ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΑΠΟΜΑΚΡΥΣΜΕΝΟΣ ΕΛΕΓΧΟΣ ΑΣΥΓΧΡΟΝΗΣ ΜΗΧΑΝΗΣ ΜΕΣΩ ΣΥΝΤΟΜΩΝ ΓΡΑΠΤΩΝ ΜΗΝΥΜΑΤΩΝ (SMS) ΜΕ ARDUINO

ΧΑΤΖΟΠΟΥΛΟΣ ΓΕΩΡΓΙΟΣ
ΧΑΤΖΟΠΟΥΛΟΣ ΓΙΑΝΝΗΣ

Επιβλέπουσα Καθηγήτρια **: Δρ. ΜΑΡΙΑ ΙΩΑΝΝΙΔΟΥ**

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τομέας Ηλεκτρικών Βιομηχανικών Διατάξεων και Συστημάτων Αποφάσεων

ΑΠΟΜΑΚΡΥΣΜΕΝΟΣ ΕΛΕΓΧΟΣ ΑΣΥΓΧΡΟΝΗΣ ΜΗΧΑΝΗΣ ΜΕΣΩ ΣΥΝΤΟΜΩΝ ΓΡΑΠΤΩΝ ΜΗΝΥΜΑΤΩΝ (SMS) ΜΕ ARDUINO

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Γεώργιος Κ. Χατζόπουλος
Γιάννης Α. Χατζόπουλος

Επιβλέπουσα : Δρ. Ιωαννίδου Μαρία
Καθηγήτρια Ε.Μ.Π.

ΕΓΚΡΙΘΗΚΕ ΑΠΟ ΤΗΝ ΤΡΙΜΕΛΗ ΕΠΙΤΡΟΠΗ.

.....	.....	.....
Μ. ΙΩΑΝΝΙΔΟΥ	**Π. ΤΣΑΡΑΜΠΑΡΗΣ**	**Ν. ΘΕΟΔΩΡΟΥ**
ΚΑΘΗΓΗΤΡΙΑ Ε.Μ.Π.	**ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.**	**ΚΑΘΗΓΗΤΗΣ Ε.Μ.Π.**

ΑΘΗΝΑ, 2014

……
Γεώργιος Κ. Χατζόπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

……
Γιάννης Α. Χατζόπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © Γεώργιος Κ. Χατζόπουλος, 2014. Με επιφύλαξη παντός δικαιώματος. All rights reserved.
Copyright © Γιάννης Α. Χατζόπουλος, 2014. Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τους συγγραφείς και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

ΠΕΡΙΛΗΨΗ

Σε αυτή την διπλωματική κατασκευάσαμε και προγραμματίσαμε μια διάταξη έτσι ώστε να μπορούμε να ελέγχουμε μια ασύγχρονη μηχανή ή ενδεχομένως και γεννήτρια μέσω σύντομων γραπτών μηνυμάτων. Για την κατασκευή της διάταξης αυτής χρησιμοποιήσαμε την υπολογιστική πλατφόρμα της οικογένειας Arduino καθώς και την μητρική της πλακέτα με τους μικροεπεξεργαστές. Χρησιμοποιήσαμε Arduino καθώς εκτός από το οικονομικό όφελος είναι και μια πλατφόρμα που είναι ευρέως διαδομένη με πολλές υποστηρικτικές εφαρμογές και μια μεγάλη κοινότητα να την υποστηρίζει. Αυτό καθιστά την διπλωματική μας να είναι ευέλικτη από πλευράς υποστήριξης αλλά και επεκτάσεων ακόμη και από τρίτους.

ABSTRACT

In this thesis we assembled and programmed a device that can control a motor or a even a generator through written messages (sms). A microcontroller of the Arduino family was used for the implementation of this device. The Arduino Uno was used, as it is very economic and its programming is based on open source software. Needless to say that the usage of these microcontrollers is wide and with many supportive applications. The above facts lead us to conclude that the designed device is very flexible and can be further developed.

ΚΕΦΑΛΑΙΟ 1Ο ΜΗΧΑΝΕΣ 9

1.1 Εισαγωγή 9
1.1.1 Θερμικές μηχανές 9
1.1.2 Υδραυλικές μηχανές 10
1.1.3 Ηλεκτρικές μηχανές 10

1.2 Ηλεκτρικές μηχανές 11

1.3 Ασύγχρονη μηχανή 12
1.3.1 Κατασκευαστικά στοιχεία 12
1.3.2 Τυλίγματα δρομέα 14

1.4 Ισοδύναμο κύκλωμα επαγωγικού κινητήρα 16

1.5 Μέθοδοι εκκίνησης κινητήρα επαγωγής 22
1.5.1 Εκκίνηση Υ/Δ σε κινητήρες επαγωγής βραχυκυκλωμένου δρομέα 23
1.5.2 Εκκίνηση με αντιστάσεις στον στάτη 24
1.5.3 Εκκίνηση επαγωγικού κινητήρα με αυτομετασχηματιστή 25
1.5.4 Εκκίνηση με μία μόνο αντίσταση στο στάτη (KUSA) 26
1.5.5 Εκκίνηση με ηλεκτρικό ρυθμιστή τάσης 27
1.5.6 Εκκίνηση με μετατροπέα συχνότητας 27
1.5.7 Κατασκευές κινητήρων για βελτίωση των συνθηκών εκκίνησης 27
1.5.8 Ασύγχρονοι κινητήρες με διπλό κλωβό 28
1.5.9 Εκκίνηση και λειτουργία τριφασικού επαγωγικού κινητήρα σε μονοφασικό δίκτυο . 29

ΚΕΦΑΛΑΙΟ 2^{Ο} Η ΙΣΤΟΡΙΑ ΤΩΝ ΜΙΚΡΟΕΠΕΞΕΡΓΑΣΤΩΝ - ΜΙΚΡΟΕΛΕΓΚΤΩΝ 31

2.1 Εισαγωγή 31

2.2 Η εξέλιξη των μικροεπεξεργαστών 31

2.3 Από τον Επεξεργαστή στον Μικροελεγκτή 33

2.4 Τα μικροϋπολογιστικά συστήματα σήμερα 35

ΚΕΦΑΛΑΙΟ 3^{Ο} Ο ΜΙΚΡΟΕΛΕΓΚΤΗΣ AVR 39

Εισαγωγή 39

ΚΕΦΑΛΑΙΟ 4^{Ο} ΔΟΜΗ ΠΡΟΓΡΑΜΜΑΤΟΣ 53

4.1 Βασικές συναρτήσεις 53

4.2 Δομές Ελέγχου 54

4.3 Σύνταξη κώδικα 60

4.4 Αριθμητικοί τελεστές 62

4.5 Τελεστές συγκρίσεως 63

4.6 Λογικοί τελεστές (τελεστές Boole) 63

4.7 Τελεστές δεικτών 63

4.8 Δυαδικοί τελεστές 64

ΚΕΦΑΛΑΙΟ 5° ΣΤΑΘΕΡΕΣ ΚΑΙ ΜΕΤΑΒΛΗΤΕΣ 73

5.1 Σταθερές 73

5.2 Τύποι Δεδομένων Μεταβλητών - Data Types 78

5.3 Μετατροπές μεταβλητών - Type casting 84

5.4 Εμβέλεια μεταβλητών και τα χαρακτηριστικά τους 85

5.5 Ιδιαίτερη ιδιότητα μεταβλητών 87

ΚΕΦΑΛΑΙΟ 6° ΣΥΝΑΡΤΗΣΕΙΣ 89

6.1 Συναρτήσεις Ψηφιακών Εισόδων – Εξόδων (Digital I/O) 90

6.2 Συναρτήσεις Αναλογικών Εισόδων – Εξόδων (Analog I/O) 90

6.3 Advanced I/O 93

6.4 Χρονικές συναρτήσεις 95

6.5 Μαθηματικές συναρτήσεις 96

6.6 Τριγωνομετρικές συναρτήσεις 97

6.7 Συναρτήσεις τυχαίων αριθμών 98

6.8 Συναρτήσεις που αφορούν bits και Bytes 99

6.9 Interrupts (Διακοπές) 101

6.10 Επικοινωνίες 103

6.11 Αντικείμενο String 108

Ο ΚΩΔΙΚΑΣ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ 111

ΣΥΜΠΕΡΑΣΜΑΤΑ 115

ΒΙΒΛΙΟΓΡΑΦΙΑ 116

Κεφάλαιο 1ο Μηχανές

1.1 Εισαγωγή

Μηχανή ονομάζεται οποιοδήποτε εργαλείο ή μέσον που μπορεί να διευκολύνει την ανθρώπινη εργασία ή που χρησιμοποιείται για τη παραγωγή έργου, είτε μεταδίδοντας είτε μετατρέποντας άλλη μορφή ενέργειας σε παραγωγή έργου.

Ανάλογα με τη μορφή της ενέργειας που χρησιμοποιούν οι κινητήριες μηχανές διακρίνονται σε:

- Θερμικές μηχανές ή θερμοκινητήρες: Στην κατηγορία αυτή ανήκουν οι ατμομηχανές, οι ατμοστρόβιλοι, οι αεριοστρόβιλοι, οι βενζινοκινητήρες ή βενζινομηχανές, οι αεριομηχανές, οι πετρελαιομηχανές ή κινητήρες Diesel κ.ά.
- Υδραυλικοί κινητήρες. Στην κατηγορία αυτή οι μηχανές καταναλώνουν υδραυλική ενέργεια.
- Ηλεκτρικοί κινητήρες ή ηλεκτροκινητήρες.

1.1.1 Θερμικές μηχανές

Θερμικές μηχανές ή θερμοκινητήρες ονομάζονται οι μηχανές οι οποίες μετατρέπουν τη θερμότητα που παράγεται από τη χημική ενέργεια της καύσης σε μηχανικό έργο.

Ανάλογα με τον τρόπο πραγματοποίησης της καύσης χωρίζονται σε δύο κύριες κατηγορίες. Αυτές είναι :

- Μηχανές εσωτερικής καύσης (ΜΕΚ), που ως μέσο για την παραγωγή έργου (εργαζόμενο μέσο) χρησιμοποιούν την καύση του καυσίμου στο εσωτερικό του κινητήρα και το έργο παράγεται από τα προκύπτοντα καυσαέρια, τα οποία βρίσκονται σε υψηλή θερμοκρασία και πίεση, όπως συμβαίνει στους κινητήρες βενζίνης και πετρελαίου των αυτοκινήτων και στον αεριοστρόβιλο του αεροπλάνου.
- Μηχανές εξωτερικής καύσης, όπου η καύση δεν λαμβάνει χώρα στο χώρο παραγωγής του έργου, αλλά έξω από αυτόν και στις οποίες το μέσο παραγωγής του έργου δεν είναι το καυσαέριο, αλλά κάποιο άλλο στοιχείο όπως π.χ. το νερό. Σε αυτήν την κατηγορία ανήκουν οι ατμοστρόβιλοι, οι ατμομηχανές κ.α.

Επίσης, οι θερμικές μηχανές, ανάλογα με τον τρόπο μετατροπής της θερμικής ενέργειας σε μηχανικό έργο, διακρίνονται σε εμβολοφόρους ή παλίνδρομες (η κατηγοριοποίηση αυτή ισχύει τόσο για της μηχανές εσωτερικής όσο και για αυτές

εξωτερικής καύσης) και σε περιστροφικές ή στροβίλους (οι μηχανές εσωτερικής καύσης ονομάζονται αεριοστρόβιλοι και αυτές εξωτερικής καύσεως ατμοστρόβιλοι).

Ειδικότερα, στις παλινδρομικές μηχανές εσωτερικής καύσεως η έναυση στον κύλινδρο μπορεί να πραγματοποιηθεί είτε με τη βοήθεια εξωτερικού μέσου (π.χ. σπινθήρα), είτε αυτόματα, λόγω μεγάλης συμπίεσης του καυσίμου μαζί με τον απαιτούμενο αέρα για την καύση. Στην πρώτη περίπτωση υπάγονται οι «κινητήρες Otto», που διακρίνονται σε αεριομηχανές και βενζινομηχανές, ενώ στη δεύτερη περίπτωση οι μηχανές Diesel, ή πετρελαιομηχανές.

1.1.2 Υδραυλικές μηχανές

Οι υδραυλικές μηχανές μετατρέπουν την κινητική ενέργεια ενός υγρού σε κίνηση, κυρίως του νερού, σε μηχανική ενέργεια και αντίστροφα.

Στην πρώτη περίπτωση έχουμε τις κινητήριες υδραυλικές μηχανές (υδραυλικές τουρμπίνες, υδραυλικούς τροχούς, κινητήρες με στήλη νερού) και στη δεύτερη περίπτωση έχουμε τις ενεργειακές υδραυλικές μηχανές (αντλίες με πιστόνι, αντλίες περιστροφής κ.α.).

Ανάμεσα στις υδραυλικές μηχανές μετάδοσης συγκαταλέγονται οι πρέσες, οι στριφτές και οι υδραυλικοί γρύλοι, μεταξύ των μετασχηματιστών οι εγχυτήρες και οι υδραυλικοί δριοί.

1.1.3 Ηλεκτρικές μηχανές

Οι ηλεκτρικές μηχανές μετατρέπουν τη μηχανική ενέργεια σε ηλεκτρική (γεννήτριες) και αντίστροφα (κινητήρες) ή μετατρέπουν ηλεκτρική ενέργεια με συγκεκριμένα χαρακτηριστικά σε ηλεκτρική ενέργεια διαφορετικών χαρακτηριστικών.

Μία μηχανή όταν μετατρέπει την κάθε μορφή ενέργειας που της προσφέρεται σε κινητική ενέργεια ονομάζεται κινητήρας. Μία μηχανή όταν μετατρέπει τη μηχανική ενέργεια που της προσφέρεται σε ηλεκτρική τότε ονομάζεται γεννήτρια.

Ανάλογα με το είδος ενέργειας που μετατρέπουν σε κινητική, οι κινητήρες διακρίνονται σε :

- Ηλεκτροκινητήρες ή ηλεκτρικούς κινητήρες. Αυτή η κατηγορία αποτελείται από τους πιο εκσυγχρονισμένους κινητήρες και συγκεκριμένα από κινητήρες που παρουσιάζουν τη μεγαλύτερη ικανότητα απόδοσης ενέργειας συγκριτικά με την ενέργεια που τους προσφέρεται. Είναι η οικογένεια των κινητήρων που χρησιμοποιούμε για βάλλουμε σε κίνηση τον ηλεκτρισμό. Διακρίνονται σε κινητήρες συνεχούς και εναλλασσόμενου ρεύματος.
- Κινητήρες εσωτερικής καύσης. Σε αυτήν την κατηγορία κινητήρων ανήκουν αυτοί που χρησιμοποιούν σαν κινητήρια δύναμη τη δύναμη που αναπτύσσεται από την ανάφλεξή του και έκρηξη του πετρελαίου ή των παραγώγων του (βενζίνη κλπ.). Είναι οι περισσότερο διαδομένοι κινητήρες της σύγχρονης εποχής, γιατί συνδυάζουν και ικανοποιητική απόδοση και σχετική ανεξαρτησία απέναντι στην πηγή διοχέτευσης της ενέργειας (η μεγάλη

κατανάλωση ηλεκτρικής ενέργειας προϋποθέτει την ύπαρξη, κοντά στο σημείο κατανάλωσης, του εργοστάσιου παραγωγής της). Διακρίνονται σε κινητήρες ντίζελ, που χρησιμοποιούν σαν καύσιμο το πετρέλαιο και διακρίνονται για την αντοχή και τη δύναμή τους και σε κινητήρες που χρησιμοποιούν σαν καύσιμο τη βενζίνη. Οι δεύτεροι χρειάζονται την ύπαρξη του ηλεκτρικού αναφλεκτήρα (μπουζί), για να δουλέψουν.

- Αεριομηχανές ή γκαζομηχανές, που χρησιμοποιούν σαν κινητήρια δύναμη τη δύναμη που παράγεται από την ανάφλεξη μείγματος ορισμένων αερίων και αέρα. Σήμερα κατασκευάζονται τέτοιου είδους κινητήρες με πολύ μεγάλη συνολική ισχύ.
- Υδραυλικοί κινητήρες. Αυτοί χρησιμοποιούν τη δύναμη που ασκεί το νερό όταν αυτό πέφτει πάνω σε μια ορισμένη επιφάνεια από ύψος που ονομάζεται "ύψος υδατόπτωσης". Φαίνεται ότι είναι μία από τις πρώτες μηχανές παραγωγής κινητικής ενέργειας που ανακάλυψε ο άνθρωπος. Χρησιμοποιούνται ευρύτατα σήμερα στην Ελλάδα για παραγωγή ηλεκτρικής ενέργειας.
- Ατμομηχανές. Εδώ κινητήρια δύναμη παρέχει ο ατμός που παράγεται με τη θέρμανση νερού ή άλλου υγρού. Παλαιότερα χρησιμοποιήθηκαν για κίνηση, κυρίως τρένων (ατμάμαξες, σήμερα χρησιμοποιούνται κυρίως για την παραγωγή ηλεκτρικής ενέργειας υπό μορφή ατμοστροβίλων.
- Αιολικοί κινητήρες ή ανεμοκινητήρες. Κι αυτοί είναι απ' τους κινητήρες που ανακάλυψε ο άνθρωπος απ' τα πρώτα χρόνια της ύπαρξής του. Είναι οι κινητήρες που χρησιμοποιούν τη δύναμη ανέμου που φυσά.
- Κινητήρες πυρηνικής ενέργειας. Δεδομένου ότι η χρησιμοποίηση αυτών των κινητήρων βρίσκεται στη βρεφική της ηλικία και ότι οι δυνατότητές τους αναμένεται να είναι τεράστιες, πιστεύεται ότι η χρησιμοποίησή τους για παραγωγή ενέργειας θ' αυξηθεί σημαντικά.

1.2 Ηλεκτρικές μηχανές

Ηλεκτρική μηχανή είναι η συσκευή που μετατρέπει τη μηχανική ενέργεια σε ηλεκτρική και αντίστροφα. Έτσι, όταν η συσκευή μετατρέπει μηχανική ενέργεια σε ηλεκτρική ονομάζεται **γεννήτρια**, ενώ όταν μετατρέπει την ηλεκτρική σε μηχανική ονομάζεται **κινητήρας.**

Οι γεννήτριες (ή αλλιώς ηλεκτρογεννήτριες) βρίσκουν διάφορες εφαρμογές με πιο βασικές τις ακόλουθες :

- Στους σταθμούς παραγωγής ηλεκτρικής ενέργειας
- Σαν εφεδρικές πηγές ηλεκτρικής ενέργειας σε νοσοκομεία, εργοστάσια, αεροδρόμια κ.α. Αυτού του είδους οι γεννήτριες είναι ευρύτερα γνωστές και σαν ηλεκτροπαραγωγό ζεύγος, καθώς αποτελούνται από τη γεννήτρια και την κινητήρια μηχανή της.
- Στην παραγωγή ηλεκτρικής ενέργειας σε πλοία, τρένα, αυτοκίνητα αεροπλάνα και απομονωμένα σπίτια ή αγροκτήματα.

Οι ηλεκτροκινητήρες βρίσκουν εφαρμογή :

- Στη βιομηχανία και σε διάφορες επαγγελματικές εφαρμογές για την κίνηση εργαλειομηχανών, αντλιών, ανεμιστήρων, γερανών, ανελκυστήρων κ.α.
- Στην κίνηση οχημάτων (ηλ. έλξη) όπως ηλεκτρικών σιδηροδρόμων, τρόλεϊ, τραμ, εκκίνηση κινητήρων αυτοκινήτων (μίζα) κ.α.
- Σε διάφορες ηλεκτρικές συσκευές οικιακής ή επαγγελματικής χρήσης όπως ψυγεία, πλυντήρια, σκούπες, μίξερ, ανεμιστήρες, ραδιομαγνητόφωνα, δράπανα κ.α.

Τα κυριότερα είδη ηλεκτρικών μηχανών είναι

- Οι ηλεκτρογεννήτριες ή απλά γεννήτριες που χρησιμοποιούνται για την παραγωγή ηλεκτρικής ενέργειας. Ανάλογα με το είδος του ηλεκτρικού ρεύματος διακρίνονται σε γεννήτριες Συνεχούς Ρεύματος (Σ.Ρ) και σε γεννήτριες Εναλλασσόμενου Ρεύματος (Ε.Ρ).
- Οι ηλεκτροκινητήρες ή απλά κινητήρες που χρησιμοποιούνται για την κίνηση διάφορων μηχανημάτων και διακρίνονται σε κινητήρες Συνεχούς Ρεύματος, όταν είναι κατασκευασμένοι για να εργάζονται με συνεχές ρεύμα, και σε κινητήρες Εναλλασσόμενου Ρεύματος, όταν είναι κατασκευασμένοι για να εργάζονται με εναλλασσόμενο ρεύμα.

Οι ηλεκτρικές μηχανές (γεννήτριες και κινητήρες) είναι αναστρέψιμες. Δηλαδή, μπορούν να λειτουργήσουν γενικά και αντίστροφα. Οι γεννήτριες σαν κινητήρες, εφόσον τροφοδοτηθούν από το δίκτυο, και οι κινητήρες σα γεννήτριες, εφόσον περιστραφεί ο δρομέας τους. Οι μηχανές Ε.Ρ. διακρίνονται σε σύγχρονες και ασύγχρονες. Σύγχρονες ονομάζονται οι ηλεκτρικές μηχανές στις οποίες υπάρχει σταθερή σχέση μεταξύ της συχνότητας του ρεύματος και της ταχύτητας περιστροφής τους, δηλαδή αυτές που στρέφονται με σταθερή ταχύτητα, ενώ ασύγχρονες ονομάζονται αυτές που η ταχύτητά τους εξαρτάται από το φορτίο.

1.3 Ασύγχρονη μηχανή

1.3.1 Κατασκευαστικά στοιχεία

Οι ασύγχρονές μηχανές είναι γνωστές και ως μηχανές επαγωγής γιατί η μεταβίβαση της ενέργειας από το στάτη στον μη τροφοδοτούμενο δρομέα γίνεται μέσω επαγωγικών φαινομένων.

Μια ηλεκτρική μηχανή επαγωγής αποτελείται ουσιαστικά από ένα σταθερό μέρος, το στάτη, ο οποίος φέρει στις αύλακές του ένα μονοφασικό, διφασικό ή τριφασικό τύλιγμα, και ένα στρεφόμενο μέρος το δρομέα.. Όπως έχει ήδη αναφερθεί παραπάνω, οι μηχανές επαγωγής μπορούν να λειτουργήσουν τόσο σαν κινητήρες όσο και σαν γεννήτριες. Σαν γεννήτριες όμως σπανίως χρησιμοποιούνται για λόγους οικονομικούς.

Όταν το τύλιγμα του στάτη του κινητήρα τροφοδοτείται από συμμετρική πηγή θα δημιουργηθεί στο διάκενο μαγνητικό πεδίο που στρέφεται με τη σύγχρονη ταχύτητα. Στο διάκενο στρέφονται επίσης αρμονικά πεδία. Το στρεφόμενο πεδίο του διακένου επάγει τάσεις στα τυλίγματα του δρομέα το οποίο αποτελείται από κλειστά κυκλώματα. Οι τάσεις αυτές έχουν ως αποτέλεσμα την ροή ρευμάτων στα τυλίγματα του δρομέα και την δημιουργία κύματος ΜΕΔ το οποίο στρέφεται ως προς τον στάτη

με την σύγχρονη ταχύτητα. Η αλληλεπίδραση των δύο πεδίων στάτη και δρομέα προκαλεί την ροπή της μηχανής επαγωγής.

Η πολυφασική μηχανή επαγωγής, όταν ο δρομέας μένει ακίνητος, μπορεί να θεωρηθεί σαν μετασχηματιστής στον οποίο το παλλόμενο πεδίο έχει αντικατασταθεί από στρεφόμενο πεδίο. Η μόνη διαφορά είναι η ύπαρξη του διακένου στη μηχανή επαγωγής η οποία γι' αυτό απαιτεί πολύ μεγαλύτερο ρεύμα διεγέρσεως που φτάνει το 40% του ονομαστικού αυτής.

Για τον περιορισμό της συνισταμένης ΜΕΔ του διακένου που είναι αναγκαία για το πεδίο του διακένου λαμβάνονται κατασκευαστικά μέτρα ως εξής :

- Οι αύλακες του στάτη διαμορφώνονται μισόκλειστες ενώ του δρομέα μισόκλειστες ή τελείως κλειστές.
- Το διάκενο μεταξύ στάτη και δρομέα γίνεται μικρό.

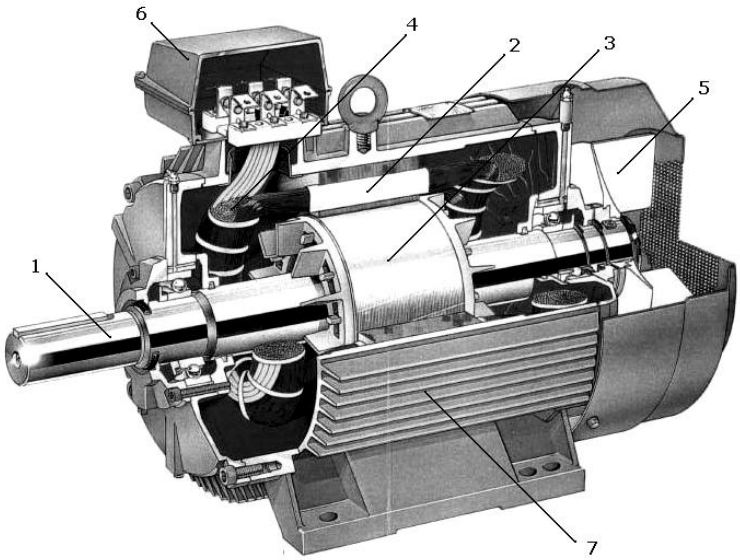
Ποιο σπάνια χρησιμοποιούνται οι ανοιχτές αύλακες. Από απόψεως κατασκευής του τυλίγματος οι ανοιχτές αύλακες είναι πιο άνετες. Εν τούτοις χρησιμοποιούνται κατά το πλείστον μισόκλειστες αύλακες διότι ανοικτές παρουσιάζονται πολλά μειονεκτήματα. Επειδή το διάκενο στις μηχανές επαγωγής είναι μικρό, οι ανοικτές αύλακες θα απαιτούσαν ένα υψηλό ρεύμα μαγνητίσεως. Επίσης προκαλούνται ισχυρές διακυμάνσεις του πεδίου του στάτη (αρμονικές χώρου) οι οποίες έχουν ως αποτέλεσμα μεγάλες απώλειες σιδήρου στον δρομέα, ιδιαίτερα με τύλιγμα κλωβού. Τέλος, παράγονται ανεπιθύμητες αρμονικές ροπές. Για τους λόγους αυτούς, ανοικτές αύλακες στον στάτη χρησιμοποιούνται μόνο σε μηχανές επαγωγής υψηλής τάσεως.
Ο πυρήνας του στάτη συντίθεται από ελάσματα χάλυβα πάχους 0,3 έως 0,6mm που φέρουν μόνωση από επικάλυψη με βερνίκι ή με ένα οξείδιο. Τα ελάσματα συσφίγγονται με φλάντζες. Με όμοια ελάσματα κατασκευάζονται και το σώμα του δρομέα.. λόγω της χαμηλής συχνότητας στον δρομέα μπορεί να χρησιμοποιηθούν παχύτερα ελάσματα χωρίς υπερβολική αύξηση των απωλειών σιδήρου.
Τα τυλίγματα του στάτη εισάγονται στις αύλακες είτε ως τυλιγμένα πηνία είτε τυλίγονται μέσα στις αύλακες είτε ως προκατασκευασμένα πηνία. Τα τυλιγμένα πηνία χρησιμοποιούνται για μηχανές μικρής ή πολύ μικρής ισχύος και για στρογγυλούς αγωγούς διαμέτρου μέχρι 2,5mm περίπου. Τα πηνία τυλίγονται έξω από τη μηχανή σε καλούπια και μετά εισάγονται στις μισόκλειστες αύλακες από τη σχισμή τους. Στη δεύτερη περίπτωση τα πηνία τυλίγονται μέσα στις μισόκλειστες αύλακες πάλι από τη σχισμή αφού προηγουμένως τοποθετηθεί σε αυτές η μόνωση αύλακος. Τέλος, τα προκατασκευασμένα και μονωμένα πηνία χρησιμοποιούνται για μηχανές υψηλής τάσεως οι οποίες έχουν ανοικτές αύλακες. Εφ' όσον οι αύλακες είναι μισόκλειστες τότε τα πηνία προκατασκευάζονται μονωμένα αλλά χωρίς τη μία κεφαλή (ακραία σύνδεση). Αφού το πηνίο τοποθετηθεί στη θέση του η κεφαλή που λείπει συγκολλάται στη θέση της.

Συνήθως κατασκευάζονται δύο τύποι δρομέων :

- Με τύλιγμα κλωβού ή βραχυκυκλωμένος δρομέας.
- Με τυλιγμένο δρομέα ή με δακτυλίους
- Δρομέας χωρίς αύλακες, ο οποίος χρησιμοποιείται για κινητήρες μεγάλης ταχύτητας σε συστήματα κινήσεως, ή για κινητήρες μικρής ισχύος στα συστήματα αυτομάτου ελέγχου.

Στο σχήμα 1.1 φαίνεται ένας τριφασικός (3Φ) κινητήρας επαγωγής βραχυκυκλωμένου κλωβού όπου διακρίνονται τα εξής :

1. ο άξονας
2. ο στάτης
3. ο δρομέας
4. το τύλιγμα του στάτη
5. η φτερωτή ανεμισμού
6. το κιβώτιο σύνδεσης τυλιγμάτων/ακροδεκτών
7. το πλαίσιο (κατασκευή ABB, 90kW, 1484 ΣΑΛ, 630kg)



Σχήμα 1. 1

1.3.2 Τυλίγματα δρομέα

Οι τρεις τύποι τυλιγμάτων δρομέα που αναφέρθηκαν στην παράγραφο 1.3.1 περιγράφονται παρακάτω κάπως εκτενέστερα.

Α) Τύλιγμα κλωβού

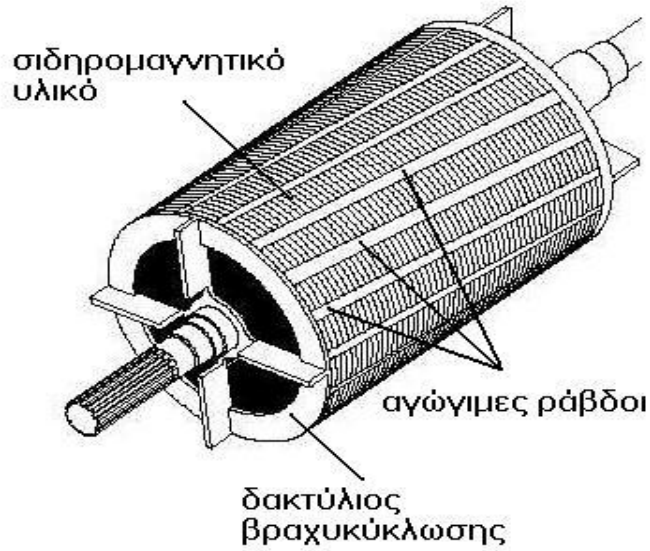
Αποτελεί τα απλούστερο τύλιγμα για τον δρομέα ενός κινητήρα επαγωγής. Το τύλιγμα αποτελείται από αριθμό ράβδων από αλουμίνιο ή χαλκό κατά τη διεύθυνση της γεννήτριας ή με κάποια λοξότητα, οι οποίες βραχυκυκλώνονται στα δύο άκρα με δακτυλίους από το ίδιο υλικό. Ο χαλκός από αλουμίνιο κατασκευάζεται με χύτευση. Ο κλωβός από χαλκό συντίθεται από τις ράβδους, που στην περίπτωση αυτή μπορεί να είναι μονωμένοι και τους δακτυλίους.

Το τύλιγμα κλωβού με S_2 ράβδους μπορεί να θεωρηθεί σαν ένα τύλιγμα με S_2 φάσεις με μία ράβδο ανά φάση. Έτσι, οι S_2 φάσεις συνδέονται κατ' αστέρα με τον έναν δακτύλιο ενώ με τον άλλο δακτύλιο βραχυκυκλώνονται. Μία άλλη θεώρηση του τυλίγματος κλωβού είναι ότι έχει $\frac{S_2}{p}$ φάσεις, όπου p ο αριθμός των πόλων του συνεργαζόμενου στάτη. Κάθε φάση αποτελείται από συγκεντρωμένα πηνία ενός

τυλίγματος και με βήμα β=180° ηλεκτρικές μοίρες, ενώ κάθε πηνίο είναι βραχυκυκλωμένο. Όλα τα πηνία του τυλίγματος κλωβού βραχυκυκλώνονται στα δύο ακραία τμήματά τους με τους δακτυλίους. Ένα τύλιγμα μπορεί να χρησιμοποιηθεί σε κινητήρα με οποιονδήποτε αριθμό πόλων.

Για την βελτίωση των συνθηκών εκκινήσεως το τύλιγμα κλωβού κατασκευάζεται με υψηλές και στενές ράβδους ή με διπλό κλωβό.

Στο σχήμα 1.2 βλέπουμε σχηματική αναπαράσταση ενός δρομέα βραχυκυκλωμένου κλωβού.



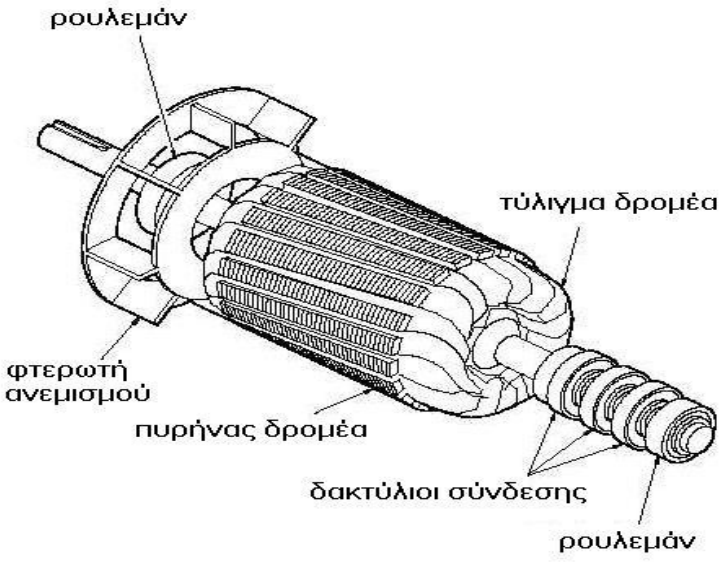
Σχήμα 1. 2

Β) Τύλιγμα δρομέα με δακτυλίους

Τυλιγμένοι δρομείς για κινητήρες μέχρι 10kW εφοδιάζονται με συμμετρικά τριφασικά τυλίγματα όπως αυτά που χρησιμοποιούνται στον στάτη. Για δρομείς κινητήρων μέσης ισχύος χρησιμοποιούνται σχεδόν πάντοτε τυλίγματα με δύο στρώματα με δύο πλευρές πηνίου σε κάθε αύλακα και με ακέραιο ή κλασματικό αριθμό αυλάκων ανά πολικό βήμα και φάση q. Η σύνδεση των τριών φάσεων του δρομέα γίνεται συνήθως κατ' αστέρα.

Ο δρομέας με δακτυλίους συγκρινόμενος με τον δρομέα κλωβού είναι ποιο ακριβός διότι κατ' αρχήν απαιτεί δακτυλίους, ψήκτρες και αντιστάσεις εκκινήσεως. Πλην αυτών, συχνά προβλέπονται διατάξεις οι οποίες επιτρέπουν μία ανύψωση ψηκτρών και βραχυκύκλωση των δακτυλίων μετά την επίτευξη της εκκινήσεως. Επιπλέον, το τύλιγμα κλωβού έχει λιγότερο αγώγιμο υλικό από το τύλιγμα δρομέα με δακτυλίους για την αυτή ισχύ. Εν τούτοις, κινητήρες με τυλιγμένο δρομέα εεξακολουθούν να χρησιμοποιούνται λόγω των ιδιοτήτων τους εκκινήσεως.

Στο σχήμα 1.3 βλέπουμε σχηματική αναπαράσταση ενός τυλιγμένου δρομέα.



Σχήμα 1. 3

Γ) Δρομείς χωρίς αύλακες

Οι δρομείς αυτοί κατασκευάζονται συνήθως από ολόσωμο σίδηρο όπου επάγονται δινορρεύματα ώστε να προκύψει χρήσιμη ροπή. Εν τούτοις ο βαθμός αποδόσεως, ο συντελεστής ισχύος και η ικανότητα υπερφορτίσεως ενός τέτοιου κινητήρα δεν είναι ευνοϊκοί. Οι κινητήρες με ολόσωμους δρομείς χρησιμοποιούνται για συστήματα κινήσεως με υψηλό αριθμό στροφών ανά λεπτό επειδή έχουν μεγάλη μηχανική αντοχή. Αναπτύσσουν σχετικώς μεγάλη ροπή εκκινήσεως και μικρό ρεύμα εκκινήσεως. Για την βελτίωση των ιδιοτήτων τους ενίοτε κατασκευάζεται ο δρομέας από στρώματα κατάλληλων υλικών. Η απλούστερη κατασκευή αυτού του τύπου είναι η τοποθέτηση φύλλου χαλκού επί της κυλινδρικής επιφάνειας του ολόσωμου σιδήρου.

