ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ

ΙΔΡΥΜΑ ΠΑΤΡΑΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΙΑΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΡΙΘΜΟΣ 1105

ΘΕΜΑ: «ΠΑΡΑΔΕΙΓΜΑΤΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΤΟΥ ΜΙΚΡΟΕΛΕΓΚΤΗ AVR» - «PROGRAMMING EXAMPLES OF THE AVR MICROCONTROLLER»

ΕΙΣΗΓΗΤΗΣ: Κος ΚΑΡΕΛΗΣ ΔΗΜΗΤΡΙΟΣ

ΣΠΟΥΔΑΣΤΕΣ:

ΜΥΛΩΝΑΣ ΑΘΑΝΑΣΙΟΣ

ΠΑΡΑΜΕΡΙΤΗΣ ΓΕΡΑΣΙΜΟΣ

ΠΑΤΡΑ 2010

ΠΡΟΛΟΓΟΣ

Στην εποχή της Πληροφορίας και του «βομβαρδισμού» των νέων τεχνολογιών την οποία διανύουμε, ο φοιτητής των σχολών που σχετίζονται με το πεδίο της πληροφορικής και των υπολογιστικών συστημάτων, έρχεται συνεχώς αντιμέτωπος με έννοιες και αντικείμενα που δεν εμπίπτουν στο πεδίο των ευρύτερων γνώσεών του και στα οποία καλείται επιτυχώς να ανταπεξέλθει μέσα από τη μελέτη και την παρακολούθηση διαλέξεων και εργαστηριακών ωρών πάνω σε αυτά.

Το παρόν σύγγραμμα με τίτλο: «Παραδείγματα Προγραμματισμού του Μικροελεγκτή AVR», αποτελεί μία προσπάθεια μετασχηματισμού της επιστημονικής και συχνά «ξύλινης» για το ευρύ κοινό γλώσσας σε εκλαϊκευμένη και καθημερινή, με απλουστευμένες έννοιες, κατανοητές και εύχρηστες από όλους, τόσο από τον απλό αναγνώστη – μαθητή λυκείου που καλείται να κάνει μία εργασία πάνω στο μάθημα της Πληροφορικής όσο και για το φοιτητή που επιθυμεί να ασχοληθεί λίγο παραπάνω με το αντικείμενο αυτό. Στο πόνημα αυτό θα συναντήσουμε θέματα που σχετίζονται με την αρχιτεκτονική των μικροελεγκτών, τα περιφερειακά τους, τη σχεδίαση και τη χρήση τους σε σχέση με διάφορες εφαρμογές.

Ένα πολύ σημαντικό ερώτημα το οποίο ενδεχομένως να απασχολεί πολλούς είναι γιατί να ασχοληθούμε στην παρούσα εργασία με τους μικροελεγκτές και ιδιαιτέρως γιατί θεωρείται τόσο σημαντική η γνώση γύρω από τη σειρά των μικροελεγκτών AVR. Το ενδιαφέρον μας για τη σύνταξη του παρόντος εγχειρήματος προέκυψε διαβάζοντας μία σχετικά πρόσφατη αναφορά η οποία εκδόθηκε από τη Sun Microsystems το περασμένο έτος και στην οποία τονιζόταν ότι έχουμε φτάσει πλέον σε μία εποχή όπου ένα μέσο σπίτι περιλαμβάνει κατά μέσο όρο 90 – 120 συστήματα μικροελεγκτών τα οποία ελέγχουν διάφορες συσκευές, από τα ψηφιακά τηλέφωνα και τους φούρνους μικροκυμάτων έως τις τηλεοράσεις και τα συστήματα ασφαλείας. Κι όμως, παρά το γεγονός ότι ο αριθμός αυτός ενδεχομένως να φαντάζει κάπως εξωπραγματικός, εντούτοις είναι υπαρκτός. Η μαζική χρήση μικροεπεξεργαστών και μικροελεγκτών εκτός των ορίων ενός εργαστηρίου και πέραν των επιστημονικών πλαισίων είναι αδιαμφισβήτητο γεγονός. Τελικά, εκείνο το οποίο αντιλαμβάνεται κανείς είναι ότι εκτός ίσως από τον ανθρώπινο εγκέφαλο (ακόμη), οι μικροελεγκτές και οι μικροεπεξεργαστές έχουν κυριολεκτικά καταλάβει κάθε έκφανση της καθημερινής μας ζωής, ενώ συχνά οδηγούν σε οικονομικότερες λύσεις και μεγαλύτερα κέρδη για τις εταιρίες μαζικής παραγωγής τους (στο πεδίο του ανταγωνισμού πάντα). Σε αυτά τα πλαίσια οι μικροελεγκτές της οικογένειας AVR κατέχουν καίρια και εξέχουσα θέση.

Για λόγους λειτουργικότητας και καλύτερης οργάνωσης, η εργασία αυτή έχει κατανεμηθεί σε πέντε (5) ενότητες, κάθε μία από τις οποίες έχει διακριτό σκοπό και στόχο. Το πρώτο κεφάλαιο φέρει τον τίτλο «Εισαγωγή» και ασχολείται με την αναλυτική παρουσίαση των όρων του μικροεπεξεργαστή και μικροελεγκτήϋ δύο εννοιών με διαφορετικά χαρακτηριστικά τις οποίες συχνά ταυτίζουμε. Αφού παρουσιαστούν οι δύο αυτές έννοιες ξεχωριστά με εννοιολογική αναφορά και ανάλυση του σκοπού λειτουργίας τους μέσα σε ένα μικροϋπολογιστικό σύστημα, καθώς και των πλεονεκτημάτων και μειονεκτημάτων που τις χαρακτηρίζουν, συντελείται μία προσπάθεια να συγκριθούν σε επίπεδο λειτουργικότητας, αρχιτεκτονικής, υπολογιστικής ισχύς και υποσυστημάτων που τις πλαισιώνουν.

Αν το πρώτο κεφάλαιο ασχολείται με την παρουσίαση των όρων του μικροεπεξεργαστή και του μικροελεγκτή, τότε το δεύτερο που έχει τον τίτλο «Η αρχιτεκτονική των μικροελεγκτών AVR», δίνει έμφαση σε μία πιο αναλυτική παρουσίαση της οικογένειας των μικροελεγκτών AVR. Γίνεται μία προσπάθεια να παρουσιαστούν αναλυτικά και με κατανοητό τρόπο τα χαρακτηριστικά της αρχιτεκτονικής τους, των ειδών μνήμης που διαθέτουν, τις εντολές τις οποίες καλούνται να κωδικοποιήσουν και να εκτελέσουν, τις θύρες επικοινωνίας (εισόδου και εξόδου) και όλων γενικά των στοιχείων και ενεργειών που σχετίζονται με την ομαλή λειτουργία και διαχείριση των δεδομένων στο εσωτερικό ενός υπολογιστικού συστήματος.

Το τρίτο κεφάλαιο έχει τον τίτλο «Περιφερειακά των μικροελεγκτών AVR» και ασχολείται με όλες εκείνες τις περιφερειακές μονάδες, οι οποίες πλαισιώνουν έναν μικροελεγκτή. Παρουσιάζεται η μονάδα UART, η οποία χρησιμοποιείται προκειμένου να υλοποιηθεί με επιτυχία η αποστολή και λήψη δεδομένων από ένα PC, για επικοινωνία με το χρήστη, αποσφαλμάτωση κώδικα και άλλα. Επίσης, η μονάδα σύγχρονης σειριακής επικοινωνίας SPI (Serial Peripheral Interface), η οποία είναι ένα περιφερειακό που χρησιμοποιείται για την επικοινωνία του μικροελεγκτή με διάφορες διατάξεις, όπως εξωτερικές μνήμες EEPROM, μετατροπείς DAC και ADC, άλλους μικροελεγκτές. Ακόμη, ο χρονιστής, ο οποίος μετρά τους παλμούς ρολογιού ενός μικροελεγκτή AVR, ο αναλογικός συγκριτής, που σχετίζεται με τη σύγκριση δύο αναλογικών σημάτων και τέλος η ενσωματωμένη μνήμη EEPROM.

Το επόμενο κεφάλαιο «Η γλώσσα Προγραμματισμού C», παρουσιάζει ένα γενικό πλαίσιο για την οργάνωση των εντολών που μπορούν να μεταφραστούν σε κώδικα μηχανής, ξεκινώντας από κάποια γενικά χαρακτηριστικά των γλωσσών μηχανής και προχωρώντας στη γλώσσα C μέσα από μία σύντομη ιστορική αναδρομή, σε έναν ορισμό της λειτουργίας της ως μίας γλώσσας γενικού σκοπού που υποστηρίζει πλήρως τον προγραμματισμό μέσω διαδικασιών (με τη μορφή

συναρτήσεων), αφού πρόκειται για μία γλώσσα μέσου και χαμηλού επιπέδου, γεγονός το οποίο σημαίνει ότι βρίσκεται κοντά στο υλικό (hardware) ενός υπολογιστή. Αναφορά γίνεται επίσης στα πλεονεκτήματα και μειονεκτήματα της γλώσσας αυτής, στους τύπους δεδομένων και τις μεταβλητές που διαθέτει, καθώς και στην παρουσίαση της δομής – του γραψίματος ενός απλού προγράμματος σε C.

Τέλος, η εργασία αυτή ολοκληρώνεται με το πέμπτο (5^{o}) κεφάλαιο και τις τρεις εφαρμογές των μικροελεγκτών AVR. Η πρώτη από αυτές αναφέρεται στα συστήματα διασφάλισης λογισμικού, δηλαδή σε ένα σχήμα εξουσιοδότησης σύμφωνα με το οποίο κάποιο συγκεκριμένο λογισμικό, ενδεχόμενα σε συνεργασία και με κάποιο αντίστοιχο υλικό, επιτρέπεται να χρησιμοποιηθεί από εξουσιοδοτημένους χρήστες και μόνον. Η δεύτερη σχετίζεται με την παρουσίαση του συστήματος χειρισμού του κώδικα Morse και η τρίτη με τη λειτουργία ενός χρονοδιακόπτη κουζίνας.

