Πανεπιστήμιο Πειραιώς – Τμήμα Πληροφορικής
Πρόγραμμα Μεταπτυχιακών Σπουδών
«Πληροφορική»

Μεταπτυχιακή Διατριβή

Τίτλος Διατριβής	**Σχεδίαση Και Κατασκευή Συστήματος Ελέγχου Φωτισμού με Μικροελεγκτές AVR.**
Ονοματεπώνυμο Φοιτητή	**Γεώργιος Τσιλιγιάννης**
Πατρώνυμο	**Σπυρίδων**
Αριθμός Μητρώου	**ΜΠΠΛ/07060**
Επιβλέπων	**Δημήτρης Γκιζόπουλος, Αναπληρωτής Καθηγητής**

Ημερομηνία Παράδοσης **Οκτώβριος 2010**

Τριμελής Εξεταστική Επιτροπή

Δημήτρης Γκιζόπουλος Αναπληρωτής Καθηγητής	Μιχάλης Ψαράκης Λέκτορας	Θεμιστοκλής παναγιωτόπουλος Καθηγητής

ΠΡΟΛΟΓΟΣ

Αντικείμενο της παρούσας εργασίας είναι η σχεδίαση και η κατασκευή ενός ελεγκτή PID και του επενεργητή (actuator) για τον αυτόματο έλεγχο φωτισμού ενός χώρου με λαμπτήρα πυρακτώσεως με χρήση μικροελεγκτών AVR της εταιρίας Atmel. Τα τελευταία χρόνια η επικράτηση της ψηφιακής τεχνολογίας είναι σχεδόν ολοκληρωτική, σε όλους σχεδόν τους τομείς όπου χρησιμοποιούνται ηλεκτρονικά εξαρτήματα. Το ευρύ πεδίο εφαρμογών των μικροελεγκτών σε πολλές πτυχές της σύγχρονης εποχής, σε συνδυασμό με το προσωπικό ενδιαφέρον για το συγκεκριμένο αντικείμενο αποτέλεσαν το εφαλτήριο για την ανάπτυξη του παραπάνω συστήματος.

Κατά την εκπόνησή της εργασίας, κατανόησα όχι μόνο τον τρόπο που πρέπει να λειτουργεί ένα σύστημα αυτομάτου ελέγχου ως σύστημα αυτοματισμού αλλά και ο ελεγκτής ως μια συσκευή που αποτελείται από επιμέρους τμήματα τα οποία πρέπει να συνεργαστούν για το συνολικό αποτέλεσμα.

Η εργασία αυτή περιλαμβάνει μια τομή ανάμεσα σε υλικό (hardware) και λογισμικό (software) παρέχοντας έτσι τριβή και στα δύο αντικείμενα καθώς και στον τρόπο που αυτά συνδυάζονται. Ήρθα σε επικοινωνία με εταιρίες, μπήκα στην διαδικασία αξιολόγησης των εξαρτημάτων που επρόκειτο να χρησιμοποιήσω, λαμβάνοντας υπ' όψιν παράγοντες όπως κόστος, αξιοπιστία και παρεχόμενο documentation για κάθε εξάρτημα. Επίσης, στον τομέα του λογισμικού η ανάπτυξη του κώδικα, η αποσφαλμάτωση και διόρθωσή του αποτέλεσαν εξίσου επίπονες αλλά και δημιουργικές διαδικασίες.

Εκτός των γνώσεων σε θέματα ηλεκτρονικών προγραμματισμού μικροελεγκτών και ανάπτυξης εφαρμογής σε ηλεκτρονικό υπολογιστή απαιτήθηκαν γνώσεις συστημάτων αυτομάτου ελέγχου. Η εμπειρία που αποκτήθηκε κρίνεται ως σημαντική κυρίως λόγω της σχέσης μεταξύ θεωρητικής προσέγγισης και πραγματικής ανάπτυξης του θέματος.

ABSTRACT

The object of this work is to design and manufacture of a PID controller and actuator for automatic control of lighting an area with incandescent lamp using AVR microcontroller company Atmel. In recent years the prevalence of digital technology is nearly complete in almost all areas where used electronics. The wide scopes of applications of microcontrollers into many aspects of modern times, coupled with an interest in this subject have been the springboard for the development of this system.

During the preparation work, understanding not only how to operate a system of automatic control in automation and controller as a device that consists of different parts that must work together for the overall result.

This work involves a balance between hardware and software thus providing friction on both objects and the way they are combined. I came in contact with companies, I was in the process of evaluating the components going to use, taking into account such factors as cost, reliability and supplied documentation for each component. Also in the field of software development of the code, debugging and correction were equally strenuous and creative processes.

Except for knowledge in computer programming microcontrollers and application development in computer skills required control systems. The experience is as important mainly because of the relationship between theoretical and pragmatic approach to develop the subject.

Ευχαριστίες

Στο σημείο αυτό δεν θα μπορούσα να μην ευχαριστήσω:

Τον Αναπληρωτή Καθηγητή του πανεπιστήμιου Πειραιά του τμήματος πληροφορικής Κ. Δημήτρη Γκιζόπουλο, επιβλέπων καθηγητή της παρούσας μεταπτυχιακής διατριβής, που μου εμπιστεύθηκε την παρούσα εργασία και μου έδωσε την ευκαιρία να αποκτήσω μια πολύτιμη γνώση και εμπειρία. Επιπλέον για τις υποδείξεις, συμβουλές και εν γένη καθοδήγηση που μου παρείχε κατά την μελέτη, υλοποίηση και παρουσίαση της εργασίας μου.

Τέλος θα ήθελα να εκφράσω τις ευχαριστίες μου στον άνθρωπο που μετά από δική του παρότρυνση παρακολούθησα αυτό το μεταπτυχιακό πρόγραμμα σπουδών και για την στήριξη που μου έκανε καθόλη την διάρκεια του προγράμματος.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΠΙΝΑΚΑΣ ΕΙΚΟΝΩΝ

1 ΚΕΦΑΛΑΙΟ 1

Ιστορία των Μικροεπεξεργαστών – Μικροελεγκτών

1.1 Εισαγωγή

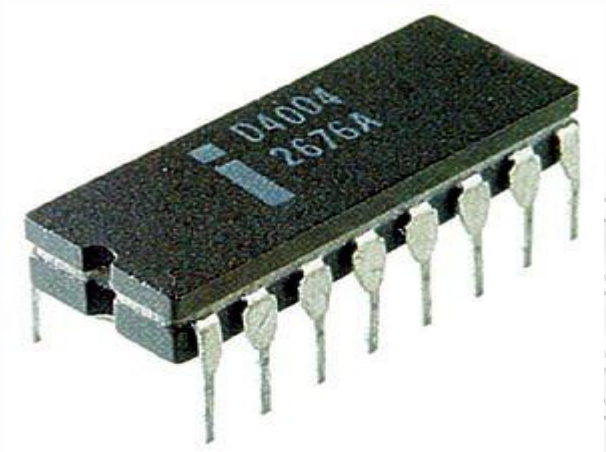
Στην σχετικά σύντομη διάρκεια ζωής του, που δεν φτάνει τα 40 χρόνια, ο μικροεπεξεργαστής (microprocessor) έχει κάνει τεράστιες προόδους. Η απόδοσή του έχει βελτιωθεί αισθητά αφού διπλασιάζεται περίπου κάθε 18 μήνες. Στις τελευταίες τρεις δεκαετίες οι μικροεπεξεργαστές είναι υπεύθυνοι για την έμπνευση και τη δημιουργία μερικών από τις μεγαλύτερες καινοτομίες στα συστήματα υπολογιστών. Αυτές οι καινοτομίες περιλαμβάνουν τους embedded μικροελεγκτές, τους προσωπικούς υπολογιστές, τους σύγχρονους σταθμούς εργασίας, συσκευές χειρός και κινητές συσκευές (όπως τους επεξεργαστές των κινητών τηλεφώνων), servers (εξυπηρετητές) εφαρμογών και αρχείων, web servers για το Internet, υπερυπολογιστές χαμηλού κόστους και ευρείας κλίμακας, δίκτυα υπολογιστών. Σήμερα πωλούνται κάθε χρόνο πάνω από 100 εκατομμύρια μικροεπεξεργαστές στις αγορές κινητής τηλεφωνίας, επεξεργαστές για εργασίες γραφείου και server. Αν συνυπολογίσουμε και τους embedded μικροεπεξεργαστές και μικροελεγκτές τότε ο συνολικός αριθμός των μικροεπεξεργαστών που πωλούνται κάθε χρόνο υπερβαίνει κατά πολύ το ένα δισεκατομμύριο.

Οι μικροεπεξεργαστές είναι επεξεργαστές συνόλου εντολών (instruction set processors, ISPs). Ένας ISP εκτελεί εντολές ενός προκαθορισμένου συνόλου εντολών (ή αλλιώς σετ εντολών). Η λειτουργικότητα ενός μικροεπεξεργαστή εξαρτάται πλήρως από το σύνολο εντολών που είναι ικανός να εκτελέσει. Όλα τα προγράμματα που τρέχουν σε έναν μικροεπεξεργαστή κωδικοποιούνται σε αυτό το σύνολο εντολών. Αυτό το προκαθορισμένο σύνολο εντολών ονομάζεται επίσης αρχιτεκτονική συνόλου εντολών (Instruction Set Architecture, ISA). Το ISA χρησιμεύει ως μία διασύνδεση ανάμεσα στο λογισμικό (software) και το υλικό (hardware), δηλαδή ανάμεσα στα προγράμματα και στους επεξεργαστές. Σε ορολογία της μεθοδολογίας σχεδίασης επεξεργαστών, το ISA είναι ο καθορισμός της σχεδίασης ενώ ο επεξεργαστής ή ο ISP είναι η υλοποίηση της σχεδίασης.

1.2 Η εξέλιξη των μικροεπεξεργαστών

Στης 15 Νοεμβρίου του έτους 1971 η Intel παρουσίασε τον πρώτο μικροεπεξεργαστή σε μορφή dual in-line package (DIP) τον Intel 4004 και ήταν ο πρώτος εμπορικά διαθέσιμος επεξεργαστής υπολογιστών. Ο Intel 4004 ήταν ένας επεξεργαστής 4 bit που αποτελείται από περίπου 2300 τρανζίστορ με συχνότητα ρολογιού λίγο πάνω από τα 100 KHz. Η αρχική του εφαρμογή ήταν η δημιουργία αριθμομηχανών.

Το έτος 2011 θα σημάνει την τεσσαρακοστή επέτειο της γέννησης των μικροεπεξεργαστών. Μικροεπεξεργαστές υψηλού-τέλους (high-end microprocessors), οι οποίοι αποτελούνται από εκατοντάδες εκατομμύρια τρανζίστορ με συχνότητα ρολογιού που αγγίζει τα 4 GHz και πολλαπλούς πυρήνες, και είναι οι δομικές μονάδες για τα συστήματα υπερυπολογιστών και πανίσχυρων συστημάτων πελάτη-εξυπηρετητή (client-server systems) που υπάρχουν στο Internet. Στη σημερινή εποχή η συχνότητα ρολογιού των μικροεπεξεργαστών υπερβαίνει τα 3 GHz και τους τέσσερις πυρήνες και αποτελείται από αρκετές δεκάδες εκατομμύρια τρανζίστορ.



Εικόνα 1: Intel D4004

Οι τέσσερις σχεδόν δεκαετίες της ιστορίας των μικροεπεξεργαστών λένε μία πραγματικά αξιοπρόσεκτη ιστορία όσων αφορά την τεχνολογική πρόοδο στην βιομηχανία υπολογιστών. Αυτή η πρόοδος φαίνεται καλύτερα στον πίνακα 1 που ακολουθεί:

	1970-1980	**1980-1990**	**1990-2000**	**2000-2010**
Πλήθος τρανζίστορ (transistor count)	2K-100K	100K-1M	1M-100M	100M-2B
Συχνότητα ρολογιού (clock frequency)	0.1-3 MHz	3-30 MHz	30 MHz-1GHz	1-5 GHz
Εντολές/κύκλο (Instructions/cycle, IPC)	0.1	0.1-0.9	0.9-1.9	1.9-2.9

Πίνακας 1: Οι δεκαετίες εξέλιξης των μικροεπεξεργαστών.

Η εξέλιξη των μικροεπεξεργαστών έχει κατά βάση ακολουθήσει τον περίφημο νόμο του Moore (Moore's law), που παρατηρήθηκε από τον Gordon Moore το 1965 και σύμφωνα με τον οποίο ο αριθμός των συσκευών που μπορούν να ολοκληρωθούν (με την έννοια της ολοκλήρωσης στην ορολογία της σχεδίασης ολοκληρωμένων κυκλωμάτων) σε ένα απλό κομμάτι πυριτίου διπλασιάζεται κάθε 18-24 μήνες. Σε λιγότερο από 30 χρόνια μάλιστα, ο αριθμός των τρανζίστορ σε ένα chip μικροεπεξεργαστή έχει αυξηθεί κατά πάνω από 4 τάξεις μεγέθους όπως φαίνεται και στον πίνακα. Στην ίδια χρονική περίοδο, η απόδοση του μικροεπεξεργαστή έχει αυξηθεί κατά πάνω από 5 τάξεις μεγέθους. Επίσης, στις τελευταίες δύο δεκαετίες η απόδοση του μικροεπεξεργαστή διπλασιάζεται κάθε 18 μήνες, ή διαφορετικά εκατονταπλασιάζεται (x100) σε κάθε δεκαετία. Αυτή η εντυπωσιακή βελτίωση της απόδοσης των μικροεπεξεργαστών αποτελεί μοναδικό φαινόμενο και δεν έχει συναντηθεί σε καμία άλλη βιομηχανία.

Κατά τη διάρκεια της πρώτης δεκαετίας η έλευση του 4bit μικροεπεξεργαστή οδήγησε γρήγορα στην παρουσίαση του 8bit μικροεπεξεργαστή. Αυτοί εξελίχθηκαν σε μικροελεγκτές που χρησιμοποιήθηκαν σε embedded εφαρμογές, από μηχανές πλυσίματος μέχρι ανελκυστήρες και μηχανές jet. Ο 8bit μικροεπεξεργαστής έγινε επίσης η καρδιά μίας νέας διάσημης υπολογιστικής πλατφόρμας, γνωστής ως προσωπικός υπολογιστής (Personal Computer, PC) και εισήγαγε την εποχή των PCs.

Η δεκαετία του 1980 υπήρξε μάρτυρας πολλών σημαντικών αλλαγών στην αρχιτεκτονική και την μικροαρχιτεκτονική των 32bit μικροεπεξεργαστών. Τα προβλήματα που αφορούν τη σχεδίαση του συνόλου εντολών έγιναν το σημαντικότερο αντικείμενο των ακαδημαϊκών και βιομηχανικών ερευνών. Το pipelining των εντολών και οι γρήγορες κρυφές μνήμες (cache memories) έγιναν αναντικατάστατες τεχνικές μικροαρχιτεκτονικής.

Κατά τη διάρκεια της δεκαετίας του 1990 οι μικροεπεξεργαστές έγιναν η πιο πανίσχυρη και δημοφιλής μορφή υπολογιστών. Η συχνότητα ρολογιού των ταχύτερων μικροεπεξεργαστών ξεπέρασαν τις συχνότητες ρολογιού των ταχύτερων υπερυπολογιστών. Οι προσωπικοί υπολογιστές και οι σταθμοί εργασίας έγιναν σημαντικά και αναντικατάστατα εργαλεία της παραγωγικότητας και των επικοινωνιών. Χρησιμοποιήθηκαν ιδιαίτερα επιθετικές τεχνικές μικροαρχιτεκτονικής για την επίτευξη απόδοσης μικροεπεξεργαστών σε επίπεδα που δεν είχαν επιτευχθεί ποτέ μέχρι τότε. Επίσης έγιναν δημοφιλείς βαθιά pipelined μηχανές (δηλαδή μηχανές με σωληνώσεις πολλών σταδίων), ικανές να επιτύχουν εξαιρετικά υψηλές συχνότητες ρολογιού και να εκτελέσουν πολλές εντολές ανά κύκλο ρολογιού. Ακόμη χρησιμοποιήθηκαν πολύ επιθετικές τεχνικές πρόβλεψης διακλαδώσεων όπως επίσης και η out-of-order εκτέλεση των εντολών ώστε να μειωθούν στο ελάχιστο οι χαμένοι κύκλοι ρολογιού λόγω καθυστερήσεων του pipeline (pipeline stalls).

Πλέον βρισκόμαστε ήδη στα τέλη της τέταρτης δεκαετίας των μικροεπεξεργαστών και φαίνεται ότι η ροπή αυτή δεν δείχνει σημάδια ελάττωσης. Οι περισσότεροι τεχνολόγοι συμφωνούν ότι ο νόμος του Moore θα συνεχίσει να ισχύει για τουλάχιστον 10 με 15 χρόνια. Η εστίαση που τις πρώτες τρεις δεκαετίες ήταν μόνο στον παραλληλισμό στο επίπεδο της εντολής (Instruction-level parallelism, ILP) έχει ήδη περάσει στον παραλληλισμό στο επίπεδο του νήματος (Thread-level parallelism, TLP) και στον παραλληλισμό στο επίπεδο της μνήμης (Memory-level parallelism, MLP). Επίσης πολλά ζητήματα που αφορούσαν παραδοσιακά την «μακροαρχιτεκτονική» γίνονται ζητήματα που αφορούν τη μικροαρχιτεκτονική (δηλαδή χαρακτηριστικά που αφορούσαν μεγάλα συστήματα τώρα υλοποιούνται πάνω σε ένα chip). Η κατανάλωση ισχύος έχει γίνει επίσης ένας πολύ σημαντικός παράγοντας της απόδοσης και απαιτούνται νέες λύσεις σε όλα τα επίπεδα της ιεραρχίας σχεδίασης, που περιλαμβάνουν την διαδικασία παραγωγής, τη λογική σχεδίαση, τη σχεδίαση σε επίπεδο μικροαρχιτεκτονικής και το rum-time περιβάλλον του λογισμικού, έτσι ώστε να διατηρηθούν τα ίδια επίπεδα βελτίωσης της απόδοσης που επιτεύχθηκαν κατά τις προηγούμενες τρεις δεκαετίες.

1.3 Χρονολογίες–Σταθμοί Στην Ιστορία Των Μικροεπεξεργαστών.

Χρονολογίες–σταθμοί στην ιστορία των μικροεπεξεργαστών μπορούν να θεωρηθούν οι παρακάτω:

- 1971: Η Intel παρουσιάζει τον πρώτο μικροεπεξεργαστή, τον 4004. Έχει δίαυλο δεδομένων πλάτους 4 bit, κατασκευάζεται με 2.300 τρανζίστορ και έχει συχνότητα λειτουργίας 108 kHz. Μέσα στην επόμενη χρονιά εμφανίζεται ο διάδοχος του ο 8008.
- 1974: Εμφάνιση του 8–bit μικροεπεξεργαστή Intel 8080 ως αποτέλεσμα της εξέλιξης του 8008. Έχει συχνότητα λειτουργίας 2 MHz και η κατασκευή του απαιτεί 6.000 τρανζίστορ. Απάντηση της Zilog με τον Z80 και της Motorola με τον 6800, ο οποίος έχει 4.000 τρανζίστορ και ίδια συχνότητα λειτουργίας με τον 8080.
- 1975: Η Intel αναβαθμίζει τον 8080 σε 8085.
- 1978: Εμφανίζεται οι πρώτοι 16–bit μικροεπεξεργαστές (δηλαδή ο δίαυλος δεδομένων τους έχει πλάτος 16 bit). Η Intel παρουσιάζει τον 8086/8088, του οποίου η συχνότητα λειτουργίας έχει ανέβει πλέον στα 10 MHz και η κατασκευή του απαιτεί 29.000 τρανζίστορ. Η Motorola εμφανίζει τον 68000 με συχνότητα λειτουργίας 8 MHz, ο οποίος περιέχει 68.000 τρανζίστορ (από αυτό το γεγονός πήρε και το όνομά του).
- 1982: Εμφανίζεται ο Intel 80286, ο οποίος περιέχει 134.000 τρανζίστορ και έχει συχνότητα λειτουργίας 12,5 MHz. Αντίστοιχα η Motorola εμφανίζει τον 68010.
- 1985: Εμφανίζεται οι πρώτοι 32–bit μικροεπεξεργαστές. Από τη μια ο Intel 80386, ο οποίος περιέχει 275.000 τρανζίστορ και συχνότητα λειτουργίας 33 MHz και από την άλλη ο Motorola 68020 με 200.000 τρανζίστορ και 16 MHz. Οι εξελίξεις πλέον είναι ραγδαίες.
- 1989: Εμφανίζεται ο 32–bit μικροεπεξεργαστής Intel 80486, ο οποίος έχει 1.200.000 τρανζίστορ και συχνότητα λειτουργίας 50 MHz.

- 1993: Εμφανίζεται ο Intel Pentium, ο οποίος περιέχει 3.100.000 τρανζίστορ και η συχνότητα λειτουργίας του έχει φτάσει στα 166 MHz.
- 1993: Η Digital παρουσιάζει τον πρώτο 64–bit μικροεπεξεργαστή Alpha.
- 1997: Η Intel ανακοινώνει τον Pentium II. Η συχνότητα λειτουργίας του βρίσκεται στα 300 MHz και το ολοκληρωμένο κύκλωμα του αποτελείται από 7.500.000 τρανζίστορ.
- 1999: Η Intel ανακοινώνει τον Pentium III με συχνότητα λειτουργίας 450 MHz (σήμερα έχει φτάσει στο 1.13 GHz). Το ολοκληρωμένο κύκλωμα αποτελείται από 9.500.000 τρανζίστορ.
- 2000: Η Intel παρουσιάζει τον Pentium 4, αρχικά σε συχνότητα 1 GHz. Η τιμή του είναι υψηλή χωρίς να παρέχει σημαντική αύξηση της ταχύτητας επεξεργασίας, σε σχέση με τον Pentium III. Τελικά η ταχύτητα του επεξεργαστή ξεπερνά τα 4 GHz.
- Από εκεί και μετά η επεξεργαστές περνάνε στο επίπεδο των πολλαπλών πυρήνων.

MICROPROCESSOR	YEAR	SPEED	WORD LENGTH	TRANSISTORS	MIPS
Intel 4004	1971	108 KHz	4-bit	2,300	.06
Intel 8008	1972	200 KHz	8-bit	3,500	.06
Intel 8080	1974	2 MHz	8-bit	6,000	.64
Intel 8086	1978	4.47 MHz	16-bit	29,000	.66
Intel 8088	1981	4.47 MHz	16-bit	29,000	.75
Intel 80286	1982	12 MHz	16-bit	134,000	2.66
Intel 80386	1985	16-33 MHz	32-bit	275,000	4
Intel 80486 (i486)	1989	20-100 MHz	32-bit	1.2 Million	70
Intel 80586 (Pentium)	1993	75-200 MHz	32-bit	3.1 Million	126 - 203
Intel Pentium Pro	1995	150-200 MHz	32-bit	5.5 Million	300
Intel Pentium MMX	1997	166-233	32-bit	4.5 Million	-

		MHz			
Intel Pentium II	**1997**	**233-450 MHz**	**32-bit**	**7.5 Million**	**-**
Intel Pentium III	**1999**	**450-933 MHz**	**32-bit**	**Over 9.5 Million**	**-**
Intel Itanium Processor	**2000**	**1 GHz**	**64-bit**	**15,000,000**	**1,200**

Πίνακας 2: Η χρονολογική εξέλιξη των μικροεπεξεργαστών.

1.4 Από Τον Επεξεργαστή Στον Μικροελεγκτή.

Ο επεξεργαστής είναι το σημαντικότερο στοιχείο κάθε πληροφοριακού συστήματος. Όμως, για τη λειτουργία ενός πλήρους ενσωματωμένου υπολογιστικού συστήματος, απαιτούνται πολλά εξωτερικά υποσυστήματα και περιφερειακά. Τέτοια είναι:

- Κύκλωμα συνδετικής λογικής (glue logic) για τη σύνδεση των εξωτερικών μνημών και άλλων περιφερειακών παράλληλης σύνδεσης στην αρτηρία δεδομένων (bus) του επεξεργαστή.
- Μνήμη προγράμματος (τύπου ROM, FLASH, EPROM κ.λ.π.) η οποία περιέχει το λογισμικό του συστήματος. Σε κάποια μοντέλα, είναι δυνατό το κλείδωμα αυτής της μνήμης, μετά την εγγραφή της, ώστε να προστατευτεί το περιεχόμενό της από αντιγραφή.
- Μεγάλη ποσότητα μνήμης RAM
- Μόνιμη μνήμη αποθήκευσης παραμέτρων λειτουργίας (τύπου EEPROM ή NVRAM) η οποία να μπορεί να γράφεται στον πυρήνα του μικροελεγκτή. Αυτή η μνήμη έχει, έναντι της FLASH, το πλεονέκτημα της δυνατότητας διαγραφής και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης (reset).
- Διαχειριστή αιτήσεων διακοπής (interrupt request controller) από τα περιφερειακά.
- Κύκλωμα επιτήρησης λειτουργίας (watchdog timer) το οποίο αρχικοποιεί το σύστημα, αν αυτό εμφανίσει σημάδια δυσλειτουργίας λόγω κολλήματος (hang).
- Τοπικό ταλαντωτή για την παροχή παλμών χρονισμού (clock).
- Ρολόι πραγματικού χρόνου (Real Time Clock, RTC) το οποίο τροφοδοτείται από ανεξάρτητη μπαταρία και γι αυτό πρέπει να έχει πολύ χαμηλή κατανάλωση ρεύματος.
- Σειρά ανεξάρτητων ψηφιακών εισόδων και εξόδων (Parallel Input-Output, PIO). για την επικοινωνία με τον εξωτερικό κόσμο.

Τα παραπάνω στοιχεία συνδέονται με τρεις διαύλους, α) το δίαυλο διευθύνσεων, β) το δίαυλο δεδομένων και γ) το δίαυλο ελέγχου.

Στην προσπάθεια των κατασκευαστών πληροφοριακών συστημάτων ώστε να έχουμε όσο το δυνατών λιγότερα εξωτερικά στοιχεία γύρω από τον μικροεπεξεργαστή γεννήθηκε η ανάγκη κατασκευής του μικροελεγκτή.

Γενικά, όλες οι οικογένειες μικροελεγκτών ενσωματώνουν τα περισσότερα από τα παραπάνω περιφερειακά, με διαφοροποιήσεις κυρίως στην ύπαρξη ή μη εσωτερικής μνήμης προγράμματος και στο είδος της. Ο μικροελεγκτής λειτουργεί αυτόνομα εκτελώντας το πρόγραμμα που βρίσκεται στη μνήμη του μόλις τροφοδοτηθεί.

Έτσι, υπάρχουν:

- Μικροελεγκτές χωρίς μνήμη προγράμματος, οι οποίοι χαρακτηρίζονται ως *ROM-less*. Αυτοί παρέχουν πάντοτε μια παράλληλη αρτηρία (bus) δεδομένων, πάνω στην οποία συνδέονται εξωτερικές μνήμες προγράμματος και RAM. Τέτοιοι τύποι μικροελεγκτών προορίζονται για πιο ισχυρά υπολογιστικά συστήματα ελέγχου, με μεγαλύτερες απαιτήσεις μνήμης.
- Μικροελεγκτές με μνήμη ROM, η οποία κατασκευάζεται με το λογισμικό της (Mask ROM) ή γράφεται μόνο μια φορά (One Time Programmable, OTP). Παρέχουν τη δυνατότητα πολύ χαμηλού κόστους, όταν αγοράζονται σε πολύ μεγάλες ποσότητες.
- Μικροελεγκτές με μνήμη FLASH, οι οποίοι μπορούν συνήθως να προγραμματιστούν πολλές φορές. Αυτή είναι η πιο διαδεδομένη κατηγορία. Συχνά ο προγραμματισμός της μνήμης μπορεί να γίνει ακόμη και πάνω στο κύκλωμα της ίδιας της ενσωματωμένης (embedded) εφαρμογής (δυνατότητα In Circuit Programming, ISP). Αυτοί οι μικροελεγκτές έχουν ουσιαστικά αντικαταστήσει τους παλαιότερους τύπους EPROM που έσβηναν με υπεριώδη ακτινοβολία (από το ειδικό τζαμάκι).

Manufacturer	Device	RAM (kB)	Flash (kB)	Active (mA)	Sleep (μA)	Release
Atmel	AT90LS8535	0.5	8	5	15	1998
	Mega128	4	128	8	20	2001
	Mega165/325/645	4	64	2.5	2	2004
General Instruments	PIC	0.025	0.5	19	1	1975
Microchip	PIC Modern	4	128	2.2	1	2002
Intel	4004 4-bit	0.625	4	30	N/A	1971
	8051 8-bit Classic	0.5	32	30	5	1995
	8051 16-bit	1	16	45	10	1996
Philips	80C51 16-bit	2	60	15	3	2000
Motorola	HC05	0.5	32	6.6	90	1988
	HC08	2	32	8	100	1993
	HCS08	4	60	6.5	1	2003
Texas	TSS400 4-bit	0.03	1	15	12	1974
Instruments	MSP430F14x 16-bit	2	60	1.5	1	2000
	MSP430F16x 16-bit	10	48	2	1	2004
Atmel	AT91 ARM Thumb	256	1024	38	160	2004
Intel	XScale PXA27X	256	N/A	39	574	2004

Πίνακας 3: - Η ιστορία των μικροελεγκτών.

Αναλυτικά τα πλεονεκτήματα των μικροελεγκτών είναι:

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα άλλο ολοκληρωμένο κύκλωμα για να λειτουργήσουν.
- Η ενσωμάτωση περιφερειακών σημαίνει ευκολότερη υλοποίηση εφαρμογών λόγω των απλούστερων διασυνδέσεων. Επίσης, οδηγεί σε χαμηλότερη κατανάλωση ισχύος, μεγιστοποιώντας τη φορητότητα και ελαχιστοποιεί το κόστος της συσκευής στην οποία ενσωματώνεται ο μικροελεγκτής.
- Χαμηλό κόστος.
- Μεγαλύτερη αξιοπιστία, και πάλι λόγω των λιγότερων διασυνδέσεων.
- Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές. Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και τις χαμηλότερες ταχύτητες λειτουργίας.

- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσής τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

Η βασική αρχιτεκτονική των μικροελεγκτών δεν διαφέρει από αυτή των κοινών μικροεπεξεργαστών, αν και στους πρώτους απαντάται συχνά η αρχιτεκτονική μνήμης τύπου Harvard, η οποία χρησιμοποιεί διαφορετικές αρτηρίες σύνδεσης της μνήμης προγράμματος και της μνήμης δεδομένων (π.χ. οι σειρές AVR από την Atmel και PIC από την Microchip). Στους κοινούς μικροεπεξεργαστές συνηθίζεται η ενιαία διάταξη μνήμης τύπου von-Neumann.

1.5 Τα Μικροϋπολογιστικά Συστήματα Σήμερα

Αν συγκρίνει κανείς τις δυνατότητες των μικροεπεξεργαστών ή μικροελεγκτών που χρησιμοποιούνται σε μικροϋπολογιστικά συστήματα με τις αντίστοιχες των επεξεργαστών των μεγάλων συστημάτων (mainframes) ή ακόμα και των προσωπικών υπολογιστών (PC), η διαφορά είναι μεγάλη σε πολλά επίπεδα. Οι ταχύτητες λειτουργίας είναι χαμηλότερες, τα μεγέθη μνήμης που μπορούν να διαχειριστούν πολύ περιορισμένα κλπ. Όμως μια τέτοια σύγκριση είναι άδικη δεδομένου ότι δεν προορίζονται για τις ίδιες εφαρμογές. Παρότι πριν από 15-20 χρόνια επεξεργαστές 8-bit χρησιμοποιούνταν ευρέως σε προσωπικούς υπολογιστές, σήμερα η χρήση τους είναι τελείως διαφορετική. Υπάρχει μια σειρά από εφαρμογές ειδικού σκοπού που δεν έχουν την ανάγκη των επιδόσεων των σύγχρονων επεξεργαστών 32-bit. Σαν τέτοιες εφαρμογές μπορεί να αναφέρει κανείς τους ενσωματωμένους μικροϋπολογιστές σε οικιακές συσκευές, συστήματα ασφαλείας, συστήματα ελέγχου αυτοκινήτων, ρομποτικά συστήματα, παραγωγικές μονάδες όπως θερμοκήπια και βιοτεχνίες, όργανα μετρήσεων όπως ιατρικά και ηλεκτρονικά όργανα, μετεωρολογικοί σταθμοί, παιχνιδομηχανές κλπ. Σε τέτοιες εφαρμογές η χρήση ισχυρότερων επεξεργαστών από αυτούς που πραγματικά χρειάζονται προσθέτει όχι μόνο κόστος, που μπορεί να είναι πολύ κρίσιμο, αλλά και ανεπιθύμητη αύξηση πολυπλοκότητας, κατανάλωσης, διαστάσεων κλπ. Για το λόγο αυτό διατίθεται σήμερα μια πλειάδα μικροελεγκτών που ενσωματώνουν στο ίδιο ολοκληρωμένο κύκλωμα την κεντρική μονάδα επεξεργασίας (ΚΜΕ, ή CPU) μαζί με έναν αριθμό περιφερειακών (μνήμη, χρονιστές / μετρητές, ακροδέκτες γενικής χρήσεως , DAC και ADC, συριακές και παράλληλες Θύρες επικοινωνίας κ.α). Διαλέγοντας το κατάλληλο μικροελεγκτή για μία εφαρμογή μπορεί να ελαχιστοποιηθεί το πλήθος των απαιτούμενων εξωτερικών εξαρτημάτων. Σε εφαρμογές που οι ανάγκες μνήμης ξεφεύγουν πολύ από τα μεγέθη που διαθέτουν οι ενσωματωμένες μνήμες σε μικροελεγκτές, είναι δυνατή η χρήση κάποιου μικροεπεξεργαστή στους διαύλους του οποίου μπορούν να συνδεθούν μνήμη κατάλληλου μεγέθους και τα απαραίτητα περιφερειακά. Αν ξεκινήσουμε από τα πιο στοιχειώδη τμήματα ενός υπολογιστή, μπορούμε να πούμε ότι ανεξάρτητα από τα μεγέθη όλα τα μικροϋπολογιστικά συστήματα καθώς και οι μεγαλύτεροι υπολογιστές αποτελούνται από τα ίδια βασικά τμήματα: ΚΜΕ, μονάδες I/O, μνήμη, και σύστημα χρονισμού (ρολόι). Η ΚΜΕ διαχειρίζεται πληροφορία σύμφωνα με τις οδηγίες ενός προγράμματος εντολών. Διαχειρίζεται επίσης ένα πλήθος γραμμών ελέγχου που της δίνουν τη δυνατότητα να ελέγξει τα περιφερειακά και να επικοινωνήσει με τον έξω κόσμο. Μερικά περιφερειακά εισόδου χρειάζεται να μετατρέψουν αναλογικά σήματα σε δυαδικά ψηφία που σε επίπεδο κυκλώματος αντιστοιχούν σε 0 και 5V τάση (π.χ. αισθητήρας Θερμοκρασίας). Άλλες μονάδες εισόδου που στηρίζονται στη χρήση διακοπτών μπορούν να παράσχουν κατευθείαν τις δύο αυτές καταστάσεις όπως ένα πληκτρολόγιο. Τις καταστάσεις αυτές τις δέχεται η ΚΜΕ σαν είσοδο. Στις συσκευές εξόδου η ΚΜΕ αποστέλλει ψηφιακά δεδομένα τα οποία οι συσκευές αυτές μπορούν να μετατρέψουν σε άλλης μορφής σήματα, για να δώσουν στον έξω κόσμο τα αποτελέσματα της επεξεργασίας των δεδομένων εισόδου. Τέτοιες συσκευές μπορεί να είναι οθόνες, beeper, ρελέ κλπ. Ένα υπολογιστικό σύστημα διαθέτει συνήθως περισσότερες από μία μονάδες εισόδου - εξόδου.

Υπάρχει συχνά σημαντική διαφορά μεταξύ των μονάδων εισόδου και εξόδου ενός μικροϋπολογιστικού συστήματος και των άλλων υπολογιστών όπως π.χ ενός προσωπικού υπολογιστή. Σε έναν προσωπικό υπολογιστή η βασική μονάδα εισόδου είναι το πληκτρολόγιο και το ποντίκι, ενώ συμπληρωματική είσοδος μπορεί να δοθεί από συσκευές όπως ο σαρωτής , το μικρόφωνο κλπ. Επίσης σε ένα PC η κύρια έξοδος είναι η οθόνη και ο εκτυπωτής. Δεδομένου

όμως ότι ένα μικροϋπολογιστικό σύστημα προορίζεται σήμερα κυρίως για εφαρμογές ελέγχου, οι είσοδοι και έξοδοί του είναι σήματα από αισθητήρες (sensors) και επενεργητές (actuators) ή διακόπτες. Π.χ., σε ένα σύστημα συναγερμού ο ενσωματωμένος μικροεπεξεργαστής δέχεται είσοδο από αισθητήρες υπέρυθρων ακτίνων, θορύβου, καπνού, υγρασίας κλπ, για να ανιχνεύσει αν εισήλθε κάποιος ή αν έχει εκδηλωθεί πυρκαγιά / πλημμύρα στον προστατευόμενο χώρο. Η κύρια έξοδος ενός τέτοιου συστήματος είναι η σειρήνα, ο φάρος και η κλήση τηλεφώνου (dialer). Στα περισσότερα συστήματα συναγερμού υπάρχει ένα στοιχειώδες πληκτρολόγιο (keypad) και μια LCD οθόνη, των οποίων όμως η χρήση είναι περισσότερο βοηθητική. Όλα τα chip μικροελεγκτών διαθέτουν δυνατότητα σύνδεσης με ένα τερματικό για να μπορούμε να το προγραμματίσουμε. Για τη σύνδεση αυτή απαιτούνται ορισμένα ηλεκτρονικά εξαρτήματα. Η συνήθης διαδικασία προγραμματισμού είναι:

- Επιλογή του κατάλληλου μικροελεγκτή ή μικροελεγκτική μονάδα ,(μικροελεγκτής με μνήμη RAM και ελάχιστο hardware).
- Σύνδεση με έναν ηλεκτρονικό υπολογιστή.
- Τον προγραμματίζουμε να εκτελεί ένα συγκεκριμένο πρόγραμμα και
- Να τον τοποθετήσουμε στη συγκεκριμένη εφαρμογή, για να εκτελέσει τη λειτουργία που του έχουμε ορίσει, όπως έλεγχος στροφών κινητήρα, σύστημα πυρασφάλειας, χρονοδιακόπτης και πολλές άλλες εφαρμογές.

2 ΚΕΦΑΛΑΙΟ 2

Ο Μικροελεγκτής AVR

2.1 Εισαγωγή

Ένα χαρακτηριστικό παράδειγμα σύγχρονων μικροελεγκτών είναι εκείνοι της οικογένειας AVR της εταιρείας ATMEL. Οι μικροελεγκτές αυτοί προσφέρονται με ένα πλήθος εναλλακτικού αριθμού ακροδεκτών, ξεκινώντας από μικρά και φτηνά ολοκληρωμένα των 8 ακροδεκτών για εφαρμογές πολύ χαμηλού κόστους με περιορισμένες απαιτήσεις σε πλήθος προγραμματιζόμενων ακροδεκτών γενικού σκοπού. Οι πιο εξελιγμένοι μικροελεγκτές της οικογένειας διαθέτουν περισσότερους από 60 προγραμματιζόμενους ακροδέκτες γενικού σκοπού. Επίσης πολλά μέλη της σειράς διατίθενται σε τρεις παραλλαγές: τους απλούς μικροελεγκτές που λειτουργούν στα 5V, τους χαμηλής κατανάλωσης στα 2.7V (κατάληξη L) και τους πολύ χαμηλούς σε κατανάλωση στα 1.8V (κατάληξη V).

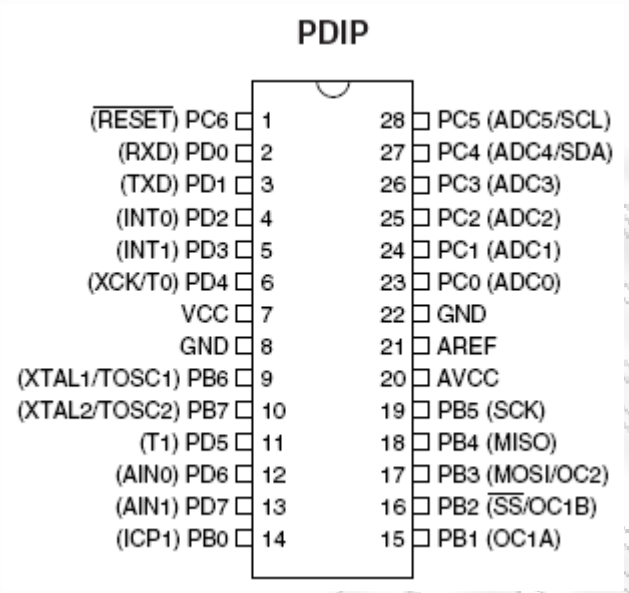
Συνήθως οι ακροδέκτες γενικού σκοπού έχουν πολυπλεγμένες περισσότερες από μία λειτουργίες, όπως για παράδειγμα είσοδοι με ικανότητα να προκαλούν διακοπή (interrupt) στον εσωτερικό επεξεργαστή, είσοδοι αναλογικών συγκριτών ή μετατροπέων αναλογικού σε ψηφιακό (ADC), είσοδοι κεντρικού ρολογιού (oscillator) ή ασύγχρονης οδήγησης μετρητών (counters), ακροδέκτες για σύνδεση με διάφορες διεπαφές όπως USART. SPI κ.α. Στα πιο εξελιγμένα μέλη της οικογένειας διατίθενται ενσωματωμένα περιφερειακά ακόμα και για την οδήγηση LCD οθόνης ή τη σύνδεση με USB interface.

Στο εσωτερικό ενός μικροελεγκτή όπως ο AVR υπάρχει ένας αριθμός από διαφορετικούς τύπους μνήμης, όπως Flash για την εγγραφή του λογισμικού συστήματος (firmware), EEPROM για την αποθήκευση διαφόρων παραμέτρων, καθώς και κάποιος αριθμός θέσεων μνήμης Ram για τις μεταβλητές τον λογισμικού. Για το λόγο αυτό οι AVR δεν βγάζουν σε ακροδέκτες την εσωτερική αρτηρία διευθύνσεων ή δεδομένων παρά μόνο ακροδέκτες γενικού σκοπού.

Με όλα τα παραπάνω περιφερειακά είναι φανερό ότι το πλήθος των εξωτερικών στοιχείων που απαιτούνται για τη δημιουργία ενός συστήματος με μικροελεγκτή AVR είναι ελάχιστο.Το βασικό μειονέκτημα μιας τέτοιας αρχιτεκτονικής μικροελεγκτή είναι η δυσκολία σε επεκτασιμότητα. Π.χ. αν οι απαιτήσεις σε μνήμη RAM είναι μεγάλες, ο μικροελεγκτής δεν είναι εύκολο να συνδεθεί με εξωτερική μνήμη, μια και δεν έχει αρτηρία διευθύνσεων και δεδομένων. Για να γίνει αυτό θα πρέπει να υλοποιηθούν τέτοιες αρτηρίες με τη χρήση ακροδεκτών γενικού σκοπού οι οποίες ωστόσο θα ήταν αδύνατο να επιτύχουν γρήγορους χρόνους προσπέλασης της μνήμης. Επίσης η συχνότητα ρολογιού στην οποία λειτουργούν τέτοιοι μικροελεγκτές δεν ξεπερνά τα 20 MHz στα πιο εξελιγμένα μοντέλα μιας σειράς όπως οι AVR mega.

2.2 Αρχιτεκτονική Του ATmega8

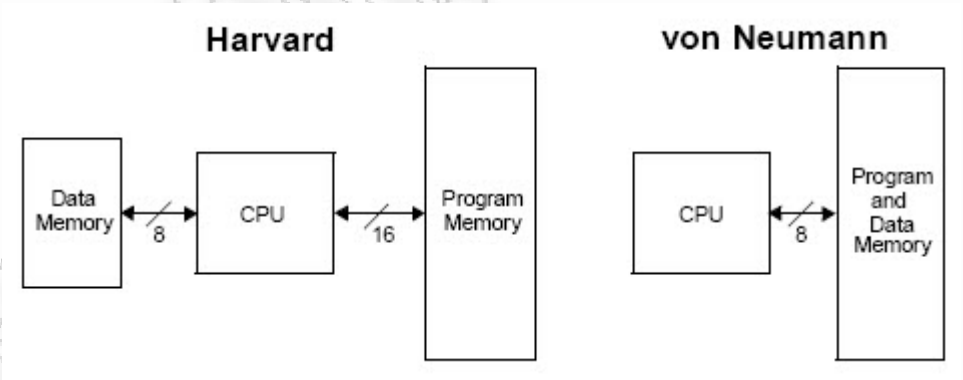
Ο μικροελεγκτής ATmega8 ανήκει στην οικογένεια AVR των μικροελεγκτών της ATMEL. Οι μικροελεγκτές AVR χρησιμοποιούν τροποποιημένη Αρχιτεκτονική Harvard 8 bit RISC και αναπτύχθηκαν από την ATMEL για πρώτη φορά το 1996. Η AVR ήταν μια από τις οικογένειες μικροελεγκτών που έκαναν χρήση της on-chip μνήμης flash για την αποθήκευση του προγράμματος, σε αντίθεση με τα Programmable ROM, EPROM ή EEPROM που χρησιμοποιούνται από άλλους μικροελεγκτές.



Εικόνα 2: Οι ακροδέκτες του ATmega8

Η βασική αρχιτεκτονική των AVR επινοήθηκε από δύο μαθητές στο Νορβηγικό Ινστιτούτο Τεχνολογίας τους Alf-Bogen EGIL και Vegard Wollan. Αργότερα η πατέντα για τους AVR μικροελεγκτές αγοράστηκε από την εταιρία ATMEL και η εσωτερική αρχιτεκτονική τους αναπτύχθηκε περαιτέρω. Η θυγατρική της ATMEL στην Νορβηγία ιδρύθηκε από τους δύο φοιτητές. Το όνομα AVR δεν αποτελεί κάτι ιδιαίτερο όσον αφορά την ερμηνεία του. Απλά ονομάστηκε έτσι και ορίζει όλοι την οικογένεια των μικροελεγκτών τύπου 8-bit RISC.

Η αρχιτεκτονική Harvard έχει τη μνήμη προγράμματος και τη μνήμη δεδομένων ως χωριστές μνήμες που προσεγγίζεται από χωριστούς διαύλους (bus). Αυτό βελτιώνει το εύρος ζώνης πέρα από την παραδοσιακή αρχιτεκτονική Von Neumann στην οποία το πρόγραμμα και τα δεδομένα προσκομίζονται από την ίδια μνήμη χρησιμοποιώντας τον ίδιο δίαυλο. Για να εκτελέσει μια εντολή, μια μηχανή Von Neumann πρέπει να κάνει γενικά περισσότερες προσβάσεις στον δίαυλο για να προσκομίσει την πληροφορία. Κατόπιν τα δεδομένα μπορεί να πρέπει να μεταφερθούν μέσω του διαύλου, να χρησιμοποιηθούν στην αριθμητική και λογική μονάδα, και ενδεχομένως να τοποθετηθούν σε μια νέα θέση μνήμης.

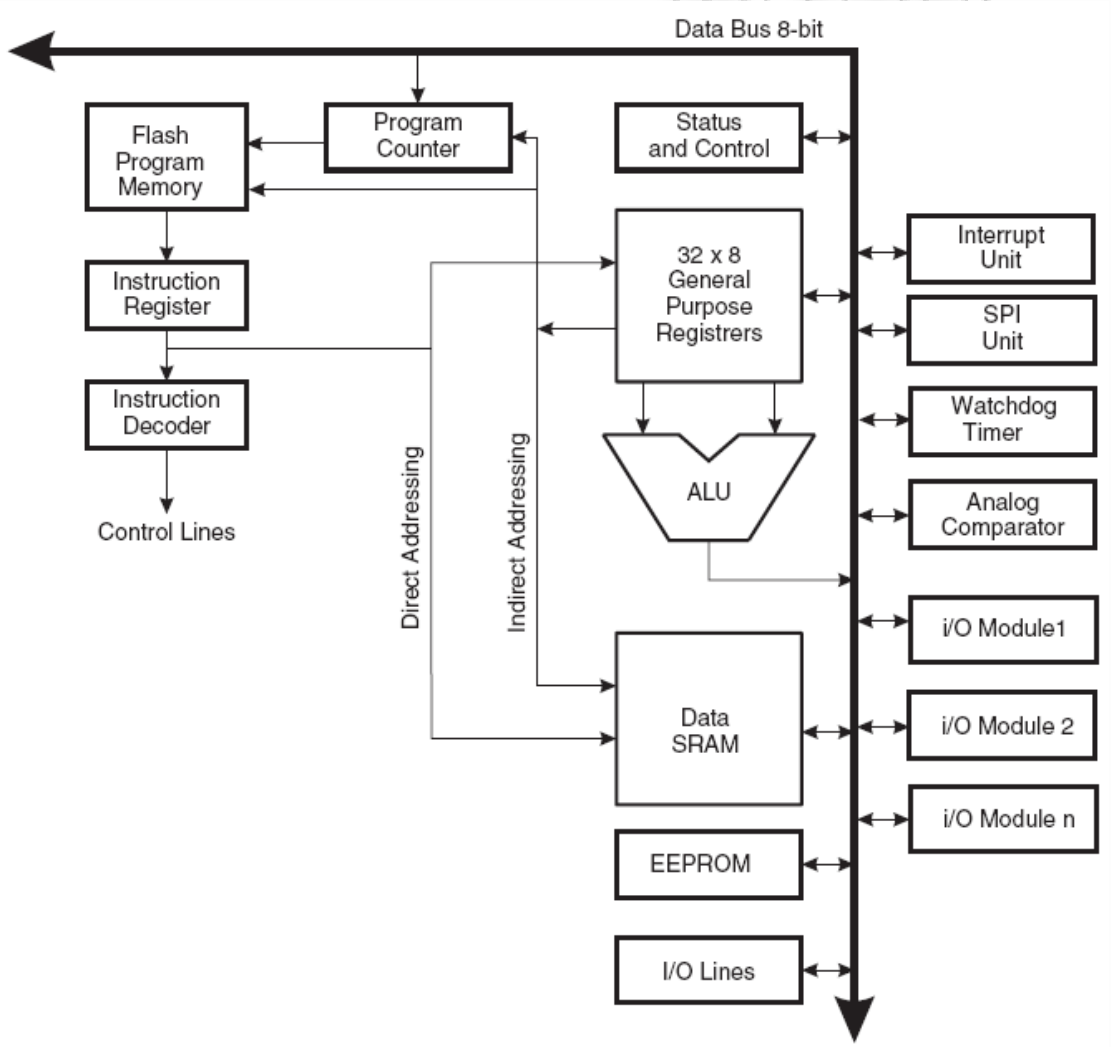


Εικόνα 3: Σύγκριση των αρχιτεκτονικών Harvard von Neumann

Όπως μπορεί κανείς να δει από αυτήν την περιγραφή, η αρχιτεκτονική αυτή μπορεί να δημιουργήσει κυκλοφοριακή συμφόρηση ή ακόμη και κορεσμό. Με την αρχιτεκτονική Harvard, η πληροφορία προσκομίζεται σε έναν απλό κύκλο ρολογιού. Ενώ η μνήμη προγράμματος προσπελαύνεται, η μνήμη δεδομένων είναι σε ένα ανεξάρτητο δίαυλο και μπορεί να διαβαστεί και να γραφτεί. Αυτοί οι χωρισμένοι δίαυλοι επιτρέπουν σε μια εντολή την εκτέλεση, ενώ η επόμενη εντολή προσκομίζεται. Μια σύγκριση των αρχιτεκτονικών Harvard και von Neumann παρουσιάζεται στην Εικόνα 3.

Τα γενικά γνωρίσματα των μικροελεγκτών αυτής της οικογένειας σε ότι αφορά την μνήμη δεδομένων και προγράμματος συνοψίζονται ως εξής:

- **Μνήμη δεδομένων.** Η Flash, η EEPROM και η SRAM είναι όλες ενσωματωμένες σε ένα και μοναδικό chip, καταργώντας την ανάγκη για την εξωτερική μνήμη. Ορισμένες συσκευές έχουν έναν εξωτερικό δίαυλο ο οποίος επιτρέπει την προσθήκη συμπληρωματικών chip μνήμης .
- **Μνήμη Προγράμματος** (Flash). Οι εντολές του προγράμματος αποθηκεύονται σε μη πτητική μνήμη Flash. Αν και είναι μικροελεγκτές 8-bit, κάθε εντολή έχει εύρος ένα ή δύο 16-bit words. Το μέγεθος της μνήμης του προγράμματος αναφέρεται συνήθως στην ονοματοδοσία της ίδια της συσκευής (για παράδειγμα ο ATmega64x έχει Flash 64 KB).
- **Εσωτερική Μνήμη Δεδομένων**. Η διευθυνσιοδότηση των δεδομένων αποτελείται από το αρχείο καταχωρητών, τους καταχωρητές I/O, και την SRAM.



Εικόνα 4:Αρχιτεκτονική του μικροελεγκτή ATmega8.

2.3 Τεχνικά Χαρακτηριστικά Του ATmega8

- AVR 8 bit μικροελεγκτής υψηλής απόδοσης και χαμηλής κατανάλωσης ισχύος.
- Προηγμένη αρχιτεκτονική RISC.
  - 120 ισχυρές εντολές που οι περισσότερες εκτελούνται σε ένα κύκλο ρολογιού.
  - 32 X 8 γενικής χρήσης καταχωρητές.
  - Πλήρης στατική λειτουργία
  - Απόδοση μέχρι 16 MIPS στα 16 MHz.
  - Ενσωματωμένο πολλαπλασιαστή δύο κύκλων ρολογιού.
- Μη πτητική μνήμη προγράμματος και δεδομένων.
  - 8K Bytes μνήμη flash, προγραμματιζόμενης και πάνω στην πλακέτα, 10.000 κύκλων εγγραφών/διαγραφών.
  - 512 Bytes EEPROM διάρκειας 100.000 κύκλων εγγραφών/διαγραφών.
  - 1K Byte εσωτερική SRAM
  - Προγραμματιζόμενο κλείδωμα για την ασφάλεια του προγράμματος
- Χαρακτηριστικά περιφερειακών
  - Δύο 8 bit χρονιστές/Απαριθμητές με ανεξάρτητο προδιαιρέτη, μια μέθοδο σύγκρισης.
  - Έναν 16 bit χρονιστή/Απαριθμητή με ανεξάρτητο προδιαιρέτη, μέθοδο σύγκρισης.
  - Απαριθμητή πραγματικού χρόνου με ξεχωριστό ταλαντωτή.
  - Τρία κανάλια PWM.
  - 6 κανάλια ADC στην συσκευασία PDIP με διακριτότητα 10 bit.
  - 8 κανάλια ADC στην συσκευασία TQFP και QFN/MLF με διακριτότητα 10 bit.
  - Σειριακή διασύνδεση δύο γραμμών.
  - Σειριακό προγραμματιζόμενο USART.
  - Master/Slave SPI σειριακή διασύνδεση.
  - Προγραμματιζόμενο Watchdog Timer με ξεχωριστό on-chip ταλαντωτή.
  - On-chip αναλογικό συγκριτή.
- Ειδικά χαρακτηριστικά του μικροελεγκτή.
  - Εσωτερικό διαβαθμιζόμενο RC ταλαντωτή.
  - Power-on Reset and Programmable Brown-out Detection
  - Εξωτερικές και εσωτερικές πηγές διακοπών.
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby.
- I/O και συσκευασία.
  - 23 προγραμματιζόμενες ακίδες εισόδου/εξόδου.
  - 28-lead PDIP, 32-lead TQFP, and 32-pad QFN/MLF
- Τροφοδοσία.
  - 2.7 - 5.5V (ATmega8L)
  - 4.5 - 5.5V (ATmega8)
- Συχνότητες λειτουργίας.
  - 0 - 8 MHz (ATmega8L)
  - 0 - 16 MHz (ATmega8)
- Κατανάλωση ισχύος στα 4 MHz, 3V, 25 ^{o}C.
  - Active: 3.6 mA
  - Idle Mode: 1.0 mA

- Power-down Mode: 0.5 μA

2.4 Η Σημασία Των Ακίδων Του ATmega8.

VCC: Ακίδα τροφοδοσίας
GND: Ακίδα αναφοράς (γη)

PORTB (PB7….PB0):
XTAL1/XTAL2/TOSC1/TOSC2

Η θύρα αυτή περιλαμβάνει 8 αμφίδρομες ακίδες. Οι ακίδες αυτές μπορούν κατά επιλογή να συνδεθούν εσωτερικά στην τροφοδοσία μέσω αντιστάσεων πρόσδεσης (pull-up resistor) όταν λειτουργούν σαν είσοδοι. Οι buffers εξόδου της θήρας B έχουν συμμετρικά χαρακτηριστικά οδήγησης με διπλή ικανότητα και να απορροφά και να πηγάζει. Σαν είσοδοι οι ακροδέκτες της θήρας B οι οποίες είναι εξωτερικά σε χαμηλό δυναμικό θα πηγάσουν ρεύμα εάν είναι ενεργοποιημένες οι pull-up αντιστάσεις. Οι ακροδέκτες της θήρας B βρίσκονται σε μια Τρίτη κατάσταση όταν οι συνθήκες του reset είναι ενεργές ακόμη και όταν το ρολόι χρονισμού δεν τρέχει.

Εξαρτάτε από την επιλογή του ρολογιού με τα fuse settings ο ακροδέκτης PB6 μπορεί να χρησιμοποιηθεί σαν είσοδος στον ανάστροφο ενισχυτή ταλαντωτή και είσοδος στο εσωτερικό κύκλωμα ρολογιού.

Εξαρτάτε από την επιλογή του ρολογιού με τα fuse settings ο ακροδέκτης PB7 μπορεί να χρησιμοποιηθεί σαν έξοδος από τον ανάστροφο ενισχυτή ταλαντωτή.

Αν ο εσωτερικός βαθμονομημένος RC ταλαντωτής χρησιμοποιείται σαν πηγή ρολογιού του chip PB7..6 χρησιμοποιείται σαν TOSC2..1 είσοδος για τον ασύγχρονο Timer/Counter2 εάν το AS2 bit στον ASSR είναι set. Τέλος πολλές από τις ακίδες της θύρας B έχουν και άλλες λειτουργίες που αναφέρονται στο data sheet του μικροελεγκτή.

PORTC (PC6….PC0):

Η θύρα αυτή περιλαμβάνει 7 αμφίδρομες ακίδες. Οι ακίδες αυτές μπορούν κατά επιλογή να συνδεθούν εσωτερικά στην τροφοδοσία μέσω αντιστάσεων πρόσδεσης (pull-up resistor) όταν λειτουργούν σαν είσοδοι. Οι buffers εξόδου της θήρας C έχουν συμμετρικά χαρακτηριστικά οδήγησης με διπλή ικανότητα και να απορροφά και να πηγάζει. Σαν είσοδοι οι ακροδέκτες της θήρας C οι οποίες είναι εξωτερικά σε χαμηλό δυναμικό θα πηγάσουν ρεύμα εάν είναι ενεργοποιημένες οι pull-up αντιστάσεις. Οι ακροδέκτες της θήρας C βρίσκονται σε μια Τρίτη κατάσταση όταν οι συνθήκες του reset είναι ενεργές ακόμη και όταν το ρολόι χρονισμού δεν τρέχει. πολλές από τις ακίδες της θύρας C έχουν και άλλες λειτουργίες που αναφέρονται στο data sheet του μικροελεγκτή.