Τέλος κινητήρες πολύ μικρής ισχύος για τα συστήματα αυτομάτου ελέγχου έχουν τον σίδηρο στο δρομέα ακίνητο ενώ το στρεφόμενο μέρος του δρομέα κατασκευάζεται από αλουμίνιο, έχει δε τη μορφή κυπέλλου με ένα έδρανο. Το κύπελλο περιβάλλει το σιδερένιο ακίνητο μέρος του δρομέα ενώ ο στάτης από σίδηρο περιβάλλει το κύπελλο.

1.4 Ισοδύναμο κύκλωμα επαγωγικού κινητήρα

Η τροφοδότηση του στάτη με εναλλασσόμενη τάση δημιουργεί κύμα ΜΕΔ που στρέφεται με τη σύγχρονη ταχύτητα ns (ΣΑΛ) :

$$n_s = \frac{120\,f}{p} \qquad (1.1)$$

Όπου f η ηλεκτρική συχνότητα τροφοδοσίας και p το πλήθος των πόλων της μηχανής. Αντίστοιχα σε rad/sec η σύγχρονη ταχύτητα είναι :

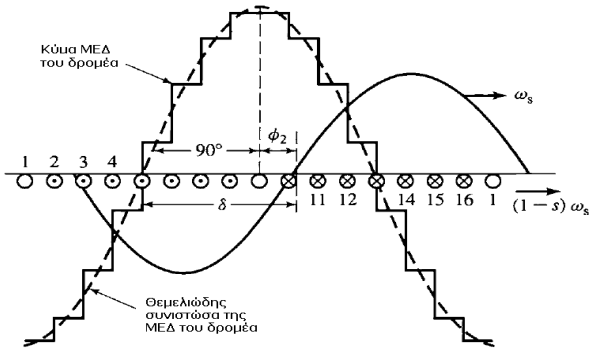
$$\omega_s = \frac{2\pi f}{p/2} \qquad (1.2)$$

Έστω ότι η ταχύτητα περιστροφής του δρομέα είναι n. Τότε ορίζεται η ολίσθηση s από τη σχέση :

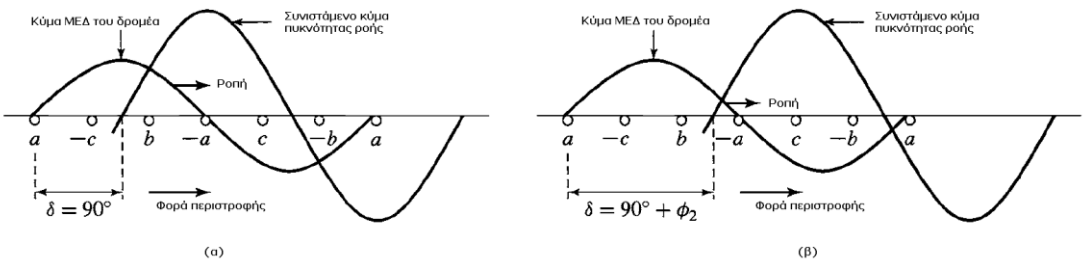
$$s = \frac{n_s - n}{n_s} \qquad (1.3)$$

Το κύμα πυκνότητας ροής του στάτη κινείται προς το δρομέα με την ταχύτητα ολισθήσεως sn_s. Επάγει, συνεπώς, στο τύλιγμα του δρομέα εναλλασσόμενες τάσεις συχνότητας sf, όπου f είναι η συχνότητα των ρευμάτων του στάτη. Επειδή ο δρομέας είναι βραχυκυκλωμένος, οι επαγόμενες στο τύλιγμα του τάσεις θα προκαλέσουν εναλλασσόμενα τριφασικά ρεύματα και συνακόλουθα ένα μαγνητικό πεδίο που θα περιστρέφεται με ταχύτητα sn_s ως προς το δρομέα. Ως προς το στάτη συνεπώς, τα μαγνητικό πεδίο του δρομέα περιστρέφεται με τη σύγχρονη ταχύτητα n_s. Το κύμα της ΜΕΔ του δρομέα σχηματίζει επομένως μία σταθερή γωνία με το συνιστάμενο κύμα ροής, η οποία υπερβαίνει τις 90 ηλεκτρικές μοίρες κατά τη γωνία συντελεστή ισχύος φ_2 του τυλίγματος του δρομέα.

Τα κύματα μαγνητικής ροής και ΜΕΔ στο διάκενο φαίνονται στα σχήματα 1.4και 1.5 για την περίπτωση τυλίγματος κλωβού και τυλιγμένου δρομέα αντίστοιχα.



Σχήμα 1. 4
Ανάπτυξη κύματος ΜΕΔ σε δρομέα με τύλιγμα κλωβού



Σχήμα 1. 5
Συνιστάμενο κύμα ΜΕΔ και κύμα ΜΕΔ δρομέα : α) cosφ=1, β) cosφ≠1

Ένα ισοδύναμο κύκλωμα μόνιμης κατάστασης μπορεί να κατασκευαστεί θεωρώντας μηχανές με συμμετρικά τυλίγματα που διεγείρονται από συμμετρικές πηγές τάσης. Επίσης, η μηχανή θεωρείται ότι είναι συνδεσμολογημένη σε αστέρα

ώστε να είναι τα ρεύματα γραμμής και οι τάσεις φασικές. Με τον τρόπο αυτόν παράγεται ένα ισοδύναμο κύκλωμα της μίας φάσης. Τα ρεύματα και οι τάσεις των άλλων φάσεων μπορούν να βρεθούν με κατάλληλη μετακίνηση της φάσης ($\pm 120^o$ σε περίπτωση 3Φ μηχανής).

Πρώτα θεωρείται ο στάτης. Το κύμα ροής του διακένου που στρέφεται με τη σύγχρονη ταχύτητα παράγει συμμετρικές πολυφασικές αντι-ηλεκτρεργερτικές δυνάμεις στις φάσεις του στάτη. Η τάση ακροδεκτών του στάτη διαφέρει από την αντι-ΗΕΔ κατά την πτώση τάσης στην σύνθετη αντίσταση σκεδάσεως του στάτη $Z_1=R_1+jX_1$. Η διανυσματική σχέση ανά φάση είναι :

$$V_1 = E_2 + I_1(R_1 + jX_1) \qquad (1.4)$$

Όπου :

V_1 η τάση ακροδεκτών του στάτη

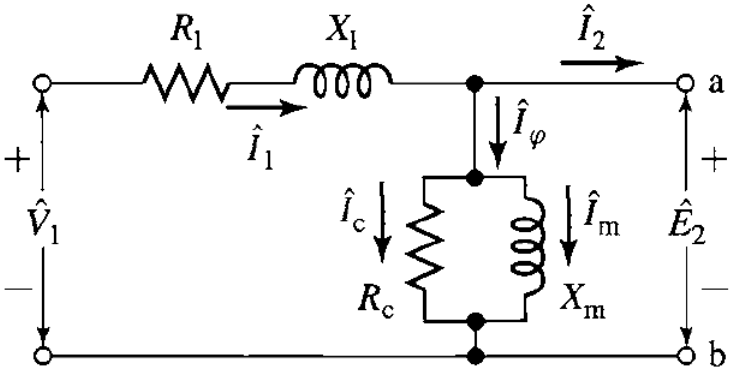
E_2 η αντί-ΗΕΔ που αναπτύσσεται από τη συνιστάμενη ροή διακένου

I_1 το ρεύμα του στάτη

R_1 η πραγματική αντίσταση του στάτη

X_1 η αντίδραση σκεδάσεως του στάση

Οι θετικές φορές των ρευμάτων και των τάσεων σημειώνονται στο ισοδύναμο κύκλωμα του σχήματος 1.6



Σχήμα 1. 6
Ισοδύναμο κύκλωμα στάτη επαγωγικού κινητήρα

Η συνιστάμενη ροή διακένου δημιουργείται από τις συνδυασμένες ΗΕΔ των ρευμάτων του στάτη και του δρομέα. Ακριβώς όπως στο ανάλογο του μετασχηματιστή, το ρεύμα μπορεί να αναλυθεί σε δύο συνιστώσες, μία συνιστώσα φορτίου και μία διεγέρσεως. Η συνιστώσα φορτίου I_2 παράγει ΜΕΔ η οποία αντιτίθεται στην ΗΕΔ του ρεύματος του δρομέα. Η συνιστώσα διεγέρσεως I_ϕ είναι το πρόσθετο ρεύμα του στάτη, το οποίο απαιτείται για τη δημιουργία της συνιστάμενης ροής διακένου και είναι συνάρτηση της ΗΕΔ E_1. Το ρεύμα διεγέρσεως μπορεί να αναλυθεί σε μία συνιστώσα απωλειών πυρήνα I_π σε φάση με την E_1 και μία συνιστώσα μαγνήτισης I_m που έπεται της E_1 κατά 90°. Στο ισοδύναμο κύκλωμα το ρεύμα διεγέρσεως μπορεί να περιληφθεί μέσω ενός παράλληλου κλάδου, που

σχηματίζεται από την αγωγιμότητα g_π των απωλειών πυρήνα και από μία μαγνητίζουσα αγωγιμότητα b_m παράλληλα με την τάση E_1 (σχήμα 1.6). τόσο η g_π όσο και η b_m συνήθως καθορίζονται στην ονομαστική συχνότητα του στάτη και για μία τιμή E_1 πλησίον της αναμενόμενης τιμής λειτουργίας. Γίνεται η παραδοχή ότι και οι δύο παραμένουν σταθερές για τις μικρές αποκλίσεις από την τιμή αυτή η σχετίζεται με την κανονική λειτουργία του κινητήρα.

Το ισοδύναμο κύκλωμα που περιγράφει τα φαινόμενα στο στάτη, ως τώρα, είναι πανομοιότυπο με του πρωτεύοντος του μετασχηματιστή. Για την ολοκλήρωση του ισοδύναμου πρέπει να ενσωματωθεί και η επίδραση του δρομέα. Τα βασικά φαινόμενα είναι όμοια είτε ο δρομέας είναι βραχυκυκλωμένου κλωβού, είτε δακτυλιοφόρος. Από την πλευρά του ισοδύναμου του στάτη του σχήματος 1.6, ο δρομέας μπορεί να παρασταθεί σαν μία ισοδύναμη σύνθετη αντίσταση

$$Z_2 = \frac{E_2}{I_2} \qquad (1.5)$$

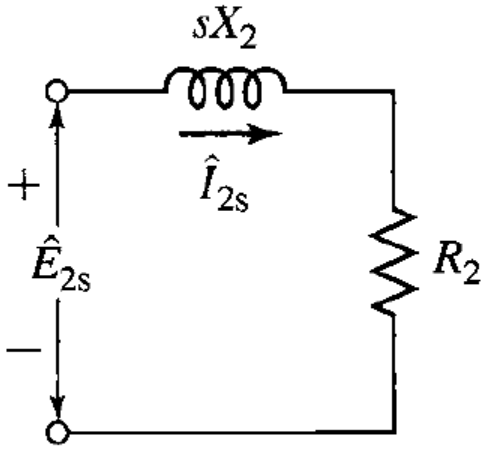
Αυτή αντιστοιχεί στη σύνθετη αντίσταση σκέδασης ενός σταθερού δευτερεύοντος. Για την ολοκλήρωση του ισοδύναμου κυκλώματος, πρέπει να προσδιορισθεί η Z_2 ανάγοντας τις ποσότητες στάτη και δρομέα (τάσης και ρεύματα) σε ποσότητες του δρομέα ανηγμένες στο στάτη.

Στην ανάλυση μετασχηματιστών, από την πλευρά του πρωτεύοντος, το δευτερεύον τύλιγμα του μετασχηματιστή μπορεί να αντικατασταθεί από ένα ισοδύναμο δευτερεύον τύλιγμα που έχει τον ίδιο αριθμό ελιγμάτων με το πρωτεύον, πολλαπλασιάζοντας με το τετράγωνο του λόγου ελιγμάτων πρωτεύοντος προς δευτερεύον. Το συνεπαγόμενο ισοδύναμο κύκλωμα προκύπτει πλήρως ανηγμένο στην πλευρά του πρωτεύοντος του μετασχηματιστή. Ομοίως, στην περίπτωση μιας πολυφασικής μηχανής, αν ο δρομέας αντικατασταθεί με έναν ισοδύναμο δρομέα με πολυφασικό τύλιγμα με τον ίδιο αριθμό φάσεων και ελιγμάτων με το στάτη αλλά παράγοντας την ίδια ΜΕΔ και ροή διακένου όπως ο δρομέας, η λειτουργία της μηχανής, όπως φαίνεται από τους ακροδέκτες του στάτη, θα είναι πανομοιότυπη. Αυτή η τεχνική θα εφαρμοστεί και εδώ καθώς αποδεικνύεται ιδιαίτερα χρήσιμη, ειδικά για μηχανές επαγωγής δρομέα τύπου κλωβού για τις οποίες η «ταυτότητα» των τυλιγμάτων ανά φάση του δρομέα δεν είναι με καμία έννοια προφανής. Ο δρομέας μιας μηχανής επαγωγής είναι βραχυκυκλωμένος, έτσι η σύνθετη αντίδραση που εμφανίζεται λόγω της επαγόμενης από το στάση τάσεως είναι απλά η σύνθετη αντίσταση βραχυκύκλωσης του τυλίγματος του δρομέα. Συνεπώς, η σχέση μεταξύ της σύνθετης αντίστασης σκέδασης του ισοδύναμου δρομέα στη συχνότητα ολισθήσεως (Z_{2s}) και της αντίστοιχης του πραγματικού δρομέα (Z_{rotor}) θα είναι :

$$Z_{2s} = \frac{E_{2s}}{I_{2s}} = R_2 + jsX_2 \qquad (1.6)$$

Σημειώνεται ότι η αντίδραση X2 είναι η αντίδραση σκέδασης του δρομέα στη συχνότητα του στάτη (fe). Καθώς η πραγματική συχνότητα των ρευμάτων του δρομέα είναι fr = s fe, η αντίδραση σκέδασης X2 μπορεί να αναχθεί στη συχνότητα αυτή απλά πολλαπλασιαζόμενη με την ολίσθηση s. Το ισοδύναμο κύκλωμα, όπως

φαίνεται από το πλαίσιο αναφοράς της συχνότητας ολισθήσεως των ρευμάτων του δρομέα, δίνεται στο σχήμα 1.7.



Σχήμα 1. 7 Ισοδύναμο κύκλωμα δρομέα επαγωγικού κινητήρα στη συχνότητα ολίσθησης

Στη συνέχεια παρατηρείται ότι το κύμα ΜΕΔ διακένου παράγεται από τη συνδυασμένη δράση του ρεύματος στάτη I_1 και του αντίστοιχου ρεύματος φορτίου I_2. Το ίδιο μπορεί να εκφραστεί παρομοίως σε όρους ρεύματος στάτη και αντίστοιχου ρεύματος δρομέα I_{2s}. Αυτά τα δύο ρεύματα είναι ίσα σε πλάτος αφού το I_{2s} ορίζεται ως το ρεύμα ενός ισοδυνάμου δρομέα με τον ίδιο αριθμό ελιγμάτων ανά φάση με το στάτη. Καθώς το συνιστάμενο κύμα ΜΕΔ διακένου καθορίζεται από το ανυσματικό άθροισμα του ρεύματος στάτη και δρομέα, είτε του πραγματικού είτε του ισοδυνάμου δρομέα, τα I και I_{2s} πρέπει επίσης να είναι ίσα σε φάση (στις αντίστοιχες ηλεκτρικές τους συχνότητες), γι' αυτό θεωρείται :

$$I_2 = I_{2s} \qquad (1.7)$$

Τέλος, το κύμα ροής του διακένου επάγει και την ΗΕΔ E_{2s} στη συχνότητα ολισθήσεως του δρομέα και την αντι-ΗΕΔ E_2 του στάτη. Αν δεν υπήρχε η επίδραση της ταχύτητας (λειτουργία με ακινητοποιημένο δρομέα) αυτές οι δύο τάσεις θα ήταν ίσες σε μέτρο καθώς ο ισοδύναμος δρομέας έχει τύλιγμα με τον ίδιο αριθμό ελιγμάτων ανά φάση όπως και ο στάτης. Ωστόσο, επειδή η σχετική ταχύτητα του κύματος ροής διακένου σε σχέση με το δρομέα είναι s η ταχύτητα σε σχέση με το στάτη, η σχέση μεταξύ των πλατών των επαγόμενων ΗΕΔ – αντι-ΗΕΔ είναι :

$$E_{2s} = sE_2 \qquad (1.8)$$

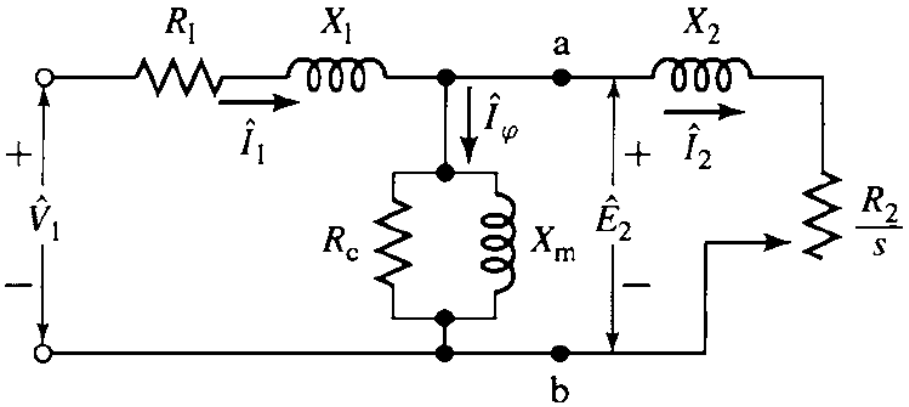
Καθώς η φάση κάθε μίας από αυτές τις τάσεις και του προκύπτοντος κύματος ροής διακένου είναι 90°, αυτές οι δύο τάσεις πρέπει επίσης να έχουν ίσους φασιθέτες στις αντίστοιχες ηλεκτρικές τους συχνότητες. Έτσι :

$$E_{2s} = E_2 \qquad (1.9)$$

Εύκολα προκύπτει ότι :

$$Z_2 = \frac{E_2}{I_2} = \frac{R_2}{s} + jX_2 \qquad (1.10)$$

Έτσι ολοκληρώνετε η κατασκευή του ανά φάση ισοδύναμου κυκλώματος της μηχανής επαγωγής. Η Z_2 είναι η σύνθετη αντίσταση του ισοδυνάμου ακίνητου δρομέα όπως φαίνεται από τους ακροδέκτες του φορτίου στο ισοδύναμο κύκλωμα του στάτη. Το τελικό αποτέλεσμα φαίνεται στο ισοδύναμο ανά φάση κύκλωμα του σχήματος

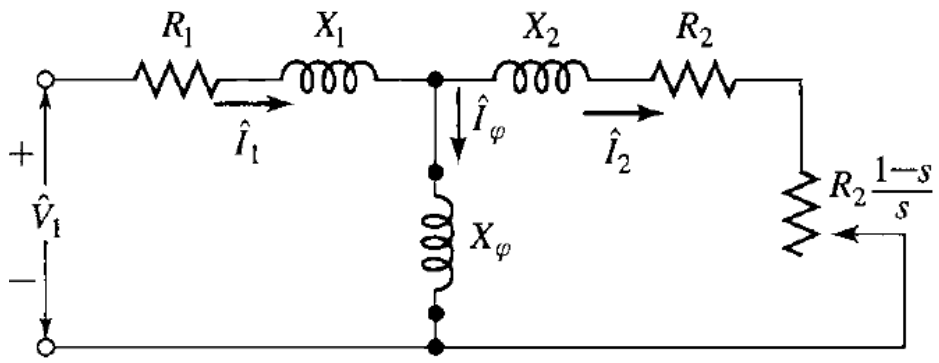


Σχήμα 1. 8 Μονοφασικό ισοδύναμο κύκλωμα επαγωγικού κινητήρα

Η συνδυασμένη δράση του φορτίου στον άξονα της μηχανής και της αντιστάσεως του δρομέα φαίνεται στην ανηγμένη αντίσταση R_2/s, που είναι συναρτήσει της ολίσθησης, άρα και του μηχανικού φορτίου. Το ρεύμα στην ανηγμένη σύνθετη αντίσταση του δρομέα ισούται με τι κομμάτι του ρεύματος του στάτη που αντιστοιχεί στο ρεύμα φορτίου. Η τάση κατά μήκος της ισούται με την τάση E_2 του κυκλώματος στάτη. Σημειώνεται ότι όταν τα ρεύματα και οι τάσεις του δρομέα ανάγονται στο στάτη, οι συχνότητες τους επίσης αλλάζουν στη συχνότητα του στάτη. Όλα τα ηλεκτρικά φαινόμενα στο δρομέα, όταν παρατηρούνται από παρατηρητή σταθερά τοποθετημένο στο στάτη της μηχανής γίνονται φαινόμενα με συχνότητα αυτή του στάτη, γιατί απλούστατα τα τυλίγματα του στάτη βλέπουν τα κύματα ΜΕΔ και ροής να ταξιδεύουν με τη σύγχρονη ταχύτητα.

Στους μετασχηματιστές ο εγκάρσιος κλάδος του ισοδυνάμου κυκλώματος, δηλαδή ο κλάδος μαγνητίσεως είναι στις περισσότερες περιπτώσεις αμελητέος. Στις μηχανές επαγωγής όμως, η μαγνητική αντίσταση που συναντά το κύμα ΜΕΔ είναι σημαντικά μεγαλύτερο απ' ότι στους μετασχηματιστές λόγω της παρεμβολής του διακένου αέρος. Έτσι απαιτείται ένα σημαντικό ρεύμα διεγέρσεως I_ϕ για να υπερνικήσει τη μαγνητική αντίσταση και να εγκαταστήσει το συνιστάμενο κύμα μαγνητικής ροής. Κατά συνέπεια ο εγκάρσιος κλάδος του ισοδυνάμου κυκλώματος δεν μπορεί να παραληφθεί εντελώς.

Μία απλοποίηση μπορεί να γίνει ωστόσο, αν αμεληθούν οι ούτως ή άλλως μικρές απώλειες πυρήνα πάνω στην αγωγιμότητα R_c. Οι απώλειες πυρήνα μπορούν σε αυτή την περίπτωση να συνυπολογιστούν μαζί με τις μηχανικές απώλειες στο συνολικό ισολογισμό της ισχύος. Με την παράλειψη της ωμικής αγωγιμότητας το ισοδύναμο κύκλωμα της μηχανής επαγωγής μπορεί να πάρει τη μορφή του σχήματος 1.9, όπου με X_φ συμβολίζεται η μαγνητίζουσα αντίδραση της μηχανής, δηλαδή της επαγωγικής αγωγιμότητας μαγνητίσεως X_m.

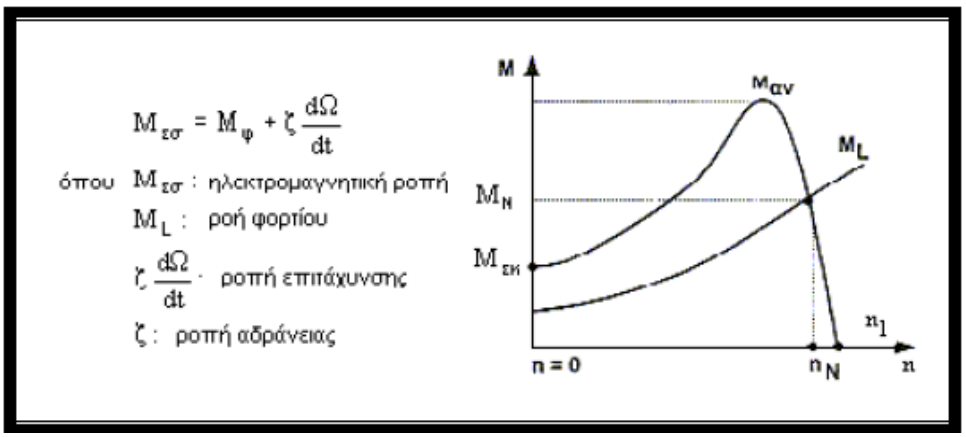


Σχήμα 1. 9 Απλοποιημένο ισοδύναμο κύκλωμα κινητήρα επαγωγής

1.5 Μέθοδοι εκκίνησης κινητήρα επαγωγής

Για να περιστραφεί ένας κινητήρας επαγωγής είναι αναγκαία η ροπή που αναπτύσσει να είναι μεγαλύτερη από τη ροπή φορτίου και τριβών μέχρι να αποκατασταθεί η ισορροπία, διότι πρέπει να υπερνικηθεί η αδράνεια. Η γενική εξίσωση που εκφράζει την ισορροπία των ροπών φαίνεται στο παρακάτω σχήμα (σχήμα) όπου διακρίνονται και οι χαρακτηριστικές $M_{\varepsilon\varsigma}$=f(n) και M_φ= f(n).

Η επιτάχυνση διαρκεί τόσο χρόνο μέχρι να γίνει $M_{\varepsilon\varsigma}$=M_φ. Το σημείο που έχει συντεταγμένες (M_N,n_N) πρέπει να είναι τέτοιο που να ανταποκρίνεται στις απαιτήσεις που τίθεται από την θερμική αντοχή της μηχανής, από το συντελεστή απόδοσης, από τη δυνατότητα υπερφόρτισης και τον κατάλληλο συντελεστή ισχύος. Η συνθήκη ευστάθειας προκύπτει από τη μελέτη του διαγράμματος $M_{\varepsilon\varsigma}$=f(n) του επαγωγικού κινητήρα σε συνδυασμό με την καμπύλη του φορτίου.



Σχήμα 1. 10 Εξίσωση ισορροπίας ροπών

Προκειμένου να υπερνικηθεί η ροπή φορτίου και η ροπή των τριβών, κατά την εκκίνηση του επαγωγικού κινητήρα, απαιτείται ένα μεγάλο ρεύμα εκκίνησης, πολύ μεγαλύτερο του ονομαστικού. Το ρεύμα αυτό ακόμα και για την μικρή διάρκεια της εκκίνησης προκαλεί μεγάλες απώλειες ισχύος και πτώση τάσεις.

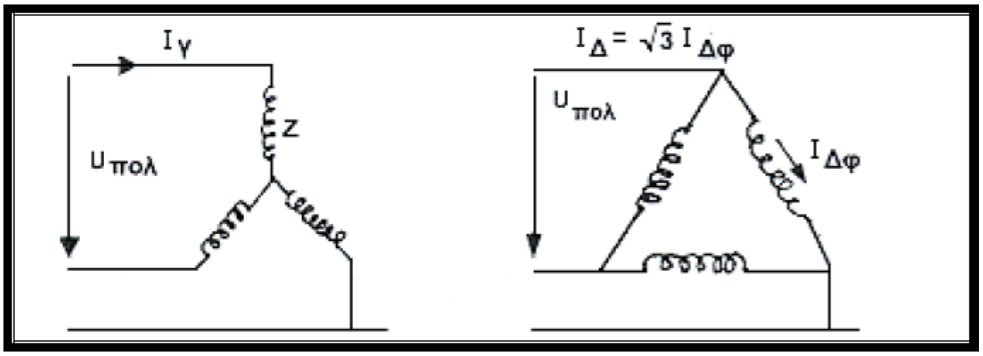
Την στιγμή της εκκίνησης είναι n=0 (s=1) και οι σχέσεις του ρεύματος και της ροπής δίνουν ότι κατά την εκκίνηση του επαγωγικού κινητήρα το ρεύμα εκκίνησης ισούται με έξι έως οκτώ φορές το ονομαστικό.

Στην εκκίνηση λοιπόν μας ενδιαφέρουν δύο μεγέθη, το ρεύμα εκκίνησης και η ροπή εκκίνησης. Άρα, προσπαθούμε να έχουμε όσο το δυνατό μικρότερο ρεύμα εκκίνησης και ροπή τέτοια που να επιτρέπει την ομαλή εκκίνηση.

Η απλούστερη μέθοδος εκκίνησης είναι η άμεση, δηλαδή η χωρίς παρεμβολή κανενός στοιχείου εκκίνησης. Υπάρχει ανάμεσα στους τεχνίτες η αντίληψη ότι επιτρέπεται από τη Δ.Ε.Η. για σύγχρονους κινητήρες μέχρι 3÷5 HP. Αυτό δεν είναι απόλυτα σωστό καθώς η Δ.Ε.Η. θέλει από 5% στο σημείο τροφοδοσίας. Έτσι σε ένα μεγάλο εργοστάσιο ισχύος π.χ. 3000 kVA μπορεί να είναι δυνατή η άμεση εκκίνηση κινητήρα έως 20-30kW με την προϋπόθεση ο κινητήρας να αντέξει στο μεγάλο ρεύμα εκκίνησης. Σε σταθμούς παραγωγής γίνεται άμεση εκκίνηση κινητήρων επαγωγής Υ.Τ. ισχύος 500kW.

1.5.1 Εκκίνηση Υ/Δ σε κινητήρες επαγωγής βραχυκυκλωμένου δρομέα

Εδώ πρέπει σε κανονική λειτουργία ο κινητήρας να κατασκευασθεί (ως προς τις περιελίξεις) έτσι ώστε να έχουμε συνδεσμολογία τριγώνου για να μπορεί να λειτουργήσει με διακόπτη Υ/Δ. η εκκίνηση αυτή γίνεται για λόγους μείωσης του ρεύματος εκκίνησης.



Σχήμα 1. 11

Στο σχήμα 1.11 για ολίσθηση s=σταθ. Θα είναι και Z=σταθ, άρα :

$$I_y = \frac{U_\pi}{\sqrt{3}Z} \text{ και } I_{\Delta\phi} = \frac{U_\pi}{Z} \Rightarrow I_\Delta = \frac{\sqrt{3}U_\pi}{Z} = \sqrt{3}I_{\Delta\phi}$$

$$\text{Άρα } \frac{I_Y}{I_\Delta} = \frac{\frac{U_\pi}{\sqrt{3}Z}}{\frac{\sqrt{3}U_\pi}{Z}} = \frac{1}{3} \qquad (1.11)$$

που ισχύει για σταθερό ¨s¨. Εδώ το εφαρμόζουμε για την εκκίνηση (n=0, s=1). Το ρεύμα της κατευθείαν εκκίνησης $I_\Delta = (5\div9)\ I_{ον}$.

Έτσι εδώ για Υ/Δ είναι κατά 1/3 λιγότερο. Για συγκεκριμένες τιμές του ρεύματος εκκίνησης (λόγω οδηγιών της Δ.Ε.Η.) χρησιμοποιούμε κινητήρα δακτυλιοφόρου δρομέα όπου το ρεύμα και η ροπή εκκίνησης συμπεριφέροντε ανάλογα.

Στην εκκίνηση η ροπή τριβών $M_{τρ} = 0$ και η σχέση της ροπής είναι :

$$M_{\varepsilon\sigma} = \frac{P_\delta}{\omega_{1m}} = \frac{3}{\omega_{1m}} I^2 \frac{R_2}{s} \qquad (1.12)$$

Στην παραπάνω σχέση (σχέση 1.12) δεχόμαστε ότι τα μεγέθη R_2 και s είναι σταθερά. Το I_2 είναι το φασικό ρεύμα. Εδώ επίσης κάνουμε την παραδοχή ότι $I_m \approx 0$ και $I_1 \approx I_2$. Έτσι η ροπή στον αστέρα για τη συγκεκριμένη ολίσθηση s=1 είναι :

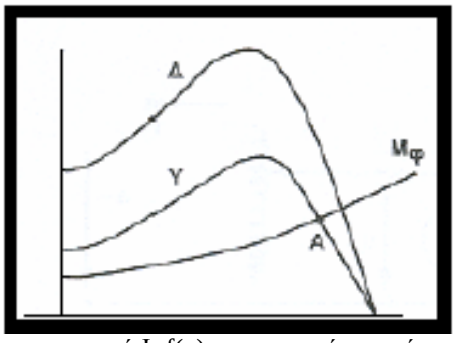
$$M_Y = \frac{3R_2}{\omega_{1m}} I_1^2 \qquad (1.13)$$

Στον αστέρα $I_{\gamma\rho}=I_\varphi$ και στο τρίγωνο $I_{\gamma\rho}=\sqrt{3}\, I_{\Delta\varphi}$. Άρα :

$$\frac{M_Y}{M_\Delta} = \frac{I_Y^2}{I_{\Delta\phi}} = \frac{\frac{U_\pi}{\sqrt{3}Z^2}}{\frac{U_\pi}{Z}} = \frac{1}{3} \qquad (1.14)$$

Δηλαδή, και $\frac{T_Y}{T_\Delta} = \frac{1}{3}$ για την εκκίνηση Υ/Δ.

Αν παρατηρήσουμε τις χαρακτηριστικές του I=f(n) του σχήματος 1.12 βλέπουμε ότι στη μετάζευξη από Υ→Δ έχουμε μια αιχμή του ρεύματος που είναι μεγάλη και συνεπώς ενοχλεί. Πρέπει να προσέξουμε να μην κάνουμε την αλλαγή Υ→Δ την χρονική στιγμή π.χ. της αναστροφής διότι η αιχμή θα αποκτήσει μεγάλες τιμές. Η εξέλιξη στο χρόνο του ρεύματος και της ροπής είναι διαφορετική όπου φαίνεται η επίδραση της αιχμής.

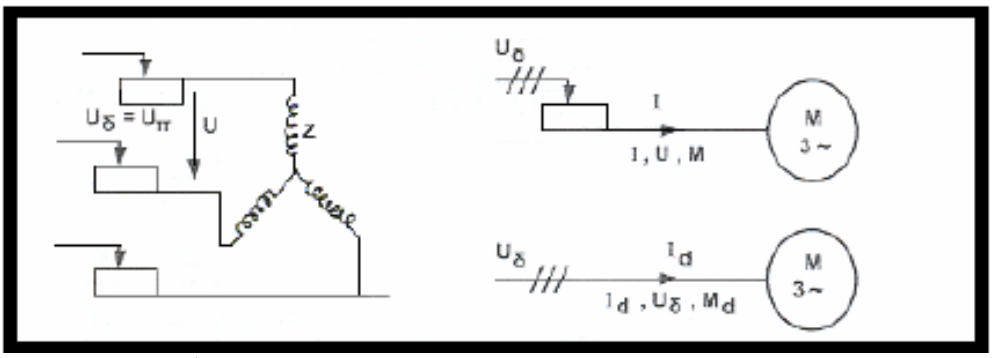


Σχήμα 1. 12 Χαρακτηριστική I=f(n) επαγωγικού κινητήρα για εκκίνηση Υ/Δ

1.5.2 Εκκίνηση με αντιστάσεις στον στάτη

Εξετάζουμε εδώ την περίπτωση I, U, M όπως και την περίπτωση των I_d, U_δ, M_d, στην κατευθείαν εκκίνηση όπως φαίνεται στο σχήμα 1.13. Εφ' όσον έχουμε τον ίδιο κινητήρα με το ίδιο Z θα ισχύει:

$$\frac{I}{I_d} = \frac{U}{U_\delta} \text{ και } \frac{M}{M_d} = \left[\frac{\mathrm{I}}{\mathrm{I}_d}\right]^2 \text{ αφού } M \approx \frac{3R_2}{s\omega_{1m}} I_1^2$$



Σχήμα 1. 13 Εκκίνηση επαγωγικού κινητήρα με αντιστάσεις στον στάτη

Άρα :

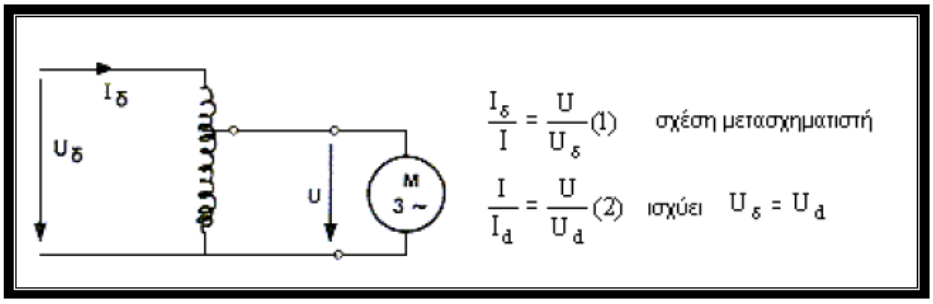
$$\frac{M}{M_d} = \left[\frac{U}{U_\delta}\right]^2 \qquad (1.15)$$

Έτσι αν αυξηθεί η τάση κατά 10% δηλαδή γίνει 1,1 τότε η ροπή θα αυξηθεί κατά $(1,1)^2$, δηλαδή 21%. Έστω ότι το ρεύμα το κατεβάζω στο 1/3, τότε η ροπή θα κατέβει στο $(1/3)^2$, δηλαδή στο 1/9. Γι' αυτό αυτή η εκκίνηση δεν χρησιμοποιείται πολύ λόγω ελάττωσης της ροπής εκκίνησης .