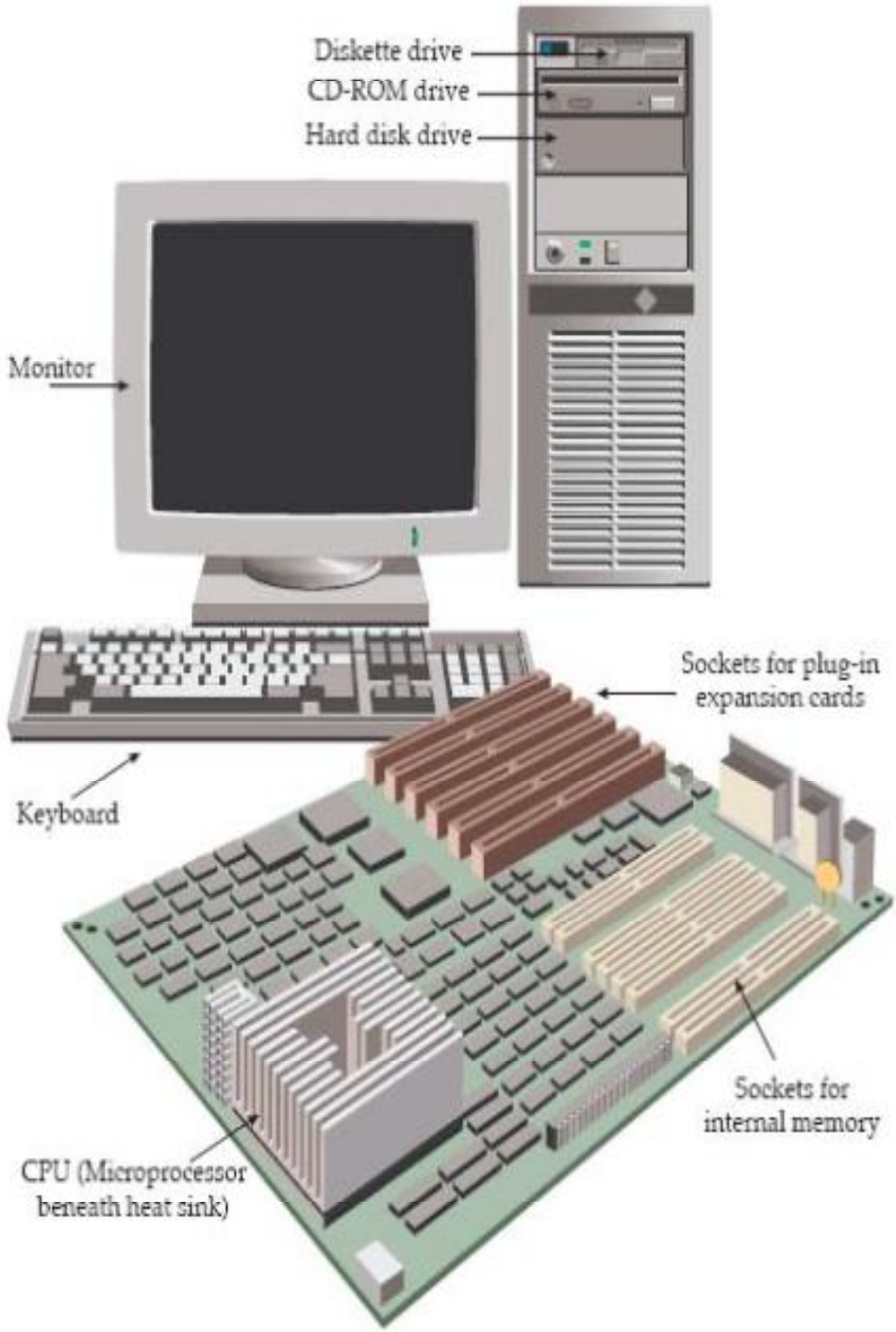
Ολοκληρώνοντας, θα θέλαμε να κλείσουμε ευελπιστώντας ότι το σύγγραμμα που κρατάτε στα χέρια σας θα συμβάλει στις σπουδές σας ή ακόμη και στο επαγγελματικό σας έργο και για το λόγο αυτό ελπίζουμε σε μία καλή κριτική και εποικοδομητική ανάγνωση.

Μυλωνάς Αθανάσιος

και

Παραμερίτης Γεράσιμος

Πάτρα – Οκτώβριος 2010


Diskette drive
CD-ROM drive
Hard disk drive
Monitor
Sockets for plug-in expansion cards
Keyboard
Sockets for internal memory
CPU (Microprocessor beneath heat sink)

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1^{Ο} - ΕΙΣΑΓΩΓΗ

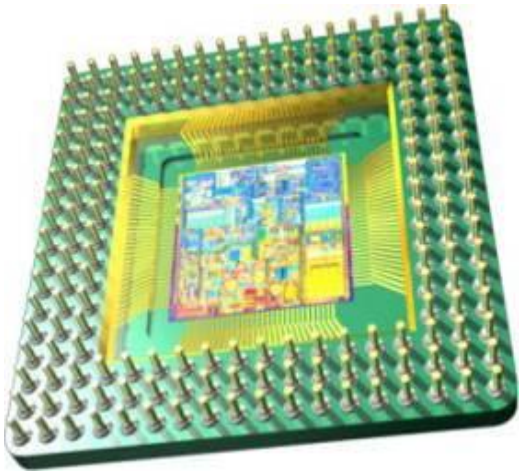
Οι ηλεκτρονικοί υπολογιστές τα τελευταία χρόνια διαδραματίζουν έναν ολοένα και πιο σημαντικό ρόλο στην καθημερινή μας ζωή. Από τα αποτελέσματα των πανελληνίων εξετάσεων και τα εκκαθαριστικά σημειώματα της εφορίας μέχρι τη μισθοδοσία χιλιάδων υπαλλήλων, καταστάσεις οι οποίες στο παρελθόν απαιτούσαν χρόνο και κόπο για την επίλυσή τους λόγω της απουσίας των ηλεκτρονικών υπολογιστών, τώρα πλέον συντελούνται με μερικούς απλούς χειρισμούς. Σε πολλές περιπτώσεις πλέον, οι Η/Υ είναι συνυφασμένοι με αυτήν την ίδια την ανθρώπινη ζωή, όπως στην τηλεϊατρική, τα χειρουργικά ρομπότς, τα συστήματα ελέγχου πυρηνικών κεφαλών, τις τηλεπικοινωνίες και τον έλεγχο αεροδιαδρόμων.

Με βάση λοιπόν αυτήν την ραγδαία εξέλιξη και τις ολοένα αυξανόμενες απαιτήσεις για εξοικονόμηση χρόνου και χρημάτων, ο ρόλος του Μηχανικού Η/Υ και Πληροφορικής, αλλά και των εταιρίων δημιουργίας λογισμικών στη σύγχρονη εποχή αποκτά αυξανόμενη βαρύτητα, καθώς για να ανταποκριθούν στις προκλήσεις και τον έντονο ανταγωνισμό θα πρέπει να διαθέτουν ένα ικανό γνωστικό υπόβαθρο με όλα τα τελευταία επιστημονικά επιτεύγματα που θα τους επιτρέπουν να οραματίζονται και να σχεδιάζουν πρωτότυπες τεχνολογικές, αλλά και ανταγωνιστικές εφαρμογές. Είναι γνωστό πλέον ότι η σύγχρονη τεχνολογία και τα μικροϋπολογιστικά συστήματα βασίζονται στους Μικροεπεξεργαστές. Ειδικότερη μορφή τους είναι οι μικροελεγκτές οι οποίοι εμφανίζονται σχεδόν σε κάθε ηλεκτρονική συσκευή και σχεδόν στο σύνολο των εφαρμογών της ηλεκτρονικής τεχνολογίας.

Ο κλάδος των μικροελεγκτών είναι ένα ταχύτατα αναπτυσσόμενο πεδίο με πληθώρα εφαρμογών στην καθημερινή ζωή λόγω του χαμηλού κόστους και της ευελιξίας τους. Στο παρόν κεφάλαιο επιχειρείται μία προσπάθεια συγκριτικής παρουσίασης και ανάλυσης των όρων μικροεπεξεργαστή και μικροελεγκτή. Έχουμε συνηθίσει να ακούμε αρκετά συχνά τους όρους αυτούς χωρίς κανέναν ιδιαίτερο διαχωρισμό, αν και στην πραγματικότητα κάθε ένας τους έχει διακριτή σημασία και μεταξύ τους εμφανίζουν σημαντικές διαφορές.

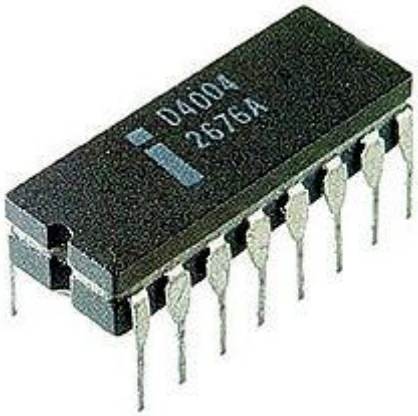
1.1 Μικροεπεξεργαστές

Ο μικροεπεξεργαστής, ο οποίος θεωρείται ως η «καρδιά» του υπολογιστή και άλλων ηλεκτρονικών συσκευών, ψηφίστηκε –και μάλιστα με μεγάλη διαφορά- ως η σημαντικότερη τεχνολογική ανακάλυψη του τελευταίου μισού αιώνα σε «γκάλοπ» που έκανε το έγκυρο επιστημονικό περιοδικό «New Scientist». Στη δεύτερη θέση βρέθηκε το Ίντερνετ και ο Παγκόσμιος Ιστός, ενώ όλες οι άλλες εφευρέσεις θεωρήθηκαν…απλές οδοντόκρεμες.



Η ακριβής ερώτηση που τέθηκε στο κοινό ήταν «Ποιά ανακάλυψη είχε την μεγαλύτερη επίπτωση στον κόσμο κατά τα τελευταία 50 χρόνια». Ο μικροεπεξεργαστής έλαβε το 48% των ψήφων και ακολούθησαν κατά σειρά το Web (31%), η εξερεύνηση του διαστήματος (3%), το κινητό τηλέφωνο (3%), το παγκόσμιο δίκτυο οπτικών ινών (3%) και τα λέιζερ (2%).