PORTD (PD7….PD0):

Η θύρα αυτή περιλαμβάνει 8 αμφίδρομες ακίδες. Οι ακίδες αυτές μπορούν κατά επιλογή να συνδεθούν εσωτερικά στην τροφοδοσία μέσω αντιστάσεων πρόσδεσης (pull-up resistor) όταν λειτουργούν σαν είσοδοι. Οι buffers εξόδου της θήρας D έχουν συμμετρικά χαρακτηριστικά οδήγησης με διπλή ικανότητα και να απορροφά και να πηγάζει. Σαν είσοδοι οι ακροδέκτες της θήρας D οι οποίες είναι εξωτερικά σε χαμηλό δυναμικό θα πηγάσουν ρεύμα εάν είναι ενεργοποιημένες οι pull-up αντιστάσεις. Οι ακροδέκτες της θήρας D βρίσκονται σε μια Τρίτη κατάσταση όταν οι συνθήκες του reset είναι ενεργές ακόμη και όταν το ρολόι χρονισμού δεν τρέχει.

PC6/RESET:

Εάν το RSTDISBL Fuse είναι προγραμματισμένο το pin PC6 χρησιμοποιείται σαν ακροδέκτης εισόδου/εξόδου. Σημειωτέων είναι ότι τα ηλεκτρικά χαρακτηριστικά του PC6 διαφέρουν από εκείνα των άλλων ακροδεκτών της θήρας C.

Εάν το RSTDISBL Fuse δεν είναι προγραμματισμένο PC6 χρησιμοποιείται σαν είσοδος Reset. Χαμηλό δυναμικό στον ακροδέκτη αυτό για χρόνο μεγαλύτερο από 1,5 μs θα παράξει ένα σήμα αρχικοποίησης του μικροελεγκτή ακόμη και όταν το ρολόι χρονισμού δεν τρέχει. Παλμοί μικρότερης διάρκειας δεν είναι εγγυημένο ότι θα παράξουν σήμα αρχικοποίησης.

AVCC:

AVcc είναι ο ακροδέκτης τροφοδοσίας για τον μετατροπέα από αναλογικό σε ψηφιακό της θήρας C (3...0), και ADC (7...6). πρέπει να συνδεθεί εξωτερικά στην τροφοδοσία του μικροελεγκτή ακόμη και αν οι μετατροπείς δεν χρησιμοποιούνται. Αν οι μετατροπείς χρησιμοποιούνται πρέπει να συνδεθεί στην τροφοδοσία μέσω ενός φίλτρου χαμηλής διέλευσης. Σημειώνετε ότι η θήρας C (5..4) χρησιμοποιεί την τροφοδοσία για τα ψηφιακά σήματα.

AREF:

AREF είναι ο ακροδέκτης αναφοράς για τους μετατροπείς από αναλογικό σε ψηφιακό.

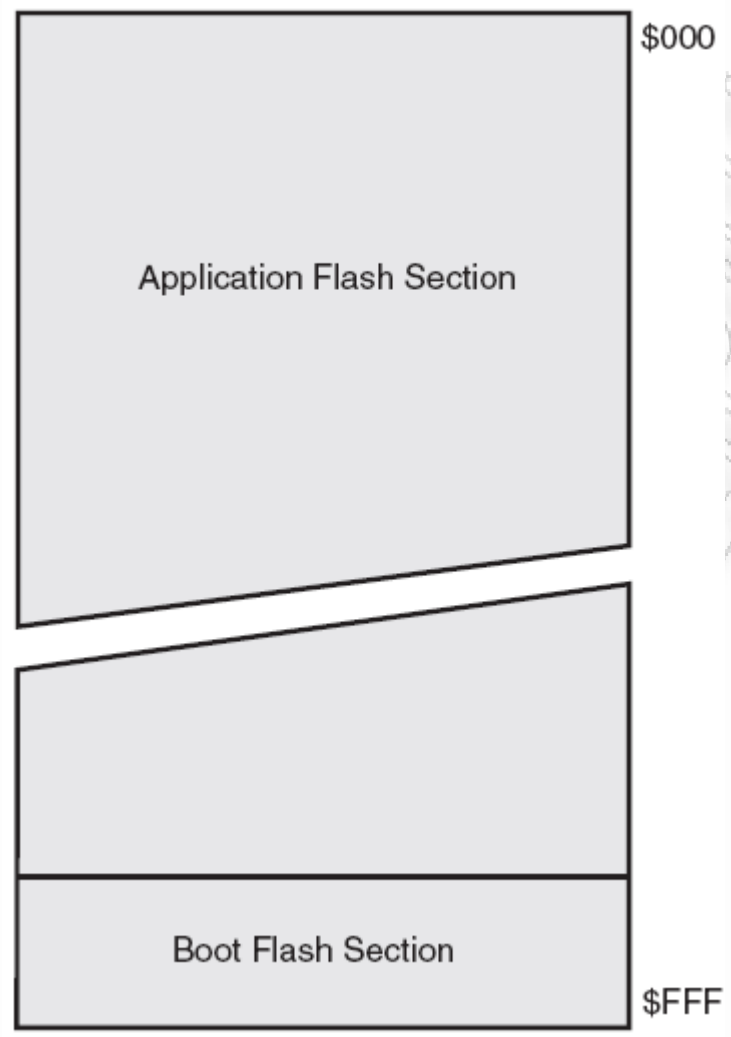
2.5 Οι Μνήμες Του ATmega8

Η παράγραφος αυτή περιγράφει τις διαφορετικές μνήμες του ATmega8. η αρχιτεκτονική AVR έχει δύο κύρια τμήματα μνήμης, το τμήμα μνήμης για τα δεδομένα και το τμήμα μνήμης για το πρόγραμμα. Επιπλέον ο ATmega8 έχει και ένα τμήμα μνήμης την EEPROM για αποθήκευση δεδομένων.

2.5.1 Η Μνήμη Προγράμματος Τεχνολογίας Flash

Ο ATmega8 έχει 8K bytes εσωτερική επαναπρογραμματιζόμενη μνήμη για την αποθήκευση του προγράμματος. Από την στιγμή που όλες οι εντολές έχουν μήκος 16 ή 32 bits , η συγκεκριμένη μνήμη είναι οργανωμένη ως 4K x 16. Για την ασφάλεια του προγράμματος η μνήμη είναι χωρισμένη σε δύο μέρη, το κομμάτι του προγράμματος της εφαρμογής και το κομμάτι του προγράμματος εκκίνησης. Η διάρκεια ζωής της είναι το λιγότερο 10000 κύκλοι εγγραφής/ διαγραφής.

Ο program counter (PC) του συγκεκριμένου μικροελεγκτή έχει μέγεθος 12 bit και κατά συνέπεια διευθυνσιοδοτεί τα 4K περιοχής μνήμης προγράμματος. Ή λειτουργία του τμήματος εκκίνησης περιγράφεται λεπτομερώς στο εγχειρίδιο του μικροελεγκτή. Πίνακες με σταθερές μπορούν να τοποθετηθούν εντός του τμήματος διευθύνσεων της μνήμης προγράμματος (λεπτομέρειες υπάρχουν στην εντολή LPM).



Εικόνα 5: Χαρτογράφηση της μνήμης flash

2.5.2 **Μνήμη Δεδομένων SRAM**

Η Εικόνα 6 δείχνει πως είναι οργανωμένη ή μνήμη SRAM του μικροελεγκτή.

Οι χαμηλότερες 1120 θέσεις μνήμης απευθύνονται στους καταχωρητές γενικής χρήσης, στην μνήμη I/O και στην εσωτερική μνήμη δεδομένων SRAM. Οι πρώτες 96 θέσεις διευθυνσιοδοτούν τους καταχωρητές και την μνήμη I/O οι επόμενες 1024 θέσεις διευθυνσιοδοτούν την εσωτερική μνήμη δεδομένων SRAM.

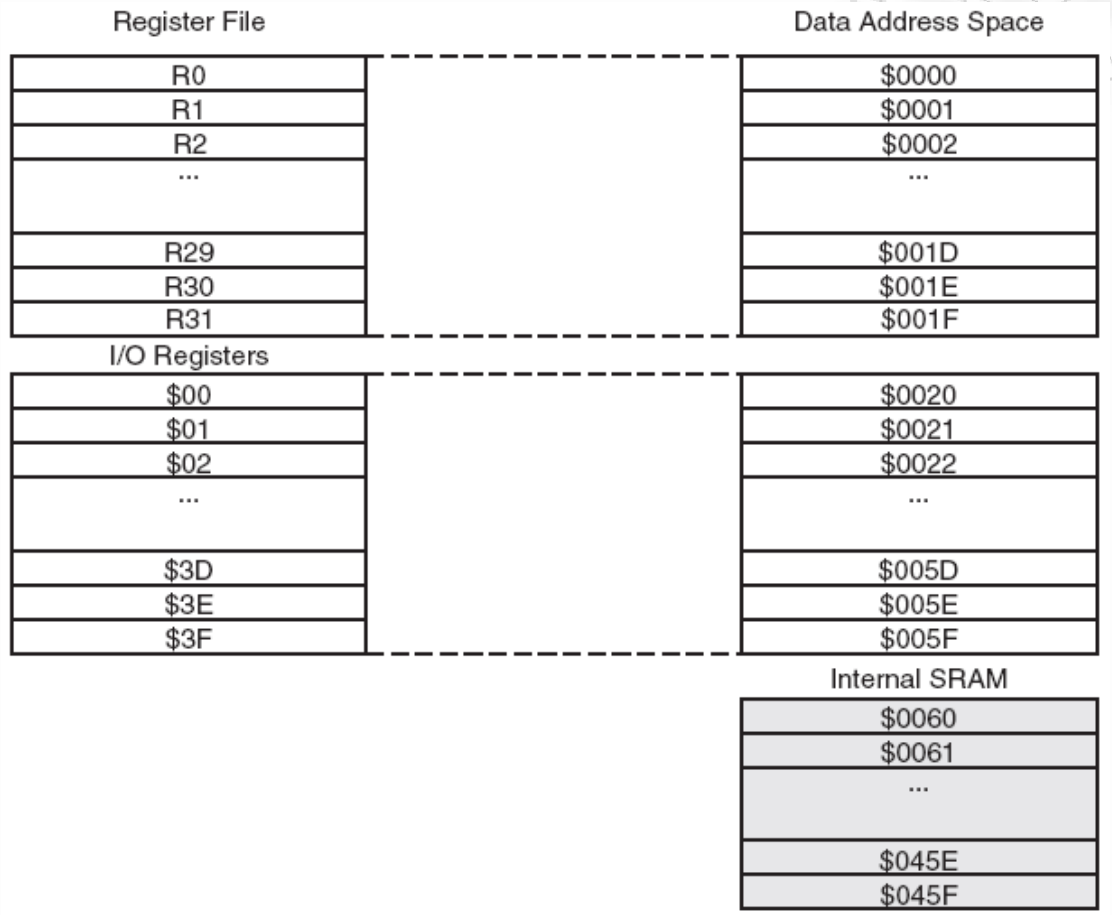
Η CPU μπορεί να δει αυτόν τον χώρο μέσω διαφορετικών τρόπων διευθυνσιοδότησης. Οι καταχωρητές R26-R31 συμπεριφέρονται σαν καταχωρητές δείκτη (X,Y,Z) μέσω των οποίων γίνονται οι έμμεσες προσπελάσεις. Η άμεση διευθυνσιοδότηση φτάνει ολόκληρο το τμήμα διευθύνσεων της μνήμης SRAM.

Με την βοήθεια του έμμεσου τρόπου με επιπλέον όρισμα καθίσταται δυνατή η προσπέλαση 63 διαφορετικών θέσεων μνήμης, έχοντας μόνο μια σταθερή βασική διεύθυνση αποθηκευμένη στον καταχωρητή X,Y ή Z. Οι ίδιοι καταχωρητές αυξομειώνονται ανάλογα όταν χρησιμοποιούνται οι έμμεσοι τρόποι με μείωση ή αύξηση διεύθυνσης.

Στο τμήμα I/0 που βρίσκεται στην μνήμη δεδομένων SRAM μπορεί να γίνει πρόσβαση κατά δύο τρόπους:

- Ως μνήμη SRAM. Περιοχή διευθύνσεων στην περίπτωση αυτή είναι $20-$5F.

- Ως I/0 καταχωρητές με τις διευθύνσεις να ξεκινούν από την διεύθυνση $00 και να φθάνουν ως και την διεύθυνση $3F. Οι 64 καταχωρητές I/O που προαναφέραμε φαίνονται αναλυτικά παρακάτω στην Εικόνα 9. Η διεύθυνση που είναι έξω από την παρένθεση είναι η διεύθυνση που έχει ο καταχωρητής. Η διεύθυνση που βρίσκεται μέσα στην παρένθεση είναι η διεύθυνση στην SRAM.



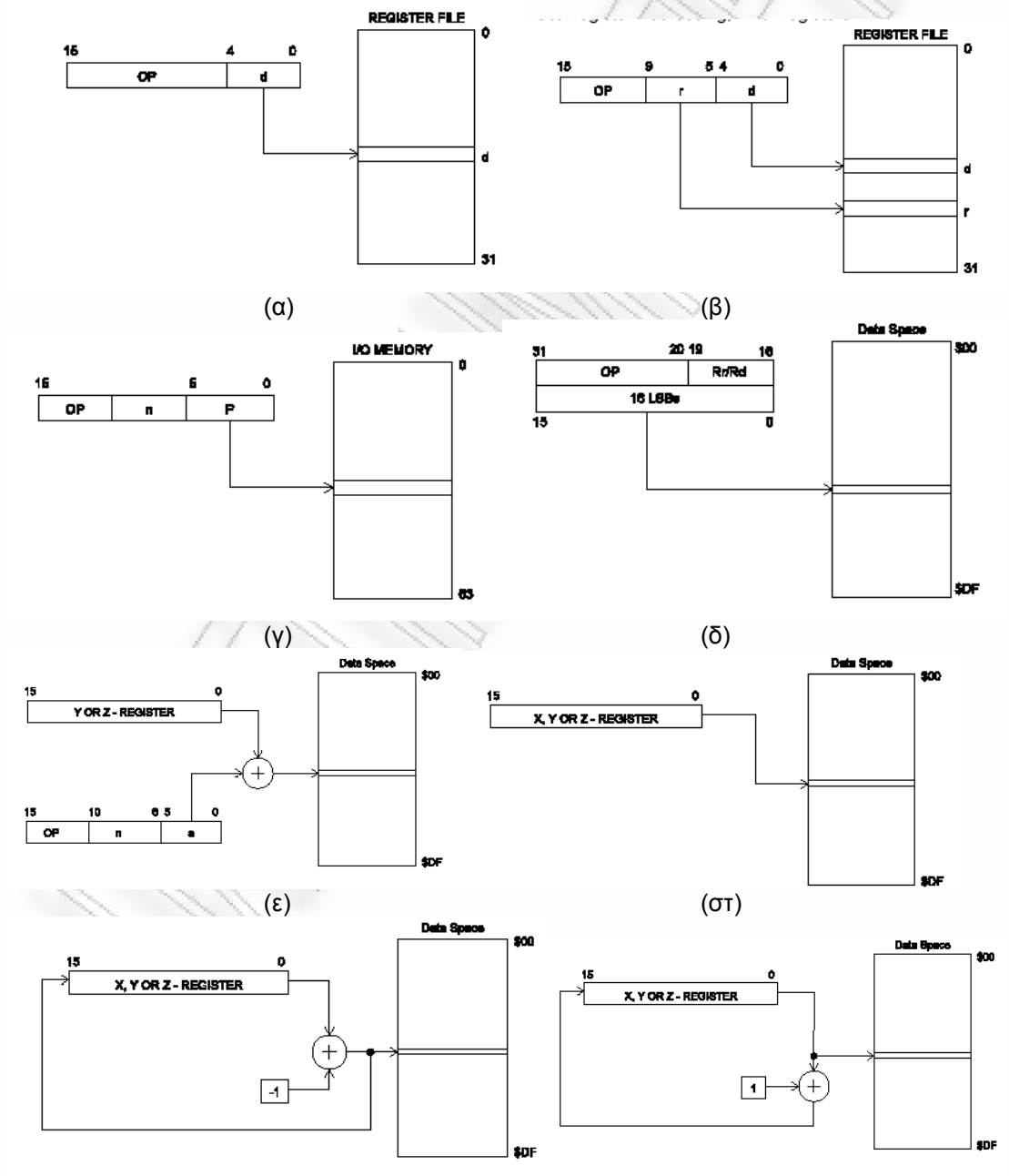
Εικόνα 6: Διάταξη της μνήμης SRAM

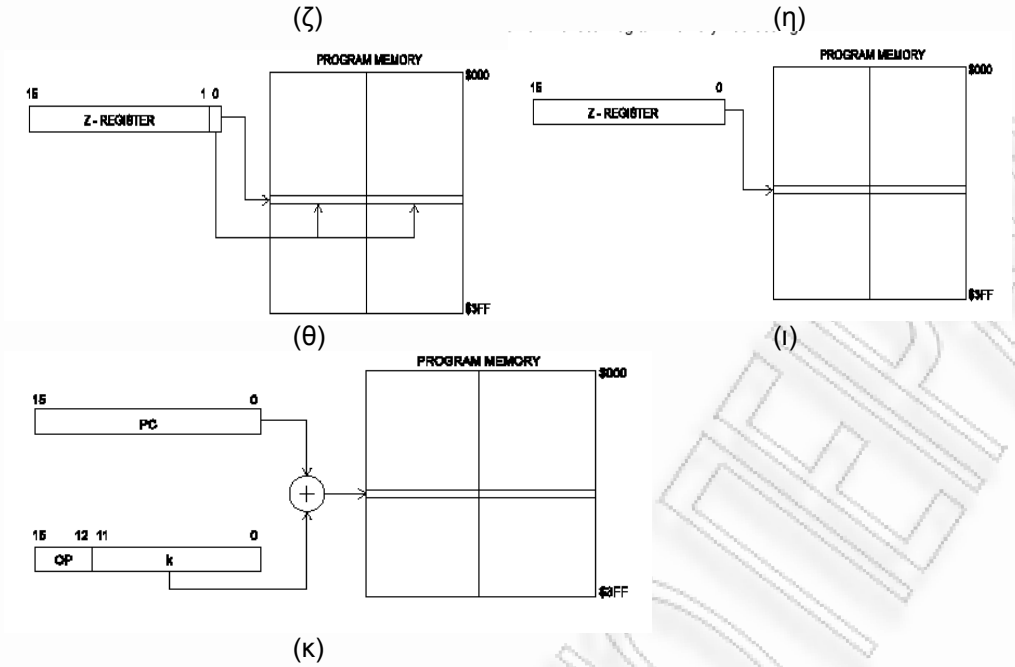
2.5.3 **Υποστηριζόμενοι Τρόποι Διευθυνσιοδότησης**

- Άμεση Διευθυνσιοδότηση (Direct Addressing). Το μοναδικό εντέλο είναι ένας καταχωρητής (Εικόνα 7.α).
- Άμεση με 2 καταχωρητές (Register Direct). Τα 2 ορίσματα μιας εντολής είναι καταχωρητές (Εικόνα 7.β).
- Άμεση σε καταχωρητή περιφερειακού (Direct I/O). Το ένα όρισμα είναι καταχωρητής εισόδου/εξόδου δηλαδή σχετικός με κάποιο περιφερειακό (Εικόνα 7.γ).
- Άμεση Δεδομένων (Data Direct). Η εντολή περιέχει ένα όρισμα που αναφέρεται σε καταχωρητή από τον οποίο θα ληφθούν ή θα αποθηκευθούν τα δεδομένα και ένα δεύτερο όρισμα των 16 bits που αναφέρεται σε κάποια θέση στη μνήμη δεδομένων (Εικόνα 7.δ).
- Έμμεση Δεδομένων με Εκτόπισμα (Data Indirect with Displacement). Το όρισμα της εντολής προστίθεται στην τιμή που περιέχει ένας από τους Y ή Z καταχωρητές και το αποτέλεσμα καθορίζει τη θέση στη μνήμη δεδομένων (Εικόνα 7.ε).
- Έμμεση Δεδομένων (Data Indirect). Το όρισμα βρίσκεται στη θέση μνήμης που καθορίζεται από έναν από τους X,Y, Z (Εικόνα 7.στ).
- Έμμεση Δεδομένων με Μείωση ή Αύξηση του Δείκτη (Data Indirect with Pre-decrement or post –increment. Στην πρώτη περίπτωση ο δείκτης (X,Y ή Z) μειώνεται κατά 1 και το

αποτέλεσμα καθορίζει τη θέση μνήμης στη μνήμη δεδομένων. Στην δεύτερη περίπτωση αφού χρησιμοποιηθεί ο δείκτης, κατόπιν η τιμή του αυξάνεται κατά 1 (Εικόνα 7.ζ/7.η).

- Σταθερή για προσπέλαση Μνήμης Προγράμματος (Constant using LPM). Χρησιμοποιείται από την εντολή LPM για τη φόρτωση ενός byte από τη μνήμη προγράμματος. Η διεύθυνση λαμβάνεται από τα 15 σημαντικότερα bits του Z ενώ το τελευταίο bit καθορίζει αν το byte θα είναι το περισσότερο ή λιγότερο σημαντικό (Εικόνα 7.θ).
- Έμμεση Προγράμματος (Program Indirect). Ο δείκτης Z καθορίζει τη θέση στη μνήμη προγράμματος την οποία θα φορτώσει ο Μετρητής Προγράμματος. Χρησιμοποιείται από εντολές διακλάδωσης όπως η IJMP, ICALL (Εικόνα 7.ι).
- Σχετική με Μετρητή Προγράμματος (Relative Program). Το όρισμα της εντολής αυξημένο κατά ένα, προστίθεται στο Μετρητή Προγράμματος. Χρησιμοποιείται από τις εντολές RJMP, RCALL (Εικόνα 7.κ).





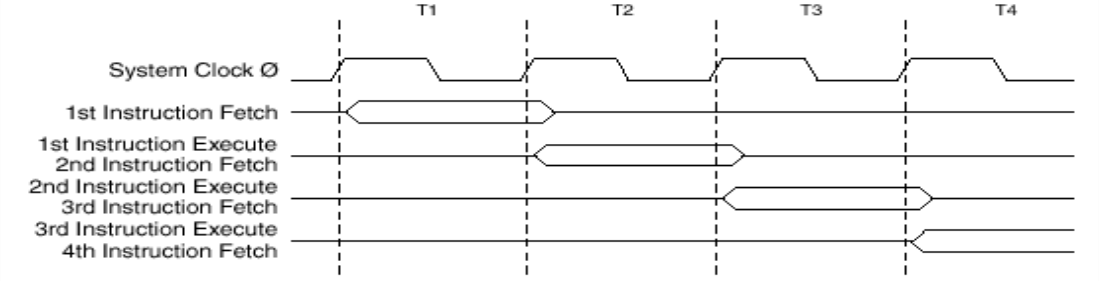
Εικόνα 7: Τρόποι Διευθυνσιοδότησης στον AVR

2.5.4 **Η μνήμη Δεδομένων EEPROM**

Ο ATmega8 περιέχει 512 bytes μνήμης EEPROM (Electrical Erasable Programmable Read Only Memory) για μόνιμη αποθήκευση δεδομένων. Όπως και η Flash Memory, η EEPROM μπορεί να διατηρήσει το περιεχόμενό της ακόμη και με απουσία τάσης. Είναι οργανωμένη σε ξεχωριστό χώρο δεδομένων και μπορούν να γραφτούν και να διαβαστούν μεμονωμένα Bytes.Η διάρκεια ζωής της είναι το λιγότερο 100.000 κύκλοι εγγραφής / διαγραφής. Για την πρόσβαση της CPU στην μνήμη EEPROM χρησιμοποιούνται ειδικοί καταχωρητές ο EEPROM καταχωρητής δεδομένων(EEDR) και ο EEPROM καταχωρητής ελέγχου (EECR) οι οποίοι περιγράφονται λεπτομερώς στο εγχειρίδιο του μικροελεγκτή. Και αυτό κάνει πολύ την όλη διαδικασία σαφώς πιο αργή από την προσπέλαση της εσωτερικής RAM του μικροελεγκτή. Ωστόσο, μερικές συσκευές της τεχνολογίας SecureAVR χρησιμοποιούν μια ειδική χαρτογράφηση της EEPROM στα δεδομένα ή την μνήμη του προγράμματος.

2.5.5 **Προσπέλαση Μνήμης Για Εκτέλεση Εντολών**

Ένα ιδιαίτερο χαρακτηριστικό των αρχιτεκτονικών Μειωμένου Συνόλου Εντολών (RISC) είναι η χρονικά επικαλυπτόμενη εκτέλεση εντολών με τη χρήση της τεχνικής διασωλήνωσης (pipelining). Οι AVR μικροελεγκτές, ως RISC που είναι, υποστηρίζουν πλήρως μια τέτοια επικάλυψη με αποτέλεσμα ενώ μια εντολή μπορεί να απαιτεί περισσότερο από έναν κύκλο ρολογιού για να εκτελεστεί, ο ρυθμός εκτέλεσης εντολών να είναι «Μία Εντολή ανά Κύκλο Ρολογιού». (Σχήμα 5-7).



Εικόνα 8: Επικάλυψη εντολών 2 κύκλων ρολογιού

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	11
0x3E (0x5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	13
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	13
0x3C (0x5C)	Reserved									
0x3B (0x5B)	GICR	INT1	INT0	–	–	–	–	IVSEL	IVCE	49, 67
0x3A (0x5A)	GIFR	INTF1	INTF0	–	–	–	–	–	–	68
0x39 (0x59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	–	TOIE0	72, 102, 122
0x38 (0x58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	–	TOV0	73, 103, 122
0x37 (0x57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	213
0x36 (0x56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	171
0x35 (0x55)	MCUCR	SE	SM2	SM1	SM0	ISC11	ISC10	ISC01	ISC00	33, 66
0x34 (0x54)	MCUCSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	41
0x33 (0x53)	TCCR0	–	–	–	–	–	CS02	CS01	CS00	72
0x32 (0x52)	TCNT0	Timer/Counter0 (8 Bits)								72
0x31 (0x51)	OSCCAL	Oscillator Calibration Register								31
0x30 (0x50)	SFIOR	–	–	–	–	ACME	PUD	PSR2	PSR10	58, 75, 123, 199
0x2F (0x4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	97
0x2E (0x4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	100
0x2D (0x4D)	TCNT1H	Timer/Counter1 – Counter Register High byte								101
0x2C (0x4C)	TCNT1L	Timer/Counter1 – Counter Register Low byte								101
0x2B (0x4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High byte								101
0x2A (0x4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low byte								101
0x29 (0x49)	OCR1BH	Timer/Counter1 – Output Compare Register B High byte								101
0x28 (0x48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low byte								101
0x27 (0x47)	ICR1H	Timer/Counter1 – Input Capture Register High byte								102
0x26 (0x46)	ICR1L	Timer/Counter1 – Input Capture Register Low byte								102
0x25 (0x45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	117
0x24 (0x44)	TCNT2	Timer/Counter2 (8 Bits)								119
0x23 (0x43)	OCR2	Timer/Counter2 Output Compare Register								119
0x22 (0x42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	119
0x21 (0x41)	WDTCR	–	–	–	WDCE	WDE	WDP2	WDP1	WDP0	43
0x20[1] (0x40)[1]	UBRRH	URSEL	–	–	–	UBRR[11:8]				158
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	156
0x1F (0x3F)	EEARH	–	–	–	–	–	–	–	EEAR8	20
0x1E (0x3E)	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	20
0x1D (0x3D)	EEDR	EEPROM Data Register								20
0x1C (0x3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	20
0x1B (0x3B)	Reserved									
0x1A (0x3A)	Reserved									
0x19 (0x39)	Reserved									
0x18 (0x38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	65
0x17 (0x37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	65
0x16 (0x36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	65
0x15 (0x35)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
0x14 (0x34)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
0x13 (0x33)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
0x12 (0x32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
0x11 (0x31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
0x10 (0x30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
0x0F (0x2F)	SPDR	SPI Data Register								131
0x0E (0x2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	131
0x0D (0x2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	129
0x0C (0x2C)	UDR	USART I/O Data Register								153
0x0B (0x2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	154
0x0A (0x2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	155
0x09 (0x29)	UBRRL	USART Baud Rate Register Low byte								158
0x08 (0x28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	194
0x07 (0x27)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	205
0x06 (0x26)	ADCSRA	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	207
0x05 (0x25)	ADCH	ADC Data Register High byte								208
0x04 (0x24)	ADCL	ADC Data Register Low byte								208
0x03 (0x23)	TWDR	Two-wire Serial Interface Data Register								173
0x02 (0x22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	173

Εικόνα 9: οι καταχωρητές του μικροελεγκτή

2.6 Ρολόι Συστήματος

Ο μικροελεγκτής για να λειτουργήσει πρέπει να παλμοδοτείται από ένα κύκλωμα χρονισμού. Ο ATmega8 μπορεί να παλμοδοτηθεί από τις πηγές που φαίνονται στον παρακάτω πίνακα και επιλέγονται από τα αντίστοιχα flash fuse bits CKSEL3...0

ΕΠΙΛΟΓΗ ΠΗΓΗΣ ΡΟΛΟΓΙΟΥ	CKSEL3...0
Εξωτερικός κρυσταλλικός / κεραμικός ταλαντωτής	1111 - 1010
Εξωτερικός κρύσταλλος χαμηλής συχνότητας	1001
Εξωτερικός RC ταλαντωτής	1000 - 0101
Εσωτερικός ρυθμιζόμενος RC ταλαντωτής	0100 - 0001
Εξωτερικό ρολόι	0000

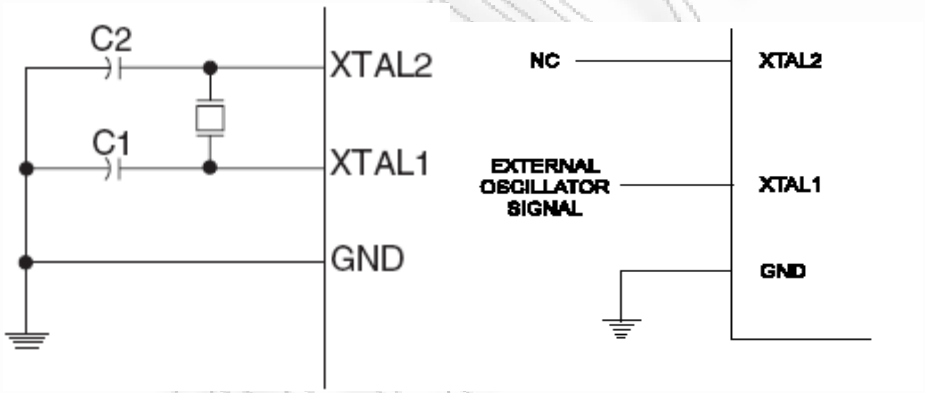
Συνήθως συναντάται σαν ένα δικτύωμα πυκνωτή - αντίσταση (RC), ή με κρύσταλλο και δύο κεραμικούς πυκνωτές (Quartz), ή με ένα πακτωμένο μικροσκοπικό κύκλωμα κρυστάλλου - πυκνωτών το λεγόμενο Resonator.

Όταν η CPU επανέρχεται μετά από διακοπή τροφοδοσίας η επιλεγμένη πηγή ρολογιού χρησιμοποιείται για να υπολογιστή ο χρόνος που θα ξεκινήσει η εκτέλεση των εντολών εξασφαλίζοντας σταθερή λειτουργία του ταλαντωτή πριν αρχίσει η εκτέλεση των εντολών. Όταν η CPU ξεκινά από RESET υπάρχει μια επιπρόσθετη καθυστέρηση επιτρέποντας στην τάση τροφοδοσίας να φτάσει σε ένα σταθερό επίπεδο πριν αρχίσει την κανονική λειτουργία.

Ο Watchdog ταλαντωτής χρησιμοποιείται για την χρονομέτρηση του τμήματος πραγματικού χρόνου του χρόνου ξεκινήματος της CPU. Η συχνότητα του Watchdog ταλαντωτή είναι εξαρτώμενη από την τάση τροφοδοσίας όπως φαίνεται στα τυπικά χαρακτηρίστηκα του μικροελεγκτή.

2.6.1 Ταλαντωτής Κρυστάλλου

Στους ακροδέκτες XTAL1/XTAL2 μπορεί να συνδεθεί ένας κρύσταλλος (crystal ή resonator) για να οδηγηθεί ο εσωτερικός ταλαντωτής είτε μέχρι τα 8 MHz ή τα 16 MHz ανάλογα με το μοντέλο του μικροελεγκτή. Η σύνδεση παρουσιάζεται στην Εικόνα 10. Επίσης το ολοκληρωμένο μπορεί να τροφοδοτηθεί με ένα εξωτερικό ρολόι το οποίο θα πρέπει να συνδεθεί στην είσοδο XTAL1. Στην περίπτωση αυτή η είσοδος XTAL2 δεν συνδέεται πουθενά.



Εικόνα 10: Συνδέσεις Ρολογιού.

2.7 Χειρισμός Διακοπών (Interrupt Handling)

Ο ATmega 8 έχει 19 πηγές διακοπών και ισάριθμες θέσεις στην περιοχή της Μνήμης Προγράμματος στις οποίες καθορίζεται η διεύθυνση της αντίστοιχης υπορουτίνας χειρισμού διακοπής (Interrupt Service Routine – ISR). Οι θέσεις αυτές είναι σταθερές και αποτελούν τον Πίνακα Διακοπών (Interrupt Vector) ο οποίος απεικονίζεται στον Πίνακα 4.

Ο κώδικας που καθορίζει την κλήση της αντίστοιχης ISR ρουτίνας μοιάζει συνήθως με τον εξής:

```
Address   Code                    Comment
000       rjmp    Reset_Hdl       ; Jump to Reset Handler
001       rjmp    Int0_Hdl        ; External Int0 handler
002       rjmp    Int1_Hdl        ; External Int1 handler
003       rjmp    T2_CompA_Hdl    ; T2 Output Compare A handler
004       rjmp    T2_OVF1_Hdl     ; T2 Overflow 1 handler
005       rjmp    T1_Capt_Hdl     ; T1 Input Capture handler
```

```
006     rjmp    T1_CompA_Hdl        ; T1 Output Compare A handler
007     rjmp    T1_CompB_Hdl        ; T1 Output Compare B handler
008     rjmp    T1_OVF_Hdl          ; T1 Overflow handler
009     rjmp    T0_OVF_Hdl          ; T0 Overflow handler
00a     rjmp    SPI_STC_comp_Hdl    ; Serial Transfer Complete
00b     rjmp    USART_RXC_comp_Hdl  ; USART, Rx Complete
00c     rjmp    USART_UDRE_emp_Hdl  ; USART Data Register Empty
00d     rjmp    USART_TXC_comp_Hdl  ; USART, Tx Complete
00e     rjmp    ADC_Comp_Hdl        ; ADC Conversion Complete
00f     rjmp    EE_RDY_Hdl          ; EEPROM Ready Handler
010     rjmp    ANA_COMP_Hdl        ; Analog Comparator
011     rjmp    TWI_Hdl             ; Two-wire Serial Interface Handler
012     rjmp    SPM_RDY_Hdl         ; Store Program Memory Ready

Reset_Hdl:
        ldi     r16, low            ; First instruction of Reset_Hdl handler
        out     SPL, r16
```

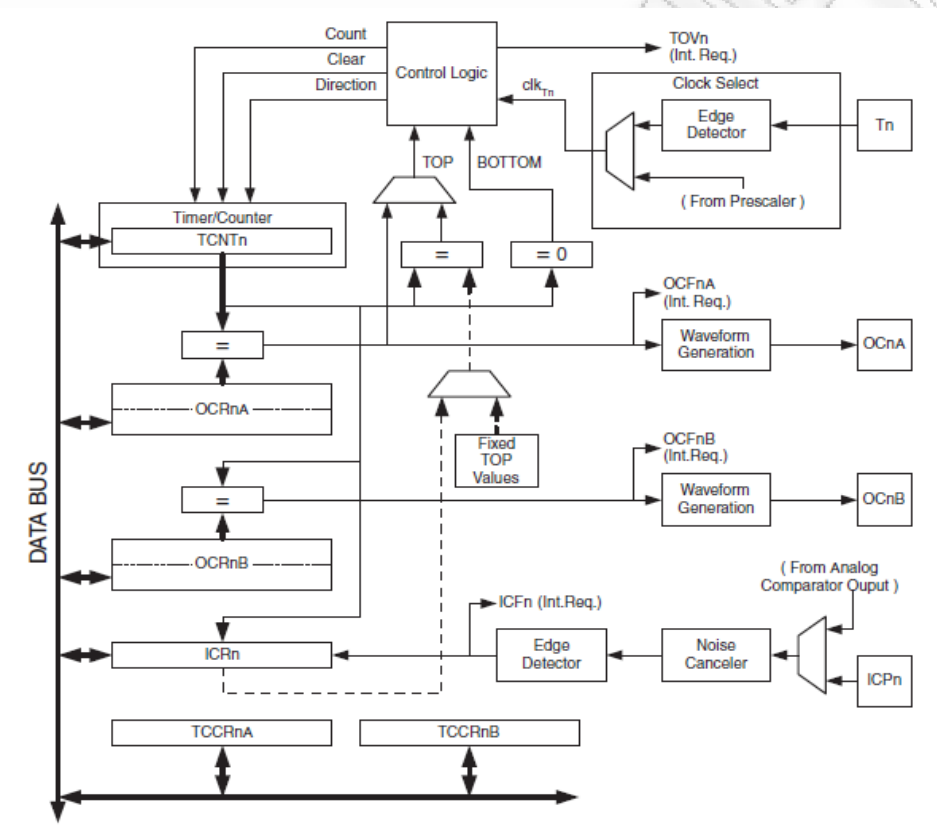
Στη θέση 0 υπάρχει μια κλήση στην υπορουτίνα Reset_Hdl. Το όνομα Reset_Hdl αντιστοιχεί στη θέση μνήμης (Προγράμματος) απ' όπου αρχίζει η υπορουτίνα για το χειρισμό Επανεκκίνησης. Αμέσως ακολουθούν 2 εντολές διακλάδωσης που καλούν τις ρουτίνες χειρισμού των διακοπών που προκαλούνται από τις εξωτερικές γραμμές INT0/INT1 και στην συνέχεια οι υπόλοιπες διακοπές.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

Πίνακας 4: Οι πηγές διακοπών του ATmega 8

2.8 Χρονιστές / Μετρητές (Timers/Counters)

Ο ATmega 8 διαθέτει 3 χρονιστές / μετρητές (timers/counters). Οι δύο είναι 8-bit και ονομάζεται T/C0 και T/C2 ενώ ο τρίτος είναι 16-bits και ονομάζεται T/C1. Και οι τρεις μπορούν να χρησιμοποιήσουν είτε εσωτερικό ρολόι οπότε λειτουργούν ως χρονιστές ή να χρησιμοποιήσουν κάποια εξωτερική πηγή παλμών των οποίων το πλήθος να μετράνε οπότε λειτουργούν ως μετρητές. Στον T/C0 και T/C1 η συχνότητα της πηγής των παλμών μπορεί να υποδιαιρεθεί με τις τιμές /8, /64, /256, /1024 μέσω του κοινού prescaler. Ο T/C2 μπορεί να λειτουργήσει και σαν γεννήτρια κυματομορφών PWM.
Ο δεύτερος χρονιστής/μετρητής T/C1 είναι περισσότερο πολύπλοκος και διαθέτει 16-bits. Η αρχιτεκτονική του παρουσιάζεται στο παρακάτω σχήμα.



Εικόνα 11: Η αρχιτεκτονική του Timer/Counter 1.

3 ΚΕΦΑΛΑΙΟ 3

Συστήματα Αυτομάτου Ελέγχου

3.1 Εισαγωγή

Σύστημα αυτομάτου ελέγχου ονομάζεται ένα σύνολο (τεχνητό ή φυσικό) στοιχείων και εξαρτημάτων, κατάλληλα συνδεδεμένα μεταξύ τους αλληλεπιδρούν επιτελώντας συγκεκριμένο έργο.

Όπως και με τις περισσότερες ειδικότητες της εφαρμοσμένης μηχανικής έτσι και ο αυτόματος έλεγχος έχει τις ρίζες του στην ανάγκη για επίλυση πρακτικών προβλημάτων. Ο αυτόματος έλεγχος απασχολεί το ανθρώπινο γένος για πάνω από δύο χιλιάδες χρόνια. Η ώθηση για την ανάπτυξη της συγκεκριμένης επιστήμης δόθηκε από τέσσερα σημαντικά γεγονότα:

- Την ακριβή μέτρηση του χρόνου από τους αρχαίους Έλληνες και τους Άραβες.
- Τη βιομηχανική επανάσταση στην Ευρώπη.
- Την εμφάνιση μαζικής επικοινωνίας μετά από τον Α' και Β' Παγκόσμιο Πόλεμο.
- Την αρχή της εποχής του διαστήματος και των υπολογιστών.

Τα συστήματα αυτομάτου ελέγχου είναι παντού γύρω μας, από τα ποιο απλά πράγματα της καθημερινής μας χρήσης μέχρι τα ποιο περίπλοκα συστήματα σε εργοστάσια και σε διαστημικούς σταθμούς ακόμη και μέσα μας. Πολλά περίπλοκα συστήματα ελέγχου περιλαμβάνονται στις λειτουργίες του ανθρώπινου σώματος. Ένα εξεζητημένο σύστημα ελέγχου που βρίσκεται στον υποθάλαμο του εγκεφάλου διατηρεί την θερμοκρασία του σώματος στους 37 ^{o}C παρά τις αλλαγές στην εξωτερική θερμοκρασία και την σωματική δραστηριότητα. Σ' ένα άλλο περίτεχνο σύστημα, το μάτι, η διάμετρος της κόρης ρυθμίζεται αυτόματα ώστε να ελέγξει το ποσό του φωτός που φτάνει στον αμφιβληστροειδή χιτώνα.

Η πράξη του ραψίματος και η οδήγηση ενός αυτοκινήτου είναι δυο τρόποι κατά τους οποίους το ανθρώπινο σώμα συμπεριφέρεται σαν ένας πολύ εξελιγμένος ελεγκτής. Τα μάτια είναι οι αισθητήρες που ανιχνεύουν την θέση της βελόνας και του νήματος, ή τη θέση του αυτοκινήτου όσον αφορά το κέντρο του δρόμου. Ένας περίπλοκος ελεγκτής, ο εγκέφαλος, συγκρίνει τις δυο θέσεις και καθορίζει τις ενέργειες που πρέπει να γίνουν για να έχουμε το επιθυμητό αποτέλεσμα. Το σώμα εκτελεί την κίνηση του ελέγχου απλά κουνώντας τη βελόνα ή στρέφοντας το τιμόνι του αυτοκινήτου. Ένας έμπειρος οδηγός θα είναι προετοιμασμένος για όλες τις μορφές διαταραχών που μπορεί να εμφανιστούν στο σύστημα όπως ένα κακό σημείο οδοστρώματος ή ένα όχημα που κινείται με χαμηλότερη ταχύτητα. Είναι εξαιρετικά δύσκολο να αναπαράγουμε με τη μορφή ενός συστήματος αυτομάτου ελεγκτή τις τόσες πολλές κρίσεις και αποφάσεις που είναι σε θέση να λάβει το μέσω άτομο καθημερινά και ασυναίσθητα.

Τα συστήματα αυτομάτου ελέγχου χρησιμοποιούνται σήμερα παντού π.χ. για να ρυθμίζουν την θερμοκρασία σπιτιών, σχολικών και γενικά μεγάλων κτιρίων. Στην παραγωγή αγαθών καθώς χρησιμοποιούνται για να διασφαλίσουν την αγνότητα και ομοιογένεια των προϊόντων καθώς και την ποιότητα προϊόντων που προέρχονται από χαρτοβιομηχανίες, βιομηχανίες σιδήρου, εργοστάσια χημικών, παραγωγής ενέργειας και άλλους τύπους εργοστασίων παραγωγής. Τα συστήματα ελέγχου βοηθούν στην προστασία του περιβάλλοντος ελαχιστοποιώντας τον όγκο των αποβλήτων που θα πρέπει να πεταχτούν, μειώνοντας έτσι τα κατασκευαστικά κόστη και ελαχιστοποιώντας το πρόβλημα απόρριψης των απορριμμάτων. Τα αποχετευτικά συστήματα και η διαχείριση των αποβλήτων είναι επίσης ένας τομέας ευρείας χρήσης των συστημάτων ελέγχου.

Ένα σύστημα ελέγχου είναι οποιαδήποτε ομάδα από εξαρτήματα η οποία πετυχαίνει ένα επιθυμητό αποτέλεσμα ή διατηρεί την τιμή μιας μεταβλητής σταθερή. Από τα προηγούμενα παραδείγματα είναι φανερό ότι μια μεγάλη γκάμα από εξαρτήματα / συσκευές μπορεί να είναι

μέρος ενός και μόνο συστήματος ελέγχου, είτε αυτά είναι ηλεκτρικά, ηλεκτρονικά, μηχανικά, υδραυλικά, πνευματικά, ανθρώπινα ή οποιοσδήποτε συνδυασμός αυτών. Το επιθυμητό αποτέλεσμα είναι μια τιμή κάποιας μεταβλητής στο σύστημα. Για παράδειγμα, η κατεύθυνση ενός αυτοκινήτου, η θερμοκρασία ενός δωματίου, η στάθμη του υγρού σε μια δεξαμενή ή η πίεση του αέρα σε μια σωλήνα. Η μεταβλητή της οποίας η τιμή ελέγχεται ονομάζεται μεταβλητή ελέγχου (controlled variable).

3.2 Ιστορία Του Αυτομάτου Ελέγχου

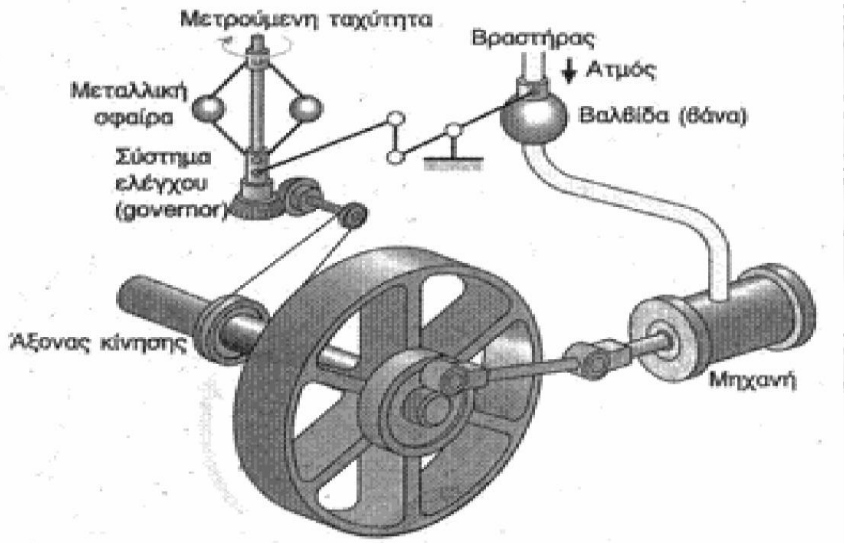
Η χρήση της ανάδρασης στην διαδικασία ελέγχου ενός συστήματος έχει μια εξαιρετικά ενδιαφέρουσα ιστορία. Η πρώτη εφαρμογή ανάδρασης σημειώθηκε στην Αρχαία Ελλάδα με αφορμή την ανάπτυξη ενός μηχανισμού τύπου φλοτέρ (ρυθμιστής στάθμης υγρών), κατά την περίοδο του 300 Π.Χ.

Ο Κτεσίβιος, ένας Έλληνας που ζούσε στην Αλεξάνδρεια, λέγεται ότι δημιούργησε μια αυτορυθμιζόμενη συσκευή ροής η οποία χρησιμοποιήθηκε στα ρολόγια νερού της εποχής. Ένα τέτοιο ρολόι χρησιμοποιεί ένα δείκτη στάθμης ώστε να σημειώνει το πέρασμα της ώρας. Η ακρίβειά του εξαρτάται από το πόσο σταθερή θα είναι η εισροή του νερού στη δεξαμενή. Ο ρυθμός της ροής μπορεί να αλλάξει αν αλλάξει και η πίεση του νερού. Ο Κτεσίβιος εισήγαγε μια δεύτερη δεξαμενή μ' ένα φλοτέρ και μια βαλβίδα σχεδιασμένα να ρυθμίζουν τη ροή του εισερχόμενου νερού στη δεύτερη δεξαμενή. Έτσι το νερό στη δεύτερη δεξαμενή μένει σταθερό και άρα η ροή στην κεντρική δεξαμενή του ρολογιού παρέμενε σταθερή. Αν υποθέσουμε ότι η ροή αυξάνεται, ανεβαίνει και η στάθμη στην δεύτερη δεξαμενή. Τότε, το φλοτέρ κλείνει την βαλβίδα και μειώνεται η εισροή στη δεύτερη δεξαμενή. Έτσι η στάθμη πέφτει ξανά και τότε η βαλβίδα ανοίγει πάλι. Με τον τρόπο αυτό κατάφερε να διατηρεί σταθερή τη ροή του νερού και άρα πέτυχε τη δημιουργία μιας σταθερής μονάδας μέτρησης του χρόνου.

Κατά το 250 π.χ., ο Φίλωνας παρουσιάζει την πρώτη λάμπα λαδιού στην οποία γίνονταν χρήση ενός μηχανισμού φλοτέρ για την διατήρηση σταθερής στάθμης του καύσιμου λαδιού. Ο Χείρων της Αλεξάνδρειας, που έζησε στον πρώτο αιώνα μ.χ., εξέδωσε ένα βιβλίο με τίτλο «Pneumatica», όπου αναφέρονταν διάφοροι μηχανισμοί ρύθμισης στάθμης υγρών με χρήση φλοτέρ.

Το πρώτο σύστημα αυτομάτου ελέγχου κλειστού βρόγχου που εφευρέθηκε στην Ευρώπη ήταν ο ρυθμιστής θερμοκρασίας του Cornelis Drebbel (1572 – 1633) από την Ολλανδία. Ο Dennis Papin (1647 – 1712) εφηύρε τον πρώτο ρυθμιστή πίεσης σε ατμολέβητες, το 1681. Ο ρυθμιστής πίεσης του Papin αποτελεί μια ασφαλιστική διάταξη παρόμοια με την βαλβίδα εκτόνωσης που χρησιμοποιείται στις χύτρες ταχύτητας.

Το πρώτο, κατά γενική ομολογία, σύστημα αυτομάτου ελέγχου κλειστού βρόγχου που εφαρμόστηκε σε βιομηχανικό περιβάλλον, ήταν η συσκευή ελέγχου κινούμενης σφαίρας (flyball governor) του James Watt, η οποία εφευρέθηκε το 1769 και αφορούσε σε ένα σύστημα ελέγχου της ταχύτητας μιας ατμομηχανής. Η μηχανική αυτή διάταξη, που φαίνεται στην εικόνα 11, μετρούσε την ταχύτητα περιστροφής του άξονα κίνησης και χρησιμοποιούσε την κίνηση μιας μεταλλικής σφαίρας για τον έλεγχο μιας βάνας (βαλβίδας) και κατά συνέπεια το ποσό του ατμού που τροφοδοτούνταν στην μηχανή. Όσο αυξάνονταν η ταχύτητα περιστροφής η σφαίρα εξαιτίας της φυγόκεντρης δύναμης ανασηκώνονταν και η κίνησή της απομακρύνονταν από τον άξονα συμμετρίας του κυλινδρικού άξονα κίνησης προκαλώντας με τον τρόπο αυτό το κλείσιμο της βάνας. για την κίνηση της σφαίρας απαιτείται ένα ποσό ενέργειας που προσφέρεται από την ίδια την μηχανή, οπότε η μέτρηση της περιστροφικής ταχύτητας διεξάγεται με σχετικά μικρή ακρίβεια.



Εικόνα 12: Η συσκευή ελέγχου κινούμενης σφαίρας «flyball governor» του James Watt.

Το πρώτο ιστορικά, σύστημα αυτομάτου ελέγχου, σύμφωνα με σχετικό ισχυρισμό των Ρώσων, ήταν ένας αυτόματος ρυθμιστής στάθμης νερού με διακόπτη τύπου φλοτέρ που εφευρέθηκε από τον I. Polzunov το 1765. Ο διακόπτης φλοτέρ ανιχνεύει την στάθμη του νερού ελέγχοντας με τον τρόπο αυτό το άνοιγμα και το κλείσιμο μιας βάνας η οποία με την σειρά της ελέγχει την εισαγωγή νερού στον βραστήρα. Η χρονική περίοδος πριν από το 1868 χαρακτηρίστηκε γενικά από την ανάπτυξη διαφόρων εμπνευσμένων και έξυπνων συστημάτων αυτομάτου ελέγχου. Ταυτόχρονα, οι διάφορες προσπάθειες με σκοπό την αύξηση της ακρίβειας των συστημάτων, οδήγησαν στην εμφάνιση αργών εξασθενίσεων των μεταβατικών ταλαντώσεων καθώς επίσης και σε ασταθή συστήματα. Έτσι δημιουργήθηκε η επιτακτική ανάγκη της ανάπτυξης μιας αντίστοιχης θεωρίας για τον αυτόματο έλεγχο. Ο J. C. Maxwell διετύπωσε μια θεωρία σχετικά με τον αυτόματο έλεγχο, βασισμένη σε ένα μαθηματικό μοντέλο διαφορικής εξίσωσης ενός ελεγκτή. Η μελέτη αυτή του Maxwell αφορούσε κυρίως στην επίδραση των διαφόρων παραμέτρων του συστήματος πάνω στην συνολική του συμπεριφορά. Κατά την ίδια χρονική περίοδο, ο I. A. Vyshnegradskii διετύπωσε μια αντίστοιχη θεωρία βασισμένη στα μαθηματικά, σχετικά με τα συστήματα ρυθμιστών Πριν από τον Δεύτερο Παγκόσμιο Πόλεμο, οι διάφορες θεωρίες που αναπτύχθηκαν στις Ηνωμένες Πολιτείες και στην Δυτική Ευρώπη, είχαν διατυπωθεί με αρκετά διαφορετικό τρόπο σε σχέση με εκείνες που αναπτύχθηκαν από την πλευρά της Ανατολικής Ευρώπης και της Ρωσίας. Την ώθηση για την χρήση συστημάτων ανάδρασης στις Ηνωμένες Πολιτείες, αποτέλεσε η ανάπτυξη της τηλεφωνίας και των ηλεκτρονικών ενισχυτών ανάδρασης από τους Bode, Nyquist και Black, της εταιρίας Bell Telephone Laboratories. Η έννοια του πεδίου της συχνότητας (frequency domain) χρησιμοποιήθηκε πρωταρχικά με σκοπό να περιγράψει την λειτουργία των ενισχυτών ανάδρασης (feedback amplifiers) σε σχέση με το εύρος ζώνης καθώς επίσης και με άλλες αντίστοιχες μεταβλητές. Σε αντίθεση με όλα αυτά, διάφοροι διαπρεπείς μαθηματικοί αλλά και διάφοροι εμπειρικοί μηχανικοί στην πρώην Σοβιετική Ένωση, ενέπνευσαν και κυριάρχησαν στον χώρο της θεωρίας του αυτομάτου ελέγχου. Έτσι, η θεωρητική προσέγγιση των Σοβιετικών διακατέχεται από μια τάση προς το πεδίο του χρόνου (time domain) διατυπωμένη κατάλληλα με την βοήθεια διαφορικών εξισώσεων.

Μια ακόμη μεγαλύτερη ώθηση στην θεωρία καθώς και την αντίστοιχη πρακτική των συστημάτων αυτομάτου ελέγχου, απετέλεσε η κήρυξη του Δευτέρου Παγκοσμίου Πολέμου, όπου ανέκυψαν διάφορες επιτακτικές ανάγκες για την σχεδίαση και κατασκευή διαφόρων συστημάτων αυτόματων πιλότων, ελέγχου θέσης οπλικών συστημάτων, ελέγχου κεραιών συστημάτων ραδιοναυτιλίας (ραντάρ) καθώς και πλήθος άλλων συστημάτων για στρατιωτική

χρήση, που βασίζονταν στην ιδέα ελέγχου με την βοήθεια της ανάδρασης. Η πολυπλοκότητα καθώς και η αναμενόμενη συμπεριφορά των παραπάνω στρατιωτικών συστημάτων δημιούργησε την ανάγκη για επέκταση των διαφόρων τεχνικών του αυτομάτου ελέγχου και έστρεψε το γενικότερο ενδιαφέρον προς τα συστήματα αυτομάτου ελέγχου καθώς επίσης και στην ανάπτυξη νέων γνωστικών πεδίων και σχετικών μεθόδων. Πριν από το 1940 οι διαδικασίες σχεδίασης των συστημάτων αυτομάτου ελέγχου βρίσκονταν ως επί το πλείστον σε επίπεδο εμπειρικών μεθόδων βασισμένων σε τεχνικές δοκιμής και σφάλματος (trial and error). Στην διάρκεια της δεκαετίας του 1940 διατυπώνονταν με ταχείς ρυθμούς διάφορες μαθηματικές και αναλυτικές μέθοδοι και προοδευτικά το αντικείμενο του αυτομάτου ελέγχου μετατράπηκε σε επίπεδο ιδιαίτερης εφαρμοσμένης επιστήμης.

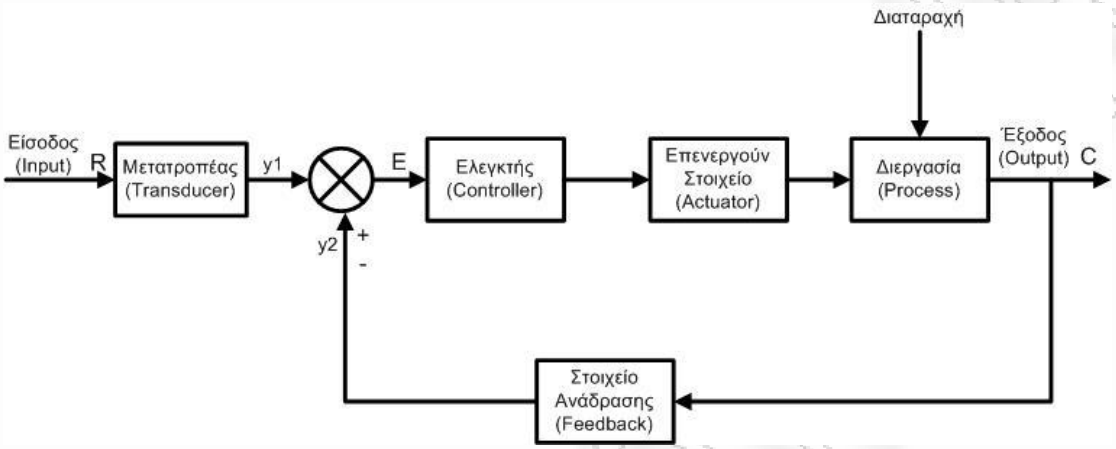
Οι διάφορες τεχνικές ανάλυσης στο πεδίο της συχνότητας συνέχισαν να κυριαρχούν στον χώρο του αυτομάτου ελέγχου και μετά τον Δεύτερο Παγκόσμιο Πόλεμο, με ιδιαίτερη έμφαση στην χρήση των μετασχηματισμών Laplace καθώς και του πεδίου της μιγαδικής συχνότητας. Κατά την διάρκεια της δεκαετίας του 1950, στα πλαίσια της αντίστοιχης θεωρίας του αυτομάτου ελέγχου δόθηκε αρκετή έμφαση στην ανάπτυξη και την χρήση μεθόδων του πεδίου της μιγαδικής μεταβλητής s (s-plane) και ιδιαίτερα της μεθόδου του γεωμετρικού τόπου ριζών. Στην δεκαετία του 1980, η χρήση των ψηφιακών υπολογιστών και των αντίστοιχων σχετικών διεργασιών, ως στοιχεία ελέγχου, γίνεται ρουτίνα. Η αντίστοιχη τεχνολογία των σύγχρονων για την εποχή εκείνη συστημάτων, με την βοήθεια των οποίων οι διάφοροι υπολογισμοί εκτελούνταν με μεγάλη ταχύτητα και ακρίβεια, αποδεικνύονταν απαγορευτική για το σύνολο των μηχανικών σχεδίασης.