Για να λυθούν τα παραπάνω προβλήματα χρησιμοποιούμε αυτομετασχηματιστή με πολλές λήψεις.

1.5.3 Εκκίνηση επαγωγικού κινητήρα με αυτομετασχηματιστή

Ο αυτομετασχηματιστής είναι, πιο φθηνός (από τους ίδιους) δηλαδή που θα δούλευαν στο ίδιο φορτίο επί ώρες επειδή στην εκκίνηση λειτουργεί για 10-15 sec. Έτσι, μπορεί να φορτιστεί με μεγάλο φορτίο, δηλαδή μπορεί να αντέξει σε μεγάλες πυκνότητες ρεύματος, π.χ. 20 A/mm^2, από ότι θα άντεχε αν δούλευε σε μόνιμη κατάσταση, πχ. 2 A/mm^2. Έτσι η ίδια κατασκευή μπορεί να είναι πολύ μικρή ως προς το μέγεθος. Επίσης, έχει τα προτερήματα (οικονομικά) π.χ. μπορεί να έχει μόνο ένα τύλιγμα. Παρακάτω στο σχήμα 1.14 δίνεται η παράστασή του σε μία φάση.



Σχήμα 1. 14 Εκκίνηση επαγωγικού κινητήρα με αυτομετασχηματιστή

Από πολλαπλασιασμό των σχέσεων (1) και (2) του σχήματος 1.14 προκύπτει:

$$\frac{I_\delta}{I} \bullet \frac{I}{I_d} = \frac{U}{U_\delta} \bullet \frac{U}{U_d} \Rightarrow \frac{I_\delta}{I_d} = \left[\frac{U}{U_d}\right]^2 = \left[\frac{U}{U_\delta}\right]^2$$

Επειδή :

$$\frac{M}{M_d} = \left[\frac{I}{I_d}\right]^2$$

Από τη σχέση (2) παίρνω :

$$\frac{M}{M_d} = \left[\frac{U}{U_\delta}\right]^2 \Rightarrow \frac{I_\delta}{I_d} = \frac{M}{M_d} = \left[\frac{U}{U_\delta}\right]^2 \qquad (1.16)$$

Η τελευταία σχέση θυμίζει την εκκίνηση Υ/Δ. Έτσι αφού $I_d \approx 6I_{ov}$, και θελήσουμε $I_\delta=(1/4)I_d$, δηλαδή $I_\delta=(1/4)*6I_{ov}=1,5I_{ov}$ και ροπή $M_\delta=(1/4)M_d$, θέτω :

$$\left[\frac{U}{U_\delta}\right]^2 = \frac{1}{4} \Rightarrow \frac{U}{U_\delta} = \frac{1}{2} \qquad (1.17)$$

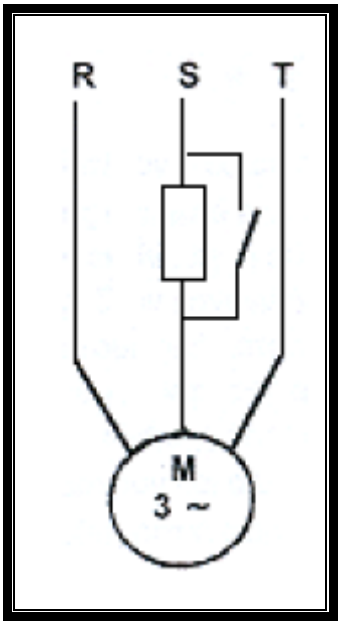
Αρκεί λοιπόν να κάνουμε την τάση μισή για να πάρουμε ρεύμα και ροπή το $\frac{1}{4}$ των αντίστοιχων ονομαστικών τιμών τους. Ειδικά για την περίπτωση όπου $I_\delta=(1/3)I_d$ ή $M_\delta=(1/3)M_d$, θα έχουμε την περίπτωση του Υ/Δ.

Έτσι με διάφορες λήψης μπορούμε να πάρουμε όποιους λόγους θέλουμε. Το επόμενο πλεονέκτημα είναι ότι μπορούμε να αποφύγουμε την αιχμή που δημιουργείται στην εκκίνηση Υ/Δ.

Αυξάνοντας στον αυτομετασχηματιστή την τάση αυξάνει η ροπή και το ροπή. Έτσι για διάφορες λήψεις έχουμε μείωση της επίδρασης της αιχμής απ' ότι είχαμε στην περίπτωση Υ/Δ. εδώ λόγω σύντομων μεταζεύξεων χρειάζονται διάφορα χρονικά ρελέ. Το πώς μπορεί να υπολογιστεί ο χρόνος μετάζευξης από βαθμίδα σε βαθμίδα γίνεται στην πράξη με δοκιμές. Υπάρχουν όμως τυποποιημένοι χρόνοι. Η συνδεσμολογία του κινητήρα μπορεί να είναι οποιαδήποτε (Υ ή Δ).

1.5.4 Εκκίνηση με μία μόνο αντίσταση στο στάτη (KUSA)

Με τη μέθοδο αυτή ο εκκινητής και ο αυτοματισμός εκκίνησης γίνεται πολύ φθηνότερος. Χρησιμοποιούμε μόνο μία αντίσταση (ωμική όπως φαίνεται και στο σχήμα 1.15.



Σχήμα 1. 15 Εκκίνηση επαγωγικού κινητήρα με μία μόνο αντίσταση στο στάτη

Το ρεύμα εκκίνησης στις δύο φάσεις R και Τα είναι μεγάλο όπως στην άμεση εκκίνηση. Στην πρώτη φάση (Κ) που έχει την πρόσθετη αντίσταση, το ρεύμα εκκίνησης είναι μικρότερο. Έτσι έχουμε μία ασύμμετρη εκκίνηση. Η συνολική ροπή εκκίνησης του ασύγχρονου κινητήρα με βραχυκυκλωμένο δρομέα ελαττώνεται. Η αντίσταση μπορεί να είναι και ρυθμιζόμενη. Έτσι, μπορούμε, με μία φθηνή σχετικά μέθοδο να προσαρμόσουμε την ροπή εκκίνησης του κινητήρα προς τη ροπή του φορτίου και ο κινητήρας να ξεκινά σιγά-σιγά. Η μέθοδος αυτή χρησιμοποιείται μόνο εφ' όσον επιτρέπεται το μεγάλο ρεύμα εκκίνησης ενώ το φορτίο στην εκκίνηση είναι μικρότερο του ονομαστικού.

1.5.5 Εκκίνηση με ηλεκτρικό ρυθμιστή τάσης

Εδώ η εναλλασσόμενη τάση τροφοδοσίας ρυθμίζεται με θυρίστορ, ενδεχομένως με ανάδραση μέσου των στροφών.

1.5.6 Εκκίνηση με μετατροπέα συχνότητας

Η τροφοδότηση γίνεται κατά την εκκίνηση με μεταβλητή συχνότητα.

1.5.7 Κατασκευές κινητήρων για βελτίωση των συνθηκών εκκίνησης

Η κύρια και σπουδαιότερη διαφορά ανάμεσα στα στοιχεία (ροπή, ρεύμα) εκκίνησης και στα ονομαστικά έγκειται στην ολίσθηση. Για την εκκίνηση s=1 ενώ η ονομαστική $s_{ον}$= 0,02÷0,05 και ο όρος $R_2/s=P_2$ στην εκκίνηση, ενώ ο ονομαστικός είναι $R_2/s=(20÷50)R_2$. στους δακτυλιοφόρους κινητήρες η απάλειψη αυτής της διαφοράς γίνεται αυξάνοντας την R_2 με την σύνδεση εξωτερικών αντιστάσεων σε

σειρά. Στους κινητήρες βραχυκυκλωμένου δρομέα προσπαθούμε με διάφορες κατασκευαστικές μεθόδους να έχουμε $R_{2εκ}$ μεγάλο και $R_{2ον}$ μικρό.

1.5.8 Ασύγχρονοι κινητήρες με διπλό κλωβό

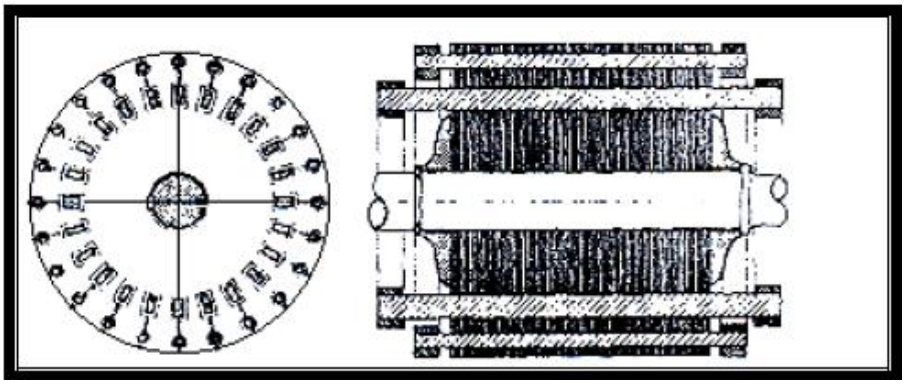
Ο κινητήρας αυτός έχει δρομέα με δύο κλωβούς. Ένας είναι εξωτερικός στη περιφέρεια του δρομέα και έναν εσωτερικό κλωβό. Κάθε κλωβός έχει τους δικούς του δακτυλίους βραχυκύκλωσης όπως φαίνεται στο σχήμα.

Οι αγωγοί του εξωτερικού κλωβού κατασκευάζονται έτσι ώστε να έχουν μεγάλη ωμική αντίσταση (μικρή διατομή, ή υλικό μεγάλης ειδικής αντίστασης) ενώ οι αγωγοί του εσωτερικού κλωβού έχουν μικρή ωμική αντίσταση (μεγάλης διατομής ή υλικό μικρότερης ειδικής αντίστασης).

Επίσης, η επαγωγική αντίσταση σκεδασμού είναι διαφορετική στους δύο κλωβούς. Ο εξωτερικός κλωβός έχει πολύ μεγαλύτερη μαγνητική απόσταση από τον στάτη παρά ο εξωτερικός κλωβός. Γι' αυτό η απώλεια σκεδασμού του εσωτερικού κλωβού, δηλαδή η ισοδύναμη επαγωγική αντίσταση σκεδασμού είναι πολύ μεγαλύτερη από αυτήν του εξωτερικού.

Επί πλέον, ο εσωτερικός κλωβός έχει πολύ μεγαλύτερο συντελεστή αυτεπαγωγής απ' ότι ο εξωτερικός κλωβός και αυτό έχει σημασία στην εκκίνηση.

Στην εκκίνηση η σύνθετη αντίσταση των ράβδων στα βαθιά αυλάκια είναι μεγαλύτερη όσο μεγαλώνει το βάθος. Στην κανονική λειτουργία γίνεται το αντίθετο. Μοιάζει σαν να μετατοπίζεται το ρεύμα ανάλογα με την λειτουργική κατάσταση του κινητήρα.



Σχήμα 1. 16 Κινητήρας επαγωγής με διπλό κλωβό

Με τους κινητήρες αυτής της κατασκευής ελαττώνεται το ρεύμα εκκίνησης [$I_{εκ}$=(2÷3) $I_{ον}$] ενώ η ροπή είναι ίση με την ονομαστική.

Υπάρχουν και άλλες όμως κατασκευές με ιδιαίτερη μορφή. Η δυνατότητα ενός κινητήρα να ξεκινήσει με αυξημένη ροπή φορτίου έχει ομαδοποιηθεί σε κλάσεις

Υπάρχουν τέσσερις κλάσεις ροπών:

KL 16, KL 13, KL 10, KL 7

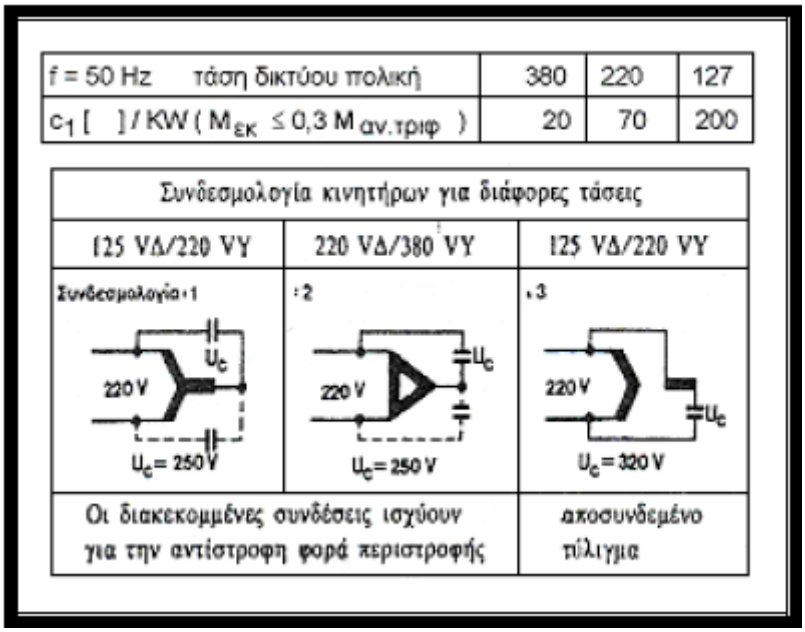
Αυτές επιτρέπουν εκκίνηση με φορτία ροπής όπως στον παρακάτω πίνακα (πίνακας 1.1), ακόμα και αν η τάση είναι κατά 5% μειωμένη.

1,6 $M_{ον}$	για L16
1,3 $M_{ον}$	για L13
1,0 $M_{ον}$	για L10
0,7 $M_{ον}$	για L7

Πίνακας 1. 1 Κλάσεις ροπών

1.5.9 Εκκίνηση και λειτουργία τριφασικού επαγωγικού κινητήρα σε μονοφασικό δίκτυο

Η συνδεσμολογία που επικράτησε για την εκκίνηση του τριφασικού κινητήρα σε μονοφασικό δίκτυο, είναι με ένα ή δύο πυκνωτές (εκκίνησης, λειτουργίας). Οι πυκνωτές συνδέονται πάντα μεταξύ μίας φάσης. Η χωρητικότητα του πυκνωτή λειτουργίας για κάθε αποδιδόμενο kW ισχύος του κινητήρα δίνεται στο παρακάτω σχήμα (σχήμα 1.17).



Σχήμα 1. 17 Σύνδεση τριφασικού επαγωγικού κινητήρα σε μονοφασικό δίκτυο

Υπάρχουν τρεις συνδεσμολογίες. Οι πυκνωτές στις συνδεσμολογίες 1 και 2 πρέπει να αντέχουν τάση ίση με το 115% Της τάσης δικτύου (=250V) και δίνονται στο σχήμα 1.17. οι πυκνωτές της συνδεσμολογίας 3 έχουν την μισή ισχύ αλλά η τάσης τους πρέπει να είναι 135% της τάσης του δικτύου (=320V).

Η ισχύς του τριφασικού κινητήρα σε μονοφασική λειτουργία είναι 70 – 80% της ισχύος σε τριφασική λειτουργία. Η ροπή εκκίνησης είναι 25 – 35% της ροπής σε τριφασικό δίκτυο.

Η χρήση τριφασικών κινητήρων σε μονοφασικό δίκτυο είναι λογικό για κινητήρες μέχρι 2kW περίπου, γιατί πέρα των 2kW το κόστος των πυκνωτών είναι σημαντικό. Η αλλαγή φοράς περιστροφής του κινητήρα αυτού γίνεται με την αλλαγή της σύνδεσης μόνο του ακροδέκτη φάσης – δικτύου του πυκνωτή. Αυτό φαίνεται στο παραπάνω σχήμα.

Κεφάλαιο 2ο Η Ιστορία των Μικροεπεξεργαστών - Μικροελεγκτών

2.1 Εισαγωγή

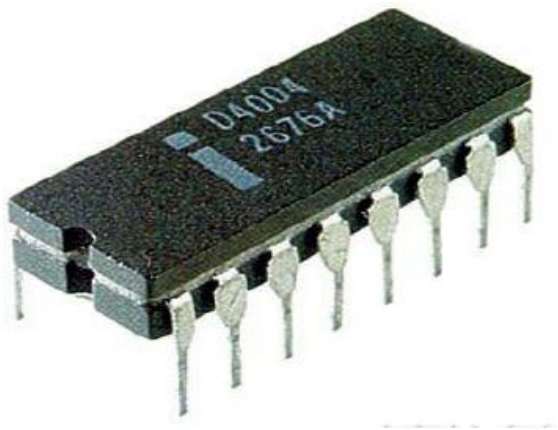
Στη σχετικά σύντομη διάρκεια ζωής του, που δε φτάνει τα 40 χρόνια, ο μικροεπεξεργαστής (Microprocessor) έχει σημειώσει τεράστια πρόοδο. Η απόδοσή του έχει βελτιωθεί αισθητά αφού διπλασιάζεται σχεδόν κάθε 18 μήνες. Στις τελευταίες τρεις δεκαετίες οι μικροεπεξεργαστές είναι υπεύθυνοι για την έμπνευση και 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 που πωλούνται ετησίως 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 επεξεργαστή 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.

2.2 Η εξέλιξη των μικροεπεξεργαστών

Στις 15 Νοεμβρίου του 1971 η Intel παρουσίασε τον πρώτο επεξεργαστή σε μορφή dual in – line package (DIP) τον Intel 4004 και ήταν ο πρώτος εμπορικά διαθέσιμος επεξεργαστής υπολογιστών. Ο Intel 4004 ήταν επεξεργαστής 4 bit που αποτελούταν από περίπου 2300 τρανζίστορ σε συχνότητα ρολογιού λίγο πάνω από τα 100 kHz. Η αρχική του εφαρμογή ήταν η δημιουργία αριθμομηχανών.

Το έτος 2011 σήμανε 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.



Σχήμα 0.1 Ο Intel 4004

Οι τέσσερις σχεδόν δεκαετίας της ιστορίας των μικροεπεξεργαστών λένε μία πραγματικά αξιοπρόσεκτη ιστορία όσων αφορά την τεχνολογική πρόοδο στην βιομηχανία υπολογιστών. Αυτή η πρόοδος φαίνεται και στο πίνακα που ακολουθεί (πίνακας 2.1).

Πίνακας 0.1 Οι δεκαετίες εξέλιξης των μικροεπεξεργαστών

	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

Η εξέλιξη των μικροεπεξεργαστών έχει κατά βάση ακολουθήσει τον περίφημο νόμο του Moore (Moore's law), που προτάθηκε από τον Gordon Moore το 1965 και σύμφωνα με τον οποίο ο αριθμός των συσκευών που μπορούν να ολοκληρωθούν 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 από τον 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 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 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 ισχυρή μορφή υπολογιστών. Η συχνότητα ρολογιού των ταχύτερων μικροεπεξεργαστών ξεπέρασαν τις συχνότητες ρολογιού των 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 επιτευχτεί 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 εντολών ώστε να μειωθούν στο ελάχιστο οι χαμένοι κύκλοι ρολογιού λόγω καθυστερήσεων του pipeline (pipeline stalls).

Πλέον, έχουμε ξεπεράσει τη τέταρτη δεκαετία 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 ακόμα. Η εστίαση που τις πρώτες τρεις δεκαετίες ήταν μόνο στον παραλληλισμό στο επίπεδο της εντολής (Instruction-level parallelism, ILP) έχει ήδη περάσει στον παραλληλισμό του επιπέδου 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 τη διαδικασία παραγωγής, τη λογική σχεδίαση, τη σχεδίαση σε επίπεδο μικροαρχιτεκτονικής και το run – time περιβάλλον του λογισμικού, έτσι ώστε να διατηρηθούν τα ίδια επίπεδα βελτίωσης της απόδοσης που επιτεύχθηκαν κατά τις προηγούμενες τρεις δεκαετίες.

2.3 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 κλπ.), η οποία περιέχει το λογισμικό του συστήματος. Σε κάποια μοντέλα, είναι δυνατό το κλείδωμα αυτής της μνήμης, μετά την εγγραφή της, ώστε να προστατεύει 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 διαγραφή και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης.
- 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 για 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 το είδος της. Ο μικροελεγκτής λειτουργεί αυτόνομα εκτελώντας το πρόγραμμα που βρίσκεται στην μνήμη του μόλις τροφοδοτηθεί.

Έτσι υπάρχουν :

- Μικροελεγκτές χωρίς μνήμη προγράμματος, οι οποίοι χαρακτηρίζονται ως ROM-less. Αυτοί παρέχουν πάντοτε μια παράλληλη αρτηρία 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 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, ICP). Αυτοί οι μικροελεγκτές έχουν ουσιαστικά αντικαταστήσει παλαιότερους τύπους EPROM που έσβηναν με υπεριώδη ακτινοβολία (από ειδικό τζαμάκι).

Αναλυτικά τα πλεονεκτήματα των μικροελεγκτών είναι :

- Αυτονομία, μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι, πολλοί μικροελεγκτές δεν χρειάζονται κανέναν 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 των χαμηλότερων ταχυτήτων λειτουργίας.
- Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους – εξόδους (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος), λόγω της μη δέσμευσης τους για τη σύνδεση εξωτερικών περιφερειακών.
- Μικρό μέγεθος συνολικού υπολογιστικού συστήματος.

Η βασική αρχιτεκτονική των μικροελεγκτών 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 τύπου Von-Neumann.

2.4 Τα μικροϋπολογιστικά συστήματα σήμερα

Αν συγκρίνει κανείς τις δυνατότητες των μικροεπεξεργαστών ή μικροελεγκτών που χρησιμοποιούνται σε μικροϋπολογιστικά συστήματα με τι 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 μία τέτοια σύγκριση είναι άδικη δεδομένου ότι δεν προορίζονται γα 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 της 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, σειριακές και παράλληλες θύρες επικοινωνίας κ.α.). διαλέγοντας τον κατάλληλο μικροελεγκτή για μια 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 μνήμες 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 τις πληροφορίες 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 ακτινών, 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 και μία 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 κατάλληλου μικροελεγκτή ή μικροελεγκτικής μονάδας (μικροελεγκτής με μνήμη RAM και ελάχιστο hardware).
- Σύνδεση με έναν ηλεκτρονικό υπολογιστή
- Τον προγραμματίζουμε να εκτελεί ένα συγκεκριμένο πρόγραμμα

- Τον τοποθετούμε στη συγκεκριμένη εφαρμογή για να εκτελέσει τη λειτουργία που του έχουμε ορίσει, όπως ο έλεγχος των στροφών ενός κινητήρα, σύστημα ασφαλείας, χρονοδιακόπτης και πολλές άλλες εφαρμογές.

Κεφάλαιο 3ο Ο μικροελεγκτής AVR

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 φθηνά 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 χαμηλής κατανάλωσης στα 1.8V (κατάληξηV).

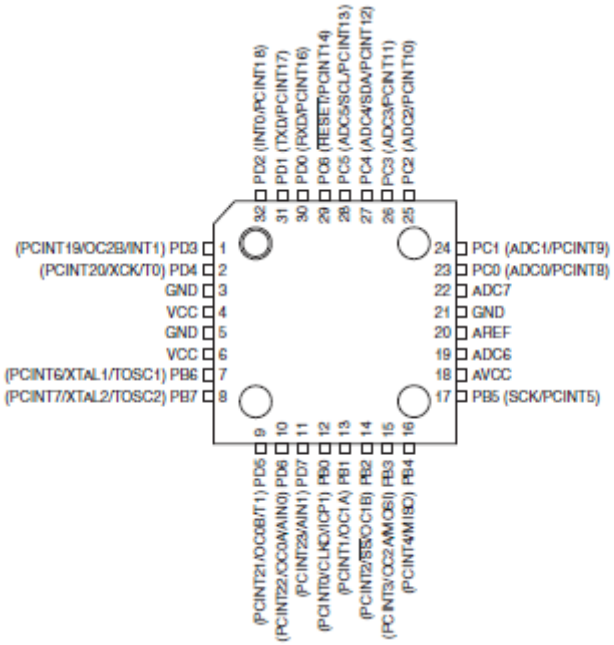
Συνήθως, οι ακροδέκτες γενικού σκοπού έχουν πολυπλεγμένες περισσότερες από μία λειτουργίες, όπως για παράδειγμα είσοδοι με ικανότητα να προκαλούν διακοπή (interrupt) στον εσωτερικό επεξεργαστή, είσοδοι αναλογικών συγκριτών ή μετατροπέων αναλογικού σε ψηφιακό (ADC), είσοδοι κεντρικού ρολογιού (oscillator) ή σύγχρονης 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 USB interface.

Στο εσωτερικό ενός μικροελεγκτή όπως ο AVR υπάρχει ένας αριθμός από διαφορετικούς τύπους μνήμης, όπως FLASH για την εγγραφή του λογισμικού συστήματος 9firmware), EEPROM για την αποθήκευση διαφόρων παραμέτρων, καθώς και κάποιος αριθμός θέσεων μνήμης RAM για τις μεταβλητές του 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 τέτοιου τύπου 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 20MHz στα πιο εξελιγμένα μοντέλα μιας σειράς οι AVR mega.

Η αρχιτεκτονική του ATmega328P

Ο μικροελεγκτής ATmega328P 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.

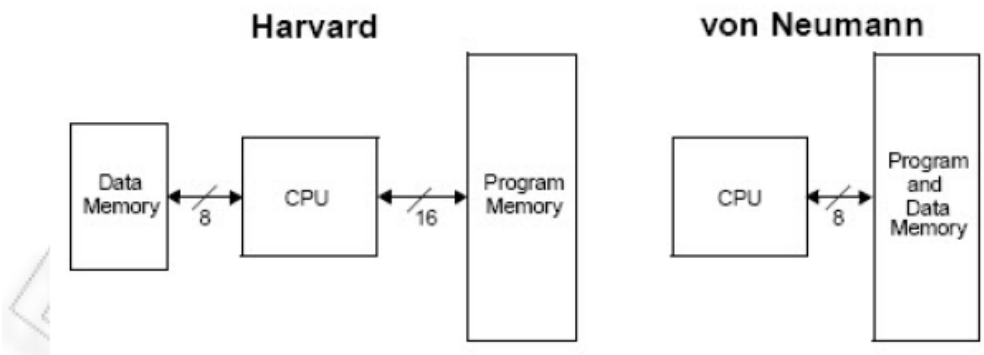


Σχήμα 3.1 Οι ακροδέκτες του ATmega328P

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 ATMEL και η εσωτερική αρχιτεκτονική τους αναπτύχθηκε περαιτέρω. Η θυγατρική της ATMEL στη Νορβηγία ιδρύθηκε από τους δύο φοιτητές. Το όνομα AVR δεν αποτελεί κάτι ιδιαίτερο όσον αφορά την ερμηνεία του. Απλά ονομάστηκε έτσι και ορίζει όλη 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 δίαυλους (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 μία εντολή, μία 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 νέα θέση μνήμης.

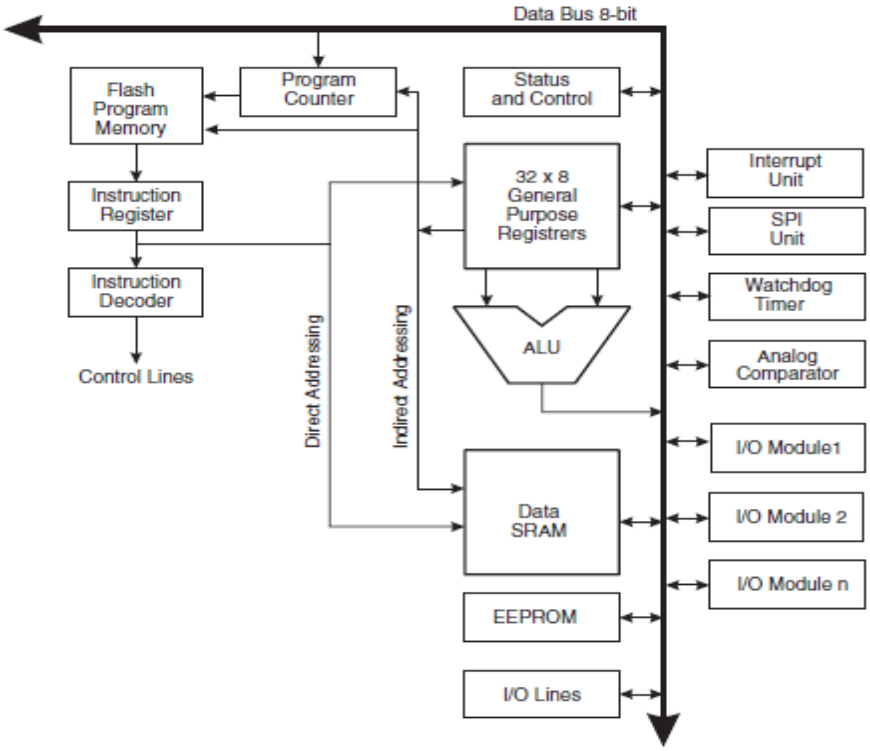
Όπως μπορεί κανείς να δει από αυτή την περιγραφή, η αρχιτεκτονική αυτή μπορεί να δημιουργήσει «κυκλοφοριακή συμφόρηση» ή ακόμη και κορεσμό. Με την αρχιτεκτονική Harvard, η πληροφορία προσκομίζεται σε έναν απλό κύκλο ρολογιού. Ενώ η μνήμη προγράμματος προσπελαύνεται, η μνήμη δεδομένων είναι σε ανεξάρτητο δίαυλο και μπορεί να διαβαστεί και να γραφτεί. Αυτοί οι χωρισμένοι δίαυλοι επιτρέπουν σε μία εντολή την εκτέλεση, ενώ η επόμενη εντολή προσκομίζεται. Μία σύγκριση των αρχιτεκτονικών Harvard και Von Neumann παρουσιάζεται και στο σχήμα 3.2.



Σχήμα 3.2 Σύγκριση Αρχιτεκτονικών Harvard – Von Neumann

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

- Μνήμη δεδομένων. Η Flash, η EPROM και η SRAM είναι όλες ενσωματωμένες σε ένα μοναδικό chip, καταργώντας την ανάγκη για την εξωτερική μνήμη. Ορισμένες συσκευές έχουν ένα 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 μνήμης του προγράμματος αναφέρεται συνήθως στην ονοματοδοσία της ίδιας της συσκευής (για παράδειγμα ο ATmega64x έχει Flash 64KB).
- Εσωτερική Μνήμη Δεδομένων. Η διευθυνσιοδότηση των δεδομένων αποτελείται από το αρχείο καταχωρητών, τους καταχωρητές I/O, και την SRAM.



Σχήμα 3.3 Αρχιτεκτονική του ATmega328P

Τεχνικά χαρακτηριστικά του ATmega328P

- Υψηλής απόδοσης, χαμηλής κατανάλωσης 8-bit μικροεπεξεργαστής AVR.
- Προηγμένη αρχιτεκτονική RISC
  - 131 ισχυρές εντολές – Οι περισσότερες εκτελέσιμες σε έναν κύκλο ρολογιού
  - 32 x 8 καταχωρητές γενικής χρήσης
  - Πλήρης στατική λειτουργία
  - Απόδοση μέχρι 16MIPS στα 16MHz
  - Ενσωματωμένο πολλαπλασιαστή δύο κύκλων
- Υψηλής αντοχής μη-πτητικά τμήματα μνήμης
  - 32K Bytes μνήμη προγράμματος Flash, αυτό-προγραμματιζόμενης πάνω στην πλακέτα, 10.000 κύκλων εγγραφής/διαγραφής
  - 1K Bytes EEPROM, 100.000 κύκλων εγγραφής/διαγραφής
  - 2K Bytes εσωτερική SRAM
  - Κλείδωμα προγράμματος για ασφάλεια λογισμικού
- Χαρακτηριστικά περιφερειακών
  - Δύο 8 bit χρονιστές/απαριθμητές με ανεξάρτητο προδιαιρέτη, μία μέθοδο σύγκρισης
  - Έναν 16 bit χρονιστή/απαριθμητή με ανεξάρτητο προδιαιρέτη, μέθοδο σύγκρισης
  - Απαριθμητή πραγματικού χρόνου με ξεχωριστό ταλαντωτή
  - Έξι κανάλια PWM
  - 8 κανάλια ADC στην συσκευασία TQFP και QFN/MLF με διακριτότητα 10bit.
  - Σειριακό προγραμματιζόμενο USART
  - Master/Slave SPI σειριακή διασύνδεση
  - Προγραμματιζόμενο Watchdog Timer με ξεχωριστό on-chip ταλαντωτή
  - On-chip αναλογικό συγκριτή
- Ειδικά χαρακτηριστικά του μικροελεγκτή
  - Power on Reset and Programmable Brown-out Detection
  - Εσωτερικό διαβαθμιζόμενο ταλαντωτή
  - Εξωτερικέ και εσωτερικές πηγές διακοπών
  - Six sleep modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O και συσκευασία
  - 23 προγραμματιζόμενες γραμμές I/O
  - 32-lead TQFP, and 32-pad QFN/MLF
- Τάση λειτουργίας:
  - 2.7V – 5.5V
- Εύρος θερμοκρασιών:
  - -40°C έως +125°C
- Ταχύτητα
  - 0 – 8 MHz
- Χαμηλή κατανάλωση ισχύος
  - Active mode: 1.5mA @3V – 4 MHz
  - Power-down mode: 1μA @ 3V

Η σημασία των ακίδων του ATmega328P

VCC: Ψηφιακή παροχή τροφοδοσίας

GND: Γείωση

Port B (PB7:0) 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 θύρας B έχουν συμμετρικά χαρακτηριστικά οδήγησης με διπλή ικανότητα, τόσο να απορροφά όσο και να πηγάζει. Σαν είσοδοι οι ακροδέκτες της θύρας B οι οποίες είναι εξωτερικά σε χαμηλό δυναμικό θα πηγάσουν ρεύμα εάν είναι ενεργοποιημένες οι pull-up αντιστάσεις. Οι ακροδέκτες της θύρας B βρίσκονται σε μία Τρίτη κατάσταση όταν οι συνθήκες του reset είναι ενεργές ακόμη και όταν το ρολόι χρονισμού δεν τρέχει.

Αναλόγως την επιλογή των ρυθμίσεων ασφαλείας του ρολογιού, ο ακροδέκτης PB6 μπορεί να χρησιμοποιηθεί σαν είσοδος στον ανάστροφο ενισχυτή ταλαντωτή και είσοδος στο εσωτερικό κύκλωμα ρολογιού.

Αναλόγως την επιλογή των ρυθμίσεων ασφαλείας του ρολογιού 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.

Port C (PC5:0)

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 θύρας C έχουν συμμετρικά χαρακτηριστικά οδήγησης με διπλή ικανότητα, τόσο να απορροφούν όσο και να πηγάζουν. Σαν είσοδοι οι ακροδέκτες της θύρας C οι οποίες είναι εξωτερικά σε χαμηλό δυναμικό θα πηγάσουν ρεύμα εάν είναι ενεργοποιημένες οι Pull-up αντιστάσεις. Οι ακροδέκτες της θύρας C βρίσκονται σε μία 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.

PC6/RESET

Εάν το RSTDISBL Fuse είναι προγραμματισμένο, το Pin PC6 χρησιμοποιείται σαν ακροδέκτης εισόδου/εξόδου. Αξιοσημείωτο είναι ότι τα ηλεκτρικά χαρακτηριστικά του PC6 διαφέρουν από εκείνα των άλλων ακροδεκτών της θύρας C

PortD (PD7:0)

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 θύρας D έχουν συμμετρικά χαρακτηριστικά οδήγησης με διπλή ικανότητα, τόσο να απορροφούν όσο και να πηγάζουν. Σαν είσοδοι οι ακροδέκτες της θύρας D οι οποίες

είναι εξωτερικά σε χαμηλό δυναμικό θα πηγάσουν ρεύμα εάν είναι ενεργοποιημένες οι Pull-up αντιστάσεις. Οι ακροδέκτες της θύρας D βρίσκονται σε μία Τρίτη κατάσταση όταν οι συνθήκες του reset είναι ενεργές ακόμη και όταν το ρολόι χρονισμού δεν τρέχει. Πολλές από τις ακίδες της θύρας D έχουν και άλλες λειτουργίες που αναφέρονται στο data sheet του μικροελεγκτή.

AV_{cc}

Ακροδέκτης τροφοδοσίας για τον μετατροπέα από αναλογικό σε ψηφιακό της θύρας PC3:0 και ADC7:6. 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 οι θύρες PC6:4 χρησιμοποιούν την τροφοδοσία για τα ψηφιακά σήματα, V_{cc}.

AREF

Είναι ο αναλογικός ακροδέκτης αναφοράς για τον A/D μετατροπέα.