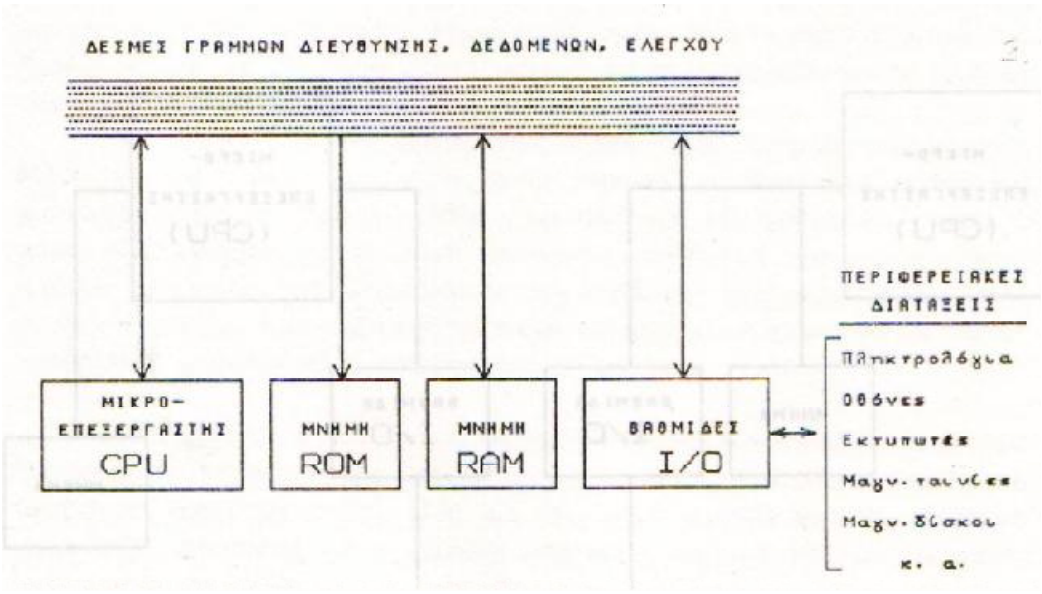
Τί είναι όμως ένας μικροεπεξεργαστής; Ένας **μικροεπεξεργαστής** περιλαμβάνει τις περισσότερες ή όλες τις λειτουργίες μιας κεντρικής μονάδας επεξεργασίας (ΚΜΕ) ενός υπολογιστή σε ένα ενιαίο ολοκληρωμένο κύκλωμα (IC). Οι πρώτοι μικροεπεξεργαστές εμφανίστηκαν στις αρχές της δεκαετίας του 1970 και χρησιμοποιήθηκαν σε ηλεκτρονικές αριθμομηχανές. Η ενσωμάτωση των μικροεπεξεργατών σε άλλες συσκευές, όπως τερματικά, εκτυπωτές κλπ, ακολούθησε σχετικά γρήγορα. Με τη χρήση ενός οκτάμπιτου μικροεπεξεργαστή, κατασκευάστηκε ο πρώτος μικροϋπολογιστής γενικού σκοπού στα μέσα της δεκαετίας του 1970. Η ραγδαία ανάπτυξη της τεχνολογίας των μικροεπεξεργαστών που ακολούθησε συνδέεται με τις αυξημένες απαιτήσεις από γλώσσες προγραμματισμού υψηλού επιπέδου.



Σχήμα 1.1:Ο Intel 4004, από τους πρώτους μικροεπεξεργαστές που κατασκευάστηκαν.

Κάθε μικροϋπολογιστής είναι ένα σύνθετο σύστημα που αποτελείται από τρεις βαθμίδες:

1. Το μικροεπεξεργαστή (Central Processing Unit, CPU), ο οποίος περιλαμβάνει την αριθμητική και λογική μονάδα (ALU), καθώς και τη μονάδα ελέγχου (CU).
2. Τη μνήμη, η οποία συγκρατεί τα προγράμματα και τα δεδομένα και
3. Τη βαθμίδα εισόδου – εξόδου (I/O), που επιτρέπει τη σύνδεση του μικροεπεξεργαστή με εξωτερικές διατάξεις, όπως πληκτρολόγια και εκτυπωτές.



Σχήμα 1.2: Βαθμίδες μικροϋπολογιστικού συστήματος

Όπως φαίνεται στο σχήμα, η μνήμη και η διάταξη I/O επικοινωνούν με το μικροεπεξεργαστή. Η επικοινωνία γίνεται μέσω μιας σειράς δεσμών γραμμών επικοινωνίας (buses). Στη σύνθεση αυτή η CPU έχει τον πλήρη έλεγχο. Η δέσμη των γραμμών επικοινωνίας

μεταφέρει τα σήματα για την εκτέλεση μίας εργασίας (εντολής) κάθε φορά. Αυτό ακριβώς επιβάλει την ταξινόμηση των εντολών σε ακουλουθία. Είναι χαρακτηριστικό ότι τόσο η μνήμη, όσο και η βαθμίδα I/O δε δίνουν εντολές στη CPU παρά μόνο δέχονται εντολές από αυτή.

Η εσωτερική οργάνωση της CPU διαφέρει στους διάφορους τύπους μικροεπεξεργαστών, ωστόσο ορισμένα στοιχεία είναι κοινά. Περιλαμβάνει τα εξής:

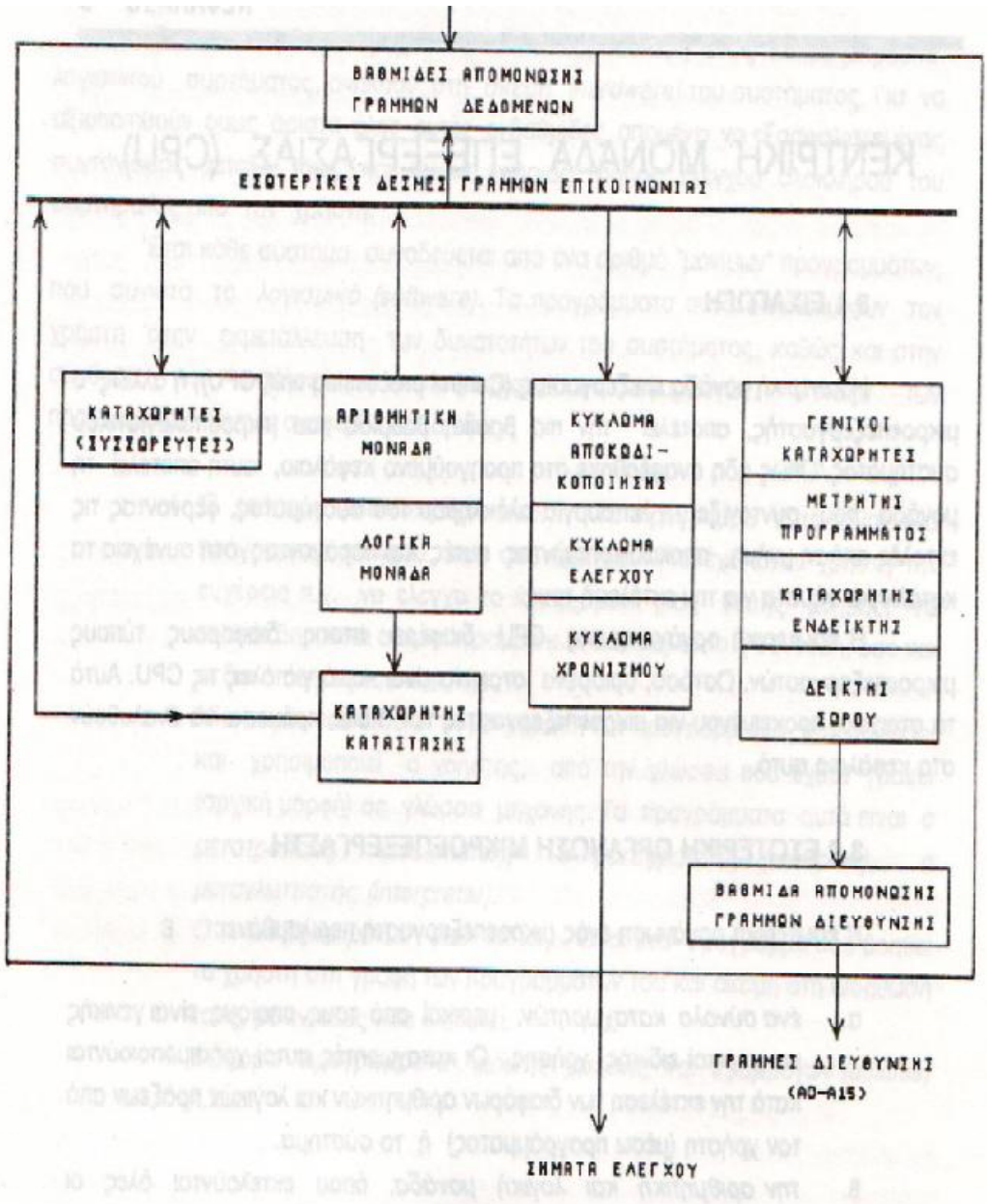
1. Έναν αριθμό καταχωρητών, οι οποίοι χρησιμοποιούνται για την αποθήκευση δεδομένων και ενδιάμεσων αποτελεσμάτων. Εκτός από τους καταχωρητές γενικής χρήσης (general purpose registers), υπάρχουν και ορισμένοι ειδικοί, μερικοί από τους οποίους δε χρησιμοποιούνται στον προγραμματισμό, όπως ο καταχωρητής κατάστασης ή αναζωογόνησης. Οι καταχωρητές αυτοί χρησιμοποιούνται κατά την εκτέλεση διαφόρων αριθμητικών και λογικών πράξεων από το χρήστη (μέσω κάποιου προγράμματος) ή το σύστημα.
2. Την αριθμητική και λογική μονάδα, όπου εκτελούνται όλες οι αριθμητικές και λογικές πράξεις, οι οποίες ορίζονται από το πρόγραμμα και
3. Τη μονάδα ελέγχου, η οποία συντονίζει τις εργασίες προσκόμισης των εντολών από τη μνήμη, της αποκωδικοποίησης και στη συνέχεια της δημιουργίας των κατάλληλων σημάτων για την εκτέλεσή τους. Οι βαθμίδες αυτές επικοινωνούν μεταξύ τους με τη βοήθεια μιας ή περισσοτέρων εσωτερικών δεσμών γραμμών επικοινωνίας (internal buses). Με αυτές γίνεται η μεταφορά δεδομένων στο εσωτερικό του μικροεπεξεργαστή.

Ένας σύγχρονος μικροεπεξεργαστής συγκροτείται συνεπώς από δύο βασικά τμήματα: τη μονάδα αριθμητικής και λογικής (ALU, Arithmetic and Logic Unit), όπου γίνονται όλες οι αριθμητικές και λογικές πράξεις και τη μονάδα ελέγχου (CU, Control Unit), όπου παράγονται τα απαραίτητα σήματα, τα οποία εξασφαλίζουν τις απαιτούμενες διαδρομές δεδομένων σύμφωνα με τις εντολές του προγράμματος. Πιο αναλυτικά αποτελείται από τις ακόλουθες μονάδες:

- **Μονάδα αποκωδικοποίησης** (Decoding Unit).
- **Αριθμητική και Λογική Μονάδα** (Arithmetic and Logical Unit, ALU): Η μονάδα στην οποία εκτελούνται μία προς μία οι αριθμητικές ή λογικές πράξεις, όπως υπαγορεύονται από τις εντολές που έχουν δοθεί στον υπολογιστή.
- **Καταχωρητές** (Registers): Μικρά κελιά μνήμης στο εσωτερικό του επεξεργαστή, που χρησιμοποιούνται για την προσωρινή αποθήκευση των δεδομένων, καθώς αυτά υφίστανται επεξεργασία. Οι καταχωρητές διαφέρουν ανάλογα με τον τύπο του

επεξεργαστή και τον κατασκευαστή, τόσο ως προς την οργάνωση όσο και ως προς τη χωρητικότητά τους.