Με την αποστολή του διαστημοπλοίου Sputnik και την εμφάνιση της εποχής του διαστήματος, δόθηκε μια νέα μεγάλη ώθηση στον χώρο του αυτομάτου ελέγχου. Έτσι δημιουργήθηκε μια νέα ανάγκη σχεδιασμού πολύπλοκων συστημάτων αυτομάτου ελέγχου υψηλής ακρίβειας για πυραύλους και διατάξεις ανιχνευτών για διαστημική χρήση. Επιπλέον, η ανάγκη ελαχιστοποίησης του συνολικού βάρους των δορυφόρων καθώς επίσης και η ανάγκη ελέγχου υψηλής ακρίβειας, έφερε στο προσκήνιο το αντικείμενο του βέλτιστου ελέγχου. Εξαιτίας των απαιτήσεων αυτών, τις τελευταίες δύο δεκαετίες οι διάφορες μέθοδοι που αναπτύχθηκαν στο πεδίο του χρόνου από τους Liapunov, Minorsky και άλλους, γνώρισαν ιδιαίτερα ζωηρό ενδιαφέρον. Προς την ίδια κατεύθυνση συνεισέφεραν αρκετά και διάφορες σύγχρονες θεωρίες βέλτιστου ελέγχου που αναπτύχθηκαν από τον L. S. Pontryagin στην Σοβιετική Ένωση καθώς και από τον R. Bellman στις Ηνωμένες Πολιτείες, όπως επίσης και διάφορες πρόσφατες μελέτες σχετικά με την τεχνολογία των εύρωστων συστημάτων (robust systems). Σήμερα πλέον είναι ξεκάθαρο ότι τόσο η ανάλυση στο πεδίο του χρόνου, όσο και στο πεδίο της συχνότητας, θα πρέπει να λαμβάνονται εξίσου υπόψη κατά τις διαδικασίες ανάλυσης και σχεδίασης συστημάτων αυτομάτου ελέγχου.

Στο μέλλον, Η χρήση των ρομπότ σε αυτό τον τομέα θα γίνει περισσότερο διαδεδομένη. Η ρομποτική έχει πολλά να συνεισφέρει σε περιοχές όπου η εργασία είναι επικίνδυνη ή ανθυγιεινή. Υψηλής ταχύτητας συστήματα μεταφοράς σε στεριά, θάλασσα και αέρα θα απαιτούν καλύτερα συστήματα ελέγχου ώστε να παρέχεται στους επιβάτες επιπλέον άνεση και ασφάλεια. Μέσω πιο αυστηρών απαιτήσεων για ασφάλεια και αποδοτικότητα σε όλους τους τομείς, υπάρχει μεγάλη προοπτική ανάπτυξης μέσω της θεωρίας και της πρακτικής των συστημάτων ελέγχου με ανάδραση.

3.3 Δομή Συστημάτων Αυτομάτου Ελέγχου

Η δομή ενός χαρακτηριστικού συστήματος αυτομάτου ελέγχου φαίνεται στο παρακάτω μπλοκ διάγραμμα.



Εικόνα 13: Δομικό διάγραμμα συστήματος αυτόματου ελέγχου κλειστού βρόγχου

Τα βασικά εξαρτήματα των Συστημάτων Αυτόματου Ελέγχου σύμφωνα με το παραπάνω μπλοκ διάγραμμα είναι τα ακόλουθα :

- Είσοδος (input)

  Μια διέγερση που εφαρμόζεται στο σύστημα από εξωτερική πηγή.
- Μετατροπέας (transducer)

  Μετατρέπει μια μορφή ενέργειας σε μια άλλη π.χ. μηχανική σε ηλεκτρική.
- Αθροιστής

  Είναι συσκευή που αθροίζει αλγεβρικά τα εισερχόμενα σήματα για να παράγει ένα σήμα εξόδου. Συνήθως αναφέρεται και σαν συγκριτής ή ανιχνευτής σφάλματος.
- Ελεγκτής (Controller)

  Σε όλα σχεδόν τα συστήματα ελέγχου η είσοδος του ελεγκτή είναι το σφάλμα που παράγεται από τον αθροιστή στα συστήματα ελέγχου κλειστού βρόγχου ή την ίδια την είσοδο στα συστήματα ελέγχου ανοικτού βρόγχου. Είναι μηχανισμός ελέγχου που παράγει μια έξοδο που οδηγεί την ελεγχόμενη διεργασία με σκοπό τον μηδενισμό του σφάλματος και γενικά την βελτιστοποίηση των χαρακτηριστικών του συστήματος.
- Ελεγχόμενη διεργασία

  Κάθε φυσική ποσότητα όπως θερμοκρασία, πίεση, ή στάθμη υγρού μπορεί να ελεγχθεί μέσω διεργασίας που περιλαμβάνει κάθε τι που επηρεάζει τις φυσικές μεταβλητές. Μ'άλλα λόγια, η ελεγχόμενη διεργασία περιλαμβάνει κάθε τι που απαιτείται για τον έλεγχο της φυσικής ποσότητας.
- Ελεγχόμενη μεταβλητή [c(t)]

  Η ελεγχόμενη μεταβλητή είναι μία φυσική ποσότητα όπως θερμοκρασία, πίεση κ.λ.π. που πρέπει να ελεγχθεί από το σύστημα. Συνήθως αναφέρεται σαν έξοδος. Το σύστημα διεγειρόμενο από την είσοδο παράγει ένα σήμα εξόδου σαν απόκριση.
- Επενεργούν στοιχείο (Actuator)

  Το Επενεργούν Στοιχείο είναι η συσκευή που αποδίδει την απαιτούμενη ενέργεια (π.χ. κινητική) στην διεργασία (π.χ. η συσκευή που αναγκάζει την διεργασία να εξασφαλίσει την έξοδο).

- Σύστημα (Plant)
    - Σύστημα τύπου follow-up: Τα συστήματα των οποίων η έξοδος θα πρέπει να μεταβάλλεται σε συνάρτηση των μεταβολών του σήματος εισόδου (π.χ. σύστημα ελέγχου θερμοκρασίας χώρου).
    - Σύστημα τύπου regulator: Τα συστήματα των οποίων η έξοδος θα πρέπει να παραμένει σταθερή ακόμα και όταν υπάρχουν μεταβολές του σήματος εισόδου (π.χ. σταθεροποιητής τάσεως DC).
- Διαταραχή (disturbance)

  Διαταραχή είναι κάθε μη επιθυμητό σήμα που επηρεάζει την έξοδο.
- Ανάδραση (feedback)

  Ένα σύστημα χρησιμοποιεί ανάδραση εάν η έξοδος ή μέρος της εξόδου επιστρέφει μέσω του κλάδου ανατροφοδότησης (ανάδρασης) στον αθροιστή, έτσι που να μπορεί να συγκριθεί με την είσοδο. Η χρήση της ανάδρασης συνήθως επιφέρει ευστάθεια και ακρίβεια στο σύστημα. Ένα σύστημα μπορεί να χρησιμοποιεί πολλές αναδράσεις. Πάντως πρωτεύουσα ανάδραση είναι εκείνη όπου το σήμα εξόδου επιστρέφει και συγκρίνεται με την είσοδο. Αν δεν υπάρχει καμία επικοινωνία μεταξύ εισόδου και εξόδου έχουμε σύστημα ανοιχτού βρόγχου, ενώ αν κάθε φορά παίρνουμε την έξοδο την ελέγχουμε και την οδηγούμε σε μια είσοδο αναφοράς έχουμε σύστημα κλειστού βρόγχου. Ο κλάδος (δρόμος) που οδηγεί την έξοδο στην είσοδο λέγεται κλάδος ανάδρασης. Αν το σήμα εξόδου προστίθεται στην είσοδο έχουμε θετική ανάδραση και αν αφαιρείται αρνητική ανάδραση.
- Κύκλωμα αντιστάθμισης

  Κύκλωμα που χρησιμοποιείται για να τροποποιήσει την συνάρτηση μεταφοράς και κατ' επέκταση την έξοδο του συστήματος έτσι που να είναι η επιθυμητή. Τα πλέον συνηθισμένα είναι: προπορείας, υστέρησης, προπορείας - υστέρησης κ.λ.π.
- Σφάλμα κλειστού Σ.Α.Ε. είναι η διαφορά της εισόδου και της εξόδου.
- Απ΄ ευθείας δρόμος (forward path) είναι ο δρόμος από το σημείο άθροισης μέχρι την έξοδο.

3.4 Κατηγορίες Συστημάτων Αυτόματου Ελέγχου(Σ.Α.Ε.)

Τα συστήματα αυτομάτου ελέγχου μπορούμε να τα κατατάξουμε σε κατηγορίες ως εξής:

- Ανάλογα με τη φύση του μέσου ελέγχου
- Ανάλογα με το αν χρησιμοποιείται ή όχι ανάδραση (ανατροφοδότηση)
- Ανάλογα με την τεχνική επεξεργασία των σημάτων ελέγχου.
- Ανάλογα με τον τύπο των εξαρτημάτων
- Ανάλογα με την εφαρμογή τους

3.4.1 Ανάλογα Με Τη Φύση Του Μέσου Ελέγχου

- Ηλεκτρικά – ηλεκτρονικά συστήματα

  Το μέσο ελέγχου είναι ένα ηλεκτρικό σήμα, που μπορεί να ενισχυθεί κατάλληλα με ηλεκτρικούς ενισχυτές και να διεγείρει κάποιον ηλεκτρικό κινητήρα, που με την σειρά του θα κάνει ανάλογη διόρθωση στον μηχανισμό με τον οποίο είναι συνδεδεμένος.

<u>Πλεονεκτήματα</u>

1. Έχουν μεγάλο βαθμό απόδοσης.
2. Δεν χρειάζονται αεροσυμπιεστές, αντλίες.
3. Είναι καθαρά συστήματα και τυχόν διαρροές δεν προκαλούν βλάβες σε διπλανές συσκευές.

Μειονεκτήματα

1. Μικρή συγκέντρωση ισχύος (δηλαδή λόγος μικρός ωφέλιμος ισχύς / βάρος εγκατάστασης)
2. Προκαλούν σπινθήρες και δεν μπορούν να χρησιμοποιηθούν σε εύφλεκτο περιβάλλον.
3. Η συντήρησή τους είναι πολύπλοκη.
4. Συνοδεύονται με ακριβές ηλεκτρονικές μονάδες.

- Πνευματικά συστήματα.

  Το μέσο ελέγχου είναι αέρας με πίεση που προέρχεται από κάποιο συμπιεστή (compressor) και διεγείρει κάποια βαλβίδα που μέσω κυλίνδρου με έμβολο κάνει την αναγκαία διόρθωση.

Πλεονεκτήματα

1. Είναι πολύ απλά στην κατασκευή και τη λειτουργία τους.
2. Χρησιμοποιούν αέρα που είναι ελεύθερος στην φύση και δεν εγκυμονεί κινδύνους πυρκαγιάς. (Η τυχόν διαρροή αέρα δεν προκαλεί βλάβες σε διπλανά συστήματα).
3. Οι μεταβολές της θερμοκρασίας του περιβάλλοντος δεν έχουν επίδραση στο ιξώδες του μέσου λειτουργίας.
4. Η συντήρηση τους είναι εύκολη και γίνεται σε αραιά χρονικά διαστήματα.
5. Αποθηκεύεται ποσότητα αέρα σε αεροφυλάκιο και έτσι λειτουργούν για λίγο, και μετά την διακοπή του αεροσυμπιεστή.

Μειονεκτήματα

1. Είναι αναγκαία η ύπαρξη τουλάχιστον δύο αεροσυμπιεστών, αεροφυλακίου και κατάλληλου δικτύου σωλήνων.
2. Παρουσιάζουν δυσκολία στην λίπανση των κινούμενων μερών.
3. Λόγω της συμπιεστότητας του αέρα δεν έχουμε ταχεία ανάπτυξη δύναμης στην έξοδο του συστήματος ούτε ακρίβεια εξόδου.

- Υδραυλικά συστήματα.

  Το μέσο ελέγχου είναι συνήθως λάδι που έρχεται από κάποια αντλία, ελέγχεται από βαλβίδα και κινεί υδραυλικό κινητήρα ή έμβολο κυλίνδρου.

Πλεονεκτήματα

1. Υψηλή απόδοση ακόμα και σε μεγάλη ισχύ και ταχύτητα.
2. Μπορούμε να επιτύχουμε μεγάλη ισχύ εξόδου.
3. Το λάδι είναι ασυμπίεστο και έχουμε ταχύτητα απόκρισης.
4. Αυτολίπανση με το κυκλοφορούν λάδι.
5. Σε περίπτωση ανωμαλίας ο επενεργητής (υδραυλικός κινητήρας ή κύλινδρος με έμβολο) παραμένει στην θέση του, γιατί υπάρχουν ειδικές ανεπίστροφες βαλβίδες στις γραμμές παροχής και επιστροφής.

Μειονεκτήματα

1. Συχνή συντήρηση των δικτύων υψηλής πίεσης για να μην υπάρχουν διαρροές.
2. Η τυχόν διαρροή λαδιού εγκυμονεί κινδύνους βλάβης των διπλανών συσκευών.
3. Με την αύξηση της θερμοκρασίας μεταβάλλεται το ιξώδες του λαδιού και φυσικά η απόδοση του συστήματος.
4. Η ύπαρξη λαδιού δημιουργεί ένα επιπλέον κόστος στην συντήρηση.
5. Αν στο υδραυλικό δίκτυο υπάρχει θυλάκιο αέρα δεν έχουμε ακρίβεια στην έξοδο (φαινόμενο αερισμού).

- Ηλεκτρουδραυλικά συστήματα

  Η βασική διαφορά τους με τα προηγούμενα είναι ότι η ρυθμιστική βαλβίδα δεν είναι μηχανική αλλά ηλεκτρική δηλαδή λειτουργεί ανάλογα με κάποιο ηλεκτρικό σήμα που θα της στείλουμε από μακριά και καθορίζει έτσι το μέγεθος και τη φορά της ροής λαδιού προς τον υδραυλικό κινητήρα (ή έμβολο).

- Ηλεκτροπνευματικά συστήματα

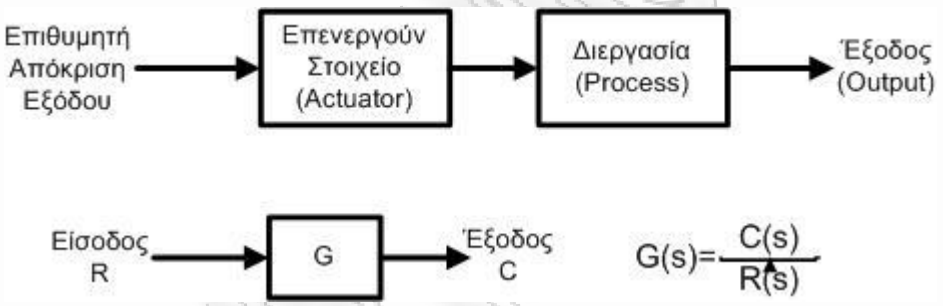
  Η βασική διαφορά τους με τα πνευματικά συστήματα είναι στο ότι διαθέτουν Ηλεκτροπνευματική βαλβίδα που μπορεί να ρυθμιστεί από μακριά με κάποιο ηλεκτρικό σήμα.

3.4.2 Ανάλογα Με Το Εάν Χρησιμοποιείται ή Όχι Ανατροφοδότηση

- Συστήματα ανοιχτού βρόγχου

  Ένα σύστημα ανοιχτού βρόγχου χρησιμοποιεί μία ενεργό συσκευή (που παράγει το σήμα εισόδου) για να ελέγξει απευθείας την διεργασία χωρίς την χρήση ανατροφοδότησης.

  1. Είναι πολύ απλής κατασκευής.
  2. Η ακρίβεια εξαρτάται από τη ρύθμιση διαφόρων στοιχείων.
  3. Δεν παρουσιάζουν συνήθως προβλήματα αστάθειας.

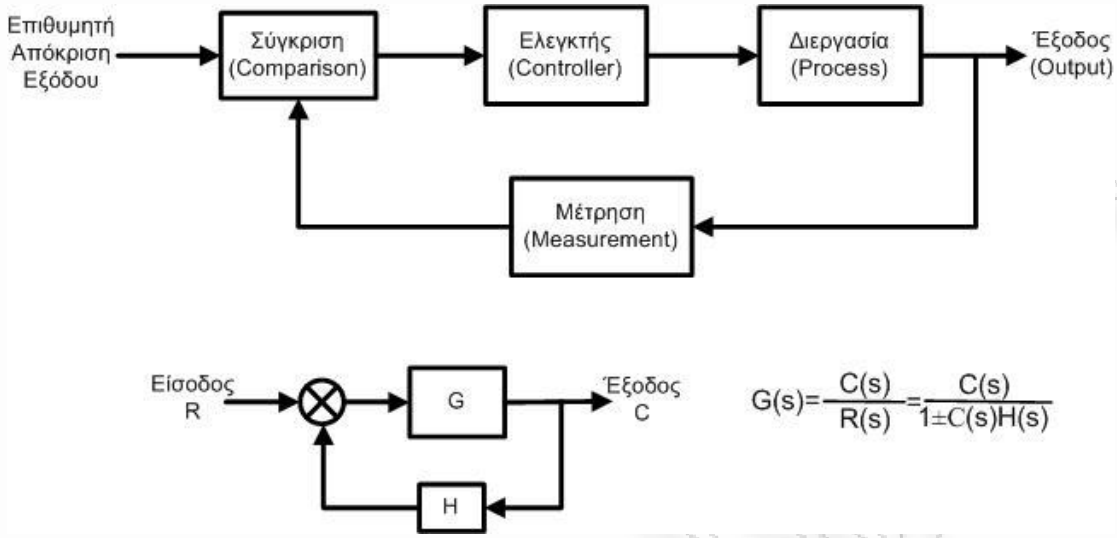


Εικόνα 14: Δομικό διάγραμμα συστήματος ελέγχου ανοιχτού βρόγχου

- Συστήματα Ελέγχου Κλειστού Βρόγχου

  Ένα σύστημα κλειστού βρόγχου χρησιμοποιεί τη μέτρηση του σήματος εξόδου και την ανατροφοδοτεί για να συγκριθεί με το σήμα αναφοράς (είσοδος – επιθυμητή έξοδος).

  1. Μεγάλη ακρίβεια.
  2. Πολυπλοκότερα συστήματα.
  3. Παρουσιάζουν προβλήματα αστάθειας.
  4. Έχουν εύρος λειτουργίας.



Εικόνα 15: Δομικό διάγραμμα συστήματος ελέγχου κλειστού βρόγχου

3.4.3 **Ανάλογα Με Την Τεχνική Επεξεργασία Των Σημάτων Ελέγχου**

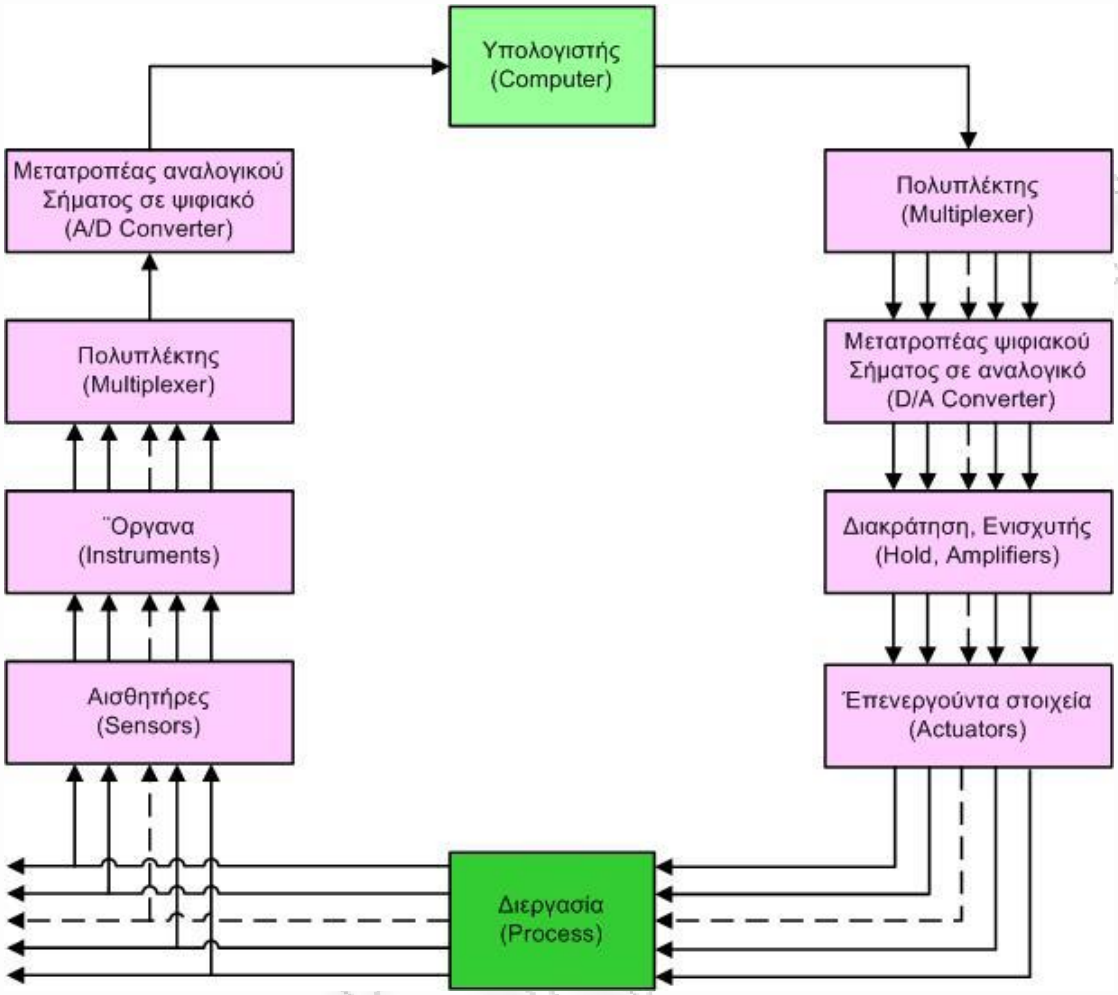
- Αναλογικά

  Οι ελεγκτές υλοποιούνται με αναλογικά εξαρτήματα (ηλεκτρονικά, μηχανικά, πνευματικά κ.λ.π.) και η επεξεργασία των σημάτων γίνεται συνεχώς στον χρόνο.

- Ψηφιακά

  Στα ψηφιακά συστήματα ελέγχου οι ελεγκτές υλοποιούνται με υπολογιστές και η επεξεργασία των σημάτων γίνεται σε διακριτά σημεία του χρόνου.

Το ψηφιακό σύστημα ελέγχου περιλαμβάνει τα εξαρτήματα που φαίνονται στο διάγραμμα που ακολουθεί.



Εικόνα 16: Δομικό διάγραμμα ψηφιακού συστήματος ελέγχου

3.4.4 **Ανάλογα Με Των Τύπο Των Εξαρτημάτων**

- Γραμμικά

  Ένα σύστημα θεωρείται γραμμικό όταν ακολουθεί την αρχής της επαλληλίας.

  Π.χ. Αν όλες οι αρχικές συνθήκες ενός συστήματος είναι μηδενικές, δηλαδή αν το σύστημα είναι σε ηρεμία, τότε το σύστημα είναι γραμμικό αν έχει την ακόλουθη ιδιότητα: Αν

  (α) μία είσοδος $u_1(t)$ παράγει μια έξοδο $y_1(t)$, και

  (β) μία είσοδος $u_2(t)$ παράγει μια έξοδο $y_2(t)$, τότε,

  (γ) η είσοδος $c_1u_1(t)+ c_2u_2(t)$ παράγει μια έξοδο $c_1y_1(t)+ c_2y_2(t)$, για οποιοδήποτε ζευγάρι εισόδων $u_1(t)$ και $u_2(t)$ και σταθερές c_1 και c_2.

  Τα γραμμικά συστήματα μπορούν συχνά ν/α παρασταθούν με γραμμικές διαφορικές εξισώσεις και γραμμικές εξισώσεις διαφοράς.

- Μη γραμμικά

  Όλα τα υπόλοιπα είναι μη γραμμικά

  Π.χ. Οι συνήθεις διαφορικές εξισώσεις $(dy/dt)^2 + y = 0$ και $d^2 y/d^2 t + \cos y = 0$

  είναι μη – γραμμικές διότι ο όρος της πρώτης εξίσωσης είναι $(dy/dt)^2$ είναι δευτέρου βαθμού, και ο όρος $\cos y$ στην δεύτερη εξίσωση δεν είναι πρώτου βαθμού πράγμα που ισχύει για όλες τις υπερβατικές συναρτήσεις.

3.4.5 **Ανάλογα Με Την Εφαρμογή Τους**

- Σερβομηχανισμοί

  Είναι εκείνα τα συστήματα ελέγχου των οποίων η έξοδος ή ελεγχόμενη μεταβλητή είναι μηχανική θέση η ρυθμός μεταβολής της μηχανικής θέσης (ταχύτητα ή επιτάχυνση). Συστήματα ελέγχου ταχύτητας περιστροφής άξονα κινητήρα DC και ελέγχου θέσεως βηματικού κινητήρα είναι χαρακτηριστικά παραδείγματα σερβομηχανισμών.

- Αριθμητικά συστήματα ελέγχου.

  Είναι εκείνα τα συστήματα που ενεργούν επί αριθμητικών δεδομένων που είναι αποθηκευμένα σε κάποιο αποθηκευτικό μέσω ηλεκτρονικής ή άλλης φύσεως.

  Τα CNC είναι χαρακτηριστικό παράδειγμα αριθμητικού συστήματος ελέγχου.

- Ακολουθιακά συστήματα ελέγχου.

  Είναι τα συστήματα που η λειτουργία τους είναι προδιαγεγραμμένη και προσδιορισμένης χρονικής διάρκειας.

- Συστήματα πολύπλοκων διεργασιών (Βιομηχανικά συστήματα)

  Παραδείγματα τέτοιων συστημάτων θα μπορούσαν να είναι: Μονάδες παραγωγής ηλεκτρικής ενέργειας, Μονάδες συναρμολόγησης οχημάτων, Μονάδες κλωστοϋφαντουργίας, Διυλιστήρια, Μονάδες βιολογικού καθαρισμού, κ.λ.π.

Θα πρέπει να τονιστεί ότι η κατάταξη ενός_συστήματος σε μια από τις παραπάνω κατηγορίες δεν είναι πάντα δυνατή και αυτό διότι ένα σύστημα μπορεί να εμπεριέχει χαρακτηριστικά που θα επέτρεπαν την κατάταξή του σε περισσότερες από μια από τις παραπάνω κατηγορίες.

Ιδιαίτερο ενδιαφέρον έχει η κατηγορία των συστημάτων ανοικτού και κλειστού βρόγχου (όποιας φύσεως και αν είναι αυτά) διότι εμπεριέχει την έννοια της αυτοματοποίησης όπως αυτή εννοείται όταν δεν παρεμβαίνει ο άνθρωπος.

3.5 **Μέθοδοι Ελέγχου.**

Μια συνεχής διαδικασία έχει αδιάκοπες εισόδους και εξόδους. Η τιμή τουλάχιστον μιας εισόδου αλλάζει με τέτοιο τρόπο που τείνει να διατηρεί την ελεγχόμενη μεταβλητή σταθερή στο σημείο ρύθμισης. Ένα από τα σημαντικότερα χαρακτηριστικά ενός ελεγκτή είναι ο τρόπος που χρησιμοποιεί το σφάλμα για να δημιουργήσει την ενέργεια ελέγχου. Οι διαφορετικοί τρόποι που ο ελεγκτής δημιουργεί την ενέργεια ελέγχου ονομάζονται μέθοδοι ελέγχου(modes of control). Η έξοδος ενός ελεγκτή συνεχούς διαδικασίας καθορίζεται από μία η περισσότερες μεθόδους ελέγχου. Οι συνήθεις μέθοδοι ελέγχου είναι οι εξής:

- Μέθοδος ελέγχου δύο θέσεων.
- Κινητή μέθοδος ελέγχου.
- Αναλογική μέθοδο ελέγχου.
- Μέθοδος ελέγχου ολοκληρώματος.
- Διαφορική μέθοδο ελέγχου.

Οι ελεγκτές συνεχών διαδικασιών μπορούν να χωριστούν σε δύο κατηγορίες: (1) σε αυτούς που το σημείο ρύθμισης είναι σταθερό για μεγάλο χρονικό διάστημα και (2) σε αυτούς που το σημείο ρύθμισης αλλάζει συνεχώς. Τα συστήματα ελέγχου της πρώτης κατηγορίας ονομάζονται σταθεροποιητές και της δεύτερης σερβοσυστήματα. Παρόλα αυτά οι μέθοδοι ανάλυσης και σχεδίασης συστημάτων ελέγχου δουλεύουν εξίσου καλά σε συστήματα και των δυο κατηγοριών.

Τα στοιχεία λογικού ελέγχου σχεδιάζονται για να ενεργήσουν στο σήμα (σφάλματος) ενεργοποίησης για να παραγάγουν το σήμα ελέγχου. Ο αλγόριθμος που εφαρμόζεται για αυτόν

το λόγο είναι ο νόμος ελέγχου ή η ενέργεια ελέγχου. Ένα μη μηδενικό σήμα σφάλματος προκύπτει είτε από μια αλλαγή στην εντολή είτε από μια διαταραχή. Ο γενικός ρόλος του ελεγκτή είναι να κρατήσει την ελεγχόμενη μεταβλητή κοντά στην επιθυμητή της τιμή όταν αυτή εμφανίζεται. Πιο συγκεκριμένα, οι στόχοι ελέγχου μπορούν να δηλωθούν ως εξής:

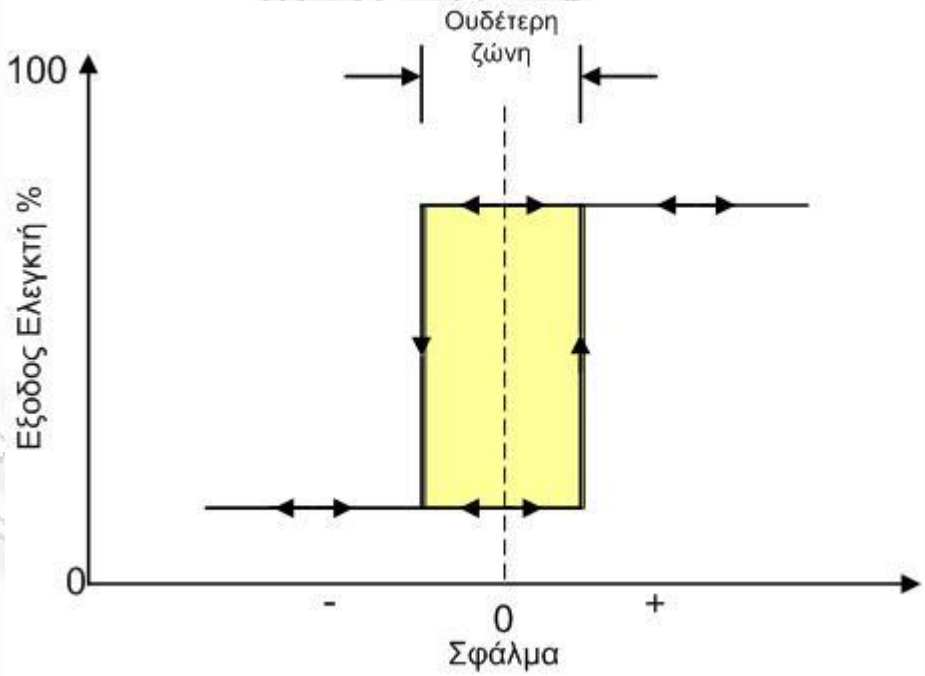
1 Ελαχιστοποίηση του σφάλματος σταθερής κατάστασης.
2 Ελαχιστοποίηση του χρόνου διευθέτησης.
3 Ικανοποίηση άλλων παροδικών προδιαγραφών, όπως η ελαχιστοποίηση της μεγίστης υπέρβασης.

Από πρακτικής άποψης, οι προδιαγραφές σχεδίασης για ένα ελεγκτή είναι πιο λεπτομερής. Παραδείγματος χάριν, το εύρος ζώνης πρέπει επίσης να διευκρινιστεί μαζί με ένα περιθώριο ασφάλειας για τη σταθερότητα. Δεν ξέρουμε ποτέ με βεβαιότητα τις αληθινές αριθμητικές τιμές των παραμέτρων του συστήματος, επομένως κάποια σχέδια ελεγκτών μπορούν να είναι πιο ευαίσθητα σε τέτοιες αβεβαιότητες παραμέτρου από κάποια άλλα σχέδια.

3.5.1 **Μέθοδος Ελέγχου Δύο Θέσεων.**

Η μέθοδος ελέγχου δύο θέσεων (two position control mode) είναι η απλούστερη και λιγότερο δαπανηρή μέθοδος ελέγχου. Η έξοδος του ελεγκτή έχει μόνο δύο πιθανές τιμές, ανάλογα με το πρόσημο του λάθος.

Αν οι δύο θέσεις είναι πλήρως ανοικτό και πλήρως κλειστό, ο ελεγκτής ονομάζεται "ελεγκτής on-off". Οι ελεγκτές δύο σημείων έχουν μια ουδέτερη ζώνη για να αποτρέψουν την άσκοπη λειτουργία. Η ουδέτερη ζώνη είναι ένα εύρος τιμών κοντά στο 0 όπου δεν εκτελείται καμία ενέργεια ελέγχου. Το σφάλμα πρέπει να περάσει από αυτή την ουδέτερη ζώνη πριν οποιαδήποτε ενέργεια ελέγχου αρχίσει να εκτελείται.

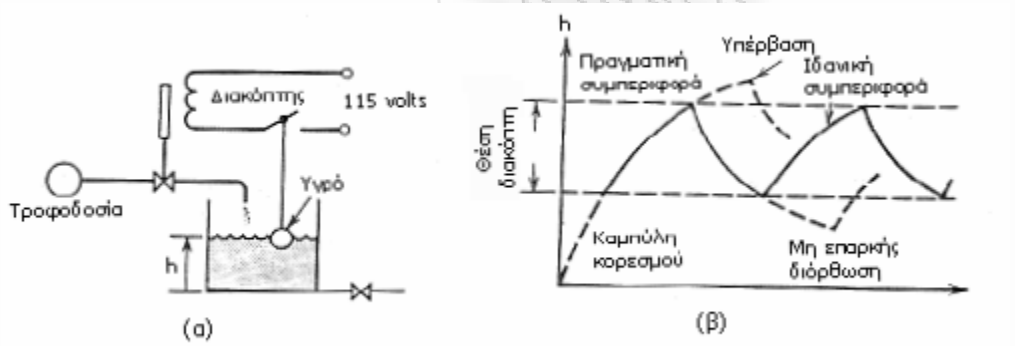


Εικόνα 17: Καμπύλη εισόδου/ εξόδου ενός ελεγκτή δύο θέσεων

Η μέθοδος ελέγχου δύο θέσεων παρέχει παλμούς ενέργειας στην διαδικασία, οι οποίοι προκαλούν μια επαναλαμβανόμενη κίνηση στην ελεγχόμενη μεταβλητή. Το πλάτος της κυκλικής αυτής κίνησης εξαρτάται από τρεις παράγοντες: (1) την χωρητικότητα της διαδικασίας, (2) την υστέρηση νεκρού χρόνου της διαδικασίας και (3) το μέγεθος της αλλαγής φορτίου που είναι ικανή να χειριστεί η διαδικασία. Το πλάτος της ταλάντωσης μειώνεται είτε αυξάνοντας την χωρητικότητα, είτε μειώνοντας την υστέρηση νεκρού χρόνου, είτε μειώνοντας το μέγεθος της αλλαγής φορτίου που μπορεί να διαχειριστεί η διαδικασία. Ο έλεγχος δύο θέσεων είναι κατάλληλος για διαδικασίες που έχουν αρκετά μεγάλη χωρητικότητα για να αντικρούσουν την συνδυασμένη επίδραση της υστέρησης νεκρού χρόνου και της δυνατότητας αλλαγής του φορτίου της διαδικασίας. Ο έλεγχος δύο θέσεων είναι απλός και φτηνός. Η χρήση του προτιμάται όπου η επαναλαμβανόμενη κίνηση μπορεί να μειωθεί σε ανεκτό βαθμό.

Ένα παράδειγμα μιας εφαρμογής ενός on-off ελεγκτή είναι ένα σύστημα επιπέδου υγρού που παρουσιάζεται στην Εικόνα 18α. Η χρονική απόκριση φαίνεται στην Εικόνα 18β και απεικονίζεται με τη σκούρα γραμμή για ένα ιδανικό σύστημα στο οποίο η βαλβίδα ελέγχου ενεργεί στιγμιαία. Η ελεγχόμενη μεταβλητή παλινδρομεί με ένα πλάτος που εξαρτάται από το πλάτος της ουδέτερης ζώνης. Αυτή η ζώνη αποτρέπει το συνεχές άνοιγμα και κλείσιμο το οποίο μπορεί να μικρύνει τον χρόνο ζωής της συσκευής. Η κυκλική συχνότητα εξαρτάται επίσης από την χρονική σταθερά της ελεγχόμενης διαδικασίας και του μεγέθους του σήματος ελέγχου.

Σε ένα πραγματικό σύστημα, σε αντιδιαστολή με το ιδανικό, ο αισθητήρας και η βαλβίδα ελέγχου δεν θα αποκριθούν στιγμιαία, αλλά θα έχουν τις χρονικές σταθερές τους. Η βαλβίδα δεν θα κλείσει τη στιγμή που το ύψος θα φθάσει στο επιθυμητό επίπεδο, αλλά θα υπάρξει κάποια καθυστέρηση κατά τη διάρκεια της οποίας η ροή συνεχίζεται στη δεξαμενή. Το αποτέλεσμα παρουσιάζεται από τη διακεκομμένη γραμμή στην Εικόνα 18β. Το αντίθετο εμφανίζεται όταν ανοίγεται η βαλβίδα. Αυτή η ανεπιθύμητη επίδραση μπορεί να μειωθεί με τη μείωση της ουδέτερης ζώνης, αλλά αυξάνεται η συχνότητα εάν γίνεται αυτό.



Εικόνα 18: (α)Έλεγχος στάθμης υγρού με on – off ελεγκτή. (β)Χρονική απόκριση.

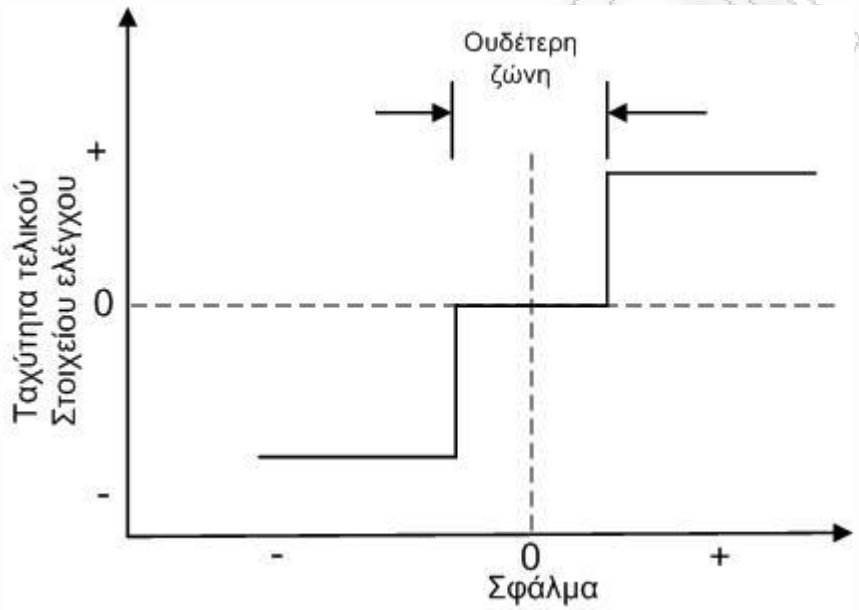
Η υπέρβαση και μη επαρκής διόρθωση στον on-off έλεγχο είναι αποδεκτά μόνο όταν η σταθερά χρόνου της διαδικασίας είναι μεγάλη σε σχέση με την χρονική καθυστέρηση των στοιχείων ελέγχου. Αυτή η καθυστέρηση σχετίζεται με τις χρονικές σταθερές των στοιχείων καθώς επίσης και με την απόστασή τους από τις εγκαταστάσεις. Εάν η βαλβίδα ελέγχου στην Εικόνα 18α είναι μακριά προς τα πάνω από τη δεξαμενή, μια σημαντική καθυστέρηση μπορεί να υπάρξει μεταξύ του χρόνου της δράσης ελέγχου και της επίδρασής της στις εγκαταστάσεις. Μια άλλη πηγή χρονικής καθυστέρησης είναι η ικανότητα του ιδίου του ελεγκτή.

Ένα οικιακό σύστημα θέρμανσης είναι ένα άλλο παράδειγμα ενός συστήματος ελέγχου δύο θέσεων. Ο αέρας στο σπίτι έχει μια σχετικά μεγάλη θερμική χωρητικότητα και ο νεκρός χρόνος υστέρησης είναι μικρός. Ο ρυθμός εισαγωγής θερμότητας από τον λέβητα είναι ικανός να θερμάνει το σπίτι στην χειρότερη μέρα του χειμώνα, και είναι μικρός σε σύγκριση με την χωρητικότητα του δωματίου. Η θερμότητα του δωματίου ταλαντώνεται σε ένα πλάτος που είναι αρκετά ανεκτό για τα όρια της ανθρώπινης άνεσης. Αυτό είναι ένα καλό παράδειγμα της μεθόδου ελέγχου δύο σημείων. Οι δύο θέσεις παρέχουν εισόδους ίσες με το μέγιστο και το ελάχιστο φορτίο της διαδικασίας (από καθόλου θερμότητα μέχρι θερμότητα ικανή για την πιο

κρύα ημέρα). Μια κακή σχεδίαση είναι αυτή που χρησιμοποιεί έναν λέβητα 10 φορές μεγαλύτερο από ότι χρειάζεται. Ο μεγάλος ρυθμός εισόδου θερμότητας από έναν υπερμεγέθη λέβητα έχει ως αποτέλεσμα μεγάλο πλάτος ταλάντωσης.

3.5.2 **Κινητή Μέθοδος Ελέγχου**

Η κινητή μέθοδος ελέγχου(floating control mode) είναι μια ειδική εφαρμογή της μεθόδου δύο σημείων στην οποία το τελικό στοιχείο ελέγχου είναι στάσιμο όσο το σφάλμα παραμένει στην ουδέτερη ζώνη. Όταν το σφάλμα είναι εκτός της νεκρής ζώνης, το τελικό στοιχείο ελέγχου αλλάζει με έναν σταθερό ρυθμό σε μια κατεύθυνση που καθορίζεται από το πρόσημο του σφάλματος. Το τελικό στοιχείο ελέγχου συνεχίζει να αλλάζει, μέχρι το σφάλμα να επιστρέψει στην νεκρή ζώνη ή μέχρι το τελικό στοιχείο ελέγχου να φτάσει στις οριακές του θέσεις. Η καμπύλη εισόδου /εξόδου ενός ελεγκτή κινητής μεθόδου απεικονίζεται στην Εικόνα 19.



Εικόνα 19: Καμπύλη εισόδου/ εξόδου ενός ελεγκτή κινητής μεθόδου.

Η μέθοδος αυτή έχει την τάση να παρουσιάζει ταλάντωση στην ελεγχόμενη μεταβλητή. Το πλάτος της ταλάντωσης εξαρτάται από τον νεκρό χρόνο υστέρησης της διαδικασίας, την χωρητικότητα της διαδικασίας και την ταχύτητα με την οποία ο ελεγκτής αυξομειώνει το τελικό στοιχείο ελέγχου. Η ταχύτητα του τελικού στοιχείου ελέγχου καθορίζει την γρηγορότερη αλλαγή φορτίου με την οποία μπορεί να συγχρονιστεί ο ελεγκτής, αλλά όχι και το μέγεθος αυτής. Το κύριο πλεονέκτημα του κινητού ελέγχου είναι η ικανότητα του να χειρίζεται αλλαγές φορτίου ρυθμίζοντας σταδιακά το τελικό στοιχείο ελέγχου. Όπως και με τον έλεγχο δύο σημείων, το πλάτος μειώνεται αυξάνοντας την χωρητικότητα, μειώνοντας τον νεκρό χρόνο υστέρησης ή μειώνοντας την ταχύτητα του τελικού στοιχείου ελέγχου. Ο κινητός έλεγχος χρησιμοποιείται όταν προσμένονται μεγάλες αλλαγές φορτίου και η χωρητικότητα είναι αρκετά μεγάλη ώστε να αντικρούσει την επίδραση του νεκρού χρόνου υστέρησης και την ταχύτητα του τελικού στοιχείου ελέγχου. Ο κινητός έλεγχος χρησιμοποιείται συχνά διότι είναι εγγενής στον τύπο του ενεργοποιητή που χρησιμοποιείται για να κατευθύνει το τελικό στοιχείο ελέγχου (για παράδειγμα ηλεκτρικά μοτέρ και υδραυλικοί κύλινδροι που λειτουργούν με ρελέ on-off ή σωληνοειδής βαλβίδες).

3.5.3 **Αναλογική Μέθοδος Ελέγχου, P**

Η αναλογική μέθοδος ελέγχου(proportional control mode) παράγει μια αλλαγή στην έξοδο του ελεγκτή που είναι ανάλογη του σήματος σφάλματος. Υπάρχει μια σταθερή γραμμική σχέση μεταξύ της τιμής της ελεγχόμενης μεταβλητής και της θέσης του τελικού στοιχείου ελέγχου.

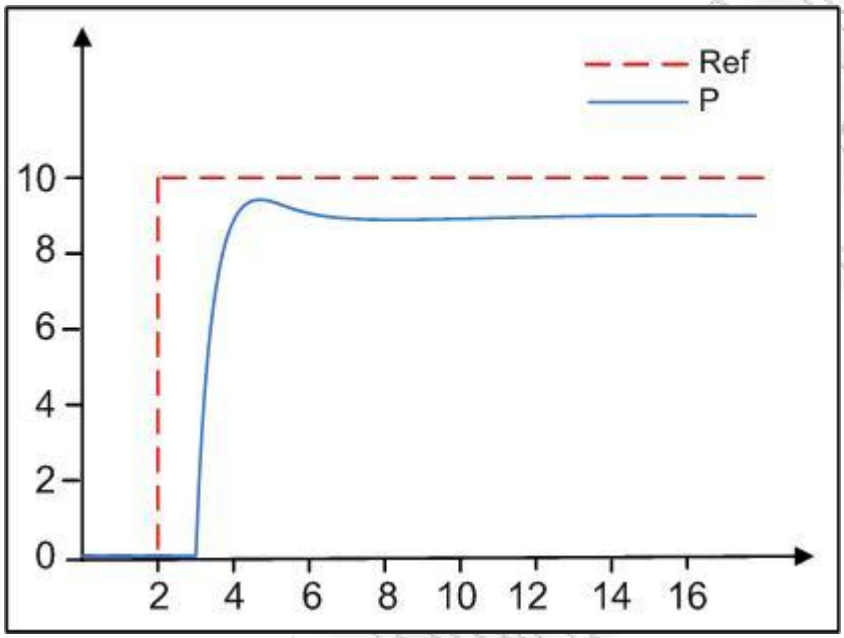
Η σχέση μεταξύ εισόδου u και σήματος σφάλματος e είναι:

$$u(t) = Kp\ e(t)$$

Η συνάρτηση μεταφοράς είναι:

$$G(s) = Kp = constant$$

Η βηματική απόκριση αναλογικού ελεγκτή φαίνεται στην παρακάτω εικόνα.



Εικόνα 20: βηματική απόκριση αναλογικού ελεγκτή

Ένα απλό παράδειγμα ενός ελεγκτή αναλογικού ελέγχου παρουσιάζεται στην παρακάτω Εικόνα 21.

Η ελεγχόμενη μεταβλητή είναι η στάθμη του υγρού στην δεξαμενή. Το φλοτέρ είναι το όργανο μέτρησης, η βαλβίδα είναι η χειριζόμενη μεταβλητή και ο μοχλός παρέχει την ενέργεια ελέγχου. Παρατηρούμε ότι υπάρχει μια διαφορετική θέση της βαλβίδας για κάθε στάθμη. Η επιθυμητή στάθμη είναι η h_0 και η θέση της βαλβίδας που ανταποκρίνεται στο h_0 είναι η v_0 (v_0 είναι η θέση της βαλβίδας όπου το σφάλμα είναι 0). Η θέση της βαλβίδας δίνεται από τη σχέση: $\frac{v - v_0}{a} = \frac{h_0 - h}{b}$

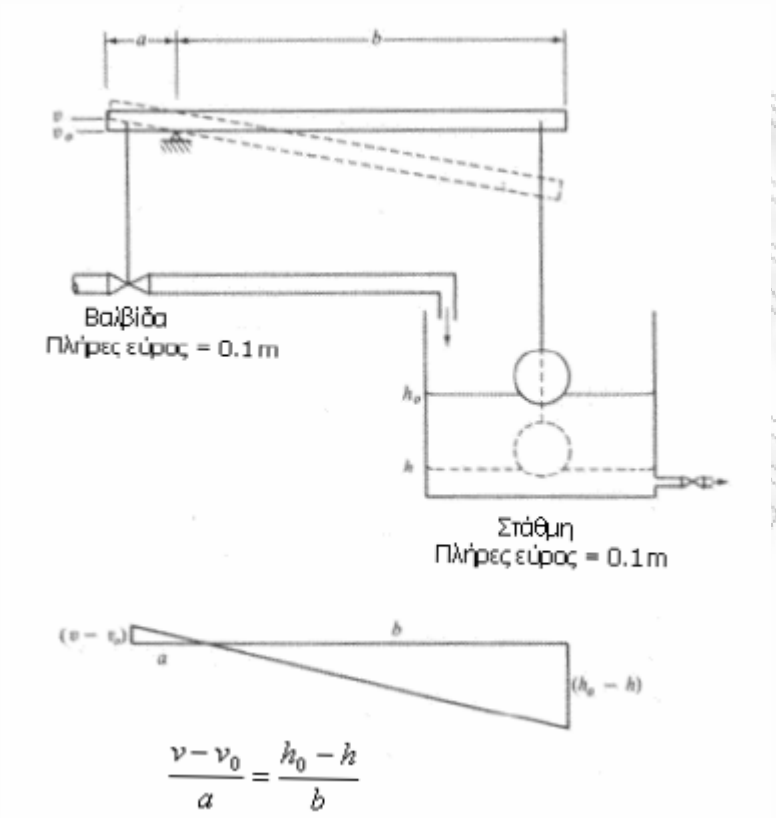
Σήμα σφάλματος = $e = h_0 - h$

$$v = \left(\frac{a}{b}\right)e + v_0$$

όπου: v = η θέση της βαλβίδας, (m)

v_0= η θέση της βαλβίδας με μηδενικό σφάλμα, (m)

e = το σήμα σφάλματος, (m)



Εικόνα 21: Παράδειγμα ενός ελεγκτή αναλογικού ελέγχου.

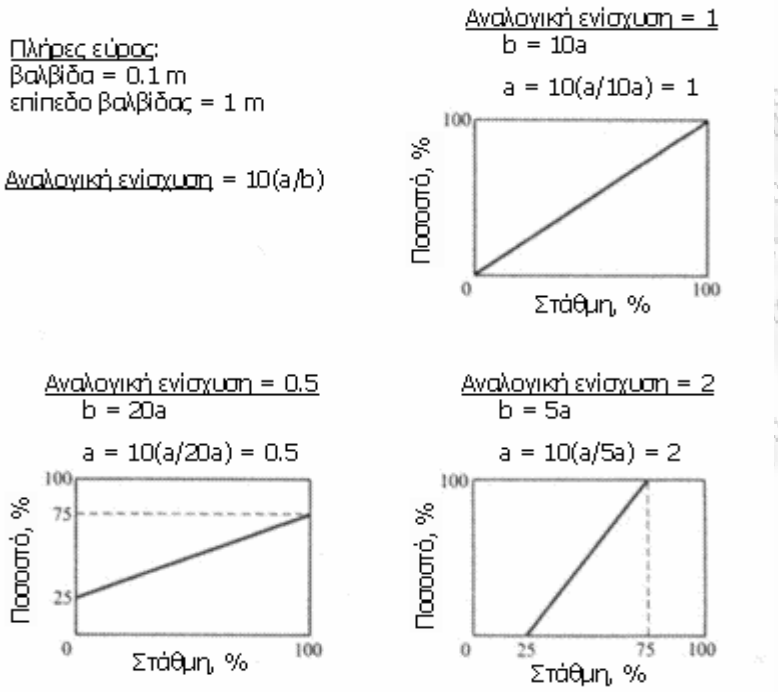
Η ενίσχυση(P) του ελεγκτή αναλογικού ελέγχου όπως φαίνεται στην εικόνα 20 είναι η αλλαγή στην θέση της βαλβίδας, διαιρούμενη με την αντίστοιχη αλλαγή στην στάθμη. Και οι δύο αυτές αλλαγές εκφράζονται ως ποσοστά του πλήρους εύρους.

Ποσοστό αλλαγής στην θέση της βαλβίδας $= \frac{100(v-v_0)}{0.1} = 1000(v-v_0)$

Ποσοστό αλλαγής στην στάθμη $= \frac{100(h_0-h)}{1} = 100(h_0-h)$

Ενίσχυση $P = \frac{1000(v-v_0)}{100(h_0-h)} = 10\left(\frac{v-v_0}{h_0-h}\right) = 10\left(\frac{a}{b}\right)$

Η Εικόνα22 περιλαμβάνει τις γραφικές παραστάσεις εισόδου/ εξόδου ελεγκτών αναλογικού ελέγχου με ενίσχυση 0.5, 1 και 2 αντίστοιχα.



Εικόνα22: Γραφικές παραστάσεις εισόδου/ εξόδου ελεγκτών αναλογικού ελέγχου με ενίσχυση 0.5, 1 και 2.

Γενικά μια αύξηση στην ενίσχυση μειώνει το μέγεθος του σφάλματος που χρειάζεται για να παράγει μια αλλαγή 100% στην θέση της βαλβίδας. Με άλλα λόγια μια υψηλή ενίσχυση απαιτεί ένα μικρό σφάλμα για να παράγει την αλλαγή στην βαλβίδα που είναι απαραίτητο για να ισορροπήσει η διαδικασία. Παρόλο που αυτό φαινομενικά σημαίνει ότι η ενίσχυση θα πρέπει να είναι όσο το δυνατόν υψηλότερη, δυστυχώς, αυξάνοντας την ενίσχυση αυξάνεται και η τάση της ταλάντωσης της ελεγχόμενης μεταβλητής. Επομένως είναι απαραίτητος ένας συμβιβασμός στον οποίο η ενίσχυση να είναι όσο το δυνατόν μεγαλύτερη χωρίς να παράγει ανεπιθύμητες ταλαντώσεις.

Ένα πρόβλημα με την μέθοδο αναλογικού ελέγχου είναι ότι δεν μπορεί να εξαλείψει ολοκληρωτικά το σφάλμα που προκύπτει από μια αλλαγή φορτίου. Ένα υπόλειμμα σφάλματος είναι πάντα αναγκαίο για να διατηρηθεί η βαλβίδα σε μια άλλη θέση εκτός της v_0. Αυτό είναι προφανές στην

εξίσωση $v = \frac{a}{b}e + v_0$ και είναι το ίδιο προφανές και στο απλό σύστημα που απεικονίζεται στην εικόνα. Αυτή η αλλαγή ή υπόλειμμα σφάλματος ονομάζεται αναλογική μετατόπιση. Το μέγεθος της μετατόπισης είναι ανάλογο του μεγέθους των αλλαγών φορτίου και αντιστρόφως ανάλογο της ενίσχυσης. Η αναλογική μέθοδος ελέγχου χρησιμοποιείται όταν η ενίσχυση μπορεί να γίνει αρκετά μεγάλη ώστε να μειώσει την αναλογική μετατόπιση σε ένα ανεκτό επίπεδο για την μέγιστη αναμενόμενη αλλαγή φορτίου.

Η αναλογική μέθοδος ελέγχου χρησιμοποιείται σε διαδικασίες με μικρή χωρητικότητα και γρήγορες αλλαγές φορτίου, όταν η ενίσχυση μπορεί να γίνει αρκετά μεγάλη ώστε να μειώσει την μετατόπιση σε ένα ανεκτό επίπεδο. Αυτό προϋποθέτει μια διαδικασία με χωρητικότητα που είναι πολύ μικρή για να επιτρέψει ένα σύστημα ελέγχου δύο θέσεων ή μια κινητή μέθοδο ελέγχου.

3.5.4 **Μέθοδος Ελέγχου Ολοκληρώματος, I**

Η μέθοδος ελέγχου ολοκληρώματος(integral control mode) αλλάζει την έξοδο του ελεγκτή κατά ένα ποσό ανάλογο του ολοκληρώματος του σήματος σφάλματος. Όσο υπάρχει σφάλμα, η μέθοδος ελέγχου ολοκληρώματος θα αλλάζει την έξοδο με ρυθμό ανάλογο του μεγέθους του σφάλματος.

Η σχέση μεταξύ εισόδου u και σήματος σφάλματος e είναι:

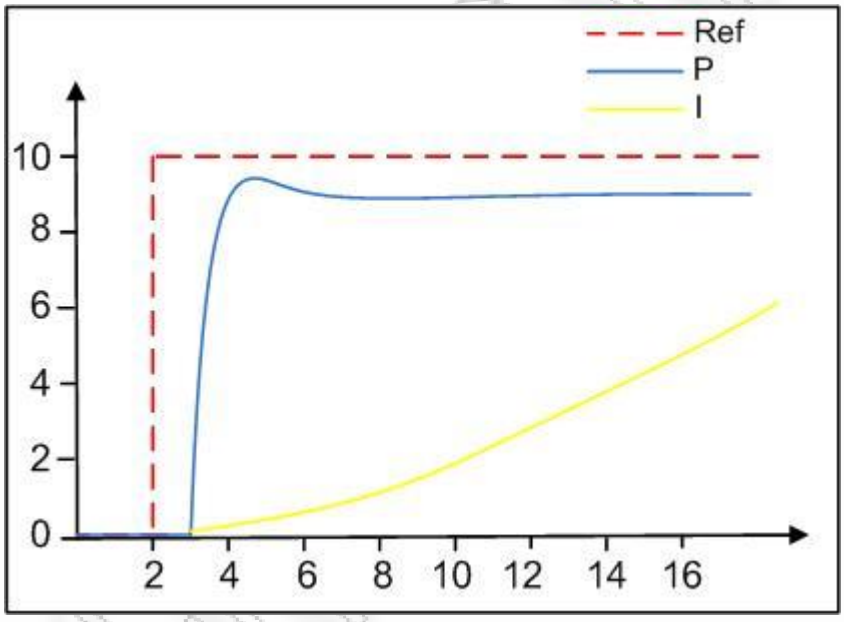
$$u(t) = K_i \int_0^t e(t)dt$$

Η συνάρτηση μεταφοράς είναι:

$$G(s) = \frac{K_i}{S}$$

όπου η Ki είναι προσαρμοζόμενη σταθερά

Η βηματική απόκριση ελεγκτή ολοκληρώματος φαίνεται στην παρακάτω εικόνα.



Εικόνα 23: Βηματική απόκριση ελεγκτή ολοκληρώματος

Διπλασιασμός τιμής σφάλματος e(t) σημαίνει διπλασιασμό ρυθμού μεταβολής (κλίσης) του u.

Η ολοκλήρωση που κάνει ο ελεγκτής και η ακόλουθη δράση του σήματος u στην έξοδο y συνεχίζεται μέχρι η τιμή της εξόδου να φθάσει πολύ κοντά στην τιμή αναφοράς r μόνιμης κατάστασης. Έτσι το σφάλμα μόνιμης κατάστασης οδηγείται στο μηδέν. Όμως ο I ελεγκτής μειώνει το βαθμό ευστάθειας του συστήματος, εφόσον αυξάνει η τάξη του όλου συστήματος.

3.5.5 **Αναλογική Και Ολοκληρωτική Μέθοδος Ελέγχου, PI**

Η μέθοδος ολοκληρώματος συνδυάζεται συχνά με την αναλογική μέθοδο για να παρέχει μια αυτόματη ενέργεια επαναφοράς που εξαλείφει την αναλογική μετατόπιση. Ο συνδυασμός αναφέρεται ως αναλογική και ολοκληρωτική μέθοδος ελέγχου (proportional plus integral control mode, PI). Η μέθοδος ολοκληρώματος παρέχει την ενέργεια επαναφοράς αλλάζοντας συνεχώς την έξοδο του ελεγκτή μέχρι το σφάλμα να μηδενιστεί.