ADC7:6 (Μόνο για τις συσκευασίες TQFP και QFN/MLF)

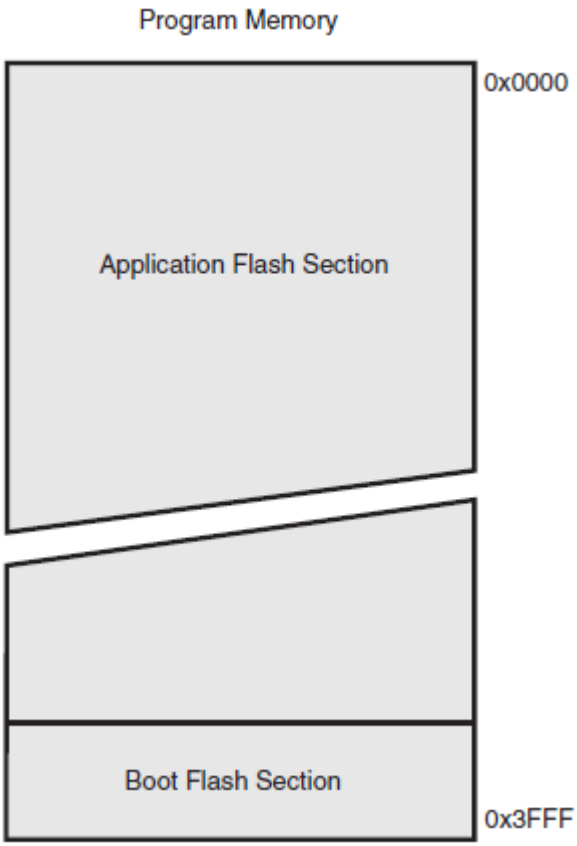
Οι μνήμες του ATmega328P

Στην παράγραφο αυτή περιγράφονται οι διαφορετικές μνήμες του ATmega328P. Η αρχιτεκτονική AVR έχει δύο κύρια τμήματα μνήμης, το τμήμα μνήμης για τα δεδομένα και το τμήμα μνήμης για το πρόγραμμα. Επιπλέον ο ATmega328P έχει και ένα τμήμα μνήμης EEPROM για την αποθήκευση δεδομένων.

Η μνήμη προγράμματος τεχνολογίας Flash

Ο ATmega328P έχει 32K bytes εσωτερική επαναπρογραμματιζόμενη μνήμη για την αποθήκευση του προγράμματος. Από τη στιγμή που όλες οι εντολές έχουν μήκος 16 ή 32 bits, η συγκεκριμένη μνήμη είναι οργανωμένη ως 16Kx16. 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 10.000 κύκλοι εγγραφής/διαγραφής.

Ο μετρητής προγράμματος (Program Counter, PC) του συγκεκριμένου μικροελεγκτή έχει μέγεθος 14 bits και κατά συνέπεια διευθυνσιοδοτεί τα 16K περιοχής προγράμματος. Η λειτουργία του τμήματος εκκίνησης περιγράφεται λεπτομερώς στο εγχειρίδιο του μικροελεγκτή. Πίνακες με σταθερές μπορούν να τοποθετηθούν εντός του τμήματος διευθύνσεων της μνήμης προγράμματος.



Σχήμα 3.4 Χαρτογράφηση της μνήμης Flash

Μνήμη δεδομένων SRAM

Το σχήμα 3.5 δείχνει τον τρόπο που είναι οργανωμένη η μνήμη SRAM του ATmega328P.

Ο ATmega328P είναι ένας σύνθετος μικροελεγκτής με περισσότερες περιφερειακές μονάδες από αυτές που μπορούν να υποστηριχθούν εντός των 64 δεσμευμένων περιοχών στο Opcode για τις εντολές εισόδου και εξόδου (I/O). Για τον Επιπρόσθετο χώρο I/O από 0X60 – 0xFF στην SRAM, μπορούν να χρησιμοποιηθούν μόνο οι εντολές ST/STS/STD και LD/LDS/LDD.

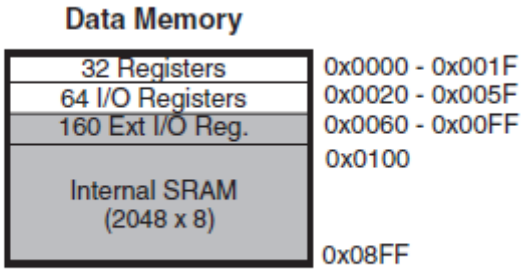
Οι χαμηλότερες 2303 θέσεις μνήμης δεδομένων απευθύνονται στον καταχωρητή αρχείου, στη μνήμη I/O, στην Επιπρόσθετη μνήμη I/O, και στα εσωτερικά δεδομένα SRAM. Οι πρώτες 32 θέσεις διευθυνσιοδοτούν τον καταχωρητή αρχείου, οι επόμενες 64 την δεδομένη μνήμη I/O, οι επόμενες 160 την Επιπρόσθετη μνήμη I/O, και οι τελευταίες 2048 την εσωτερική μνήμη δεδομένων SRAM.

Η απευθείας διευθυνσιοδότηση καλύπτει ολόκληρο το χώρο των δεδομένων.

Η έμμεση διευθυνσιοδότηση με επιπλέον όρισμα καθιστά δυνατή την προσπέλαση 63 διαφορετικών θέσεων μνήμης, έχοντας μόνο μία σταθερή βασική διεύθυνση αποθηκευμένη στον καταχωρητή Y ή Z.

Στην περίπτωση που χρησιμοποιείται καταχώρηση έμμεσης διευθυνσιοδότησης με αυτόματη προ-ελάττωση (pre-decrement) ή μετά-αύξηση (post-increment), οι καταχωρητές διευθυνσιοδότησης X, Y και Z αντιστοίχως αυξάνονται ή ελαττώνονται.

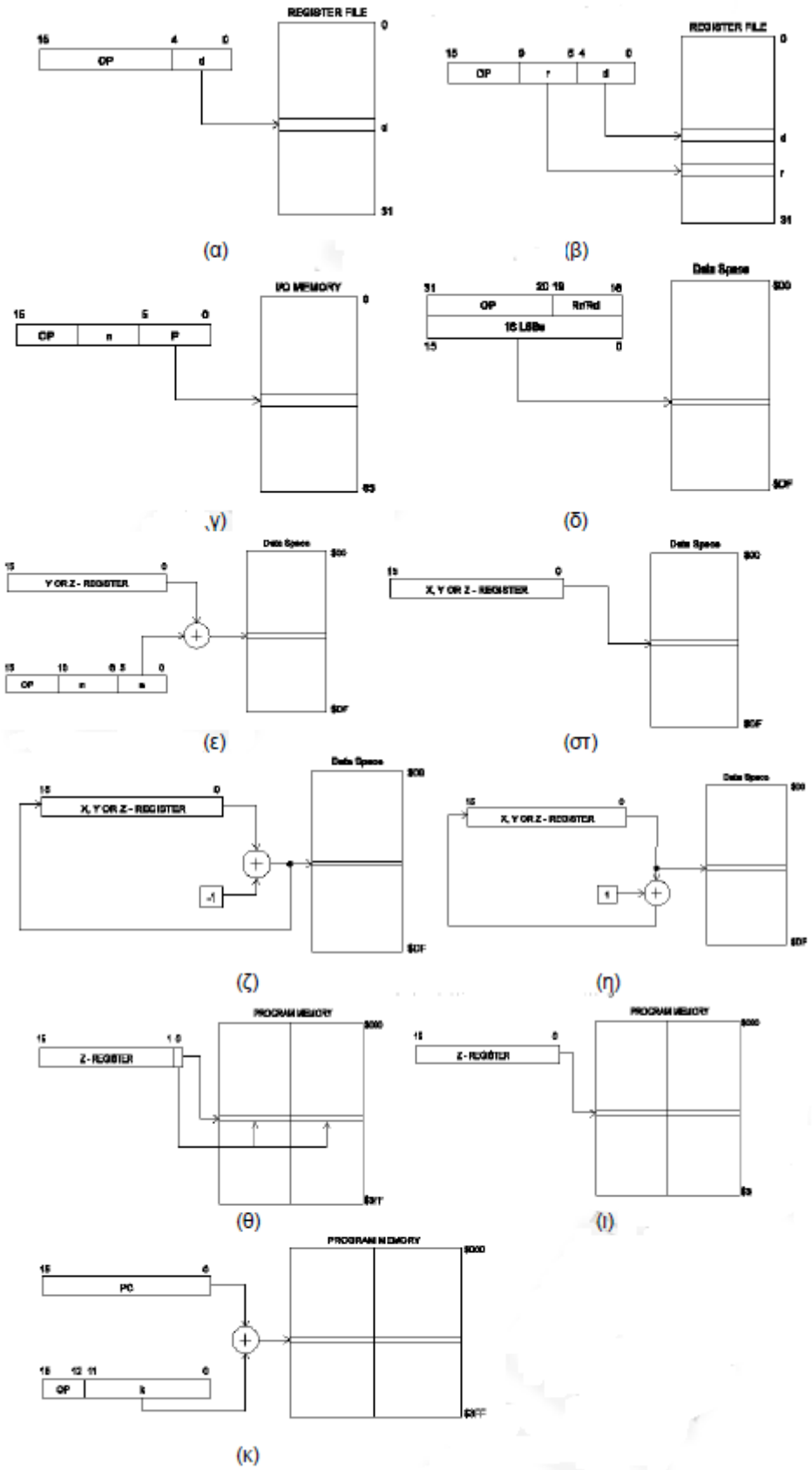
Οι 32 καταχωρητές γενικού σκοπού, οι 64 καταχωρητές I/O, οι 160 επιπρόσθετοι καταχωρητές I/O, και τα 2048 bytes της εσωτερικής μνήμης δεδομένων SRAM στον ATmega328P είναι προσιτές μέσω όλων αυτών των τρόπων διευθυνσιοδότησης.



Σχήμα 3.5 Διάταξη μνήμης SRAM

Υποστηριζόμενη τρόποι διευθυνσιοδότησης

- Άμεση Διευθυνσιοδότηση (Direct Addressing). Το μοναδικό όρισμα είναι ένας καταχωρητής (σχήμα 3.6.α)
- Άμεση με 2 καταχωρητές (Register Direct). Τα 2 ορίσματα είναι καταχωρητές (σχήμα 3.6.β)
- Άμεση σε καταχωρητή περιφερειακού (Direct I/O). Το ένα όρισμα είναι καταχωρητής εισόδου/εξόδου, δηλαδή σχετικός με κάποιο περιφερειακό (σχήμα 3.6.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 (σχήμα3.6.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 (σχήμα3.6.ε)
- Έμμεση Δεδομένων (Data Indirect). Το όρισμα βρίσκεται στη θέση μνήμης που καθορίζεται από έναν από τους καταχωρητές X, Y και Z (σχήμα 3.6.στ)
- Έμμεση Δεδομένων με μείωση ή Αύξηση του Δείκτη (Data Indirect with Pre-decrement or Post-decrement). Στην πρώτη περίπτωση ο δείκτης (X, Y ή Z) μειώνεται κατά ένα και το αποτέλεσμα καθορίζει τη θέση μνήμης στη μνήμη δεδομένων. Στη δεύτερη περίπτωση αφού χρησιμοποιηθεί ο δείκτης, κατόπιν η τιμή του αυξάνεται κατά ένα (σχήματα 3.6.ζ/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 το λιγότερο σημαντικό. (σχήμα 3.6.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 οι IJMP και ICALL.(σχήμα 3.6.ι)
- Σχετική με Μετρητή Προγράμματος (Relative Program). Το όρισμα της εντολής αυξημένο κατά ένα, προστίθεται στο Μετρητή Προγράμματος. Χρησιμοποιείται από τις εντολές RJMP και RCALL.(σχήμα3.6.κ)



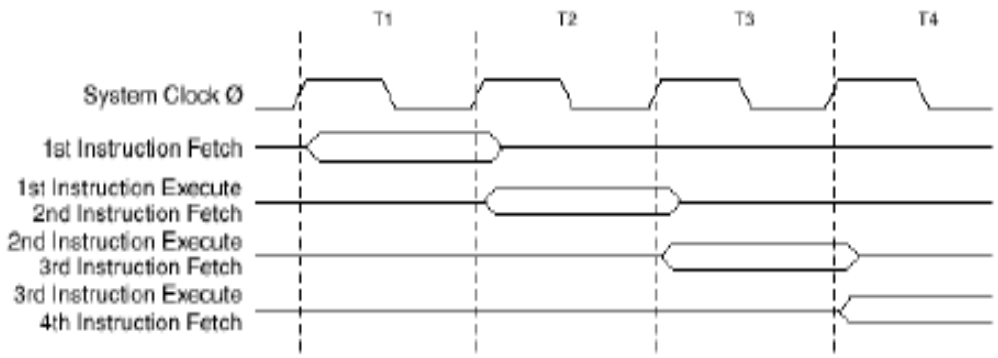
Σχήμα 3.6 Τρόποι διευθυνσιοδότησης στον AVR

Η μνήμη Δεδομένων EEPROM

Ο ATmega328P περιέχει 1K bytes μνήμης EEPROM (Electric Erasable Programmable Read Only Memory), για μνήμη αποθήκευσης δεδομένων. Όπως και η Flash Memory, η EEPROM μπορεί να διατηρήσει το περιεχόμενό της ακόμη και με απουσία τάσης τροφοδοσίας. 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 την όλη διαδικασία πολύ 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 τη μνήμη του προγράμματος.

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, υποστηρίζουν πλήρως μια τέτοια επικάλυψη με αποτέλεσμα ενώ μια εντολή μπορεί να απαιτεί περισσότερο από έναν κύκλο ρολογιού για να εκτελεστεί, ο ρυθμός εκτέλεσης των εντολών να είναι «Μία Εντολή ανά Κύκλο Ρολογιού». (Σχήμα 3.7)



Σχήμα 3.7 Εκτέλεση εντολών με επικάλυψη (pipelining)

Ρολόι Συστήματος

Ο μικροελεγκτής για να λειτουργήσει πρέπει να παλμοδοτείται από ένα κύκλωμα χρονισμού. Ο ATmega328P μπορεί να παλμοδοτηθεί από τις πηγές που φαίνονται στον παρακάτω πίνακα (πίνακας 3.1) και επιλέγονται από τα αντίστοιχα glash fuse bits CKSEL3..0.

Συνήθως συναντάται σαν ένα δικτύωμα πυκνωτή-αντιστάτη (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 υπολογιστεί ο χρόνος που θα ξεκινήσει η εκτέλεση των εντολών, εξασφαλίζοντας σταθερή λειτουργία του ταλαντωτή πριν αρχίσει η

εκτέλεση των εντολών. Η CPU ξεκινά από RESET, υπάρχει μια επιπρόσθετη καθυστέρηση επιτρέποντας στην τάση τροφοδοσίας να φτάσει σε ένα σταθερό επίπεδο πριν αρχίσει την κανονική λειτουργία.

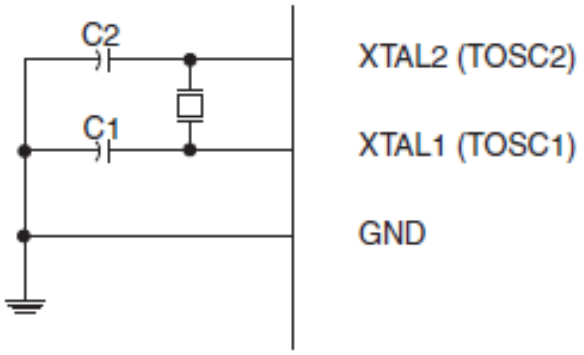
Επιλογή Συσκευή Χρονισμού	**CKSEL3..0**
Κρυσταλλικός Ταλαντωτής Χαμηλής Ισχύος	1111-1000
Κρυσταλλικός Ταλαντωτής Πλήρους Αιώρησης	0111-0110
Κρυσταλλικός Ταλαντωτής Χαμηλής Συχνότητας	0101-0100
Εσωτερικός RC Ταλαντωτής 128kHz	0011
Βαθμονομημένος Εσωτερικός RC Ταλαντωτής	0010
Εξωτερικό Ρολόι	0000
Reserved	0001

Πίνακας 3.1

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 χαρακτηριστικά του μικροελεγκτή.

Ταλαντωτής Κρυστάλλου

Οι ακροδέκτες XTAL1 και XTAL2 είναι είσοδος και έξοδος, αντίστοιχα, ενός εναλλασσόμενου ενισχυτή ο οποίος μπορεί ρυθμιστεί ώστε να χρησιμοποιηθεί σαν ένας on-chip Ταλαντωτής όπως φαίνεται στο σχήμα 3.8.



Σχήμα 3.8 Ταλαντωτής Κρυστάλλου Χαμηλής Ισχύος

Ο κρυσταλλικός ταλαντωτής είναι ένας χαμηλής ισχύος ταλαντωτής, με μειωμένη τάση αιώρησης στην έξοδο XTAL2. Αυτό παρέχει χαμηλότερη κατανάλωση ισχύος, όμως δεν είναι σε θέση να οδηγήσει άλλες εισόδους ρολογιού, και επίσης μπορεί να είναι πιο ευάλωτο στο θόρυβο σε περιοχές με ηχορύπανση. Σε αυτές τις περιπτώσεις προτιμάται ο Κρυσταλλικός Ταλαντωτής Πλήρους Αιώρησης.

Οι C1 και C2 πρέπει πάντοτε να είναι ίσοι τόσο για τους κρυστάλλους όσο και για τους resonators. Η βέλτιστη τιμή των πυκνωτών εξαρτάται από το είδος του κρυστάλλου η του resonator που χρησιμοποιείται, την περισσεύουσα χωρητικότητα και τον ηλεκτρομαγνητικό θόρυβο του περιβάλλοντος.

Χειρισμός Διακοπών

Ο Atmega328P διαθέτει 26 πηγές διακοπών και ισάριθμες θέσεις στην περιοχή 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 3.2.

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

```
Address  Labels Code                    Comments
0x0000          jmp    RESET            ; Reset Handler
0x0002          jmp    EXT_INT0         ; IRQ0 Handler
0x0004          jmp    EXT_INT1         ; IRQ1 Handler
0x0006          jmp    PCINT0           ; PCINT0 Handler
0x0008          jmp    PCINT1           ; PCINT1 Handler
0x000A          jmp    PCINT2           ; PCINT2 Handler
0x000C          jmp    WDT              ; Watchdog Timer Handler
0x000E          jmp    TIM2_COMPA       ; Timer2 Compare A Handler
0x0010          jmp    TIM2_COMPB       ; Timer2 Compare B Handler
0x0012          jmp    TIM2_OVF         ; Timer2 Overflow Handler
0x0014          jmp    TIM1_CAPT        ; Timer1 Capture Handler
0x0016          jmp    TIM1_COMPA       ; Timer1 Compare A Handler
0x0018          jmp    TIM1_COMPB       ; Timer1 Compare B Handler
0x001A          jmp    TIM1_OVF         ; Timer1 Overflow Handler
0x001C          jmp    TIM0_COMPA       ; Timer0 Compare A Handler
0x001E          jmp    TIM0_COMPB       ; Timer0 Compare B Handler
0x0020          jmp    TIM0_OVF         ; Timer0 Overflow Handler
0x0022          jmp    SPI_STC          ; SPI Transfer Complete Handler
0x0024          jmp    USART_RXC        ; USART, RX Complete Handler
0x0026          jmp    USART_UDRE       ; USART, UDR Empty Handler
0x0028          jmp    USART_TXC        ; USART, TX Complete Handler
0x002A          jmp    ADC              ; ADC Conversion Complete Handler
0x002C          jmp    EE_RDY           ; EEPROM Ready Handler
0x002E          jmp    ANA_COMP         ; Analog Comparator Handler
0x0030          jmp    TWI              ; 2-wire Serial Interface Handler
0x0032          jmp    SPM_RDY          ; Store Program Memory Ready Handler
;
0x0033RESET:    ldi    r16, high(RAMEND); Main program start
0x0034          out    SPH,r16          ; Set Stack Pointer to top of RAM

0x0035          ldi    r16, low(RAMEND)
0x0036          out    SPL,r16
0x0037          sei                      ; Enable interrupts
0x0038          <instr>  xxx
  ...     ...     ...   ...
```

Στη θέση 0 υπάρχει μία κλήση στην υπορουτίνα Reset. Το όνομα Reset αντιστοιχεί στη θέση Μνήμης Προγράμματος από 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 INT01/INT02 και στη συνέχεια οι υπόλοιπες διακοπές.

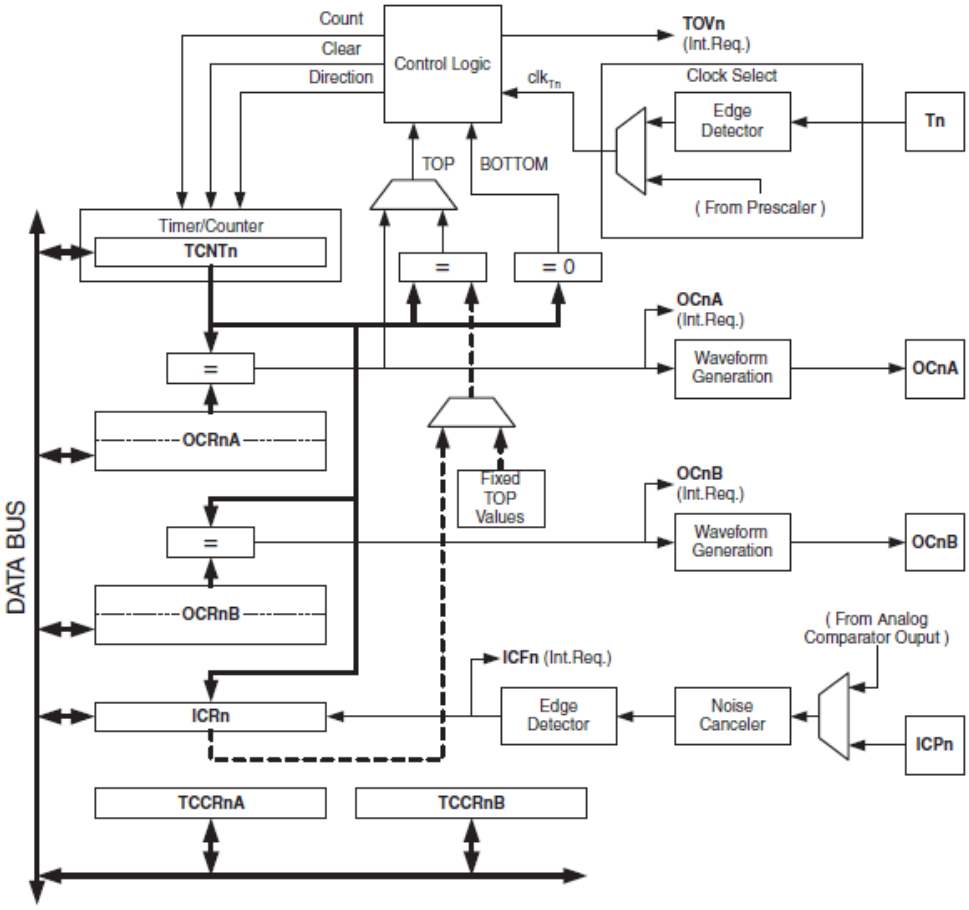
Vector No.	Program Address	Source	Interrupt Definition
1	0x0000	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Counter1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

Σχήμα 3.8 Οι πηγές διακοπών του ATmega328P

Χρονιστές/Μετρητές (Timers/Counters)

Ο ATmega328P διαθέτει τρεις (3) χρονιστές/μετρητές. Οι δύο είναι 8-bit και ονομάζονται 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, είτε να χρησιμοποιήσουν κάποια εξωτερική πηγή παλμών, των οποίων να μετράνε το πλήθος, οπότε λειτουργούν ως μετρητές. Στους T/C0 και T/C1 η συχνότητα οδήγησης των παλμών μπορεί να υποδιαιρεθεί με τις τιμές /8, /64, /256 και /1024 μέσω του κοινού prescaler. Και οι τρεις χρονιστές/μετρητές μπορούν να λειτουργήσουν και σαν γεννήτριες κυματομορφών PWM.

Ο χρονιστής/μετρητής T/C1 διαθέτει πιο περίπλοκη αρχιτεκτονική από τους άλλους δύο και διαθέτει 16-bits. Το Block διάγραμμά του παρουσιάζεται στο παρακάτω σχήμα (σχήμα 3.9)



Σχήμα 3.9 Block διάγραμμα του χρονιστή/μετρητή T/C1

ΚΕΦΑΛΑΙΟ 4° Δομή Προγράμματος

4.1 Βασικές συναρτήσεις

Όλα τα προγράμματα Arduino περιλαμβάνουν δύο θεμελιώδεις συναρτήσεις στον κώδικά τους. Αυτές είναι setup() και η loop(). Μέσω αυτόν των δύο συναρτήσεων επιτυγχάνεται η αρχικοποίηση και η δυναμική ανταπόκριση του συστήματος.

4.1.1 setup()

Η συνάρτηση setup καλείται όταν γίνεται εκκίνηση ενός sketch. Χρησιμοποιείται για να δηλωθούν μεταβλητές, να οριστούν οι λειτουργίες των ακίδων, να ορισθούν βιβλιοθήκες κλπ. Η συνάρτηση setup εκτελείται μόνο μια φορά μετά από εκκίνηση ή επανεκκίνηση της πλακέτας Arduino.

```
void setup()
      {
            Serial.begin(9600);
            pinMode(buttonPin, INPUT);
      }
```

4.1.2 loop()

Μετά την εκτέλεση της συνάρτησης setup, κατά την οποία ορίζονται οι αρχικές τιμές, εκτελείται η συνάρτηση loop. Η συνάρτηση loop κάνει ακριβώς αυτό που δηλώνει, δηλαδή συνεχείς ανακυκλώσεις του κώδικα που ακολουθεί. Μέσω της συνάρτησης loop επιτυγχάνεται η συνεχής μεταβολή και ανταπόκριση του προγράμματος στα διάφορα δεδομένα και καταστάσεις που εισάγονται στο σύστημα. Μέσω της συνάρτησης loop γίνεται ο ενεργός έλεγχος της πλακέτας Arduino.

Παράδειγμα προγράμματος:

```
const int buttonPin = 3;
//Η συνάρτηση setup εκκινεί την σειριακή επικοινωνία και
//αντιστοιχεί την ακίδα σε λειτουργία διακόπτη.

void setup()
      {
                        Serial.begin(9600);
                        pinMode(buttonPin, INPUT);
      }

// Η συνάρτηση loop ελέγχει την ακίδα που δηλώθηκε αρχικά
// σαν είσοδος και μεταφέρει το σήμα μεταβολής κατάστασης
// όταν κλείσει ο διακόπτης.
```

```
void loop()
      {
            if (digitalRead(buttonPin) == HIGH)
                              Serial.write('H');
                        else
                              Serial.write('L');

                        delay(1000);
      }
```

4.2 Δομές Ελέγχου

Η δομή ελέγχου αποτελεί μια πρωταρχική έννοια στις περισσότερες γλώσσες υψηλού επιπέδου. Η δομή ελέγχου στην απλούστερη ερμηνεία της αποτελεί ένα σύνολο εντολών το οποίο καθορίζει την ροή του ελέγχου στο πρόγραμμα. Με άλλα λόγια η δομή ελέγχου αποτελεί ένα σύνολο από συναρτήσεις, εντολές και δηλώσεις.

4.2.1 If (υπόθεση) και ==, != , <, > (τελεστές σύγκρισης)

Η if χρησιμοποιείται πάντα σε συνδυασμό με έναν τελεστή σύγκρισης. Η if ελέγχει εάν πληρείται μία συγκεκριμένη κατάσταση, όπως το εάν μία είσοδος έχει ξεπεράσει ένα συγκεκριμένο όριο. Η σύνταξη της if είναι η εξής:

```
if (metabliti > 20)
      {
            // κάνε κάτι εδώ
      }
```

Το πρόγραμμα ελέγχει εάν η τιμή της «metabliti» είναι μεγαλύτερη από 20. Εάν είναι μεγαλύτερη τότε λαμβάνει δράση. Γενικά εάν η έκφραση εντός της παρένθεσης είναι αληθής τότε εκτελείται ο κώδικας μέσα στις αγκύλες. Εάν δεν είναι αληθής τότε προσπερνάτε ο κώδικας μέσα στις αγκύλες.

Οι αγκύλες μπορούν να απαλειφθούν μετά τη δήλωση της if. Εάν απαλειφθούν οι αγκύλες εισάγεται ένα ερωτηματικό «;» στο τέλος της επόμενης γραμμής το οποίο ορίζει τον τερματισμό της υποθετική δήλωσης.

Η σύνταξη είναι η εξής:

```
if (x > 20) digitalWrite(LEDpin, HIGH);

if (x > 20)
digitalWrite(LEDpin, HIGH);

if (x > 20){ digitalWrite(LEDpin, HIGH); }

if (x > 120){
  digitalWrite(LEDpin1, HIGH);
  digitalWrite(LEDpin2, HIGH);
}                       // όλες οι συντάξεις είναι σωστές
```

Σημείωση: Οι δηλώσεις που υπολογίζονται εντός των παρενθέσεων απαιτούν έναν ή περισσότερους τελεστές.

4.2.2 if / else

Η εντολή if/else επιτρέπει καλύτερο έλεγχο επί της ροής του προγράμματος από ότι η απλή εντολή if. Η εντολή if/else επιτρέπει την ταυτόχρονη πραγματοποίηση πολλαπλών ελέγχων. Για παράδειγμα μπορεί να ελέγχεται μια αναλογική είσοδος και εάν η στάθμη της τάσης εισόδου είναι μικρότερη κάποιας τιμής να εκτελείται μια συγκεκριμένη εργασία ενώ εάν είναι μεγαλύτερη να εκτελείται κάποια άλλη.
Η σύνταξη του κώδικα σε αυτήν την περίπτωση είναι η εξής:

```
if (pinTriaInput < 300)
{
  // Δράση 1
}
else
{
  // Δράση 2
}
```

Η else μπορεί να προηγείται ενός ελέγχου if, ώστε να εκτελούνται περισσότεροι ταυτόχρονοι έλεγχοι. Κάθε έλεγχος οδηγεί στον επόμενο μέχρι να βρεθεί ένας ικανοποιήσιμος κλάδος ελέγχου. Όταν εντοπιστεί ένας ικανοποιήσιμος κλάδος ελέγχου τότε εκτελείται ο αντίστοιχος κώδικας που βρίσκεται στις αγκύλες και το πρόγραμμα μεταπηδά στην γραμμή που ακολουθεί την δομή if/else. Εάν δεν ικανοποιηθεί κανένας έλεγχος τότε εκτελείται ο κώδικας που ακολουθεί την τελική εντολή «else», εφόσον έχει οριστεί.

Σημείωση: Μια δομή if/else μπορεί να χρησιμοποιηθεί με ή χωρίς την τελική εντολή «else», ενώ μπορεί να χρησιμοποιηθεί απεριόριστος αριθμός εντολών «else if»
Η σύνταξη του κώδικα σε αυτήν την περίπτωση είναι η εξής:

```
if (pinTriaInput < 300)
{
  // Δράση 1
}
else if (pinTriaInput >= 900)
{
  // Δράση 2
}
else
{
  // Δράση 3
}
```

4.2.3 switch / case statements

Όπως και η εντολή if έτσι και η switch/ case επιτρέπει στον προγραμματιστή να ελέγχει την ροή ελέγχου του προγράμματος και να καθορίζει κώδικα που να εκτελείται σε διάφορες περιστάσεις. Πιο συγκεκριμένα η εντολή «switch» συγκρίνει την τιμή μιας μεταβλητής με την τιμή που καθορίζει η εντολή «case». Όταν βρεθεί μια εντολή «case» της οποίας η τιμή συμπίπτει με αυτήν της μεταβλητής τότε εκτελείται ο αντίστοιχος κώδικας.

Η εντολή «break» σηματοδοτεί την έξοδο από τον βρόχο συγκρίσεως της switch/ case και τυπικά τοποθετείται στο τέλος κάθε εντολής switch/ case. Η εντολή switch/ case συντάσσεται ως εξής:

```
switch (var) {
    case 2:
      //κάνει κάτι όταν η var ισούται με 2
      break;
    case 5:
      // κάνει κάτι όταν η var ισούται με 5
      break;
    default:
      // εάν δεν ταιριάζει τίποτα, τότε πράξε την default
      // η default είναι προαιρετική
  }
```

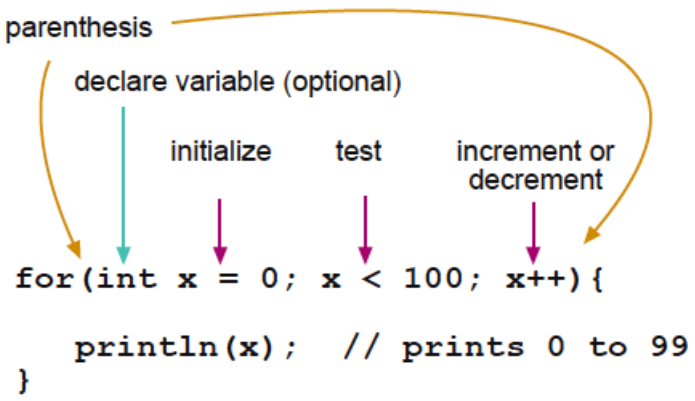
Σημείωση: Γενικά στη θέση var τοποθετείται η μεταβλητή της οποίας η τιμή συγκρίνεται με την τιμή κάθε case. Η τιμή που παρουσιάζεται δίπλα από κάθε case ονομάζεται label.

4.2.4 for statement

Η εντολή for χρησιμοποιείται για την επανάληψη ενός συνόλου εντολών που βρίσκονται ανάμεσα σε αγκύλες. Χρησιμοποιείται ένας μετρητής για τον τερματισμό των επαναλήψεων. Η εντολή for είναι χρήσιμη για κάθε επαναληπτική διεργασία και συχνά χρησιμοποιείται για τη συλλογή δεδομένων από πίνακες ή pin.

Η κεφαλίδα της for αποτελείται από τρία μέρη:

```
for (αρχικοποίηση; συνθήκη; μετρητής) {
//δήλωση(-εις);
}
```



Εικόνα 1. (πηγή Arduino.cc)

Η αρχικοποίηση λαμβάνει χώρα πρώτη και μόνο μία φορά. Σε κάθε επανάληψη η συνθήκη ελέγχεται και εάν είναι αληθής εκτελούνται οι δηλώσεις εντός των αγκυλών και αυξομειώνεται ο μετρητής, στη συνέχεια η συνθήκη επανελέγχεται και ου το καθεξής. Όταν η συνθήκη δεν πληρείται διακόπτεται η επαναληπτική διαδικασία.

Παράδειγμα:

```
// Dim an LED using a PWM pin
int PWMpin = 10; // LED in series with 470 ohm resistor
on pin 10

void setup()
{
  // δεν απαιτείται
}

void loop()
{
   for (int i=0; i <= 255; i++){
      analogWrite(PWMpin, i);
      delay(10);
   }
}
```

Η εντολή for παρουσιάζει πολύ μεγαλύτερη ευελιξία από ότι οι εντολές for άλλων γλωσσών όπως η BASIC. Οποιαδήποτε δήλωση (ή και όλες) μπορεί να απαλοιφθεί, αν και τα ερωτηματικά (;) πρέπει να παραμείνουν. Επίσης οι δηλώσεις αρχικοποίησης και αυξομείωσης μπορούν να αποτελούνται από οποιαδήποτε συνάρτηση. Οι συναρτήσεις που περιλαμβάνονται στις δηλώσεις της εντολής for μπορούν να περιλαμβάνουν διάφορες μεταβλητές καθώς και οποιοδήποτε τύπο δεδομένων, ακόμη και αριθμούς κινητής υποδιαστολής.