- **Μονάδα ελέγχου** (Control Unit): Ελέγχει τη ροή δεδομένων από και προς την ALU, τους καταχωρητές, τη μνήμη και τις περιφερειακές μονάδες εισόδου/εξόδου.
- **Μονάδα προσκόμισης** (Fetch Unit): Μεταφέρει τις εντολές από τη μνήμη στον επεξεργαστή.
- **Μονάδα προστασίας** (Protection Unit): Εξασφαλίζει το αποδεκτό της κάθε διεργασίας που εκτελεί ο επεξεργαστής, ώστε να μη τροποποιούνται δεδομένα που δεν πρέπει ή να μην εκτελούνται μη αποδεκτές εντολές, όπως π.χ. διαίρεση αριθμού με το μηδέν.



Σχήμα 1.3: Εσωτερική οργάνωση CPU

Το σχήμα δίνει μία παράσταση του εσωτερικού της CPU. Όπως φαίνεται σ'αυτό, η δέσμη γραμμών δεδομένων, καθώς και αυτή των διευθύνσεων του συστήματος ξεκινούν από αντίστοιχες βαθμίδες απομόνωσης (buffers) του μικροεπεξεργαστή.Η διασύνδεση όλων αυτών των μονάδων αποτελεί το υλικό (hardware) του συστήματος. Ένα πρόγραμμα υπολογιστή δεν είναι τίποτε περισσότερο από μια λίστα με οδηγίες για έναν μικροεπεξεργαστή να πραγματοποιήσει. Ο μικροεπεξεργαστής είναι ουσιαστικά μια Μονάδα Κεντρικής Επεξεργασίας (CPU) υλοποιημένη μέσα σε ένα ολοκληρωμένο κύκλωμα που σκοπό έχει να προγραματιστεί. Αυτό σημαίνει ότι ο προγραμματιστής ενός υπολογιστή, μπορεί να περιγράψει τις ενέργειες που θα κάνει ένας μικροεπεξεργαστής και πως θα αντιδράσει σε εξωτερικές συνθήκες με τη βοήθεια των γλωσσών προγραμματισμού. Ό,τι κάνει ο μικροεπεξεργαστής δεν περιλαμβάνει τίποτε περισσότερο από μια σειρά από αυτές τις οδηγίες βήμα-βήμα. Ένα πρόγραμμα υπολογιστή είναι απλά μια λίστα με τις οδηγίες ενός μικροεπεξεργαστή.

Γενικά υπάρχει μία πληθώρα εργαλείων που βοηθούν στον προγραμματισμό των μικροεπεξεργαστών και με τον τρόπο αυτό δίνεται η δυνατότητα στο χρήστη να πραγματοποιεί περίπλοκες λειτουργίες και εντολές, χωρίς κανέναν περιορισμό. Ο μικροεπεξεργαστής σήμερα χρησιμοποιείται σχεδόν παντού, από τις πιο σύνθετες και εξελιγμένες επιστημονικές συσκευές έως τα παιδικά παιχνίδια. Αποτελεί το απαραίτητο συστατικό σχεδόν κάθε συσκευής οικιακής, ιατρικής, ακόμη και τηλεπικοινωνιακών συστημάτων, μουσικών οργάνων ή συσκευών μετρήσεων (παλμογράφοι, πολύμετρα) και μαζί με τον υπολογιστή αποτελούν τη «δίδυμη» τεχνολογική επανάσταση που ακολούθησε μετά την αρχική επανάσταση της μηχανής, αρχικά της ατμομηχανής και μετά της εσωτερικής καύσης. Η μηχανή επεξέτεινε τη φυσική δύναμη του ανθρώπου και ο μικροεπεξεργαστής-υπολογιστής τη νοητική. Ο συνδυασμός τους έχει μεταμορφώσει πια δραστικά (και μερικές φορές δραματικά) τον ανθρώπινο πολιτισμό.

Όπως δήλωσε ο πρωτοπόρος Φεντερίκο Φράγκιν (μέλος της ερευνητικής ομάδας που ανέπτυξε τον Intel 4004, τον πρώτο εμπορικό μικροεπεξεργαστή στον κόσμο), οι μικροεπεξεργαστές και οι ημιαγωγοί συνεχίζουν σήμερα να συν-εξελίσσονται, με την εξέλιξη στο ένα πεδίο να τροφοδοτεί το άλλο, σε έναν «ενάρετο» κύκλο ανάπτυξης και συνεχών βελτιώσεων, που οδηγεί σε ολοένα πιο μικροσκοπικά, φθηνά και γρήγορα προϊόντα.

Ο μικροεπεξεργαστής, που εφευρέθηκε στις αρχές της δεκαετίας του ΄70 στις ΗΠΑ, χρησιμοποιήθηκε αρχικά στους υπολογιστές ως μια ενδιάμεση και φθηνή λύση στον έλεγχο συστημάτων μεταξύ των μεγάλων υπολογιστών και του ειδικού υλικού που έπρεπε να κατασκευαστεί κατά περίπτωση και σήμερα η επεξεργαστική ισχύς του βρίσκεται πίσω από

οποιαδήποτε εξέλιξη, από τους υπερ-υπολογιστές έως τα «έξυπνα» κινητά τηλέφωνα, αλλά και σε πολλές χιλιάδες ακόμα ηλεκτρονικά προϊόντα (π.χ. παιχνίδια), αλλάζοντας έτσι στην πορεία τις ζωές δισεκατομμυρίων ανθρώπων. Σήμερα δεν υπάρχει πια καμία βιομηχανία και σχεδόν κανένας άνθρωπος στον κόσμο, που άμεσα ή έμμεσα να μην έχει επηρεαστεί από τους μικροεπεξεργαστές.

Αυτή η εκρηκτική ανάπτυξη των τελευταίων ετών και η πληθώρα των αναγκών δημιούργησε και μια μεγάλη ποικιλία διαφορετικών τύπων μικροϋπολογιστών. Το ερώτημα βέβαια που προκύπτει είναι γιατί να κατασκευάσουμε χιλιάδες διαφορετικούς τύπους μικροϋπολογιστών – μικροεπεξεργαστών, αλλά το κυριότερο εκατομμύρια αντίτυπά τους. Η απάντηση είναι απλή και στηρίζεται στο γεγονός ότι χρειάζονται πλέον σε καθε ηλεκτρική συσκευή. Μπορούμε να διακρίνουμε τρεις κατηγορίες μικροεπεξεργαστών:

Α. Αυτούς που χρησιμοποιούνται σε υπολογιστές γενικής χρήσης, δηλαδή στους προσωπικούς υπολογιστές για εφαρμογές γραφείου, όπως επεξεργασία κειμένου ή για εφαρμογές που απαιτούν αριθμητικές πράξεις και διακρίνονται για τα προχωρημένα αρχιτεκτονικά χαρακτηριστικά τους. Σε αυτούς τους μικροεπεξεργαστές η επιδίωξη είναι να έχουμε υψηλές επιδόσεις σε ένα ευρύ φάσμα εφαρμογών.

Β. Επεξεργαστές σήματος, οι οποίοι σχετίζονται με το χειρισμό και την επεξεργασία σημάτων. Χωρίς τη χρήση τους δε θα ήταν εφικτή η υλοποίηση της ψηφιακής τεχνολογίας, καθώς στη σημερινή ψηφιακή εποχή, κάθε είδος σήματος, όπως εικόνα ή ήχος δειγματοληπτείται και η αντίστοιχη τιμή του μετατρέπεται σε έναν δυαδικό αριθμό. Διαθέτει τα εξής πλεονεκτήματα:

Επεξεργασία	Αποθήκευση	Μετάδοση
• Ευελιξία • Υλοποίηση αλγορίθμων με λογισμικό	• Μεγάλη χωρητικότητα • Ακρίβεια • Ελαχιστοποίηση σφαλμάτων • Ευκολία και ταχύτητα ενταμίευσης/ ανάκτησης	• Υψηλές ταχύτητες • Ευκολία και ευελιξία • Υλοποίηση με λογισμικό πρωτοκόλλων, δρομολόγησης και ελέγχου σφαλμάτων • Ακρίβεια • Ελαχιστοποίηση

		σφαλμάτων

Ο μικροεπεξεργαστής (micro-processor), είναι η βασικότερη μονάδα ενός μικροϋπολογιστικού συστήματος, γιατί:

- Συντονίζει τη λειτουργία ολόκληρου του συστήματος, φέρνοντας τις εντολές από τη μνήμη, αποκωδικοποιώντας τες και στη συνέχεια παράγοντας τα κατάλληλα σήματα για την εκτέλεσή τους.
- Ταυτόχρονα ελέγχει τις υπόλοιπες περιφερειακές μονάδες, μεταφέροντας δεδομένα από και προς τη μνήμη, καθώς επίσης από και προς τις μονάδες εισόδου / εξόδου.
- Επιπλέον, εκτελεί αριθμητικές και λογικές πράξεις.
- Ανταποκρίνεται σε σήματα διακοπών και ελέγχου και τέλος
- Διακλαδώνει την ομαλή ακολουθιακή ροή ενός προγράμματος σε άλλο σημείο, σε υπορουτίνα, επιστροφή από υπορουτίνα και αποκρίνεται σε διακοπές από εξωτερικά σήματα ή από το πρόγραμμα.

Γ. Οι μικροελεγκτές στους οποίους θα αναφερθούμε αναλυτικά στη συνέχεια.

1.1.1 Σχεδιαστές μικροεπεξεργαστών

Οι πιο σημαντικές και ευρέως διαδεδομένες εταιρίες σχεδιασμού μικροεπεξεργαστών της αγοράς είναι οι ακόλουθες:

- Advanced Micro Devices - Advanced Micro Devices, κατασκευαστής επεξεργαστών κυρίως για προσωπικούς υπολογιστές.
- ARM Ltd - μια από τις λίγες εταιρίες που δεν κατασκευάζει επεξεργαστές, αλλά παραχωρεί δικαιώματα κατασκευής επεξεργαστών τις σε τρίτους. Η αρχιτεκτονική ARM είναι από τις πλέον δημοφιλής στην κατηγορία των μικροελεγκτών για ενσωματωμένα (embedded) συστήματα.
- Freescale Semiconductor (πρώην τμήμα της Motorola) - σχεδιαστής αρκετών ενσωματωμένων και SoC επεξεργαστών για PowerPC.
- IBM Microelectronics - Μικροηλεκτρονικό τμήμα της IBM, αναπτύσσει επεξεργαστές που προορίζονται για POWER και PowerPC υπολογιστές, αλλά και πολλών επεξεργαστών που χρησιμοποιούνται σε κονσόλες βιντεοπαιχνιδιών.