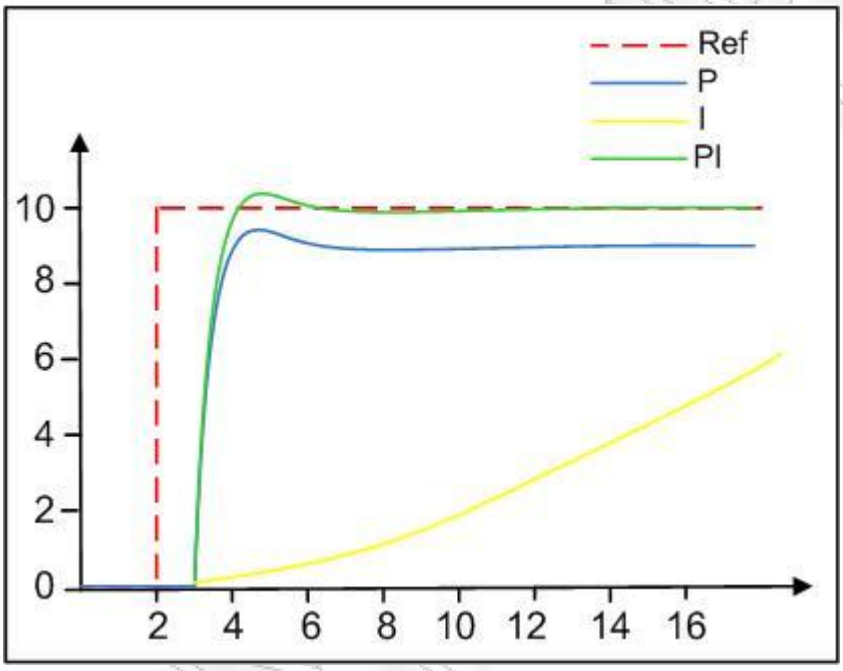
Η σχέση μεταξύ εισόδου u και σήματος σφάλματος e είναι:

$$u(t) = Kp\, e(t) + K_i \int_0^t e(t)\,dt$$

Η συνάρτηση μεταφοράς είναι:

$$G(s) = Kp + \frac{K_i}{S}$$

Η Εικόνα 24 απεικονίζει την απόκριση ενός ελεγκτή PI.



Εικόνα 24: Βηματική απόκριση ελεγκτή PI.

Η αναλογική μέθοδος παρέχει μια αλλαγή στην έξοδο του ελεγκτή που είναι ανάλογη του σήματος σφάλματος. Η μέθοδος ολοκληρώματος παρέχει μια επιπρόσθετη αλλαγή στην έξοδο που είναι ανάλογη του ολοκληρώματος του σήματος σφάλματος. Ο αντίστροφος του ρυθμού ενέργειας ολοκληρώματος(I) είναι ο χρόνος που χρειάζεται η μέθοδος ολοκληρώματος για να ταυτιστεί με την αλλαγή στην έξοδο που παράγει η αναλογική μέθοδος.

Ένα πρόβλημα με την μέθοδο ολοκληρώματος είναι ότι αυξάνει την τάση της ταλάντωσης της ελεγχόμενης μεταβλητής. Η ενίσχυση του ελεγκτή αναλογικής μεθόδου πρέπει να μειωθεί όταν συνδυάζεται με την μέθοδο ολοκληρώματος. Αυτό μειώνει την ικανότητα του ελεγκτή να ανταποκριθεί σε απότομες αλλαγές φορτίου. Αν η διαδικασία έχει μεγάλο νεκρό χρόνο υστέρησης, το σήμα σφάλματος δεν θα αντανακλά αμέσως το πραγματικό σφάλμα στην διαδικασία. Αυτή η υστέρηση συνήθως έχει ως αποτέλεσμα την υπερσύνδεση της μεθόδου ολοκληρώματος, δηλαδή η μέθοδος ολοκληρώματος συνεχίζει να αλλάζει την έξοδο του ελεγκτή ενώ το σφάλμα έχει μειωθεί στο μηδέν διότι ενεργεί σε ένα "παλιό" σήμα.

Η μέθοδος ελέγχου PI χρησιμοποιείται σε διαδικασίες με μεγάλες αλλαγές φορτίου όταν η αναλογική μέθοδος δεν είναι ικανή από μόνη της να ελαττώσει την μετατόπιση σε ένα ανεκτό επίπεδο. Η μέθοδος ολοκληρώματος παρέχει μια ενέργεια επαναφοράς που εξαλείφει την αναλογική μετατόπιση.

3.5.6 **Διαφορική Μέθοδος Ελέγχου, D**

Η διαφορική μέθοδος ελέγχου ή μέθοδος ελέγχου παραγώγου (derivative control mode) αλλάζει την έξοδο του ελεγκτή αναλογικά με το ρυθμό αλλαγής του σήματος σφάλματος. Αυτή η αλλαγή μπορεί να προκύψει από μια διαφορά στην μετρήσιμη μεταβλητή, στο σημείο ρύθμισης ή και στα δύο. Η διαφορική μέθοδος είναι μια προσπάθεια πρόβλεψης ενός σφάλματος παρατηρώντας πόσο γρήγορα αλλάζει το σφάλμα μέχρι κάποια στιγμή και χρησιμοποιεί τον ρυθμό αυτό για να παραχθεί μία ενέργεια που θα μειώσει το αναμενόμενο σφάλμα. Η διαφορική μέθοδος συμβάλει στην έξοδο του ελεγκτή μόνο όταν το σφάλμα αλλάζει. Για αυτόν τον λόγο η μέθοδος αυτή χρησιμοποιείται πάντα σε συνδυασμό με την αναλογική μέθοδο ή την μέθοδο PI.

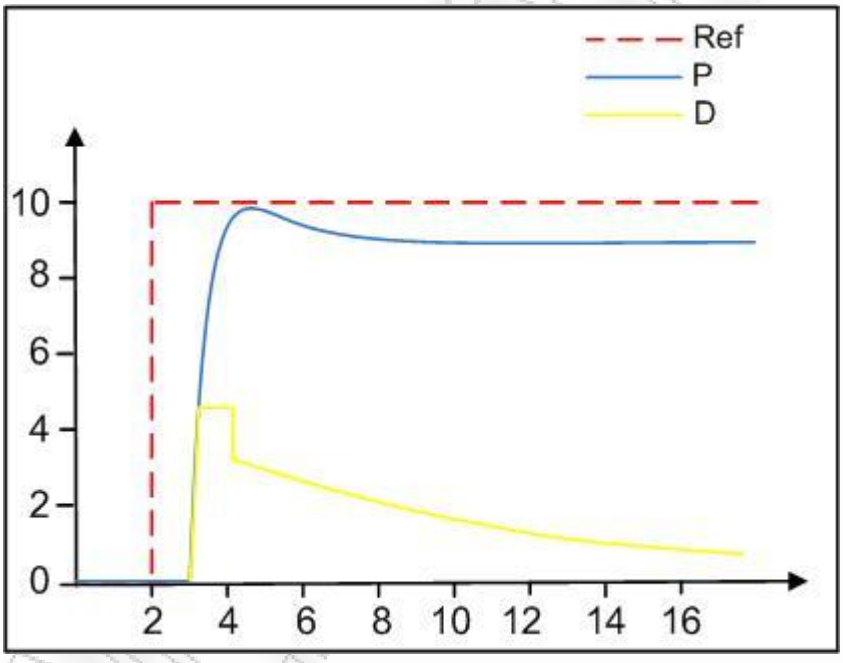
Η σχέση μεταξύ εισόδου u και σήματος σφάλματος e είναι:

$$u(t) = K_d \frac{de(t)}{dt}$$

Η συνάρτηση μεταφοράς είναι:

$$G(s) = K_d S$$

Η Εικόνα 25 απεικονίζει την απόκριση ενός ελεγκτή D.



Εικόνα 25: Βηματική απόκριση διαφορικού ελεγκτή D.

Σε κάθε στιγμή η έξοδος της διαφορικής μεθόδου ελέγχου είναι ανάλογη της κλίσης ή του ρυθμού αλλαγής του σήματος σφάλματος. Η βηματική απόκριση αναδεικνύει τον λόγο που η ιδανική διαφορική μέθοδος ελέγχου δεν χρησιμοποιείται ποτέ σε πραγματικούς ελεγκτές. Η καμπύλη σφάλματος έχει άπειρη κλίση όταν συμβαίνει η βηματική αλλαγή. Η ιδανική διαφορική μέθοδος ελέγχου πρέπει να ανταποκριθεί με μία άπειρη αλλαγή στην έξοδο του ελεγκτή. Στους πραγματικούς ελεγκτές η απόκριση της διαφορικής μεθόδου σε ταχέως μεταβλητά σήματα είναι περιορισμένη. Αυτό μειώνει κατά πολύ την ευαισθησία του ελεγκτή σε ανεπιθύμητες αιχμές θορύβου που παρουσιάζονται συχνά στην πραγματικότητα.

3.5.7 **Αναλογική Και Διαφορική Μέθοδος Ελέγχου, PD**

Η διαφορική μέθοδος χρησιμοποιείται κάποιες φορές με την αναλογική μέθοδο για να μειώσει την τάση για ταλαντώσεις και να επιτρέψει μεγαλύτερη αναλογική ενίσχυση. Ο συνδυασμός

της αναλογικής και της διαφορικής μεθόδου αναφέρεται ως μέθοδος ελέγχου PD. Η αναλογική μέθοδος παρέχει μια αλλαγή στην έξοδο του ελεγκτή που είναι ανάλογη του σήματος σφάλματος. Η διαφορική μέθοδος παρέχει μια επιπρόσθετη αλλαγή στην έξοδο του ελεγκτή που είναι ανάλογη του ρυθμού αλλαγής του σήματος σφάλματος. Η διαφορική μέθοδος προσμένει την μελλοντική τιμή του σφάλματος και αλλάζει κατάλληλα την έξοδο του ελεγκτή. Αυτή η ενέργεια προσμονής καθιστά την διαφορική μέθοδο χρήσιμη στον έλεγχο διαδικασιών με ξαφνικές αλλαγές φορτίου. γι' αυτόν τον λόγο η διαφορική μέθοδος χρησιμοποιείται συνήθως με την αναλογική μέθοδο ή την μέθοδο PI, όταν οι ξαφνικές αλλαγές φορτίου παράγουν υπερβολικά σφάλματα. Η διαφορική μέθοδος ελέγχου αντικρούει την αλλαγή της ελεγχόμενης μεταβλητής και αυτό βοηθά στην απόσβεση των ταλαντώσεων της ελεγχόμενη μεταβλητής.

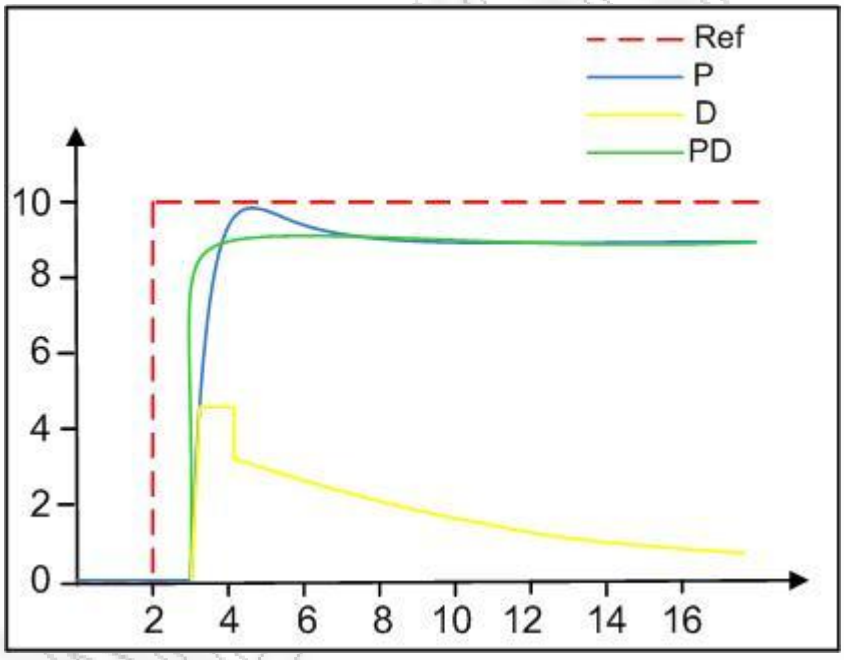
Η σχέση μεταξύ εισόδου u και σήματος σφάλματος e είναι:

$$u(t) = K_p e(t) + K_d \frac{de(t)}{dt}$$

Η συνάρτηση μεταφοράς είναι:

$$G(s) = Kp + K_d S$$

Η Εικόνα 26 απεικονίζει την απόκριση ενός ελεγκτή PD.



Εικόνα 26: Βηματική απόκριση ελεγκτή PD

Η μέθοδος PD χρησιμοποιείται σε διαδικασίες με ξαφνικές αλλαγές φορτίου όταν η αναλογική μέθοδος από μόνη της δεν είναι ικανή να περιορίσει το σφάλμα σε ένα ανεκτό επίπεδο. Η διαφορική μέθοδος παρέχει μια ενέργεια προσμονής που μειώνει το μέγιστο σφάλμα που προκαλείται από ξαφνικές αλλαγές φορτίου. Επίσης επιτρέπει μεγαλύτερη ενίσχυση η οποία βοηθά στην μείωση της αναλογικής μετατόπισης.

3.5.8 **Αναλογική – Ολοκληρωτική – Διαφορική Μέθοδος Ελέγχου, PID**

Ο σχεδιασμός σερβομηχανισμών θέσης με έλεγχο PI δεν είναι απολύτως ικανοποιητικός λόγω των δυσκολιών που συναντιούνται όταν η απόσβεση c είναι μικρή. Αυτό το πρόβλημα μπορεί να λυθεί με τη χρησιμοποίηση του πλήρους PID κανόνα ελέγχου.

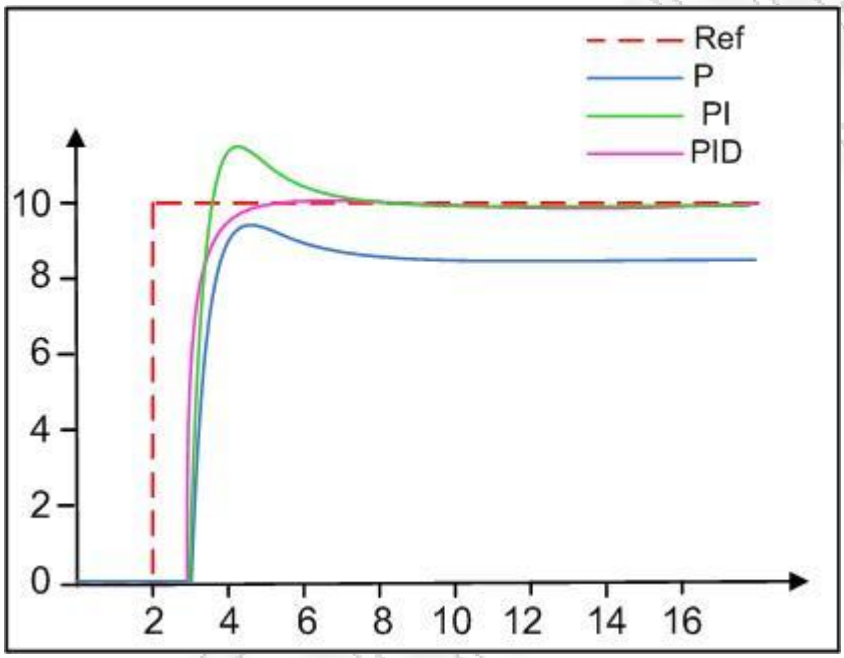
Η σχέση μεταξύ εισόδου u και σήματος σφάλματος e είναι:

$$u(t) = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt}$$

Η συνάρτηση μεταφοράς είναι:

$$G(s) = Kp + \frac{K_i}{S} + K_d S$$

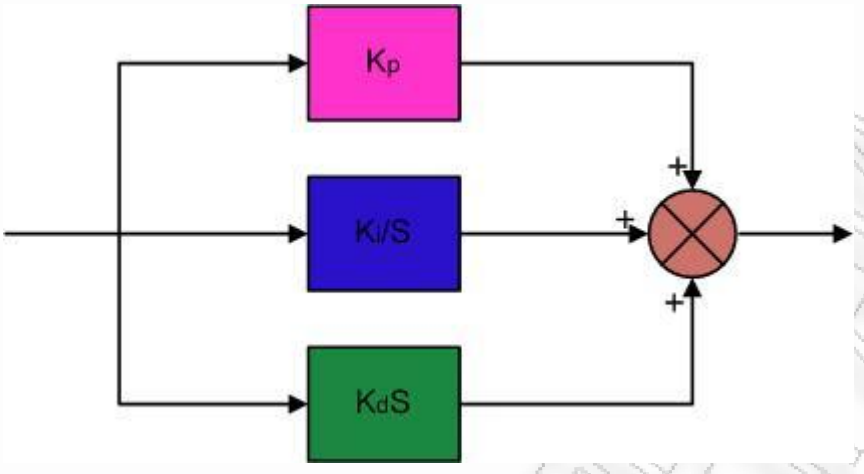
Η Εικόνα 27 απεικονίζει την απόκριση ενός ελεγκτή PID.



Εικόνα 27: Βηματική απόκριση ελεγκτή PID

Η παρουσία του K_D χαλαρώνει κάπως την απαίτηση να είναι αρκετά μεγάλο το K_P ώστε να επιτευχθεί η σταθερότητα. Τα σφάλματα σταθερής κατάστασης είναι μηδέν και η μεταβατική απόκριση μπορεί να βελτιωθεί επειδή τρεις από τους συντελεστές της χαρακτηριστικής εξίσωσης μπορούν να επιλεχτούν.

Όπως φαίνεται και πιο πάνω η μέθοδος ελέγχου PID είναι ένας συνδυασμός των μεθόδων P,I και D. Ένας ελεγκτής PID αναφέρεται και ως ελεγκτής τριών όρων. Η μέθοδος ολοκληρώματος χρησιμοποιείται για να εξαλείψει την αναλογική μετατόπιση που προκαλείται από μεγάλες αλλαγές φορτίου. Η διαφορική μέθοδος μειώνει την τάση για ταλαντώσεις και παρέχει μια ενέργεια ελέγχου που αναμένει αλλαγές στο σήμα σφάλματος. Η διαφορική μέθοδος είναι πολύ χρήσιμη όταν η διαδικασία έχει ξαφνικές αλλαγές φορτίου.



Εικόνα 28: Μπλοκ διάγραμμα ενός ελεγκτή PID.

3.6 Ψηφιακοί Ελεγκτές

Οι ψηφιακοί ελεγκτές(digital controllers) είναι βασισμένοι σε μικροεπεξεργαστές και χρησιμοποιούνται ευρέως σε βιομηχανικά συστήματα ελέγχου. Υπάρχουν πολλοί λόγοι για την μεγάλη απήχηση των ψηφιακών ελεγκτών. Η ισχύς των μικροεπεξεργαστών παρέχει εξελιγμένες δυνατότητες όπως προσαρμόσιμη αυτορύθμιση, έλεγχος πολλών μεταβλητών και έμπειρα συστήματα. Η ικανότητα του μικροεπεξεργαστή να επικοινωνεί μέσω ενός διαύλου ή ενός τοπικού δικτύου είναι ένας ακόμη λόγος για την ευρεία απήχηση του ψηφιακού ελεγκτή. Οι ψηφιακοί ελεγκτές που χρησιμοποιούνται για τον έλεγχο κλειστού βρόγχου υλοποιούν τις μεθόδους ελέγχου P,PI, PD ή PID.

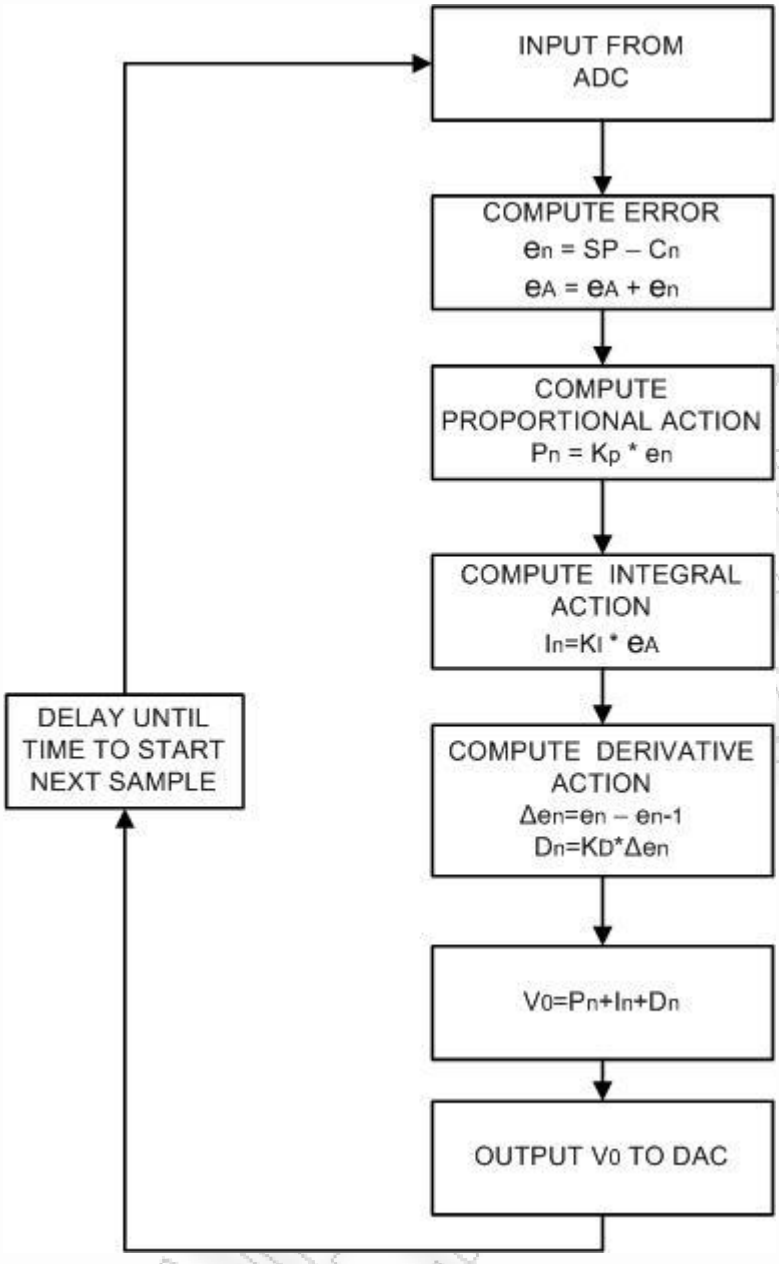
Ένας ψηφιακός ελεγκτής μετρά την ελεγχόμενη μεταβλητή σε συγκεκριμένους χρόνους, που χωρίζονται από ένα χρονικό διάστημα που ονομάζεται χρόνος δειγματοληψίας(sampling time), Δt. Κάθε δείγμα(ή μέτρηση) της ελεγχόμενης μεταβλητής μετατρέπεται σε έναν δυαδικό αριθμό έτσι ώστε να του επιτραπεί η είσοδός του σε έναν ψηφιακό υπολογιστή ή μικροϋπολογιστή. Ο υπολογιστής αφαιρεί κάθε δείγμα της μετρήσιμης μεταβλητής από την επιθυμητή τιμή για να υπολογίσει ένα σύνολο από δείγματα σφάλματος.

$e_1 = sp - c_{m1}$ = πρώτο δείγμα σφάλματος
$e_2 = sp - c_{m2}$ = δεύτερο δείγμα σφάλματος
$e_3 = sp - c_{m3}$ = τρίτο δείγμα σφάλματος

$e_n = sp - c_{mn}$ = n-οστό δείγμα σφάλματος

Μετά τον υπολογισμό κάθε δείγματος σφάλματος, ένας ψηφιακός ελεγκτής PID ακολουθεί μια διαδικασία που ονομάζεται αλγόριθμος PID για να υπολογίσει την έξοδο του ελεγκτή βασιζόμενος στα δείγματα σφάλματος e_1, e_2, e_3, ..., e_n.

Ο αλγόριθμος ενός ελεγκτή PID φαίνεται στο παρακάτω διάγραμμα ροής.



Εικόνα 29: Διάγραμμα ροής ενός αλγόριθμου PID.

4 ΚΕΦΑΛΑΙΟ 4

Το Περιβάλλον Ανάπτυξης Εφαρμογών AVR Studio 4

4.1 Εισαγωγή

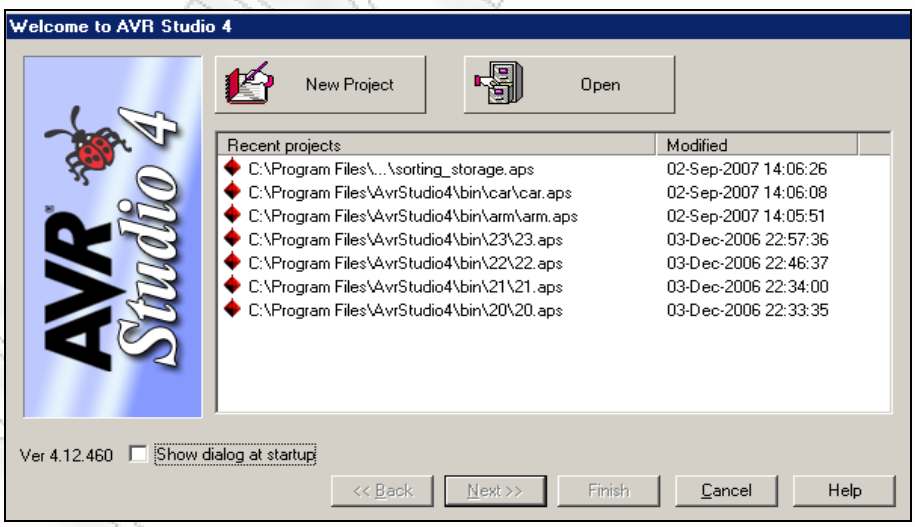
Για τον προγραμματισμό των μικροελεγκτών AVR της ATMEL υπάρχουν διάφοροι compilers στην αγορά και ο προγραμματισμός τους μπορεί να γίνει εκτός από την γλώσσα assembly και σε γλώσσες υψηλού επιπέδου όπως είναι η C++ και η Visual basic.

Για την ανάπτυξη του λογισμικού μέρους του συστήματος χρησιμοποιήθηκε το ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών (Integrated Development Environment) AVR Studio 4 version 4.14 της εταιρείας Atmel. Ακολούθως, πρώτα γίνεται μια περιγραφή του περιβάλλοντος ανάπτυξης, μετά περιγράφονται ο συμβολομεταφραστής και ο προσομοιωτής και τέλος δίνεται μια περιγραφή του τρόπου σύνδεσης του περιβάλλοντος ανάπτυξης με την αναπτυξιακή κάρτα STK500 και τον προγραμματιστή εντός του συστήματος AVRISP.

4.2 Περιγραφή Του Περιβάλλοντος Ανάπτυξης

Μέσω της εφαρμογής αυτής, ο χρήστης έχει τη δυνατότητα συγγραφής και αποσφαλμάτωσης του προγράμματος, αλλά και συμβολομετάφρασης ή μεταγλώττισης αυτού μέσω του συμβολομεταφραστή AVR (AVR Assembler) ή του μεταγλωττιστή GCC AVR (AVR GCC Compiler) αντίστοιχα. Στη συνέχεια το πρόγραμμα μπορεί να κατέβει στη μνήμη του μικροελεγκτή με αποτέλεσμα τον προγραμματισμό αυτού, έτσι ώστε να επιτελέσει την επιθυμητή λειτουργία.

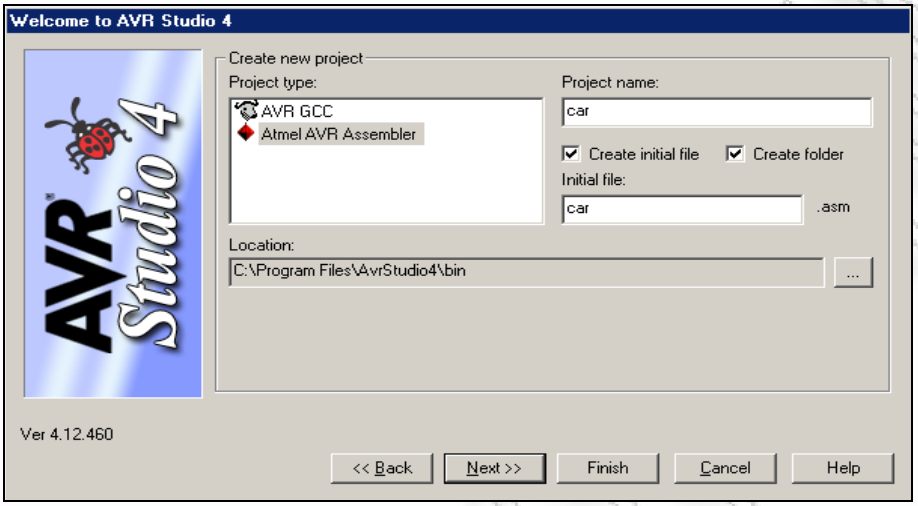
Με την εκκίνηση του AVR Studio, ζητείται από το χρήστη να επιλέξει εάν επιθυμεί να δημιουργήσει ένα νέο project ή να ανοίξει ένα ήδη υπάρχον. Τα παραπάνω φαίνονται στην επόμενη εικόνα.



Εικόνα 30: Παράθυρο διάλογου κατά την εκκίνηση του AVR Studio.

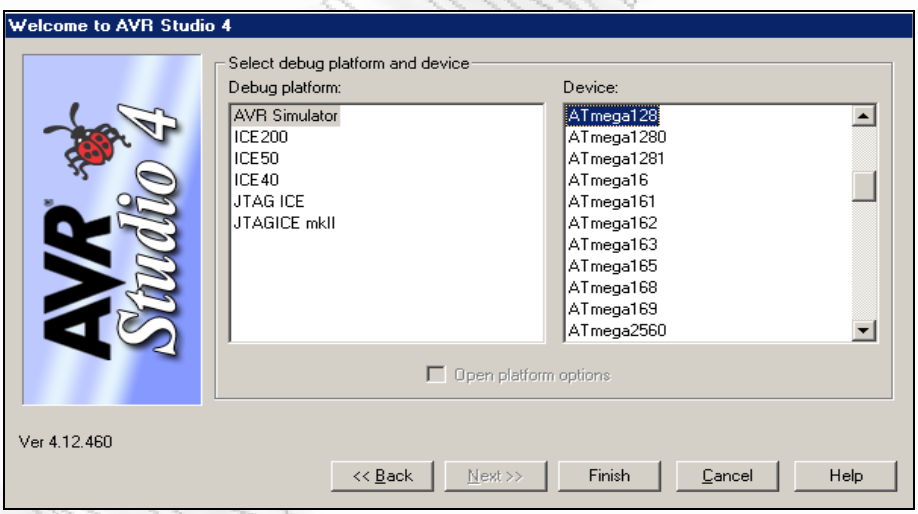
Στην πρώτη περίπτωση, η οποία είναι και η πιο γενική, ο χρήστης οδηγείται σε ένα δεύτερο παράθυρο διαλόγου όπου καλείται να επιλέξει ανάμεσα στον AVR GCC Compiler και στον AVR Assembler. Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε ο AVR Assembler.

Επίσης, θα πρέπει να δώσει και το όνομα του project, ενώ δημιουργείται και ένα αρχείο με το ίδιο όνομα με κατάληξη .c ή .asm αντίστοιχα. Τέλος, μπορεί να καθορίσει και τη θέση του project στο δίσκο. Τα παραπάνω φαίνονται στην επόμενη εικόνα.



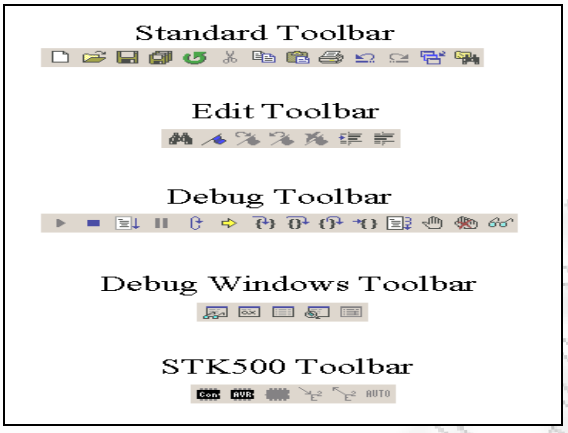
Εικόνα 31: Παράθυρο διαλόγου για τη δημιουργία νέου project.

Στο τρίτο και τελευταίο παράθυρο διαλόγου, ο χρήστης έχει να επιλέξει την πλατφόρμα αποσφαλμάτωσης και τη διάταξη. Στη συγκεκριμένη περίπτωση χρησιμοποιήθηκε ο AVR Simulator και η CPU ATmega8.



Εικόνα 32: Παράθυρο διαλόγου για την επιλογή πλατφόρμας και διάταξης.

Ολοκληρώνοντας την προκαταρκτική αυτή διαδικασία, ο χρήστης οδηγείται στο κύριο τμήμα της εφαρμογής όπου μπορεί, πλέον, να ξεκινήσει τη διαδικασία ανάπτυξης του προγράμματος. Κατά τη διαδικασία αυτήν, έχει στη διάθεσή του μια σειρά εργαλείων, η μορφή των οποίων φαίνονται στην επόμενη εικόνα.



Εικόνα 33: Κατηγορίες εργαλείων για την ανάπτυξη του προγράμματος.

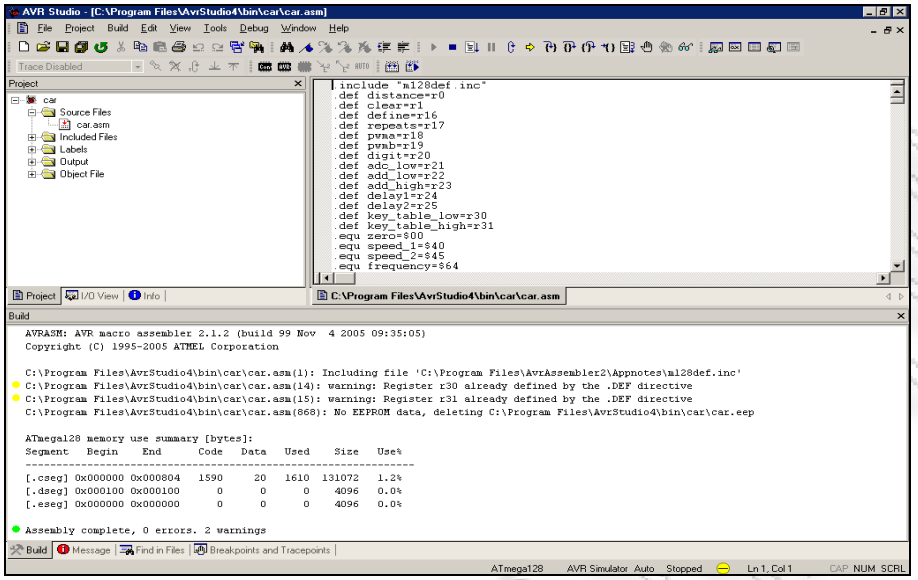
Οι δύο πρώτες κατηγορίες εργαλείων περιλαμβάνουν τα βασικά εργαλεία των Windows, τα οποία συναντά κανείς σε όλες τις εφαρμογές που τρέχουν στο περιβάλλον αυτό.

Η τρίτη κατηγορία εργαλείων περιλαμβάνει τα βασικά εργαλεία για την αποσφαλμάτωση του προγράμματος. Μετά την αποσφαλμάτωση, το πηγαίο αρχείο (Source File) μετατρέπεται σε αντικείμενο αρχείο (Object File) και ο χρήστης μπορεί είτε να ελέγξει τη λειτουργία του μέσω του προσομοιωτή AVR (AVR Simulator) ή του εξομοιωτή AVR (AVR In-Circuit Emulator) είτε να το κατεβάσει στη μνήμη του μικροελεγκτή. Μέσω των διαφόρων εργαλείων της κατηγορίας αυτής, υπάρχει η δυνατότητα εκτέλεσης του προγράμματος με διαφόρους τρόπους, αλλά και η χρήση σημείων διακοπής της διαδικασίας (Breakpoints).

Η τέταρτη κατηγορία εργαλείων περιλαμβάνει τέσσερα παράθυρα, τα οποία μπορεί να παρακολουθεί ο χρήστης κατά την εκτέλεση του προγράμματος: το παράθυρο παρακολούθησης (Watch Window), το παράθυρο καταχωρητή (Register Window), το παράθυρο μνήμης (Memory Window) και το παράθυρο αποσυναρμολόγησης (Disassembly Window). Τα παράθυρα αυτά είναι ιδιαίτερα σημαντικά, δεδομένου ότι ο χρήστης μπορεί να ελέγχει το περιεχόμενο των μνημών και των καταχωρητών κατά την εκτέλεση του προγράμματος.

Η πέμπτη κατηγορία εργαλείων περιλαμβάνει τα εργαλεία για σύνδεση με την αναπτυξιακή κάρτα STK500. Στην περίπτωση αυτήν, ο χρήστης μπορεί να συνδεθεί με την κάρτα, έτσι ώστε να κατεβάσει το πρόγραμμα στη μνήμη του μικροελεγκτή. Σε διαφορετική περίπτωση, μπορεί να χρησιμοποιήσει τον προσομοιωτή ή τον εξομοιωτή του AVR Studio.

Αναφορικά με το κύριο μέρος της εφαρμογής, αυτό χωρίζεται σε τρία τμήματα: το παράθυρο σύνταξης του προγράμματος (Editor Window), το παράθυρο των όψεων project, εισόδου/ εξόδου και πληροφορίας (Project View, I/O View και Info View) και το παράθυρο εξόδου (Output View) των όψεων κατασκευής, μηνύματος, εύρεσης σε αρχεία και σημείων διακοπής της διαδικασίας (Build View, Message View, Find In Files View και Breakpoints View). Η μορφή των τμημάτων αυτών φαίνονται στην επόμενη εικόνα.



Εικόνα 34: Τμήματα του κυρίου μέρους της εφαρμογής.

Το πρώτο τμήμα βρίσκεται στη δεξιά πλευρά και εκεί ο χρήστης μπορεί να γράψει το πρόγραμμα.

Το δεύτερο τμήμα βρίσκεται στην αριστερή πλευρά και αποτελείται από τρεις όψεις. Η πρώτη όψη περιλαμβάνει τα πηγαία αρχεία (Source Files), τα συμπεριλαμβανόμενα αρχεία (Included Files), τις ετικέτες (Labels), τα αρχεία εξόδου (Output Files) και το αντικείμενο αρχείο (Object File). Η δεύτερη όψη περιλαμβάνει τα περιεχόμενα των καταχωρητών γενικής χρήσης, των καταχωρητών ειδικής χρήσης, των θυρών εισόδου/ εξόδου και της στοίβας. Η τρίτη όψη περιλαμβάνει ορισμένες πληροφορίες σχετικά με τις διευθύνσεις των διανυσμάτων διακοπής και των καταχωρητών εισόδου/ εξόδου όπως, επίσης, τη διάταξη των ακροδεκτών όσον αφορά στο μικροελεγκτή που έχει επιλεγεί να χρησιμοποιηθεί.

Το τρίτο τμήμα βρίσκεται στην κάτω πλευρά και αποτελείται από τέσσερις όψεις. Η πρώτη όψη περιλαμβάνει τα μηνύματα και τις προειδοποιήσεις κατά τη διαδικασία αποσφαλμάτωσης του προγράμματος. Η δεύτερη όψη περιλαμβάνει τα μηνύματα, τις προειδοποιήσεις, τα λάθη και τις πληροφορίες της αποσφαλμάτωσης. Η τρίτη όψη περιλαμβάνει την εύρεση του αρχείου, το οποίο σχετίζεται με μια συγκεκριμένη γραμμή του προγράμματος. Η τέταρτη όψη περιλαμβάνει τα σημεία διακοπής της διαδικασίας, τα οποία χρησιμοποιήθηκαν από το πρόγραμμα.

Για επιπλέον πληροφορίες σχετικά με το ολοκληρωμένο περιβάλλον ανάπτυξης εφαρμογών AVR Studio 4, ο αναγνώστης παραπέμπεται στο εγχειρίδιο του κατασκευαστή.

4.3 Περιγραφή Του Συμβολομεταφραστή

Για τη συμβολομετάφραση του προγράμματος χρησιμοποιήθηκε ο συμβολομεταφραστής AVR Assembler 2, ο οποίος αποτελεί εξέλιξη του AVR Assembler. Ο συμβολομεταφραστής αυτός μετατρέπει τον πηγαίο κώδικα (Source Code) σε αντικείμενο κώδικα (Object Code). Στη συνέχεια ο κώδικας αυτός μπορεί να χρησιμοποιηθεί ως είσοδος είτε για τον προσομοιωτή AVR Simulator είτε για τον εξομοιωτή AVR In-Circuit Emulator. Επίσης, δημιουργείται και ένας δεκαεξαδικός κώδικας (Hex Code), ο οποίος μπορεί να προγραμματίσει τη μνήμη του μικροελεγκτή.

Ο συμβολομεταφραστής αυτός υποστηρίζει:

- Όλο το σύνολο εντολών των μικροελεγκτών AVR.
- Ένα μεγάλο αριθμό ψευδοεντολών και μακροεντολών.

- Ένα μεγάλο αριθμό σταθερών εκφράσεων, τελεστέων, τελεστών και συναρτήσεων.

Για επιπλέον πληροφορίες σχετικά με τους συμβολομεταφραστές AVR Assembler 2 και AVR Assembler, ο αναγνώστης παραπέμπεται στο εγχειρίδιο του κατασκευαστή.

4.4 Περιγραφή Του Προσομοιωτή

Για την προσομοίωση του μικροελεγκτή AVR χρησιμοποιήθηκε ο προσομοιωτής AVR Simulator. Ο προσομοιωτής αυτός προσομοιώνει τη CPU του μικροελεγκτή, συμπεριλαμβανομένων όλων των εντολών, των διακοπών και των περισσοτέρων υπομονάδων εισόδου/ εξόδου. Μέσω της επιλογής AVR Simulator Options, ο χρήστης μπορεί να κάνει τις απαραίτητες ρυθμίσεις όσον αφορά στη λειτουργία του προσομοιωτή.

Στη συνέχεια αναφέρονται οι υπομονάδες εκείνες του μικροελεγκτή AVR, οι οποίες υποστηρίζονται από τον προσομοιωτή αυτόν:

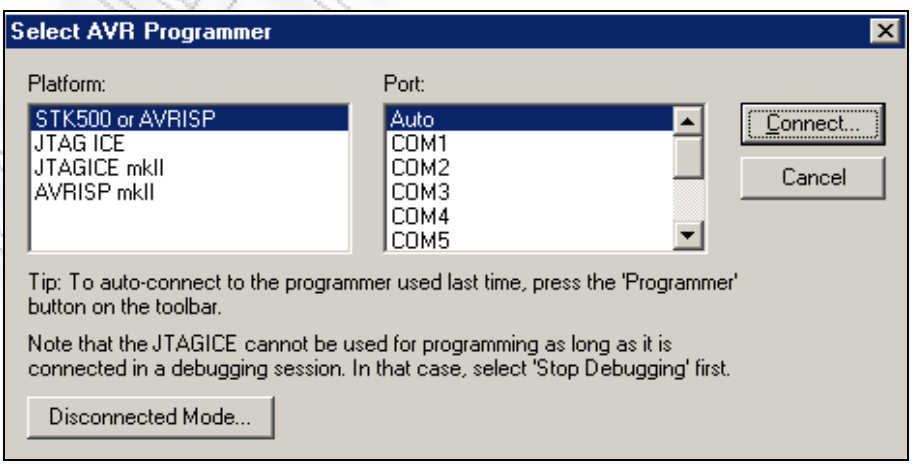
- Όλες οι θύρες εισόδου/ εξόδου (I/O Ports).
- Όλες οι εξωτερικές διακοπές (External Interrupts).
- Όλοι οι χρονιστές / μετρητές των 8-bit (8-bit Timers/ Counters).
- Όλοι οι χρονιστές/ μετρητές των 16-bit (16-bit Timers/ Counters).
- Η σειριακή διασύνδεση περιφερειακών (Serial Peripheral Interface).
- Ο χρονιστής επιτήρησης (Watchdog Timer).

Για επιπλέον πληροφορίες σχετικά με τον προσομοιωτή AVR Simulator, ο αναγνώστης παραπέμπεται στο εγχειρίδιο του κατασκευαστή.

4.5 Σύνδεση Με Την Αναπτυξιακή Κάρτα STK500.

Στην παρούσα εργασία, τα προγράμματα που αναπτύχθηκαν για τον επενεργητή και τον ελεγκτή PID κατέβηκαν στη μνήμη των μικροελεγκτών AVR από την αναπτυξιακή κάρτα STK500. ο μικροελεγκτής του επενεργητή προγραμματίστηκε επάνω στην αναπτυξιακή κάρτα STK500 ενώ ο μικροελεγκτής του PID μέσω του προγραμματιστή εντός του συστήματος AVRISP.

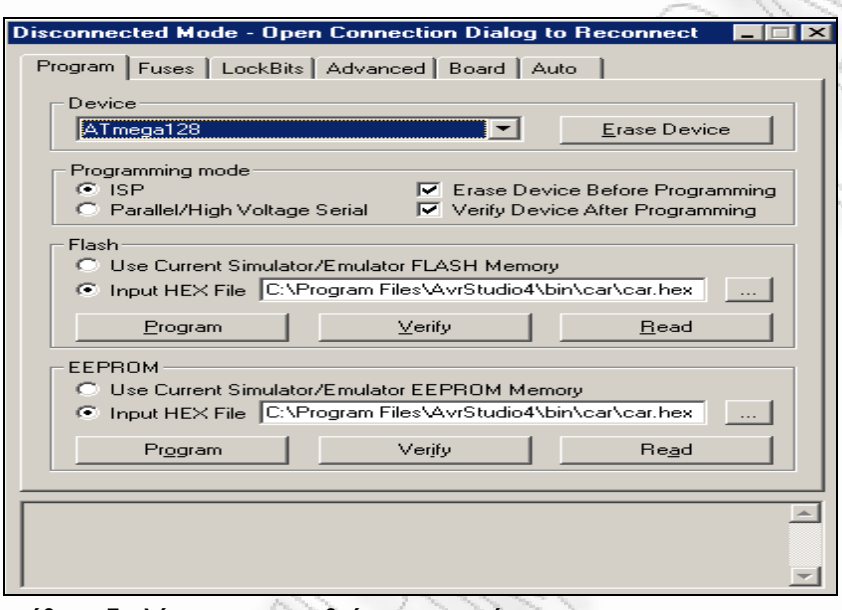
Σε κάθε περίπτωση, η σύνδεση με τον αντίστοιχο μικροελεγκτή επιτυγχάνεται μέσω της πέμπτης κατηγορίας εργαλείων (STK500 Toolbar), με χρήση των εργαλείων CON ή AVR. Πατώντας την πρώτη επιλογή, ο χρήστης οδηγείται στο παράθυρο διαλόγου που φαίνεται στο επόμενο σχήμα. Πατώντας τη δεύτερη επιλογή, το παράθυρο αυτό παραλείπεται και η σύνδεση θα γίνει με χρήση των ίδιων ρυθμίσεων, οι οποίες έγιναν κατά την προηγούμενη σύνδεση.



Εικόνα 35: Παράθυρο διαλόγου για την επιλογή του προγραμματιστή.

Μετά τη σύνδεση (Connect), ο χρήστης έχει τη δυνατότητα να κάνει μια σειρά ρυθμίσεων, οι οποίες χωρίζονται σε έξι κατηγορίες: ρυθμίσεις προγράμματος (Program Settings), ρυθμίσεις των Fuse Bits (Fuse Bits Settings), ρυθμίσεις των Lock Bits (Lock Bits Settings), προχωρημένες ρυθμίσεις (Advanced Settings), ρυθμίσεις πλακέτας (Board Settings) και αυτόματες ρυθμίσεις (Auto Settings).

Το παράθυρο διαλόγου για την πρώτη κατηγορία ρυθμίσεων φαίνονται στην επόμενη εικόνα.



Εικόνα 36: Παράθυρο διαλόγου για τις ρυθμίσεις προγράμματος.

Το παράθυρο αυτό χωρίζεται σε τέσσερα τμήματα: το τμήμα της διάταξης (Device), το τμήμα της μεθόδου προγραμματισμού (Programming Mode), το τμήμα της μνήμης προγράμματος ταχείας αποθήκευσης (Flash) και το τμήμα της μνήμης EEPROM.

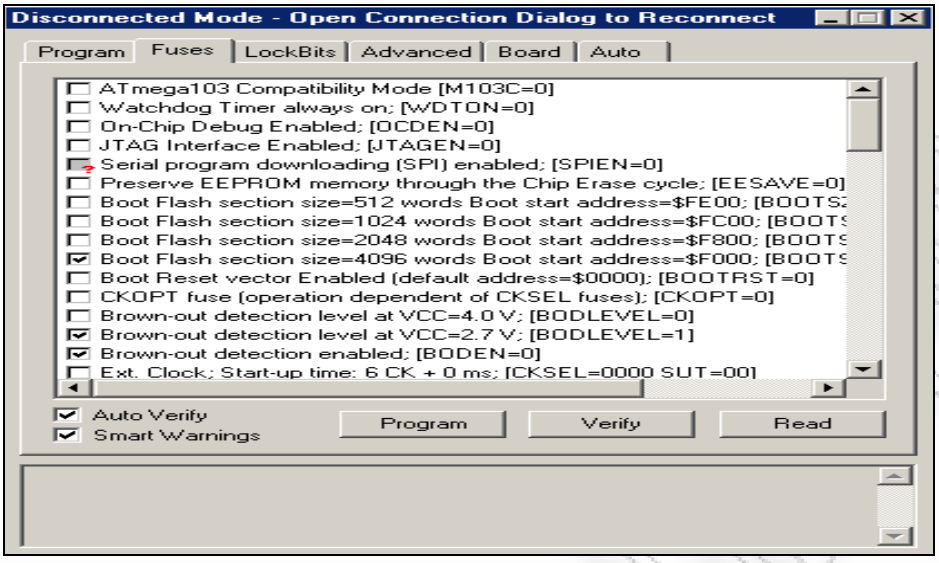
Στο πρώτο τμήμα, ο χρήστης καλείται να επιλέξει τη διάταξη που πρόκειται να χρησιμοποιήσει. Επίσης, μπορεί να καθαρίσει, τόσο τη Flash, όσο και την EEPROM μνήμη του μικροελεγκτή.

Στο δεύτερο τμήμα, ζητείται από το χρήστη να καθορίσει τη μέθοδο προγραμματισμού που θα χρησιμοποιηθεί. Επίσης, μπορεί να καθαρίσει τη Flash μνήμη του μικροελεγκτή, πριν τον προγραμματισμό αυτού και να επιβεβαιώσει το περιεχόμενο, τόσο της Flash, όσο και της EEPROM μνήμης του μικροελεγκτή, μετά τον προγραμματισμό αυτού.

Στο τρίτο τμήμα, ο χρήστης ορίζει το αρχείο που πρόκειται να προγραμματίσει τη Flash μνήμη του μικροελεγκτή. Αυτό μπορεί να γίνει είτε με χρήση του τρέχοντος αρχείου που βρίσκεται στη Flash μνήμη του προσομοιωτή ή του εξομοιωτή του AVR Studio είτε με τον καθορισμό της θέσης του αρχείου, το οποίο θα πρέπει να είναι τύπου “Hex”. Στη συνέχεια ο χρήστης μπορεί να προγραμματίσει το μικροελεγκτή (Program), να επιβεβαιώσει τον προγραμματισμό αυτού (Verify) και να διαβάσει το πρόγραμμα (Read).

Στο τέταρτο τμήμα, ο χρήστης ορίζει το αρχείο που πρόκειται να προγραμματίσει την EEPROM μνήμη του μικροελεγκτή. Αυτό μπορεί να γίνει είτε με χρήση του τρέχοντος αρχείου που βρίσκεται στην EEPROM μνήμη του προσομοιωτή ή του εξομοιωτή του AVR Studio είτε με τον καθορισμό της θέσης του αρχείου, το οποίο θα πρέπει να είναι τύπου “Hex”. Στη συνέχεια ο χρήστης μπορεί να προγραμματίσει το μικροελεγκτή (Program), να επιβεβαιώσει τον προγραμματισμό αυτού (Verify) και να διαβάσει το πρόγραμμα (Read).

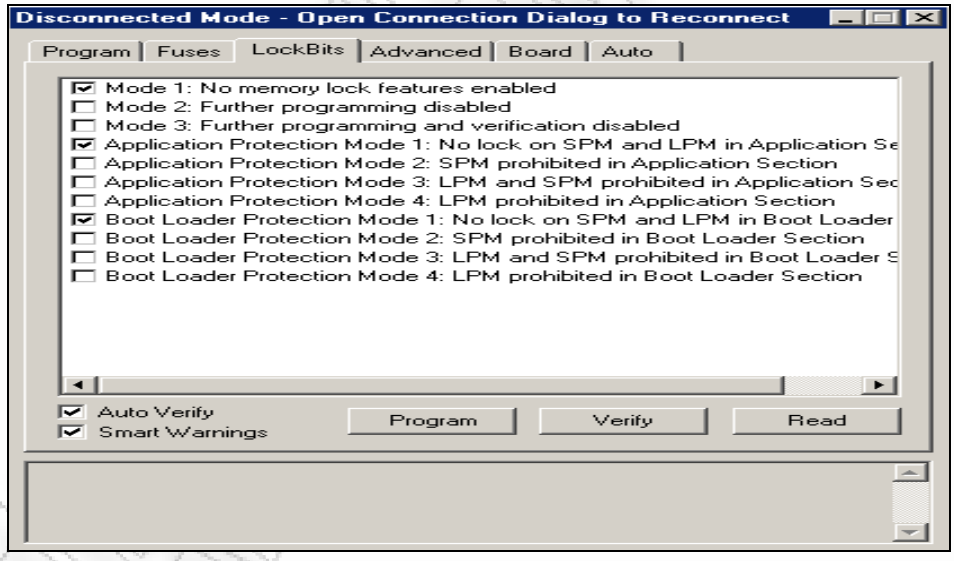
Το παράθυρο διαλόγου για τη δεύτερη κατηγορία ρυθμίσεων φαίνονται στην επόμενη εικόνα.



Εικόνα 37: Παράθυρο διαλόγου για τις ρυθμίσεις των Fuse Bits.

Στο παράθυρο αυτό, ο χρήστης προγραμματίζει τα κατάλληλα Fuse Bits. Επιλέγοντας ένα συγκεκριμένο Fuse Bit, αυτό ενεργοποιείται, δηλαδή, παίρνει την τιμή μηδέν στη φυσική του διεύθυνση. Στη συνέχεια ο χρήστης μπορεί να προγραμματίσει το Fuse Bit (Program), να επιβεβαιώσει τον προγραμματισμό αυτού (Verify) και να διαβάσει την τιμή του (Read).

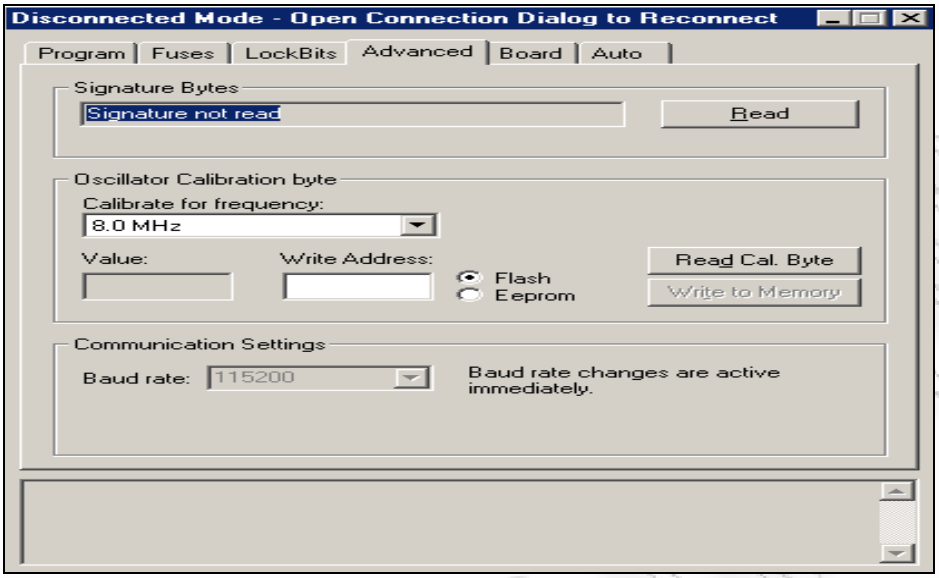
Το παράθυρο διαλόγου για την τρίτη κατηγορία ρυθμίσεων φαίνονται στην επόμενη εικόνα.



Εικόνα 38: Παράθυρο διαλόγου για τις ρυθμίσεις των Lock Bits.

Στο παράθυρο αυτό, ο χρήστης προγραμματίζει τα κατάλληλα Lock Bits. Επιλέγοντας ένα συγκεκριμένο Lock Bit, αυτό ενεργοποιείται, δηλαδή, παίρνει την τιμή μηδέν στη φυσική του διεύθυνση. Στη συνέχεια ο χρήστης μπορεί να προγραμματίσει το Lock bit (Program), να επιβεβαιώσει τον προγραμματισμό αυτού (Verify) και να διαβάσει την τιμή του (Read).

Το παράθυρο διαλόγου για την τέταρτη κατηγορία ρυθμίσεων φαίνεται στην επόμενη εικόνα.



Εικόνα 39: Παράθυρο διαλόγου για τις προχωρημένες ρυθμίσεις.

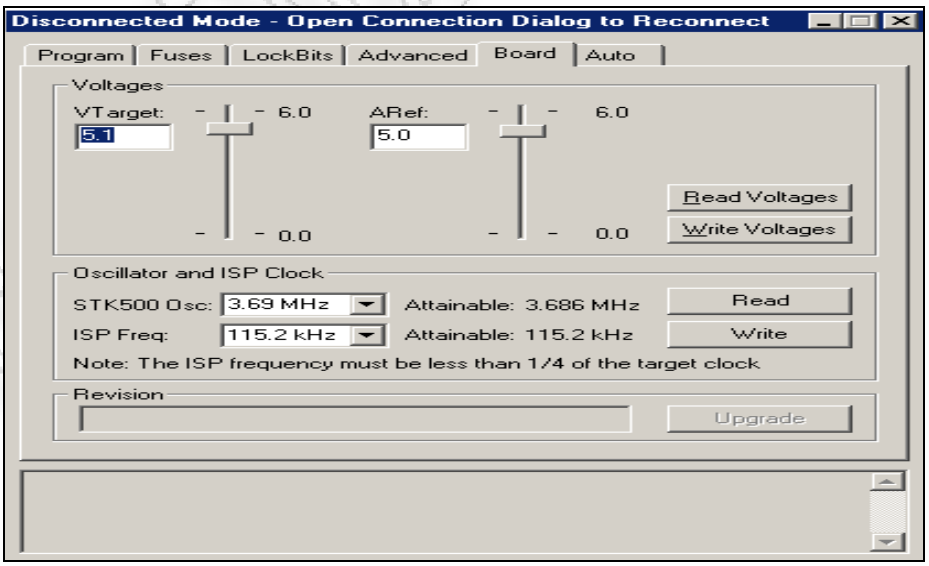
Το παράθυρο αυτό χωρίζεται σε τρία τμήματα: το τμήμα των bytes υπογραφής (Signature Bytes), το τμήμα του byte βαθμονόμησης ταλαντωτή (Oscillator Calibration Byte) και το τμήμα των ρυθμίσεων της επικοινωνίας (Communication Settings).

Στο πρώτο τμήμα, ο χρήστης έχει τη δυνατότητα να ελέγξει (Read) εάν το byte υπογραφής ανταποκρίνεται στην υπό χρήση διάταξη. Το byte αυτό παίζει το ρόλο του αναγνωριστή.