Παραδείγματα:

1. Η χρήση πολλαπλασιασμού στην δήλωση αυξομείωσης της εντολής for δίνει την δυνατότητα πραγματοποίησης μιας λογαριθμικής ακολουθίας.

```
for(int x = 2; x < 100; x = x * 1.5){
println(x);
}
```

2. Η εντολή for μπορεί να χρησιμοποιηθεί για τον έλεγχο ενός LED.

```
void loop()
{
   int x = 1;
   for (int i = 0; i > -1; i = i + x){
      analogWrite(PWMpin, i);
      if (i == 255) x = -1;// αλλαγή κατεύθυνσης στην
      delay(10);           // κορυφή
   }
}
```

4.2.5 Βρόγχοι while

Μέσω της εντολής while πραγματοποιούνται συνεχείς και άπειροι βρόγχοι μέχρις ότου η συνθήκη ανάμεσα στις παρενθέσεις καταστεί ψευδής. Εάν δεν μεταβληθεί η συνθήκη ελέγχου η εντολή θα πραγματοποιεί βρόγχους συνεχώς. Η συνθήκη ελέγχου μπορεί να είναι είτε μια μεταβλητή που μειώνεται είτε ένας εξωτερικός αισθητήρας. Η εντολή συντάσσεται ως εξής:

```
while(συνθήκη ελέγχου){
// δηλώσεις
}
```

4.2.6 Εντολή do-while

Η εντολή do-while λειτουργεί όπως και η εντολή while μόνο που πραγματοποιεί τον έλεγχο της συνθήκης κατά τον τερματισμό του βρόγχου και όχι στην αρχή όπως η απλή while. Συνεπώς οι προς εκτέλεση δηλώσεις προηγούνται της συνθήκης ελέγχου και εκτελούνται τουλάχιστον μία φορά. Η εντολή συντάσσεται ως εξής:

```
do
{
    // εκτελεστέες δηλώσεις
} while (συνθήκη ελέγχου);
```

Παράδειγμα:

Έλεγχος τιμής αισθητήρα που είναι συνδεδεμένος σε κάποιο pin της πλακέτας του μικροελεγκτή.

```
do
```

```
{
  delay(50);   // αναμονή για σταθεροποίηση του αισθητήρα
  x = readSensors();  // έλεγχος του αισθητήρα

} while (x < 100);
```

4.2.7 Εντολή break

Η εντολή break χρησιμοποιείται για την έξοδο από εντολές do, for, while καθώς και από εντολές switch. Μέσω της εντολής δίνεται break η δυνατότητα να παρακαμφθεί η συνθήκη ελέγχου των εντολών αυτών.

Παράδειγμα:

Έλεγχος τιμής αισθητήρα και έξοδος από τον βρόγχο ελέγχου εάν ξεπεραστεί ένα προκαθορισμένο όριο.

```
for (x = 0; x < 255; x ++)
{
    digitalWrite(PWMpin, x);
    sens = analogRead(sensorPin);
    if (sens > threshold){ // έξοδος εάν ξεπεραστεί το
       x = 0;              // προκαθορισμένο όριο
       break;
    }
    delay(50);
}
```

4.2.8 Εντολή continue

Η εντολή continue χρησιμοποιείται σε συνδυασμό με τις εντολές do, for και while. Μέσω της εντολής continue το πρόγραμμα προχωρά σε παράληψη της συγκεκριμένης επανάληψης ενός βρόγχου ελέγχου. Στη συνέχεια γίνεται έλεγχος των συνθηκών και το πρόγραμμα προχωρά σε ολοκλήρωση των υπόλοιπων επαναλήψεων που υπολείπονται για την ολοκλήρωση της εντολής (βρόγχος ελέγχου - do,for,while).

Παράδειγμα:

```
for (x = 0; x < 255; x ++)
{
    if (x > 40 && x < 120){  // δημιουργείται ένα άλμα
        continue;            // για αυτές τις τιμές
    }

    digitalWrite(PWMpin, x);
    delay(50);
}
```

4.2.9 Εντολή return

Μέσω της εντολής return επιτυγχάνεται ο τερματισμός μιας συνάρτησης και η επιστροφή της τιμής της συνάρτησης στην καλούσα συνάρτηση. Η εντολή συντάσσεται ως εξής:

```
return;
```

είτε

```
return value; // και οι δύο μορφές είναι αποδεκτές
```

Η παράμετρος value μπορεί να είναι οποιαδήποτε μεταβλητή ή τύπος σταθεράς.

4.2.10 Εντολή goto

Μέσω της εντολής goto επιτυγχάνεται η μεταφορά της ροής του προγράμματος σε άλλο σημείο του κώδικα. Η εντολή συντάσσεται ως εξής:

label: // ακολουθείται από κώδικα

goto label; // στέλνει τη ροή προγράμματος στη θέση με τη σήμανση label

4.3 Σύνταξη κώδικα

Εντολή #define

Η #define αποτελεί ένα συστατικό της γλώσσας προγραμματισμού το οποίο επιτρέπει στον προγραμματιστή να προσδίδει ένα όνομα σε μια σταθερή τιμή πριν γίνει η μεταγλώττιση του προγράμματος. Κατά την διάρκεια της διαδικασίας μεταγλώττισης γίνεται αντικατάσταση των αναφορών που ορίζονται από την #define με τις προκαθορισμένες τιμές. Η χρήση της #define δεν επιφέρει κάποια επιβάρυνση στην μνήμη του συστήματος του μικροελεγκτή.

Ορισμένες φορές παρουσιάζονται ανεπιθύμητες παρενέργειες όταν το όνομα τις σταθεράς που έχει εκχωρηθεί περιλαμβάνεται σε άλλες μεταβλητές ή σταθερές. Σε αυτήν την περίπτωση το κείμενο αντικαθίσταται από την τιμή που έχει οριστεί. Γενικά προτιμάται η χρήση της εντολής const για τον καθορισμό σταθερών.

Εντολή const

Η εντολή const χρησιμοποιείται για σταθερές και μεταβλητές. Η εντολή const καθορίζει την συμπεριφορά μιας μεταβλητής την καθιστά μη μετατρέψιμη, χωρίς να περιορίζει την χρησιμοποίησή της. Οι σταθερές που ορίζονται από την εντολή const ικανοποιούν τους κανόνες που διέπουν την εμβέλεια μεταβλητών, αυτό καθιστά την εντολή const την προσφιλέστερη μέθοδο καθορισμού σταθερών.

Εντολή #include

Η εντολή #include χρησιμοποιείται για να συμπεριληφθούν στο πρόγραμμα (sketch) εξωτερικές βιβλιοθήκες. Δίνεται η δυνατότητα στο πρόγραμμα να αποκτήσει πρόσβαση σε όλες τις βιβλιοθήκες της C καθώς και σε ειδικές βιβλιοθήκες του Arduino. Η #include όπως και η #define δεν απαιτούν την χρήση «;» κατά την σύνταξή τους. Η εντολή #include συντάσσεται ως εξής:

```
#include <onomaBibliothikis.h>
```

Ερωτηματικό «;»

Το ερωτηματικό χρησιμοποιείται, για να δηλωθεί ο τερματισμός μιας συνάρτησης. Εάν παραληφθεί το ερωτηματικό στο τέλος μιας συνάρτησης τότε θα παρουσιαστεί σφάλμα κατά την μεταγλώττιση του προγράμματος.

Αγκύλες «{}»

Σε αρκετές γλώσσες υψηλού επιπέδου γίνεται εκτεταμένη χρήση αγκυλών στην σύνταξη συναρτήσεων. Γενικά οι αγκύλες χρησιμοποιούνται, για να ορίσουν την αρχή και το τέλος των δηλώσεων που περιλαμβάνονται σε μια εντολή. Κάθε «{» αγκύλη πρέπει πάντα να ακολουθείται από μια «}» αγκύλη, αφού στην αντίθετη περίπτωση θα παρουσιαστεί σοβαρό σφάλμα κατά την μεταγλώττιση του προγράμματος.

Παρατηρήσεις

Οι παρατηρήσεις είναι κείμενα που ενημερώνουν τον αναγνώστη για τον τρόπο λειτουργίας του προγράμματος. Ο μεταγλωττιστής δεν λαμβάνει υπόψη τις παρατηρήσεις που είναι γραμμένες στον κώδικα. Αυτό έχει σαν αποτέλεσμα οι παρατηρήσεις να μην εξάγονται στον επεξεργαστή και να μην επιβαρύνουν την λειτουργία του μικροελεγκτή. Η χρήση παρατηρήσεων στον κώδικα αποσκοπεί στην επεξήγηση (ή υπενθύμιση) του τρόπου λειτουργίας του προγράμματος. Η εκτεταμένη χρήση παρατηρήσεων στον κώδικα του προγράμματος είναι θεμιτή και ενθαρρύνεται, αφού βοηθά στην κατανόηση αλλά και στην αντιμετώπιση προβλημάτων που σχετίζονται με αυτόν.

Για την εισαγωγή παρατηρήσεων στον κώδικα χρησιμοποιούνται δύο ειδών συμβολισμοί:

α) «//» για παρατηρήσεις που καταλαμβάνουν μία γραμμή.

β) «/* κείμενο */» για παρατηρήσεις που καταλαμβάνουν πάνω από μία γραμμή.

4.4 Αριθμητικοί τελεστές

Τελεστής ανάθεσης «=»

Ο τελεστής ανάθεσης αποθηκεύει την τιμή που βρίσκεται στα δεξιά(η οποία μπορεί να αποτελεί μία πράξη ή μία συνάρτηση) του στην μεταβλητή που βρίσκεται στα αριστερά του. Η χρήση του = κατά τον προγραμματισμό διαφέρει από την αλγεβρική του χρήση, όπου δηλώνει ισότητα.

Τελεστές αριθμητικών πράξεων άθροισης, αφαίρεσης, πολλαπλασιασμού και διαίρεσης «+, -, *, /»

Οι τελεστές αριθμητικών πράξεων επιστρέφουν την τιμή του συνόλου, της διαφοράς, του γινομένου και του πηλίκου αντίστοιχα. Η πράξη που λαμβάνει χώρα δίνει αποτέλεσμα ίδιου τύπου δεδομένων με τους αριθμούς που χρησιμοποιήθηκαν αρχικά, δηλαδή η διαίρεση του 8 με το 3 θα δώσει αποτέλεσμα 2, αφού το 8 και το 3 είναι ακέραιοι αριθμοί. Επίσης είναι δυνατό να εμφανιστούν σφάλματα υπερχείλισης εφόσον το αποτέλεσμα της πράξης είναι μεγαλύτερο από τον αριθμό που μπορεί να αποθηκεύσει ο τύπος δεδομένων. Εάν οι αριθμοί που χρησιμοποιούνται στην πράξη είναι διαφορετικού τύπου τότε το αποτέλεσμα της πράξης προκύπτει στη μορφή τύπου δεδομένων με την μεγαλύτερη αριθμητική χωρητικότητα. Εάν, για παράδειγμα, πραγματοποιηθεί πράξη μεταξύ ενός αριθμού τύπου κινητής υποδιαστολή και ενός αριθμού διπλής ακρίβειας τότε θα χρησιμοποιηθεί ο τύπος κινητής υποδιαστολής για την εκτέλεση της πράξης.

Οι παραπάνω τελεστές συντάσσονται ως εξής:

```
αποτέλεσμα = τιμή1 + τιμή2;
αποτέλεσμα = τιμή1 – τιμή2;
αποτέλεσμα = τιμή1 * τιμή2;
αποτέλεσμα = τιμή1 / τιμή2;
```

Τελεστής υπολογισμού υπολοίπου διαίρεσης % (modulo)

Ο τελεστής modulo επιστρέφει το υπόλοιπο της διαίρεσης δύο αριθμών και είναι πολύ χρήσιμος, για να συγκρατεί εντός ορίων μία μεταβλητή (π.χ. το μέγεθος ενός πίνακα). Ο τελεστής modulo δεν γίνεται να χρησιμοποιηθεί με αριθμούς κινητής υποδιαστολής. Η σύνταξη του τελεστή γίνεται ως εξής:

```
αποτέλεσμα = διαιρετέος % διαιρέτης
```

Παράδειγμα χρήσης του τελεστή:

```
x = 9 % 7;    // ο x περιέχει 2
x = 13 % 9;   // ο x περιέχει 4
x = 6 % 6;    // ο x περιέχει 0
x = 7 % 9;    // ο x περιέχει 4
```

4.5 Τελεστές συγκρίσεως

Η χρήση των τελεστών σύγκρισης γίνεται σε συνδυασμό με την εντολή if η οποία έχει αναλυθεί εκτενέστερα στην παράγραφο 1.2.1 . Πιο συγκεκριμένα οι ερμηνείες των τελεστών είναι οι εξής:

```
 x == y // ο x είναι ίσος με τον  y
 x != y // ο x δεν είναι ίσος με τον y
 x <  y // ο x είναι μικρότερος του y
 x >  y // ο x είναι μεγαλύτερος του y
 x <= y // ο x είναι μικρότερος ή ίσος του y
 x >= y // ο x είναι μεγαλύτερος ή ίσος του y
```

4.6 Λογικοί τελεστές (τελεστές Boole)

Οι λογικοί τελεστές είναι οι logical AND (&&), logical OR (||), NOT (!) και μπορούν να χρησιμοποιηθούν εντός άλλων εντολών όπως η υποθετική if. Αναλυτικότερα η logical AND επιστρέφει true εάν επιβεβαιώνονται όλες οι συνθήκες της, για παράδειγμα συνθήκες μπορούν να αποτελούν οι έξοδοι δύο διακοπτών. Συντάσσεται ως εξής:

```
if (digitalRead(5) == HIGH  && digitalRead(6) == HIGH) {
// διαβάζει δύο διακόπτες
  …
}
```

Η logical OR επιστρέφει true εάν επιβεβαιώνεται έστω και μία από τις συνθήκες της. Συντάσσεται ως εξής:

```
if (digitalRead(5) == HIGH  || digitalRead(6) == HIGH) {
// διαβάζει έστω και έναν διακόπτη
  …
}
```

Η NOT επιστρέφει true εάν η συνθήκη δεν επιβεβαιώνεται ή είναι μη πραγματική. Η NOT συντάσσεται ως εξής:

```
if (!x) { // εάν ο x είναι 0 τότε εκτελείται ο κώδικας
          //εντός των {}
  …
}
```

4.7 Τελεστές δεικτών

Στην γλώσσα προγραμματισμού του μικροελεγκτή (C και wiring) μπορούν να χρησιμοποιηθούν οι τελεστές δεικτών * και &.

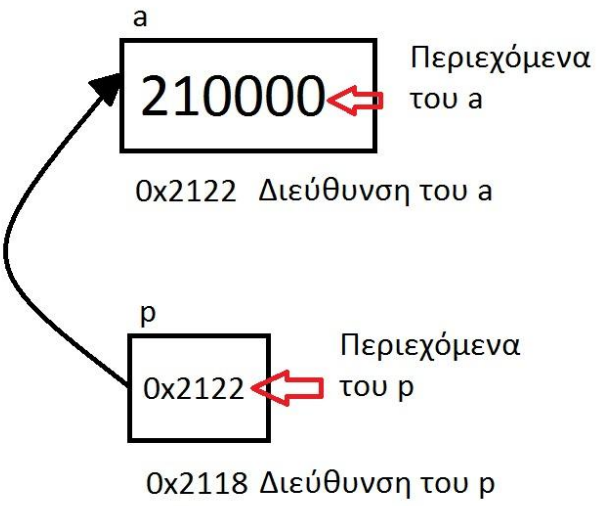
Ο τελεστής * χρησιμοποιείται για να δηλώσει ότι μία μεταβλητή αποτελεί δείκτη. Δηλαδή ότι μια μεταβλητή θα χρησιμοποιηθεί για να δείξει στη θέση μνήμης μιας άλλης μεταβλητής. Η δήλωση γίνεται ως εξής:

```
int *c;
```

Ο τελεστής & χρησιμοποιείται για την ανάθεση δείκτη, δηλαδή για την ανάθεση της θέσης μνήμης μιας μεταβλητής σε μια μεταβλητή δείκτη. Η μορφή της σύνταξης του τελεστή & είναι η ακόλουθη:

```
p=&a
```

Στο παρακάτω σχήμα ακολουθεί μια σχηματική παράσταση της παραπάνω εντολής.



Εικόνα 1. (πηγή Γιάννης Χατζόπουλος)

Η μία μεταβλητή δείχνει στη θέση μνήμης της άλλης μεταβλητής. Μία απευθείας αναφορά στα δεδομένα της μεταβλητής a μπορεί να γίνει μέσω της χρήσης του τελεστή *. Η αναφορά στα δεδομένα της a γίνεται με την έκφραση *p , μία δήλωση της μορφής `*p==100;` θα αποθηκεύσει στην μεταβλητή a την τιμή 100.

4.8 Δυαδικοί τελεστές

Οι δυαδικοί τελεστές πραγματοποιούν υπολογισμούς όχι σε δεκαδική μορφή αλλά απευθείας σε δυαδική μορφή. Με αυτόν τον τρόπο δίνεται η δυνατότητα να μεταβληθούν συγκεκριμένα bit των μεταβλητών, επιτρέποντας έτσι την επίλυση προβλημάτων που προκύπτουν κατά τον προγραμματισμό. Οι δυαδικοί τελεστές χρησιμοποιούνται για πράξεις μεταξύ ακεραίων τιμών και επιτρέπουν την μεταβολή και απομόνωση συγκεκριμένων bit καθώς και την δεξιά ή αριστερά ολίσθηση των bit που απαρτίζουν την τιμή μιας μεταβλητής.

Δυαδικός τελεστής AND &

Μία πράξη με τον δυαδικό τελεστή AND έχει ως αποτέλεσμα το 1 μόνο εάν και οι δύο είσοδοι-αριθμοί είναι 1, σε οποιαδήποτε άλλη περίπτωση έχει ως

αποτέλεσμα το 0. Η πράξη του δυαδικού τελεστή AND έχει ως σύμβολο το & και συντάσσεται ως εξής:

```
μεταβλητή = 1ος αριθμός & 2ος αριθμός
```

Στη συνέχεια ακολουθεί ένα παράδειγμα πράξης με τον δυαδικό τελεστή AND:

```
0  0  1  1    1ος αριθμός
0  1  0  1    2ος αριθμός
----------
0  0  0  1    1ος αριθμός & 2ος αριθμός
```

Στην γλώσσα προγραμματισμού του Arduino ο τύπος int αποτελείται από 16 bit, αυτό έχει ως αποτέλεσμα κατά την διάρκεια της πράξης AND(&) μεταξύ δύο ακεραίων να πραγματοποιούνται 16 ταυτόχρονες πράξεις AND(&) μεταξύ των επιμέρους bit που απαρτίζουν τους δύο ακεραίους. Μία από τις πιο συνηθισμένες χρήσεις της εντολής AND(&) είναι η απομόνωση συγκεκριμένων bit, μέσω της χρήσης μιας δυαδικής μάσκας, που μας επιτρέπει να αφήσουμε αναλλοίωτα συγκεκριμένα bit της ακέραιας τιμής.

Δυαδικός τελεστής OR |

Η πράξη του δυαδικού τελεστή OR έχει ως αποτέλεσμα 1 εάν οποιαδήποτε από τις δύο εισόδους-αριθμούς είναι 1,εάν και οι δύο είναι 0 τότε δίνει αποτέλεσμα 0. Η πράξη του δυαδικού τελεστή ORσυμβολίζεται με | και συντάσσεται ως εξής:

```
μεταβλητή = 1ος αριθμός | 2ος αριθμός
```

Στη συνέχεια ακολουθεί ένα παράδειγμα πράξης με τον δυαδικό τελεστή OR:

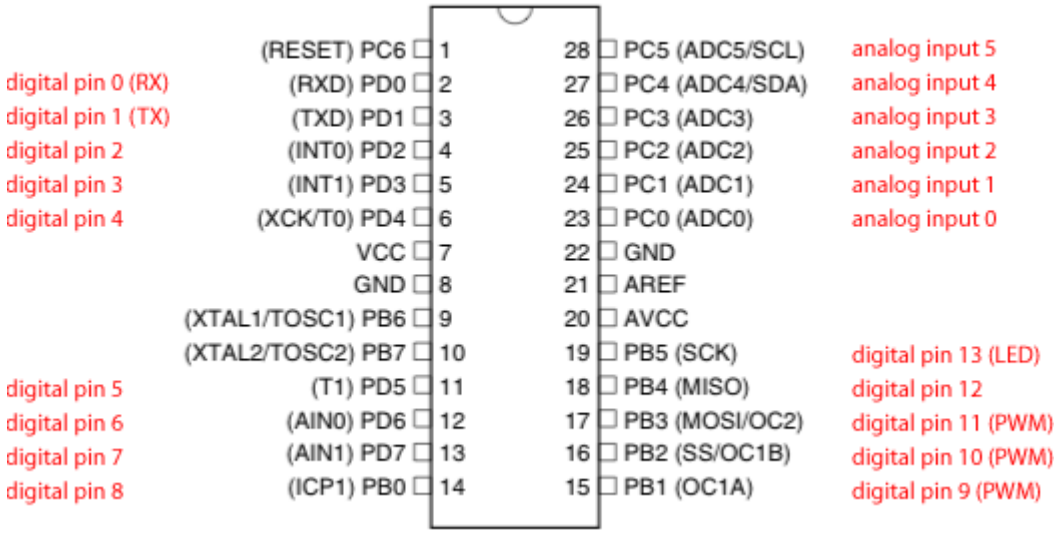
```
0  0  1  1    1ος αριθμός
0  1  0  1    2ος αριθμός
----------
0  1  1  1    1ος αριθμός | 2ος αριθμός
```

Μια συνηθισμένη διεργασία που πραγματοποιείται μέσω των δυαδικών εντολών AND και OR είναι η Ανάγνωση – Μετατροπή – Εγγραφή σε συγκεκριμένες port (πόρτα) του μικροελεγκτή. Στον προγραμματισμό των μικροελεγκτών μία θύρα (port) αποτελεί έναν αριθμό μήκους 8 bit ο οποίος υποδηλώνει την κατάσταση των ακίδων (pin) του μικροελεγκτή. Η εγγραφή σε μία θύρα (port) επηρεάζει ταυτόχρονα όλα τα pin (ακίδες) στα οποία αναφέρεται η θύρα (port).

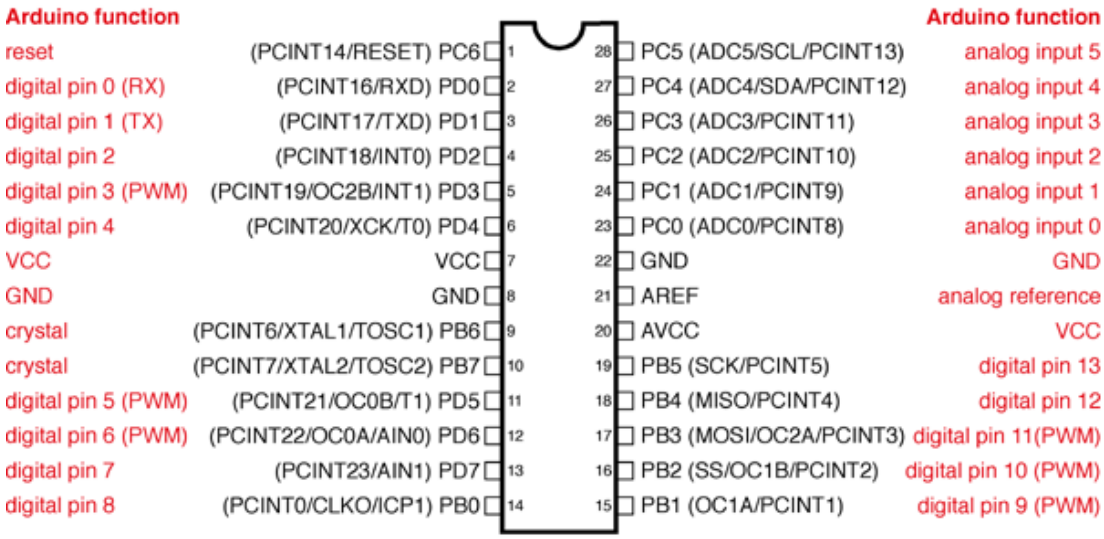
Ο καταχωρητής Arduino περιλαμβάνει τρεις θύρες:
1)Την θύρα B (port B) η οποία περιλαμβάνει τις ψηφιακές ακίδες(pin) 8 έως 13.
2)Την θύρα C (port C) η οποία περιλαμβάνει τις αναλογικές ακίδες(pin).
3)Την θύρα D (port D) η οποία περιλαμβάνει τις ψηφιακές ακίδες(pin) 0 έως 7.

Κάθε θύρα (port) ελέγχεται από τρεις καταχωρητές, οι οποίοι ταυτόχρονα αποτελούν και μεταβλητές της γλώσσας προγραμματισμού του Arduino. Οι καταχωρητές επιτρέπουν ακριβέστερο και γρηγορότερο έλεγχο των ακίδων (pin) εισόδων / εξόδων του μικροελεγκτή. Ο καταχωρητής DDR ελέγχει την συμπεριφορά των ακίδων(pin) ως εισόδους ή εξόδους. Ο καταχωρητής PORT ελέγχει εάν οι ακίδες (pin) της θύρας βρίσκονται σε κατάσταση HIGH ή LOW και ο καταχωρητής PIN διαβάζει την κατάσταση των ακίδων (pin) εισόδου μέσω της εντολής pinMode(). Στους καταχωρητές DDR και PORT μπορούν να πραγματοποιηθούν οι λειτουργίες της γραφής και της ανάγνωσης, ενώ στον καταχωρητή PIN η μόνη λειτουργία που πραγματοποιείται είναι αυτή της ανάγνωσης.

Παρακάτω παρουσιάζονται τα διαγράμματα ακίδων δύο μικροεπεξεργαστών που χρησιμοποιούνται για την υλοποίηση του μικροελεγκτή Arduino.



Εικόνα 1. Μικροεπεξεργαστής ATmega8 (Πηγή Arduino.cc)



Εικόνα 1. Μικροεπεξεργαστής ATmega168 (Πηγή Arduino.cc)

Στα παραπάνω σχήματα διακρίνονται οι θύρες των μικροεπεξεργαστών που απαρτίζουν τον μικροελεγκτή και η αντιστοίχησή τους στις ακίδες. Αναλυτικότερα έχουμε τρεις θύρες στις οποίες αντιστοιχούν από τρεις καταχωρητές αντίστοιχα.

Η **Θύρα B** αποτελείται από τις ψηφιακές ακίδες (pin) 8 έως 13, τα δύο υψηλότερα bit (PB6 και PB7) αντιστοιχίζονται στις ακίδες του ρολογιού(κρυστάλλου) και δεν γίνεται να χρησιμοποιηθούν. Οι καταχωρητές της Θύρας B είναι οι εξής:

DDRB - Ο καταχωρητής κατεύθυνσης δεδομένων της θύρας B καθορίζει εάν οι ακίδες της θύρας B λειτουργούν ως είσοδοι ή ως έξοδοι και επιτρέπει την εγγραφή και ανάγνωση δεδομένων στη θύρα.
PORTB - Ο καταχωρητής δεδομένων της Θύρας B καθορίζει εάν οι ακίδες βρίσκονται σε κατάσταση HIGH ή LOW και επιτρέπει την εγγραφή και ανάγνωση δεδομένων στη θύρα.
PINB - Ο καταχωρητής εισόδων της θύρας B, περιέχει την κατάσταση των ακίδων που αντιστοιχούν σε εισόδους και επιτρέπει μόνο την ανάγνωση δεδομένων.

Η **Θύρα C** αποτελείται από τις αναλογικές ακίδες (pin) 0 έως 5 Οι καταχωρητές της Θύρας B είναι οι εξής:

DDRC - Ο καταχωρητής κατεύθυνσης δεδομένων της θύρας C καθορίζει εάν οι ακίδες της θύρας B λειτουργούν ως είσοδοι ή ως έξοδοι και επιτρέπει την εγγραφή και ανάγνωση δεδομένων στη θύρα.
PORTC - Ο καταχωρητής δεδομένων της Θύρας C καθορίζει εάν οι ακίδες βρίσκονται σε κατάσταση HIGH ή LOW και επιτρέπει την εγγραφή και ανάγνωση δεδομένων στη θύρα.
PINC - Ο καταχωρητής εισόδων της θύρας C, περιέχει την κατάσταση των ακίδων που αντιστοιχούν σε εισόδους και επιτρέπει μόνο την ανάγνωση δεδομένων.

Η **Θύρα D** αποτελείται από τις ψηφιακές ακίδες (pin) 0 έως 7. Οι καταχωρητές της Θύρας D είναι οι εξής:

DDRD - Ο καταχωρητής κατεύθυνσης δεδομένων της θύρας D καθορίζει εάν οι ακίδες της θύρας D λειτουργούν ως είσοδοι ή ως έξοδοι και επιτρέπει την εγγραφή και ανάγνωση δεδομένων στη θύρα.
PORTD - Ο καταχωρητής δεδομένων της Θύρας D καθορίζει εάν οι ακίδες βρίσκονται σε κατάσταση HIGH ή LOW και επιτρέπει την εγγραφή και ανάγνωση δεδομένων στη θύρα.
PIND - Ο καταχωρητής εισόδων της θύρας D, περιέχει την κατάσταση των ακίδων που αντιστοιχούν σε εισόδους και επιτρέπει μόνο την ανάγνωση δεδομένων.

Τα ονόματα όλων των καταχωρητών αποτελούν ταυτόχρονα και μεταβλητές της γλώσσας προγραμματισμού του Arduino. Μέσω αυτής της λειτουργίας επιτρέπεται ο έλεγχος και ο καθορισμός κάθε θύρας.

Η PORTD αποτελεί ταυτόχρονα και μια ενσωματωμένη μεταβλητή η οποία αναφέρεται στην κατάσταση εξόδου των ψηφιακών ακίδων(pin) 0,1,2,3,4,5,6,7. Εάν σε κάποια θέση bit υπάρχει 1 τότε το αντίστοιχο pin (ακίδα) βρίσκεται σε κατάσταση HIGH. Για παράδειγμα εάν δοθεί η εντολή `PORTD = B10000110;`

τότε τα pin (ακίδες) 1,2 και 7 θέτονται σε κατάσταση HIGH. Επίσης δίνεται η δυνατότητα να μεταβληθούν τα pin (ακίδες) 0 και 1 μέσω των οποίων γίνεται η σειριακή επικοινωνία του μικροεπεξεργαστή με άλλες συσκευές.

Παράδειγμα

Στο παράδειγμα αυτό λαμβάνονται οι τιμές των bit που αντιστοιχούν στην θύρα D μέσω της PORTD και στη συνέχεια γίνεται καθαρισμός των bit που αντιστοιχούν σε pin που θέλουμε να ελέγξουμε. Στη συνέχεια γίνεται συνδυασμός των bit που έχουν μεταβληθεί με μία νέα τιμή ώστε να επιτευχθεί το θεμιτό αποτέλεσμα.

```
int i;     // μετρητής
int j;

void setup(){
DDRD = DDRD | B11111100; // καθορίζεται η κατεύθυνση
//(είσοδος ή έξοδος)των δεδομένων των bit που
//αντιστοιχούν στα pin 2 έως 7 με τα pin 0 και 1 να
//παραμένουν στην αρχική τους κατάσταση (xx | 00 == xx).
Serial.begin(9600);
}

void loop(){
for (i=0; i<64; i++){

PORTD = PORTD & B00000011;  // Μηδενισμός των bit 2 - 7,
//τα bit 0 και 1 παραμένουν τα ίδια (xx & 11 == xx).
j = (i << 2);//γίνεται ολίσθηση αριστερά κατά 2 θέσεις
//ώστε να αποφευχθούν τα bit 0 και 1.
PORTD = PORTD | j; //συνδυασμός των δεδομένων της θύρας
//με τα νέα δεδομένα ώστε να παρουσιαστούν στα LED.
Serial.println(PORTD, BIN);
delay(100);
   }
}
```

Δυαδικός τελεστής XOR ^

Η πράξη του δυαδικού τελεστή XOR αποδίδει τα ίδια αποτελέσματα με αυτήν της OR μόνο που το αποτέλεσμα του είναι 0 (αντί για 1) εάν και οι δύο είσοδοι είναι 1. Αναλυτικότερα, εάν οι είσοδοι – bit είναι διαφορετικές τότε το αποτέλεσμα είναι 1, ενώ εάν είναι ίδιες τότε το αποτέλεσμα είναι 0. Η πράξη του δυαδικού τελεστή XOR παρίσταται με το σύμβολο ^ και συντάσσεται ως εξής:

```
μεταβλητή = 1ος αριθμός ^ 2ος αριθμός
```

Τα αποτελέσματα της πράξης του δυαδικού τελεστή XOR φαίνονται παρακάτω:

```
0  0  1  1    1ος αριθμός
0  1  0  1    2ος αριθμός
----------
0  1  1  0    1ος αριθμός ^ 2ος αριθμός
```

Πρακτικά ο δυαδικός τελεστής XOR χρησιμοποιείται για την εναλλαγή της κατάστασης συγκεκριμένων bit μιας δυαδικής έκφρασης ακεραίων. Έτσι εάν σε κάποια θέση πόρτας κάποιο bit έχει την τιμή 1 τότε αυτή γίνεται 0, ενώ εάν είναι 0 παραμένει ως έχει. Στο παράδειγμα που ακολουθεί φαίνεται πως επιτυγχάνεται αυτό.

```
// Εναλλαγή του Pin 5
void setup(){
DDRD = DDRD | B00100000; //ορισμός του pin 5 ως έξοδος
Serial.begin(9600);
}

void loop(){
PORTD = PORTD ^ B00100000;  //εναλλαγή της τιμής του bit5
//(εάν είναι 1)ενώ τα υπόλοιπα παραμένουν ανέγγιχτα.
delay(100);
}
```

Δυαδικός τελεστής NOT ~

Η πράξη με τον δυαδικό τελεστή NOT έχει ως αποτέλεσμα να αντικαθιστά κάθε bit ενός αριθμού με το αντίθετό του, τα 0 με 1 και τα 1 με 0. Σε αντίθεση με τις υπόλοιπες πράξεις ο δυαδικός τελεστής εφαρμόζεται σε έναν αριθμό και όχι σε δύο. Το αποτέλεσμα της πράξης ενός αριθμού με τον δυαδικό τελεστή NOT φαίνεται παρακάτω.

```
0  1       αριθμός1
----
1  0     ~ αριθμός1
```

Τελεστές ολίσθησης <<, >>

Η ολίσθηση των bit, δεξιά ή αριστερά επιτρέπει την επίλυση αρκετών προβλημάτων προγραμματισμού. Στο προγραμματιστικό περιβάλλον του Arduino η εντολή δεξιάς ή αριστερής ολίσθησης συντάσσεται ως εξής:

```
μεταβλητή << αριθμός θέσεων
μεταβλητή >> αριθμός θέσεων
```

Η εντολή ολίσθησης επενεργεί στην μεταβλητή που συντάσσεται αριστερά του τελεστή και ολισθαίνει τα bit του περιεχομένου της μεταβλητής όσες θέσεις δηλώνει η τιμή δεξιά του τελεστή. Ο τύπος της μεταβλητής μπορεί να είναι byte, int ή long, ενώ η μέγιστή τιμή θέσεων ολίσθησης μπορεί να είναι 32. Κατά την

ολίσθηση των περιεχομένων δεξιά η αριστερά εξαφανίζονται τα δεξιά ή αριστερά bit αντίστοιχα. Το αποτέλεσμα μιας ολίσθησης φαίνεται παρακάτω.

```
0  0  1  1  1  1  0  0    περιεχόμενο μεταβλητής
---------------------
1  1  1  1  0  0  0  0    περιεχόμενο μεταβλητής << 2
```

Η δεξιά ολίσθηση διαφέρει από την αριστερή ανάλογα με τον τύπο δεδομένων που ολισθαίνουν. Αν ο τύπος δεδομένων είναι για παράδειγμα int το σημαντικότερο bit (15) καθορίζει το πρόσημο του αριθμού. Κατά την δεξιά ολίσθηση θα έχουμε εισαγωγή 1 στις αριστερές θέσεις που δημιουργούνται κατά την ολίσθηση, το οποίο ενδέχεται να μην αποτελεί το θεμιτό αποτέλεσμα. Το φαινόμενο αυτό ονομάζεται επέκταση προσήμου. Η χρήση μη προσημασμένου ακεραίου παρεμποδίζει την επέκταση προσήμου κατά την ολίσθηση. Στον παρακάτω κώδικα φαίνεται η διαφοροποίηση από την εκτέλεση των δύο περιπτώσεων της εντολής.