- Intel Corp - Intel, κατασκευαστής αξιοσημείωτων μικροελεγκτών και επεξεργαστών, όπως οι 8051, IA-32, IA-64 και XScale. Επιπλέον παράγει περιφερειακά τσιπ για χρήση με τους επεξεργαστές της.
- MIPS Technologies - ανέπτυξαν την αρχιτεκτονική MIPS, μια πρωτοπόρος σχεδίαση των RISC επεξεργαστών.
- NEC Electronics - ανέπτυξαν την αρχιτεκτονική 78K0 8-bit, την αρχιτεκτονική 78K0R 16-bit, και την αρχιτεκτονική V850 32-bit.
- Sun Microsystems - Sun Microsystems, ανέπτυξαν την αρχιτεκτονική SPARC, μια σχεδίαση RISC.
- Texas Instruments - Σχεδιαστές και κατασκευαστές αρκετών διαφορετικών χαμηλής κατανάλωσης μικροελεγκτών μεταξύ των άλλων ημιαγωγικών προϊόντων τους.
- Transmeta - Δημιουργοί επεξεργαστών χαμηλής κατανάλωσης, όπως οι Crusoe και Efficeon, που είναι συμβατοί με το σετ εντολών x86.
- VIA Technologies - Κατασκευαστής, από την Ταϊβάν, χαμηλής κατανάλωσης επεξεργαστών, συμβατοί με το σετ εντολών x86.
- Atmel - Κατασκευαστής δημοφιλών μικροελεγκτών CISC και RISC χαμηλού κόστους, όπως η σειρά AVR.

1.2 Μικροελεγκτές:

Ο **μικροελεγκτής** (αγγλικά, *microcontroller*) είναι ένας τύπος επεξεργαστή, ουσιαστικά μια παραλλαγή μικροεπεξεργαστή, ο οποίος μπορεί να λειτουργήσει με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών ενσωματωμένων υποσυστημάτων που διαθέτει. Χρησιμοποιείται ευρύτατα σε όλα τα ενσωματωμένα συστήματα (embedded systems) ελέγχου χαμηλού και μεσαίου κόστους, όπως αυτά που χρησιμοποιούνται σε αυτοματισμούς, ηλεκτρονικά καταναλωτικά προϊόντα (από ψηφιακές φωτογραφικές μηχανές έως παιχνίδια), ηλεκτρικές συσκευές και κάθε είδους αυτοκινούμενα τροχοφόρα οχήματα.

Ο μικροελεγκτής είναι ένας υπολογιστής χωρίς περιφερειακά, σε ένα ολοκληρωμένο κύκλωμα. Πρόκειται για μια κατηγορία μικροεπεξεργαστή, ο οποίος βρίσκει εφαρμογή στις ηλεκτρονικές συσκευές, στις τηλεπικοινωνίες και στη βιομηχανία. Κάθε συσκευή, μηχάνημα ή ηλεκτρονικό σύστημα εκτός από τον έλεγχο της σωστής και αποδοτικής λειτουργίας του, χρειάζεται επίσης χειρισμό, ρύθμιση και παρακολούθηση. Το ζητούμενο στις εφαρμογές αυτές είναι να βρίσκονται στην ίδια ψηφίδα με το μικροεπεξεργαστή και οι περιφερειακές μονάδες. Τα ολοκληρωμένα κυκλώματα που περιλαμβάνουν στην ίδια ψηφίδα εκτός από το Μικροεπεξεργαστή και θύρες για είσοδο – έξοδο δεδομένων (σε παράλληλη και σειριακή μορφή), μετατροπείς αναλογικών σημάτων σε ψηφιακή μορφή και αντίστροφα (A/D και D/A), χρονιστές και μνήμες (ROM και RAM) ονομάζονται Μικροελεγκτές (Microcontrollers). Ο Μικροελεγκτής είναι υπεύθυνος για την είσοδο και έξοδο, την επεξεργασία, αποθήκευση και μετάδοση των αναλογικών και ψηφιακών σημάτων μιας εφαρμογής.

Για να κατανοήσουμε πόσο σημαντικός είναι ο μικροελεγκτής μέσα σε ένα μιροϋπολογιστικό σύστημα, αρκεί να εστιάσουμε την προσοσχή μας σε μια περίπτωση όπου δεν έχουμε στη διάθεσή μας μια διάταξη μικροελεγκτή. Έστω λοιπόν ότι διαθέτουμε μόνο μία απλή μονάδα κεντρικής επεξεργασίας (CPU). Για να κατασκευάσουμε ένα σύστημα ικανό να συνδεθεί με διάφορες εξωτερικές διατάξεις (κινητήρες, μονάδες απεικόνισης και άλλες), θα χρειαστούμε εξωτερική μνήμη προγράμματος και μνήμη δεδομένων ή RAM, εκτός των άλλων περιφερειακών διατάξεων που απαιτούνται για τη διασύνδεση της CPU με τις εξωτερικές μονάδες, δηλαδή τους κινητήρες, τις μονάδες απεικόνισης, τους αισθητήρες κ.λ.π. Κάτι τέτοιο θα ήταν εντελώς εξωπραγματικό και ο αριθμός των απαιτούμενων εξαρτημάτων για μια τέτοια υλοποίηση θα ήταν τεράστιος.

Όταν όλες οι διακριτές μονάδες ενός υπολογιστή που αποτελούν έναν μικροϋπολογιστή τοποθετηθούν μέσα στο σώμα του ίδιου ολοκληρωμένου κυκλώματος, η διάταξη που προκύπτει καλείται μικροελεγκτής. Αποτελείται από τη CPU, μνήμες RAM και ROM και μονάδα διασύνδεσης περιφερειακών, ενώ χρησιμοποιείται κυρίως για να ελέγχει συσκευές. Μοναδικό μειονέκτημα που φαίνεται να εμφανίζει είναι το γεγονός ότι για να γράψει κάποιος ένα πρόγραμμα χρειάζεται προσομοιωτή.

Αναλυτικά, τα πλεονεκτήματα των μικροελεγκτών είναι:

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

Ένας μικροελεγκτής ωστόσο εμφανίζει και ορισμένα μειονεκτήματα:

- Δεν αλλάζει το πρόγραμμά του, αφού είναι γραμμένο στη ROM.
- Είναι δύσκολος ο προγραμματισμός του.
- Έχει μεγάλο χρόνο ανάπτυξης. Για να ολοκληρωθεί ένα προϊόν μπορεί να απαιτηθεί από 1 εβδομάδα μέχρι 1 χρόνο.

1.2.1 Πρόσθετες Λειτουργίες Μικροελεγκτών

Ανάλογα με την εφαρμογή για την οποία προορίζεται ένας μικροελεγκτής, μπορεί να περιέχει και:

- Μία ή περισσότερες ασύγχρονες σειριακές θύρες επικοινωνίας (Universal Asynchronous Receiver Transmitter, UART).
- Σύγχρονες σειριακές θύρες επικοινωνίας (πχ I^2C, SPI, Ethernet).
- Ολόκληρα υποσυστήματα για την άμεση υποστήριξη από υλικολογισμικό (hardware) των πιο σύνθετων πρωτοκόλλων επικοινωνίας όπως CAN, HDLC, ISDN, ADSL.
- Μονάδα άμεσης εκτέλεσης πράξεων κινητής υποδιαστολής (Floating Point Processing Unit, FPU), η οποία είναι πάντοτε πιο γρήγορο από την ALU του επεξεργαστή. Τέτοιες μονάδες χαρακτηρίζουν τους μικροελεγκτές με δυνατότητες ψηφιακής επεξεργασίας σήματος (Digital Signal Processing, DSP). Τα τελευταία χρόνια, με την ευρύτατη διάδοση των φορητών συσκευών ήχου και εικόνας, παρατηρείται μια τάση σύγκλισης τωνν μικροελεγκτών με τους DSP.
- Περισσότερες από μία εισόδους για μετατροπή αναλογικού σήματος σε ψηφιακό (Analog to Digital converter, ADC).
- Μετατροπέα ψηφιακού σε αναλογικό σήμα (Digital to Analog converter, DAC).
- Ελεγκτή οθόνης υγρών κρυστάλλων (Liquid Crystal Display, LCD).
- Υποσύστημα προγραμματισμού πάνω στο κύκλωμα (τύπου ISP, βλ. παραπάνω). Χάρη σε αυτό το κύκλωμα, είναι δυνατός ο επαναπρογραμματισμός (αναβάθμιση λογισμικού) της εφαρμογής, συνδέοντας στη συσκευή μια εξωτερική συσκευή προγραμματισμού (συνήθως σε θύρα UART RS232) ή ακόμη και από το διαδίκτυο. Αυτή η δυνατότητα απαιτεί την προϋπαρξη λογισμικού υποδοχής (bootstrap) μέσα στη μνήμη προγράμματος και επομένως δεν μπορεί να γίνει σε τελείως άδεια μνήμη προγράμματος.
- Υποσύστημα προγραμματισμού (τύπου ISP) και διάγνωσης (συνήθως είναι το καθιερωμένο πρότυπο JTAG). Χάρη σε αυτό, είναι δυνατός ο προγραμματισμός της μνήμης προγράμματος χωρίς να προαπαιτείται κάποιο πρόγραμμα υποδοχής. Γι αυτό το λόγο, είναι ιδιαίτερα χρήσιμο στον αρχικό προγραμματισμό, πχ κατά τη συναρμολόγηση, ή σε περίπτωση προβλήματος (bug) στο λογισμικό υποδοχής το οποίο να καθιστά αδύνατη την κανονική αναβάθμιση.

1.2.2 Διαδεδομένες κατηγορίες μικροελεγκτών

Λόγω του ισχυρότατου ανταγωνισμού αλλά και της τάσης ενσωμάτωσης των μικροελεγκτών σε κάθε ηλεκτρική και ηλεκτρονική συσκευή, η βιομηχανία μικροελεγκτών έχει καταλήξει στην παραγωγή ανταγωνιστικών μοντέλων μαζικής παραγωγής καθώς και μικροελεγκτών για πιο εξειδικευμένες εφαρμογές. Έτσι διακρίνονται οι εξής κυρίως κατηγορίες:

- Μικροελεγκτές (καμιά φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης, με πολύ μικρό αριθμό ακροδεκτών (ακόμη και λιγότερους από 8). Σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια, ώστε να χρειάζονται ελάχιστα ή και καθόλου εξωτερικά εξαρτήματα και να μη μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς, όπως πχ οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και 8051 (Intel, Atmel, Dallas κα).
- Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης, με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό κοινών περιφερειακών, όπως θύρες UART, I^2C, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους.
- Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης, με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο αυτό έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον ένα στον άλλο κατασκευαστή. Πχ μεταξύ των μικροελεγκτών τύπου ARM ή MIPS, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό, όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου κατασκευαστή (αρκεί, φυσικά, να υποστηρίζει κι αυτός το σύνολο εντολών ARM ή MIPS, αντίστοιχα).
- Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως Modem.