Στο δεύτερο τμήμα, ο χρήστης μπορεί να διαβάσει το byte βαθμονόμησης ταλαντωτή (Read) και να το γράψει σε μια θέση της μνήμης (Write). Το byte αυτό ορίζεται κατά τη διαδικασία κατασκευής της διάταξης και θα πρέπει να γραφεί στον καταχωρητή OSCCAL, έτσι ώστε να συντονίσει τον εσωτερικό RC ταλαντωτή με την επιλεγόμενη πηγή χρονισμού.

Στο τρίτο τμήμα, ο χρήστης ορίζει την ταχύτητα της επικοινωνίας (Baud Rate). Οι αλλαγές στο τμήμα αυτό ενεργοποιούνται άμεσα.

Το παράθυρο διαλόγου για την πέμπτη κατηγορία ρυθμίσεων φαίνεται στην επόμενη εικόνα.



Εικόνα 40: Παράθυρο διαλόγου για τις ρυθμίσεις πλακέτας.

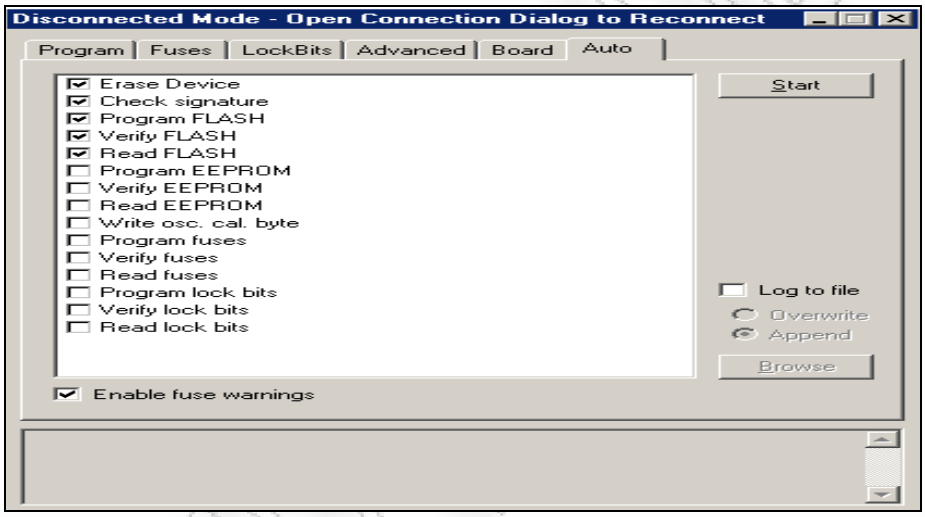
Το παράθυρο αυτό χωρίζεται σε τρία τμήματα: το τμήμα των τάσεων (Voltages), το τμήμα των πηγών χρονισμού του ταλαντωτή και του προγραμματιστή εντός του συστήματος (Oscillator and ISP Clock) και το τμήμα της βελτιωμένης έκδοσης (Revision).

Στο πρώτο τμήμα, ο χρήστης έχει τη δυνατότητα να διαβάσει (Read) και να γράψει (Write) την τάση λειτουργίας του μικροελεγκτή και την τάση αναφοράς της μονάδας ADC αυτού. Οι τάσεις αυτές ρυθμίζονται από 0 Volts έως 6 Volts σε βήματα των 0.1 Volts. Σημειώνεται πως η δεύτερη τάση πρέπει να είναι μικρότερη της πρώτης.

Στο δεύτερο τμήμα, ο χρήστης μπορεί να διαβάσει (Read) και να γράψει (Write) τη συχνότητα λειτουργίας του χρονιστή της αναπτυξιακής κάρτας STK500 και του προγραμματιστή εντός του συστήματος. Σημειώνεται πως η δεύτερη συχνότητα πρέπει να είναι μικρότερη από το 1/5 της πρώτης.

Στο τρίτο τμήμα, ο χρήστης βλέπει ορισμένα στοιχεία για την τρέχουσα έκδοση του μικροελεγκτή και επιλέγει ή όχι μια αναβάθμιση αυτού (Upgrade).

Το παράθυρο διαλόγου για την έκτη κατηγορία ρυθμίσεων φαίνεται στην επόμενη εικόνα.



Εικόνα 41: Παράθυρο διαλόγου για τις αυτόματες ρυθμίσεις.

Στο παράθυρο αυτό, ο χρήστης επιλέγει τις ενέργειες που θα εκτελεστούν κατά τον προγραμματισμό του μικροελεγκτή (Start). Αυτές αφορούν τον προγραμματισμό, την επιβεβαίωση και την ανάγνωση των Flash και EEPROM μνημών, αλλά και των Fuse και Lock Bits του μικροελεγκτή όπως, επίσης, τον καθαρισμό των μνημών αυτού, τον έλεγχο του byte υπογραφής και του byte βαθμονόμησης ταλαντωτή.

4.6 Η Αναπτυξιακή Κάρτα STK500

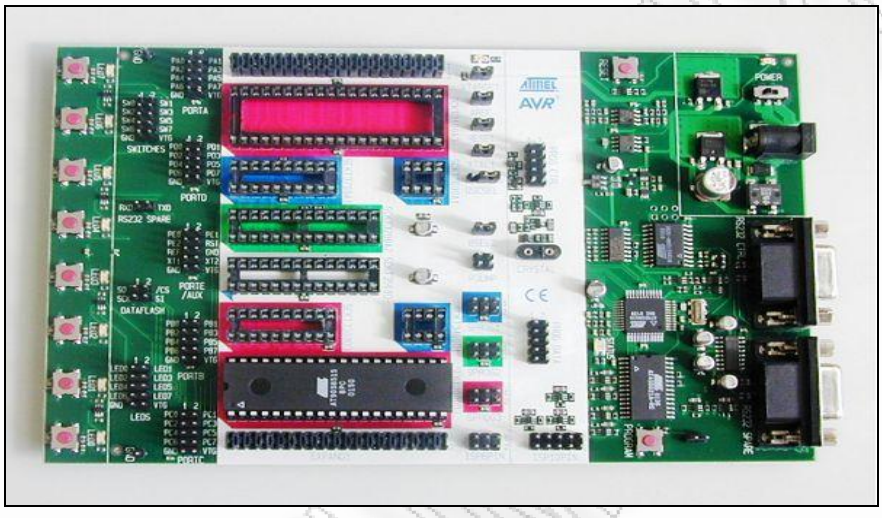
Για τον προγραμματισμό των μικροελεγκτών χρησιμοποιήθηκε η αναπτυξιακή κάρτα STK500, της εταιρείας Atmel. Ακολούθως, γίνεται μια περιγραφή της κάρτας STK500**.**

Τα βασικά χαρακτηριστικά της κάρτας αυτής είναι:

- Συμβατότητα με το AVR Studio.
- Δυνατότητα σύνδεσης με τον υπολογιστή μέσω της θύρας RS232 και επιπρόσθετη θύρα RS232 γενικής χρήσης.
- Ρυθμιζόμενη παροχή ισχύος για τάσεις από 10 Volts έως 15 Volts.
- Οχτώ υποδοχές για μικροελεγκτές AVR των 8, 20, 28 και 40 ακροδεκτών αντίστοιχα.

- Προγραμματισμός εντός του συστήματος όλων των μικροελεγκτών AVR.
- Σειριακός και παράλληλος προγραμματισμός υψηλής τάσης όλων των μικροελεγκτών AVR.
- Οχτώ διακόπτες πίεσης (Push Buttons) και οχτώ δίοδοι εκπομπής φωτός (Light Emitting Diodes) γενικής χρήσης.
- Δυνατότητα σύνδεσης με όλους τους ακροδέκτες των μικροελεγκτών AVR.
- Ενσωματωμένη μνήμη ταχείας αποθήκευσης δεδομένων AT45D021 χωρητικότητας 2 Mbits.

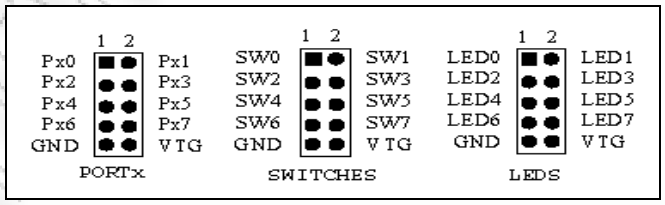
Η μορφή της αναπτυξιακής κάρτας STK500 φαίνεται στην επόμενη εικόνα.



Εικόνα 42: Η αναπτυξιακή κάρτα STK500.

Το block διάγραμμα της κάρτας αυτής χωρίζεται σε δύο τμήματα: το τμήμα ελέγχου (Control Section) και το τμήμα στόχου (Target Section). Παρατηρώντας κανείς τη μορφή της κάρτας μπορεί να διακρίνει τα δομικά στοιχεία αυτής. Στην αριστερή πλευρά (πράσινο τμήμα) βρίσκονται οι πιεστικοί διακόπτες και οι δίοδοι εκπομπής φωτός, αλλά και οι έξοδοι αυτών, οι οποίες μπορούν να συνδεθούν απευθείας μέσω ενός επιπέδου καλωδίου των δέκα συρμάτων (10-Wire Cable), με κάποια από τις θύρες του μικροελεγκτή. Χρησιμοποιούνται, κυρίως, για απλές εφαρμογές κατά τα πρώτα στάδια εξοικείωσης με την κάρτα.

Δίπλα ακριβώς βρίσκονται οι θύρες του μικροελεγκτή. Σημειώνεται πως λόγω του ότι η κάρτα αυτή δεν υποστηρίζει το μικροελεγκτή AVR με CPU την Atmega128, στις θύρες αυτές δεν περιλαμβάνονται η E, η F και η G. Η σύνδεση των θυρών με το κύκλωμα της εφαρμογής γίνεται, επίσης, μέσω ενός επιπέδου καλωδίου των δέκα συρμάτων. Στην παρακάτω Εικόνα 43 δίνεται η διάταξη των ακροδεκτών των στοιχείων που περιγράφηκαν προηγουμένως. Στην ίδια πλευρά της κάρτας διακρίνονται και η επιπρόσθετη RS232 θύρα για εξωτερική χρήση, αλλά και η ενσωματωμένη μνήμη ταχείας αποθήκευσης δεδομένων AT45D021.



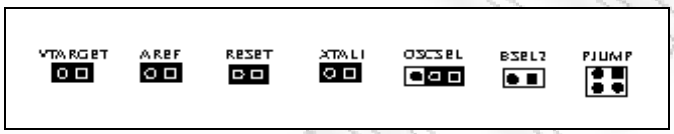
Εικόνα 43: Διάταξη ακροδεκτών των θυρών, των πιεστικών διακοπτών και των διόδων εκπομπής φωτός.

Στο κέντρο της κάρτας (άσπρο τμήμα) βρίσκονται οι οχτώ υποδοχές, στις οποίες τοποθετούνται οι μικροελεγκτές που πρόκειται να προγραμματιστούν. Σημειώνεται πως ένας

μόνο μικροελεγκτής θα πρέπει να χρησιμοποιείται κάθε φορά, ενώ απαιτείται ιδιαίτερη προσοχή κατά την τοποθέτηση του μικροελεγκτή στην κατάλληλη υποδοχή.

Ο προγραμματισμός του μικροελεγκτή μπορεί να γίνει με δύο τρόπους: προγραμματισμός εντός του συστήματος (In-System Programming) ή προγραμματισμός υψηλής τάσης σειριακός ή παράλληλος (Serial or Parallel High Voltage Programming). Στην πρώτη περίπτωση, η οποία χρησιμοποιήθηκε στην παρούσα εργασία, γίνεται χρήση της σειριακής διασύνδεσης περιφερειακών (Serial Peripheral Interface) του μικροελεγκτή, προγραμματίζοντας, τόσο τη Flash, όσο και την EEPROM μνήμη αυτού. Για τη λειτουργία αυτήν απαιτείται η σύνδεση της βάσης ISP6PIN με μια από τις βάσεις SPROG1, SPROG2 ή SPROG3. Η πρώτη σύνδεση χρησιμοποιείται εάν ο μικροελεγκτής έχει τοποθετηθεί στη μπλε υποδοχή, η δεύτερη εάν έχει τοποθετηθεί στην πράσινη υποδοχή και η τρίτη εάν έχει τοποθετηθεί στην κόκκινη υποδοχή. Η σύνδεση πραγματοποιείται μέσω ενός επιπέδου καλωδίου των έξι συρμάτων (6-Wire Cable).

Δίπλα ακριβώς βρίσκονται η μονάδα ελέγχου μνήμης (Memory Control Unit) και οχτώ γέφυρες βραχυκύκλωσης (Jumpers). Στο παρακάτω σχήμα δίνεται η εξ' ορισμού συνδεσμολογία των γεφυρών αυτών.

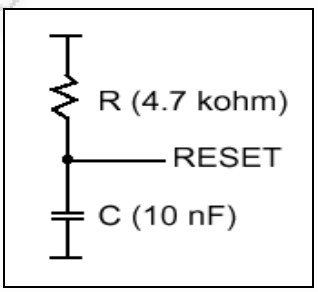


Εικόνα 44: Εξ' ορισμού συνδεσμολογία των οχτώ γεφυρών βραχυκύκλωσης.

Η πρώτη γέφυρα ελέγχει την τάση τροφοδοσίας μέσω του AVR Studio. Εάν η γέφυρα αυτήν είναι βραχυκυκλωμένη, χρησιμοποιείται η τροφοδοσία που παρέχεται από την κάρτα, η οποία ρυθμίζεται από 0 Volts έως 6 Volts και παρέχει ρεύμα 0.5 Amperes στο τμήμα στόχου. Στην αντίθετη περίπτωση, το κύκλωμα της εφαρμογής θα πρέπει να παρέχει την τάση αναφοράς στην κάρτα μέσω των αντίστοιχων ακροδεκτών του μικροελεγκτή.

Η δεύτερη γέφυρα τροφοδοτεί την ADC μονάδα του μικροελεγκτή με την αναλογική τάση αναφοράς. Εάν η γέφυρα αυτήν είναι βραχυκυκλωμένη, η σύνδεση πραγματοποιείται εσωτερικά και ρυθμίζεται μέσω του AVR Studio από 0 Volts έως 6 Volts. Στην αντίθετη περίπτωση, το κύκλωμα της εφαρμογής θα πρέπει να παρέχει την τάση αναφοράς στην ADC μονάδα μέσω των αντίστοιχων ακροδεκτών του μικροελεγκτή.

Η τρίτη γέφυρα ελέγχει το σήμα επανατοποθέτησης της κάρτας. Εάν η γέφυρα αυτήν είναι βραχυκυκλωμένη, η MCU ελέγχει το σήμα αυτό του μικροελεγκτή. Στην αντίθετη περίπτωση, το σήμα αυτό ελέγχεται από ένα εξωτερικό κύκλωμα επανατοποθέτησης μέσω του αντίστοιχου ακροδέκτη του μικροελεγκτή. Μια τυπική μορφή ενός τέτοιου συστήματος δίνεται στην παρακάτω Εικόνα 45. Σημειώνεται πως εάν η τιμή της εξωτερικής αντίστασης πρόσδεσης είναι μικρή (μικρότερη από 4.7 Kohms), η κάρτα δεν είναι σε θέση να ελέγξει το σήμα επανατοποθέτησης.



Εικόνα 45: Εξωτερικό κύκλωμα επανατοποθέτησης.

Οι επόμενες δύο γέφυρες ελέγχουν τις πηγές χρονισμού του συστήματος. Εάν η γέφυρα XTAL1 είναι βραχυκυκλωμένη, η εσωτερική μονάδα ρολογιού συστήματος της κάρτας χρησιμοποιείται ως χρονιστής του μικροελεγκτή. Στην αντίθετη περίπτωση, το ρόλο αυτόν

αναλαμβάνει ένας εξωτερικός χρονιστής ή ένας κρύσταλλος. Όσον αφορά στη γέφυρα OSCSEL, παρέχει τρεις δυνατότητες όταν η γέφυρα XTAL1 είναι βραχυκυκλωμένη. Εάν είναι βραχυκυκλωμένη δεξιά, ως κύριος χρονιστής του συστήματος χρησιμοποιείται ένας χρονιστής, ο οποίος παράγεται από την MCU μέσω του software. Εάν είναι βραχυκυκλωμένη αριστερά, ως κύριος χρονιστής του συστήματος χρησιμοποιείται ένας ενσωματωμένος στην κάρτα κρύσταλλος. Εάν δεν υπάρχει κανένα βραχυκύκλωμα, ο εσωτερικός χρονιστής της κάρτας δε χρησιμοποιείται.

Οι επόμενες τρεις γέφυρες, αλλά και οι βάσεις Prog Ctrl και Prog Data χρησιμοποιούνται κατά τον προγραμματισμό σε υψηλή τάση και δε θα περιγραφούν.

Στα άκρα του κεντρικού μέρους της κάρτας βρίσκονται δύο υποδοχές προέκτασης (Expand 0 και Expand 1), στις οποίες συνδέθηκε ο προσαρμογέας STK501.

Όσον αφορά στη δεξιά πλευρά της κάρτας (πράσινο τμήμα), εκεί βρίσκονται δύο πιεστικοί διακόπτες (Push Buttons) και τρεις δίοδοι εκπομπής φωτός (Light Emitting Diodes) εξειδικευμένης χρήσης. Ο πρώτος διακόπτης χρησιμοποιείται για την επανατοποθέτηση του μικροελεγκτή. Ο δεύτερος διακόπτης χρησιμοποιείται για την αναβάθμιση της Flash μνήμης, σε περίπτωση μελλοντικής ανίχνευσης λογισμικού μη συμβατό με την υπό χρήση MCU. Αναφορικά με τις διόδους εκπομπής φωτός, το πρώτο led (Main Power Led) ενεργοποιείται με την παροχή ισχύος στην κάρτα, ενώ το δεύτερο (Target Power Led) με την παροχή ισχύος στον ενσωματωμένο μικροελεγκτή. Σχετικά με το τρίτο (Status Led), αυτό είναι τριών χρωμάτων: κατά την εκκίνηση είναι κόκκινο, κατά τη διάρκεια προγραμματισμού του μικροελεγκτή γίνεται κίτρινο και εάν ο προγραμματισμός επιτύχει γίνεται πράσινο, αλλιώς γίνεται ξανά κόκκινο.

Στην άκρη του τμήματος αυτού βρίσκεται μια υποδοχή για παροχή ισχύος και ένας διακόπτης τροφοδοσίας. Η τάση τροφοδοσίας θα πρέπει να κυμαίνεται μεταξύ 10 Volts και 15 Volts. Στη συγκεκριμένη περίπτωση δόθηκε τάση των 12 Volts. Επίσης, υπάρχει και μια θύρα RS232 για σύνδεση της κάρτας με τον υπολογιστή. Η σύνδεση αυτήν επιτυγχάνεται μέσω ενός θηλυκού RS232 καλωδίου των εννέα ακροδεκτών.

Για επιπλέον πληροφορίες σχετικά με την αναπτυξιακή κάρτα STK500, την αντιμετώπιση πιθανών σφαλμάτων κατά τη λειτουργία της (Troubleshooting Guide) και τη χειροκίνητη αναβάθμιση του λογισμικού της (Manual Firmware Upgrade), ο αναγνώστης παραπέμπεται στο εγχειρίδιο του κατασκευαστή.

Στη δεξιά πλευρά του προσαρμογέα βρίσκονται, επίσης, η επιπρόσθετη θύρα RS232 γενικής χρήσης και η επιπρόσθετη SRAM (XMEM). Για επιπλέον πληροφορίες σχετικά με τον προσαρμογέα STK501, ο αναγνώστης παραπέμπεται στο εγχειρίδιο του κατασκευαστή.

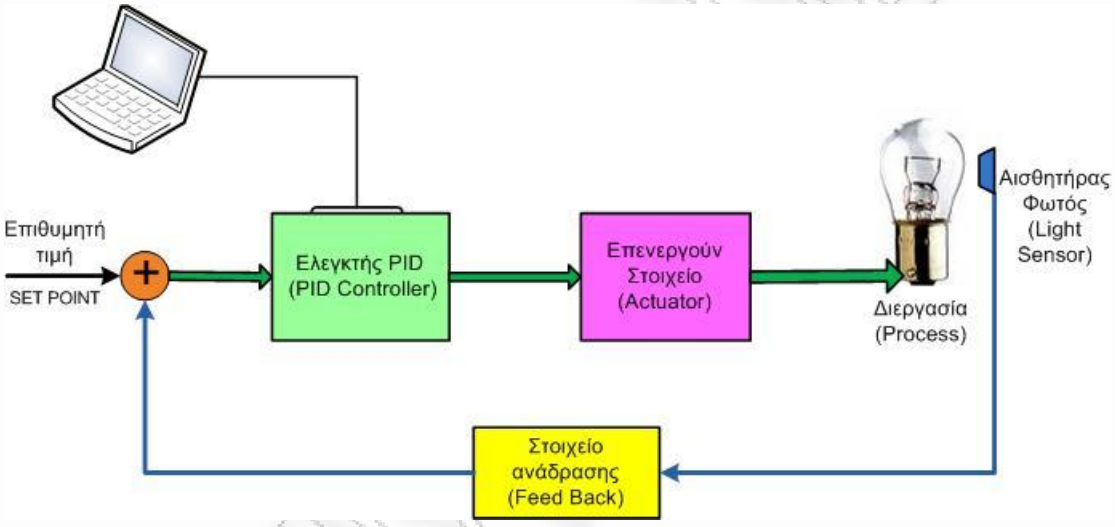
5 ΚΕΦΑΛΑΙΟ 5

Σχεδίαση – Υλοποίηση Του Επενεργητή

5.1 Εισαγωγή

Στο κεφάλαιο αυτό εξηγείται με λεπτομέρειες η διαδικασία που ακολουθήθηκε κατά το σχεδιασμό του ελεγκτή PID και του επενεργητή για τον έλεγχο του φωτισμού με λαμπτήρα πυρακτώσεως καθώς και για την κατασκευή ως την λειτουργία και τις τελικές δοκιμές.

Το σύστημα αυτομάτου ελέγχου που σχεδιάσθηκε και υλοποιήθηκε φαίνεται στην παρακάτω Εικόνα 46.



Εικόνα 46: Το σύστημα αυτομάτου ελέγχου που υλοποιείται στην διατριβή.

Όπως φαίνεται στο παραπάνω μπλοκ διάγραμμα του συστήματος το φυσικό σύστημα (Διεργασία) που πρέπει να ελέγξουμε είναι ο ηλεκτρικός λαμπτήρας πυρακτώσεως και η μεταβλητή ελέγχου είναι η φωτεινή ροή του λαμπτήρα.

Από την εμπειρία μου στα συστήματα αυτομάτου ελέγχου σε βιομηχανικό περιβάλλων αποφάσισα ο ελεγκτής PID και ο επενεργητής να κατασκευαστούν σε ξεχωριστές πλακέτες και να ελέγχονται από δύο ξεχωριστούς μικροελεγκτές. Αυτό έχει το μειονέκτημα μεγαλύτερου κόστους της κατασκευής και την ανάγκη το σήμα που θα βγάζει ο μικροελεγκτής του PID να πρέπει να μετατραπεί από ψηφιακό σε αναλογικό με D/A μετατροπέα και αντίστοιχα για να το πάρει ο μικροελεγκτής του επενεργητή θα πρέπει να μετατραπεί πάλει από αναλογικό σε ψηφιακό με A/D μετατροπέα. Αλλά οι ξεχωριστές πλακέτες έχουν τα παρακάτω πλεονεκτήματα.

- Μπορούν να χρησιμοποιηθούν αυτόνομα το κάθε κύκλωμα π.χ. ο επενεργητής μπορεί να χρησιμοποιηθεί σε αυτό το σύστημα σε έλεγχο ανοιχτού συστήματος (open loop) χωρίς την ανάγκη ύπαρξης του PID λειτουργώντας σαν Dimmer.
- Ο PID μπορεί να χρησιμοποιηθεί σε άλλο σύστημα αυτομάτου ελέγχου με διαφορετικό επενεργητή που θα μπορεί να δεχθεί αναλογικό σήμα 0 – 10 V γιατί μπορεί να δεχθεί για ανατροφοδότηση αναλογικό σήμα 0 – 5 V.
- Η απλότητα στην σχεδίαση των κυκλωμάτων και στην κατασκευή των πλακετών είναι πάρα πολύ σημαντική γιατί δεν μπορούμε να διαθέσουμε επαγγελματικά εργαλεία.

Επιπλέον έχουμε πολύ σημαντική βοήθεια στον προγραμματισμό στην χρίση των timer, ADC, I/O κ.λ.π.

- Και βέβαια το μεγάλο πλεονέκτημα είναι όταν χαλάσει κάποιο εξάρτημα κατά την λειτουργία ή κατά τις δοκιμές η αντικατάσταση είναι ποιο εύκολη και ποιο οικονομική αλλά και πολύ ποιο γρήγορα στο να εντοπίσουμε την βλάβη.

5.2 Σχεδίαση Του Επενεργητή (Actuator)

Η αρχή λειτουργίας της παραγωγής φωτισμού με λαμπτήρα πυρακτώσεως στηρίζεται στο φαινόμενο της θέρμανσης μεταλλικού νήματος (συνήθως από βολφράμιο) με την βοήθεια ηλεκτρικού ρεύματος ,με απουσία οξυγόνου. Το νήμα αναπτύσσει υψηλή θερμοκρασία και ακτινοβολεί φως. Παράλληλα βέβαια παράγεται και θερμότητα, η οποία είναι ανεπιθύμητη (απώλειες). Γενικά στους λαμπτήρες πυρακτώσεως, μόνο ποσοστό 10-20% της καταναλισκόμενης ηλεκτρικής ενέργειας μετατρέπεται σε φωτεινή ενέργεια (φως).

Μερικά βασικά φωτομετρικά μεγέθη είναι τα παρακάτω:

- Φωτεινή Ενέργεια: Ονομάζεται η ενέργεια που ακτινοβολείτε από μια φωτεινή πηγή. Μονάδα μετρήσεως είναι το ωριαίο Lumen. Φωτεινή ενέργεια: Q (Lumen * h).
- Φωτεινή ισχύς ή φωτεινή ροή: Ονομάζεται το ολικό ποσό φωτεινής ενέργειας, που εκπέμπεται, από την φωτεινή πηγή σε κάθε δευτερόλεπτο. Φωτεινή ισχύς: Φ (Lm) Μονάδα μετρήσεως είναι το Lumen. Αν υποθέσουμε ότι σε χρόνο (T) ακτινοβολείτε από μια πηγή, φωτεινή ενέργεια Q τότε η φωτεινή ισχύς είναι:

$$\Phi = \frac{Q}{T}$$

Q: σε Lm*h, T: σε h

Η φωτεινή ισχύς έχει συγγένεια με την ηλεκτρική ισχύ και το λούμεν με το βατ.

1 Lumen = 0,0016w

- Φωτεινή ένταση: Ονομάζεται η ποσότητα φωτεινής ισχύος που διέρχεται από τη μονάδα της στερεάς γωνίας, που έχει την γωνία, σαν κορυφή προς ορισμένη κατεύθυνση. Η φωτεινή ένταση συμβολίζεται με το γράμμα (I).

Αν μια πηγή αποδίδει φωτεινή ισχύ (Φ) με μια φωτεινή δέσμη στερεάς γωνίας (ω), τότε η φωτεινή ένταση είναι:

$$I = \frac{\Phi}{\omega}$$

Η φωτεινή ένταση πηγής είναι διαφορετική στις διάφορες κατευθύνσεις. Μετρείται σε σύγκριση με πρότυπο πηγή που ονομάζεται νέο κηρίο. Η μονάδα νέο κηρίο ονομάζεται και Candela (Cd).

- Ένταση φωτισμού επιφανείας: Ονομάζεται η ποσότητα φωτεινής ροής που δέχεται η μονάδα εμβαδού φωτιζόμενης επιφάνειας. Συμβολίζεται με το γράμμα (E). Μονάδα μετρήσεως είναι το Lux

Αν (Φ) είναι η φωτεινή ισχύς, που προσπίπτει σε μια επιφάνεια εμβαδού (P) τότε:

$$E = \frac{\Phi}{f}$$

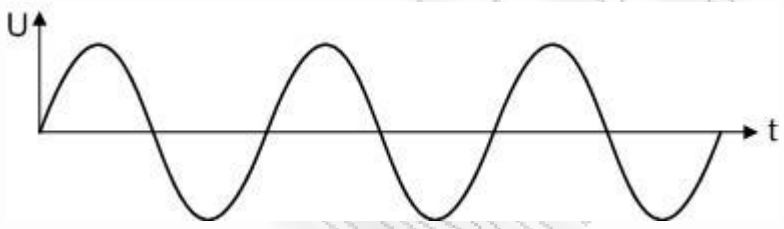
όπου Φ: σε Lumen, f: σε m^2, E: σε Lux

Η απόδοση κάθε συστήματος είναι ίση με το πηλίκο της ωφέλιμης προς τη δαπανούμενη ισχύ του. Στην περίπτωση ηλεκτρικού λαμπτήρα ως ωφέλιμη ισχύς θεωρείται η φωτεινή ροή Φολ=4πI και ως δαπανούμενη η ηλεκτρική ισχύς. Για "ωμικό" φορτίο που είναι ο λαμπτήρας η ηλεκτρική ισχύς είναι : P=Uεν. Ιεν (Uεν, Ιεν οι ενεργές τιμές τάσης και ρεύματος αντίστοιχα) Επομένως, η απόδοση του ηλεκτρικού λαμπτήρα είναι: $n = \frac{4\pi \mathrm{I}}{U_{\varepsilon\nu} I_{\varepsilon\nu}}$

και, βεβαίως, είναι αδιάστατο μέγεθος, ως πηλίκο ομοειδών μεγεθών (Φ, P) αλλά όχι καθαρός αριθμός αφού εκφράζεται σε Lumen/Watt. Με τη βοήθεια της προηγούμενης σχέσης προέκυψε η τιμή : 625 Lm/Watt που αποτελεί την απόδοση του ιδανικού φωτεινού ηλεκτρικού λαμπτήρα, που θα μετέτρεπε ολόκληρη την ηλεκτρική ισχύ που του προσφέρεται σε φωτεινή ισχύ.

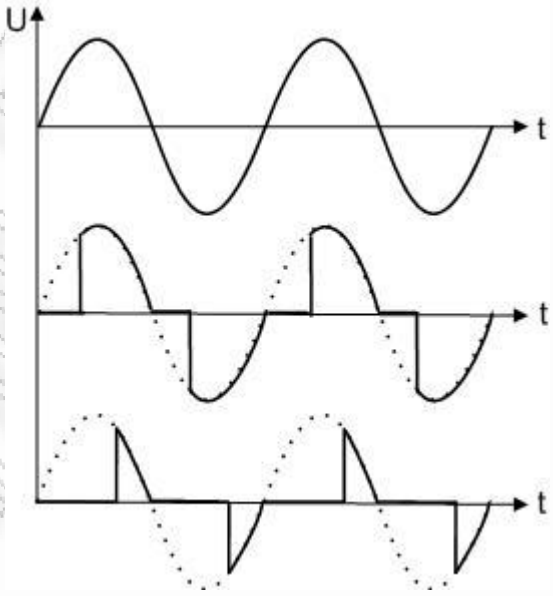
Από τα παραπάνω είναι φανερό ότι για να ελέγξουμε την ένταση του φωτός που παράγει ο λαμπτήρας πρέπει να ελέγξουμε την ηλεκτρική ισχύ με την οποία τροφοδοτούμε τον λαμπτήρα. Για να το πετύχουμε αυτό θα χρησιμοποιήσουμε την τεχνική του ελέγχου της γωνίας της φάσης(AC phase angle control).

Στο παρακάτω σχήμα βλέπουμε την ημιτονοειδή κυματομορφή της τάσης που έχουμε από το δίκτυο της ΔΕΗ και με την οποία θα τροφοδοτείται ο λαμπτήρας.



Εικόνα 47: κυματομορφή της τάσης του δικτύου.

Αν το φορτίο γίνεται ΟΝ στην αρχή του ημικύκλιου του ημίτονου θα λαμβάνει όλη την ισχύ από την γραμμή τροφοδοσίας ενώ αν δεν γίνεται ΟΝ στην διάρκεια του ημικύκλιου του ημίτονου δεν θα λαμβάνει καθόλου ισχύ. Ενώ αν γίνεται ΟΝ για κάποιο τμήμα του ημικύκλιου θα λαμβάνει ένα τμήμα από την ολική διαθέσιμη ισχύ όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 48: Τμήματα της τάσης του δικτύου.

Η συχνότητα της τάσεως του δικτύου είναι f=50 Hz. επομένως η περίοδος είναι T=1/f=1/50=20 msec. Άρα η κάθε ημιπερίοδος είναι 10 msec. Από αυτό συνεπάγεται ότι για να

μπορέσουμε να κόψουμε τα τμήματα της τάσης που θα εφαρμόζεται στον λαμπτήρα θα πρέπει να έχουμε ένα πάρα πολύ γρήγορο διακόπτη με πολύ γρήγορη απόκριση.

Από την θεωρία των ηλεκτρονικών ισχύος τόσο γρήγοροι είναι οι διακόπτες ξηρού στοιχείου όπως είναι το Transistor, το Thyristor και το Triac.

Τα Transistor ισχύος και ιδικά η κατηγορία των MOSFET έχουν ικανότητες διακοπής οι οποίες είναι πάρα πολύ γρήγορες και η συχνότητα διακοπής ξεπερνά τα 10^4 Hz, αλλά η ισχύς την οποία μπορούν να διακόψουν δεν είναι πολύ μεγάλη και επιπλέον η ικανότητα διακοπής τους είναι μόνο για συνεχείς τάσης. Στο Thyristor η συχνότητα διακοπής είναι πολύ μικρότερή από αυτή των Transistor ισχύος αλά οι ισχύς την οποία μπορούν να διακόψουν είναι πάρα πολύ μεγάλη της τάξεως του 10^8 VA αλλά και αυτά άγουν μόνο όταν πολώνονται ορθά δηλ. Κατά την θετική ημιπερίοδο της τάσης. Τα Triac έχουν διακοπτικές ικανότητες αντίστοιχες με αυτές των Thyristor και επιπλέον έχουν την δυνατότητα να άγουν και στην θετική και στην αρνητικοί ημιπερίοδο της τάσης που θα εφαρμοστή στο φορτίο.

Στην περίπτωση μας όπου θέλουμε να ελέγχουμε και τις θετικές και τις αρνητικές ημιπεριόδους της τάσης που θα τροφοδοτεί τον λαμπτήρα το κατάλληλο διακοπτικό στοιχείο που πρέπει να επιλέξουμε για τον οδηγό ισχύος (Power driver) στον λαμπτήρα είναι το Triac.

Για την λειτουργία του Triac πρέπει να του δίνουμε παλμούς έναυσης στην πύλη του στο χρονικό σημείο που θέλουμε να ξεκινήσει να άγει και για τόσο χρόνο έτσι ώστε το ρεύμα που περνά από το triac που είναι το ρεύμα του φορτίου να ξεπεράσει την τιμή του ρεύματος συγκράτησης του triac. Την τιμή του ρεύματος συγκράτησης μας την δίνει ο κατασκευαστής του Triac στο data sheet του Triac μαζί με όλα τα υπόλοιπα ηλεκτρικά χαρακτηριστικά του. Η σβέση του Triac προκαλείτε από μόνη της όταν η τάση του φορτίου διέρχεται από το μηδέν γιατί σε ατό το σημείο η τιμή του ρεύματος πέφτει κάτω από την τιμή συγκράτησης του Triac σε αγωγή. Άρα και η τάση που θα εφαρμόζεται στο λαμπτήρα θα είναι από το σημείο έναυσης μέχρι την επόμενη διέλευση της τάσης από το μηδέν όπως φαίνεται στην Εικόνα 48. άρα ένα τμήμα του κυκλώματος του επενεργητή θα είναι το κύκλωμα οδήγησης ισχύος (Power driver) με κύριο στοιχείο το triac και όλα τα υπόλοιπα εξαρτήματα που χρειάζεται για να λειτουργήσει.

Για να ξέρουμε το χρονικό σημείο στο οποίο θα δώσουμε παλμό έναυσης στην πύλη του Triac θα πρέπει να ξέρουμε το χρονικό σημείο από το οποίο διέρχεται η τάση τροφοδοσίας του λαμπτήρα από το μηδέν. Επομένως ένα δεύτερο τμήμα που θα πρέπει να περιλαμβάνει ο επενεργητής είναι το κύκλωμα ανίχνευσης διέλευσης της τάσης από το μηδέν(Zero crossing detector).

Από τα παραπάνω προκύπτει ότι στον ενεργοποιητή θα πρέπει να υπάρχει μια λογική μονάδα ελέγχου ή οποία θα επεξεργάζεται των αλγόριθμο σύμφωνα με τον οποίο θα δίνει τους παλμούς έναυσης στην πύλη του triac στο κατάλληλο χρονικό σημείο σύμφωνα με το σήμα που θα παίρνει από τον ελεγκτή PID και το σήμα που θα παίρνει από το κύκλωμα Zero crossing detector όταν η τάση διέρχεται από το μηδέν. Τα σήματα αυτά θα τα δέχεται η λογική μονάδα ελέγχου μέσω μονάδων εισόδου – εξόδου από που από αυτές τις μονάδες θα δίνει και το σήμα έναυσης στο triac.

Επειδή το σήμα που θα δίνει ο ελεγκτής PID είναι αναλογικό και η λογική μονάδα ελέγχου θα είναι ψηφιακή για να μπορέσει να διαβάσει η λογική μονάδα ελέγχου το σήμα από τον PID θα πρέπει μεταξύ αυτών των δύο να υπάρξει ένα ακόμη κύκλωμα το οποίο θα κάνει την μετατροπή του αναλογικού σήματος του PID σε ψηφιακή μορφή. Η τάση τροφοδοσίας των κυκλωμάτων θα είναι τα 5V έτσι το V max reference του ADC μετατροπέα θα είναι τα 5V άρα και το μέγιστο σήμα που θα μπορεί να μετατρέψει σε ψηφιακή μορφή θα είναι τα 5V.

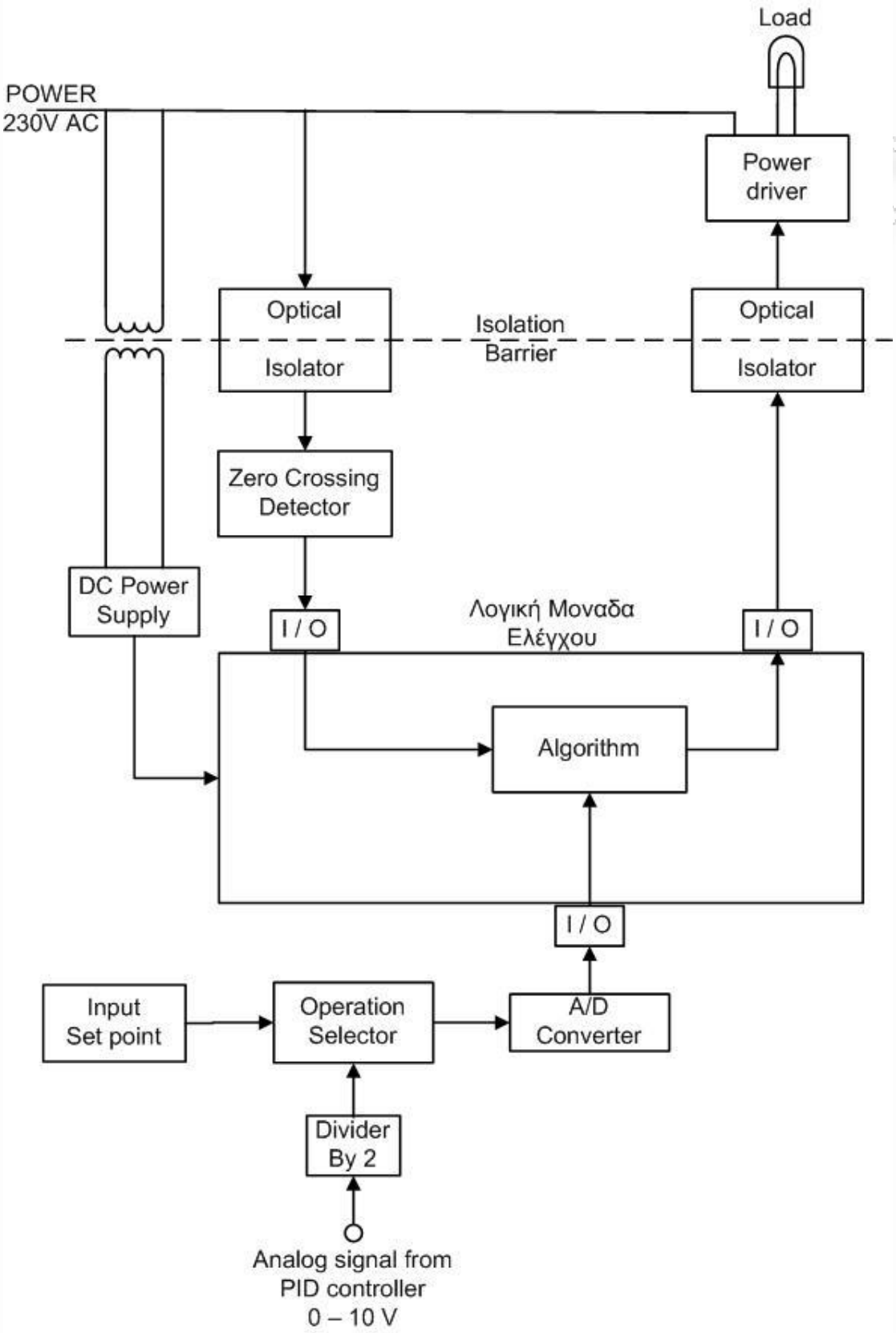
Το αναλογικό σήμα που θα δίνει ο PID θα είναι από 0 ως 10 V για λόγους τυποποίησης και να μπορεί ο επενεργητής να λειτουργήσει σε άλλο σύστημα αυτοματισμού με άλλων ελεγκτή. Επειδή όμως ο ADC μπορεί να μετατρέψει μέγιστο σήμα μέχρι 5V που είναι το μισό από το μέγιστο σήμα που θα δίνει ο PID πρέπει να παρεμβάλουμε μεταξύ του σήματος το PID και του ADC ένα κύκλωμα διαίρεσης του σήματος δια δύο για να έχουμε σωστή μετατροπή του αναλογικού σήματος σε ψηφιακό.

Για να μπορεί ο επενεργητής να λειτουργεί και σε ανοιχτό σύστημα ελέγχου θα πρέπει να έχει την δυνατότητα επιλογής λειτουργίας σε ανοιχτό ή κλειστό σύστημα καθώς επίσης και όταν επιλέγουμε ανοιχτό σύστημα την δυνατότητα να του δίνουμε την επιθυμητή τιμή εξόδου.

Έτσι στο κύκλωμα του επενεργητή προσθέτουμε την μονάδα επιλογής λειτουργίας καθώς και την μονάδα εισόδου επιθυμητής τιμής ανοιχτού συστήματος. Επομένως ο επενεργητής έχει τις παρακάτω μονάδες:

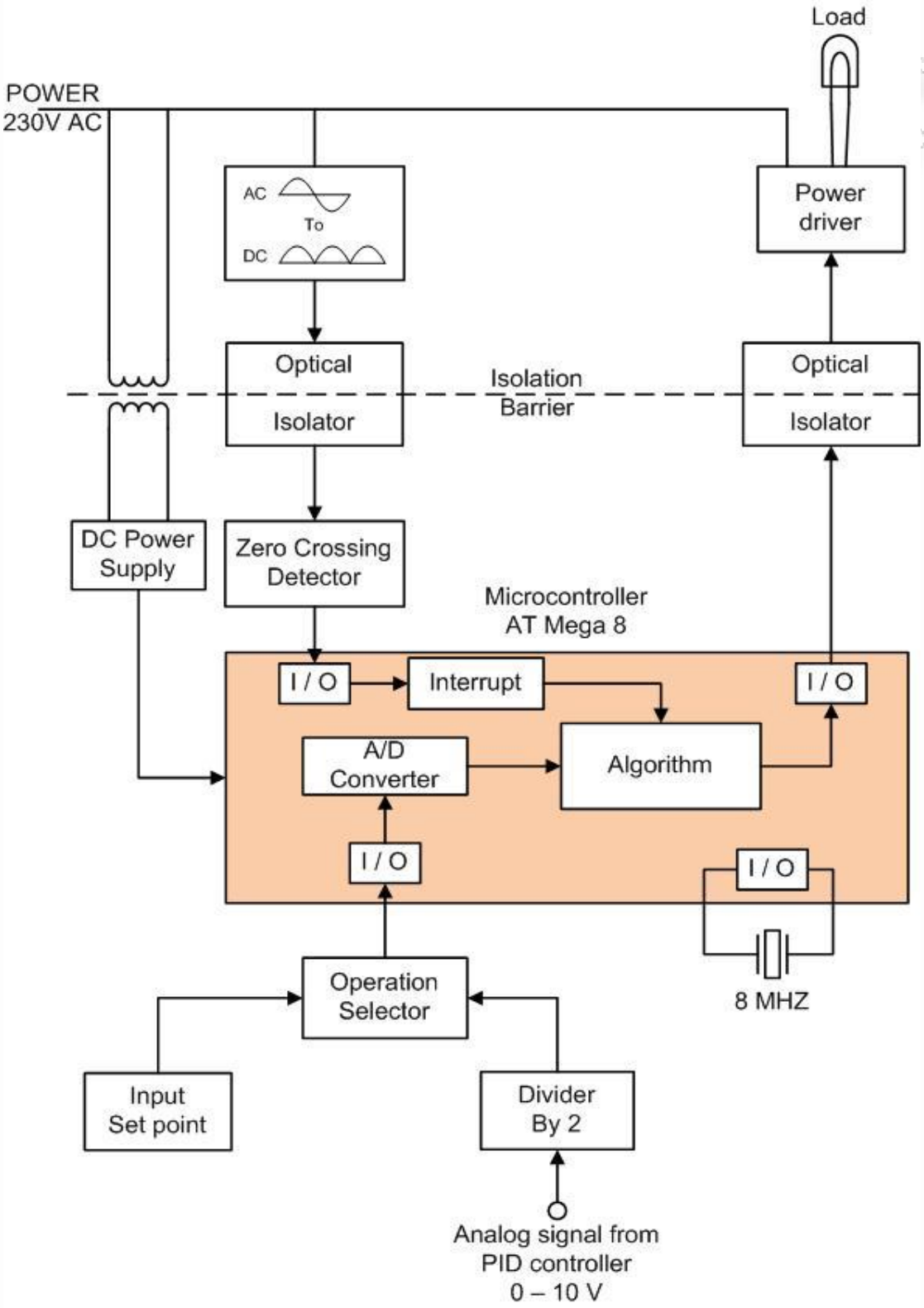
- Μονάδα οδήγησης ισχύος(Power driver).
- Μονάδα ανίχνευσης διέλευσης της τάσης από το μηδέν(Zero crossing detector).
- Μονάδα λογικού ελέγχου.
- Μονάδες εισόδου - εξόδου προς την λογική μονάδα.
- Μονάδα μετατροπής αναλογικού σήματος σε ψηφιακό.
- Μονάδα επιλογής λειτουργίας.
- Μονάδα εισόδου επιθυμητής τιμής σε λειτουργία ανοιχτού συστήματος.
- Μονάδα διαίρεσης αναλογικού σήματος δια δύο.

Η διασύνδεση των παραπάνω μονάδων φαίνεται στο παρακάτω μπλοκ διάγραμμα.



Εικόνα 49: Μπλοκ διάγραμμα των μονάδων του επενεργητή.

Σαν λογική μονάδα ελέγχου επιλέχθηκε να είναι ένας μικροελεγκτής της εταιρίας ATMEL ο ATMega8 ο οποίος διαθέτει ενσωματωμένες και τις μονάδες εισόδου – εξόδου και τον μετατροπέα από αναλογικό σήμα σε ψηφιακό. Το νέο μπλοκ διάγραμμα είναι.



Εικόνα 50: Μπλοκ διάγραμμα των μονάδων του επενεργητή με τον μικροελεγκτή.

5.3 Επιλογή Εξαρτημάτων Και Σχεδίαση Του Κυκλώματος.

Στην παράγραφο αυτή θα αναληθή η επιλογή των εξαρτημάτων για κάθε μία από τις μονάδες του παραπάνω μπλοκ διαγράμματος του επενεργητή και θα γίνει η σχεδίαση του σχηματικού κυκλώματος και τις πλακέτας δηλ. του PCB.

Επειδή στον επενεργητή χρησιμοποιείται υψηλή τάση (τάση του δικτύου παροχής από την ΔΕΗ AC 230 V), ο επενεργητής πρέπει να είναι σύμφωνος με τις ηλεκτρικές διατάξεις ασφαλείας. Πρέπει να εξασφαλισθεί γαλβανική απομόνωση μεταξύ των εξαρτημάτων που συνδέονται με το δίκτυο παροχής ισχύος και του υπόλοιπου κυκλώματος. Οποιοδήποτε εξάρτημα χρησιμοποιηθεί για να εξασφαλίσει τέτοια απομόνωση θα έχει μονωτική ικανότητα τουλάχιστον 2000 V. Στην σχεδίαση των γραμμών του τυπωμένου κυκλώματος θα υπάρχει απόσταση μεταξύ των γραμμών που έχουν AC τάση και των υπολοίπων τουλάχιστον 2mm απόσταση για να εξασφαλισθεί επαρκή απομόνωση.

- Μονάδα οδήγησης ισχύος(Power driver).

Όπως αναφέρθηκε παραπάνω το κύριο εξάρτημα στην μονάδα οδήγησης ισχύος θα είναι ένα triac. Τα κύρια στοιχεία για την επιλογή του triac είναι ή ισχύς που πρέπει να δόση στο φορτίο, ή μέγιστη ανάστροφη τάση που μπορεί να δεχθεί, ο μέγιστος ρυθμός μεταβολής της τάσης dV/dt, το ρεύμα αυτοσυγκράτησης. Έτσι βάση των προηγούμενων χαρακτηριστικών επιλέγεται το BT137. τα φίλτρα και τα υπόλοιπα εξαρτήματα που χρειάζεται δίνονται από τον κατασκευαστή του triac. Σαν εξάρτημα απομόνωσης μεταξύ της AC τάσης και του υπόλοιπου κυκλώματος επιλέγεται το MCO3021 το οποίο το ίδιο κάνει και απομόνωση και είναι και οδηγός για την πύλη του triac με μονωτική ικανότητα 7500 V AC peak / 60Hz με διάρκεια 1sec. τα υπόλοιπα εξαρτήματα που χρειάζεται δίνονται από τον κατασκευαστή στο data sheet του MCO3021.

- Μονάδα εισόδου επιθυμητής τιμής σε λειτουργία ανοιχτού συστήματος.

Για την είσοδο της επιθυμητής τιμής στην περίπτωση τις λειτουργίας του επενεργητή σε ανοικτό σύστημα επιλέγεται ένα ποτενσιόμετρο το οποίο θα τροφοδοτείτε με την τάση τροφοδοσίας του μικροελεγκτή. Θα κάνει διαίρεση τάσης και θα τροφοδοτεί τον ίδιο μετατροπέα ADC του μικροελεγκτή που θα τροφοδοτεί και το αναλογικό σήμα από τον PID ελεγκτή. Με αυτό τον τρόπο ο αλγόριθμος που θα εκτελείται στον μικροελεγκτή θα είναι ο ίδιος χωρίς να χρειάζεται να γράψουμε επιπλέον κώδικα γιατί δεν γνωρίζει αν το σήμα που λαμβάνει ο μετατροπέας ADC προέρχεται από τον PID ελεγκτή ή από το ποτενσιόμετρο επιθυμητής τιμής.

- Μονάδα επιλογής λειτουργίας.

Σαν μονάδα επιλογής λειτουργίας επιλέχθηκε ένας επιλογικός διακόπτης δύο θέσεων ο οποίος στην μια θέση συνδέει την είσοδο του αναλογικού σήματος από τον PID με τον μετατροπέα ADC του μικροελεγκτή και στην άλλη θέση συνδέει την έξοδο του ποτενσιόμετρου επιθυμητής τιμής με τον μετατροπέα ADC του μικροελεγκτή.

- Μονάδα λογικού ελέγχου.

Για μονάδα λογικού ελέγχου επιλέχθηκε ο μικροελεγκτής των 8 bit ATmega8 της εταιρίας ATMEL που φαίνεται στην παρακάτω Εικόνα 51.

Εικόνα 51: Ο μικροελεγκτής ATmega 8 της ATMEL

Η συγκεκριμένη εταιρία διαθέτη μια μεγάλη γκάμα από μικροελεγκτές με ευρύ φάσμα εισόδων – εξόδων, μνήμης κ.λ.π. επιλέχθηκε ο συγκεκριμένος γιατί διαθέτει 23 προγραμματιζόμενες γραμμές εισόδου - εξόδου, τρεις timers έναν 16 bit και δύο 8 bit που χρειάζονται στην εκτέλεση του αλγόριθμου. Έχει 8 Kbyte μνήμη Flash για πρόγραμμα, 1 Kbyte μνήμη SRAM για δεδομένα και 512 byte μνήμη EEPROM. Επιπλέον διαθέτει έξη ενσωματωμένους μετατροπής αναλογικού σήματος σε ψηφιακό με ακρίβεια 10 bit και μέγιστη τάση εισόδου 5V. Αυτό σημαίνει ότι εφόσον πριν των ADC υπάρχει μονάδα διαίρεσης δια δύο το μέγιστο σήμα που θα φτάνει στον ADC είναι τα 5V και επειδή η ανάλυση του ADC είναι 10 bit η αλλαγή της ψηφιακής τιμής θα είναι: (5 V/1024)*2=9,76 mV άρα θα έχουμε μεταβολή τις τιμής που παίρνουμε από τον ADC όταν το σήμα εισόδου μεταβάλλεται κατά 9,76 mV. Η τιμή αυτή είναι παρά πολύ ικανοποιητική για τις απαιτήσεις του συστήματος. Ο μικροελεγκτής ATmega 8 μπορεί να έχει συχνότητα χρονισμού (Clock) ως 16 MHz. Επιλέγεται να χρησιμοποιήσουμε κρύσταλλο 8 MHz γιατί αυτή η συχνότητα είναι πάρα πολύ ικανοποιητική για την εκτέλεση των εντολών του προγράμματος του μικροελεγκτή.

- Μονάδα διαίρεσης αναλογικού σήματος δια δύο.

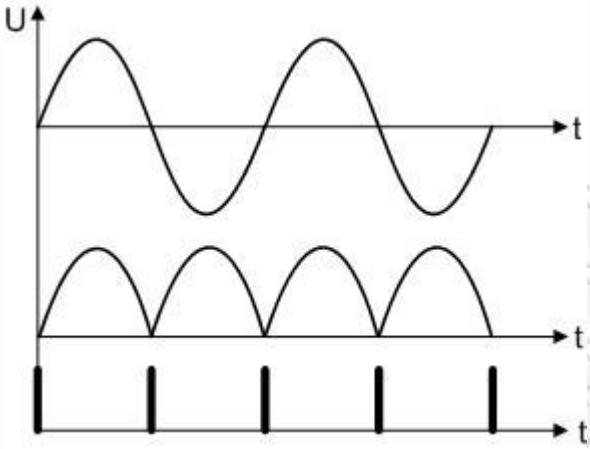
Η μονάδα διαίρεσης του αναλογικού σήματος επειδή θα δέχεται το σήμα από τον PID ελεγκτή θα πρέπει να έχει πάρα πολύ μεγάλη αντίσταση εισόδου για να μην επηρεάζει την λειτουργία του PID ελεγκτή. Έτσι δεν μπορεί να χρησιμοποιηθεί ένα πολύ απλό εξάρτημα όπως είναι ο διαιρέτης τάσης με αντιστάσεις. Ένα εξάρτημα που έχει πολύ μεγάλη αντίσταση εισόδου και εξόδου είναι ο τελεστικός ενισχυτής. Έτσι από την θεωρία των τελεστικών ενισχυτών επιλέγω το κατάλληλο κύκλωμα που κάνει διαίρεση του σήματος εισόδου χωρίς να του κάνει αναστροφή και υπολογίζω τις τιμές των περιφερειακών αντιστάσεων που χρειάζεται για να λειτουργήσει και να κάνει ακριβός διαίρεση δια δύο.

- Μονάδα ανίχνευσης διέλευσης της τάσης από το μηδέν(Zero crossing detector).

Για να μπορεί ο μικροελεγκτής να κάνει ΟΝ το λαμπτήρα στο σωστό χρονικό σημείο της ημιπεριόδου της τάσης τροφοδοσίας του θα πρέπει να έχει μια αναφορά πότε αυτή διέρχεται από το μηδέν για λόγους ασφαλείας ο ανιχνευτής διέλευσης της τάσης από το μηδέν πρέπει να έχει απομόνωση μεταξύ της υψηλής τάσης τροφοδοσίας του λαμπτήρα και των υπόλοιπων ηλεκτρονικών χαμηλής τάσης του επενεργητή. Αυτή η απομόνωση μπορεί να επιτευχθεί εύκολα και με χαμηλό κόστος με έναν οπτικό μονωτήρα(Optoisolator). Ένα εξάρτημα το οποίο χρησιμοποιεί φως για να εξασφαλίσει γαλβανική απομόνωση. Τυπικά μια φωτοδίοδος χρησιμοποιείται ως πηγή φωτός και ένα transistor ευαίσθητο στο φως δηλ. μεταβαίνει σε κατάσταση αγωγής όταν πέσει επάνω στην βάση του φως χρησιμοποιείται ως δέκτης.

IC3 είναι ο απομονωτής και είναι το 6N138 της κατασκευάστριας εταιρίας Fairchild Semiconductor. Το 6N138 έχει στο στάδιο εξόδου διπλό transistor (Darlington transistor) για υψηλό κέρδος. Εξασφαλίζοντας λόγο μεταφοράς ρεύματος (Current transfer ratio CTR) τυπικά 2000%. Αυτό σημαίνει ότι το ρεύμα εξόδου αλλάζει κατά 200 φορές του ρεύματος εισόδου. Για το λόγο αυτό το εξάρτημα αυτό είναι κατάλληλο για κατευθείαν οδήγηση εισόδων μικροελεγκτών. Τα τυπικά χαρακτηριστικά εισόδου είναι: Forward voltage V_F=1.30 V, forward current I_F=1.6mA, maximum forward current=20 mA.

Το 6N138 επειδή στην είσοδο του έχει φωτοδίοδο μπορεί να άγει μόνο κατά τις θετικές ημιπεριόδους της τάσης. Έτσι για να μπορούμε να έχουμε και τις αρνητικές θα πρέπει να τις γυρίσουμε στην θετική πλευρά αυτό το πετυχαίνουμε με τέσσερις διόδους σε σχήμα γέφυρας και με αυτό τον τρόπο τροφοδοτούμε το 6N138 μόνο με θετικές τάσης όπως φαίνεται στο παρακάτω σχήμα.



Εικόνα 52: Κυματομορφή της τάσης, ανορθωμένης τάσης και των παλμών στο σημείο μηδέν.

Η τροφοδοσία του 6N138 με όλες τις ημιπεριόδους θετικές εξασφαλίζει ότι η τροφοδοσία στο φορτίο θα είναι συμμετρικοί και στις αρνητικές και στις θετικές ημιπεριόδους.