Περίπτωση επέκτασης προσήμου:

```
int x=-16;  //            1111111111110000
int y=x>>3; //αποτέλεσμα: 1111111111111110
```

Αντιμετώπιση φαινομένου:

```
int x=-16;                //           1111111111110000
int y=(unsigned int)x>>3; //αποτέλεσμα 0001111111111110
```

Η ολίσθηση bit παρουσιάζει δύο ακόμη ιδιότητες. Κατά την αριστερή ολίσθηση επιτυγχάνεται ο πολλαπλασιασμός ενός αριθμού με δυνάμεις του 2,δηλαδή όσες φορές αναφέρει η τιμή στα δεξιά του τελεστή. Για παράδειγμα:

```
1 <<  0  ==    1
1 <<  1  ==    2
1 <<  2  ==    4
1 <<  3  ==    8
```

Αντίστοιχα, κατά την δεξιά ολίσθηση έχουμε διαίρεση ενός αριθμού με δυνάμεις του 2, τόσες όσες αναφέρει η τιμή στα δεξιά του τελεστή. Για παράδειγμα:

```
int x=40;
int y=x>>3; //ακέραιη διαίρεση του x με 8, y=5
```

Τελεστές αυξομείωσης

Στην γλώσσα προγραμματισμού του Arduino, όπως και σε άλλες γλώσσες προγραμματισμού, χρησιμοποιούνται οι τελεστές αυξομείωσης μεταβλητών ++ και -- . Ανάλογα με την τοποθέτηση του τελεστή πριν ή μετά την μεταβλητή διαφέρει και το επιστρεφόμενο αποτέλεσμα της πράξης. Αναλυτικότερα:

```
x++; //αυξάνεται ο x κατά ένα και επιστρέφεται η αρχική
     //τιμή του x.
++x; //αυξάνεται ο x κατά ένα και επιστρέφεται η νέα τιμή
     //του x.
```

```
x--; //μειώνεται ο x κατά ένα και επιστρέφεται η αρχική
     //τιμή του x.
--x; //μειώνεται ο x κατά ένα και επιστρέφεται η νέα τιμή
     //του x.
```

Η μεταβλητή χ μπορεί να είναι τύπου int ή μη προσημασμένου long

Συντομεύσεις πράξεων

Για την απλοποίηση του χρησιμοποιούμενου κώδικα δίνεται η δυνατότητα συντόμευσης των μαθηματικών και δυαδικών πράξεων πράξεων. Παρακάτω παρατίθενται οι βασικές απλοποιήσεις με τις ερμηνείες τους.

```
x += y; // ισοδύναμο με την έκφραση x = x + y;
x -= y; // ισοδύναμο με την έκφραση x = x - y;
x *= y; // ισοδύναμο με την έκφραση x = x * y;
x /= y; // ισοδύναμο με την έκφραση x = x / y;
x &= y; // ισοδύναμο με την έκφραση x = x & y;
x |= y; // ισοδύναμο με την έκφραση x = x | y;
```

ΚΕΦΑΛΑΙΟ 5° Σταθερές και Μεταβλητές

5.1 Σταθερές

Οι σταθερές είναι προκαθορισμένες μεταβλητές που έχουν οριστεί ως const και η τιμή τους δεν δύναται να μεταβληθεί κατά την διάρκεια του προγράμματος. Οι σταθερές ομαδοποιούνται σε διάφορες κατηγορίες.

5.1.1 Λογικές σταθερές (Boolean) : True (Αληθής) - False (Ψευδής)

False

Η false (ψευδής) είναι η πιο εύκολα ερμηνεύσιμη σταθερά. Η false ορίζεται ως λογικό «0».

True

Η true (αληθής) συνήθως ορίζεται ως λογικό «1», πράγμα το οποίο δεν είναι ψευδές, όμως η true έχει και μια ευρύτερη ερμηνεία. Οποιοσδήποτε μη μηδενικός ακέραιος είναι true, υπό την έννοια της άλγεβρας Boole. Άρα το -1, το 2 ακόμα και το -200 ορίζονται ως true.

5.1.2 Σταθερές HIGH (Ύψους) - LOW (Βάθους)

Στο σημείο αυτό κρίνεται σκόπιμο να αναλυθούν οι σχετικές εντολές ορισμού, ανάγνωσης και εγγραφής pin (ακίδων) του μικροελεγκτή.

pinMode

Η εντολή pinMode ορίζει την κατάσταση του pin (ακίδα) μέσω της εντολής pinMode. Δίνεται η δυνατότητα να ρυθμιστεί η χρήση του pin (ακίδας) ως είσοδος ή ως έξοδος.

digitalRead:

Η εντολή digitalRead χρησιμοποιείται κατά την λειτουργία ενός pin (ακίδα) ως είσοδο, σκοπός της είναι να διαβάζει το συγκεκριμένο pin (ακίδα) και εάν η στάθμη της τάσης είναι πάνω από 3 Volt να αναφέρει HIGH.

digitalWrite:

Η εντολή digitalRead αποδίδει την τιμή HIGH(5V) ή LOW(0V) σε ένα συγκεκριμένο pin (ακίδα) του μικροελεγκτή.

HIGH

Η σημασία της HIGH (σε σχέση με κάποιο pin) είναι διαφορετική εάν το pin είναι ορισμένο ως είσοδος ή ως έξοδος. Όταν ένα pin έχει οριστεί ως είσοδος με την εντολή pinMode και έχει αναγνωστεί με την εντολή digitalRead θα αναφέρει HIGH εάν παρουσιαστεί μια τάση μεγαλύτερη των 3 Volt.

Ένα pin μπορεί να οριστεί ως είσοδος με την εντολή pinMode και στη συνέχεια να γίνει HIGH με την εντολή digitalWrite, αυτό θα ρυθμίσει τις αντιστάσεις των 20kΩ που χρησιμοποιούνται για pullup, έτσι ώστε να οδηγήσουν μια είσοδο σε κατάσταση HIGH εκτός εάν επιβληθεί κατάσταση LOW από το εξωτερικό κύκλωμα, που είναι συνδεδεμένο στο pin.

Όταν ένα pin οριστεί ως έξοδος μέσω της εντολής pinMode και τεθεί σε κατάσταση HIGH μέσω της εντολής digitalWrite, τότε η έξοδος παρουσιάζει σταθερή τάση 5V. Σε αυτήν την κατάσταση μπορεί να τροφοδοτήσει με ρεύμα ένα LED το οποίο είναι συνδεδεμένο σε σειρά με μία αντίσταση προς τον ουδέτερο ή προς μία άλλη έξοδο που έχει οριστεί ως LOW.

LOW

Η LOW παρουσιάζει, επίσης, διαφορετική ερμηνεία ανάλογα με το εάν ένα pin οριστεί ως είσοδος ή ως έξοδος. Όταν ένα pin ρυθμιστεί ως είσοδος μέσω της εντολής pinMode και αναγνώσει μέσω της εντολής digitalWrite τάση μικρότερη ή ίση των 2 Volt τότε αναφέρει LOW.

Ένα pin ορίζεται ως είσοδος μέσω της εντολής pinMode και η ανάγνωσή του γίνεται μέσω της εντολής digitalWrite. Ο μικροελεγκτής αναφέρει Low εάν εμφανιστεί μια τάση 2 Volt ή μικρότερη στο άκρο pin.

Όταν ένα pin έχει προρυθμιστεί ως έξοδος και έχει τεθεί σε κατάσταση LOW τότε παρουσιάζει τάση 0 Volt. Σε αυτήν την κατάσταση μπορεί να συνδεθεί στην κάθοδο ενός LED και να το ανάψει εφόσον η άνοδος είναι συνδεδεμένη στα 5 Volt μέσω μιας αντίστασης προστασίας ή σε ένα pin που έχει τεθεί σε κατάσταση HIGH.

Ρυθμίζοντας δύο OUTPUTS σε HIGH – LOW, αντίστοιχα, μπορεί να ελεγχθεί η ενεργοποίηση ενός LED.

5.1.3 Σταθερές Input (εισόδου) - Output (εξόδου)

Είσοδοι - Inputs

Οι είσοδοι είναι ρυθμισμένες ώστε να παρουσιάζουν υψηλή αντίσταση εισόδου(100MΩ), ως αποτέλεσμα παρουσιάζουν ελάχιστη κατανάλωση ρεύματος. Μπορούν χρησιμοποιηθούν για να διαβάσουν έναν αισθητήρα, όχι όμως για να ανάψουν ένα LED.

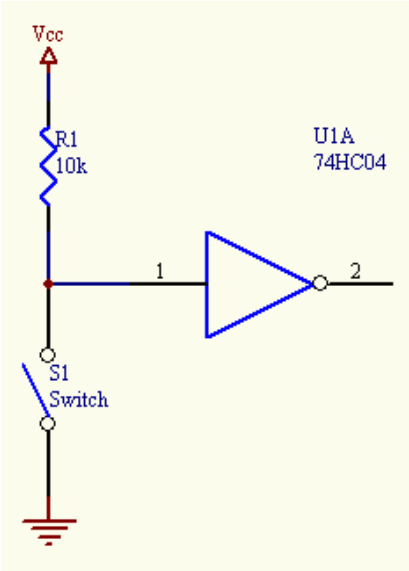
Κατά την λειτουργία ως INPUT καλό είναι να υπάρχει μια σύνδεση με την «γη» μέσω μιας αντίστασης γειώσεως, για προστασία, ώστε να απορροφά οποιοδήποτε ρεύμα.

Έξοδοι - Outputs

Τα pin που ρυθμίζονται ως έξοδοι παρουσιάζουν χαμηλή αντίσταση εξόδου, είναι ικανά να παρέχουν ποσότητα ρεύματος(40 mA) ώστε να τροφοδοτήσουν άλλα κυκλώματα. Τα pin που χρησιμοποιούνται ως έξοδοι μπορούν να τροφοδοτήσουν LED, αλλά είναι ευαίσθητα σε βραχυκυκλώματα. Δεν είναι ικανά για να τροφοδοτήσουν κινητήρες ή να οπλίσουν ηλεκτρονόμους, για αυτόν τον λόγο απαιτείται κύκλωμα προσαρμογής.

Αντιστάσεις pull-up

Στα ψηφιακά ηλεκτρονικά κυκλώματα οι είσοδοι των πυλών που δεν είναι συνδεδεμένες σε κάποια πηγή ή στη γη ενδέχεται να παρουσιάσουν αυξημένη ευαισθησία σε ηλεκτρικούς θορύβους, με αποτέλεσμα οι έξοδοί τους να μεταβάλλονται λανθασμένα και να παρουσιάζονται προβλήματα αστάθειας στο κύκλωμα. Για την επίλυση αυτού του προβλήματος πραγματοποιείται η σύνδεση σε κάποια τάση (π.χ. 5V) μέσω μιας αντίστασης, όπως φαίνεται και στο παρακάτω σχήμα.



Εικόνα 2. Κύκλωμα αντίστασης Pull-Up (Πηγή: www.seattlerobotics.org)

Αναλυτικότερα όταν ο διακόπτης S1 είναι ανοιχτός η είσοδος 1 της πύλης παρουσιάζει τάση Vcc (π.χ. 5 Volt), αφού δεν ρέει ρεύμα λόγω της υψηλής αντίστασης εισόδου των ολοκληρωμένων.

Μετά το κλείσιμο του διακόπτη S1 η είσοδος 1 συνδέεται απ'ευθείας με τη γη και η τάση της γίνεται 0. Ανάμεσα στον κόμβο τροφοδοσίας και την γη ρέει μια μικρή ποσότητα ρεύματος η οποία δεν επηρεάζει την λειτουργία του λογικού κυκλώματος. Με αυτήν την συνδεσμολογία εξασφαλίζεται αναισθησία σε ηλεκτρομαγνητικούς θορύβους που μπορούν να παρουσιάσουν αρκετά δυσάρεστα αποτελέσματα.

Ο μικροελεγκτής Arduino διαθέτει την δυνατότητα σύνδεσης αντιστάσεων pull-up στις εισόδους του. Οι αντιστάσεις pull-up είναι της τάξεως των 20kΩ.

INPUT_PULLUP:

Η εντολή INPUT_PULLUP χρησιμοποιείται για να ρυθμίσει τις αντιστάσεις που χρησιμοποιούνται για pullup, ώστε να οδηγήσουν μια είσοδο σε κατάσταση HIGH εκτός εάν επιβληθεί η κατάσταση LOW από το εξωτερικό κύκλωμα (το οποίο μπορεί να είναι διακόπτης ή αισθητήρας κ.τ.λ.).

Ανάγνωση ψηφιακής σειριακής εισόδου

Αρχικά πρέπει να γίνει έναρξη μια σειριακής επικοινωνίας μεταξύ του μικροεπεξεργαστή και του pin που θα επιλεγεί με ρυθμό 9600 bps. Η δήλωση γίνεται στην setup function.

```
Serial.begin(9600);
```

Στη συνέχεια πρέπει να γίνει δήλωση κατάστασης λειτουργίας του pin. Δηλαδή είσοδος ή έξοδος. δήλωση γίνεται στην setup function.

```
pinMode(2,INPUT);
```

Η είσοδος είναι έτοιμη να λειτουργήσει και να αποστείλει δεδομένα (0 και 1 στον υπολογιστή).

Στη συνέχεια απαιτείται να αντιστοιχηθεί η είσοδος σε κάποια μεταβλητή του προγράμματος η οποία και θα περιλαμβάνει τα δεδομένα που λαμβάνονται από την είσοδο (π.χ. κάποιον αισθητήρα). Η αντιστοίχηση εισόδου-μεταβλητής γίνεται στον τμήμα loop του προγράμματος.

Εφόσον τα δεδομένα της εισόδου είναι 0 ή 1 ο τύπος μεταβλητής εισόδου μπορεί να είναι: int datatype . Η δήλωση της μεταβλητής γίνεται ως εξής:

```
int sensorValue = digitalRead(2);
```

Όταν ο μικροεπεξεργαστής αναγνώσει την είσοδο μπορούμε να τυπώσουμε την είσοδο αυτή στην οθόνη του υπολογιστή σε δεκαδική μορφή, μέσω της εντολής Serial.println(). Η εντολή εμφάνισης συντάσσεται ως εξής:

```
Serial.println(sensorValue);
delay(1);
```

5.1.4 Ακέραιες σταθερές - Integer Constants

Ακέραιες τιμές

Οι ακέραιοι είναι ο πρωτεύον τύπος δεδομένου που χρησιμοποιείται για την αποθήκευση αριθμών.

Στον μικροελεγκτή Arduino Uno (και στους μικροελεγκτές που περιλαμβάνουν επεξεργαστές ATMega) ο τύπος int αποθηκεύει προσημασμένους ακέραιους αριθμούς μήκους 16 bit (2 byte) και παραχωρεί ένα εύρος τιμών από το -32,768 έως το 32,767 (ελάχιστη τιμή το -2^15 και μέγιστη τιμή το (2^15) - 1).

Ο τύπος int αποθηκεύει τους αρνητικούς αριθμούς σε συμπλήρωμα ως προς 2. Το σημαντικότερο bit του αριθμού, το οποίο καλείται και ως bit πρόσημου, δηλώνει

ότι ο αριθμός είναι αρνητικός, τα υπόλοιπα bit αντιστρέφονται και προστίθεται «1» στο τέλος.

Το Arduino διευθετεί τους αρνητικούς αριθμούς έτσι ώστε οι πράξεις να γίνονται σωστά και με τα αναμενόμενα αποτελέσματα. Όμως, υπάρχει η περίπτωση να υπάρξουν επιπλοκές με την εντολή μετακίνησης όλων των bit κατά μία θέση δεξιά (δεξής τελεστής ολίσθησης >>).

Παράδειγμα σύνταξης:

```
int var = val;
```

Όπου var το όνομα της ακεραίας μεταβλητής και val η ακέραια τιμή που θέλουμε να αποδώσουμε στη μεταβλητή αυτή.

Όταν οι μεταβλητές υπερβούν το μέγιστο εύρος τιμών τους, τότε αυτόματα «μεταφέρεται» στην τιμή που αντιστοιχεί στο ελάχιστο εύρος τιμών τους.

Παράδειγμα:

```
int x;
   x = -32768;
   x = x - 1;        //ο x περιλαμβάνει τον 32.767
                     //περιστρέφεται με αρνητική φορά.

   x = 32767;
   x = x + 1;        //ο x περιλαμβάνει τον -32.768
                     //περιστρέφεται με θετική φορά.
```

Ακέραιες σταθερές

Οι ακέραιες σταθερές είναι αριθμοί που χρησιμοποιούνται απευθείας στον κώδικα ενός sketch, όπως π.χ. 123, για να αποδώσουν τιμή σε κάποια μεταβλητή. Εξ ορισμού οι αριθμοί αυτοί χρησιμοποιούνται ως θετικοί ακέραιοι (int), μπορούν όμως να μεταβληθούν με την χρήση των τελεστών «U» και «L».

Εξ ορισμού οι αριθμοί θεωρούνται ότι έχουν βάση το 10, δηλαδή ότι αναφερόμαστε στο δεκαδικό σύστημα. Εάν θέλουμε να αποδώσουμε μια τιμή σε κάποιο άλλο σύστημα (δυαδικό, οκταδικό ή δεκαεξαδικό) πρέπει να τοποθετήσουμε τον αντίστοιχο δείκτη στην αρχή της αριθμητικής ακολουθίας.

Βάση Συστήματος	**Παράδειγμα**	**Δείκτης**
Δυαδικό	B11110011	«B»
Οκταδικό	0173	«0»
Δεκαδικό		
Δεκαεξαδικό	0x7B	«0x»

Παρατηρήσεις:

I. Για το δυαδικό σύστημα έχουμε το περιορισμό ότι μπορούν να αποδοθούν μόνο τιμές 8 bit έτσι το σύνολο τιμών είναι 0 - 255 στο δεκαδικό. Οι αποδεκτοί χαρακτήρες είναι 0 και 1.
II. Για το οκταδικό σύστημα αποδεκτοί είναι οι χαρακτήρες 0 - 7.

III. Για το δεκαεξαδικό σύστημα αποδεκτοί είναι οι χαρακτήρες 0 - 9, A - F και a - f.

5.1.5 Σταθερές κινητής υποδιαστολής – Floating Point Constants

Παρόμοια με τις ακέραιες σταθερές ορίζονται και οι σταθερές κινητής υποδιαστολής. Οι σταθερές κινητής υποδιαστολής αντικαθίστανται, κατά την διάρκεια της μεταγλώττισης με την τιμή που παριστάνει η παράσταση τους.

Παράδειγμα:

```
n=.00007;
```

Οι σταθερές κινητής υποδιαστολής μπορούν να εκφραστούν με μία μεγάλη ποικιλία επιστημονικών σημειογραφιών. Τα σύμβολα e και E είναι αποδεκτά.

Παράδειγμα:

Σταθερά κινητής υποδιαστολής	**Αποτέλεσμα υπολογισμού**	**Εναλλακτικό αποτέλεσμα υπολογισμού**
10.0	10	
1.45E6	1.45*10^6	1450000
45e-11	45.0 * 10^-11	.00000000045

5.2 Τύποι Δεδομένων Μεταβλητών - Data Types

void

Ο τύπος void χρησιμοποιείται για να χαρακτηρίσει συναρτήσεις που δεν πρόκειται να επιστρέψουν κάποιο δεδομένο στη συνάρτηση που τις κάλεσε.

boolean

Οι μεταβλητές που ορίζονται ως Boolean μπορούν να λάβουν μόνο δύο τιμές, είτε true είτε false. Κάθε μεταβλητή τύπου boolean καταλαμβάνει ένα Byte μνήμης.

char

Ο τύπος δεδομένων char αποθηκεύει αλφαριθμητικούς χαρακτήρες οι οποίοι καταλαμβάνουν μέγεθος ενός Byte. Οι αλφαριθμητικοί χαρακτήρες αποθηκεύονται ως αριθμοί με βάση την κωδικοποίηση ASCII και μετατρέπονται σε αλφαβητικούς χαρακτήρες για να παρουσιαστούν. Στον τύπο char περιλαμβάνονται και προσημασμένοι αριθμοί από το -128 έως το 127. Γεια την εισαγωγή ενός αλφαριθμητικού χαρακτήρα σε κάποια μεταβλητή χρησιμοποιείται το εξής σύμβολο: ’ B ’ όπου B ο αλφαριθμητικός χαρακτήρας. Για την εισαγωγή περισσότερων χαρακτήρων χρησιμοποιείται η εξής σύνταξη: ”BCD”. Αναλυτικότερα

```
char mCharacter = 'B';
char mCharacter = 66; // και οι 2 εκφράσεις είναι
//ισοδύναμες.
```

Στην δεύτερη περίπτωση η ανάθεση τιμής γίνεται μέσω του αριθμού ASCII.

unsigned char

Ο τύπος δεδομένων unsigned char αποθηκεύει μη προσημασμένους αλφαριθμητικούς χαρακτήρες οι οποίοι καταλαμβάνουν μέγεθος ενός Byte. Το εύρος αριθμητικών τιμών που μπορούν να καταλάβουν είναι από 0 έως 255. Όπως και οι char έτσι και οι unsigned char μπορούν να αποθηκεύσουν αλφαβητικούς χαρακτήρες μέσω της κωδικοποίησης ASCII.

byte

Ο τύπος δεδομένων byte αποθηκεύει έναν αριθμό μήκους 8 bit με εύρος τιμών από το 0 έως το 255. Η σύνταξη της μεταβλητής είναι η εξής:

```
byte a=B10101;//Το "B" υποδηλώνει ότι ο αριθμός είναι
//στο δυαδικό σύστημα,(B10101=20 στο δεκαδικό σύστημα).
```

unsigned int

Στον μικροελεγκτή UNO, καθώς και σε άλλους μικροελεγκτές που χρησιμοποιούν τον μικροεπεξεργαστή της σειράς ATMEGA, οι μη προσημασμένοι ακέραιοι αριθμοί (unsigned int) αποθηκεύουν μια τιμή μήκους 2 Byte (16 bit) ακριβώς όπως και οι προσημασμένοι ακέραιοι. Η διαφορά τους έγκειται στο εύρος τιμών που μπορούν να λάβουν, αυτό περιλαμβάνει το αριθμητικό διάστημα μεταξύ 0 έως 65,535 (2^{16}-1). Στους μη προσημασμένους ακέραιους (unsigned int) το υψηλότερο bit δεν περιλαμβάνει πρόσημο, όπως στους int. Οι μη προσημασμένοι ακέραιοι συντάσσονται ως εξής:

```
unsigned int var = val;
```

Όπου var η μεταβλητή και val η τιμή της μεταβλητής.

world

Οι μεταβλητές world έχουν μήκος δεδομένων 16 bit και αποθηκεύουν έναν αριθμό μεταξύ 0 έως 65535 όπως και οι unsigned int.

long

Οι long αποτελούν τύπο δεδομένων που αποθηκεύει προσημασμένους αριθμούς μήκους 4 Byte (32 bit). Το εύρος αριθμητικών τιμών της μεταβλητής είναι από -2,147,483,648 έως 2,147,483,647 και συντάσσονται όπως και οι int.

unsigned long

Ο τύπος unsigned long αποθηκεύει 4 Byte (32 bit) θετικών αριθμητικών δεδομένων. Το εύρος τιμών τους εκτείνεται από το 0 έως το 4,294,967,295, (2^{32}-1) και συντάσσονται όπως και οι unsigned int.

short

Ο τύπος δεδομένων short αποθηκεύει μια προσημασμένη αριθμητική τιμή μήκους 2 Byte (16 bit). Το εύρος τιμών του τύπου δεδομένων short βρίσκεται μεταξύ του -32,768 έως 32,767, (-2^{15} έως 2^{15}-1) και συντάσσεται όπως και ο τύπος δεδομένων int.

float

Ο τύπος δεδομένων float αποθηκεύει έναν αριθμό κινητής υποδιαστολής, δηλαδή έναν δεκαδικό αριθμό, μήκους δεδομένων 4 Byte (32 bit). Το εύρος τιμών που μπορεί να λάβει μια μεταβλητή τύπου float είναι από $-3,4028235*10^{38}$ έως $3,4028235*10^{38}$. Οι δεκαδικοί αριθμοί χρησιμοποιούνται για να καταγράψουν αναλογικές και συνεχείς τιμές με μεγαλύτερη ακρίβεια.

Ο τύπος δεδομένων float διαθέτει ακρίβεια λίγων δεκαδικών ψηφίων, σε αντίθεση με άλλα συστήματα όπου μπορεί να γίνει χρήση του τύπου δεδομένων double, σους μικροελεγκτές Arduino ο τύπος double διαθέτει το ίδιο μέγεθος, συνεπώς και την ίδια ακρίβεια με τον τύπο float.

Οι μαθηματικοί υπολογισμοί με την χρήση δεκαδικών αριθμών είναι συνήθως πιο αργοί από ότι οι υπολογισμοί ακεραίων. Για αυτό τον λόγο συνήθως αποφεύγονται ή αντικαθίστανται όπου αυτό είναι εφικτό με υπολογισμούς ακεραίων, για να αυξηθεί η ταχύτητα.

Πίνακες Arrays

Οι πίνακες ορίζονται ως σύνολα μεταβλητών (ίδιου τύπου δεδομένων) οι οποίες καταλαμβάνουν συνεχόμενες θέσεις μνήμης. Σε κάθε θέση μνήμης ενός πίνακα αποδίδεται ένας μοναδικός αριθμός ευρετηρίου μέσω του οποίου γίνεται εφικτή η πρόσβαση στα δεδομένα. Ένας πίνακας μπορεί να χρησιμοποιηθεί για να διευκολύνει την αποθήκευση και την επεξεργασία μεγάλων ποσοτήτων δεδομένων που παρουσιάζουν κάποια συσχέτιση.

Οι πίνακες μπορούν να περιλαμβάνουν αλφαριθμητικούς χαρακτήρες (ακέραιους αριθμούς, δεκαδικούς αριθμούς, χαρακτήρες ASCII) καθώς και δείκτες. Η δήλωση και η αρχικοποίηση ενός πίνακα γίνεται με διάφορους τρόπους. Ακολούθως αναλύονται οι συνηθέστεροι τρόποι δήλωσης και αρχικοποίησης πινάκων.

Στην πρώτη περίπτωση παρουσιάζεται μία απλή δήλωση πίνακα ακεραίων. Αρχικά δηλώνεται ο τύπος δεδομένων που περιλαμβάνει ο πίνακας (int,float,char

κλπ.), στη συνέχεια δηλώνεται το όνομα του πίνακα και τελικά δηλώνεται το μέγεθος του πίνακα, δηλαδή το εύρος θέσεων μνήμης που καταλαμβάνει.

```
float theFloats[5];
```

Στη συνέχεια παρουσιάζεται ένας ενδιαφέρον τρόπος δήλωσης κατά τον οποίο δεν δηλώνεται το μέγεθος του πίνακα, αλλά πραγματοποιείται αρχικοποίηση, δηλαδή εισάγονται κάποιες αρχικές τιμές στις θέσεις μνήμης του πίνακα. Ο μεταγλωττιστής (compiler) της C διαβάζει τα δεδομένα εντοπίζει το εύρος τους και στη συνέχεια αποδίδει της απαιτούμενες θέσεις μνήμης στον πίνακα που δηλώθηκε.

```
int taPin[] = {2, 4, 8, 3, 6, 7, 9};
```

Ακολούθως παρουσιάζεται ο τυπικός τρόπος δήλωσης και αρχικοποίησης ενός πίνακα, όπου έχουμε εξ'αρχής δήλωση της ονομασίας, του μήκους και αρχικοποίηση του πίνακα.

```
int oiMet[6] = {2, 4, -8, 3, 2};
```

Τέλος έχουμε την δήλωση ενός πίνακα που απαρτίζεται από αλφαριθμητικούς χαρακτήρες (char) που αρχικοποιείται με μια ολόκληρη λέξη.

```
char toMinima[8] = "Xairete";
```

Για να αποδοθεί μια τιμή σε θέση μνήμης ενός πίνακα χρησιμοποιείται η εξής σύνταξη:

```
oiMet[0] = 1;
```

Για την ανάκληση της τιμής από μια θέση μνήμης ενός πίνακα σε μια μεταβλητή, ακολουθείται η παρακάτω σύνταξη:

```
y = oiMet[0]; //η τιμή της y θα είναι 1.
```

Για την πρόσβαση στα δεδομένα ενός πίνακα πρέπει να ληφθεί υπόψη ότι η αρίθμηση του αριθμού μητρώου του πίνακα ξεκινάει από τον αριθμό 0 , άρα οποιαδήποτε διαδικασία ανάκλησης δεδομένων από έναν πίνακα αποδίδει τα δεδομένα της τελευταίας θέσης μνήμης χρησιμοποιώντας τον αριθμό μεγέθους του πίνακα μειωμένο κατά 1. Για παράδειγμα:

Έστω ένας πίνακας 5 θέσεων μνήμης:

```
int theArray[5]={7,2,1,34,16};
     //theArray [4] περιλαμβάνει το 16
     //theArray [5]περιλαμβάνει τυχαία δεδομένα μιας
     //άλλης θέσης μνήμης.
```

Εάν κατά λάθος γίνει ανάκληση και μεταβολή δεδομένων μιας θέσεις μνήμης που βρίσκεται εκτός του πίνακα μπορεί το πρόγραμμα να παρουσιάσει ανεπιθύμητη συμπεριφορά και αστάθεια.

Για την ανάκληση ή εγγραφή δεδομένων σε έναν πίνακα συχνά χρησιμοποιείται η εντολή επανάληψης for, όπου ο μετρητής επανάληψης της εντολής χρησιμοποιείται ως αριθμός μητρώου του πίνακα. Στο παράδειγμα που ακολουθεί παρουσιάζεται η εκτύπωση των δεδομένων ενός πίνακα σε μία από τις σειριακές θύρες του ελεγκτή.

```
int j;
for (j = 0; j < 7; j = j + 1) {
  Serial.println(taPin[j]);
}
```

Ακολουθίες αλφαβητικών χαρακτήρων

Οι ακολουθίες αλφαβητικών χαρακτήρων (text strings) μπορούν να αναπαρασταθούν με δύο τρόπους, είτε με το αντικείμενο String είτε με πίνακες δεδομένων τύπου char. Το σύνολο των δεδομένων των θέσεων μνήμης ενός πίνακα char απαρτίζει ένα string.

Η ανάγνωση καθώς ακόμη και η εγγραφή ενός string μπορεί να γίνει με μία εντολή επανάληψης όπως η for όπου ο μετρητής της εντολής αποτελεί και τον αριθμό μητρώου του πίνακα, έτσι επιτρέπεται η ανάγνωση ή εγγραφή όλων των θέσεων μνήμης του πίνακα.

string με πίνακα τύπου char

Η δήλωση μιας μεταβλητής τύπου string απαρτιζόμενη από έναν πίνακα δεδομένων τύπου char δημιουργεί μία μεταβλητή πεπερασμένου μήκους η οποία μπορεί να λάβει αλφαριθμητικές τιμές (τύπου δεδομένων char) σε κάθε μία θέση του πίνακα. . Το σύνολο των δεδομένων των θέσεων μνήμης ενός πίνακα char απαρτίζει ένα string.

Η δήλωση και αρχικοποίηση ενός string μπορεί να γίνει με διάφορους τρόπους, το γεγονός αυτό προσδίδει αυξημένη ευελιξία κατά τον προγραμματισμό. Στη συνέχεια αναλύονται οι πιο συνηθισμένες μορφές δήλωσης των string - πινάκων char.

Δίνεται η δυνατότητα δήλωσης ενός string - πίνακα char στον οποίο γίνεται καθορισμός του μεγέθους, μέσω του καθορισμού των θέσεων μνήμης, αλλά δεν γίνεται αρχικοποίηση, δηλαδή δεν αποδίδονται τιμές στις θέσεις μνήμης του string - πίνακα char. Η σύνταξη είναι η εξής:

```
char String1[15];
```

Στον επόμενο τρόπο που παρουσιάζεται πραγματοποιείται δήλωση και αρχικοποίηση ενός string με αρχικοποίηση των επιμέρους θέσεων μνήμης ξεχωριστά. Αυτός ο τρόπος ενδείκνυται για string τα οποία δεν περιλαμβάνουν λέξεις αλλά διάφορες μεμονωμένες αλφαριθμητικές τιμές.

```
char String2[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o'};
```

Στη συνέχεια παρουσιάζεται μια παραλλαγή του παραπάνω τρόπου αρχικοποίησης, η οποία όπως παρατηρείται περιλαμβάνει τον χαρακτήρα `'\0'` στην θέση τελικής τιμής αρχικοποίησης του πίνακα char, αυτό συμβαίνει γιατί κατά κανόνα όλα τα string τερματίζονται με τον χαρακτήρα 0 (μηδέν). Η παραπάνω ιδιότητα επιτρέπει σε διάφορες λειτουργίες (όπως η Serial.print()) να εντοπίζουν που βρίσκεται το σημείο τερματισμού ενός string, ειδάλλως οι λειτουργίες θα συνέχιζαν να διαβάζουν θέσεις μνήμης εκτός του πίνακα char που δεν περιλαμβάνονται στο string.

Λόγω των προαναφερθέντων ιδιοτήτων κρίνεται αναγκαία η δήλωση μίας παραπάνω θέσης μνήμης η οποία όμως δεσμεύεται από το 0 του τερματισμού και δεν μπορεί να χρησιμοποιηθεί από τα δεδομένα του string που αποθηκεύονται. Στην περίπτωση του String2 που αναφέρθηκε, η εισαγωγή του `'\0'` που αγνοήθηκε γίνεται αυτόματα στην τελευταία θέση μνήμης του πίνακα char.

```
char String3[8] = {'a', 'r', 'd', 'u', 'i', 'n', 'o',
'\0'};
```

Ακολούθως παρουσιάζεται ένας ακόμη τρόπος, σε αυτήν την περίπτωση η δήλωση του αριθμού των θέσεων μνήμης αγνοήθηκε σκοπίμως ώστε να συμπληρωθεί αυτόματα από τον compiler με βάση το μήκος του string που δηλώθηκε για αρχικοποίηση. Για τον πίνακα String4 δεσμεύονται αυτόματα οχτώ θέσεις μνήμης, οι πρώτες επτά θέσεις καταλαμβάνονται από την αλληλουχία αλφαριθμητικών χαρακτήρων της λέξης που εισάγεται για την αρχικοποίηση του πίνακα char, ενώ η όγδοη θέση καλύπτεται από τον χαρακτήρα τερματισμού του string `\0`.

Στην παρακάτω μέθοδο δήλωσης η αρχικοποίηση του πίνακα δεν πραγματοποιείται με μεμονωμένες αλφαριθμητικές τιμές αλλά με μία ολόκληρη λέξη η οποία εισάγεται με διπλές αποστρόφους, ενώ οι μεμονωμένοι χαρακτήρες με απλές.

```
char String4[ ] = "arduino";
```

Στην παρακάτω περίπτωση η δήλωση και η αρχικοποίηση του πίνακα γίνεται με τον απαιτούμενο αριθμό θέσεων μνήμης ορισμένο επακριβώς.

```
char String5[8] = "arduino";
```

Επίσης είναι δυνατό κατά τη δήλωση ενός string να αφεθούν κάποιες θέσεις μνήμης κενές για μελλοντική χρήση, μια τέτοια περίπτωση φαίνεται στον παρακάτω κώδικα.

```
char String6[15] = "arduino";
```

Σε αρκετές εφαρμογές όπου απαιτείται χρήση μεγάλων κειμένων (όπως η χρήση οθονών LCD) γίνεται δήλωση πινάκων απαρτιζόμενων από μεμονωμένα

string, δηλαδή γίνεται αξιοποίηση πινάκων δύο διαστάσεων (πίνακες απαρτιζόμενοι από άλλους πίνακες). Κατά την δήλωση του πίνακα αυτού του είδους πρέπει να εισάγεται ένας αστερίσκος μετά την λέξη char ώστε να επισημανθεί ότι ο πίνακας που δημιουργείται είναι πίνακας δεικτών και ότι δεν περιλαμβάνει αλφαριθμητικά δεδομένα.

```
char* bigString[]={"This is string 1", "This is string
2", "This is string 3","This is string 4", "This is
string 5","This is string 6"};
```

5.3 Μετατροπές μεταβλητών - Type casting

Μέσω των μετατροπών επιτυγχάνεται η μεταβολή του τύπου δεδομένων μιας μεταβλητής που περιλαμβάνει μια τιμή. Αυτή η μετατροπή μπορεί να χρησιμεύσει σε κάποιο σημείου του προγράμματος για την αλλαγή του τρόπου χρήσης μια μεταβλητής και προσδίδει ευελιξία κατά τον προγραμματισμό.

char()

Μέσω της εντολής char επιτυγχάνεται μετατροπή μιας μεταβλητής σε τύπο δεδομένων char. Η μετατροπή συντάσσεται ως εξής:

```
char(y)
```

Όπου η y περιλαμβάνει οποιοδήποτε τύπο δεδομένων.

byte()

Μέσω της εντολής byte επιτυγχάνεται μετατροπή μιας μεταβλητής σε τύπο δεδομένων byte. Η μετατροπή συντάσσεται ως εξής:

```
byte(y)
```

Όπου η y περιλαμβάνει οποιοδήποτε τύπο δεδομένων.

int()

Μέσω της εντολής int επιτυγχάνεται μετατροπή μιας μεταβλητής σε τύπο δεδομένων byte. Η μετατροπή συντάσσεται ως εξής:

```
int(y)
```

Όπου η y περιλαμβάνει οποιοδήποτε τύπο δεδομένων.

word()

Μέσω της εντολής word επιτυγχάνεται μετατροπή μιας μεταβλητής σε τύπο δεδομένων word ή η δημιουργία μιας λέξης από δύο byte. Η μετατροπή συντάσσεται ως εξής:

```
word(y)
word(l,r)
```

Όπου η y περιλαμβάνει οποιοδήποτε τύπο δεδομένων, η l περιλαμβάνει το υψηλής σημασίας byte της λέξης ενώ η r το χαμηλής σημασίας byte της λέξης.

long

Μέσω της εντολής long επιτυγχάνεται μετατροπή μιας μεταβλητής σε τύπο δεδομένων long. Η μετατροπή συντάσσεται ως εξής:

```
long(y)
```

Όπου η y περιλαμβάνει οποιοδήποτε τύπο δεδομένων.

float

Μέσω της εντολής float επιτυγχάνεται μετατροπή μιας μεταβλητής σε τύπο δεδομένων float. Η μετατροπή συντάσσεται ως εξής:

```
float(y)
```

Όπου η y περιλαμβάνει οποιοδήποτε τύπο δεδομένων.

