  ;0
                    LCALL SENDCHR
                    RET

;************************************ Send message "STEP OK!" ************************************

SEND_STEPOK:        MOV A,#53H                  ;S
                    LCALL SENDCHR
                    MOV A,#54H                  ;T
                    LCALL SENDCHR
                    MOV A,#45H                  ;E
                    LCALL SENDCHR
                    MOV A,#50H                  ;P
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#4FH                  ;O
                    LCALL SENDCHR
                    MOV A,#4BH                  ;K
                    LCALL SENDCHR
                    MOV A,#21H                  ;!
                    LCALL SENDCHR
                    RET

;************************************ Send message "OK" ************************************

SEND_OK:            MOV A,#4FH                  ;O
                    LCALL SENDCHR
                    MOV A,#4BH                  ;K
                    LCALL SENDCHR
                    MOV A,#21H                  ;!
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    RET

;************************************ Send message "END" ************************************
SEND_END:           MOV A,#45H                  ;E
                    LCALL SENDCHR
                    MOV A,#4EH                  ;N
                    LCALL SENDCHR
                    MOV A,#44H                  ;D
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    MOV A,#20H                  ;SPACE
                    LCALL SENDCHR
                    RET
```

# Παράρτημα Ε

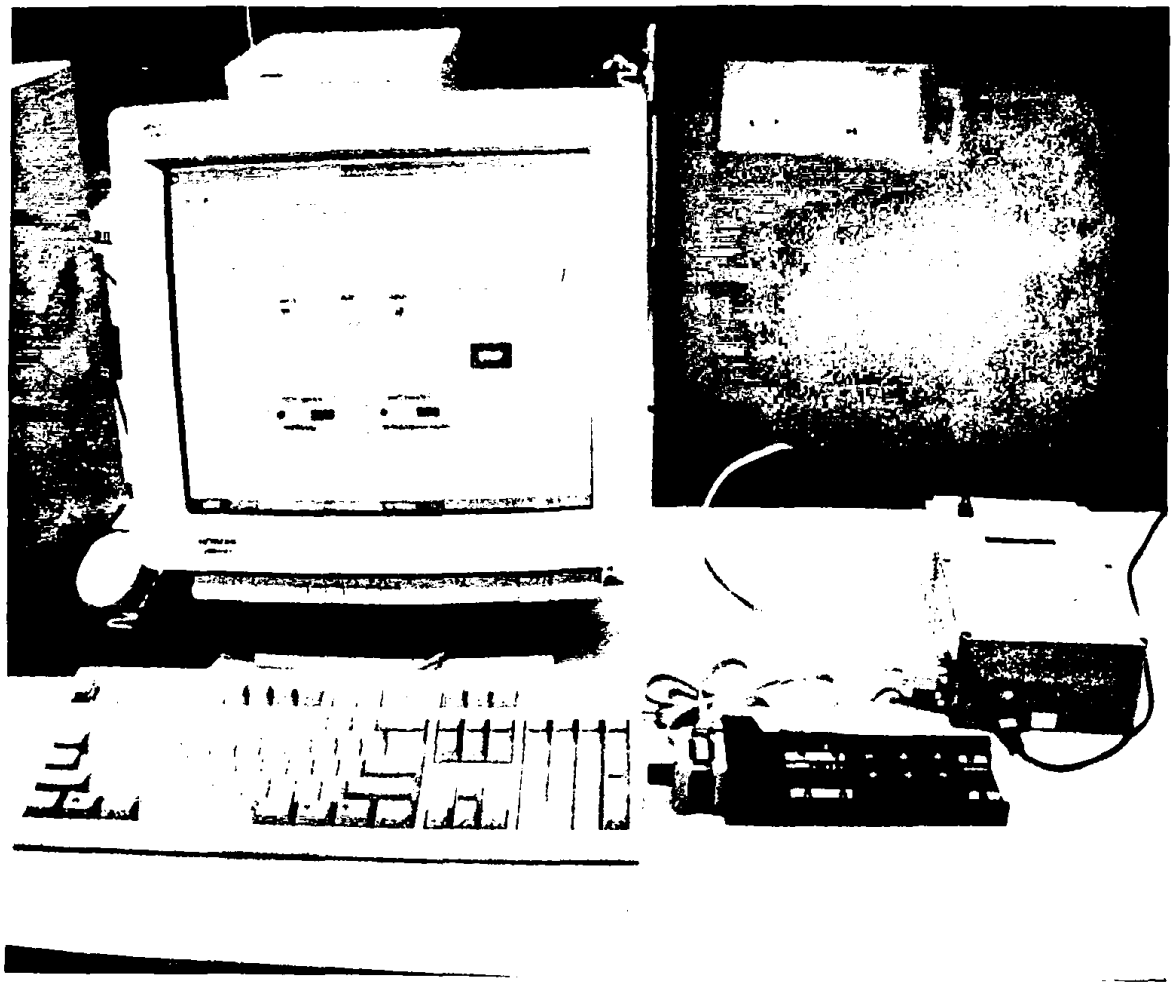
## Τυπωμένα κυκλώματα και φωτογραφίες



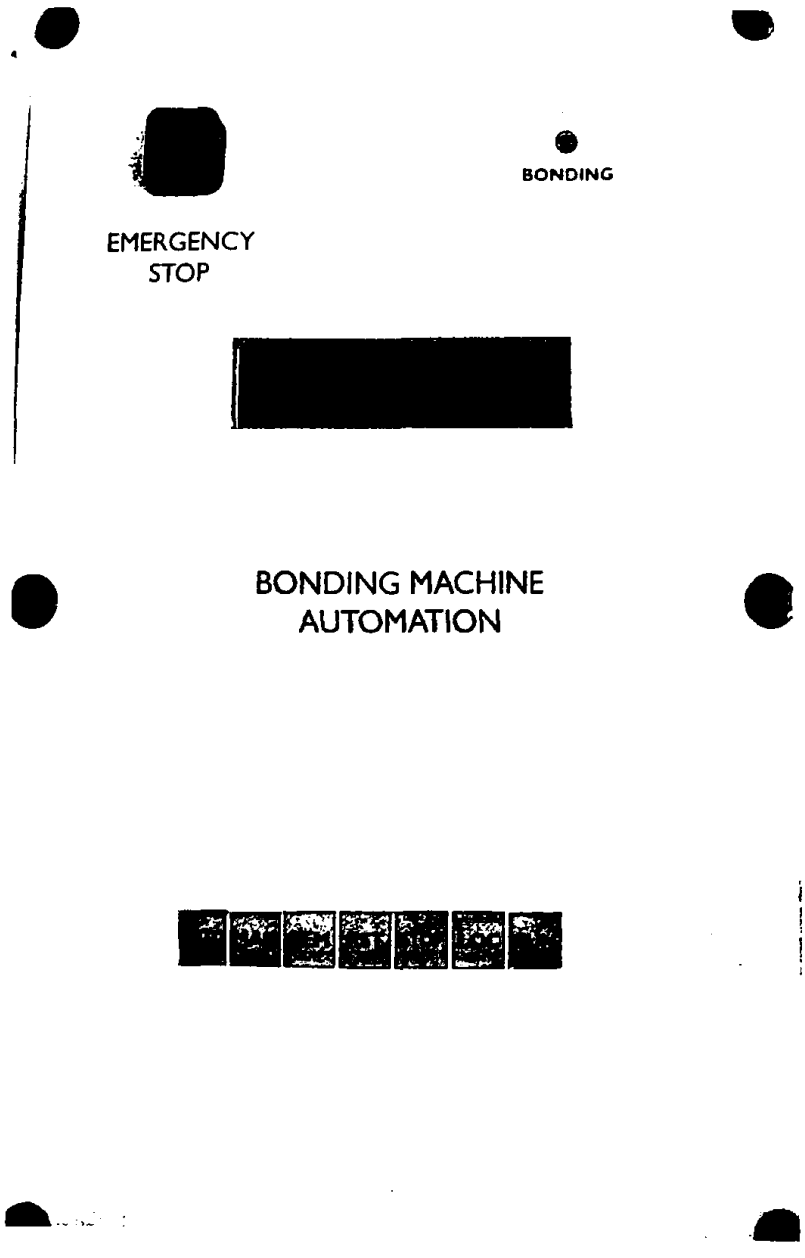

**Σχήμα Ε-1:Το τυπωμένο κύκλωμα της κάτω όψης της πλακέτας οδήγησης του βηματικού Κινητήρα**



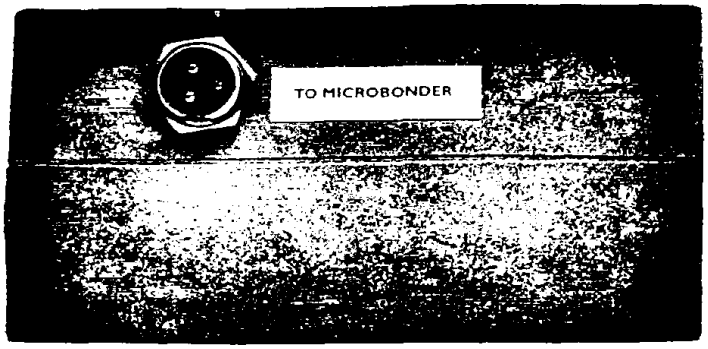

**Σχήμα Ε-2: Το τυπωμένο κύκλωμα της κάτω όψης του χειριστηρίου της συσκευής**



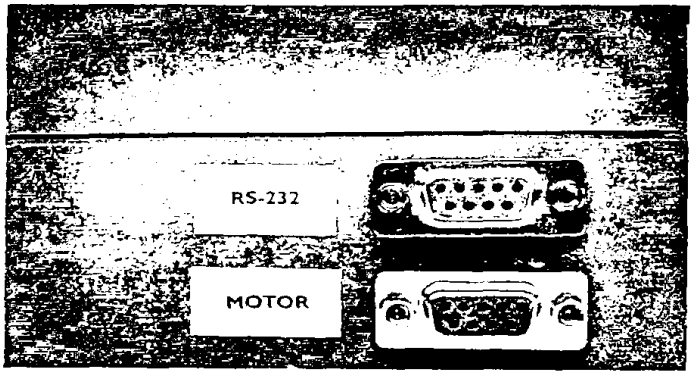
**Φ1: Γενική φωτογραφία του συστήματος**



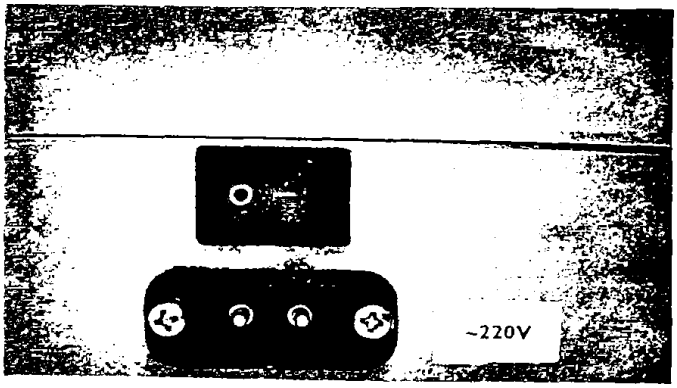

**Φ2: Η πρόσοψη της μονάδας**



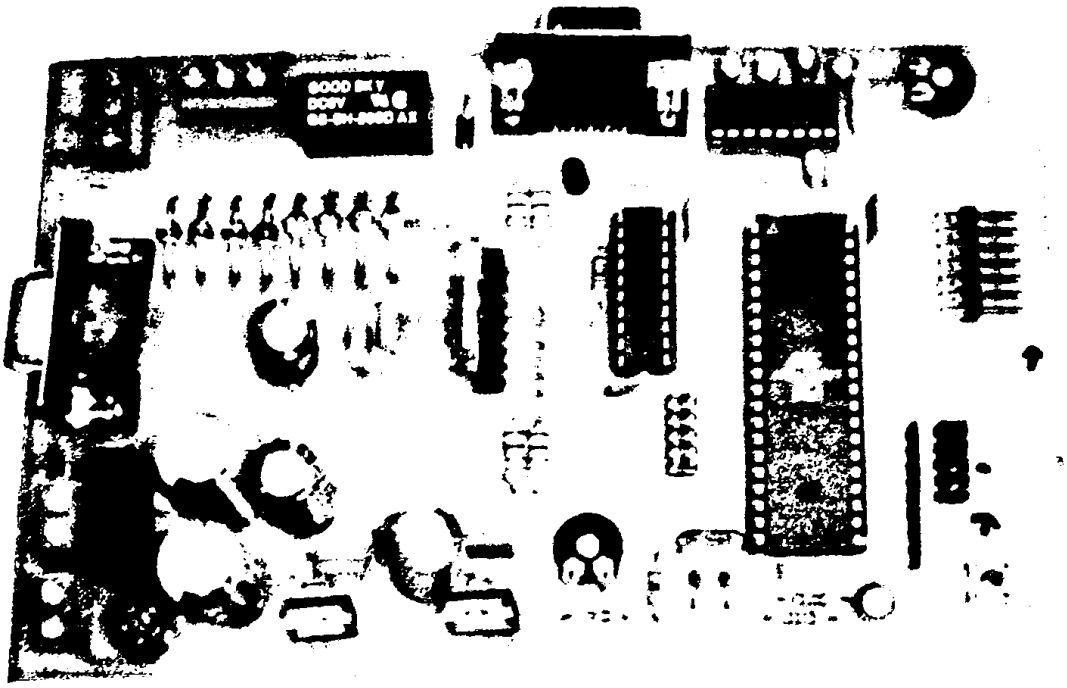

**Φ3: Ακροδέκτης σύνδεσης συσκευής μικροσυγκόλλησης με τη μονάδα της εφαρμογής**




**Φ4 : Ακροδέκτες σύνδεσης τράπεζας ακριβείας και επικοινωνίας με τον Η/Υ**




**Φ5 :.Διακόπτης ON/OFF και τροφοδοσία της μονάδας**



**Φ6: Η κύρια πλακέτα του συστήματος**