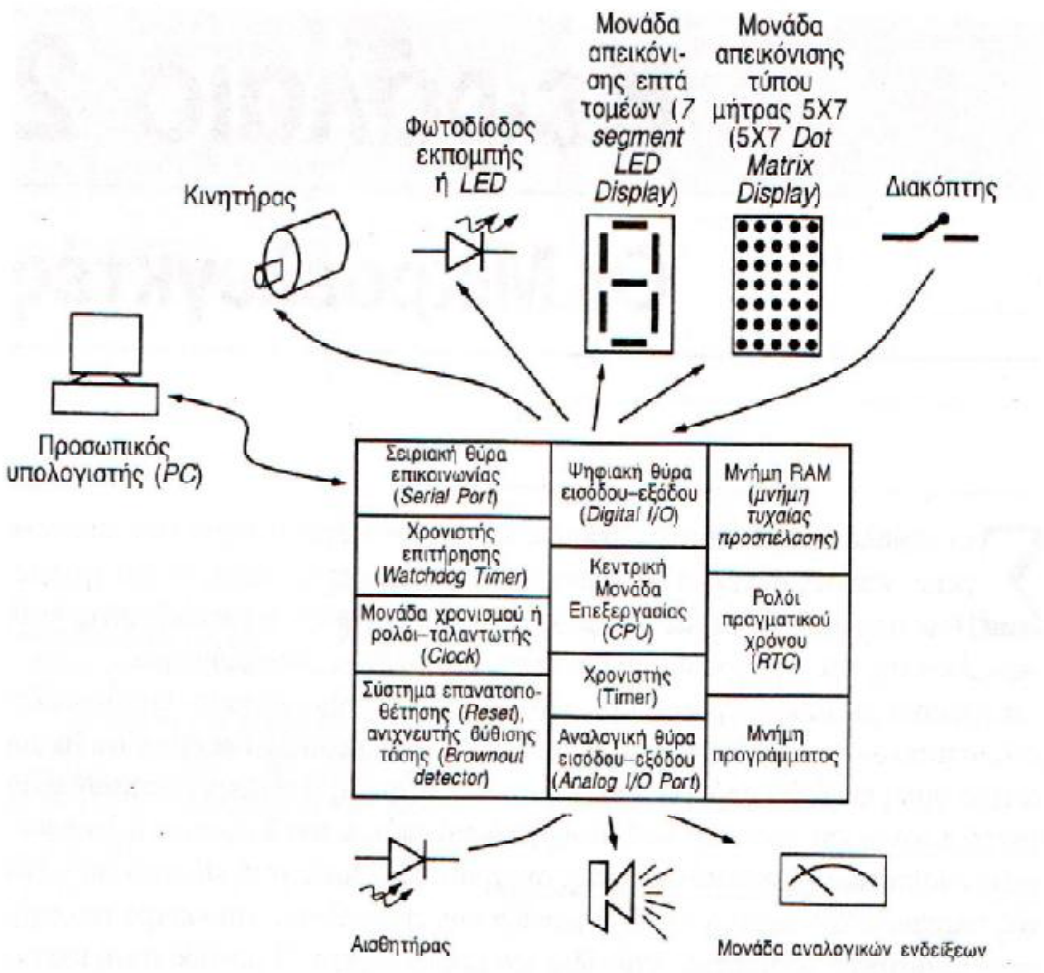
Η μεγάλη μερίδα πωλήσεων των μικροελεγκτών εξακολουθεί να αφορά αυτούς των 8 bit, καθώς είναι η κατηγορία με το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού για το

ίδιο αποτέλεσμα, ιδίως επειδή οι σύγχρονες οικογένειες οκτάμπιτων μικροελεγκτών έχουν πολύ βελτιωμένες επιδόσεις σε σχέση με το παρελθόν.

Η επιτυχία μιας οικογένειας μικροελεγκτών καθορίζεται σε μεγάλο βαθμό από τη διαθεσιμότητα και την ευχρηστία των σχετικών εργαλείων ανάπτυξης, όπως μεταφραστές από γλώσσες υψηλού επιπέδου σε γλώσσα κατανοητή από τον μικροελεγκτή (assembly), προγραμματιστές της εσωτερικής μνήμης και εργαλεία εκσφαλμάτωσης (debuggers). Στους μικροελεγκτές, τα εργαλεία αυτά δεν αποτελούνται ποτέ μόνο λογισμικό, καθώς δεν υπάρχει τυποποιημένος τρόπος επικοινωνίας με αυτούς. Στον τομέα των εργαλείων ανάπτυξης, δραστηριοποιούνται όχι μόνο οι ίδιοι οι κατασκευαστές μικροελεγκτών αλλά και εξειδικευμένες εταιρείες.

Η πιο διαδεδομένη γλώσσα προγραμματισμού των μικροελεγκτών είναι η C, η C++ και οι παραλλαγές τους. Σε τμήματα του λογισμικού όπου απαιτείται ταχύτητα η μικρό μέγεθος χρησιμοποιούμενης μνήμης, μπορεί να χρησιμοποιείται η Assembly. Όμως οι μεγαλύτερες απαιτήσεις σε λειτουργικότητα και η ευκολία προγραμματισμού της C έναντι της assembly, σε συνδυασμό με την επάρκεια μνήμης των σύγχρονων μικροελεγκτών, έχουν γενικά εκτοπίσει την Assembly από τις περισσότερες εφαρμογές.

Η εργασία με τους μικροελεγκτές είναι ουσιαστικά διασκέδαση, καθώς από σχεδιαστική άποψη, η χρήση τους είναι αρκετά εύκολη και προσιτή. Στο σχήμα που ακολουθεί εμφανίζονται οι δυνατότητες ενός μικροελεγκτή και συγκεκριμένα ενός τυπικού μικροελεγκτή AVR. Η μονάδα που εμφανίζεται στο κέντρο του σχήματος αντιστοιχεί ουσιαστικά στον ίδιο το μικροελεγκτή. Η μονάδα αυτή μπορεί να συνδεθεί κατάλληλα με διάφορα είδη κινητήρων, με μια ποικιλία διατάξεων απεικόνισης (Display) ως συσκευών εξόδου, να επικοινωνήσει με έναν προσωπικό υπολογιστή (PC), να διαβάσει τιμές από εξωτερικούς αισθητήρες, ενώ ακόμη μπορεί να συνδεθεί και σε ένα είδος τοπικού δικτύου άλλων παρόμοιων μικροελεγκτών και όλες αυτές οι δυνατότητες δεν απαιτούν ιδιαίτερα μεγάλο αριθμό εξωτερικών εξαρτημάτων. Το γεγονός αυτό οδηγεί σε πιο συμπαγή και για το λόγο αυτό πιο αξιόπιστα συστήματα, σε σχετικά χαμηλό κόστος (εξαιτίας του μικρού αριθμού των εξωτερικών εξαρτημάτων και των λιγότερων γενικά συνδέσεων που απαιτούνται για το λόγο αυτό).



Σχήμα 1.4: Απεικόνιση ποικίλων δυνατοτήτων του μικροελεγκτή να συνδεθεί με διάφορες εξωτερικές ως προς αυτόν διατάξεις, με τον ελάχιστο απαιτούμενο αριθμό πλέον εξαρτημάτων.

Στο σημείο αυτό θα παρουσιάσουμε συνοπτικά τα διάφορα συστατικά ενός μικροελεγκτή:

Ä Μονάδα Κεντρικής Επεξεργασίας – CPU (Central Processing Unit): αποτελεί την «καρδιά» ενός μικροελεγκτή αφού εκτελεί την ανάκληση δεδομένων (fetch) από τη μνήμη προγράμματος υπό μορφή εντολών, ενώ παράλληλα αποκωδικοποιεί τις εντολές και στη συνέχεια τις εκτελεί. Η μονάδα CPU, αποτελείται από καταχωρητές (registers), την αριθμητική λογική μονάδα (ALU – Arithmetic Logic Unit), τον αποκωδικοποιητή εντολών (Instruction Decoder) κι διάφορα κυκλώματα ελέγχου.

Ä Μνήμη Προγράμματος: σε αυτήν αποθηκεύονται οι εντολές που σχηματίζουν τον κορμό του προγράμματος. Διακρίνεται στην εσωτερική και την εξωτερική μνήμη προγράμματος.

Ä Μνήμη RAM: η μνήμη τυχαίας προσπέλασης αποτελεί τη μνήμη δεδομένων του ελεγκτή, γεγονός το οποίο σημαίνει ότι χρησιμοποιείται από τον ελεγκτή για την αποθήκευση

δεδομένων. Η CPU, χρησιμοποιεί τη μνήμη RAM για την αποθήκευση μεταβλητών, καθώς επίσης και τη λεγόμενη Στοίβα (Stack). Η Στοίβα με τη σειρά της χρησιμοποιείται από τη CPU, για την προσωρινή αποθήκευση των λεγόμενων διευθύνσεων επιστροφής, με σκοπό να συνεχίσει την εκτέλεση ενός προγράμματος το οποίο έχει διακοπεί.

Ä Ταλαντωτής Χρονισμού: έναπρόγραμμα εκτελείται από το μικροελεγκτή με έναν καθορισμένο ρυθμό. Ο ρυθμός αυτός καθορίζεται από τη συχνότητα λειτουργίας του ταλαντωτή χρονισμού, ο οποίος μπορεί να είναι ένας εσωτερικός ταλαντωτής τύπου RC ή ένας ταλαντωτής που υλοποιείται με κάποιο εξωτερικό στοιχείο χρονισμού, όπως για παράδειγμα ένας κρύσταλλος χαλαζία, ένα κύκλωμα συντονισμού LC ή ακόμη και ένα απλό κύκλωμα RC. Η λειτουργία του ταλαντωτή ξεκινά αμέσως μετά την εφαρμογή της τάσης τροφοδοσίας.