5.4 Εμβέλεια μεταβλητών και τα χαρακτηριστικά τους

Εμβέλεια μεταβλητών

Στην γλώσσα προγραμματισμού του μικροελεγκτή Arduino (η οποία όπως έχει αναφερθεί βασίζεται στην C και στην C++) περιλαμβάνεται η ιδιότητα της εμβέλειας μεταβλητών (variable scope). Σε αντίθεση με άλλες γλώσσες όπως η BASIC, όπου οι μεταβλητές είναι κοινές για όλο το πρόγραμμα και υπόκεινται σε χρήση από όλες τις συναρτήσεις του προγράμματος(global), στην C και στην C++ η χρήση των μεταβλητών που δηλώνονται μέσα σε μια συνάρτηση περιορίζεται στα όρια της συνάρτησης αυτής και δεν επεκτείνεται στα υπόλοιπα μέρη του προγράμματος. Οι μεταβλητές αυτές καλούνται μεταβλητές τοπικής χρήσης (local variables). Σε γλώσσες όπως η C και η C++ οι μεταβλητές που δηλώνονται εκτός συναρτήσεων μπορούν να χρησιμοποιηθούν από όλα τα επιμέρους τμήματα του προγράμματος και θεωρούνται μεταβλητές καθολικής χρήσης (global variables).

Συχνά το μέγεθος και η πολυπλοκότητα ενός προγράμματος επιβάλλουν κάθε συνάρτηση να χρησιμοποιεί και να μεταβάλλει τις δικές της μεταβλητές ώστε να αποφεύγονται σφάλματα χρήσης κοινών μεταβλητών. Τέτοιο παράδειγμα

αποτελούν οι επαναληπτικές εντολές όπως η for όπου η μεταβλητή του μετρητή επαναλήψεων αρχικοποιείται στο εσωτερικό της εντολής.

Χαρακτηρισμοί μεταβλητών

Σε πολλές περιπτώσεις χρησιμοποιούνται συγκεκριμένοι χαρακτηρισμοί, για να αποδοθούν ειδικές ιδιότητες σε μεταβλητές. Οι ιδιότητες αυτές χρησιμοποιούνται πριν τον καθορισμό του τύπου δεδομένων της μεταβλητής, για να μεταβάλουν τον τρόπο με τον οποίο αντιμετωπίζονται οι μεταβλητές από τον μεταγλωττιστή (compiler) και το υπόλοιπο του προγράμματος. Οι χαρακτηρισμοί αυτοί αποτελούν οδηγίες προς τον μεταγλωττιστή, ο οποίος μεταγλωττίζει τον κώδικα της C/C++ σε γλώσσα μηχανής την οποία και αντιλαμβάνεται ο επεξεργαστής ATMega του Arduino. Οι ιδιότητες που αποδίδονται αναλύονται στη συνέχεια.

static

Ο χαρακτηρισμός μιας μεταβλητής ως static χρησιμοποιείται, για να δημιουργηθούν μεταβλητές που έχουν όριο χρήσης μόνο εντός της συνάρτησης όπου αρχικοποιούνται. Η επιπλέον ιδιότητα που τους προσδίδεται είναι ότι τα δεδομένα που αποθηκεύονται στην μεταβλητή δεν καταστρέφονται μετά το πέρας της συνάρτησης, αλλά παραμένουν διαθέσιμα αποθηκευμένα στην μεταβλητή. Οι μεταβλητές που λαμβάνουν τον χαρακτηρισμό static μπορούν δημιουργηθούν και να αρχικοποιηθούν μόνο την πρώτη φορά που καλείται μια συνάρτηση.

volatile

Η ιδιότητα που προσδίδεται σε μια μεταβλητή με τον χαρακτηρισμό της ως volatile επιβάλλει στον μεταγλωττιστή να φορτώσει τα δεδομένα της μεταβλητής από μία θέση μνήμης RAM και όχι από κάποιον καταχωρητή που απλά αποθηκεύει προσωρινά τις μεταβλητές, αυτό παρέχει την δυνατότητα η τιμή της μεταβλητής να δύναται να μεταβληθεί από κάποιον εξωγενή παράγοντα πέρα από τον κώδικα. Στην περίπτωση του μικροελεγκτή Arduino παράγοντες που μπορούν να επηρεάσουν μια μεταβλητή που έχει χαρακτηριστεί ως volatile είναι οι διακοπές (interrupts).

const

Ο χαρακτηρισμός μιας μεταβλητής ως const καθιστά την μεταβλητή μη μεταβαλλόμενη με αποτέλεσμα να χρησιμοποιείται μόνο για ανάγνωση, με αυτόν τον τρόπο παρέχεται η δυνατότητα δημιουργίας σταθερών για περαιτέρω χρήση από το πρόγραμμα. Οποιαδήποτε προσπάθεια μεταβολής μιας σταθεράς θα επιφέρει σφάλμα μεταγλώττισης.

5.5 Ιδιαίτερη ιδιότητα μεταβλητών

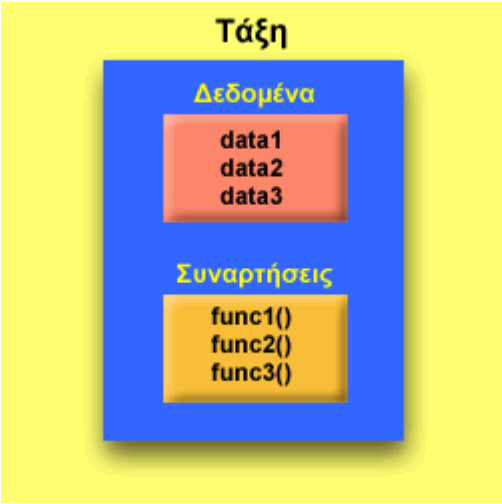
sizeof

Η εντολή sizeof υπολογίζει το μέγεθος σε bytes που καταλαμβάνει μια μεταβλητή ή ένας πίνακας. Η εντολή χρησιμοποιείται κατά κόρον σε περιπτώσεις πινάκων (όπως strings) και επιτρέπει τον υπολογισμό του μεγέθους πινάκων, για να γίνει διαχωρισμός τους.

ΚΕΦΑΛΑΙΟ 6^{ο} Συναρτήσεις

Οι συναρτήσεις αποτελούν ένα από τα θεμελιώδη συστατικά της γλώσσας προγραμματισμού που χρησιμοποιείται στον μικροελεγκτή Arduino. Όταν τα προγράμματα αρχίζουν να γίνονται μεγαλύτερα η διάταξη των εντολών σε απλές λίστες καθίσταται δύσχρηστη, για αυτό το λόγο υιοθετήθηκαν οι συναρτήσεις. Όπως αναφέρθηκε και στο κεφάλαιο 1σχεδόν όλο το πρόγραμμα αποτελείται από δύο καθολικές συναρτήσεις την setup και την loop. Σε αυτό το σημείο πρέπει να σημειωθεί ότι η γλώσσα προγραμματισμού αποτελεί μια αντικειμενοστρεφή γλώσσα όπου το πρόγραμμα αποτελείται από αντικείμενα.

Η κύρια ιδέα ενός προγράμματος αντικειμενοστραφούς γλώσσας είναι ο συνδυασμός σε μία μόνο μονάδα των δεδομένων και των συναρτήσεων που επενεργούν σε αυτά τα δεδομένα. Μία τέτοια μονάδα καλείται αντικείμενο. Κάθε αντικείμενο απαρτίζεται από κάποια επιμέρους υποτμήματα, τις συναρτήσεις μέλη και αποτελούν τον μόνο τρόπο, για να προσπελαστούν τα δεδομένα ενός αντικειμένου. Για την προσπέλαση των δεδομένων γίνεται κλήση της κατάλληλης συνάρτησης μέλους η οποία επιστρέφει τα ζητούμενα δεδομένα. Τονίζεται ότι στην περίπτωση του αντικειμενοστρεφούς προγραμματισμού η προσπέλαση των δεδομένων γίνεται έμμεσα μέσω των συναρτήσεων μελών και όχι άμεσα. Το πρότυπο από το οποίο δημιουργείται ένα αντικείμενο αποτελεί την τάξη, σημειώνεται ότι περισσότερα του ενός αντικείμενα μπορούν να προέλθουν από μία τάξη – πρότυπο. Οι τάξεις (κλάσεις) αποτελούνται από ένα σύνολο δεδομένων και συναρτήσεων οι οποίες επεξεργάζονται τα δεδομένα της τάξης.



Εικόνα 3. Σχηματικό διάγραμμα των περιεχομένων μιας τάξης (Πηγή Τ.Ε.Ι. Ανατολικής Μακεδονίας)

Οι συναρτήσεις όμως μπορούν να υπάρξουν αυτόνομες, ανεξάρτητα από τάξεις (κλάσεις). Στη συνέχεια ακολουθεί ανάλυση των βασικότερων συναρτήσεων που χρησιμοποιούνται στην γλώσσα προγραμματισμού του Arduino.

6.1 Συναρτήσεις Ψηφιακών Εισόδων – Εξόδων (Digital I/O)

pinMode()

Όπως έχει αναφερθεί και σε προηγούμενη παράγραφο με την συνάρτηση pinMode() μπορεί να καθοριστεί η λειτουργία ενός pin ως είσοδος (INPUT), έξοδος (OUTPUT) ή ως INPUT PULLUP, όπου έχουμε ενεργοποίηση των εσωτερικών αντιστάσεων. Η συνάρτηση συντάσσεται ως εξής:

```
pinMode(αριθμός του pin, λειτουργία)
```

digitalWrite()

Με τη χρήση της συνάρτησης digitalWrite αποδίδεται μια συγκεκριμένη λογική τιμή (HIGH ή LOW) σε κάποιο ψηφιακό pin εισόδου ή εξόδου. Για ένα ψηφιακό pin που έχει ρυθμιστεί ως έξοδος η τιμή της τάσης θα γίνει 5 Volt εάν αποδοθεί η τιμή HIGH και 0 Volt εάν αποδοθεί η τιμή LOW .

Στην περίπτωση που ένα ψηφιακό pin χρησιμοποιηθεί ως είσοδος και του αποδοθεί η τιμή HIGH τότε ενεργοποιούνται οι εσωτερικές αντιστάσεις (pull up resistors), ενώ εάν του αποδοθεί η τιμή LOW τότε απενεργοποιούνται οι εσωτερικές αντιστάσεις. Η συνάρτηση συντάσσεται ως εξής:

```
digitalWrite(αριθμός του pin, τιμή)
```

digitalRead()

Δια της χρήσης της συνάρτησης digitalRead() επιτυγχάνεται η ανάγνωση της τιμής μια ψηφιακής εισόδου είτε αυτή είναι HIGH είτε αυτή είναι LOW. Η σύνταξή της γίνεται ως εξής:

```
digitalRead(αριθμός του pin εισόδου)
```

6.2 Συναρτήσεις Αναλογικών Εισόδων – Εξόδων (Analog I/O)

analogReference()

Μέσω της χρήσης της συνάρτησης analogReference() δίνεται η δυνατότητα να ρυθμιστεί η τιμή της τάσης αναφοράς των αναλογικών εισόδων. Δίνονται οι εξής δυνατότητες ρύθμισης της τάσης αναφοράς:

DEFAULT: σε αυτήν την περίπτωση η τάση αναφοράς είναι είτε 5 είτε 3,3 Volt ανάλογα με τον τύπο του μικροελεγκτή.

INTENAL: σε αυτήν την περίπτωση η τάση αναφοράς εξαρτάται από τον τύπο του μικροεπεξεργαστή που διαθέτει ο μικροελεγκτής και είναι 1,1 Volt για τους

μικροελεγκτές που διαθέτουν μικροεπεξεργαστές ATmega168 ATmega328 και 2,56 Volt για μικροελεγκτές που διαθέτουν τον μικροεπεξεργαστή ATmega8 (δεν διατίθεται αυτή η δυνατότητα για τον μικροελεγκτή Arduino Mega).

$INTENAL_1V_1$: με αυτήν την επιλογή τίθεται η εσωτερική τάση αναφοράς στα 1,1 Volt και διατίθεται μόνο για τον μικροελεγκτή Arduino Mega.

$INTENAL_2V_{56}$: με αυτήν την επιλογή τίθεται η εσωτερική τάση αναφοράς στα 2,56 Volt και διατίθεται μόνο για τον μικροελεγκτή Arduino Mega.

EXTERNAL: σε αυτήν την περίπτωση η τάση που εφαρμόζεται στο pin AREF (από 0 έως 5 Volt) χρησιμοποιείται ως τάση αναφοράς.

Κάθε μία από της παραπάνω περιπτώσεις λειτουργίας αποτελεί και μια διαφορετική κατάσταση τάσης αναφοράς.

Σε καμία περίπτωση δεν πρέπει να χρησιμοποιηθεί τάση μικρότερη των 0 Volt ή μεγαλύτερη των 5 Volt διότι θα προκληθεί βλάβη στον μικροελεγκτή. Επίσης η δήλωση της συνάρτησης analogReference() σε κατάσταση EXTERNAL πρέπει να γίνεται πριν από την χρήση της συνάρτησης analogRead() γιατί σε αντίθετη περίπτωση γίνεται βραχυκύκλωμα της ενεργού εσωτερικής τάσης αναφοράς και της τάσης που εφαρμόζεται στο pin αναφοράς.

Εναλλακτικά η εφαρμογή της τάσης αναφοράς στο pin AREF μπορεί να γίνει μέσω μιας αντίστασης 5 kΩ επιτρέποντας την εναλλαγή των τάσεων αναφοράς ανάμεσα σε εσωτερική και εξωτερική. Πρέπει να σημειωθεί ότι στο pin AREF βρίσκεται μια αντίσταση 32kΩ συνδεδεμένη κατά τέτοιο τρόπο που με την εξωτερική αντίσταση των 5kΩ σχηματίζουν ένα διαιρέτη τάσης που έχει σαν αποτέλεσμα την μείωση της τάσης αναφοράς. Η συνάρτηση συντάσσεται ως εξής:

```
analogReference( τύπος της κατάστασης τάσης αναφοράς )
```

analogRead()

Μέσω της συνάρτησης analogRead() πραγματοποιείται ανάγνωση ενός pin που έχει ρυθμιστεί ως αναλογική είσοδος. Ο μικροελεγκτής Arduino διαθέτει έναν 6-κάναλο μετατροπέα από αναλογικό σε ψηφιακό σήμα, εύρους 10 bit. Με την χρήση αυτού του μετατροπέα, παρέχεται η δυνατότητα μετατροπής του αναλογικού σήματος εύρους από 0 έως 5 Volt σε ψηφιακό σήμα που λαμβάνει ακέραιες τιμές από 0 έως 1023 μονάδες. Η παρεχόμενη ακρίβεια μέτρησης προκύπτει αν διαιρέσουμε τα 5 Volt δια του 2^{10}=1024 (μονάδες),εξάγοντας 0,0049 Volt/μονάδα (4,9 mV/unit). Εύκολα συνεπάγεται ότι με τη χρήση της διάταξης διαιρέτη τάσης που αναφέρθηκε στο προηγούμενο εδάφιο και της συνάρτησης analogReference() επιτυγχάνεται αυξομείωση της ακρίβειας μέτρησης. Απαιτούνται 100 μsec (0,0001 sec) για μία μέτρηση αναλογικής εισόδου, έτσι συμπεραίνουμε ότι σε 1 sec πραγματοποιούνται 10000 μετρήσεις (10 kHertz).

Εάν ένα αναλογικό pin έχει ρυθμιστεί ως αναλογική είσοδος δεν είναι συνδεδεμένο σε κάποια πηγή σήματος ή έστω σε γείωση, κατά την εκτέλεση της συνάρτησης analogRead() θα παρουσιάσει μια κυμαινόμενη τάση (θόρυβος) εξαρτώμενη από διάφορους παράγοντες όπως τα σήματα των γειτονικών αναλογικών pin (φαινόμενο crosstalk) ή ακόμα και από την απόσταση ενός χεριού προς την πλακέτα του μικροελεγκτή.

Η συνάρτηση συντάσσεται με τον εξής τρόπο:

```
analogRead( ο αριθμός του pin στο οποίο θα γίνει ανάγνωση σήματος)
```

analogWrite()

Μέσω της συνάρτησης analogWrite() εφαρμόζεται ένα αναλογικό σήμα ως έξοδος σε ένα αναλογικό pin. Η υλοποίηση του σήματος γίνεται μέσω της μεθόδου Pulse Wide Modulation (PWM). Μέσω αυτού του αναλογικού σήματος επιτυγχάνεται έλεγχος εναύσεως ενός LED καθώς και έλεγχος της ταχύτητας περιστροφής ενός κινητήρα. Μετά την κλήση της συνάρτησης analogWrite() στο pin εξόδου εφαρμόζεται μια κυματομορφή τετραγωνικών παλμών με το προκαθορισμένο εύρος έως ότου διακοπεί από την επανάκληση της analogWrite() ή την κλήση της analogRead() και analogWrite() στο ίδιο pin. Η συχνότητα του σήματος κυμαίνεται στα 490 Hertz περίπου.

Πρέπει να σημειωθεί ότι δεν απαιτείται η ρύθμιση ενός pin ως εξόδου μέσω της pinMode() πριν την κλήση της συνάρτησης analogWrite(). Στου μικροελεγκτές Arduino με μικροεπεξεργαστή ATmega168 και ATmega328 η συνάρτηση μπορεί να εφαρμοστεί στα pin 3, 5, 6, 9, 10 και 11, στο Arduino Mega στα pin 2 έως 13 και στους παλαιότερους μικροελεγκτές με ATmega8 στα pin 9, 10 και 11. Η συνάρτηση συντάσσεται με τον εξής τρόπο:

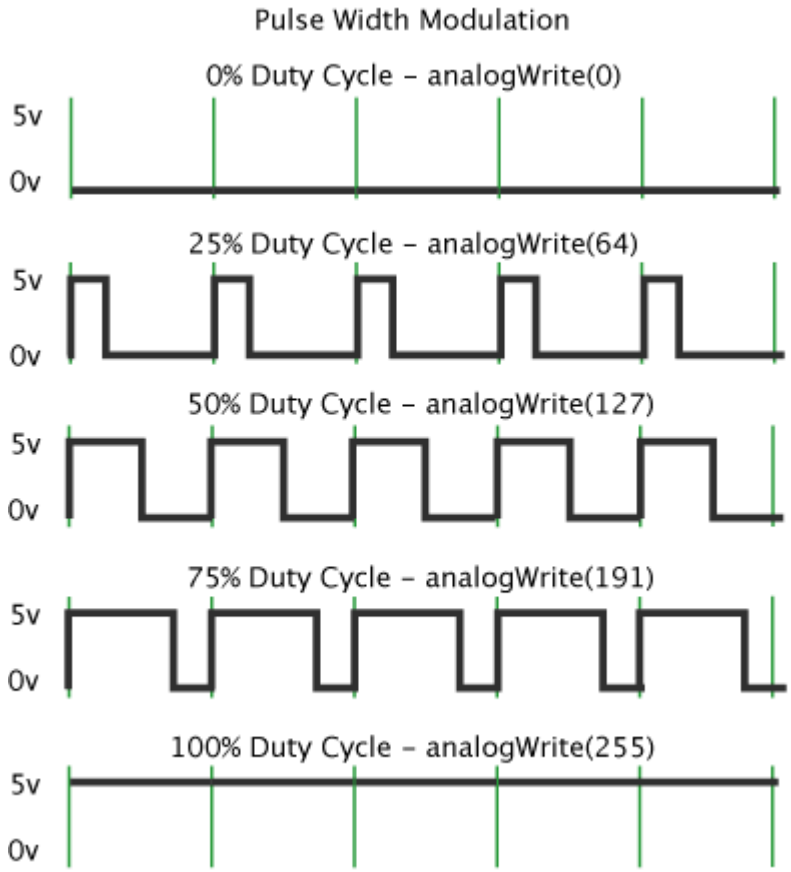
```
analogWrite ( pin, τιμή ),
```

όπου στο πεδίο pin εγγράφεται ο αριθμός του pin που θα χρησιμοποιηθεί ως αναλογική έξοδος και στο πεδίο τιμή εγγράφεται η χρονική διάρκεια του παλμού.

Pulse Width Modulation (PWM) – Διαμόρφωση Πλάτους Παλμού

Η διαμόρφωση πλάτους παλμού (PWM) αποτελεί μια από τις πιο συνηθισμένες μεθόδους ελέγχου της ροής ενέργειας σε κάποιο ηλεκτρικό φορτίο. Η μέθοδος PWM βασίζεται στην αυξομείωση της διάρκειας ενός θετικού τετραγωνικού παλμού κατά την διάρκεια μιας περιόδου. Ο τετραγωνικός παλμός μπορεί να καταλαμβάνει διάφορα ποσοστά της χρονικής διάρκειας της περιόδου με αποτέλεσμα διαφορετικές ποσότητες ενέργειας να τροφοδοτούνται στο φορτίο σε κάθε περίπτωση. Αναλυτικότερα στον μικροελεγκτή Arduino το αναλογικό σήμα ουσιαστικά εξάγεται από την επεξεργασία ενός ψηφιακού σήματος, η παραπάνω λειτουργία μπορεί να παρομοιαστεί με την « έναυση και σβέση » ενός διακόπτη που συνδέει μια πηγή συνεχούς τάσης στάθμης 5 Volt με κάποιο φορτίο, σε συχνότητα 490 Hertz. Η χρονική διάρκεια κατά την οποία ο διακόπτης παραμένει κλειστός και διατηρεί το φορτίο συνδεδεμένο προς την πηγή αποτελεί το χρονικό εύρος του παλμού. Αυξομειώνοντας το χρονικό εύρος του παλμού επιτυγχάνεται η διαμόρφωση του αναλογικού σήματος εξόδου.

Στο πεδίο τιμή της εντολής analogWrite(pin,τιμή) προσδιορίζεται η χρονική διάρκεια του θετικού τετραγωνικού παλμού με ακέραιες τιμές από το 0 έως το 255. Στα παρακάτω διαγράμματα παρουσιάζονται οι κυματομορφές των σημάτων εξόδου για διάφορα χρονικά εύρη του θετικού τετραγωνικού παλμού.



Εικόνα 3. (πηγή Arduino.cc)

6.3 Advanced I/O

tone()

Η κλήση της συνάρτησης tone() δημιουργεί μια τετραγωνική κυματομορφή καθοριζόμενης συχνότητας και με εύρος τετραγωνικού παλμού το 50 τοις εκατό της περιόδου του κύματος σε κάποιο pin, το σήμα αυτό χρησιμοποιείται για την παραγωγή ηχητικών σημάτων. Η χρονική διάρκεια αναπαραγωγής του σήματος σε κάποιο pin μπορεί να καθοριστεί μέσω της συνάρτησης ή να διακοπεί με την κλήση της συνάρτησης noTone(). Στο pin που έχει επιλεχθεί μπορεί να συνδεθεί ένα μικρό ηχείο ή ένα buzzer. Πρέπει να σημειωθεί ότι μόνο μια συνάρτηση παραγωγής ήχου μπορεί να εκτελείται τη φορά, εάν κληθεί μια συνάρτηση tone() την ώρα που αναπαράγεται το σήμα κάποιας άλλης συνάρτησης tone() τότε απλά η δεύτερη συνάρτηση δεν θα εκτελεσθεί. Πρέπει να τονισθεί ότι η χρήση της tone() σε συνδυασμό με την αναλογική έξοδο PWM στα pin 3 και 11 μπορεί να επιφέρει παρεμβολές στο ηχητικό σήμα. Η συνάρτηση συντάσσεται ως εξής:

```
tone(το pin εξόδου, η επιλεγόμενη συχνότητα)
tone(το pin εξόδου, η επιλεγόμενη συχνότητα, η
επιλεγόμενη διάρκεια)
```

noTone()

Μέσω της συνάρτησης noTone() γίνεται η παύση των ηχητικών σημάτων που εφαρμόζονται σε κάποιο pin εξόδου. Σε περίπτωση που απαιτείται η αναπαραγωγή διάφορων σε σημάτων σε διαφορετικά pin πρέπει να γίνει πρώτα παύση ενός σήματος μέσω της noTone() και στη συνέχεια έναρξη του επόμενου ηχητικού σήματος μέσω της tone(). Η noTone() συντάσσεται ως εξής:

```
noTone( το νούμερο του pin όπου θα γίνει παύση
αναπαραγωγής )
```

shiftOut()

Μέσω της συνάρτησης shiftOut επιτυγχάνεται η μεταφορά των δεδομένων ενός byte σε κάποια έξοδο με ρυθμό ενός bit τη φορά. Η εκκίνηση της αποστολής των bit προς την έξοδο μπορεί να ξεκινήσει είτε από το πιο σημαντικό bit είτε από το λιγότερο σημαντικό bit. Κάθε bit εγγράφεται με την σειρά του στο pin εξόδου, μόλις ολοκληρωθεί η αποστολή ενός bit στην έξοδο ένας κρουστικός παλμός εξάγεται στο pin ρολογιού (clock pin), για να δηλώσει ότι το bit είναι διαθέσιμο. Η εντολή συντάσσεται ως εξής:

```
shiftOut(dataPin, clockPin, bitOrder, τιμή)
```

dataPin: το pin όπου εξάγεται η ακολουθία των bit του Byte,
clockPin: το pin ρολογιού,
bitOrder: η φορά εξόδου των bit. Περιλαμβάνει δύο επιλογές την **MSBFIRST** ή την **LSBFIRST**. Η **MSBFIRST ξεκινά την εξαγωγή από το πιο σημαντικό bit και** η **LSBFIRST ξεκινά την εξαγωγή από το λιγότερο σημαντικό bit,**
τιμή: **την προς έξοδο τιμή του B**yte,

Η συγκεκριμένη συνάρτηση αποτελεί μια υλοποίηση εξόδου δεδομένων μέσω λογισμικού. Υπάρχει η δυνατότητα υλοποίησης εξόδου δεδομένων μέσω του υλικού (hardware) σε υψηλότερη ταχύτητα.

shiftIn()

Η συνάρτηση shiftIn() χρησιμοποιείται με σκοπό την εισαγωγή δεδομένων με ρυθμό ενός bit τη φορά. Η εισαγωγή των δεδομένων στον μικροελεγκτή γίνεται μέσω ενός pin, προρυθμισμένου ως είσοδο και μπορεί να ξεκινήσει είτε από το πιο σημαντικό bit, είτε από το λιγότερο σημαντικό bit. Κατά την μετάδοση ενός bit και την ανάγνωση του bit στο pin εισόδου, το bit του ρολογιού μεταβαίνει σε κατάσταση high και αμέσως μετά μεταβαίνει σε κατάσταση low. Η εντολή συντάσσεται ως εξής:

```
Εισερχόμενο Byte= shiftIn(dataPin, clockPin, bitOrder)
```

dataPin: το pin από όπου εισάγεται η ακολουθία των bit του Byte,
clockPin: το pin ρολογιού,
bitOrder: η φορά εξόδου των bit. Περιλαμβάνει δύο επιλογές την **MSBFIRST** ή την **LSBFIRST**. Η **MSBFIRST ξεκινά την εξαγωγή από το πιο σημαντικό bit και** η **LSBFIRST ξεκινά την εξαγωγή από το λιγότερο σημαντικό bit,**

pulseIn()

Η συνάρτηση pulseIn() χρησιμοποιείται για να μετρηθεί η χρονική διάρκεια ενός παλμού. Όταν κληθεί η συνάρτηση pulseIn() αρχίζει να παρακολουθεί το pin και όταν μεταβεί σε κατάσταση HIGH τότε αρχίζει να προσμετρείται η χρονική διάρκεια έως ότου μεταβεί σε κατάσταση LOW. Η συνάρτηση επιστρέφει το μήκος της χρονικής διάρκειας σε μsec ενώ επιστρέφει 0 εάν δεν ανιχνευθεί κάποιος παλμός μέσα σε συγκεκριμένο χρονικό διάστημα.

Η συνάρτηση επιστρέφει αξιόπιστα αποτελέσματα για χρονικά διαστήματα από 10 μsec έως 3 λεπτά και συντάσσεται ως εξής:

```
pulseIn(pin, τιμή)
pulseIn(pin, value, timeout)
```

pin: το pin όπου θα γίνει η ανάγνωση παλμού,
τιμή: ο τύπος του παλμού που θα αναγνωσθεί (HIGH ή LOW),
timeout: (προαιρετική δυνατότητα) ο χρόνος αναμονής (σε μsec) μέχρι να ξεκινήσει ο παλμός (εξ' ορισμού 1 sec).

6.4 Χρονικές συναρτήσεις

millis()

Η συνάρτηση millis() επιστρέφει τον χρόνο (σε msec) κατά τον οποίο ο μικροελεγκτής εκτελεί ένα συγκεκριμένο πρόγραμμα. Ο μετρητής επανεκκινεί την μέτρηση μετά από περίπου 50 ημέρες.

micros()

Η συνάρτηση micros() επιστρέφει τον χρόνο (σε μsec) κατά τον οποίο ο μικροελεγκτής εκτελεί ένα συγκεκριμένο πρόγραμμα. Ο μετρητής επανεκκινεί την μέτρηση μετά από περίπου 70 ημέρες.

delay()

Μέσω της συνάρτησης delay() εισάγεται μια χρονική παύση κατά την εκτέλεση του προγράμματος για κάποιο προκαθορισμένο διάστημα (σε msec). Η συνάρτηση ακολουθεί τον εξής τρόπο σύνταξης:

```
delay (msec)
```

delayMicroseconds()

Κατά την κλήση της συνάρτησης delayMicroseconds() πραγματοποιείται παύση της εκτέλεσης του προγράμματος για κάποιο συγκεκριμένο χρονικό διάστημα και καθορίζεται σε μsec. Η συγκεκριμένη συνάρτηση προσφέρει ακρίβεια χρονικής μέτρησης μέχρι τα 16383 μsec, λόγω περιορισμών της κατασκευής του μικροελεγκτή. Για την εισαγωγή μεγαλύτερων χρονικών παύσεων πρέπει να χρησιμοποιείται η συνάρτηση delay(). Η σύνταξη της είναι η εξής:

```
delayMicroseconds(μsec)
```

6.5 Μαθηματικές συναρτήσεις

max(x, y)

Η συνάρτηση max(x, y) επιστρέφει τον μεγαλύτερο ανάμεσα στους δύο αριθμούς που δέχεται ως ορίσματα, πρέπει να σημειωθεί ότι τα δύο ορίσματα μπορούν να ανήκουν σε διαφορετικούς τύπους δεδομένων, να αποτελούν μεταβλητές ή να αποτελούν αριθμητικές τιμές. Σε πολλές περιπτώσεις η χρήση της max() γίνεται για να οριστεί το κάτω όριο του εύρους τιμών μιας μεταβλητής.

min(x, y)

Η συνάρτηση min(x, y) επιστρέφει τον μικρότερο αριθμό ανάμεσα στα δύο ορίσματα και σε αυτήν την περίπτωση τα ορίσματα μπορούν να ανήκουν σε διαφορετικούς τύπους δεδομένων και να αναπαρίστανται είτε με αριθμητικές τιμές είτε με συναρτήσεις. Η συνάρτηση μπορεί να χρησιμοποιηθεί για να οριστεί το άνω όριο του εύρους τιμών μιας μεταβλητής.

abs()

Η συνάρτηση abs()υπολογίζει και επιστρέφει την απόλυτη τιμή ενός αριθμού. Λόγω του τρόπου υλοποίησης της συνάρτησης πρέπει να αποφεύγεται η χρήση άλλων συναρτήσεων εντός των παρενθέσεων.

constrain()

Η συνάρτηση constrain() χρησιμοποιείται, για να διατηρηθεί η τιμή μιας μεταβλητής ανάμεσα σε ένα εύρος τιμών. Η σύνταξή της έχει ως εξής:

```
constrain(x, a, b)
```

όπου x: η προς περιορισμό μεταβλητή,
a: το κάτω όριο

b: το άνω όριο

map()

Η συνάρτηση map(value, fromLow, fromHigh, toLow, toHigh) χρησιμοποιείται, για να μεταβληθεί από ένα εύρος τιμών σε ένα άλλο εύρος τιμών. Η συνάρτηση δεν μπορεί να περιορίσει τη μεταβλητή στο νέο εύρος τιμών, για τον περιορισμό της συνάρτησης πρέπει να χρησιμοποιηθεί η συνάρτηση constrain(). Η συνάρτηση συντάσσεται ως εξής:

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

value: η συνάρτηση,
fromLow: το κάτω άκρο του υφιστάμενου εύρους τιμών,
fromHigh: το άνω άκρο του υφιστάμενου εύρους τιμών,
toLow: το κάτω άκρο του μελλοντικού εύρους τιμών,
toHigh: το άνω άκρο του μελλοντικού εύρους τιμών,

pow()

Η συνάρτηση pow χρησιμοποιείται, για να υψωθεί σε κάποια δύναμη μία μεταβλητή ή ένας αριθμός. Η pow() διαθέτει την ικανότητα να εκτεθεί η βάση σε δύναμη που περιλαμβάνει δεκαδικό αριθμό. Στη συνέχεια ακολουθεί ο τρόπος σύνταξης.

```
pow(βάση, εκθέτης)
```

sqrt()

Η συνάρτηση sqrt() υπολογίζει την τετραγωνική ρίζα του ορίσματος που βρίσκεται ανάμεσα στις παρενθέσεις. Πρέπει να τονιστεί ότι το όρισμα μπορεί να είναι οποιοσδήποτε τύπος δεδομένων. Η συνάρτηση επιστρέφει τετραγωνική ρίζα ενός αριθμού σε τύπο δεδομένων διπλής ακρίβειας (double). Η σύνταξη της συνάρτησης είναι η ακόλουθη:

```
sqrt(x)
```

6.6 Τριγωνομετρικές συναρτήσεις

sin()

Η συνάρτηση sin() επιστρέφει το ημίτονο μια γωνίας, το εύρος τιμών των αποτελεσμάτων αποτελεί το διάστημα [-1,1]. Το αποτέλεσμα της συνάρτησης είναι τύπου δεδομένων double (διπλής ακρίβειας) ενώ η γωνία, που αποτελεί το όρισμα είναι τύπου δεδομένων float και μετράται σε ακτίνια (rad).

cos()

Η συνάρτηση cos() υπολογίζει το συνημίτονο μιας γωνίας (εκφρασμένο σε ακτίνια- τύπος δεδομένων float). Το εύρος των τιμών εξόδου αποτελεί το διάστημα [-1,1], αναφέρεται ότι η έξοδος είναι τύπου δεδομένων double.

tan()

Η συνάρτηση tan() επιστρέφει την τιμή της εφαπτομένης μιας γωνίας (μετρημένης σε ακτίνια) και έχει εύρος τιμών το αριθμητικό διάστημα (-∞, +∞). Το όρισμα της συνάρτησης είναι τύπου δεδομένων float και η τιμή εφαπτομένης είναι τύπου δεδομένων double.

6.7 Συναρτήσεις τυχαίων αριθμών

randomSeed(seed)

Η συνάρτηση randomSeed() εκκινεί την γεννήτρια τυχαίων αριθμών σε κάποιο αυθαίρετο σημείο της στοχαστικής της ακολουθίας. Η στοχαστική ακολουθία αν και μακρά είναι πάντα η ίδια. Εάν απαιτείται οι τυχαίοι αριθμοί να διαφέρουν ανάμεσα σε δύο κλήσεις της συνάρτησης τότε μπορεί να εκκινηθεί με όρισμα μία τυχαία είσοδο όπως ένα μη συνδεδεμένο pin. Αντιστρόφως, η συνάρτηση μπορεί να χρησιμοποιηθεί ώστε να δημιουργηθούν δύο ακολουθίες αριθμών που θα είναι ίδιες, αυτό γίνεται εφικτό με την χρήση του ίδιου αριθμού ως όρισμα. Η συνάρτηση συντάσσεται ως εξής:

randomSeed(όρισμα συνάρτησης)

όπου όρισμα μπορεί να αποτελεί η ανάγνωση ενός μη συνδεδεμένου bit (που όπως έχει αναφερθεί παρουσιάζει θόρυβο) η μία αριθμητική τιμή.

random()

Η random αποτελεί μια συνάρτηση που επιστρέφει έναν τυχαίο αριθμό που προκύπτει από την γεννήτρια αριθμών και βασίζεται σε μια ακολουθία. Η συνάρτηση συντάσσεται με τους ακόλουθους τρόπους.

```
random(max)
random(min, max)
```

όπου min η ελάχιστη τιμή που μπορεί να επιστραφεί σαν αποτέλεσμα από την συνάρτηση.
όπου max-1 η μέγιστη τιμή που μπορεί να επιστραφεί σαν αποτέλεσμα από την συνάρτηση.