Η μέγιστη τάση είναι 230 V * $\sqrt{2}$ =325 V. Χρησιμοποιώντας δύο αντιστάσεις σε σειρά των 15K από το νόμο του Ohm το μέγιστο ρεύμα που θα περάσει από το 6N138 είναι

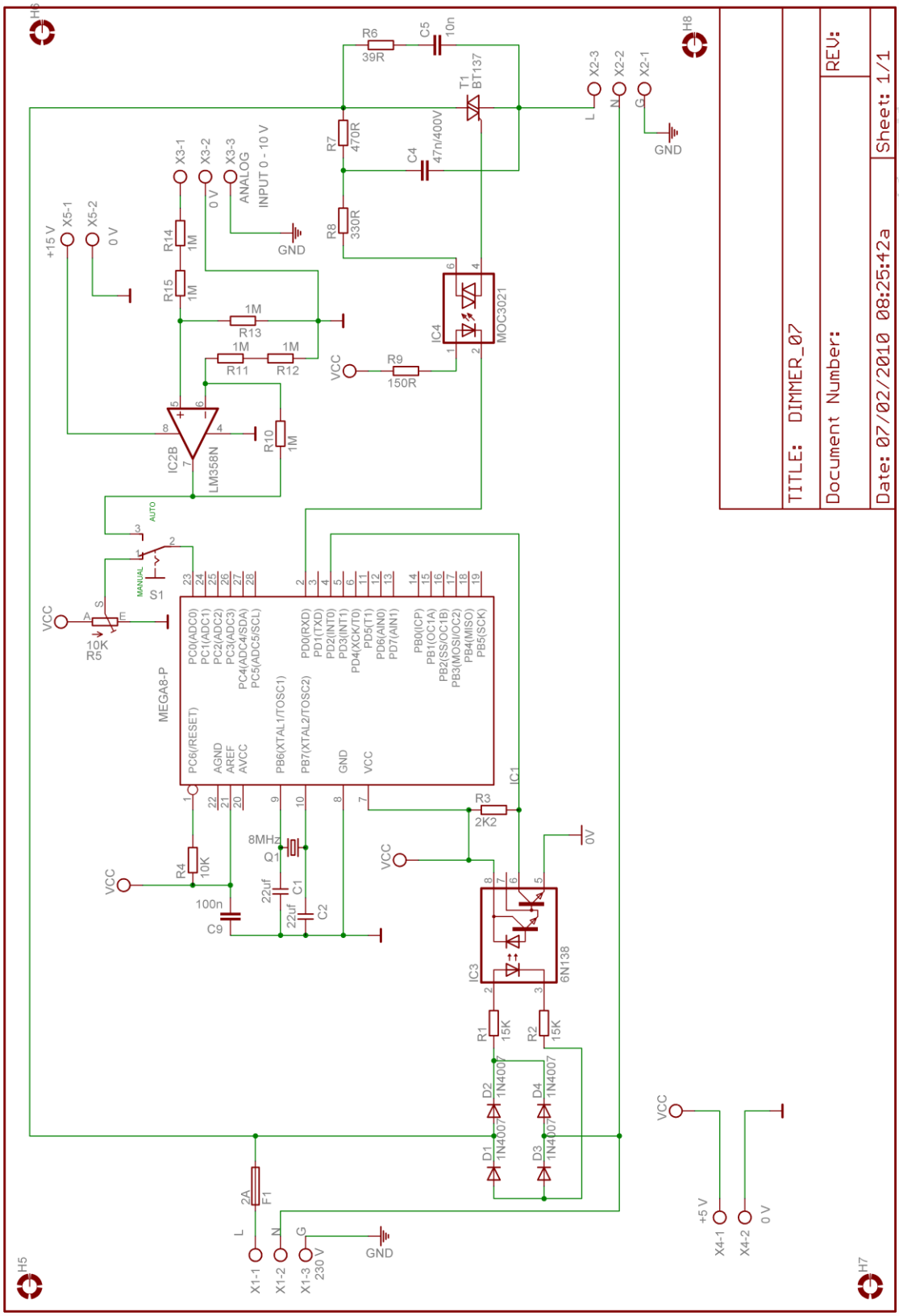
$$I = \frac{V}{R} = \frac{V - V_D}{r} = \frac{230V - 1.4V}{30000\Omega} = 0{,}00762A = 7{,}62mA$$

Όπου V_D η πτώση τάσης πάνω στις διόδους ανόρθωσης.

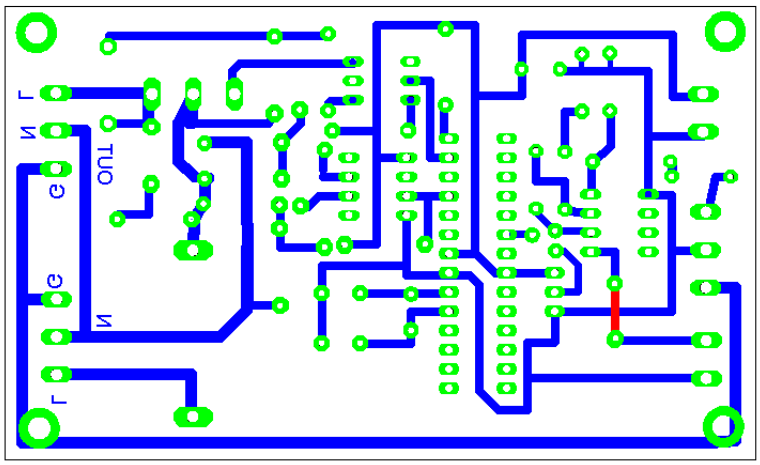
Η εναλλασσόμενη τάση στην οποία το 6N138 είναι εγγυημένο ότι θα γίνει ON είναι: V=I*R=1.6mA*30KΩ= 48V. Το σημείο αυτό στην κυματομορφή της τάσης τροφοδοσίας θα είναι Sin^{-1} (48V/230V)=12^o. Στην πράξη το 6N138 θα άγει πολύ πιο νωρίτερα αλλά και αυτό να μην συμβαίνει αν σκεφτούμε ότι η κάθε ημιπερίοδος είναι 180^o η 12^o είναι πολύ κοντά στην διέλευση της τάσης από το μηδέν.

Το ηλεκτρονικό κύκλωμα του επενεργητή καθώς και το τυπωμένο κύκλωμα (PCB) της πλακέτας σχεδιάσθηκε με το πρόγραμμα Eagle.

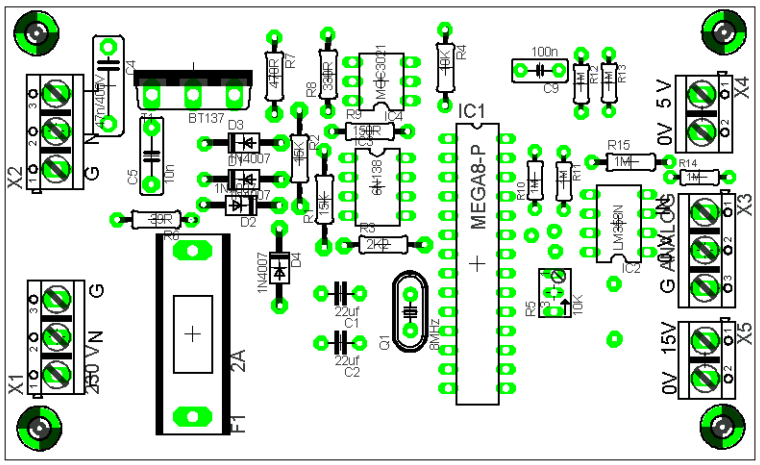
Στις παρακάτω εικόνες βλέπουμε αντίστοιχα Το σχηματικό διάγραμμα του επενεργητή, το τυπωμένο κύκλωμα του επενεργητή και την τοποθέτηση των εξαρτημάτων στην πλακέτα του επενεργητή.



Εικόνα 53: Το θεωρητικό κύκλωμα του επενεργητή.



Εικόνα 54: Το τυπωμένο κύκλωμα του επενεργητή.

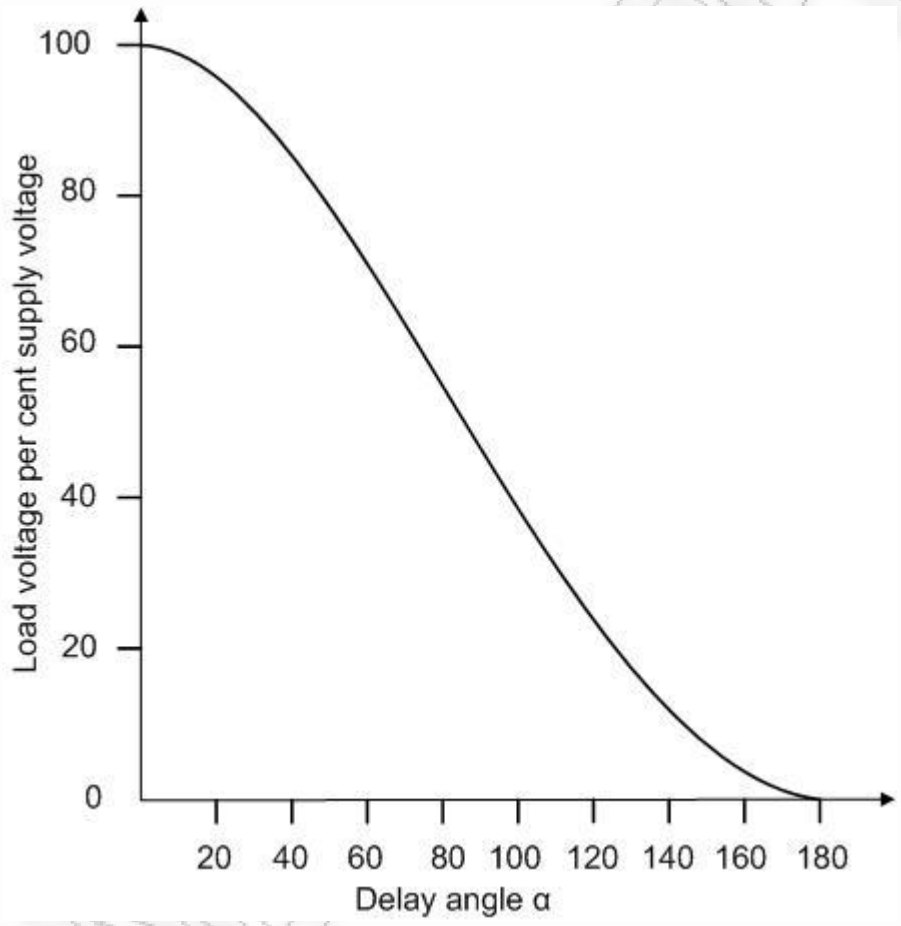


Εικόνα 55: Η Τοποθέτηση των εξαρτημάτων στην πλακέτα του επενεργητή.

5.4 Το Πρόγραμμα Του Μικροελεγκτή Για Τον επενεργητή.

Ο ελεγκτής PID βγάζει σήμα 0 – 10 V και βγάζει μεγαλύτερο σήμα όταν θέλει να δόση περισσότερη ισχύ στο φορτίο π.χ. όταν θα θέλει να δόση όλη την ισχύ στο φορτίο το σήμα που θα δόση θα είναι 10V, ενώ όταν θα θέλει να δόση μηδενική ισχύ στο φορτίο θα δόση σήμα 0V. Όπως αναφέρθηκε παραπάνω ο έλεγχος της ισχύος στο φορτίο (στο λαμπτήρα πυρακτώσεως) θα γίνει με την τεχνική του ελέγχου της γωνίας της φάσης. Αυτό σημαίνει ότι όσο μεγαλύτερο σήμα θα παίρνει από τον ελεγκτή σε τόσο μικρότερη γωνία α θα πρέπει να δίνη παλμό έναυσης στο triac.

Από την θεωρία των ηλεκτρονικών ισχύος στην παρακάτω εικόνα βλέπουμε το ποσοστό της τάσης τροφοδοσίας που θα δίνεται στο λαμπτήρα ανάλογα με την γωνία α που θα δίνουμε έναυση στο triac.



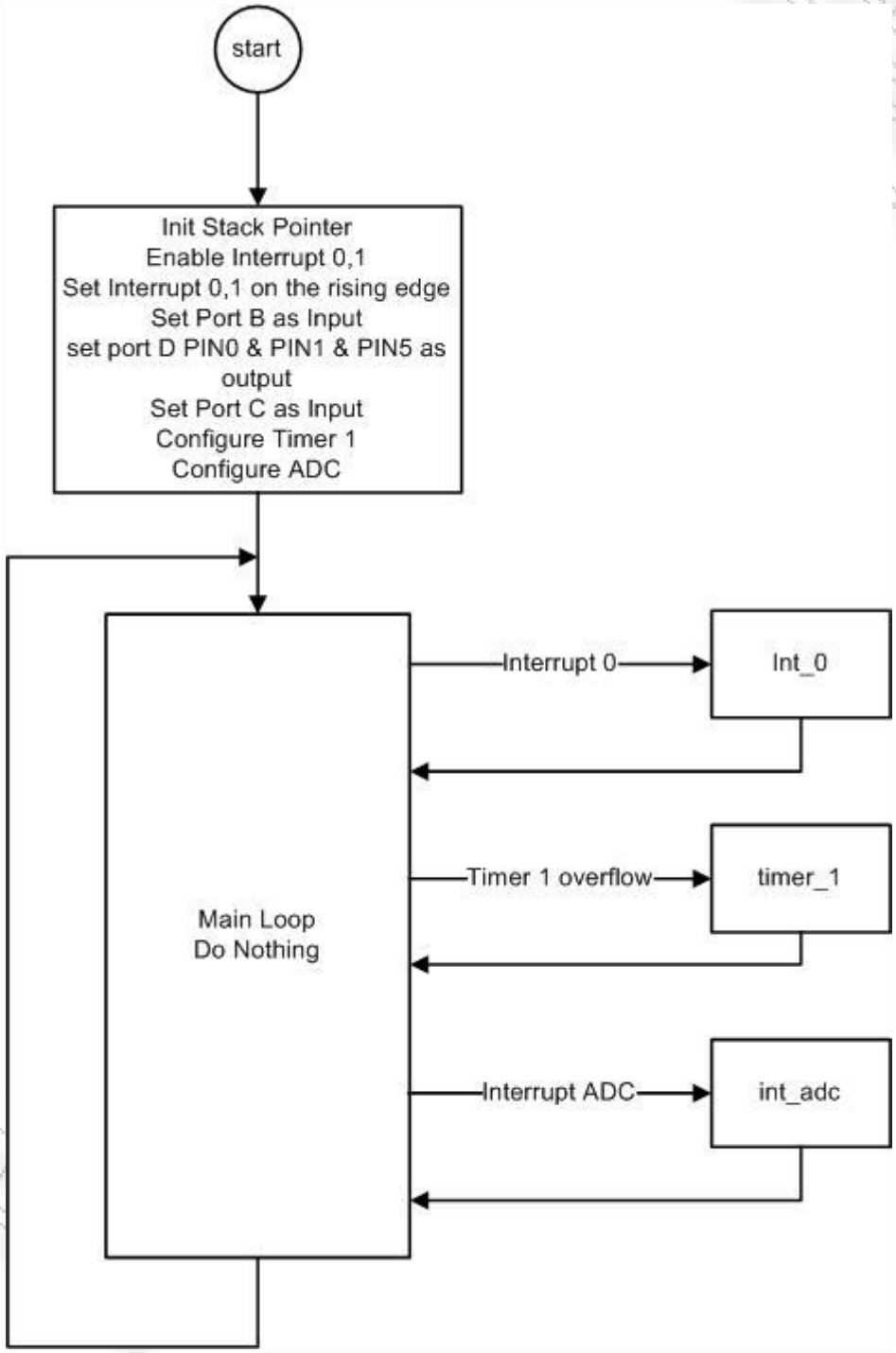
Εικόνα 56: Καμπύλη γωνίας έναυσης α και τάσης στο φορτίο

Όπως βλέπουμε η παραπάνω καμπύλη δεν είναι τελείως γραμμική. Αυτό συμβαίνει στην αρχή και στο τέλος της ημιπεριόδου της τάσης λόγο της ημιτονοειδούς μορφής που έχει και επειδή στην αρχή της ημιπεριόδου η φωτεινότητα που εκπέμπει ο λαμπτήρας είναι σχεδόν στο 100 % ενώ στο τέλος είναι σχεδόν στο 0 % αυτό δεν επηρεάζει πολύ στο να θεωρήσουμε την καμπύλη γραμμική.

Αφού λοιπών θεωρούμε την καμπύλη γραμμική αυτό σημαίνει ότι ανάλογα με το σήμα που στέλνει ο ελεγκτής PID θα πρέπει να καθυστερούμαι την γωνία έναυσης του triac. Ξέρουμε από την συχνότητα που έχει το δίκτυο παροχής τάσης τα 50 Hz ότι η ημιπερίοδος της τάσης είναι 10 msec άρα και το εύρος της γωνίας είναι 10 msec. Επομένως αυτό που πρέπει να κάνουμε είναι να αντιστοιχήσουμε το εύρος του σήματος που δίνη ο ελεγκτής PID με το εύρος των 10 msec. Δηλ. Όταν ο ελεγκτής δίνει σήμα 0 V που σημαίνει μηδενική ισχύ στο φορτίο θα πρέπει η καθυστέρηση στην έναυση του triac να είναι 10 msec. Όταν ο ελεγκτής δίνει σήμα 5V

που σημαίνει 50% ισχύ στο φορτίο θα πρέπει η καθυστέρηση στην έναυση του triac να είναι 5 msec.

Στο παρακάτω σχήμα βλέπουμε ένα μπλοκ διάγραμμα της δομής του προγράμματος του μικροελεγκτή του επενεργητή.



Εικόνα 57: Δομή του προγράμματος του μικροελεγκτή στον επενεργητή.

Όπως βλέπουμε στο παραπάνω διάγραμμα το πρόγραμμα είναι όλο δομημένο πάνω στις διακοπές (Interrupts) που διαθέτει ο μικροελεγκτής και στο κύριο μέρος του προγράμματος δεν γίνεται τίποτα παρά μόνο αναμονή για κάποια διακοπή.

Όταν ξεκινά ο μικροελεγκτής μετά από κλείσιμο ή από reset και ξεκινά η εκτέλεση του προγράμματος πρώτα εκτελείται το τμήμα του προγράμματος που κάνει τις αρχικοποιήσεις στα διάφορα τμήματα του μικροελεγκτή ώστε στην κύρια εκτέλεση του προγράμματος να κάνει ο μικροελεγκτής την λειτουργία πού θέλουμε. Οι αρχικοποιήσεις αυτές είναι η εξής:

- Θέτουμε τον stack pointer να είναι στο τέλος της περιοχής της μνήμης RAM.
- Ενεργοποιούμε τα εξωτερικά Interrupt 0 και Interrupt 1.
- Θέτουμε τα εξωτερικά Interrupt 0 και Interrupt 1 να ενεργοποιούνται στην θετική ακμή του σήματος στην είσοδο του μικροελεγκτή.
- Θέτουμε τα PIN τις πόρτας B να λειτουργούν σαν είσοδοι
- Θέτουμε τα PIN τις πόρτας C να λειτουργούν σαν είσοδοι
- Θέτουμε τα PIN 0, PIN 1 και PIN 5 της πόρτας D να λειτουργούν σαν έξοδοι και όλα τα υπόλοιπα να λειτουργούν σαν είσοδοι.
- Αρχικοποιούμε τον Timer 1 ώστε να κάνει Interrupt μετά την λήξη του χρόνου που πρέπει να μετρήσει και ο χρονισμός του να γίνεται από το ρολόι του μικροελεγκτή, το οποίο όπως είπαμε παραπάνω είναι 8 MHz και η περίοδος του είναι T=1/f= 1/8000000 = 125 nsec. Ο Timer 1 είναι 16 bit και επομένως το μέγιστο που μπορεί να μετρήσει είναι 65535 * 125 nsec =8.191 msec. Αυτός ο χρόνος δεν μας ικανοποιεί γιατί όπως είπαμε παραπάνω η μέγιστη καθυστέρηση που θέλουμε είναι τα 10 msec. Ο μικροελεγκτής όμως μας δίνει την δυνατότητα πριν οδηγηθεί το ρολόι στον timer 1 να του κάνουμε προδιαίρεση σε μια από τις επόμενες τιμές: $f_{clk}/8$, $f_{clk}/64$, $f_{clk}/256$ και $f_{clk}/1024$. αν του κάνουμε μια προδιαίρεση δια 8 αυτό μας δίνει ότι ο Timer 1 μπορεί να μετρήσει μέχρι 65,535 msec και αυτός ο χρόνος καλύπτει τις δικές μας ανάγκες.
- Αρχικοποιούμε τον μετατροπέα του αναλογικού σήματος σε ψηφιακό ώστε να κάνει την μετατροπή του σήματος και κατόπιν να προκαλεί διακοπή για να διαβάσουμε την τιμή της μετατροπής.

Μετά τις αρχικοποιήσεις θέτουμε σε λογικό 1 το pin 0 τις πόρτας D για να είναι σβηστός ο λαμπτήρας και μετά ακολουθεί το κυρίως πρόγραμμα που είναι η ρουτίνα MAIN.

MAIN: είναι το κυρίως πρόγραμμα το οποίο δεν κάνει καμία λειτουργία αλλά είναι ένας ατελείωτος βρόχος από τον οποίο όταν γίνεται κάποια διακοπή η εκτέλεση του προγράμματος μεταβαίνει στην αντίστοιχη ρουτίνα την εκτελεί και επιστρέφει πίσω. Η όλη λειτουργία είναι στα Interrupts τα οποία είναι τα εξής:

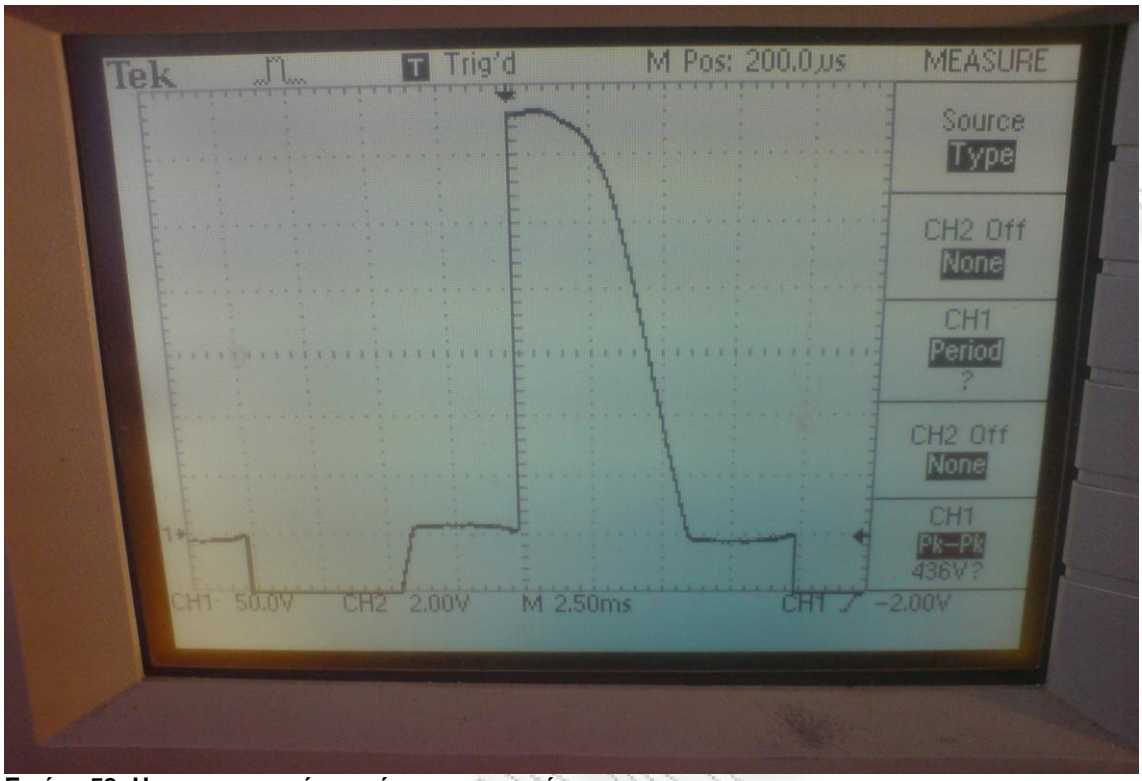
- Interrupt 0: το Interrupt αυτό γίνεται από το κύκλωμα ανίχνευσης της τάσης από το μηδέν για να ξέρουμε το χρονικό σημείο στο οποίο η τάση είναι στο μηδέν. Όταν ενεργοποιείται καλείται η ρουτίνα int_0. Στην ρουτίνα αυτή διαβάζουμε συνεχώς το σήμα που προκαλεί το Interrupt για κάποιο χρόνο ώστε να βεβαιώσουμε ότι δεν πρόκειται για κάποιο spike στην είσοδο του μικροελεγκτή. Αν πρόκειται για λάθος σήμα βγαίνουμε από την ρουτίνα. Αν πρόκειται για το σωστό σήμα διαβάζουμε το σήμα από τον ελεγκτή PID δηλ. Την τιμή που έχουμε αποθηκεύσει στην μνήμη RAM από τον ADC και υπολογίζουμε τον χρόνο που πρέπει να μετρήσει ο timer 1 από την σχέση T=10230 – $10*ADC_{value}$ που είναι ο χρόνος για την έναυση του triac. Μετά τον υπολογισμό του χρόνου φορτώνουμε την τιμή στους καταχωρητές του timer 1 και ενεργοποιούμε το interrupt του Timer 1.
- Interrupt ADC: Το Interrupt αυτό γίνεται από τον μετατροπέα του αναλογικού σήματος σε ψηφιακό μετά από ολοκλήρωση της μετατροπής για να διαβάσουμε την τιμή της μετατροπής. Όταν ενεργοποιείται καλείται η ρουτίνα int_adc. Στην ρουτίνα αυτή διαβάζουμε τους καταχωρητές που περιέχουν το αποτέλεσμα της μετατροπής και το αποθηκεύουμε στη μνήμη RAM το low byte στην διεύθυνση $100 και το high byte στην διεύθυνση $101 και κατόπιν αρχικοποιούμε τον ADC να ξεκινήσει νέα μετατροπή.

- Timer 1 Overflow: το Interrupt αυτό γίνεται από τον timer 1 που διαθέτη ο μικροελεγκτής όταν περάσει ο χρόνος που του έχουμε βάλει για να μετρήσει. Όταν ενεργοποιείται καλείται η ρουτίνα timer_1. Στην ρουτίνα αυτή κάνουμε stop τον Timer 1 και οδηγούμαι το PD0 σε λογικό μηδέν γα να γίνει η πυροδότηση στο Triac.

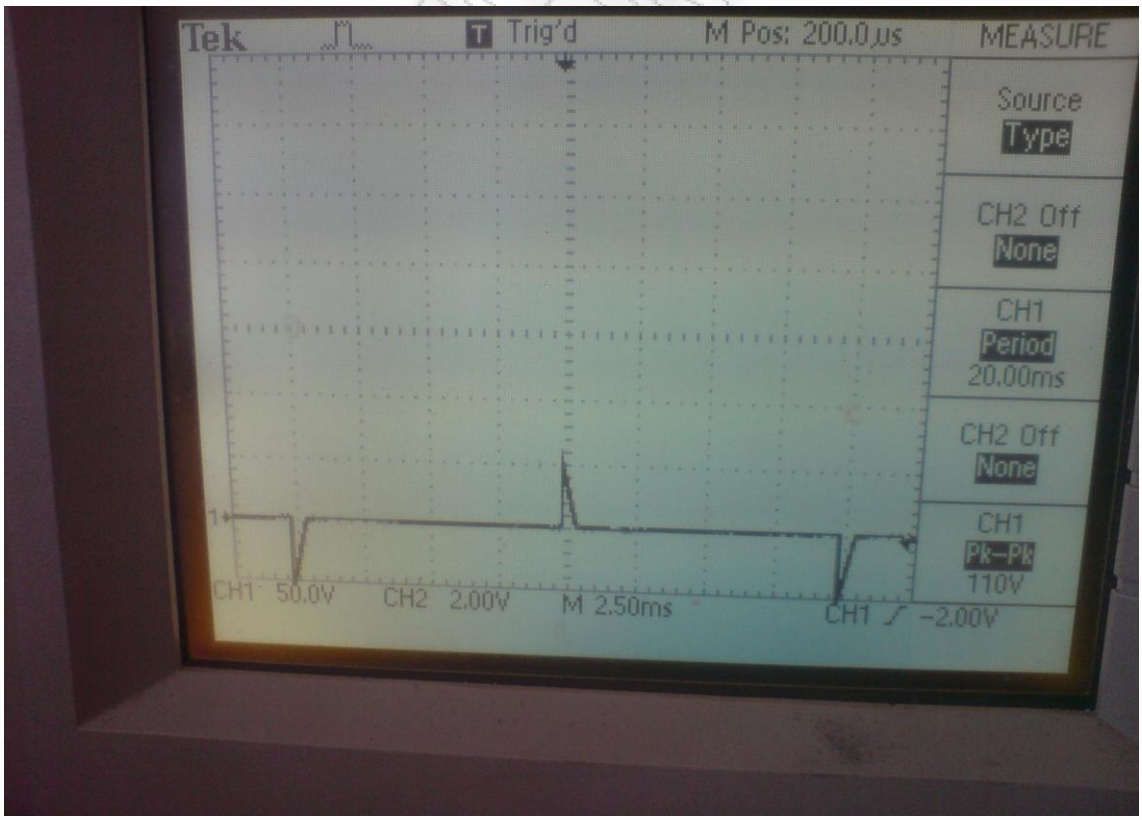
Ο κώδικας του προγράμματος βρίσκεται στο παράρτημα Α.

Μετά τον προγραμματισμό του μικροελεγκτή το κύκλωμα του επενεργητή τέθηκε σε λειτουργία για δοκιμές σε λειτουργία ανοικτού βρόχου αφού δεν έχει κατασκευαστή ακόμη ο ελεγκτής PID. Μετά τις πρώτες δοκιμές και την αποσφαλμάτωση του προγράμματος ο επενεργητής λειτούργησε κανονικά. Στις παρακάτω εικόνες βλέπουμε τις κυματομορφές που πάρθηκαν με παλμογράφο από την τροφοδοσία του λαμπτήρα και τους παλμούς που δίνει το κύκλωμα ανίχνευσης διέλευσης τις τάσης από το μηδέν.

Εικόνα 58: Οι παλμοί από το κύκλωμα ανίχνευσης διέλευσης της τάσης από το μηδέν.



Εικόνα 59: Η κυματομορφή της τάσης στο φορτίο



Εικόνα 60: Η κυματομορφή της τάσης στο φορτίο

6 ΚΕΦΑΛΑΙΟ 6

Σχεδίαση – Υλοποίηση Του PID Ελεγκτή

6.1 Εισαγωγή

Στο κεφάλαιο αυτό εξηγείται με λεπτομέρειες η διαδικασία που ακολουθήθηκε κατά το σχεδιασμό του ελεγκτή PID για τον έλεγχο του φωτισμού με λαμπτήρα πυρακτώσεως καθώς και για την κατασκευή ως την λειτουργία και τις τελικές δοκιμές.

Το σύστημα αυτομάτου ελέγχου που σχεδιάσθηκε και υλοποιήθηκε φαίνεται στην Εικόνα 46 του κεφαλαίου 5. Στο μπλοκ διάγραμμα του συστήματος το φυσικό σύστημα (Διεργασία) που θα ελέγξουμε είναι ο ηλεκτρικός λαμπτήρας πυρακτώσεως και η μεταβλητή ελέγχου είναι η φωτεινή ροή του λαμπτήρα. Ο ελεγκτής PID θα εκτελεί τον αλγόριθμο τριών όρων σύμφωνα με την επιθυμητή τιμή που θα παίρνει. Για την εκτέλεση του αλγορίθμου θα χρησιμοποιηθεί ένας μικροελεγκτής επομένως ο ελεγκτής θα είναι ένας ψηφιακός ελεγκτής PID.

Η επιθυμητή τιμή θα δίνεται ή από υπολογιστή μέσω σειριακού interface που θα διαθέτει ή από ποτενσιόμετρο σε ποσοστό από 0% ως 100% της φωτεινής ροής που μπορεί να δώσει ο λαμπτήρας πυρακτώσεως. Για την λειτουργία του ο ελεγκτής χρειάζεται ανατροφοδότηση ώστε να υπολογίζει το σφάλμα της πραγματικής τιμής με την επιθυμητή για να βγάζει την κατάλληλη έξοδο. Η βαθμίδα ανατροφοδότησης θα μετατρέπει την φωτεινή ροή σε ηλεκτρικό σήμα το οποίο μπορεί να διαβάσει ο μικροελεγκτής για την εκτέλεση του αλγορίθμου.

6.2 Σχεδίαση Του Ελεγκτή PID.

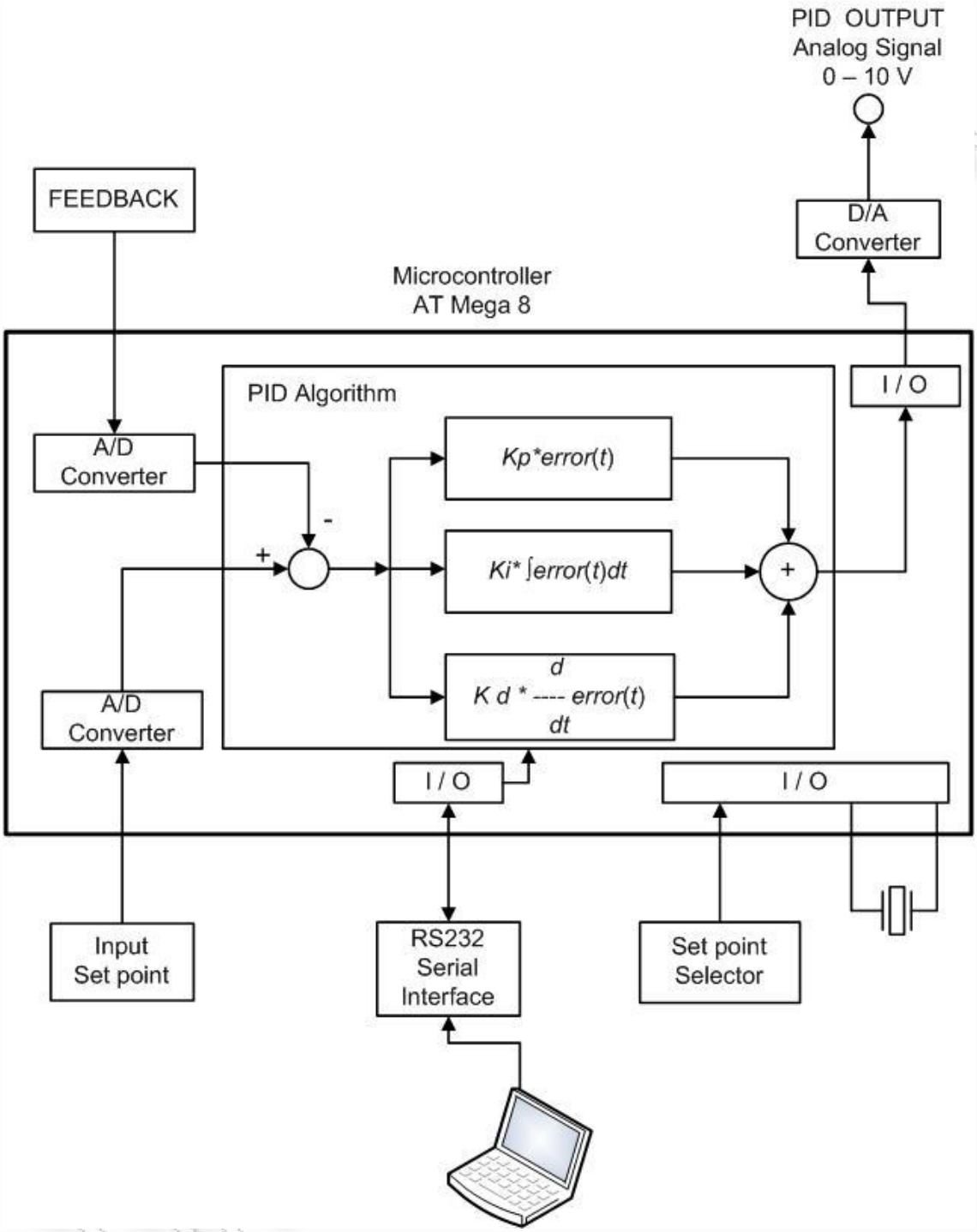
Όπως αναφέρθηκε και παραπάνω ο ελεγκτής PID θα είναι ένας ψηφιακός ελεγκτής αυτό σημαίνει ότι θα υπάρχει μια κεντρική μονάδα επεξεργασίας ή οποία θα εκτελεί τον αλγόριθμο του ελεγκτή και θα υπολογίζει την έξοδο που θα βγάζει ο ελεγκτής. Εφόσον η έξοδος προέρχεται από την κεντρική μονάδα επεξεργασίας είναι σε ψηφιακή μορφή και για να γίνει η έξοδος αναλογικό σήμα 0 – 10V θα πρέπει να παρεμβληθεί μια βαθμίδα μετατροπής του ψηφιακού σήματος σε αναλογικό. Για την εκτέλεση του αλγορίθμου πρέπει να υπάρχει η μονάδα ανατροφοδότησης ή οποία μετρά την ελεγχόμενη μεταβλητή και μετατρέπει το σήμα σε ηλεκτρικό το οποίο περνώντας από μια μονάδα μετατροπής αναλογικού σήματος σε ψηφιακό μπορεί να διαβαστεί από την κεντρική μονάδα επεξεργασίας.

Η παραμετροποίηση του ελεγκτή θα γίνεται από ηλεκτρονικό υπολογιστή και για την επικοινωνία του ελεγκτή με τον υπολογιστή θα υπάρχει ένα σειριακό interface. Η επιθυμητή τιμή θα μπορεί να τίθεται ή από τον υπολογιστή ή τοπικά από τον ελεγκτή σε ποσοστό επί τις εκατό της φωτεινής ροής. Για να ξέρει ή κεντρική μονάδα επεξεργασίας από που θα παίρνει την επιθυμητή τιμή θα υπάρχει μια μονάδα επιλογής επιθυμητής τιμής. Επομένως οι διάφορες μονάδες που πρέπει να έχει ο ελεγκτής είναι οι παρακάτω:

- Κεντρική μονάδα επεξεργασίας.
- Μονάδα μετατροπής ψηφιακού σήματος σε αναλογικό.
- Μονάδα ανατροφοδότησης.
- Μονάδα μετατροπής του αναλογικού σήματος της ανατροφοδότησης σε ψηφιακό.
- Μονάδα επιλογής επιθυμητής τιμής τοπικά ή από PC.
- Μονάδα εισαγωγής επιθυμητής τιμής τοπικά από τον ελεγκτή.
- Μονάδα μετατροπής αναλογικού σήματος σε ψηφιακό για την επιθυμητή τιμή.
- Μονάδα σειριακού Interface RS232.

Το μπλοκ διάγραμμα του ελεγκτή PID με τις διάφορες βαθμίδες φαίνεται στην παρακάτω

Εικόνα **61**.



Εικόνα 61: Μπλοκ διάγραμμα των μονάδων του ελεγκτή PID.

6.3 Επιλογή Εξαρτημάτων Και Σχεδίαση Του Κυκλώματος.

Στην παράγραφο αυτή θα αναληθή η επιλογή των εξαρτημάτων για κάθε μία από τις μονάδες του παραπάνω μπλοκ διαγράμματος του ελεγκτή PID και θα γίνει η σχεδίαση του σχηματικού κυκλώματος και τις πλακέτας δηλ. του PCB.

- Κεντρική μονάδα επεξεργασίας.

Όπως και στην περίπτωση του επενεργητή έτσι και εδώ για κεντρική μονάδα επεξεργασίας επιλέγεται ο μικροελεγκτής των 8 bit ATmega 8 της εταιρίας ATMEL. Η συγκεκριμένη επιλογή γίνεται γιατί διαθέτει 23 προγραμματιζόμενες γραμμές εισόδου - εξόδου, τρεις timers έναν 16 bit και δύο 8 bit που χρειάζονται στην εκτέλεση του αλγόριθμου. Έχει 8 Kbyte μνήμη Flash για πρόγραμμα, 1 Kbyte μνήμη SRAM για δεδομένα και 512 byte μνήμη EEPROM. Επιπλέον διαθέτει έξη ενσωματωμένους μετατροπής αναλογικού σήματος σε ψηφιακό με ακρίβεια 10 bit και μέγιστη τάση εισόδου 5V. Έτσι με τον συγκεκριμένο μικροελεγκτή θα μπορέσουμε να ενσωματώσουμε σε αυτόν και τις βαθμίδες μετατροπής από αναλογικό σήμα σε ψηφιακό που χρειάζονται για να μπορεί η κεντρική μονάδα επεξεργασίας να διαβάζει την ανατροφοδότηση και την επιθυμητή τιμή όταν αυτή δεν τίθεται από υπολογιστή.

Το μέγιστο σήμα που μπορεί να δεχθεί ο ADC είναι τα 5V και επειδή η ανάλυση του ADC είναι 10 bit η αλλαγή της ψηφιακής τιμής θα είναι: 5 V/1024 =4,88 mV άρα θα έχουμε μεταβολή τις τιμής που παίρνουμε από τον ADC όταν το σήμα εισόδου μεταβάλλεται κατά 4,88 mV. Η τιμή αυτή είναι παρά πολύ ικανοποιητική για τις απαιτήσεις του συστήματος. Ο μικροελεγκτής ATmega8 μπορεί να έχει συχνότητα χρονισμού (Clock) ως 16 MHz. Επιλέγεται να χρησιμοποιήσουμε κρύσταλλο 8 MHz γιατί αυτή η συχνότητα είναι πάρα πολύ ικανοποιητική για την εκτέλεση των εντολών του προγράμματος του μικροελεγκτή.

- Μονάδα μετατροπής ψηφιακού σήματος σε αναλογικό.

Όπως αναφέρθηκε παραπάνω ο αλγόριθμος εκτελείται από την κεντρική μονάδα επεξεργασίας πού είναι ψηφιακή. Επομένως και το αποτέλεσμα που θα βγάλει για έξοδο θα είναι σε ψηφιακή μορφή. Για να πάρουμε το αναλογικό σήμα 0 – 10V που θέλουμε στην έξοδο θα παρεμβάλουμε ένα μετατροπέα από ψηφιακό σήμα σε αναλογικό (DAC). Στην αγορά υπάρχουν πολλοί μετατροπείς με διάφορα χαρακτηρίστηκα. Όπως παράλληλοι, σειριακοί, με ακρίβεια μετατροπής από 8bit ως 16bit και ο χρόνος μετατροπής κυμαίνεται σε διάφορες τιμές καθώς και το κόστος τους. Επειδή στην εφαρμογή μας δεν θέλουμε ούτε γρήγορους χρόνους ούτε μεγάλη ακρίβεια επιλέγουμε έναν μετατροπέα τον 8bit που είναι ο DAC0830 και είναι παράλληλος για απλότητα στην σχεδίαση του κυκλώματος.

- Μονάδα ανατροφοδότησης.

Η μονάδα ανατροφοδότησης είναι η μονάδα που μετατρέπει την ελεγχόμενη μεταβλητή σε σήμα άλλης μορφής συνήθως ηλεκτρικό ώστε να μπορεί να μετρηθεί. Έτσι και εδώ η μονάδα ανατροφοδότησης θα μετατρέψει την φωτεινή ροή σε ηλεκτρικό σήμα. Ένα εξάρτημα που μπορεί να κάνει αυτήν την μετατροπή είναι η φωτοαντίσταση ή οποία αλλάζει την τιμή της ανάλογα με την ένταση του φωτός που προσπίπτει πάνω της. Και χρησιμοποιώντας την με κατάλληλο κύκλωμα παίρνουμε το ηλεκτρικό σήμα. Στην Εικόνα 62 βλέπουμε την καμπύλη της φωτοαντίστασης σε συνάρτηση με το μήκος κύματος λ.

Εικόνα 62: Καμπύλη ιδανικής και πραγματικής φωτοαντίστασης.

Η σχέση μετατροπής όπως φαίνεται από την παραπάνω Εικόνα 62 δεν είναι τελείως γραμμική σε όλο το μήκος κύματος στην πραγματική φωτοαντίσταση. Η μη γραμμικότητα συμβαίνει στην αρχή και λίγο πριν το οριακό μήκος κύματος. Στην αγορά υπάρχουν αισθητήρια φωτός τα οποία είναι γραμμικά σε όλο το μήκος κύματος και παράγουν αναλογικό σήμα 0 - 10V αλλά είναι πολύ ακριβά. Η μη γραμμική περιοχή τις φωτοαντίστασης στα σημεία που είναι δεν θα επηρεάζει πάρα πολύ το σύστημα μας και σε σχέση με το πολύ χαμηλό κόστος πού έχει μας κάνει και επιλέγουμε να χρησιμοποιήσουμε την φωτοαντίσταση σαν μονάδα ανατροφοδότησης του κλειστού βρόχου του συστήματος.

- Μονάδα επιλογής επιθυμητής τιμής τοπικά ή από PC.

Την επιθυμητή τιμή θα μπορούμε να την δώσουμε ή από ηλεκτρονικό υπολογιστή μέσω του σειριακού interface ή τοπικά στον ελεγκτή από την μονάδα εισόδου επιθυμητής τιμής που διαθέτει. Έτσι η κεντρική μονάδα επεξεργασίας πρέπει να ξέρει από που εμείς θέλουμε να παίρνει την επιθυμητή τιμή για την εκτέλεση του αλγορίθμου.

Σαν μονάδα επιλογής επιθυμητής τιμής επιλέχθηκε να είναι ένας επιλογικός διακόπτης δύο θέσεων ο οποίος στην μια θέση δίνει σήμα στην κεντρική μονάδα επεξεργασίας να χρησιμοποιήσει την τιμή που διάβασε τελευταία από το PC και στην άλλη θέση να διαβάσει το σήμα από την μονάδα εισόδου επιθυμητής τιμής.

- Μονάδα εισαγωγής επιθυμητής τιμής τοπικά από τον ελεγκτή.

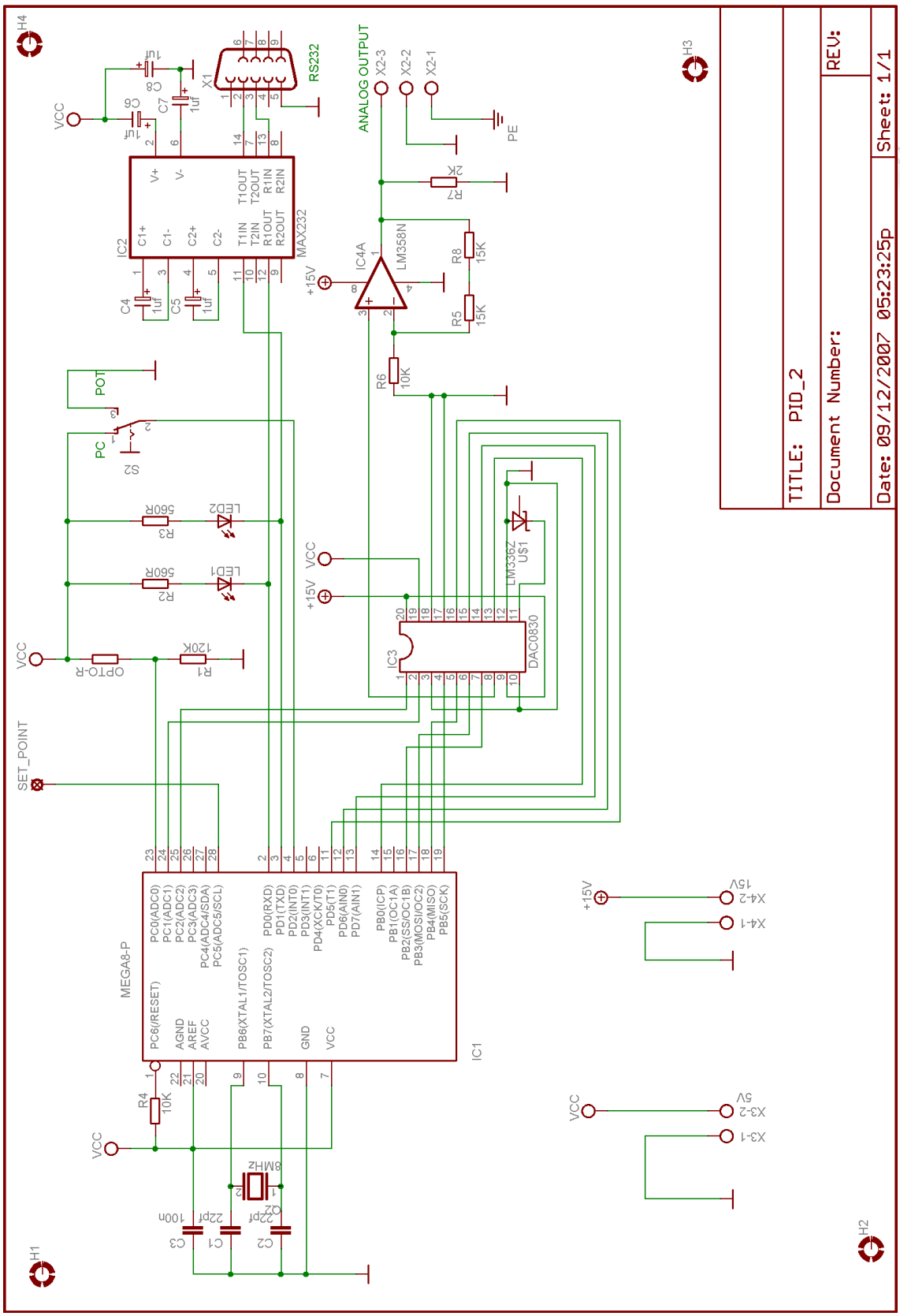
Για την είσοδο της επιθυμητής τιμής στην περίπτωση που θέλουμε να αλλάξουμε την τιμή της και δεν έχουμε συνδεμένο PC στον ελεγκτή επιλέγεται ένα ποτενσιόμετρο το οποίο θα τροφοδοτείτε με την τάση τροφοδοσίας του μικροελεγκτή δηλ. τα 5V και αυτό θα τροφοδοτεί τον μετατροπέα ADC του μικροελεγκτή. Από το ποτενσιόμετρο θα δίνουμε την επιθυμητή τιμή σαν ποσοστό επί τις εκατό τις φωτεινής έντασης που θα θέλουμε να έχουμε. Δηλ. το ποτενσιόμετρο θα δίνει σήμα από 0 – 5V το οποίο αυτό θα σημαίνει για τον ελεγκτή 0% - 100% της εντάσεως που μπορεί να δώσει ο λαμπτήρας.

- Μονάδα σειριακού Interface RS232.

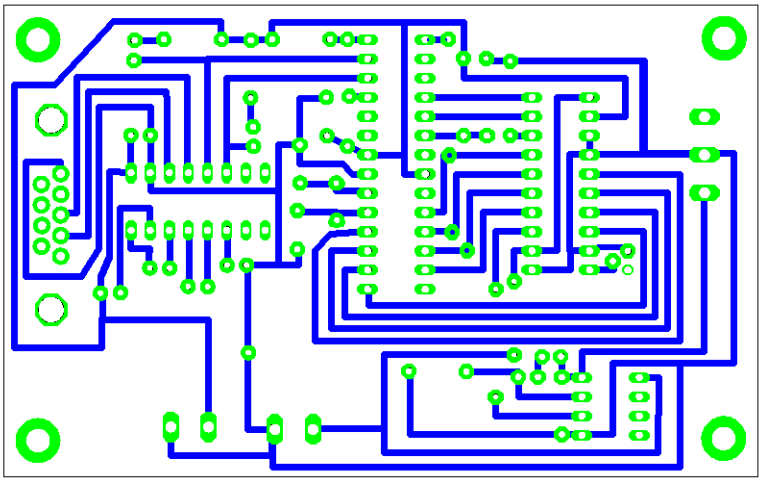
Όπως αναφέρθηκε και παραπάνω η κεντρική μονάδα επεξεργασίας του ελεγκτή θα πρέπει να μπορεί να επικοινωνήσει με ηλεκτρονικό υπολογιστή για την ρύθμιση των παραμέτρων του PID, για την είσοδο επιθυμητής τιμής κ.λ.π. για την επικοινωνία μεταξύ δύο συσκευών υπάρχουν διάφοροι τρόποι και πρωτόκολλα όπως είναι η σειριακή επικοινωνία RS232C, I^2C, USB, ETHERNET κ.λ.π. και ανάλογα με τον όγκο τον δεδομένων που θέλουμε να μεταδίδουν οι συσκευές και την ταχύτητα που πρέπει να γίνεται η μετάδοση επιλέγουμε τον τύπο επικοινωνίας. Στην εφαρμογή που θα κάνουμε ο όγκος των δεδομένων που θέλουμε να μεταδώσουμε είναι πολύ μικρός και η ταχύτητα μετάδοσης δεν είναι κρίσιμη. Έτσι επιλέγουμε να κάνουμε σειριακή επικοινωνία RS232. Ένα ακόμη πλεονέκτημα που έχουμε με αυτόν τον τύπο επικοινωνίας είναι το γεγονός ότι ο ίδιος ο μικροελεγκτής διαθέτει σύγχρονο και ασύγχρονο σειριακό μεταδότη – παραλήπτη (USART). Το USART του μικροελεγκτή λειτουργεί με στάθμες TTL επομένως θα πρέπει αυτές να τις κάνουμε σε στάθμες RS232 για να μπορεί να γίνει η επικοινωνία με τον υπολογιστή. Ένα εξάρτημα που κάνει μετατροπή στάθμης από RS232 σε TTL είναι το ολοκληρωμένο MAX232 το οποίο για να λειτουργήσει χρειάζεται μόνο κάποιους εξωτερικούς πυκνωτές.

Το ηλεκτρονικό κύκλωμα του ελεγκτή PID καθώς και το τυπωμένο κύκλωμα (PCB) της πλακέτας όπως και στην περίπτωση του επενεργητή σχεδιάσθηκε με το πρόγραμμα Eagle.

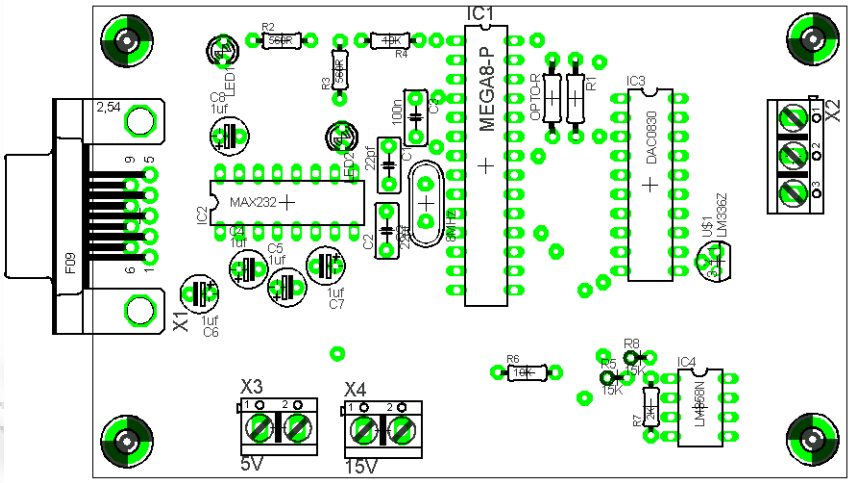
Στις παρακάτω εικόνες βλέπουμε αντίστοιχα Το σχηματικό διάγραμμα του ελεγκτή, το τυπωμένο κύκλωμα και την τοποθέτηση των εξαρτημάτων στην πλακέτα του ελεγκτή.



Εικόνα 63: Το θεωρητικό κύκλωμα του ελεγκτή PID.



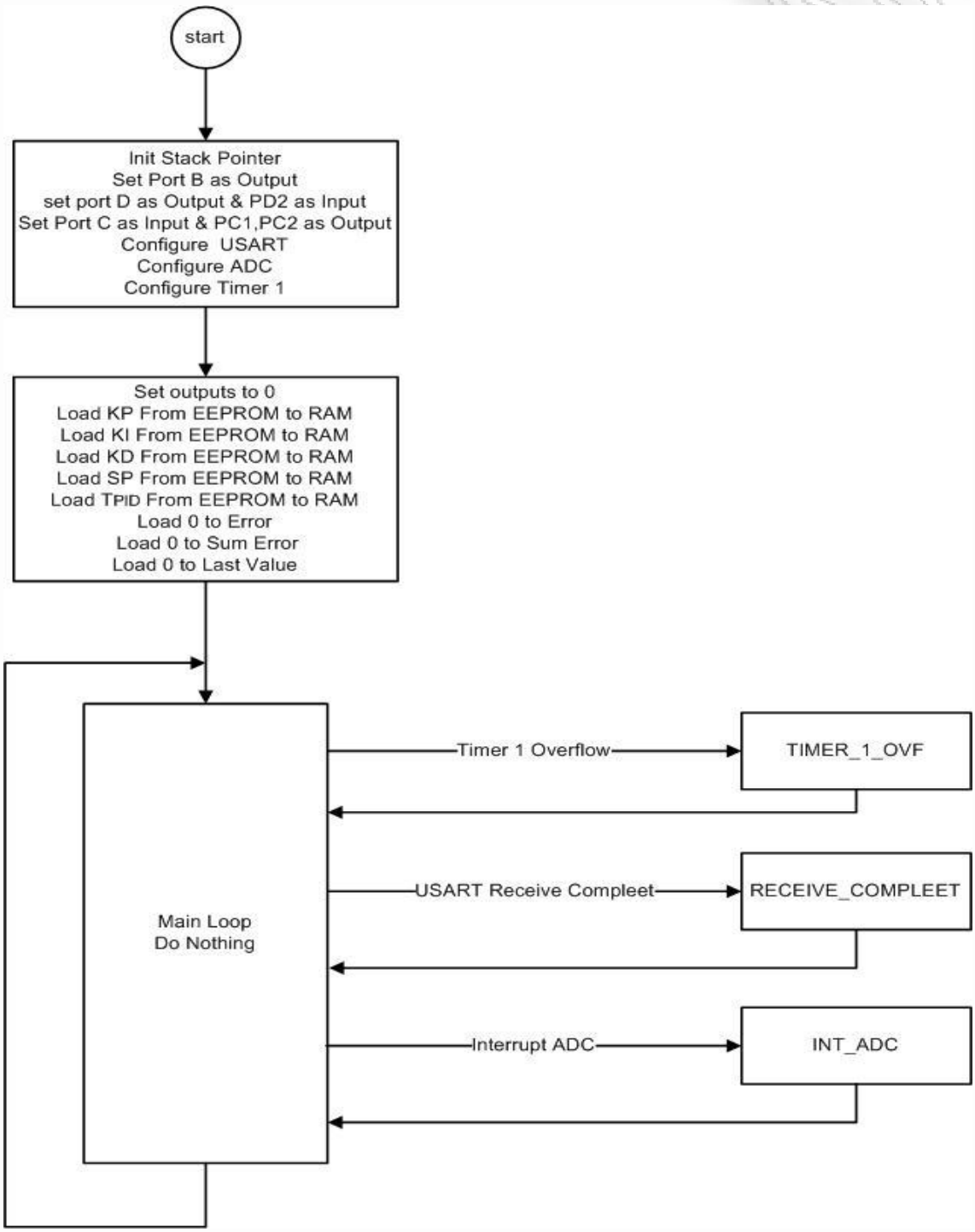
Εικόνα 64: Το τυπωμένο κύκλωμα του ελεγκτή PID.



Εικόνα 65: Η Τοποθέτηση των εξαρτημάτων στην πλακέτα του ελεγκτή PID.

6.4 Το Πρόγραμμα Του Μικροελεγκτή Για Τον PID Ελεγκτή.

Όπως και στην περίπτωση του επενεργητή το πρόγραμμα είναι όλο δομημένο πάνω στις διακοπές (Interrupts) που διαθέτει ο μικροελεγκτής. Στο κύριο μέρος του προγράμματος δεν γίνεται τίποτα παρά μόνο αναμονή για κάποια διακοπή. Ή διακοπές που γίνονται είναι από τον Timer 1 μετά από Overflow, από τον ADC μετά την ολοκλήρωση της μετατροπής κάποιου καναλιού, και από το USART μετά την ολοκλήρωση της λήψης ενός Byte. Το διάγραμμα ροής του προγράμματος του μικροελεγκτή φαίνεται στην παρακάτω εικόνα.



Εικόνα 66: Το διάγραμμα ροής του προγράμματος του ελεγκτή PID.