Ä Σύστημα επανατοποθέτησης και Κύκλωμα ανίχνευσης βυθίσεων τάσης: το σύστημα επανατοποθέτησης ή μηδενισμού ή απλά Reset, εξασφαλίζει το γεγονός ότι όλες οι εσωτερικές μονάδες και τα κυκλώματα ελέγχου του μικροελεγκτή θα ξεκινήσουν να λειτουργούν κατά την εφαρμογή της τροφοδοσίας, από κάποια προκαθορισμένη αρχική κατάσταση, ενώ όλοι οι καταχωρητές του συστήματος βρίσκονται σε κατάλληλες αρχικές τιμές. Το Κύκλωμα ανίχνευσης βύθισης της τάσης τροφοδοσίας είναι ένα επίσης εσωτερικό κύκλωμα ελέγχου το οποίο παρακολουθεί συνεχώς το επίπεδο της τάσης τροφοδοσίας και εφόσον ανιχνευτεί κάποια στιγμιαία βύθιση στην τάση αυτή τότε αυτόματα θέτει τον μικροελεγκτή σε λειτουργία επανατοποθέτησης, έτσι ώστε να προστατευθούν τα πειρεχομένα των καταχωρητών και της μνήμης από πιθανή καταστροφή ή αλλοίωση, πράγμα που θα οδηγούσε τον μικροελεγκτή σε εσφαλμένη λειτουργία.

Ä Σειριακή Θύρα Επικοινωνίας: εξυπηρετεί διαμεσολαβητικό σκοπό καθώς η λειτουργία της βασίζεται στο ότι λαμβάνει δεδομένα από το μικροελεγκτή τα οποία ολισθαίνει προς την έξοδο υπό μορφή ενός δυαδικού στοιχείου bit τη φορά. Παρομοίως, λαμβάνει δεδομένα από την αντίστοιχη είσοδό της και πάλι με τη μορφή ενός bit τη φορά, σχηματίζοντας έτσι με 8 τέτοια bits μια λέξη του 1 byte, την οποία και αντιγράφει στο εσωτερικό του ελεγκτή. Μπορεί να λειτουργήσει σε οποιαδήποτε ταχύτητα μετάδοσης δεδομένων τυχόν απαιτηθεί.

Ä Ψηφιακή Θύρα Εισόδου – Εξόδου: χρησιμοποιούνται για την ανταλλαγή δεδομένων από και προς το εξωτερικό περιβάλλον. Τα δεδομένα ανταλλάσσονται υπό τη μορφή ομάδων των 8 bits ή του 1 byte.

Ä Αναλογική θύρα εισόδου – εξόδου: η ύπαρξη αναλογικών εισόδων προϋποθέτει Μετατροπείς Αναλογικού Σήματος σε Ψηφιακό. Ένας μικροελεγκτής είναι δυνατόν να διαθέτει μία ενσωματωμένη μονάδα μετατροπής, η οποία χρησιμοποιείται για την ανάγνωση δεδομένων από αισθητήρες, όπως για παράδειγμα αισθητήρες πίεσης και θερμοκρασίας. Παράλληλα,

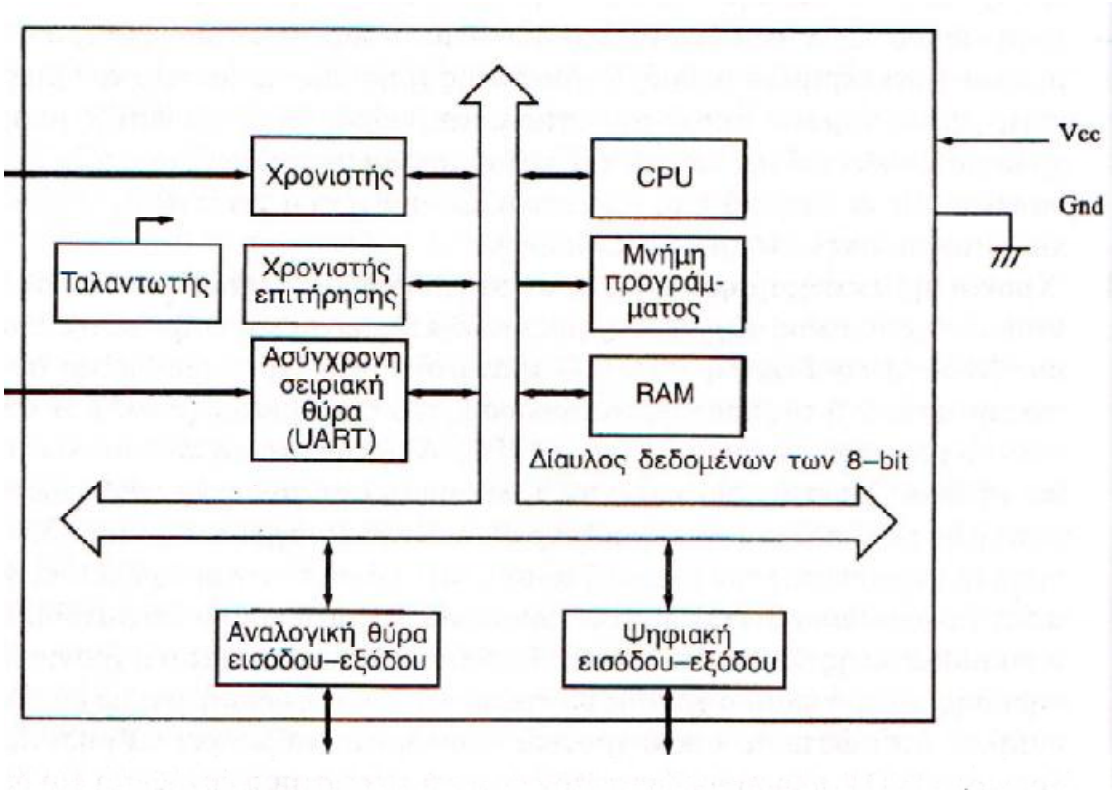
μπορούμε να έχουμε και αναλογικές εξόδους, χρησιμοποιώντας κάποιες μονάδες οι οποίες καλούνται Μετατροπείς Ψηφιακού Σήματος σε Αναλογικό. Οι μετατροπείς αυτοί χρησιμοποιούνται για την οδήγηση κινητήρων, ειδικών μονάδων απεικόνισης, για την αναπαραγωγή σημάτων ήχου ή μουσικής και αλλού.

Ä Χρονιστής (Timer)χρησιμοποιείαι για το χρονισμό ή τη σηματοδότηση διαφόρων γεγονότων, όπως για παράδειγμα για να παράγει το ζητούμενο ρυθμό που απαιτείται προκειμένου να καταφέρουμε να στείλουμε επιτυχώς δεδομένα σε μια εξωτερική οθόνη. Μπορεί να χρησιμοποιηθεί επίσης και για την καταμέτρηση γεγονότων, τα οποία μπορούν να είναι είτε εσωτερικά είτε εξωτερικά.

Ä Χρονιστής Επιτήρησης (WatchDog Timer): χρησιμοποιείται προκειμένου να αποφευχθεί μια πιθανή κατάρρευση του συστήματος (crash), οπότε και ενεργοποιείται η λειτουργία ενός αυξανόμενα εσωτερικού μετρητή σε συγκεκριμένο ρυθμό. Αν το πρόγραμμα χρήσης δε μηδενίσει (ή επαναθέσει το μετρητή αυτό) τότε κάποια στιγμή θα επέλθει η λεγόμενη υπερχείλιση του παραπάνω μετρητή και θα επανατοποθετηθεί ο ελεγκτής σε λειτουργία reset. Η λογικής αυτής της τεχνικής ελέγχου στηρίζεται στην υπόθεση ότι αν το πρόγραμμα χρήσης δε μηδενίσει το χρονιστή επιτήρησης αυτό πιθανότατα σημαίνει ότι το πρόγραμμα έχει αποτύχει σε κάποια προσπάθειά του είτε εξαιτίας πιθανής κατάρρευσης ή γενικότερα κάποιας απρόβλεπτης συμπεριφοράς οπότε είναι προτιμότερο να εκκινήσει διαδικασία επανατοποθέτησης.

Ä Ρολόι Πραγματικού Χρόνου: ο σκοπός του είναι η μέτρηση και η διατήρηση της τρέχουσας ώρας της ημέρας, της ημερομηνίας κ.λ.π. Μπορεί να χρησιμοποιηθεί για τη σηματοδότηση συγκεκριμένων γεγονότων με γνώμονα την τρέχουσα ώρα.

Μια τυπική μορφή μικροελεγκτή είναι αυτή που παρουσιάζεται στο σχήμα που ακολουθεί. Από τις διάφορες κατηγορίες μικροελεγκτών, εκείνη που γενικά έχει τη μεγαλύτερη απήχηση στη διεθνή αγορά είναι η κατηγορία των 8 bit, τόσο εξαιτίας του χαμηλού τους κόστους, όσο και λόγω της διαθεσιμότητας των διατάξεων αυτών σε μια μεγάλη ποικιλία, από την άποψη των επιδόσεων και των διαθέσιμων ολοκληρωμένων περιφερειακών μονάδων.



Σχήμα 1.5: Ένας μικροελεγκτής των 8 bit.

1.2.3 Κατασκευαστές Μικροελεγκτών

Μερικοί από τους γνωστότερους κατακευαστές μικροελεγκτών είναι οι εξής:

- ARM (δεν κατασκευάζει αλλά παραχωρεί δικαιώματα χρήσης του πυρήνα της)
- Atmel
- Epson
- Freescale Semiconductor (πρώην Motorola)
- Hitachi
- Maxim (μετά την εξαγορά της Dallas)
- Microchip
- NEC
- Toshiba
- Texas Instruments

1.3 Σύγκριση Μικροεπεξεργαστή – Μικροελεγκτή

Αναμφίβολα, τόσο οι μικροεπεξεργαστές όσο και οι Μικροελεγκτές, αποτελούν απαραίτητο και αναπόσπαστο κομμάτι των σύγχρονων τεχνολογικών εφαρμογών. Εντούτοις, οι δύο αυτές συσκευές διαφέρουν μεταξύ τους σε αρκετά σημεία. Μία ουσιώδης διαφορά έγκειται στη λειτουργικότητά τους. Για να λειτουργήσει ένας μικροεπεξεργαστής, θα πρέπει να συνδεθεί με άλλες συσκευές, όπως η μνήμη (memory) ή μια συσκευή αποστολής και λήψης δεδομένων. Συνεπώς, αποτελεί την καρδιά του συστήματος.

Από την άλλη πλευρά, ο μικροελεγκτής είναι σχεδιασμένος έτσι που να εμπεριέχει όλες τις παραπάνω συσκευές και να μην απαιτείται η παρουσία άλλων για τη λειτουργία του, αφού όλα τα απαραίτητα περιφερειακά (peripherals) είναι εξαρχής ενσωματωμένα στη δομή του. Επομένως, με την παρουσία του μικροελεγκτή, εξοικονομείται χώρος και χρόνος κατά την κατασκευή ενός συστήματος που βασίζεται σε αυτό.