6.8 Συναρτήσεις που αφορούν bits και Bytes

lowByte()

Μέσω της συνάρτησης lowByte() εξάγεται το χαμηλότερης σημασίας Byte μιας μεταβλητής (π.χ. τύπου δεδομένων word). Η εντολή έχει την εξής σύνταξη:

```
lowByte(y)
```

όπου y μια μεταβλητή οποιουδήποτε τύπου δεδομένων.

highByte()

Με την συνάρτηση highByte() εξάγεται το υψηλότερης σημασίας Byte μιας μεταβλητής (ή το δεύτερο λιγότερο σημαντικό Byte μιας μεταβλητής). Η συνάρτηση ακολουθεί της εξής σύνταξη:

```
highByte(y)
```

όπου y μια μεταβλητή οποιουδήποτε τύπου δεδομένων μήκους δύο Byte.

bitRead()

Με την χρήση της ακόλουθης συνάρτησης επιστρέφεται ένα συγκεκριμένο bit ενός αριθμού. Παρακάτω παρουσιάζεται ο τρόπος σύνταξης της συνάρτησης:

```
bitRead(y, l)
```

όπου y ο αριθμός που θα αναγνωσθεί και l το bit του οποίου η τιμή θα εξαχθεί (αρχίζοντας από το λιγότερο σημαντικό bit).

bitWrite()

Μέσω της συνάρτησης bitWrite() εγγράφεται μια συγκεκριμένη τιμή σε ένα συγκεκριμένο bit ενός αριθμού. Η σύνταξη της συνάρτησης παρουσιάζεται παρακάτω:

```
bitWrite(y, l, b)
```

όπου y ο αριθμός που θα μεταβληθεί,

όπου l η θέση του bit όπου θα γίνει η εγγραφή,
όπου b η τιμή που θα εγγραφεί στο bit (0 ή 1),

bitSet()

Μέσω της συνάρτησης bitSet() λαμβάνει την τιμή ένα (1) ένα συγκεκριμένα bit μιας αριθμητικής μεταβλητής. Η εντολή συντάσσεται με τον εξής τρόπο:

```
bitSet(y, l)
```

όπου y η αριθμητική μεταβλητή και l η θέση του bit ξεκινώντας τη μέτρηση από το λιγότερο σημαντικό bit.

bitClear()

Μέσω της συνάρτησης bitClear() τίθεται στην τιμή 0 ένα συγκεκριμένο bit μιας αριθμητικής μεταβλητής. Ακολουθεί ο τρόπος σύνταξης:

```
bitClear(y, l) ,
```

όπου y η αριθμητική μεταβλητή και l η θέση του bit που θα λάβει την τιμή 0 ξεκινώντας την αρίθμηση από το λιγότερο σημαντικό bit.

bit()

Μέσω της συνάρτησης bit() υπολογίζεται η αριθμητική τιμή ενός συγκεκριμένου bit. (π.χ. bit(0)=1, bit(1)=2, bit(2)=4, bit(4)=8 κ.ο.κ.) Η συνάρτηση συντάσσεται με τον εξής τρόπο:

```
bit(l) ,
```

όπου l η θέση του bit του οποίου η τιμή θα υπολογιστεί.

6.9 Interrupts (Διακοπές)

Στον προγραμματισμό η διακοπή (Interrupt) αποτελεί ένα σήμα που εκπέμπεται είτε από το υλικό, είτε από το λογισμικό προς τον επεξεργαστή και απαιτεί την άμεση προτεραιότητά του για την λήψη κάποιας δράσης. Η διακοπή (Interrupt) ειδοποιεί τον επεξεργαστή για μια κατάσταση υψηλής προτεραιότητας που απαιτεί την άμεση διακοπή του εκτελούμενου κώδικα – του νήματος. Ο επεξεργαστής ανταποκρίνεται στην αίτηση διακοπής αναστέλλοντας την διεργασία που εκτελούσε, αποθηκεύοντας την κατάστασή του και εκτελώντας ένα πρόγραμμα διαχείρισης διακοπής (ISR – Interrupt Service Routine). Αφού ολοκληρωθεί η εκτέλεση του προγράμματος διαχείρισης διακοπής ο επεξεργαστής συνεχίζει την εκτέλεση του προηγούμενου κώδικα - νήματος. Υφίστανται δύο είδη διακοπών, οι εξωτερικές που προέρχονται από το υλικό (hardware interrupts) και οι εσωτερικές που προέρχονται από το λογισμικό (software interrupts).

Η χρήση των διακοπών (interrupts) βοηθάει στο να εξοικονομούνται ζωτικοί πόροι του επεξεργαστή και του υλικού, οι οποίοι θα διατίθετο σε βρόχους ελέγχου (polling loops) εξωτερικών συμβάντων. Οι βρόχοι ελέγχου είτε λαμβάνουν μορφή ηλεκτρονικών διατάξεων υλικού (hardware) είτε βρίσκονται ενσωματωμένοι στη μνήμη του προγράμματος.

External Interrupts (Εξωτερικές Διακοπές)

attachInterrupt()

Η εντολή attachInterrupt() καθορίζει ποια εντολή εκτελείται όταν μια εξωτερική διακοπή λάβει χώρα. Οι περισσότεροι μικροελεγκτές Arduino διαθέτουν δύο εξωτερικές διακοπές, την διακοπή 0 (στο ψηφιακό pin 2) και την διακοπή 1 (στο ψηφιακό pin 3). Ο τρόπος με τον οποίο συντάσσεται η εντολή παρουσιάζει ιδιαίτερο ενδιαφέρον και παρουσιάζεται ακολούθως.

```
attachInterrupt(interrupt, function, mode)
attachInterrupt(pin, function, mode)
```

Στο πεδίο interrupt ορίζεται ο αριθμός της διακοπής (0 ή 1), ενώ στο πεδίο pin ορίζεται αριθμός του ψηφιακού pin της διακοπής.
Στο πεδίο function ορίζεται η συνάρτηση που καλείται όταν πραγματοποιείται διακοπή. Η συνάρτηση που καλείται ονομάζεται και Interrupt Service Routine (ISR) και δεν λαμβάνει παραμέτρους ούτε επιστρέφει δεδομένα μετά την ολοκλήρωση της εκτέλεσής της.
Τέλος στο πεδίο mode καθορίζεται η στιγμή ενεργοποίησης της διακοπής και διατίθενται τέσσερεις δυνατές επιλογές:
α) **LOW:** Η διακοπή ενεργοποιείται όταν το pin (της διακοπής) βρίσκεται σε κατάσταση LOW.
β) **CHANGE:** Η διακοπή ενεργοποιείται όταν αλλάξει η κατάσταση του pin.
γ) **RISING:** Η διακοπή ενεργοποιείται όταν το pin μεταβεί από κατάσταση LOW σε κατάσταση HIGH.

δ) **FALLING:** Η διακοπή ενεργοποιείται όταν η κατάσταση του pin αλλάξει από HIGH σε LOW.

Τονίζεται ότι κατά την εκτέλεση της διακοπής δεν λειτουργεί η συνάρτηση delay() καθώς και η χρονική συνάρτηση millis(), επίσης πρέπει να λαμβάνεται υπόψη ότι δεδομένα που λαμβάνονται σειριακά μπορεί να απολεσθούν. Επίσης πρέπει όλες οι μεταβλητές που καλούνται κατά την διάρκεια της διακοπής να χαρακτηρίζονται ως volatile.

detachInterrupt()

Η εντολή detachInterrupt() «αποσυνδέει» οποιαδήποτε εντολή έχει συνδεθεί σε μια συγκεκριμένη διακοπή και την απενεργοποιεί. Η σύνταξη που ακολουθείτε για αυτήν την εντολή είναι η εξής:

```
detachInterrupt(interrupt)
```

όπου στο πεδίο interrupt ορίζεται η διακοπή που θα απενεργοποιηθεί.

Interrupts()

Η συνάρτηση interrupts ενεργοποιεί τις διακοπές του μικροελεγκτή σε περίπτωση που έχουν απενεργοποιηθεί. Όπως αναφέρθηκε προηγουμένως οι διακοπές επιτρέπουν την παρακολούθηση λειτουργιών στο παρασκήνιο, χωρίς να δεσμεύεται ιδιαίτερα μεγάλη ποσότητα πόρων του συστήματος. Σε ελάχιστες περιπτώσεις οι διακοπές ενδέχεται να επηρεάζουν τον χρονισμό του μικροελεγκτή γεγονός που ενδέχεται να μην είναι αποδεκτό για την εκτέλεση συγκεκριμένων τμημάτων κώδικα σε αυτή την περίπτωση επιλέγεται η απενεργοποίηση των διακοπών.

Η συνάρτηση Interrupts δεν διαθέτει κάποιο όρισμα και απλώς καλείται με τον ακόλουθο τρόπο.

```
Interrupts()
```

noInterrupts()

Η συνάρτηση noInterrupts απενεργοποιεί την λειτουργία των διακοπών του μικροελεγκτή σε περίπτωση που εκτελείται κώδικας που το απαιτεί. Η συνάρτηση δεν διαθέτει κάποιο συγκεκριμένο όρισμα.

6.10 Επικοινωνίες

Για την επικοινωνία του μικροελεγκτή με άλλες συσκευές όπως υπολογιστές, αισθητήρες, άλλους μικροελεγκτές καθώς και συστήματα επέκτασης δυνατοτήτων χρησιμοποιούνται ειδικές κλάσεις (τάξεις) συναρτήσεων. Οι τάξεις αυτές είναι οι Serial, Stream και η Wire. Στη συνέχεια αναλύονται εκτενέστερα η λειτουργία των κλάσεων καθώς και των ειδικών συναρτήσεων που της απαρτίζουν.

Stream

Η stream αποτελεί την βασική κλάση που σχετίζεται με την ροή αλφαριθμητικών χαρακτήρων και δυαδικών αριθμών. Η κλάση δεν καλείται απευθείας αλλά επικαλείται κάθε φορά που πραγματοποιείται μια κλήση συνάρτησης που βασίζεται σε αυτήν.

Η Stream καθορίζει όλες τις λειτουργίες ανάγνωσης του μικροελεγκτή Arduino. Κάθε φορά που γίνεται χρήση λειτουργιών που περιλαμβάνουν την read() ή κάποιας παρόμοιας εντολής, μπορεί να υποτεθεί με ασφάλεια ότι πραγματοποιείται κλήση της κλάσης (τάξης) Stream. Για συναρτήσεις όπως η print() η stream κληρονομεί από την κλάση (τάξη) Print.

Οι βιβλιοθήκες συναρτήσεων (κλάσεις) που βασίζονται στην Stream είναι αρκετές και περιλαμβάνουν τις εξής:

- Serial
- Wire
- Ethernet Client
- Ethernet Server
- SD

Στης συνέχεια θα αναλυθούν οι συναρτήσεις που περιλαμβάνονται στην κλάση Stream.

available()

Η συνάρτηση available() αναφέρει την ποσότητα δεδομένων (αριθμός σε Bytes) που έχει παραληφθεί και είναι διαθέσιμα για ανάγνωση. Η συνάρτηση available περιλαμβάνεται στην κλάση Stream και κατά συνέπεια μπορεί να χρησιμοποιηθεί από όλες τις κλάσεις (τάξεις) που κληρονομούν από αυτήν (Serial, Wire κ.α.)

Η σύνταξη της συνάρτησης είναι η εξής:

```
stream.available()
```

read

Η συνάρτηση read διαβάζει τους χαρακτήρες μιας εισερχόμενης ροής δεδομένων και τους αποθηκεύει στον buffer. Η συνάρτηση συντάσσεται ως εξής:

```
stream.read()
```

flush

Η flush διαγράφει όλους τους εξερχόμενους χαρακτήρες που βρίσκονται στον buffer μετά την αποστολή τους. Η μορφή της συνάρτησης είναι η ακόλουθη:

```
stream.flush()
```

streamTimeout()

Μέσω της συνάρτησης setTimeout() ορίζεται η μέγιστη χρονική διάρκεια κατά την οποία αναμένονται χαρακτήρες από μία ροή δεδομένων. Μετά την παρέλευση του οριζόμενου χρονικού διαστήματος ολοκληρώνονται η συναρτήσεις ανάγνωσης stream. Η προκαθορισμένη χρονική διάρκεια ανάγνωσης είναι 1000msec. Η συνάρτηση συντάσσεται με τον ακόλουθο τρόπο:

stream.setTimeout(time) ,

όπου time η επιθυμητή διάρκεια ανάγνωσης σε msec.

find

Η find διαβάζει τα δεδομένα μιας ροής χαρακτήρων, έως ότου συναντήσει μια προκαθορισμένη ακολουθία (string) και επιστρέφει true Εάν δεν πραγματοποιηθεί εύρεση, στον καθορισμένο χρόνο τότε επιστρέφει false. Η find συντάσσεται με τον ακόλουθο τρόπο:

```
stream.find(string)
```

findUntil()

Η findUntil αποτελεί μια υποπερίπτωση της find μόνο που εισάγεται η δυνατότητα να σταματήσει η ανάγνωση των δεδομένων όταν αναγνωσθεί μία ακολουθία χαρακτήρων (string) που έχει οριστεί να δηλώνει τον τερματισμό.

Η συνάρτηση διαβάζει τα δεδομένα μέχρι να συναντήσει την τερματική ακολουθία και εάν αναγνωσθεί η ζητούμενη ακολουθία χαρακτήρων επιστρέφει true.

```
stream.findUntil(string, termatikos)
```

peek()

Η peek διαβάζει ένα Byte την φορά χωρίς να προχωρήσει στην ανάγνωση του επόμενου. Σε περίπτωση που δεν υπάρχουν διαθέσιμα δεδομένα προς ανάγνωση επιστρέφει -1. Τρόπος σύνταξης:

```
stream.peek()
```

readBytes

Η συνάρτηση readBytes διαβάζει χαρακτήρες προερχόμενους από ροή δεδομένων στο buffer, η ανάγνωση τερματίζεται όταν έχει διαβαστεί ένα συγκεκριμένο μήκος ή παρέλθει ο χρόνος που έχει οριστεί. Η readBytes επιστρέφει τον αριθμό των Bytes που βρίσκονται στον buffer και επιστρέφει 0 εάν δεν βρεθούν δεδομένα. Παρακάτω παρουσιάζεται η σύνταξη της συνάρτησης:

```
stream.readBytes(buffer,length)
```

Υποπερίπτωση της readBytes() αποτελεί η συνάρτηση **readBytesUntil()** η οποία διαβάζει χαρακτήρες προερχόμενους από ροή δεδομένων στο buffer, έως ότου αναγνωσθεί μια συγκεκριμένη ακολουθία ή παρέλθει ένας ορισμένος χρόνος.

Η σύνταξη της συνάρτησης είναι ακόλουθη:

```
stream.readBytesUntil(character,buffer,length)
```

`character:` η ακολουθία χαρακτήρων τερματισμού.
`buffer:` ο καταχωρητής – απομονωτής όπου αποθηκεύονται τα δεδομένα (char[] ή byte[]).
`length:` ο αριθμός των bytes που πρέπει να διαβαστούν.

readString()

Μέσω της συνάρτησης readString() επιτυγχάνεται η ανάγνωση χαρακτήρων από μία ροή δεδομένων και η αποθήκευσή τους σε μορφή string. Η συνάρτηση επιστρέφει ως αποτέλεσμα μια ακολουθία χαρακτήρων (ένα string).

Υποπερίπτωση της παραπάνω συνάρτησης αποτελεί η συνάρτηση readStringUntil(char) όπου η ανάγνωση διακόπτεται όταν διαβαστεί ένας συγκεκριμένος χαρακτήρας.

parseInt() & parseFloat()

Στην περίπτωση που απαιτείται η εύρεση της απόλυτης τιμής του πρώτου ακέραιου αριθμού (τύπου long) σε μία ροή δεδομένων, χρησιμοποιείται η συνάρτηση parseInt(). Οι αριθμοί που δεν είναι ακέραιοι δεν λαμβάνονται υπόψη. Η συνάρτηση τερματίζεται όταν διαβαστεί ο ακέραιος χαρακτήρας και έχει την ακόλουθη μορφή:

```
stream.parseInt(string)
```

όπου string η ακολουθία που θα αναγνωσθεί.

Συχνά παρουσιάζεται η ανάγκη ανάγνωσης δεκαδικών αριθμών, στην περίπτωση αυτή χρησιμοποιείται η συνάρτηση parseFloat η οποία επιστρέφει την απόλυτη τιμή του πρώτου δεκαδικού αριθμού που διαβάζεται. Η συνάρτηση

τερματίζεται όταν διαβαστεί ό πρώτος δεκαδικός αριθμός και έχει την ακόλουθη μορφή:

```
stream.parseFloat(string)
```

όπου string η ακολουθία που θα αναγνωσθεί.

Serial και σειριακή μετάδοση δεδομένων

Η serial χρησιμοποιείται για την επικοινωνία της πλακέτας Arduino με ηλεκτρονικούς υπολογιστές και άλλες συσκευές. Όλες οι πλακέτες Arduino διαθέτουν μία τουλάχιστον σειριακή θύρα (γνωστή και ως UART ή USART – universal asynchronous receiver/transmitter). Η USART μετατρέπει τα δεδομένα σε τέτοια μορφή (σειριακή) ώστε να μπορούν να μεταφερθούν από ένα καλώδιο. Μέσω σειριακών θυρών πραγματοποιείται η μεταφορά δεδομένων αρκετών προτύπων επικοινωνίας, όπως τα RS-232, RS-422, RS-485 ή EIA. Ο χαρακτηρισμός της θύρας ως universal υποδηλώνει ότι η μορφή των δεδομένων και η ταχύτητα μετάδοσης είναι ελέγξιμες. Οι μέθοδοι και τα πλάτη των ηλεκτρικών σημάτων που μεταφέρονται καθορίζονται από κυκλώματα που δεν περιλαμβάνονται στη θύρα σειριακής επικοινωνίας. Η θύρα σειριακής επικοινωνία μπορεί να αποτελεί τμήμα ενός μικροελεγκτή.

Η σειριακή θύρα δέχεται bytes δεδομένων και μεταδίδει κάθε ένα bit του byte σε μορφή ακολουθίας, στον προορισμό μία δεύτερη σειριακή θύρα συνθέτει το αρχικό byte από τα ληφθέντα bit. Κάθε σειριακή θύρα περιλαμβάνει 2 τουλάχιστον shift register (για λήψη και για μετάδοση αντίστοιχα). Η σειριακή μετάδοση προτιμάται γιατί είναι αρκετά οικονομικότερη από την παράλληλη μετάδοση, όπου χρησιμοποιούνται περισσότεροι αγωγοί.

Όλες οι εργασίες υλικού (hardware) της σειριακής θύρας ελέγχονται από ένα σήμα ρολογιού που τυπικά λειτουργεί σε οχταπλάσια ταχύτητα από την ταχύτητα μεταφοράς (λογικό εάν ληφθεί υπόψη ότι πρέπει να μεταφερθούν 8 bit). Ο δέκτης ελέγχει την κατάσταση του εισερχόμενου σήματος σε κάθε παλμό ρολογιού αναζητώντας την αρχή του 1ου bit. Εάν η χρονική διάρκεια του bit αποτελεί τουλάχιστον το μισό του χρόνου τότε το bit θεωρείται αποδεκτό και σημαίνει την αρχή ενός νέου χαρακτήρα. Στην αντίθετη περίπτωση θεωρείται ένας λανθασμένος παλμός και αγνοείται. Στη συνέχεια η κατάσταση του αγωγού επανελέγχεται και το σήμα καταχωρείται στον αντίστοιχο shift register. Μετά την ολοκλήρωση των μεταφορών (8 για ένα Byte) τα δεδομένα της μεταφοράς γίνονται διαθέσιμα στο σύστημα του δέκτη. Σε αυτό το σημείο η σειριακή θύρα θα δηλώσει την διαθεσιμότητα δεδομένων με την χρήση flag και την πραγματοποίηση interrupt (διακοπής) στον επεξεργαστή για την μεταφορά των δεδομένων.

Οι εντολές της κλάσης serial χρησιμοποιούν για επικοινωνία τα ψηφιακά pin (ακίδες) 0 (RX - δέκτης) και 1 (TX - πομπός). Οπότε όταν γίνεται χρήση των συναρτήσεων της κλάσης serial πρέπει να αποφεύγεται η χρήση των συγκεκριμένων pin ως εισόδους ή εξόδους. Η επικοινωνία με ηλεκτρονικούς υπολογιστές πραγματοποιείται ως επί το πλείστον μέσω της USB. Για την επικοινωνία από υπολογιστή και συγκεκριμένα από το περιβάλλον ανάπτυξης λογισμικού, με την

πλακέτα Arduino μπορεί να χρησιμοποιηθεί το ενσωματωμένο monitor αρκεί να επιλεγεί το κατάλληλο baud rate μέσω της συνάρτησης begin.

Στην κλάση serial κληρονομούνται συναρτήσεις της κλάσης stream που αναλύθηκε προηγουμένως. Οι συναρτήσεις αυτές χρησιμοποιούνται για τον ίδιο λόγο και με τον ίδιο τρόπο που χρησιμοποιούνται και στην κλάση stream απλά τα δεδομένα προέρχονται από την σειριακή θύρα. Οι συναρτήσεις αυτές είναι οι εξής:

- find()
- findUntil()
- flush()
- parseInt()
- parseFloat()
- peek()
- readBytes()
- readBytesUntil()
- setTimeout()

Στη συνέχεια αναλύονται κάποιες συναρτήσεις της κλάσης Serial που παρέχουν την δυνατότητα επικοινωνίας.

if(Serial)

Μέσω της συνάρτησης if(serial) επιβεβαιώνεται εάν η σειριακή θύρα βρίσκεται σε κατάσταση ετοιμότητας για επικοινωνία. Επιστρέφεται TRUE εάν η θύρα είναι έτοιμη και FALSE στην αντίθετη περίπτωση.

Serial.available()

Η συνάρτηση Serial.available επιστρέφει τον αριθμό των Bytes (ακέραιο) που έχουν παραληφθεί και βρίσκονται διαθέσιμα στον buffer της σειριακής θύρας. Το μέγεθος του buffer είναι 64 Bytes

Serial.begin(speed,config)

Η συνάρτηση Serial.begin(speed,config) θέτει την ταχύτητα επικοινωνίας της σειριακής θύρας σε bit/sec (baud). Η ταχύτητα σύνδεσης υποδηλώνεται στην παράμετρο speed. Για την επικοινωνία με έναν ηλεκτρονικό υπολογιστή χρησιμοποιούνται οι ρυθμοί 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 ή 115200. Επίσης μπορούν χρησιμοποιηθούν και διαφορετικοί ρυθμοί για επικοινωνία με συσκευές που απαιτούν ειδικούς ρυθμούς επικοινωνίας.

Στην παράμετρο config, η οποία είναι προαιρετική, υποδηλώνεται το είδος των δεδομένων, εάν γίνεται χρήση bit ισοτιμίας (και το είδος) καθώς και ο αριθμός των bit τερματισμού.

Serial.end()

Η συνάρτηση Serial.end() απενεργοποιεί την σειριακή θύρα και καθιστά τα ψηφιακά pin 0 (RX)και 1(TX) ελεύθερα για χρήση ως είσοδοι ή έξοδοι.

Serial.print (val, format)

Η συνάρτηση Serial.print() επιτρέπει την εκτύπωση δεδομένων στη σειριακή θύρα με την μορφή δεδομένων ASCII. Η συγκεκριμένη συνάρτηση μπορεί να χρησιμοποιηθεί με διάφορους τρόπους σύνταξης για να εκτυπώσει δεδομένα. Οι δεκαδικοί αριθμοί μπορούν να εκτυπωθούν με ακρίβεια δύο δεκαδικών εάν δεν γίνει καθορισμός στην παράμετρο format, ενώ οι χαρακτήρες και οι ακολουθίες χαρακτήρων (strings) αποστέλλονται όπως δηλώνονται στην παράμετρο val.

Μέσω της παραμέτρου format (η οποία είναι προαιρετική) προσδιορίζεται ο τύπος του αριθμού (δεκαδικός, δυαδικός, οκταδικός, δεκαεξαδικός) ή ο αριθμός των δεκαδικών ψηφίων εάν πρόκειται για δεκαδικό αριθμό.

Μία υποπερίπτωση της Serial.print() μπορεί να θεωρηθεί η Serial.println() της οποίας η πρακτική διαφορά είναι ότι με τα την μετάδοση της εκτύπωσης γίνεται και αλλαγή της γραμμής εκτύπωσης των δεδομένων.

Serial.write()

Με την συνάρτηση Serial.write() επιτυγχάνεται η αποστολή δυαδικών δεδομένων στη σειριακή θύρα είτε ως μεμονωμένα Bytes είτε ως ακολουθία Bytes. Η συνάρτηση, ανάλογα με το τι μεταδίδεται παρουσιάζει διάφορους τρόπους σύνταξης:

```
Serial.write(value)
Serial.write(string)
Serial.write(buffer, length)
```

Όπου value κάποια τιμή που μεταδίδεται μήκους ενός Byte, string κάποια αλληλουχία Byte που μεταδίδεται και στην τρίτη περίπτωση μεταδίδεται ένας πίνακας απαρτιζόμενος από Byte όπου buffer η ονομασία του πίνακα και length το μήκος του πίνακα.

6.11 Αντικείμενο String

String

Η κλάση String επιτρέπει την χρήση και διαχείριση ακολουθιών χαρακτήρων (string) με περισσότερες δυνατότητες από ότι οι πίνακες χαρακτήρων. Μπορούν να συνενωθούν, να επισυναφθούν, να αντικατασταθούν τμήματα ακολουθιών κ.α. Ένα αντικείμενο της κλάσης String καταλαμβάνει περισσότερο χώρο από τον απλό πίνακα ακολουθιών προσφέρει όμως πολλές περισσότερες δυνατότητες. Ένα αντικείμενο κλάσης String σε σχέση με την σταθερά string γράφεται με κεφαλαίο S.

string(val, base)

Η συνάρτηση String(val, base) δημιουργεί ένα αντικείμενο κλάσης String χρησιμοποιώντας άλλους τύπους δεδομένων. Η μεταβλητή val καθορίζει τον τύπο δεδομένου που διαμορφώνεται ως String (string, char, byte, int, long, unsigned int, unsigned long). Η μεταβλητή base είναι προαιρετική και καθορίζει το σύστημα στο οποίο γίνεται η διαμόρφωση των δεδομένων (δεκαδικό, δυαδικό κτλ.)

string.charAt(n)

Η συνάρτηση String.charAt προσφέρει πρόσβαση στον n-οστό χαρακτήρα του αντικειμένου.

string.compareTo(string2)

Η συγκεκριμένη συνάρτηση δίνει τη δυνατότητα σύγκρισης δύο αντικειμένων String.

string.concat(string,string2)

Η συνάρτηση αυτή συνθέτει ένα νέο string απαρτιζόμενο από τα δύο προηγούμενα που καθορίζονται ως μεταβλητές.

string.endsWith(string2)

Η συνάρτηση ελέγχει εάν ένα string τερματίζεται με το string2.

string.equals(string2)

Η συνάρτηση ελέγχει εάν ένα string είναι ίσο με το string2, λαμβάνοντας υπόψη και τους κεφαλαίους χαρακτήρες.

string.equalsIgnoreCase(string2)

Η συνάρτηση ελέγχει για ισότητα αδιαφορώντας για τους κεφαλαίους χαρακτήρες.

string.getBytes(buf, len)

Η συνάρτηση αντιγράφει ένα string σε έναν buffer (πίνακα).

string.indexOf(val)

Η συνάρτηση εντοπίζει έναν χαρακτήρα ή ένα υπο-string(μια υποακολουθία χαρακτήρων) εντός ενός string.

string.lastIndexOf()

Η συνάρτηση εντοπίζει την τελευταία εμφάνιση ενός χαρακτήρα ή ενός υπό-string εντός ενός άλλου string.

string.length()

Η συνάρτηση επιστρέφει το μήκος ενός string δηλαδή τον αριθμό των χαρακτήρων που περιλαμβάνει.

string.replace(substring1, substring2)

Η συνάρτηση αυτή αντικαθιστά ένα συγκεκριμένο substring (τμήμα του string) με ένα άλλο.

string.reserve(size)

Η συνάρτηση αυτή επιτρέπει την δέσμευση ενός χώρου μνήμης με σκοπό την εκμετάλλευση από ένα string.

string.startsWith(string2)

Η συνάρτηση ελέγχει εάν ένα string ξεκινά με ένα άλλο.

string.substring(from,to)

Η συνάρτηση αυτή διαχωρίζει ένα τμήμα ενός string, δημιουργώντας ένα νέο το οποίο αποτελεί substring (υπό-string) του αρχικού. Δίνεται η δυνατότητα καθορισμού του χαρακτήρα από όπου αρχίζει (from) και τελειώνει (to) το νέο string.

string.toCharArray(buf, len)

Η συνάρτηση αντιγράφει τα δεδομένα ενός string σε έναν πίνακα χαρακτήρων.

string.toLowerCase()

Μετατρέπει όλα τα περιεχόμενα του string από κεφαλαία σε μικρά.

string.toUpperCase()

Μετατρέπει όλα τα περιεχόμενα του string από μικρά σε κεφαλαία.

string.trim()

Αφαιρεί τα κενά μέσα από το string.

Ο κώδικας του προγράμματος

```
int8_t answer;
int x;
int onModulePin= 2;
char SMS[200];
int LEDSTART=10;
int LEDSTOP=11;


void setup(){

    pinMode(onModulePin, OUTPUT);
    pinMode(LEDSTART, OUTPUT);
    pinMode(LEDSTOP, OUTPUT);
    Serial.begin(115200);

    Serial.println("Ekkinisi...");
    power_on();

    delay(3000);

// Eisagei to PIN tis kartas - Edw einai apenergopoiimeno
//sendATcommand("AT+CPIN=****", "OK", 2000);

    delay(3000);

    Serial.println("Setting SMS mode...");
    sendATcommand("AT+CMGF=1", "OK", 1000);
// Thetei tin morfi tou SMS se keimeno
sendATcommand("AT+CPMS=\"SM\",\"SM\",\"SM\"", "OK",
1000);    // epilegei tin thesi mnimis tis sim
sendATcommand("AT+CMGD=1,4","OK", 5000); // Delete all
SMS
}


void loop(){
   answer = sendATcommand("AT+CMGR=1", "+CMGR:", 2000);
// Diabazei to proto SMS
    if (answer == 1)
    {
        answer == 0;
        while(Serial.available() == 0);
        // Auti i epannaliptiki domi elegxei gia SMS
        do{
// Ean yparxoun dedomena stin seiriaki thyra(USART) ta
elegxei
            if(Serial.available() > 0){
                SMS[x] = Serial.read();
```

```
                x++;
                // Elegxei ean to minima einai START
                if (strstr(SMS, "START") != NULL)
                {
                    answer == 1;
                digitalWrite(LEDSTART,HIGH);
                delay(20000);
                digitalWrite(LEDSTART,LOW);
   sendATcommand("AT+CPMS=\"SM\",\"SM\",\"SM\"", "OK",
1000);    //Epilegei ti thesi mnimis tis kartas SIM
    sendATcommand("AT+CMGD=1,4","OK", 5000); // Diagrafei
ola ta SMS
              }

            }
            if(Serial.available() > 0){
                SMS[x] = Serial.read();
                x++;
                // Elegxei ean to minima einai STOP
            if (strstr(SMS, "STOP") != NULL)
                {
                    answer == 1;
                digitalWrite(LEDSTOP,HIGH);
                delay(20000);
                digitalWrite(LEDSTOP,LOW);

sendATcommand("AT+CPMS=\"SM\",\"SM\",\"SM\"", "OK",
1000);    //Epilegei ti thesi mnimis tis kartas SIM
    sendATcommand("AT+CMGD=1,4","OK", 5000); //Diagrafei
ola ta SMS
              }
            }
        }while(answer == 0);    //Anamenei gia tin
apantisi

        SMS[x] = '\0';

        Serial.print(SMS);

    }
    else
    {
        Serial.print("error ");
        Serial.println(answer, DEC);
    }


}


void power_on(){
```

```
    uint8_t answer=0;

    // Elegxei tin ekkinisi tou shield
    answer = sendATcommand("AT", "OK", 2000);
    if (answer == 0)
    {
        // palmos ekkinisis
        digitalWrite(onModulePin,HIGH);
        delay(3000);
        digitalWrite(onModulePin,LOW);

        // Perimenei gia apantisi apo to shield
        while(answer == 0){
            answer = sendATcommand("AT", "OK", 2000);
        }
    }

}
//Sunartisi ekteleseis AT entolwn tou shield
int8_t sendATcommand(char* ATcommand, char*
expected_answer, unsigned int timeout){
//orizetai mia sunartisi pou lambanei ws metablites ton
titlo tis AT command, tin anamenomeni apantisi p.x. OK
START STOP ktl. , kai to xroniko diastima oloklirwsis se
msec
    uint8_t x=0,  answer=0;
    char response[100];
    unsigned long previous;

    memset(response, '\0', 100);    // Ekkinisi tou
string

    delay(100);

    while( Serial.available() > 0) Serial.read();    //

    Serial.println(ATcommand);    // Apostoli (ektelesi)
tis AT command


        x = 0;
    previous = millis();

    // edw anamenetai i apantisi
    do{
        // ean yparxoun dedomena stin seiriaki thyra ta
diabazei kai elegxei gia tin apantisi
        if(Serial.available() != 0){
            response[x] = Serial.read();
            x++;
```

```
            // elegxetai ean i apantisi briskete sto
shield
            if (strstr(response, expected_answer) !=
NULL)
            {
                answer = 1;
            }
        }

    }while((answer == 0) && ((millis() - previous) <
timeout));

    return answer;
```

Συμπεράσματα

Κατά την εκπόνηση της διπλωματικής εργασίας αναπτύχθηκε το πνεύμα της συνεργασίας και της εύρεσης κοινού στόχου. Ανακαλύψαμε πως από την ιδέα στην εφαρμογή οι δυσκολίες ήταν πολλές και οι περισσότερες μη αναμενόμενες. Επίσης για την εφαρμογή χρειάστηκε μεγάλη έρευνα εξυπηρετώντας τους εκπαιδευτικούς σκοπούς μιας διπλωματικής.

Όσον αφορά την πρακτική εφαρμογή της διπλωματικής κατασκευάσαμε μια πλατφόρμα η οποία είναι χρηστικότατη και ιδιαίτερα επεκτάσιμη. Κατά την υλοποίηση της μας δημιουργήθηκαν πολλές ιδέες για εφαρμογές (όπως αγροτικές). Καθώς ο τρόπος που την κατασκευάσαμε την καθιστά ικανή να δέχεται και να επεξεργάζεται πολλά σήματα (π.χ. από διάφορους αισθητήρες) και επίσης μέσω των αυτόματων διακοπτών η χρήση της δεν περιορίζεται μόνο σε ασύγχρονες μηχανές αλλά ενδεχομένως θα μπορούσε να ελέγχει διάφορες συσκευές.

Τελικά θα μπορούσαμε να πούμε ότι ήταν μια πετυχημένη ιδέα από την αρχή μέχρι το τέλος της και δίνει την δυνατότητα και σε άλλους φοιτητές να την επεκτείνουν και να την εξειδικεύσουν.

Βιβλιογραφία

[1] Νικ. Π. Τρωγαδάς, «Εργαστηριακές Ασκήσεις Ηλεκτρικών Μηχανών AC», ΤΕΙ Λαρισας, Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Ηλεκτρολογίας.

[2] Ν. Π. Τρωγαδάς, «Ηλεκτρικές Μηχανές Σύγχρονες – Ασύγχρονες Μηχανές», ΤΕΙ Λαρισας, Σχολή Τεχνολογικών Εφαρμογών, Τμήμα Ηλεκτρολογίας.

[3] Κ. Βουρνά, Γ. Κονταξή, «Εισαγωγή στα Συστήματα Ηλεκτρικής Ενέργειας», Αθήνα 2001.

[4] Ι. Α. Τεγόπουλος, «Ηλεκτρικές Μηχανές, Μέρος Β», Εκδόσεις Συμμετρία, Αθήνα 1991.

[5] ATmega328P Textbook, ATMEL

[6] Γεώργιος Τσιλιγιάννης, «Σχεδίαση Και Κατασκευή Συστήματος Ελέγχου Φωτισμού Με Μικροελεγκτές AVR», Μεταπτυχιακή Διατριβή, Πανεπιστήμιο Πειραιώς, Οκτώβριος 2010.

[7] M. Mano, M. D. Ciletti «Ψηφιακή Σχεδίαση», Εκδόσεις Παπασωτηρίου, Αθήνα 2010.

[8] S. J. Chapman «Ηλεκτρικές Μηχανές AC – DC» Εκδόσεις Τζιόλα, Θεσσαλονίκη 1993.

[9] http://www.cooking-hacks.com/documentation/tutorials/arduino-gprs-gsm quadband-sim900

[10] http://www.arduino.cc/

[11] https://www.microchip.com/

[12] http://en.wikipedia.org/wiki/GSM

[13] http://code.google.com/p/gsm-shield-arduino/