Όταν ξεκινά ο μικροελεγκτής μετά από κλείσιμο ή από reset και ξεκινά η εκτέλεση του προγράμματος πρώτα εκτελείται το τμήμα του προγράμματος που κάνει τις αρχικοποιήσεις στα διάφορα τμήματα του μικροελεγκτή ώστε στην κύρια εκτέλεση του προγράμματος να κάνει ο μικροελεγκτής την λειτουργία πού θέλουμε. Οι αρχικοποιήσεις αυτές είναι η εξής:

- Θέτουμε τον stack pointer να είναι στο τέλος της περιοχής της μνήμης RAM.
- Θέτουμε τα PIN τις πόρτας B να λειτουργούν σαν έξοδοι.
- Θέτουμε τα PIN τις πόρτας D να λειτουργούν σαν έξοδοι και το Pin 2 σαν είσοδος.
- Θέτουμε τα PIN τις πόρτας C να λειτουργούν σαν είσοδοι και τα Pin 1 &2 σαν έξοδοι.
- Καθορίζουμε την λειτουργία του USART για την σειριακή επικοινωνία.
- Αρχικοποιούμε τον μετατροπέα του αναλογικού σήματος σε ψηφιακό ώστε να κάνει την μετατροπή του σήματος και κατόπιν να προκαλεί διακοπή για να διαβάσουμε την τιμή της μετατροπής.
- Αρχικοποιούμε τον Timer 1 ώστε να προκαλεί διακοπή μετά την λήξη του χρόνου που πρέπει να μετρήσει και ο χρονισμός του να γίνεται από το εξωτερικό ρολόι χρονισμού του μικροελεγκτή.

Στο σημείο αυτό τελειώνει το τμήμα των αρχικοποιήσεων του μικροελεγκτή και ξεκινά η κανονική εκτέλεση του προγράμματος. Για την εκτέλεση του αλγορίθμου ο ελεγκτής χρειάζεται τις παραμέτρους του PID οι οποίες για να μην χάνονται όταν σβήνει ο μικροελεγκτής της αποθηκεύουμε στην μνήμη EEPROM που διαθέτει ο μικροελεγκτής. Επειδή το διάβασμα και η εγγραφή στην μνήμη EEPROM παίρνει αρκετούς κύκλους Κατά την εκτέλεση του προγράμματος οι τιμές αυτές θα πρέπει να είναι στην μνήμη RAM που η CPU έχει γρήγορη πρόσβαση. Έτσι μετά το τμήμα τον αρχικοποιήσεων έχουμε ένα τμήμα όπου διαβάζουμε τις τιμές των παραμέτρων από την EEPROM και τις αποθηκεύουμε στην μνήμη RAM.

Ο καθορισμός των διευθύνσεων των παραμέτρων και για την μνήμη EEPROM και για την μνήμη RAM έχει γίνει σε ένα ξεχωριστό αρχείο το ram_def.inc. Έτσι αφού ξέρουμε σε ποια διεύθυνση της μνήμης EEPROM είναι αποθηκευμένη η κάθε παράμετρος και σε ποια διεύθυνση της μνήμης RAM πρέπει να αποθηκευθεί εκτελούμε την ανάγνωση από την EEPROM και την εγγραφή στη μνήμη RAM.

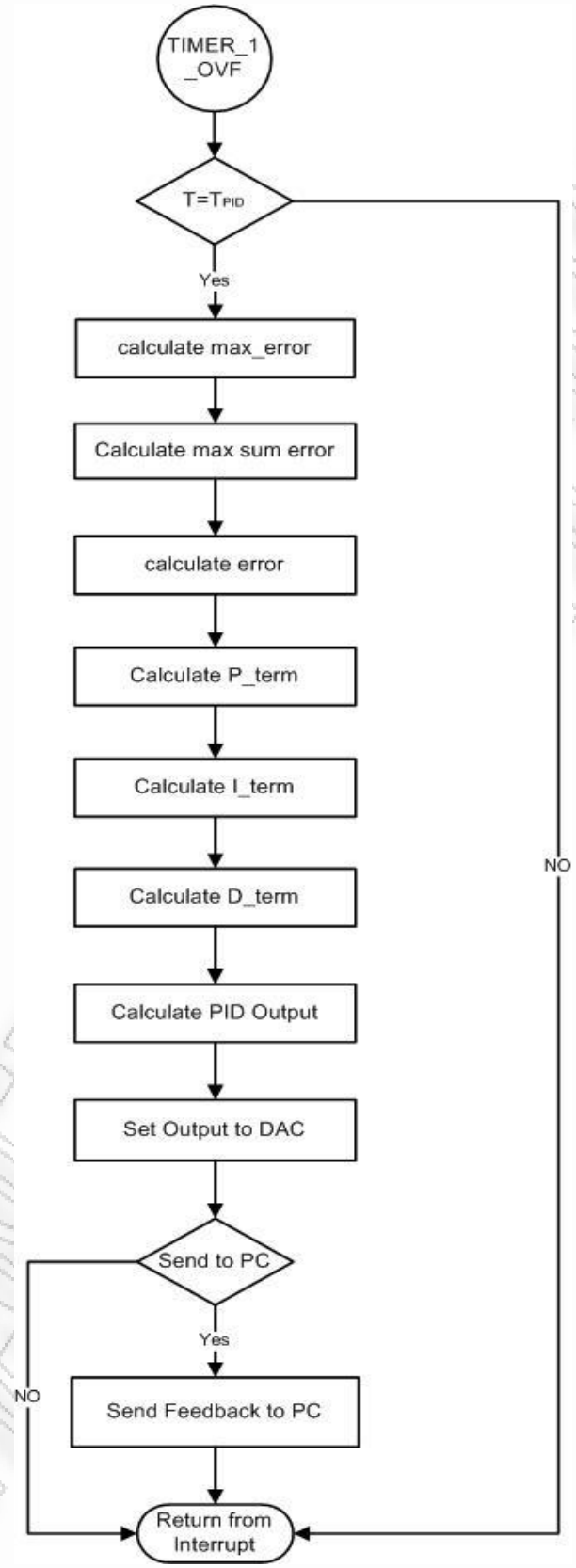
Μετά την εγγραφή των παραμέτρων στην μνήμη RAM μηδενίζουμε τις μεταβλητές Error, Sum Error και Last value. Η επεξήγηση αυτών των μεταβλητών θα γίνει παρακάτω όπου αναλύεται ο αλγόριθμος του PID ελεγκτή.

Μετά το μηδενισμό των μεταβλητών στο πρόγραμμα ακολουθεί ο κύριος βρόχος όπου αυτός είναι ένας ατελείωτος βρόχος αλλά και εντός του βρόχου αυτού δεν εκτελείται κάποια λειτουργία από τον μικροελεγκτή παρά μόνο γίνεται αναμονή για κάποια από τις διακοπές. Οι διακοπές που μπορεί να γίνουν είναι διακοπή από τον Timer 1 όταν περάσει ο χρόνος των 10 msec που του βάζουμε να μετρήσει. Στην διακοπή αυτή εκτελείται ο αλγόριθμος του PID και βγάζουμε το αποτέλεσμα του αλγορίθμου στις εξόδους που οδηγούνται στον μετατροπέα ψηφιακού σήματος σε αναλογικό και εν συνεχεία οδηγούμε τα σήματα ελέγχου του μετατροπέα για να κάνει την μετατροπή.

Μια άλλη διακοπή που γίνεται είναι από τον μετατροπέα αναλογικού σήματος σε ψηφιακό. Στην διακοπή αυτή διαβάζουμε την τιμή τις μετατροπής και την αποθηκεύουμε στην μνήμη RAM για να χρησιμοποιηθεί στην εκτέλεση του αλγορίθμου.

Τέλος μια Τρίτη διακοπή που γίνεται είναι από το USART μετά την ολοκλήρωση της λήψης ενός Byte. Στην διακοπή αυτή διαβάζεται το Byte που λήφθηκε και σύμφωνα με το πρωτόκολλο επικοινωνίας γίνεται η κατάλληλη ενέργεια. Το πρωτόκολλο επικοινωνίας αναλύεται παρακάτω στο τμήμα που αναλύεται το πρόγραμμα της διακοπής.

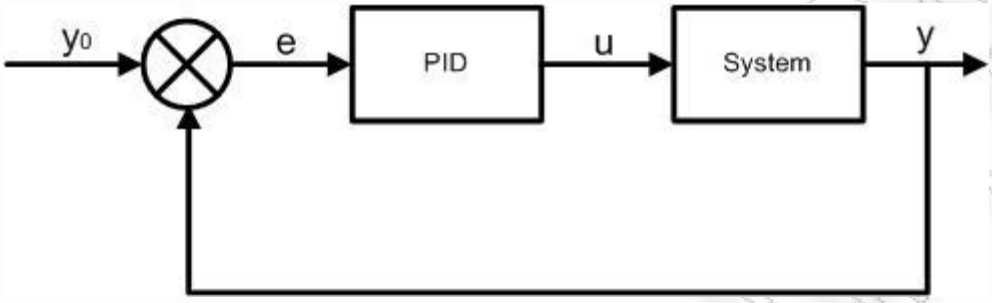
Στην παρακάτω εικόνα βλέπουμε το διάγραμμα ροής για το πρόγραμμα της διακοπής από τον Timer 1.



Εικόνα 67: Το διάγραμμα ροής του υποπρογράμματος για την διακοπή του Timer 1.

Στο υποπρόγραμμα της διακοπής από τον Timer 1 εκτελείται ο αλγόριθμος του PID ελεγκτή. Ο PID ελεγκτής μπορεί να χρησιμοποιηθεί για να ελέγξει κάθε μετρούμενη μεταβλητή υπό τον όρο ότι αυτή η μεταβλητή μπορεί να επηρεάζεται από τον χειρισμό κάποιον άλλων μεταβλητών της διεργασίας. Πολλές μέθοδοι ελέγχου έχουν χρησιμοποιηθεί κατά το παρελθόν αλλά ο PID ελεγκτής έχει γίνει το βιομηχανικό πρότυπο εξαιτίας της απαλότητας του και της καλής επίδοσης πού έχει.

Στην παρακάτω Εικόνα 68 φαίνεται το διάγραμμα ενός συστήματος με PID έλεγχο.



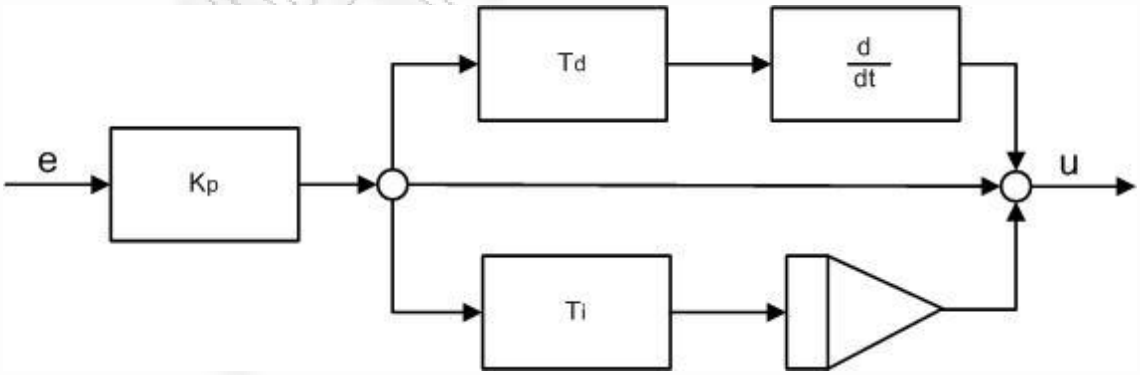
Εικόνα 68: Το διάγραμμα συστήματος με PID έλεγχο.

Ο PID ελεγκτής συγκρίνει την μετρούμενη μεταβλητή ελέγχου y με την επιθυμητή τιμή αναφοράς y_0. η διαφορά ή σφάλμα e είναι που επεξεργάζεται για να υπολογιστή μια νέα τιμή εισόδου στο σύστημα u. Αυτή η νέα τιμή εισόδου θα προσπαθήσει να ρυθμίσει την ελεγχόμενη μεταβλητή του συστήματος στην επιθυμητή τιμή.

Σε αντίθεση με τους απλούς αλγορίθμους ελέγχου ο PID αλγόριθμος είναι ικανός να χειριστεί τις εισόδους του συστήματος βασισμένος στο παρελθόν τους και στο ρυθμό μεταβολής τους. Αυτό δίνει μια ποιο ακριβή και σταθερή μέθοδο ελέγχου.

Η βασική ιδέα είναι ότι ο ελεγκτής διαβάζει την παρούσα κατάσταση του συστήματος με κάποιο αισθητήριο και τότε αφαιρεί την τιμή αυτή από την επιθυμητή τιμή για να παραχθεί η τιμή του σφάλματος. Το σφάλμα θα διαχειριστή με τρεις τρόπους. διαχείριση του παρόντος δια μέσου του αναλογικού τμήματος (Proportional term), ανάκτηση από το παρελθόν χρησιμοποιώντας το ολοκληρωτικό τμήμα (Integral term) και πρόβλεψη του μέλλοντος δια μέσου του διαφορικού τμήματος (Derivate term). Η διαδικασία αυτή γίνεται ανά κάποια χρονικά διαστήματα. Τα διαστήματα αυτά είναι η περίοδος δειγματοληψίας T. Ο χρόνος δειγματοληψίας πρέπει να είναι λιγότερος από τον μικρότερο σταθερό χρόνο στο σύστημα.

Στην Εικόνα 69 φαίνεται το διάγραμμα PID ελεγκτή όπου T_i και T_d σημαίνουν τις σταθερές χρόνου του ολοκληρωτικού και διαφορικού τμήματος.



Εικόνα 69: Διάγραμμα PID ελεγκτή.

Η συνάρτηση μεταφοράς του ελεγκτή στην Εικόνα 69 είναι:

$$\frac{u}{e}(s) = H(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

Αυτό δίνει την έξοδο u στο πεδίο του χρόνου:

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int_0^t e(\sigma)\,d\sigma + T_d\frac{de(t)}{dt}\right)$$

Κατά προσέγγιση η διακριτή μορφή του ολοκληρωτικού και διαφορικού τμήματος είναι:

$$\int_0^t e(\sigma)\,d\sigma \approx \sum_{k=0}^{n} e(k) \qquad \frac{de(t)}{dt} \approx \frac{e(n) - e(n-1)}{T} \qquad t=nT$$

Όπου n είναι το διακριτό βήμα στο χρόνο t

Αυτό δίνει την συνάρτηση μεταφοράς του ελεγκτή

$$u(n) = K_p e(n) + K_i\sum_{k=0}^{n} e(k) + K_d\left(e(n) - e(n-1)\right)$$

Όπου:

$$K_i = \frac{K_p T}{T_i} \quad \text{και} \quad K_d = \frac{K_p T_d}{T}$$

Ο ελεγκτής είναι ποιο βελτιωμένος βασίζοντας το διαφορικό τμήμα μόνο στην ελεγχόμενη μεταβλητή. Έτσι ο αλγόριθμος του ελεγκτή δίνεται από την ακόλουθη συνάρτηση μεταφοράς:

$$u(n) = K_p e(n) + K_i\sum_{k=0}^{n} e(k) + K_d\left(y(n) - y(n-1)\right)$$

Όπως αναφέρθηκε και παραπάνω ο ελεγκτής έχει ένα χρόνο δειγματοληψίας. Ο χρόνος αυτός δειγματοληψίας μπορεί και καθορίζεται από τον χρήστη μέσω της σειριακής επικοινωνίας από το PC. Ο Timer 1 καθορίζεται να κάνει την διακοπή κάθε 10 msec έτσι ο μικρότερος δυνατός χρόνος δειγματοληψίας του ελεγκτή θα μπορεί να είναι τα 10 msec. Έτσι ο χρήστης δηλώνει από το PC σαν χρόνο δειγματοληψίας έναν αριθμό και αυτός ο αριθμός θα είναι Χ 10 msec έτσι όταν ο χρήστης θα θέλει να βάλει σαν χρόνο δειγματοληψίας π.χ. 30 msec θα πρέπει απλά στο κατάλληλο πεδίο να βάλει τον αριθμό 3. Ο χρόνος των 10 msec που θα προκαλείται διακοπή από τον Timer 1 προέρχεται από το γεγονός ότι θα πρέπει να έχει προλάβει να ολοκληρωθεί η εκτέλεση της υπορουτίνας και η επικοινωνία με το PC πριν γίνει νέα διακοπή και αυτό μας το εξασφαλίζει ο χρόνος των 10 msec.

Όπως ξέρουμε ο μικροελεγκτής που χρησιμοποιούμε είναι των 8 bit αυτό σημαίνει ότι όλοι οι καταχωρητές του είναι των 8 bit επομένως και οι προσημασμένοι αριθμοί που μπορούμε να χειριστούμε είναι από -128 ως 127. Αυτό το εύρος για την εφαρμογή που αναπτύσσουμε δεν είναι ικανοποιητικό γιατί χρειαζόμαστε μεγαλύτερο εύρος τιμών. Έτσι αποφασίζουμε να χρησιμοποιήσουμε εύρος τιμών 16 bit. Αφού θα έχουμε αριθμούς τον 16 bit για να κάνουμε τις μαθηματικές πράξεις μεταξύ αριθμών γράφουμε κάποιες ρουτίνες που εκτελούν τις πράξεις και θα καλούνται στο σημείο που θέλουμε να κάνουμε την πράξη. Οι ρουτίνες αυτές είναι οι εξής:

mul16x16_32: εκτελεί πολλαπλασιασμό μεταξύ δύο μη προσημασμένων αριθμών 16bit.

muls16x16_32: εκτελεί πολλαπλασιασμό μεταξύ δύο προσημασμένων αριθμών 16bit.

div_8bit: εκτελεί διαίρεση μεταξύ δύο προσημασμένων αριθμών 8bit.

div_16bit: εκτελεί διαίρεση μεταξύ δύο προσημασμένων αριθμών 16bit.

Όταν ξεκινά η εκτέλεση του κώδικα της διακοπής από τον Timer 1 το πρώτο πράγμα που κάνουμε είναι να απενεργοποιήσουμε τις διακοπές ώστε να μπορεί να ολοκληρωθεί η εκτέλεση της ρουτίνας χωρίς να έχουμε μεταπήδηση της εκτέλεσης του προγράμματος σε άλλο σημείο και εν συνεχεία θέτουμε πάλη τον timer να κάνει διακοπή μετά από χρόνο 10 msec. Κατόπιν εξετάζεται αν έχει περάσει ο χρόνος δειγματοληψίας που έχουμε θέση. Αν δεν έχει παρέλθει ο χρόνος δειγματοληψίας τότε γίνεται ενεργοποίηση των διακοπών και έξοδος από την ρουτίνα και επιστροφή στο κυρίως τμήμα του προγράμματος. Όταν έχει περάσει ο χρόνος δειγματοληψίας συνεχίζεται η εκτέλεση του κώδικα όπως φαίνεται και στο διάγραμμα ροής στην Εικόνα 67. για να αποφύγουμε την υπερχείλιση κατά την εκτέλεση του αλγορίθμου θέτουμε κάποια όρια για τον ελεγκτή. Αυτά είναι το μέγιστο σφάλμα και το μέγιστο αθροιστικό σφάλμα. Έτσι στην συνέχεια του προγράμματος γίνονται τα εξής.

Υπολογισμός του μέγιστου σφάλματος.

Υπολογισμός του μέγιστου αθροιστικού σφάλματος.

Υπολογισμός του σφάλματος

Υπολογισμός του αναλογικού τμήματος

Υπολογισμός του νέου αθροιστικού σφάλματος

Υπολογισμός του ολοκληρωτικού τμήματος

Υπολογισμός του διαφορικού τμήματος

Υπολογισμός της εξόδου του PID

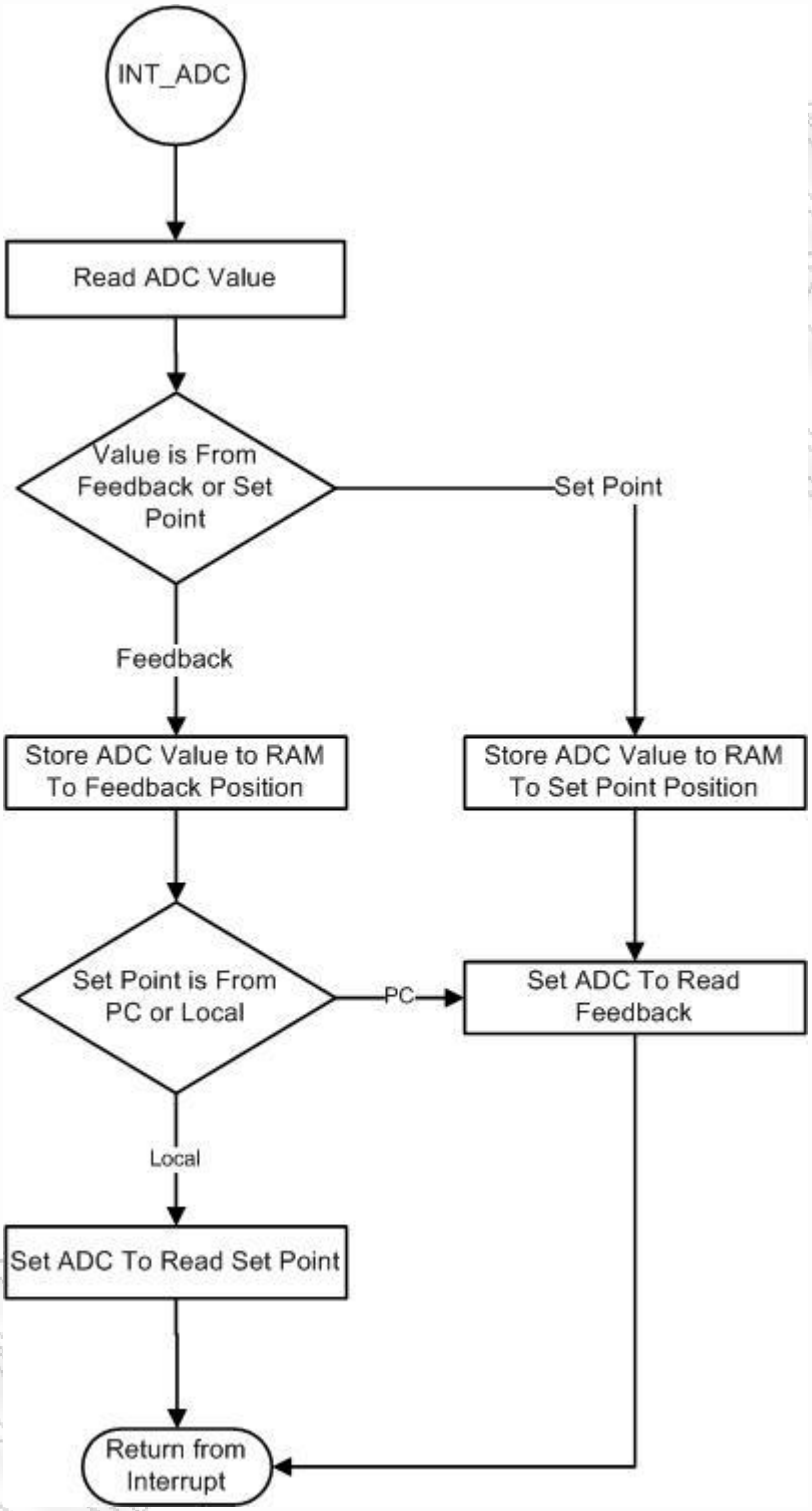
Έξοδος του αποτελέσματος στον DAC

Έλεγχος του DAC για την μετατροπή

Μετά την ολοκλήρωση της μετατροπής γίνεται έλεγχος αν ο χρήστης έχει δηλώσει από το PC να στέλνεται το αποτέλεσμα του αλγορίθμου στο PC. Αν από τον έλεγχο προκύπτει να μην γίνει αποστολή στο PC τότε γίνεται ενεργοποίηση των διακοπών και έξοδος στο κυρίως πρόγραμμα. Αν πρέπει να γίνει αποστολή στο PC τότε καλείται ή ρουτίνα αποστολής δεδομένων στο PC. Η λειτουργία τη ρουτίνας αυτής εξηγείται παρακάτω στην σειριακή επικοινωνία. Μετά την ολοκλήρωση της αποστολής γίνεται ενεργοποίηση των διακοπών και έξοδος στο κυρίως πρόγραμμα. Έτσι ολοκληρώνεται η εκτέλεση του προγράμματος της διακοπής από τον Timer 1.

Όπως αναφέρθηκε παραπάνω μια άλλη διακοπή που γίνεται είναι από τον μετατροπέα αναλογικού σήματος σε ψηφιακό. Όπως έχει αναφερθεί παραπάνω ο μικροελεγκτής διαθέτει έξη κανάλια για αναλογικό σήμα. Η μετατροπή από όλα τα κανάλια γίνεται από έναν μετατροπέα αναλογικού σε ψηφιακό. Έτσι πριν ξεκινήσει μια μετατροπή πρέπει να δηλώσουμε στον μετατροπέα από ποιο κανάλι θα πάρει το σήμα για την μετατροπή. Στην εφαρμογή που κάνουμε θα χρησιμοποιήσουμε δύο κανάλια το ADC0 στο οποίο έχουμε συνδέσει το σήμα από το αισθητήριο για την ανάδραση του συστήματος και το ADC5 στο οποίο έχουμε συνδέσει το ποτενσιόμετρο για να δίνουμε επιθυμητή τιμή τοπικά από τον ελεγκτή. Το κανάλι ADC5 πρέπει να το διαβάζουμε μόνο όταν έχουμε δηλώσει στον ελεγκτή ότι η είσοδος επιθυμητής τιμής γίνεται τοπικά από τον ελεγκτή και όχι από το PC. Έτσι όταν η επιθυμητή τιμή θα δίνεται από το PC ο ADC θα μετατρέπει μόνο το κανάλι ADC0 ενώ αν η επιθυμητή τιμή δίνεται τοπικά από τον ελεγκτή ο ADC θα μετατρέπει εναλλάξ μια φορά το κανάλι ADC0 και μια φορά το κανάλι ADC5.

Το διάγραμμα ροής του προγράμματος που εκτελείται μετά την διακοπή από τον ADC φαίνεται στην παρακάτω Εικόνα 70.



Εικόνα 70: Το διάγραμμα ροής του υποπρογράμματος για την διακοπή του ADC.

Τέλος μια λειτουργία που θα γίνεται από τον ελεγκτή θα είναι η σειριακή επικοινωνία με τον ηλεκτρονικό υπολογιστή για ανταλλαγή δεδομένων. Όπως αναφέρθηκε και παραπάνω ο μικροελεγκτής διαθέτει ο ίδιος ασύγχρονο σειριακό Interface full duplex. Ένα χαρακτηριστικό για να γίνει σωστά η σειριακή επικοινωνία είναι ότι και οι δύο συσκευές θα πρέπει να στέλνουν και να λαμβάνουν με τον ίδιο ρυθμό μετάδοσης. Οι τιμές που πρέπει να θέσουμε στον καταχωρητή UBRR για το ρυθμό μετάδοσης εξαρτάται από το ρολόι χρονισμού του μικροελεγκτή και για την συχνότητα τον 8 MHz που είναι το ρολόι του μικροελεγκτή φαίνεται στον παρακάτω πίνακα.

Table 62. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

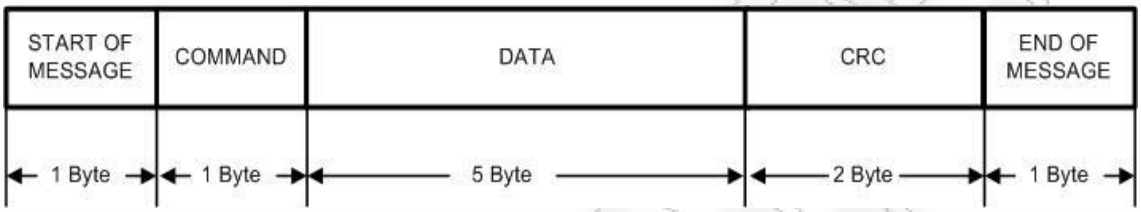
Εικόνα 71: Τιμές UBRR για τυποποιημένες τιμές ρολογιού του μικροελεγκτή.

Όπως βλέπουμε στον παραπάνω πίνακα όποιο Baud rate και να χρησιμοποιήσουμε υπάρχει ένα μικρό ποσοστό σφάλματος στην επικοινωνία μεταξύ των δύο συσκευών. Αυτό σημαίνει πως κάποια Byte θα λαμβάνονται λανθασμένα από τις συσκευές. Για να αποφύγουμε το σφάλμα αυτό θα εφαρμόσουμε και μια μέθοδο ελέγχου ότι η πληροφορία παραλήφθηκε σωστά από τον παραλήπτη. Η μέθοδος που θα εφαρμόσουμε για τον έλεγχο αυτό είναι η CRC (cyclic redundancy check). Θα εφαρμόζεται μόνο στα Byte τον δεδομένων και θα είναι δύο Byte τα οποία θα στέλνονται από τον αποστολέα στο τέλος μετά τα Byte των δεδομένων και πριν το Byte του τέλους μετάδοσης του μηνύματος. Ο παραλήπτης μετά το τέλος μετάδοσης του μηνύματος υπολογίζει και αυτός το CRC μα τον ίδιο αλγόριθμο και το συγκρίνει μα αυτό που έλαβε. Αν τα δύο CRC είναι ίδια τότε τα δεδομένα είναι σωστά και τα χρησιμοποιεί κατάλληλα. Αν τα δύο CRC δεν είναι ίδια τα δεδομένα απορρίπτονται. Ο υπολογισμός του CRC θα γίνεται με βάση τον παρακάτω αλγόριθμο:

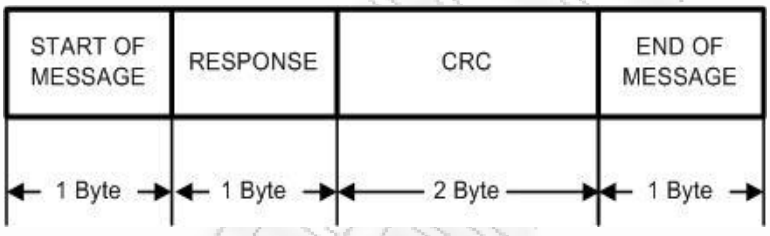
```
Initialize the checksum as FFFFhex.
For each byte
        Checksum = Checksum XOR (current byte)
        For I = 0 to 7
                If ( ( Checksum AND 1 ) = 0 )
                        Checksum = Right_Bit_Shift Checksum 1 bit
                Else
                        Checksum = (Right_Bit_Shift Checksum 1 bit) XOR A001hex
        Next I
Next Byte
```

Για την σωστή επικοινωνία μεταξύ των συσκευών πρέπει να υπάρχει ένα πρωτόκολλο επικοινωνίας. Έτσι και εδώ καθορίζουμε το πρωτόκολλο με το οποίο θα ανταλλάσσουν πληροφορία ο ελεγκτής με τον ηλεκτρονικό υπολογιστή. Τα δεδομένα που θα μεταδίδονται θα είναι με την μορφή ASCii χαρακτήρων. Όπως αναφέρθηκε παραπάνω στον ελεγκτή θα χρησιμοποιούμε εύρος τιμών 16 bit αυτό σημαίνει ότι ο μέγιστος αριθμός θα είναι 32767 άρα για την αποστολή ενός τέτοιου αριθμού σε ASCii χαρακτήρες χρειαζόμαστε 5 Byte.

Το πρωτόκολλο έχει δύο μόνο τύπους μηνυμάτων: τις εντολές (commands) και τις αποκρίσεις (responses). Όταν η μια συσκευή στέλνει μια εντολή περιμένει να πάρει μια απάντηση αν το μήνυμα μεταδόθηκε σωστά έτσι ώστε αν δεν έχει μεταδοθεί σωστά να γίνει πάλη αποστολή του μηνύματος. Οι δύο τύποι μηνυμάτων δεν έχουν την ίδια μορφή, η μορφή του κάθε μηνύματος φαίνεται στις παρακάτω εικόνες.



Εικόνα 72: Η μορφή του μηνύματος εντολής.



Εικόνα 73: Η μορφή του μηνύματος απάντησης.

Ή ανάλυση των μηνυμάτων έχει ως εξής:

- Start of Message

  STX = 02Hex Σημαίνει την αρχή μετάδοσης ενός μηνύματος.

- Command

  A0Hex: Εγγραφή της παραμέτρου KP
  A1Hex: Εγγραφή της παραμέτρου KI
  A2Hex: Εγγραφή της παραμέτρου KD
  A3Hex: Εγγραφή της παραμέτρου Set Point
  A4Hex: Εγγραφή της παραμέτρου T PID
  A5Hex: Αποστολή της εξόδου στο PC

  B0Hex: Ανάγνωση της παραμέτρου KP
  B1Hex: Ανάγνωση της παραμέτρου KI
  B2Hex: Ανάγνωση της παραμέτρου KD
  B3Hex: Ανάγνωση της παραμέτρου Set Point
  B4Hex: Ανάγνωση της παραμέτρου T PID
  B5Hex: Ανάγνωση της εξόδου του PID
  B6Hex: Ανάγνωση της ανατροφοδότησης
  B7Hex: Ανάγνωση όλων των παραπάνω.

Με τις εντολές A0Hex ως A5Hex στέλνονται οι τιμές των παραμέτρων από τον ηλεκτρονικό υπολογιστή στον ελεγκτή και ο ελεγκτής τα αποθηκεύει στην μνήμη RAM και στην EEPROM.

Με τις εντολές B0Hex ως B6Hex ο ηλεκτρονικός υπολογιστής ζητά από τον ελεγκτή να του μεταδώσει την τιμή της αντίστοιχης παραμέτρου και με το B7Hex όλες τις παραμέτρους.

- Data

  Στο τμήμα αυτό του μηνύματος η συσκευή μεταδίδει την τιμή της παραμέτρου σε μορφή ASCii χαρακτήρα. Το μήκος του τμήματος αυτού είναι 5 Byte. Αν ο αριθμός που θα μεταδώσει η συσκευή έχει λιγότερους από 5 χαρακτήρες τότε συμπληρώνουμε με τον χαρακτήρα “0” ώστε να έχουμε 5 Byte. Π.χ. για την αποστολή του αριθμού 178 θα στείλουμε τους χαρακτήρες “00178”.

- CRC

  Μεταδίδονται δύο Byte για τον έλεγχο σωστής μετάδοσης του μηνύματος. Ο αλγόριθμος υπολογισμού των δύο Byte περιγράφεται παραπάνω.

- End of Message

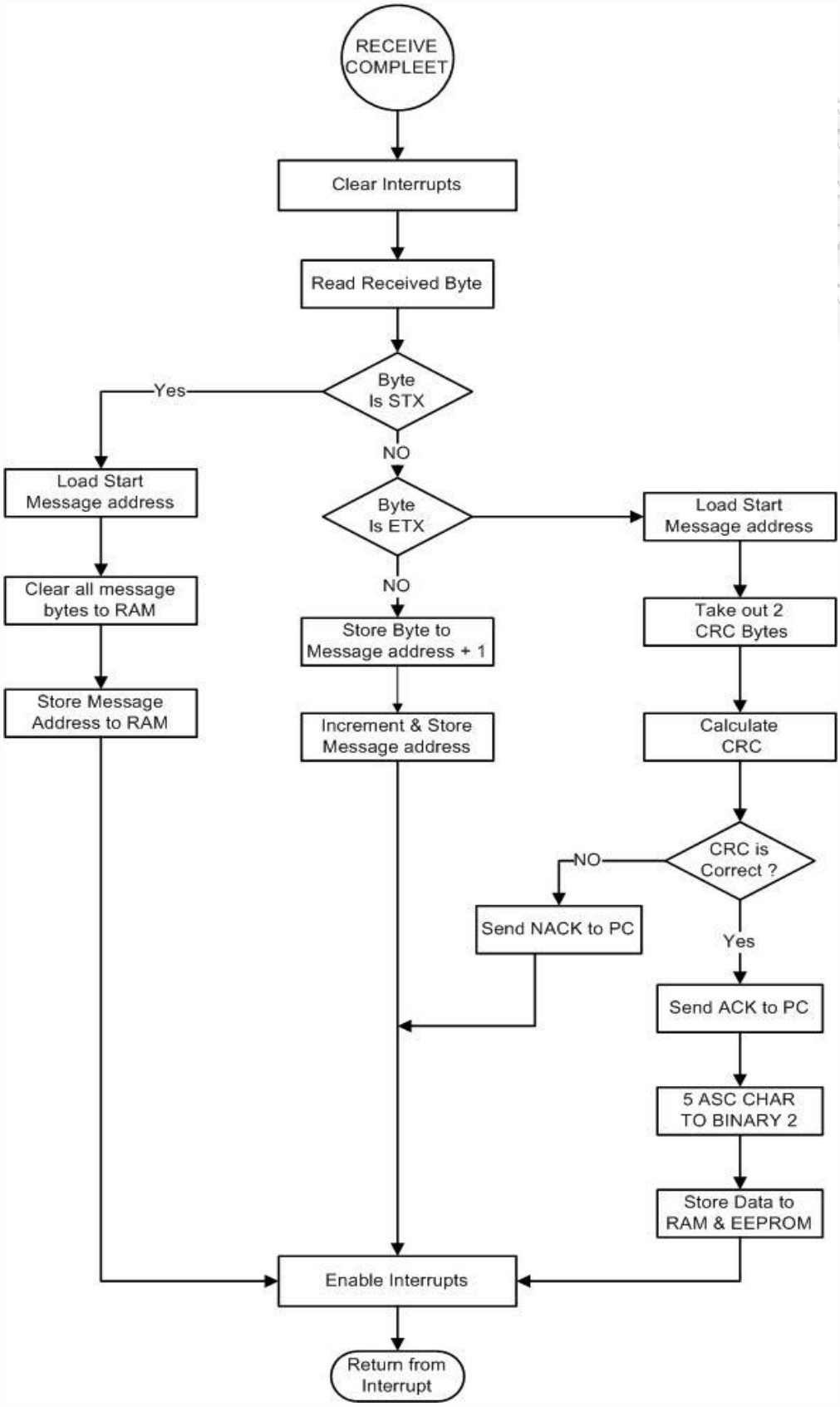
  ETX = 03Hex Σημαίνει το τέλος μετάδοσης ενός μηνύματος.

- Response

  ACK=06Hex Όταν το μήνυμα έχει μεταδοθεί σωστά.

  NAK=15Hex Όταν το μήνυμα έχει μεταδοθεί λανθασμένα.

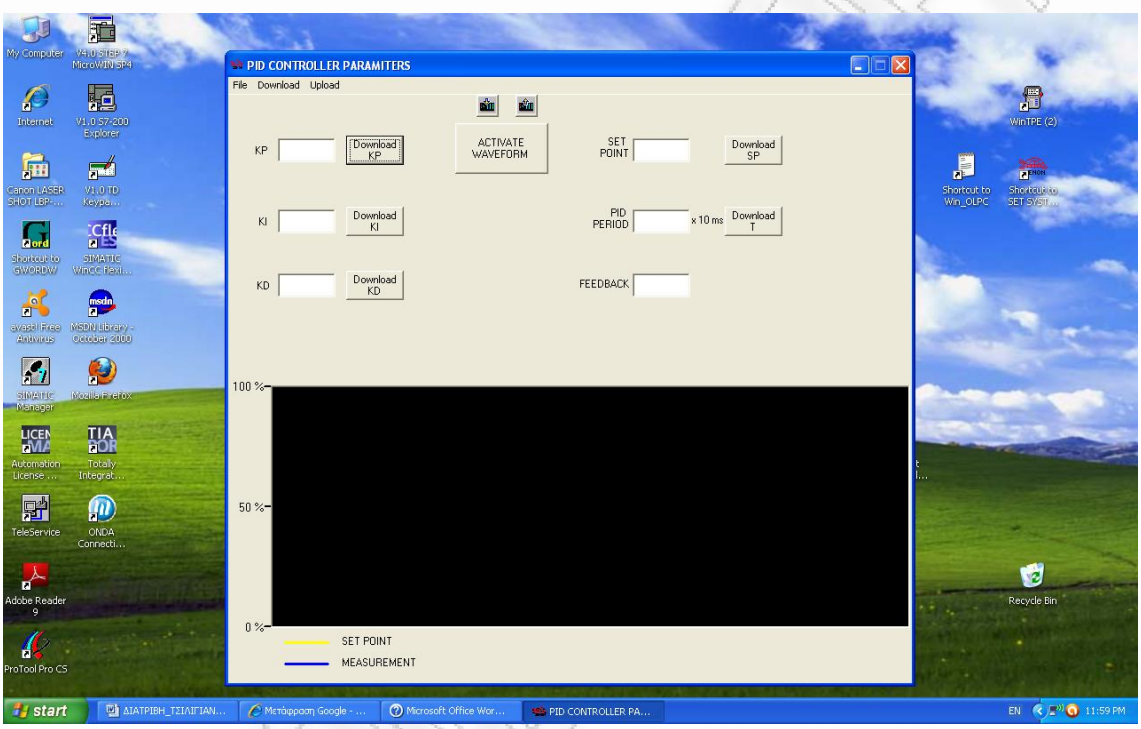
Το διάγραμμα ροής του προγράμματος στον ελεγκτή που υλοποιεί το παραπάνω πρωτόκολλο φαίνεται στην παρακάτω εικόνα.



Εικόνα 74: Το διάγραμμα ροής του υποπρογράμματος για την διακοπή από το USART.

Για την σειριακή επικοινωνία του ελεγκτή με τον υπολογιστή εκτός από το πρόγραμμα στον μικροελεγκτή χρειάζεται και ένα πρόγραμμα στον υπολογιστή το οποίο είναι φτιαγμένο ώστε να υλοποιεί το πρωτόκολλο που αναλύσαμε ποιο πάνω μέσω της σειριακής πόρτας του υπολογιστή. Το πρόγραμμα αυτό αναπτύχθηκε σε Visual Basic. Από την εφαρμογή αυτή ο χρήστης θα μπορεί να γράψει και να διαβάσει τις παραμέτρους του ελεγκτή PID K_P,K_I, K_D, T_{PID} επιπλέον θα μπορεί να δώσει την επιθυμητή τιμή στον ελεγκτή και να διαβάσει αυτή που έχει ο ελεγκτής και εκτελεί τον αλγόριθμο. Μια άλλη λειτουργία που κάνει η εφαρμογή είναι να μας δείχνει την επιθυμητή τιμή και την έξοδο του συστήματος σε κυματομορφή. Για την αποστολή αυτόν τον δεδομένων από τον ελεγκτή στον υπολογιστή υπάρχει επιλογή στο πρόγραμμα του υπολογιστή.

Η οθόνη του προγράμματος του υπολογιστή φαίνεται στην παρακάτω εικόνα.



Εικόνα 75: Η οθόνη της εφαρμογής στον υπολογιστή.

Μετά την ανάπτυξη του προγράμματος στον υπολογιστή, την κατασκευή της πλακέτας του ελεγκτή PID έγινε η διασύνδεση της εξόδου του ελεγκτή με την πλακέτα του επενεργητή. Ή όλη κατασκευή τέθηκε σε λειτουργία. Το πρώτο πράγμα που έπρεπε να γίνει είναι η ρύθμιση των παραμέτρων του PID.

Η διαδικασία της ρύθμισης των παραμέτρων PID ελεγκτή ώστε να λειτουργεί σωστά ο ελεγκτής γίνεται υπολογίζοντας τις παραμέτρους από το μαθηματικό μοντέλο του συστήματος. Στις περισσότερες περιπτώσεις και ειδικά σε συστήματα της βιομηχανίας δεν γνωρίζουμε το μαθηματικό μοντέλο του συστήματος και έτσι δεν μπορούν να υπολογιστούν από αυτό οι παράμετροι του ελεγκτή. Για το λόγο αυτό έχουν αναπτυχθεί εμπειρικές μέθοδοι υπολογισμού των παραμέτρων PID ελεγκτή. Η ποιο γνωστή εμπειρικοί μέθοδοι υπολογισμού παραμέτρων είναι η μέθοδος Cohen – Coon και η μέθοδος Zeigler – Nichols.

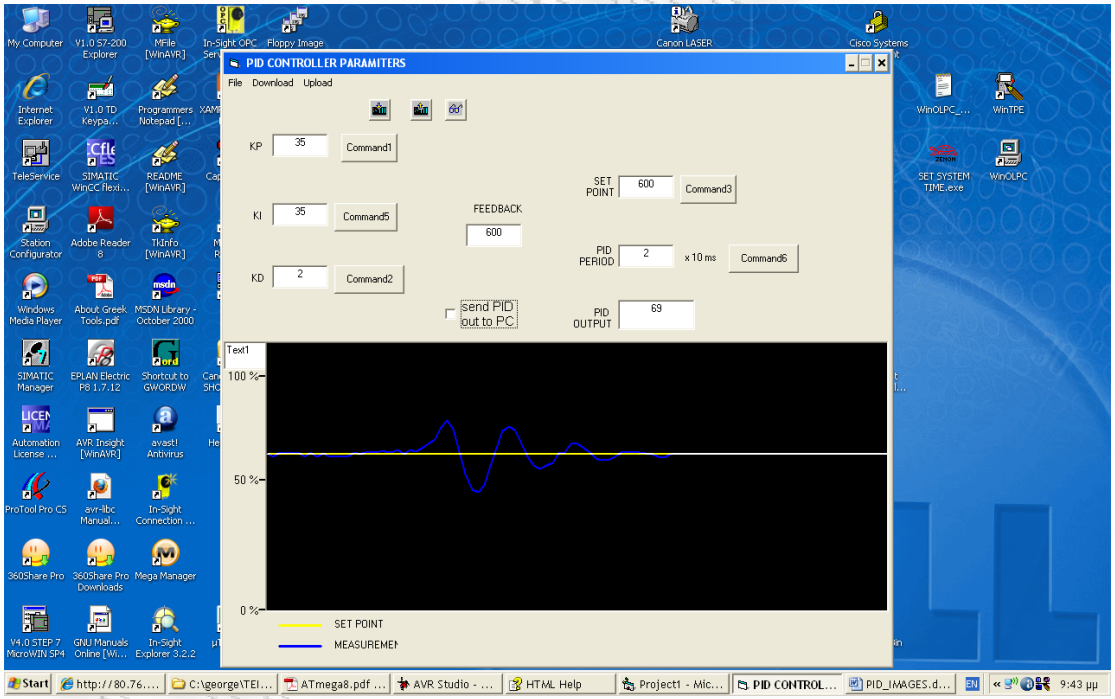
Στο σύστημα της εργασίας η διεργασία είναι απλή και θα μπορούσαμε να υπολογίσουμε το μαθηματικό μοντέλο της και από αυτό τις παραμέτρους του PID ελεγκτή αλλά ο σκοπός της εργασίας δεν ήταν αυτός. Έτσι για τον υπολογισμό τον παραμέτρων ακολουθούμε την εμπειρική μέθοδο των Zeigler – Nichols η οποία έχει ως εξής:

Οι παράμετροι των όρων ολοκλήρωσης και διαφόρισης τοποθετούνται στη χαμηλότερη δυνατή τιμή (δηλαδή τα KI, KD =0) και το κέρδος Kc αυξάνεται σταδιακά μέχρι να παρατηρηθεί ταλάντωση σταθερού εύρους στην έξοδο. Το κέρδος Kc σε αυτή τη περίπτωση αντιστοιχεί στο

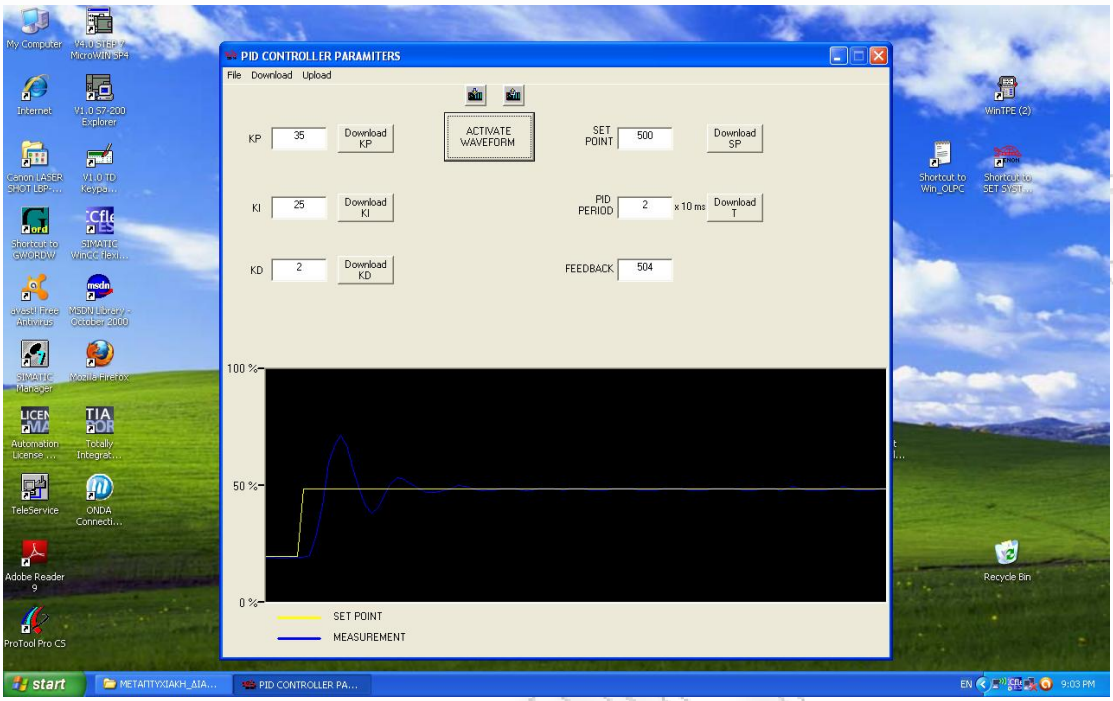
Κκρισ. ενώ η περίοδος είναι Το και στη συνέχεια από τις εξισώσεις του παρακάτω πίνακα υπολογίζονται οι παράμετροι του ελεγκτή PID.

Controller	Parameters		
	Kp	Ki	Kd
P	$0,5K_C$		
PI	$0,45K_C$	$1,2Kp/T_0$	
PID	$0,6K_C$	$2Kp/T_0$	$0,125\ KpT_0$

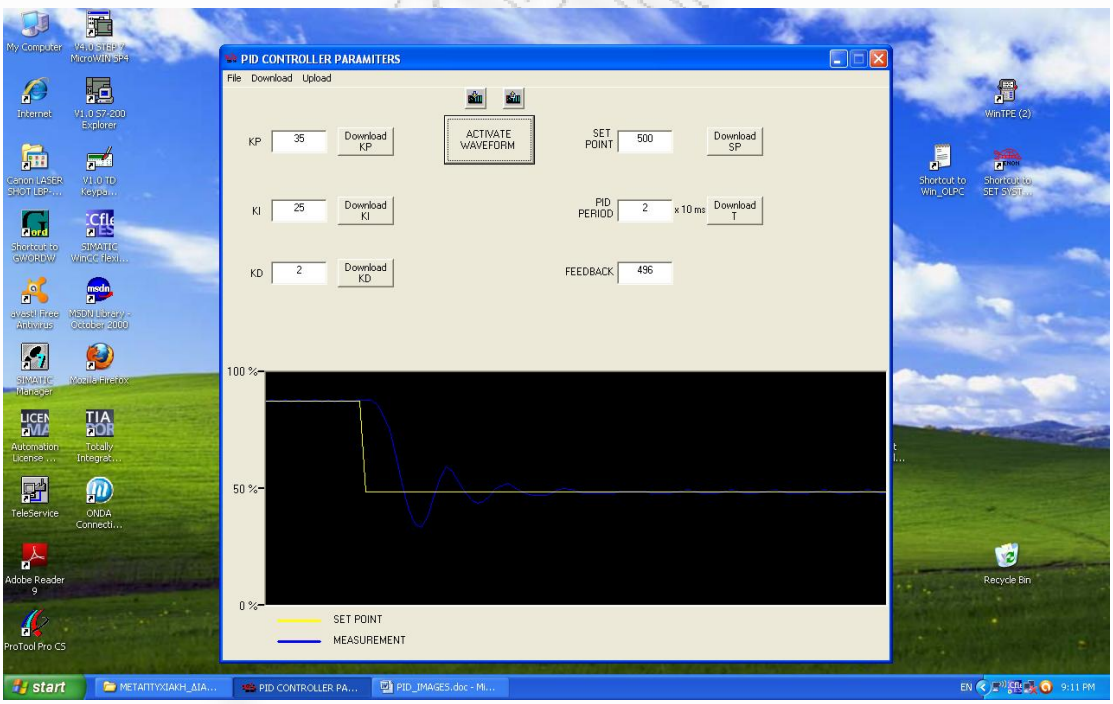
Μετά τον υπολογισμό των παραμέτρων και την εισαγωγή τους στον ελεγκτή μέσω της εφαρμογής στο PC το σύστημα τέθηκε σε λειτουργία και στις παρακάτω εικόνες βλέπουμε τις κυματομορφές που παίρνουμε στο PC σε λειτουργία χωρίς διαταραχές στο σύστημα, σε αλλαγή της επιθυμητής τιμής και σε επίδραση από διαταραχή στο σύστημα.



Εικόνα 76: Η έξοδος του συστήματος σε μεταβολή της εντάσεως φωτισμού.



Εικόνα 77: Η έξοδος του συστήματος σε αύξηση της επιθυμητής τιμής.



Εικόνα 78: Η έξοδος του συστήματος σε μείωση της επιθυμητής τιμής.

ΣΥΜΠΕΡΑΣΜΑΤΑ

Από την όλη διαδικασία της μελέτης, της σχεδίασης και της κατασκευής του ελεγκτή PID και του επενεργητή μπορούμε να πούμε ότι η λειτουργία και η επιδόσεις του συστήματος μας αφήνει πάρα πολύ ικανοποιημένους.

Ουσιαστικά η εργασία χωρίστηκε σε δύο κύρια μέρη: την μελέτη και κατασκευή της διάταξης οδήγησης της ισχύος στον λαμπτήρα πυρακτώσεως(επενεργητής) και την υλοποίηση και εφαρμογή της διάταξης ελέγχου με την σχεδίαση και κατασκευή του ελεγκτή PID. Βάση των αποτελεσμάτων, τόσο των κυματομορφών της διάταξης ισχύος που κατασκευάστηκε όσο και των αποκρίσεων στην εφαρμογή του ελέγχου, θα μπορούσαμε να πούμε πως ο στόχος που είχαμε αρχικά θέσει έχει επιτευχθεί πλήρως.

Η εργασία αυτή περιλαμβάνει και την σχεδίαση σε υλικό (hardware) και την ανάπτυξη σε λογισμικό (software) παρέχοντας έτσι γνώση και εμπειρία και στα δύο αντικείμενα καθώς και στον τρόπο που αυτά συνδυάζονται. Κατά την εκπόνηση της εργασίας παρουσιάστηκαν διάφορα προβλήματα και στην σχεδίαση του υλικού αλλά και κατά την ανάπτυξη του λογισμικού της εργασίας.

Ένα από τα ποιο σοβαρά και κρίσιμα προβλήματα που έπρεπε να επιλυθεί ήταν το γεγονός ότι ο λαμπτήρας τροφοδοτείται από το δίκτυο παροχής ηλεκτρικής ισχύος της ΔΕΗ και αυτό εμπεριέχει τον κίνδυνο ηλεκτροπληξίας λόγο της υψηλής τάσης του δικτύου. Η λύση που δόθηκε είναι με γαλβανική απομόνωση μεταξύ του κυκλώματος ισχύος και των κυκλωμάτων χαμηλής τάσης με χρήση οπτικής σύζευξης (optocuplers).

Ένα άλλο πρόβλημα που αντιμετώπισα ήταν ο θόρυβος στα αναλογικά σήματα και στην τροφοδοσία των μικροελεγκτών. Η σημείωση AVR042 (Application Note AVR042: AVR Hardware design considerations) που παρέχει η εταιρεία Atmel αναφέρει ορισμένες μεθόδους που μπορούνε να χρησιμοποιηθούνε για την επίλυση των προβλημάτων αυτών. Επίσης πρόβλημα από θόρυβο αντιμετώπισα στο σήμα που δίνει στον μικροελεγκτή το τμήμα ανίχνευσης διέλευσης της τάσης από το μηδέν. Η λύση που δόθηκε σε αυτό το πρόβλημα ήταν μέσω του λογισμικού του μικροελεγκτή διαβάζοντας το σήμα πολλές φορές πριν προχωρήσουμε στην εκτέλεση του κώδικα της διακοπής που προκαλεί αυτό το σήμα.

Εκτός από τα παραπάνω προβλήματα πού αφορούσαν το κατασκευαστικό κομμάτι της εργασίας υπήρξαν και προβλήματα κατά την ανάπτυξη του λογισμικού των μικροελεγκτών. Ένα τέτοιο πρόβλημα στον μικροελεγκτή του επενεργητή ήταν στην ανάπτυξη του αλγορίθμου για την πυροδότηση του triac γιατί όπως φαίνεται από την Εικόνα 56 η γωνία έναυσης α και η τάση που εφαρμόζεται στο φορτίο δεν έχουν γραμμική σχέση. Η λύση που δόθηκε σε αυτό το πρόβλημα είναι να θεωρηθεί κατά προσέγγιση γραμμική γιατί μετά από πολλές δοκιμές και πειραματισμούς συμπέρανα ότι δεν επηρεάζει σημαντικά την λειτουργία του επενεργητή.

Ένα πρόβλημα που αντιμετώπισα πάλι από μη γραμμικότητα είναι στην λειτουργία του αισθητήριου ανάδρασης του ελεγκτή PID. Όπως φαίνεται στην Εικόνα 62 η πραγματική φωτοαντίσταση δεν έχει γραμμική σχέση με το μήκος κύματος λ και αυτό δημιουργούσε πρόβλημα στην λειτουργία του αλγορίθμου του PID. Για την λύση αυτού του προβλήματος έγινε ερεύνα για την αγορά αισθητήριου με γραμμική σχέση. Από την έρευνα αγοράσθηκε αισθητήρας Linear light sensor LLS05-A από το κατάστημα Futurlec (http://www.futurlec.com). Η χρησιμοποίηση αυτού του αισθητηρίου αντί να φέρει καλύτερα αποτελέσματα αντιθέτως έφερε χειρότερα γιατί ναι μεν έχει γραμμική σχέση αλλά είναι πάρα πολύ ευαίσθητο στις μεταβολές του φωτός με αποτέλεσμα να δημιουργεί ταλάντωση στην λειτουργία του PID ελεγκτή. Άλλα αισθητήρια που βρέθηκαν κατά την έρευνα αγοράς με γραμμική σχέση ήταν πάρα πολύ ακριβά στην τιμή τους. Και έτσι το αισθητήριο ανάδρασης αντικαταστάθηκε πάλη από την φωτοαντίσταση.

Το αποτέλεσμα της παραπάνω εργασίας δεν είναι το τέλειο επομένως και η κατασκευή και το λογισμικό μπορούν να δε δεχθούν βελτιώσεις. Τα κυριότερα σημεία που μπορούν να δεχθούν βελτίωση είναι:

Αλλαγή του αισθητηρίου ανάδρασης με αισθητήριο που να έχει γραμμική σχέση με το μήκος κύματος λ.

Αλλαγή του μετατροπέα ψηφιακού σήματος σε αναλογικό με μεγαλύτερης ανάλυσης.

Βελτίωση του αλγορίθμου για την πυροδότηση του triac λόγο της μη γραμμικής σχέσεως της γωνίας έναυσης α και της τάσης στο φορτίο.

Το ποιο σημαντικό βέβαια σημείο για βελτίωση είναι ο αλγόριθμος του PID. Στην εργασία χρησιμοποιήθηκε ένας απλοποιημένος αλγόριθμος. Η απομάκρυνση των ορίων και των υπερχειλίσεων καθώς και η διόρθωση στον κορεσμό θα κάνει τον ελεγκτή ποιο ισχυρό και με μεγαλύτερη ακρίβεια.

Η εκπόνηση μιας εργασίας με κατασκευαστικό κομμάτι είναι μια δύσκολη υπόθεση και απαιτεί πολύ κόπο, δουλειά και προσπάθεια από την πλευρά του φοιτητή. Ο τελικός απολογισμός από την εκπόνηση της παρούσας εργασίας είναι πολλές γνώσεις και εμπειρία και ένα αποτέλεσμα το οποίο ανταμείβει διότι παίρνει ζωή μετά από τόσες προσπάθειες.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. “Προγραμματίζοντας Τον Μικροελεγκτή AVR”, του Dhananjay V. Gadre.
2. “ATmega8 Datasheet”, της εταιρείας Atmel.
3. “AVR Instruction Set”, της εταιρείας Atmel.
4. “AVR Studio 4 User’s Guide”, της εταιρείας Atmel.
5. “AVR Assembler User’s Guide”, της εταιρείας Atmel.
6. “AVR Simulator User’s Guide”, της εταιρείας Atmel.
7. “STK500 User’s Guide”, της εταιρείας Atmel.
8. “AVRISP User’s Guide”, της εταιρείας Atmel.
9. Σημειώσεις για τα μαθήματα “Μικροϋπολογιστές” του τμήματος αυτοματισμού του ΤΕΙ Πειραιά.
10. bose b. k. modern power electronics and ac drives prentice-hall n.j. 2002
11. Σημειώσεις για τα μαθήματα “Ηλεκτρονικά Ισχύος” του τμήματος αυτοματισμού του ΤΕΙ Πειραιά.
12. Σημειώσεις για τα μαθήματα “Σ.Α.Ε. Ι” του τμήματος αυτοματισμού του ΤΕΙ Πειραιά.
13. Σημειώσεις για τα μαθήματα “Σ.Α.Ε. ΙΙ” του τμήματος αυτοματισμού του ΤΕΙ Πειραιά.
14. Σημειώσεις για τα μαθήματα “ΨΗΦΙΑΚΑ Σ.Α.Ε.” του τμήματος αυτοματισμού του ΤΕΙ Πειραιά.
15. “Introduction To Control System Technology“, Robert N. Bateson.
16. “Συστήματα αυτομάτου ελέγχου“, Παρασκευόπουλου Τόμος Α
17. “6N138 Datasheet”, Fairchild Semiconductor.
18. “DAC0830 Datasheet”, National Semiconductor.
19. “LM358 Datasheet”, National Semiconductor.

Internet Links

1. http://www.atmel.com
2. http://www.avrfreaks.net
3. http://www.national.com
4. http://www.avrbeginners.net
5. http://www.ti.com
6. http://www.st.com
7. http://www.intersil.com
8. http://www.maxim-ic.com

ΠΑΡΑΡΤΗΜΑ Α

Το Πρόγραμμα Του Μικροελεγκτή Για Τον επενεργητή.

```
.include "m8def.inc"

;*****Global Register Variables ********
.def    temp    =r23            ;temporary variable
.def    flag    =r30

;Interupt vectors
        rjmp    reset
        rjmp    int_0
        nop
        nop
        nop
        nop
        rjmp    timer_1
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        rjmp    int_adc

reset:
        ldi     temp,low(ramend)
        out     spl,temp                ;set stack pointer to $cf
        ldi     temp,high(ramend)
        out     sph,temp
        ldi     temp,$f0
        out     GICR,temp               ;enable INT_0 and INT_1
        ldi     temp,0b00001111         ;$0f
        out     MCUCR,temp              ;set INT_0 and INT_1 on the rising edge
        ldi     temp,$00
        out     DDRB,temp               ; set portB as INPUT
        ldi     temp,0b00100011
        out     DDRD,temp               ;set portD PIN0 & PIN1 & PIN5 as output
        ldi     temp,$01
        out     PORTD,temp
        ldi     temp,$00                ;Set port C as INPUT
        out     DDRC,temp
```

```
;Configure Timer 1
        ldi     temp,16
        out     TIMSK,temp
        ldi     temp,$A
        out     TCCR1B,temp
        ldi     temp,low(30000)
        out     OCR1AL,temp
        ldi     temp,high(30000)
        out     OCR1AH,temp


;Configure ADC
        ldi     temp,0b01000000
        out     ADMUX,temp
        ldi     temp,0b11101000
        out     ADCSRA,temp

        sbi     PORTD,0

MAIN:   nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        rjmp    MAIN


int_0:
        ldi     r27,$1F
X1:     in      r24,PIND
        sbrs    r24,2
        reti
        dec     r27
        brne    X1
        sbi     PORTD,0
        lds     r16,$100            ;load the value from adc to r16,r17
        lds     r17,$101
        ldi     r18,low(10)         ;load the value 10 to r18,r19
        ldi     r19,high(10)
```

```
        rcall   mul_16bit           ;multiplay  r16,17 * r18,r19  (10 * ADC)
        ldi     r19,high(10230)     ;Load the value 10230 to r18,r19
        ldi     r18,low(10230)
        sub     r18,r16             ;Subtruct  (10230-10*ADC)
        sbc     r19,r17
        ldi     r16,low(10000)
        ldi     r17,high(10000)
        cp      r18,r16
        cpc     r19,r17
        brlo    next
        ldi     r18,low(10000)
        ldi     r19,high(10000)


next:   out     OCR1AL,R18          ;Load the result to Timer 1
        out     OCR1AH,R19
        ldi     temp,16
        out     TIMSK,temp          ;Enable Timer 1 Interrupt
        ldi     temp,$A
        out     TCCR1B,temp
        ldi     temp,$ff
        out     GIFR,temp
        reti


timer_1:
        ldi     temp,8
        out     TCCR1B,temp         ;Stop Timer 1
        cbi     PORTD,0             ;go PD0 to Zero to open the light
        ldi     temp,$ff
        out     GIFR,temp
        ldi     temp,$f0
        out     GICR,temp           ;enable INT_0 and INT_1
        sei
        reti

int_adc:
        in      r20,ADCL
        sts     $100,r20            ;store low byte to RAM address $100
        in      r21,ADCH
        sts     $101,r21            ;Store high byte to RAM address $101
        ldi     temp,0b11101000
        out     ADCSRA,temp
        reti
```

```
;***************************************************************************
;*    16X16 bit signed multiplication                                      *
;*                                                                         *
;***************************************************************************
mul_16bit:
.def    mc16sL          =r18
.def    mc16sH          =r19
.def    mp16sL          =r16
.def    mp16sH          =r17
.def    m16s0           =r16            ;result byte 0 (LSB)
.def    m16s1           =r17            ;result byte 1
.def    m16s2           =r20            ;result byte 2
.def    m16s3           =r21            ;result byte 3 (MSB)
.def    mcnt16s         =r22            ;Loop counter
mpy16s:clr      m16s3
       sub      m16s2,m16s2
       ldi      mcnt16s,16
m16s_1:brcc     m16s_2
       add      m16s2,mc16sL
       adc      m16s3,mc16sH
m16s_2:sbrc     mp16sL,0
       sub      m16s2,mc16sL
       sbrc     mp16sL,0
       sbc      m16s3,mc16sh
       asr      m16s3
       ror      m16s2
       ror      m16s1
       ror      m16s0
       dec      mcnt16s
       brne     m16s_1
       ret
```

ΠΑΡΑΡΤΗΜΑ Β

Το Πρόγραμμα Του Μικροελεγκτή Για Τον Ελεγκτή PID.

```
;set EEPROM address for KP, KI, KD, SP, F
.equ    KP_factor       =$00
.equ    KI_factor       =$02
.equ    KD_factor       =$04
.equ    SP_factor       =$06        ;set point
.equ    F_factor        =$08        ;Frequency to calculate PID X10ms
.equ    SEND_factor     =$0A        ;1:To send PID out to PC after PID calculation

;set SRAM addresses for the data
.equ    errorH          =$100
.equ    errorL          =$101
.equ    error_n_1H      =$102
.equ    error_n_1L      =$103
.equ    sum_errorH      =$104
.equ    sum_errorL      =$105
.equ    max_errorH      =$106
.equ    max_errorL      =$107
.equ    max_sum_errorH  =$108
.equ    max_sum_errorL  =$109
.equ    last_valueH     =$10A
.equ    last_valueL     =$10b
.equ    pidH            =$10c
.equ    pidL            =$10d
.equ    fbH             =$10e
.equ    fbL             =$10f       ;feadback

.equ    p_termH         =$110
.equ    p_termL         =$111
.equ    i_termH         =$112
.equ    i_termL         =$113
.equ    d_termH         =$114
.equ    d_termL         =$115
.equ    kpH             =$116
.equ    kpL             =$117
.equ    kiH             =$118
.equ    kiL             =$119
.equ    kdH             =$11a
.equ    kdL             =$11b
.equ    set_pH          =$11c
.equ    set_pL          =$11d
.equ    f_pid           =$11e       ;Frequency to calculate PID X10ms
.equ    count_f         =$11f       ;Timer 1 Overflou Counter
```

```
.equ    to_send_PID             =$121

;set SRAM addresses for the serial comunication
.equ    address_H               =$200
.equ    address_L               =$201
.equ    code_byte               =$202
.equ    data_H                  =$203
.equ    data_L                  =$204
.equ    asc5                    =$250

.equ    STX                     =$02
.equ    ETX                     =$03
.equ    ACK                     =$06
.equ    NAK                     =$15

.equ    receive_KP      =$A0    //Write recived from PC parameter KP to RAM and EEPROM
.equ    receive_KI      =$A1    //Write recived from PC parameter KI to RAM and EEPROM
.equ    receive_KD      =$A2    //Write recived from PC parameter KD to RAM and EEPROM
.equ    receive_SP      =$A3    //Write recived from PC Set Point to RAM and EEPROM
.equ    receive_FPID    =$A4    //Write recived from PC parameter FPID to RAM and EEPROM
.equ    rec_to_send     =$A5    //Write recived from PC to send PID out to RAM and EEPROM
.equ    rec_out         =$A6

.equ    send_KP         =$B0    //Send to PC parameter KP which is stored to RAM
.equ    send_KI         =$B1    //Send to PC parameter KI which is stored to RAM
.equ    send_KD         =$B2    //Send to PC parameter KD which is stored to RAM
.equ    send_SP         =$B3    //Send to PC parameter SP (Set Point) which is stored to RAM
.equ    send_FPID       =$B4    //Send to PC parameter FPID which is stored to RAM
.equ    send_OUT_PID    =$B5    //Send to PC parameter PID  OUTPUT which is stored to RAM
.equ    send_FBCK       =$B6    //Send to PC Feedback which is stored to RAM
.equ    send_ALL        =$B7    //Send to PC all parameters
.equ    send_SSS        =$B8    //Send to PC all parameters

.equ    max_int8                =127
.equ    n_max_int8              =-127
.equ    max_int                 =32767
.equ    n_max_int               =-32768
.equ    max_long                = 2147483647
.equ    n_max_long              =(-2147483647 - 1)
.equ    max_i_term              = max_int/2
.equ    scale_factor            =32
```

```
.include "m8def.inc"
.include "C:\george\PID_FROM_AVR_21_2_2010\ram_def.inc"

;difine registers
.def    loop_counter    =r5
.def    tempL           =r16            ;temporary variable
.def    tempH           =r17            ;temporary variable
.def    temp_bL         =r18            ;temporary variable
.def    temp_bH         =r19            ;temporary variable
.def    temp_cL         =r20            ;temporary variable
.def    temp_cH         =r21            ;temporary variable
.def    temp_dL         =r22            ;temporary variable
.def    temp_dH         =r23            ;temporary variable
.def    rmp             =r24
.def    rs_read         =r25
.def    rBin1H          =r26
.def    rBin1L          =r27

.equ    CS              =2              ;Chip Select for DAC is in PC2
.equ    WR1             =1              ;Write for DAC is in PC1
.equ    PC_POT          =2              ;SET POINT select is in PD2     /PC_POT=1       SET
POINT is from PC, PC_POT=0 SET POINT is from POTENTIOMETER


;Interrupt vectors
                rjmp    RESET
                nop
                nop
                nop
                nop
                nop
                nop
                nop
                rjmp    TIMER_1_OVF
                nop
                nop
                rjmp    RECEIVE_COMPLEET
                nop
                nop
                rjmp    INT_ADC
```

```
RESET:  ldi   tempL,low(ramend)   ;Load low address byte for stack pointer
        ldi   tempH,high(ramend)  ;Load high address byte for stack pointer
        out   SPH,tempH           ;init high byte of stack pointer
        out   SPL,tempL           ;init low byte of stack pointer
        ldi   tempL,$FF
        out   DDRB,tempL          ;set portB as output
        ldi   tempL,0b11111011
        out   DDRD,tempL          ;set portD as output and PD2 as Input
        ldi   tempL,0b00000110    ;Set port C as INPUT and PC1, PC2 as Output
        out   DDRC,tempL

;Set baut rate   9600==51,  19200==25
        ldi   tempL,high(25)
        ldi   tempH,low(25)
        out   UBRRH,tempL
        out   UBRRL,tempH
;Enable receiver and Transmitter
        ldi   tempL,0b10011000
        out   UCSRB,tempL
;Set Frame format: 8 Data bits, 1 stop bit,
        ldi   tempL,0b10000111
        out   UCSRC,tempL
;Configure Analog to Digital Converter for ADC0
        ldi   tempL,0b01000000
        out   ADMUX,tempL
        ldi   tempL,0b11001000
        out   ADCSRA,tempL
;Configure Timer 1
        ldi   tempL,0b00100100
        out   TIMSK,tempL
        ldi   tempL,$3
        out   TCCR1B,tempL

        ldi   tempH,$F6           ;Timer overflow after 20ms
        ldi   tempL,$3F
        out   TCNT1H,tempH
        out   TCNT1L,tempL

        ldi   tempL,$00       ;Set the outputs to DAC to zero and CS to 1
        out   PORTB,tempL
        cbi   PORTD,5
        cbi   PORTD,6
        cbi   PORTD,7
        sbi   PORTC,CS
        cbi   PORTC,WR1
```

```
        ldi     ZL,low(KP_factor)       ;load from EEPROM KP
        ldi     ZH,high(KP_factor)
        rcall   EERead
        sts     kpH,tempH               ;store KP to RAM
        sts     kpL,tempL
        ldi     ZL,low(KI_factor)       ;load from EEPROM KI
        ldi     ZH,high(KI_factor)
        rcall   EERead
        sts     kiH,tempH               ;store KI to RAM
        sts     kiL,tempL
        ldi     ZL,low(KD_factor)       ;load from EEPROM KD
        ldi     ZH,high(KD_factor)
        rcall   EERead
        sts     kdH,tempH               ;store KD to RAM
        sts     kdL,tempL
        ldi     ZL,low(SP_factor)       ;load SET POINT from EEPROM
        ldi     ZH,high(SP_factor)
        rcall   EERead
        sts     set_pH,tempH            ;store SET POINT to RAM
        sts     set_pL,tempL
        ldi     ZL,low(F_factor)        ;load Friquency Counter from EEPROM
        ldi     ZH,high(F_factor)
        rcall   EERead
        sts     f_pid,tempL             ;store Friquency Counter to RAM
        ldi     ZL,low(SEND_factor)     ;Store to send PID out to PC to EEPROM
        ldi     ZH,high(SEND_factor)
        rcall   EERead
        sts     to_send_PID,tempL
        ldi     tempL,$0
        sts     errorH,tempL            ;Load 0 to error
        sts     errorL,tempL
        sts     sum_errorH,tempL        ;Load 0 to sum error
        sts     sum_errorL,tempL
        sts     last_valueH,tempL       ;Load 0 to last value
        sts     last_valueL,tempL
        sts     count_f,tempL
        sei     ;Enable global Interrupts
MAIN:
        nop
        nop
        nop
        nop
        nop
        nop
        rjmp    MAIN
```

```
;//For timer overflow after 10ms  : TCNT1H=$FB, TCNT1L=$1F
TIMER_1_OVF:
                cli
                ldi     tempL,$1F               ;Timer overflow after 10ms
                ldi     tempH,$FB
                out     TCNT1H,tempH
                out     TCNT1L,tempL
                lds     temp_bL,count_f
                inc     temp_bL
                sts     count_f,temp_bL
                lds     tempL,f_pid
                cp      temp_bL,tempL
                brlo    next
                ldi     temp_bL,$00
                sts     count_f,temp_bL
                rcall   PID
                rcall   OUT_DAC
                lds     rmp,to_send_PID
                cpi     rmp,0
                breq    next
                ldi     temp_cL,STX             ;Load biginning of the message to send
                ldi     temp_cH,ETX             ;Load end of the message to send
                rcall   SUB_S_send_FBCK
next:           nop
                sei
                reti

OUT_DAC:
                lds     tempL,pidL
                lds     tempH,pidH
                cbi     PORTC,WR1
                ldi      tempH, 0
                out     PORTB, tempH
                cbi     PORTD,5
                cbi     PORTD,6
                cbi     PORTD,7
                sbrc    tempL,0
                sbi     PORTB,2
                sbrc    tempL,1
                sbi     PORTB,3
                sbrc    tempL,2
                sbi     PORTB,4
                sbrc    tempL,3
                sbi     PORTB,5
                sbrc    tempL,4
```

```
                sbi     PORTD,5
                sbrc    tempL,5
                sbi     PORTD,6
                sbrc    tempL,6
                sbi     PORTD,7
                sbrc    tempL,7
                sbi     PORTB,0
                nop
                nop
                nop
                cbi     PORTC,CS
                nop
                nop
                nop
                sbi     PORTC,CS
                ret

INT_ADC:        cli
                in      tempL,ADMUX
                cpi     tempL,0b01000000
                brne    POT
                in      tempL,ADCL
                sts     fbL,tempL           ;store low byte to RAM address $100
                in      tempH,ADCH
                sts     fbH,tempH           ;Store high byte to RAM address $101
                sbic    PIND,PC_POT
                rjmp    L001
                ldi     tempL,0b01000101
                out     ADMUX,tempL
                ldi     tempL,0b11001000
                out     ADCSRA,tempL
                sei
                reti
POT:            in      tempL,ADCL
                sts     set_pL,tempL        ;store low byte to RAM address $11d
                in      tempH,ADCH
                sts     set_pH,tempH        ;Store high byte to RAM address $11c

L001:           ldi     tempL,0b01000000
                out     ADMUX,tempL
                ldi     tempL,0b11001000
                out     ADCSRA,tempL
                sei
                reti
```

```
PID:
; calculate max_error
                cls                     ;Clear Interrupt
                lds     ZH,kpH          ;Load parameter KP from RAM
                lds     ZL,kpL
                adiw    ZH:ZL,1         ;Add 1 to KP
                mov     temp_bL,ZL      ;Move KP to R19:R18
                mov     temp_bH,ZH
                ldi     tempL,low(max_int)  ;Load max int
                ldi     tempH,high(max_int)
                rcall   div_16bit       ;Divade max_int / (KP+1)
                sts     max_errorH,tempH    ;Store Result to max_error in RAM
                sts     max_errorL,tempL
;Calculate max sum error
                ldi     tempL,low(max_int)  ;Load max_int to R17:R16
                ldi     tempH,high(max_int)
                sts     max_sum_errorL,tempL ;Store max_int to max_sum_error in RAM
                sts     max_sum_errorH,tempH
;calculate error
                lds     tempL,set_pL    ;Load set point from RAM
                lds     tempH,set_pH
                lds     temp_bL,fbL     ;Load feedback from RAM
                lds     temp_bH,fbH
                rcall   sub_16s         ;Subtract feedback from set point
                sts     errorL,tempL    ;Store result to error in RAM
                sts     errorH,tempH
;Calculate P_term
                tst     tempH
                brmi    n_1             ;If error is negative then jmp to label n_1
                lds     temp_bL,max_errorL  ;Else load max_error to R19:R18
                lds     temp_bH,max_errorH
                cp      tempL,temp_bL   ;compare with the positive max_error
                cpc     tempH,temp_bH
                brcs    n_2             ;If from comparison error > +max_error
                ldi     temp_cL,low(max_int) ;then p_term = max_int
                ldi     temp_cH,high(max_int)
                sts     p_termL,temp_cL ;Store P_term to RAM
                sts     p_termH,temp_cH
                rjmp    n_3             ;and jump to label n_3 to calculate sum_error
n_1:
                ldi     ZL,$ff          ;Calculate the negative max error value
                ldi     ZH,$ff
                lds     temp_bL,max_errorL
                lds     temp_bH,max_errorH
                eor     ZL,temp_bL
```

```
                eor     ZH,temp_bH
                adiw    ZH:ZL,1
                lds     temp_bL,errorL          ;Load error from RAM to R19:R18
                lds     temp_bH,errorH
                cp      temp_bL,ZL              ;Compare error with negative max_error
                cpc     temp_bH,ZH
                brcc    n_2                     ;If error < -max_error then
neg_2:          ldi     tempL,low(n_max_int)
                ldi     tempH,high(n_max_int)
                sts     p_termL,tempL           ;p_term = -max_error else jump to label n_2
                sts     p_termH,tempH
                rjmp    n_3                     ;jump to label n_3 to calculate sum_erro
n_2:    ;Calculate p_term = kp * error
                lds     temp_bL,kpL             ;Load kp from RAM to register
                lds     temp_bH,kpH
                lds     tempL,errorL            ;load error from RAM to register
                lds     tempH,errorH
                rcall   muls16x16_32            ;multiplay kp * error
                tst     r23                     ;If result is negative
                brmi    neg_1                   ;Then jump to neg_1
                brne    nn_2                    ;If not negative check if not zero jmp to nn_2
                cpi     r22,0                   ;Else check the therd byte
                brne    nn_2                    ;If not zero jmp to nn_2
                sbrc    r21,7
                rjmp    nn_2
                rjmp    nn_3                    ;Else jmp to store the result from multiplication
nn_2:           ldi     r20,low(max_int)
                ldi     r21,high(max_int)
nn_3:           sts     p_termL,r20             ;store result from multiplication
                sts     p_termH,r21             ;or $7FFF to RAM p_term
                rjmp    n_3
neg_1:          cpi     r23,$FF
                brne    neg_2
                cpi     r22,$FF
                brne    neg_2
                sbrs    r21,7
                rjmp    neg_2
                rjmp    nn_3
n_3:    ;calculate sum_error + error
                lds     tempL,sum_errorL        ;Load sum_error from RAM to register
                lds     tempH,sum_errorH
                lds     temp_bL,errorL          ;Load error from RAM to register
                lds     temp_bH,errorH
                rcall   add_16s
                tst     tempH
```

```
                brmi    n_4                     ;If result is negatve then jump to label n_4
                lds     temp_cL,max_sum_errorL  ;Load max_sum_error from RAM
                lds     temp_cH,max_sum_errorH
                cp      tempL,temp_cL           ;Compare sum_error with max_sum_error
                cpc     tempH,temp_cH
                brcs    n_6             ;If sum_error < max_sum_error then jump to label n_6
                ldi     tempL,low(max_i_term)
                ldi     tempH,high(max_i_term)
                sts     sum_errorL,tempL        ;else sum_error = max_i_term
                sts     sum_errorH,tempH
                sts     i_termL,tempL           ;and i_term = max_i_term
                sts     i_termH,tempH
                rjmp    n_5                     ;and jump to calculate d_term
n_4:
                lds     temp_cL,max_sum_errorL  ;load max_sum_error from RAM
                lds     temp_cH,max_sum_errorH
                ldi     ZH,$ff                  ;Calculate the value of the
                ldi     ZL,$ff                  ;negative max_sum_error
                eor     ZL,temp_cL
                eor     ZH,temp_cH
                adiw    ZH:ZL,1
                cp      tempL,ZL                ;Compare sum_error with -max_sum_error
                cpc     tempH,ZH
                brcc    n_6             ;If sum_error >= -max_sum_error jump to label n_6
                ldi     ZL,low(-max_i_term)     ;Elsesum_error=-max_iterm,i_term=max_iterm
                ldi     ZH,high(-max_i_term)    ;(-max_i_term)
                sts     sum_errorL,ZL
                sts     sum_errorH,ZH
neg_6:          sts     i_termL,ZL                      ;
                sts     i_termH,ZH
                rjmp    n_5                     ;and jump to calculate d_term
n_6:            sts     sum_errorL,tempL        ;Store sum_error to RAM
                sts     sum_errorH,tempH
                lds     r18,kiL                 ;and calculate i_term=ki * sum_error
                lds     r19,kiH
                rcall   muls16x16_32
                tst     r23                     ;If result is negative
                brmi    neg_i                   ;Then jump to neg_i
                brne    nn_6                    ;If not negative check if not zero jmp to nn_6
                cpi     r22,0                   ;Else check the therd byte
                brne    nn_6                    ;If not zero jmp to nn_6
                sbrc    r21,7
                rjmp    nn_6
                rjmp    nn_7                    ;Else jmp to store the result from multiplication
```

```
nn_6:           ldi     tempL,low(max_int)      ;(max_i_term)
                ldi     tempH,high(max_int)     ;(max_i_term)
                sts     i_termL,tempL           ;store result from multiplication
                sts     i_termH,tempH           ;or $7FFF to RAM i_term
                rjmp    n_5
nn_7:           sts     i_termL,r20             ;store result from multiplication
                sts     i_termH,r21             ;to RAM i_term
                rjmp    n_5
neg_i:          cpi     r23,$FF
                brne    neg_8
                cpi     r22,$FF
                brne    neg_8
                sbrs    r21,7
                rjmp    neg_8
                rjmp    nn_7
neg_8:          ldi     tempL,low(n_max_int)
                ldi     tempH,high(n_max_int)
                sts     i_termL,tempL           ;store to RAM i_term
                sts     i_termH,tempH
n_5: ;calculate d_term
                lds     tempL,last_valueL       ;Load last_value from RAM to register
                lds     tempH,last_valueH
                lds     temp_bL,fbL             ;load feedback value from RAM to register
                lds     temp_bH,fbH
                sts     last_valueL,temp_bL     ;Store feedback to last_value
                sts     last_valueH,temp_bH
                rcall   sub_16s                 ;Subtract last_value - feedback
                lds     r18,kdL         ;Calculated_term=KD*(last_valua-process_value
                lds     r19,kdH
                rcall   muls16x16_32
                sts     d_termL,r20             ;Store calculated d_term to RAM
                sts     d_termH,r21
;calculate result p_term + i_term + d_term
                lds     tempL,p_termL           ;Load p_term from RAM to register
                lds     tempH,p_termH
                lds     temp_bL,i_termL         ;Load i_term from RAM to register
                lds     temp_bH,i_termH
                rcall   add_16s                 ;add p_term and i_term
                lds     temp_bL,d_termL         ;Load d_term from RAM to register
                lds     temp_bH,d_termH
                rcall   add_16s                 ;Add result(p_term+i_term) and d_term
                ldi     temp_bL,low(128)
                ldi     temp_bH,high(128)
                rcall   div_16bit
                ldi     temp_bL,low(255)
```

```
                ldi     temp_bH,high(255)
                rcall   add_16s
                ldi     temp_bL,low(2)
                ldi     temp_bH,high(2)
                rcall   div_16bit
                sts     pidL,tempL          ;Store result of PID to RAM
                sts     pidH,tempH
n_8:            ret

;Serial comunication with the PC
RECEIVE_COMPLEET:
                cli
                sbis    UCSRA,RXC
                rjmp    RECEIVE_COMPLEET
                in      rs_read,UDR
                cpi     rs_read,STX
                breq    L_STX
                cpi     rs_read,ETX
                breq    L_ETX_1
//STORE BYTE TO RAMM
                lds     ZH,address_H        //Load curent address from RAM
                lds     ZL,address_L
                st      Z+,rs_read
                sts     address_H,ZH
                sts     address_L,ZL
                sei
                reti

L_STX:
                ldi     ZH,high(code_byte)          //Load start address
                ldi     ZL,low(code_byte)
                ldi     tempL,0
                st      Z,tempL
                std     Z+1,tempL
                std      Z+2,tempL
                std     Z+3,tempL
                std      Z+4,tempL
                std     Z+5,tempL
                std     Z+6,tempL
                std     Z+7,tempL
                sts     address_H,ZH        //Store curent address to RAM
                sts     address_L,ZL
                sei
                reti
```

```
L_ETX_1:    rjmp    L_ETX

ACKNOL:
            ldi     temp_cL,STX         ;Load biginning of the message to send
            ldi     temp_cH,ETX         ;Load end of the message to send
A1:         sbis    UCSRA,UDRE
            rjmp    A1
            out     UDR,temp_cL
            ldi     temp_bL,ACK
A4:         sbis    UCSRA,UDRE
            rjmp    A4
            out     UDR,temp_bL
A5:         sbis    UCSRA,UDRE
            rjmp    A5
            out     UDR,temp_cH
            ret

NACK:
            ldi     ZH,high(code_byte)  //Load start address
            ldi     ZL,low(code_byte)
            sts     address_H,ZH        //Store curent address to RAM
            sts     address_L,ZL
            ldi     temp_cL,STX         ;Load biginning of the message to send
            ldi     temp_cH,ETX         ;Load end of the message to send
NA1:        sbis    UCSRA,UDRE
            rjmp    NA1
            out     UDR,temp_cL
            ldi     temp_bL,NAK
NA2:        sbis    UCSRA,UDRE
            rjmp    NA2
            out     UDR,temp_bL
            ldi     temp_bL,$91         ;dec 145
NA3:        sbis    UCSRA,UDRE
            rjmp    NA3
            out     UDR,temp_bL
            ldi     temp_bL,$9E         ;dec 158
NA4:        sbis    UCSRA,UDRE
            rjmp    NA4
            out     UDR,temp_bL
NA5:        sbis    UCSRA,UDRE
            rjmp    NA5
            out     UDR,temp_cH
            sei
            reti
```

```
L_ETX:      ldi     ZH,high(code_byte)    //Load start address
            ldi     ZL,low(code_byte)
            lds     tempL,address_L
            lds     tempH,address_H
            sub     tempL,ZL
            sbc     tempH,ZH
            subi    tempL,2               //take out the tow bytes of CRC
            ldi     ZH,high(code_byte)    //Load start address
            ldi     ZL,low(code_byte)
            rcall   crc_culculate         //call calculate CRC
            lds     ZH,address_H          //Load curent address from RAM
            lds     ZL,address_L
            ld      temp_bL,-Z
            ld      temp_bH,-Z
            cp      tempL,temp_bL
            brne    NACK
            cp      tempH,temp_bH
            brne    NACK
            rcall   ACKNOL
            ldi     ZH,high(code_byte)    //Load start address
            ldi     ZL,low(code_byte)
            sts     address_H,ZH          //Store curent address to RAM
            sts     address_L,ZL
            ld       temp_bL,Z
            adiw    ZH:ZL,6
            cpi     temp_bL,receive_KP
            breq    LABEL_R_KP
            cpi     temp_bL,receive_KI
            breq    LABEL_R_KI
            cpi     temp_bL,receive_KD
            breq    LABEL_R_KD
            cpi     temp_bL,receive_SP
            breq    LABEL_R_SP
            cpi     temp_bL,receive_FPID
            breq    LABEL_R_FPID
            cpi     temp_bL,rec_to_send
            breq    LABEL_R_rec_to_send
            cpi     temp_bL,rec_out
            breq    sssss
            rjmp    LABEL_SEND

LABEL_R_KP: rcall   Asc5ToBin2
            mov     tempH,rBin1H
            mov     tempL,rBin1L
            sts     kpL,tempL             ;store KP to RAM
```

```
                sts     kpH,tempH
                ldi     ZL,low(KP_factor)       ;Store KP to EEPROM
                ldi     ZH,high(KP_factor)
                rcall   EEWrite
                sei
                reti

LABEL_R_KI:     rcall   Asc5ToBin2
                mov     tempH,rBin1H
                mov     tempL,rBin1L
                sts     kiL,tempL               ;store KI to RAM
                sts     kiH,tempH
                ldi     ZL,low(KI_factor)       ;Store KI to EEPROM
                ldi     ZH,high(KI_factor)
                rcall   EEWrite
                sei
                reti

LABEL_R_KD:     rcall   Asc5ToBin2
                mov     tempH,rBin1H
                mov     tempL,rBin1L
                sts     kdL,tempL               ;store KD to RAM
                sts     kdH,tempH
                ldi     ZL,low(KD_factor)       ;Store KD to EEPROM
                ldi     ZH,high(KD_factor)
                rcall   EEWrite
                sei
                reti

LABEL_R_SP:     rcall   Asc5ToBin2
                mov     tempH,rBin1H
                mov     tempL,rBin1L
                sts     set_pL,tempL            ;store SET POINT to RAM
                sts     set_pH,tempH
                ldi     ZL,low(SP_factor)       ;Store SET POINT to EEPROM
                ldi     ZH,high(SP_factor)
                rcall   EEWrite
                sei
                reti

sssss:          rjmp    LABEL_R_rec_out

LABEL_R_FPID:
                rcall   Asc5ToBin2
                mov     tempH,rBin1H
```

```
                mov     tempL,rBin1L
                sts     f_pid,tempL             ;store Friquency for PID to RAM
                ldi     ZL,low(F_factor)        ;Store Friquency for PID to EEPROM
                ldi     ZH,high(F_factor)
                rcall   EEWrite
                sei
                reti

LABEL_R_rec_to_send:
                rcall   Asc5ToBin2
                mov     tempH,rBin1H
                mov     tempL,rBin1L
                sts     to_send_PID,tempL       ;store to send PID out to PC to RAM
                ldi     ZL,low(SEND_factor)     ;Store to send PID out to PC to EEPROM
                ldi     ZH,high(SEND_factor)
                rcall   EEWrite
                sei
                reti

LABEL_R_rec_out:
                rcall   Asc5ToBin2
                mov     tempH,rBin1H
                mov     tempL,rBin1L
                sts     pidL,tempL              ;store PID_out from PC to RAM
                sts     pidH,tempH
                rcall   OUT_DAC
                sei
                reti

LABEL_SEND:
                ldi     temp_cL,STX             ;Load biginning of the message to send
                ldi     temp_cH,ETX             ;Load end of the message to send
                cpi     temp_bL,send_KP
                breq    LABEL_S_KP
                cpi     temp_bL,send_KI
                breq    LABEL_S_KI
                cpi     temp_bL,send_KD
                breq    LABEL_S_KD
                cpi     temp_bL,send_SP
                breq    LABEL_S_SP
                cpi     temp_bL,send_FPID
                breq    LABEL_S_FPID
                cpi     temp_bL,send_OUT_PID
                breq    LABEL_S_OUT_PID
                cpi     temp_bL,send_FBCK
```

```
                breq    LABEL_S_send_FBCK
                cpi     temp_bL,send_ALL
                breq    LABEL_S_send_ALL
                sei
                reti

LABEL_S_KP:
                rcall   SUB_S_KP
                sei
                reti
LABEL_S_KI:
                rcall   SUB_S_KI
                sei
                reti
LABEL_S_KD:
                rcall   SUB_S_KD
                sei
                reti
LABEL_S_SP:
                rcall   SUB_S_SP
                reti

LABEL_S_FPID:
                rcall   SUB_S_FPID
                sei
                reti
LABEL_S_OUT_PID:
                rcall   SUB_S_OUT_PID
                sei
                reti
LABEL_S_send_FBCK:
                rcall   SUB_S_send_FBCK
                sei
                reti
LABEL_S_send_ALL:
                rcall   SUB_S_KP
                rcall   SUB_S_KI
                rcall   SUB_S_KD
                rcall   SUB_S_SP
                rcall   SUB_S_FPID
                rcall   SUB_S_OUT_PID
                rcall   SUB_S_send_FBCK
                sei
                reti
```

```
SUB_S_KP:   ldi     temp_cL,STX
L1:         sbis    UCSRA,UDRE
            rjmp    L1
            out     UDR,temp_cL
            ldi     temp_bL,send_KP
L2:         sbis    UCSRA,UDRE
            rjmp    L2
            out     UDR,temp_bL
            lds     rBin1L,kpL              ;Load KP from RAM
            lds     rBin1H,kpH
            rcall   Bin2ToAsc5
            ldi     ZL,low(asc5)
            ldi     ZH,high(asc5)
            ldi     rmp,5
L3:         ld      tempL,Z+
L4:         sbis    UCSRA,UDRE
            rjmp    L4
            out     UDR,tempL
            dec     rmp
            brne    L3
            ldi     ZL,low(asc5-1)
            ldi     ZH,high(asc5-1)
            ldi     temp_bL,send_KP
            st      Z,temp_bL
            ldi     tempL,6
            rcall   crc_culculate
L5:         sbis    UCSRA,UDRE
            rjmp    L5
            out     UDR,tempH
L6:         sbis    UCSRA,UDRE
            rjmp    L6
            out     UDR,tempL
            ldi     temp_cH,ETX
L7:         sbis    UCSRA,UDRE
            rjmp    L7
            out     UDR,temp_cH
            ret
SUB_S_KI:
            ldi     temp_cL,STX
L11:        sbis    UCSRA,UDRE
            rjmp    L11
            out     UDR,temp_cL
            ldi     temp_bL,send_KI
L12:        sbis    UCSRA,UDRE
            rjmp    L12
```

```
                out     UDR,temp_bL
                lds     rBin1L,kiL              ;Load KI from RAM
                lds     rBin1H,kiH
                rcall   Bin2ToAsc5
                ldi     ZL,low(asc5)
                ldi     ZH,high(asc5)
                ldi     rmp,5
L13:            ld      tempL,Z+
L14:            sbis    UCSRA,UDRE
                rjmp    L14
                out     UDR,tempL
                dec     rmp
                brne    L13
                ldi     ZL,low(asc5-1)
                ldi     ZH,high(asc5-1)
                ldi     temp_bL,send_KI
                st      Z,temp_bL
                ldi     tempL,6
                rcall   crc_culculate
L15:            sbis    UCSRA,UDRE
                rjmp    L15
                out     UDR,tempH
L16:            sbis    UCSRA,UDRE
                rjmp    L16
                out     UDR,tempL
                ldi     temp_cH,ETX
L17:            sbis    UCSRA,UDRE
                rjmp    L17
                out     UDR,temp_cH
                ret
SUB_S_KD:       ldi     temp_cL,STX
L21:            sbis    UCSRA,UDRE
                rjmp    L21
                out     UDR,temp_cL
                ldi     temp_bL,send_KD
L22:            sbis    UCSRA,UDRE
                rjmp    L22
                out     UDR,temp_bL
                lds     rBin1L,kdL              ;Load KD from RAM
                lds     rBin1H,kdH
                rcall   Bin2ToAsc5
                ldi     ZL,low(asc5)
                ldi     ZH,high(asc5)
                ldi     rmp,5
L23:            ld      tempL,Z+
```

```
L24:    sbis    UCSRA,UDRE
        rjmp    L24
        out     UDR,tempL
        dec     rmp
        brne    L23
        ldi     ZL,low(asc5-1)
        ldi     ZH,high(asc5-1)
        ldi     temp_bL,send_KD
        st      Z,temp_bL
        ldi     tempL,6
        rcall   crc_culculate
L25:    sbis    UCSRA,UDRE
        rjmp    L25
        out     UDR,tempH
L26:    sbis    UCSRA,UDRE
        rjmp    L26
        out     UDR,tempL
        ldi     temp_cH,ETX
L27:    sbis    UCSRA,UDRE
        rjmp    L27
        out     UDR,temp_cH
        ret
SUB_S_SP:
        ldi     temp_cL,STX
L31:    sbis    UCSRA,UDRE
        rjmp    L31
        out     UDR,temp_cL
        ldi     temp_bL,send_SP
L32:    sbis    UCSRA,UDRE
        rjmp    L32
        out     UDR,temp_bL
        lds     rBin1L,set_pL           ;Load SP from RAM
        lds     rBin1H,set_pH
        rcall   Bin2ToAsc5
        ldi     ZL,low(asc5)
        ldi     ZH,high(asc5)
        ldi     rmp,5
L33:    ld      tempL,Z+
L34:    sbis    UCSRA,UDRE
        rjmp    L34
        out     UDR,tempL
        dec     rmp
        brne    L33
        ldi     ZL,low(asc5-1)
        ldi     ZH,high(asc5-1)
```

```
                ldi     temp_bL,send_SP
                st      Z,temp_bL
                ldi     tempL,6
                rcall   crc_culculate
L35:            sbis    UCSRA,UDRE
                rjmp    L35
                out     UDR,tempH
L36:            sbis    UCSRA,UDRE
                rjmp    L36
                out     UDR,tempL
                ldi     temp_cH,ETX
L37:            sbis    UCSRA,UDRE
                rjmp    L37
                out     UDR,temp_cH
                ret

SUB_S_FPID:
                ldi     temp_cL,STX
L41:            sbis    UCSRA,UDRE
                rjmp    L41
                out     UDR,temp_cL
                ldi     temp_bL,send_FPID
L42:            sbis    UCSRA,UDRE
                rjmp    L42
                out     UDR,temp_bL
                lds     rBin1L,f_pid            ;Load f_pid from RAM
                ldi     rBin1H,0
                rcall   Bin2ToAsc5
                ldi     ZL,low(asc5)
                ldi     ZH,high(asc5)
                ldi     rmp,5
L43:            ld      tempL,Z+
L44:            sbis    UCSRA,UDRE
                rjmp    L44
                out     UDR,tempL
                dec     rmp
                brne    L43
                ldi     ZL,low(asc5-1)
                ldi     ZH,high(asc5-1)
                ldi     temp_bL,send_FPID
                st      Z,temp_bL
                ldi     tempL,6
                rcall   crc_culculate
L45:            sbis    UCSRA,UDRE
                rjmp    L45
```

```
                out     UDR,tempH
L46:            sbis    UCSRA,UDRE
                rjmp    L46
                out     UDR,tempL
                ldi     temp_cH,ETX
L47:            sbis    UCSRA,UDRE
                rjmp    L47
                out     UDR,temp_cH
                ret

SUB_S_OUT_PID:
                ldi     temp_cL,STX
L51:            sbis    UCSRA,UDRE
                rjmp    L51
                out     UDR,temp_cL
                ldi     temp_bL,send_OUT_PID
L52:            sbis    UCSRA,UDRE
                rjmp    L52
                out     UDR,temp_bL
                lds     rBin1L,pidL             ;Load PID OUTPUT from RAM
                lds     rBin1H,pidH
                rcall   Bin2ToAsc5
                ldi     ZL,low(asc5)
                ldi     ZH,high(asc5)
                ldi     rmp,5
L53:            ld      tempL,Z+
L54:            sbis    UCSRA,UDRE
                rjmp    L54
                out     UDR,tempL
                dec     rmp
                brne    L53
                ldi     ZL,low(asc5-1)
                ldi     ZH,high(asc5-1)
                ldi     temp_bL,send_OUT_PID
                st      Z,temp_bL
                ldi     tempL,6
                rcall   crc_culculate
L55:            sbis    UCSRA,UDRE
                rjmp    L55
                out     UDR,tempH
L56:            sbis    UCSRA,UDRE
                rjmp    L56
                out     UDR,tempL
                ldi     temp_cH,ETX
L57:            sbis    UCSRA,UDRE
```

```
                rjmp    L57
                out     UDR,temp_cH
                ret

SUB_S_send_FBCK:
                ldi     temp_cL,STX
L61:            sbis    UCSRA,UDRE
                rjmp    L61
                out     UDR,temp_cL
                ldi     temp_bL,send_FBCK
L62:            sbis    UCSRA,UDRE
                rjmp    L62
                out     UDR,temp_bL
                lds     rBin1L,fBL              ;Load FEDBACK from RAM
                lds     rBin1H,fBH
                rcall   Bin2ToAsc5
                ldi     ZL,low(asc5)
                ldi     ZH,high(asc5)
                ldi     rmp,5
L63:            ld      tempL,Z+
L64:            sbis    UCSRA,UDRE
                rjmp    L64
                out     UDR,tempL
                dec     rmp
                brne    L63
                ldi     ZL,low(asc5-1)
                ldi     ZH,high(asc5-1)
                ldi     temp_bL,send_FBCK
                st      Z,temp_bL
                ldi     tempL,6
                rcall   crc_culculate
L65:            sbis    UCSRA,UDRE
                rjmp    L65
                out     UDR,tempH
L66:            sbis    UCSRA,UDRE
                rjmp    L66
                out     UDR,tempL
                ldi     temp_cH,ETX
L67:            sbis    UCSRA,UDRE
                rjmp    L67
                out     UDR,temp_cH
                ret
SUB_S_sss:
                ldi     temp_cL,STX
L81:            sbis    UCSRA,UDRE
```

```
        rjmp    L81
        out     UDR,temp_cL
        ldi     temp_bL,send_SSS
L82:    sbis    UCSRA,UDRE
        rjmp    L82
        out     UDR,temp_bL
        mov     rBin1L,tempL
        mov     rBin1H,tempH
        rcall   Bin2ToAsc5
        ldi     ZL,low(asc5)
        ldi     ZH,high(asc5)
        ldi     rmp,5
L83:    ld      tempL,Z+
L84:    sbis    UCSRA,UDRE
        rjmp    L84
        out     UDR,tempL
        dec     rmp
        brne    L83
        ldi     ZL,low(asc5-1)
        ldi     ZH,high(asc5-1)
        ldi     temp_bL,send_SSS
        st      Z,temp_bL
        ldi     tempL,6
        rcall   crc_culculate
L85:    sbis    UCSRA,UDRE
        rjmp    L85
        out     UDR,tempH
L86:    sbis    UCSRA,UDRE
        rjmp    L86
        out     UDR,tempL
        ldi     temp_cH,ETX
L87:    sbis    UCSRA,UDRE
        rjmp    L87
        out     UDR,temp_cH
        ret
```

```
;***************************************************************************
;*      Load data from EEPROM to tempL and tempH
;*      The address indicated from ZL and ZH
;***************************************************************************
EERead:
            sbic    EECR,EEWE       ;if EEWE not clear
            rjmp    EERead          ;wait more
            out     EEARH,ZH        ;output address high byte,
            out     EEARL,ZL        ;output address low byte
            sbi     EECR,EERE       ;set EEPROM Read strobe
            in      tempH,EEDR      ;get data
            adiw    ZL:ZH,1
L_1:        sbic    EECR,EEWE
            rjmp    L_1
            out     EEARH,ZH
            out     EEARL,ZL
            sbi     EECR,EERE       ;set EEPROM Read strobe
            in      tempL,EEDR      ;get data
            ret


;***************************************************************************
;*      Load data from tempL and tempH to EEPROM
;*      The address indicated from ZL and ZH
;***************************************************************************
EEWrite:
            cli                     ;disable global interrupts
            sbic    EECR,EEWE       ;if EEWE not clear
            rjmp    EEWrite         ;wait more
            out     EEARH,ZH        ;output address high byte
            out     EEARL,ZL        ;output address low byte
            out     EEDR,tempH      ;output data
            sbi     EECR,EEMWE      ;set master write enable
            sbi     EECR,EEWE       ;set EEPROM Write strobe
            adiw    ZL:ZH,1
L_2:        sbic    EECR,EEWE       ;if EEWE not clear
            rjmp    L_2             ;wait more
            out     EEARH,ZH        ;output address high byte
            out     EEARL,ZL        ;output address low byte
            out     EEDR,tempL      ;output data
            sbi     EECR,EEMWE      ;set master write enable
            sbi     EECR,EEWE       ;set EEPROM Write strobe
            ret
```

```
;***************************************************************************
;*      Unsigned multiply of two 16bits numbers with 32bits result.
;*      r19:r18:r17:r16 = r23:r22 * r21:r20
;***************************************************************************
mul16x16_32:
.def    mc16sL          =r16            ;multiplicand low byte
.def    mc16sH          =r17            ;multiplicand high byte
.def    mp16sL          =r18            ;multiplier low byte
.def    mp16sH          =r19            ;multiplier high byte
.def    m16s0           =r20            ;result byte 0 (LSB)
.def    m16s1           =r21            ;result byte 1
.def    m16s2           =r22            ;result byte 2
.def    m16s3           =r23            ;result byte 3 (MSB)
                clr     r2
                mul     mc16sH, mp16sH      ;AH * BH
                movw    m16s3:m16s2, r1:r0
                mul     mc16sL, mp16sL      ;AL * BL
                movw    m16s1:m16s0, r1:r0
                mul     mc16sH, mp16sL      ;AH * BL
                add     m16s1, r0
                adc     m16s2, r1
                adc     m16s3, r2
                mul     mp16sH, mc16sL      ;BH * AL
                add     m16s1, r0
                adc     m16s2, r1
                adc     m16s3, r2
                ret



;***************************************************************************
;*      Signed multiply of two 16bits numbers with 32bits result.
;*      R23:R22:R21:R20 = R17:R16 * R19:R18
;***************************************************************************
muls16x16_32:
.def    mc16sL          =r16            ;multiplicand low byte
.def    mc16sH          =r17            ;multiplicand high byte
.def    mp16sL          =r18            ;multiplier low byte
.def    mp16sH          =r19            ;multiplier high byte
.def    m16s0           =r20            ;result byte 0 (LSB)
.def    m16s1           =r21            ;result byte 1
.def    m16s2           =r22            ;result byte 2
.def    m16s3           =r23            ;result byte 3 (MSB)
        clr             r2
        muls    mc16sH,mp16sH       ;(signed)AH * (signed)BH
```

```
        movw    m16s3:m16s2,r1:r0
        mul     mc16sL,mp16sL       ;AL * BL
        movw    m16s1:m16s0,r1:r0
        mulsu   mc16sH,mp16sL       ;(signed)AH * BL
        sbc     m16s3,r2
        add     m16s1,r0
        adc     m16s2,r1
        adc     m16s3,r2
        mulsu   mp16sH,mc16sL       ;(signed)BH * AL
        sbc     m16s3,r2
        add     m16s1,r0
        adc     m16s2,r1
        adc     m16s3,r2
        ret

;*****************************************************************************
;*    8/8 bit signed division
;*****************************************************************************
div_8bit:
.def    d8s             =r18
.def    drem8s          =r19
.def    dres8s          =r16        ;result
.def    dd8s            =r16        ;dividend /d?a??et???
.def    dv8s            =r17        ;divisor
.def    dcnt8s          =r20        ;Loop counter

div8s:          mov     d8s,dd8s
                eor     d8s,dv8s
                sbrc    dv8s,7
                neg     dv8s
                sbrc    dd8s,7
                neg     dd8s
                sub     drem8s,drem8s
                ldi     dcnt8s,9
d8s_1:          rol     dd8s
                dec     dcnt8s
                brne    d8s_2
                sbrc    d8s,7
                neg     dres8s
                ret
d8s_2:          rol     drem8s
                sub     drem8s,dv8s
                brcc    d8s_3
                add     drem8s,dv8s
                clc
```

```
                rjmp    d8s_1
d8s_3:          sec
                rjmp    d8s_1
;*****************************************************************************
;*    16/16 bit signed division    r17:r16/r19:r18                           *
;*****************************************************************************
div_16bit:
.def    d16s            =r21                ;sign register
.def    drem16sL        =r22                ;remainder low byte
.def    drem16sH        =r23                ;remainder high byte
.def    dres16sL        =r16                ;result low byte
.def    dres16sH        =r17                ;result high byte
.def    dd16sL          =r16                ;dividend low byte
.def    dd16sH          =r17                ;dividend high byte
.def    dv16sL          =r18                ;divisor low byte
.def    dv16sH          =r19                ;divisor high byte
.def    dcnt16s         =r20                ;loop counter
div16s:         mov     d16s,dd16sH
                eor     d16s,dv16sH
                sbrs    dd16sH,7
                rjmp    d16s_1
                com     dd16sH
                com     dd16sL
                subi    dd16sL,low(-1)
                sbci    dd16sL,high(-1)
d16s_1:         sbrs    dv16sH,7
                rjmp    d16s_2
                com     dv16sH
                com     dv16sL
                subi    dv16sL,low(-1)
                sbci    dv16sL,high(-1)
d16s_2:         clr     drem16sL
                sub     drem16sH,drem16sH
                ldi     dcnt16s,17
d16s_3:         rol     dd16sL
                rol     dd16sH
                dec     dcnt16s
                brne    d16s_5
                sbrs    d16s,7
                rjmp    d16s_4
                com     dres16sH
                com     dres16sL
                subi    dres16sL,low(-1)
                sbci    dres16sH,high(-1)
d16s_4:         ret
```

```
d16s_5:     rol     drem16sL
            rol     drem16sH
            sub     drem16sL,dv16sL
            sbc     drem16sH,dv16sH
            brcc    d16s_6
            add     drem16sL,dv16sL
            adc     drem16sH,dv16sH
            clc
            rjmp    d16s_3
d16s_6:     sec
            rjmp    d16s_3

; Bin2ToAsc5
; converts a 16-bit-binary to a 5 digit ASCII-coded decimal
Bin2ToAsc5:
            ldi     ZL,low(asc5)
            ldi     ZH,high(asc5)
            rcall   Bin2ToBcd5          ;convert binary to BCD
            ldi     rmp,5;4             ;Counter is 4 leading digits
            mov     loop_counter,rmp
Bin2ToAsc5c:
            ld      rmp,Z               ;read a BCD digit
            subi    rmp,-'0' ;Add ASCII-0
            st      Z+,rmp              ;store and inc pointer
            dec     loop_counter        ;more chars?
            brne    Bin2ToAsc5c         ;yes, go on
            sbiw    ZL,5                ;Pointer to beginning of the BCD
            ret

; Bin2ToBcd5
; converts a 16-bit-binary to a 5-digit-BCD
Bin2ToBcd5:
            ldi     rmp,HIGH(10000)     ;Start with tenthousands
            mov     temp_bH,rmp
            ldi     rmp,LOW(10000)
            mov     temp_bL,rmp
            rcall   Bin2ToDigit         ;Calculate digit
            ldi     rmp,HIGH(1000)      ;Next with thousands
            mov     temp_bH,rmp
            ldi     rmp,LOW(1000)
            mov     temp_bL,rmp
            rcall   Bin2ToDigit         ;Calculate digit
            ldi     rmp,HIGH(100)       ;Next with hundreds
            mov     temp_bH,rmp
            ldi     rmp,LOW(100)
```

```
                mov     temp_bL,rmp
                rcall   Bin2ToDigit             ;Calculate digit
                ldi     rmp,HIGH(10)            ;Next with tens
                mov     temp_bH,rmp
                ldi     rmp,LOW(10)
                mov     temp_bL,rmp
                rcall   Bin2ToDigit             ;Calculate digit
                st      z,rBin1L                ;Remainder are ones
                sbiw    ZL,4                    ;Put pointer to first BCD
                ret                             ;and return


; Bin2ToDigit
; converts one decimal digit by continued subraction of a binary coded decimal
Bin2ToDigit:
                clr     rmp                     ;digit count is zero
Bin2ToDigita:
                cp      rBin1H,temp_bH          ;Number bigger than decimal?
                brcs    Bin2ToDigitc            ;MSB smaller than decimal
                brne    Bin2ToDigitb            ;MSB bigger than decimal
                cp      rBin1L,temp_bL          ;LSB bigger or equal decimal
                brcs    Bin2ToDigitc            ;LSB smaller than decimal
Bin2ToDigitb:
                sub     rBin1L,temp_bL          ;Subtract LSB decimal
                sbc     rBin1H,temp_bH          ;Subtract MSB decimal
                inc     rmp                     ;Increment digit count
                rjmp    Bin2ToDigita            ;Next loop
Bin2ToDigitc:
                st      z+,rmp                  ;Save digit and increment
                ret                             ;done

;* Asc5ToBin2
;* converts a 5-digit ASCII to a 16-bit-binary
Asc5ToBin2:
                ldi     YL,1
                ldi     YH,0
                ldi     rmp,6                   ; five chars, one too much
                mov     loop_counter,rmp
                clr     rBin1H                  ;Clear result
                clr     rBin1L
                clt                             ;Clear T-Bit
Asc5ToBin2a:
                dec     loop_counter            ; all chars read?
                breq    Asc5ToBin2d             ;last char
                ld      rmp,-Z                  ;read a char
```

```
Asc5ToBin2b:
                subi    rmp,'0'             ;treat digit
                brcs    Asc5ToBin2e         ;Last char was invalid
                cpi     rmp,10              ;digit > 9
                brcc    Asc5ToBin2e         ;Last char invalid
                mov     tempL,rmp
                ldi     tempH,0
                mov     temp_bL,YL
                mov     temp_bH,YH
                rcall   mul16x16_32
                add     rBin1L,temp_cL
                adc     rBin1H,temp_cH
                ldi     tempL,10
                ldi     tempH,0
                mov     temp_bL,YL
                mov     temp_bH,YH
                rcall   mul16x16_32
                mov     YL,temp_cL
                mov     YH,temp_cH
                brts    Asc5ToBin2e         ;Overflow occurred
                ld      rmp,-Z
                brcc    Asc5ToBin2c         ;no overflow to MSB
                inc     rBin1H              ;Overflow to MSB
                breq    Asc5ToBin2e         ;Overflow of MSB
Asc5ToBin2c:
                dec     loop_counter        ;downcount number of digits
                brne    Asc5ToBin2b         ;convert more chars
Asc5ToBin2d:                                ;End of ASCII number reached ok
                ret
Asc5ToBin2e:                                ;Last char was invalid
                set                         ;Set T-Flag for error
                ret                         ;and return with error condition set

crc_culculate:  cli
                mov     rmp,tempL           ;Num Bytes to calculate CRC
                ldi     tempL,$FF
                ldi     tempH,$FF
                ldi     temp_bL,STX
                eor     tempL,temp_bL
                rcall   crc_ror
crc_L1:         ld      temp_bl,Z+
                eor     tempL,temp_bL
                rcall   crc_ror
                dec      rmp
                brne    crc_L1
```

```
                ldi     temp_bL,ETX
                eor     tempL,temp_bL
                rcall   crc_ror
                ret

crc_ror:        ldi     temp_cL,8
                mov     loop_counter,temp_cL
crc_a1:         sbrs    tempL,0
                rjmp    crc_b
                clc
                ror     tempH
                ror     tempL
                ldi     temp_cL,$01
                ldi     temp_cH,$A0
                eor     tempH,temp_cH
                eor     tempL,temp_cL
                dec     loop_counter
                brne    crc_a1
                ret
crc_b:
                clc
                ror     tempH
                ror     tempL
                dec     loop_counter
                brne    crc_a1
                ret
add_16s:
                tst     tempH
                brpl    as16_1
; neg Z
                tst     temp_bH
                brpl    as16_2
; neg + neg
                add     tempL,temp_bL
                adc     tempH,temp_bH
                brmi    as16_end
                ldi     tempL,$00               ; neg + neg overflow
                ldi     tempH,$80
                ret

as16_2:                                         ; neg + pos
                add     tempL,temp_bL
                adc     tempH,temp_bH
                ret
```

```
as16_3:                                 ; pos + neg
        add     tempL,temp_bL
        adc     tempH,temp_bH
        ret
as16_1:                                 ; pos Z
        tst     temp_bH
        brmi    as16_3
        add     tempL,temp_bL           ; pos + pos
        adc     tempH,temp_bH
        brpl    as16_end
; pos + pos overflow
        ldi     tempL,0xFF
        ldi     tempH,0x7F
as16_end:
        ret

sub_16s:
        tst     tempH
        brpl    sb16_1
; neg Z
        tst     temp_bH
        brpl    sb16_2
; neg - neg
        sub     tempL,temp_bL
        sbc     tempH,temp_bH
        ret


sb16_2:                                 ; neg - pos
        sub     tempL,temp_bL
        sbc     tempH,temp_bH
        brmi    sb16_end
; neg - pos overflow
        ldi     tempL,0x00
        ldi     tempH,0x80
        ret

sb16_3:                                 ; pos - neg
        sub     tempL,temp_bL
        sbc     tempH,temp_bH
        brpl    sb16_end
        ldi     tempL,$FF               ; pos - neg overflow
        ldi     tempH,$7F
        ret
```

```
sb16_1:     tst     temp_bH          ; pos A
            brmi    sb16_3
            sub     tempL,temp_bL    ; pos - pos
            sbc     tempH,temp_bH
sb16_end:   ret


SUB_SSS:
            ldi     temp_cL,STX
S31:        sbis    UCSRA,UDRE
            rjmp    S31
            out     UDR,temp_cL
            ldi     temp_bL,send_SSS
S32:        sbis    UCSRA,UDRE
            rjmp    S32
            out     UDR,temp_bL
            mov     rBin1L,tempL
            mov     rBin1H,tempH
            rcall   Bin2ToAsc5
            ldi     ZL,low(asc5)
            ldi     ZH,high(asc5)
            ldi     rmp,5
S33:        ld      tempL,Z+
S34:        sbis    UCSRA,UDRE
            rjmp    S34
            out     UDR,tempL
            dec     rmp
            brne    S33
            ldi     ZL,low(asc5-1)
            ldi     ZH,high(asc5-1)
            ldi     temp_bL,send_SSS
            st      Z,temp_bL
            ldi     tempL,6
            rcall   crc_culculate
S35:        sbis    UCSRA,UDRE
            rjmp    S35
            out     UDR,tempH
S36:        sbis    UCSRA,UDRE
            rjmp    S36
            out     UDR,tempL
            ldi     temp_cH,ETX
S37:        sbis    UCSRA,UDRE
            rjmp    S37
            out     UDR,temp_cH
            ret
```