Επιπρόσθετα, μία ακόμη διαφόρα έγκειται στην υπολογιστική ισχύ την οποία διαθέτουν. Στους σύγχρονους μικροεπεξεργαστές για μη ενσωματωμένα συστήματα (πχ τους μικροεπεξεργαστές των προσωπικών υπολογιστών), δίνεται έμφαση στην υπολογιστική ισχύ. Η ευελιξία ανάπτυξης διαφορετικών εφαρμογών είναι μεγάλη, καθώς η λειτουργικότητα του τελικού συστήματος καθορίζεται από τα εξωτερικά περιφερειακά τα οποία διασυνδέονται με την κεντρική μονάδα (μικροεπεξεργαστή), η οποία δεν είναι εξειδικευμένη. Αντίθετα, στους μικροεπεξεργαστές για ενσωματωμένα συστήματα (μικροελεγκτές), οι οποίοι έχουν μικρότερες ή και μηδαμινές δυνατότητες συνεργασίας με εξωτερικά περιφερειακά, αυτού του είδους η ευελιξία είναι περιορισμένη, καθώς και η υπολογιστική ισχύς. Οι μικροελεγκτές δίνουν έμφαση στο μικρό αριθμό ολοκληρωμένων κυκλωμάτων που απαιτείται για τη λειτουργία μιας συσκευής, το χαμηλό κόστος και την εξειδίκευση.

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

Μία ακόμη διαφορά σχετίζεται με τα συνήθη υποσυστήματα, τα οποία πλαισιώνουν έναν μικροεπεξεργαστή και έναν μικροελεγκτή. Στον μικροεπεξεργαστή, το ολοκληρωμένο κύκλωμα που τον αποτελεί περιέχει μόνο την *Λογική και Αριθμητική Μονάδα* (ALU), στοιχειώδεις καταχωρητές (registers), προσωρινή μνήμη RAM πολύ υψηλής ταχύτητας (cache memory) και, κάποιες φορές, τον ελεγκτή μνήμης (memory controller). 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 προστατευτεί το περιεχόμενό της από αντιγραφή.
- Μεγάλη ποσότητα μνήμης RAM.
- Μόνιμη μνήμη αποθήκευσης παραμέτρων λειτουργίας (τύπου EEPROM ή NVRAM) η οποία να μπορεί να γράφεται τον πυρήνα του μικροελεγκτή. Αυτή η μνήμη έχει, έναντι της FLASH, το πλεονέκτημα της δυνατότητας διαγραφής και εγγραφής οποιουδήποτε μεμονωμένου byte.
- Κύκλωμα αρχικοποίησης (reset).
- Διαχειριστή αιτήσεων διακοπής (interrupt request controller) από τα περιφερειακά.
- Κύκλωμα επιτήρησης τροφοδοσίας (brown-out detection) το οποία παρακολουθεί την τροφοδοσία και αρχικοποιεί ολόκληρο το σύστημα όταν αυτή πέσει κάτω από τα ανεκτα όρια, προλαμβάνοντας έτσι την αλλοίωση των δεδομένων.
- 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).
- Έναν ή περισσότερους χρονιστές-απαριθμητές υψηλής ταχύτητας (hardware timer-counter) για τη δημιουργία καθυστερήσεων, μέτρηση διάρκειας γεγονότων, απαρίθμηση γεγονότων και άλλων λειτουργιών ακριβούς χρονισμού.
- Ρολόι πραγματικού χρόνου (Real Time Clock, RTC) το οποίο τροφοδοτείται από ανεξάρτητη μπαταρία και γι αυτό πρέπει να έχει πολύ χαμηλή κατανάλωση ρεύματος.
- Σειρά ανεξάρτητων ψηφιακών εισόδων και εξόδων (Parallel Input-Output, PIO).

Γενικά, όλες οι οικογένειες μικροελεγκτών ενσωματώνουν τα περισσότερα από τα παραπάνω περιφερειακά και έτσι δεν απαιτούνται εξωτερικά υποσυστήματα. Οι διαφοροποιήσεις εντοπίζονται κυρίως στην ύπαρξη ή μη εσωτερικής μνήμης προγράμματος και στο είδος της. Έτσι, υπάρχουν:

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

ΚΕΦΑΛΑΙΟ 2^{Ο} - Η ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ AVR

2.1 ΚΑΤΗΓΟΡΙΕΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΜΙΚΡΟΕΛΕΓΚΤΩΝ

Όπως είδαμε στο προηγούμενο κεφάλαιο, κάθε μικροϋπολογιστικό σύστημα διακρίνεται για τα ιδιαίτερα χαρακτηριστικά και τις ιδιότητες τις οποίες παρουσιάζει και για το λόγο αυτό οι αρχιτεκτονικές των μικροελεγκτών ποικίλουν. Διακρίνουμε τις εξής κατηγορίες αρχιτεκτονικών σχημάτων:

1. Αυτή που λαμβάνει υπόψη το συνολικό αριθμό των εντολών. Έτσι έχουμε την αρχιτεκτονική CISC (Complex Instruction Set Computer ή Αρχιτεκτονική Σύνθετου Ρεπερτορίου Εντολών), την αρχιτεκτονική RISC (Reduced Instruction Set Computer ή Αρχιτεκτονική Μειωμένου Ρεπερτορίου Εντολών) και την αρχιτεκτονική MISC (Minimum Instruction Set Computer ή Αρχιτεκτονική Ελαχίστου Ρεπερτορίου Εντολών). Ωστόσο, οι παραπάνω όροι έχουν υποστεί πολλές παραφράσεις από τους διάφορους ειδικούς προγραμματιστές και πωλητές του είδους, ενώ συχνά προκαλείται σύγχυση καθώς συμβαίνει δύο μικροελεγκτές να παρουσιάζουν πολλές κοινές ιδιότητες (για παράδειγμα ο CISC και ο RISC).

2. Αυτή που αφορά στον τρόπο με τον οποίο πραγματοποιείται η πρόσβαση στη μνήμη προγράμματος και τη μνήμη δεδομένων. Ένα τέτοιο μοναδικό μοντέλο μνήμης, το οποίο είναι γνωστό με την ονομασία Αρχιτεκτονική Princeton ή Αρχιτεκτονική Von Neumann και το οποίο σε αντίθεση με την Αρχιτεκτονική Harvard, προβλέπει ξεχωριστό χώρο μνήμης για την αποθήκευση προγράμματος και για την αποθήκευση δεδομένων αντίστοιχα.

3. Με βάση τον τρόπο αποθήκευσης και διαχείρισης που υφίστανται τα δεδομένα εντός της CPU. Αντικειμενικός σκοπός ενός συστήματος μικροελεγκτή είναι η διαχείριση δεδομένων. Την εργασία αυτή την επιτυγχάνει με τη βοήθεια του προγράμματος χρήσης. Ο τρόπος λοιπόν διαχείρισης και αποθήκευσης δεδομένων εντός της CPU, καθώς επίσης και οι διάφοροι τρόποι προσπέλασης, διαμορφώνουν μία βάση με την οποία ταξινομούνται οι διάφορες αρχιτεκτονικές των επεξεργαστών και κατά συνέπεια ένα ακόμη διαφορετικό σχήμα ταξινόμησης.

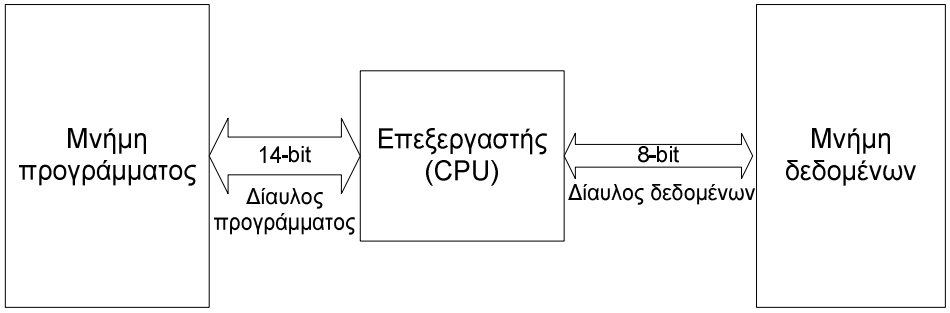
Σύμφωνα με τα παραπάνω διακρίνουμε τέσσερα βασικά πρότυπα ή μοντέλα:

1) Το μοντέλο Στοίβας (STACK),
2) Το μοντέλο Συσσωρευτή (ACCUMULATOR),
3) Το μοντέλο Καταχωρητής – Μνήμη (Register-Memory) και

4) Το μοντέλο πολλών Καταχωρητών (Register-Register) ή μοντέλο Φόρτωσης – Αποθήκευσης (Load-Store).

2.2 ΟΙ ΜΙΚΡΟΕΛΕΓΚΤΕΣ AVR

Από τα γενικά χαρακτηριστικά της αρχιτεκτονικής των μικροελεγκτών θα προχωρήσουμε στην ενότητα αυτή στην παρουσίαση της οικογένειας των μικροεπεξεργαστών AVR στο πεδίο της αρχιτεκτονικής τους. Στο επόμενο κεφάλαιο θα αναφερθούμε στις περιφερειακές μονάδες που πλαισιώνουν το σύστημα. Με βάση την αρχιτεκτονική Harvard, ο μικροελεγκτής AVR περιλαμβάνει έναν επεξεργαστή RISC (Reduced Instruction Set Computer) και η μονάδα CPU συνεργάζεται ταυτόχρονα με μία μνήμη προγράμματος (program memory) και μία ξεχωριστή μνήμη δεδομένων (data memory), γεγονός το οποίο καθιστά την αρχιτεκτονική Harvard αποδοτική αφού καθίσταται δυνατόν να εκτελείται κάποια εντολή και παράλληλα να εγγράφεται ή να διαβάζεται η μνήμη. Με αυτόν τον τρόπο επιτυγχάνεται η εκτέλεση της εντολής σε ένα μόνο χρόνο μηχανής. Όπως φαίνεται και στο σχήμα που ακολουθεί, υπάρχει διαφορετικός δίαυλος για τη μεταφορά δεδομένων (data bus) και διαφορετικός για τη μεταφορά εντολών (instruction bus).



Σχήμα 2.1: Σύνδεση του επεξεργαστή με τη μνήμη προγράμματος και δεδομένων.

Οι μικροελεγκτές AVR της εταιρίας ATMEL παρουσιάζουν τα ακόλουθα χαρακτηριστικά:

Ä Συνδυάζουν την αρχιτεκτονική RISC με ως επί τω πλείστον σταθερού μήκους εντολές, διαδικασίες αποθήκευσης – φόρτωσης στη μνήμη και 32 καταχωρητές γενικής χρήσης.

Ä Διαθέτουν μηχανισμό συνεχούς διοχέτευσης εντολών (instruction pipeline) σε δύο στάδια, που επιταχύνει σημαντικά τη διαδικασία εκτέλεσης.

Ä Οι περισσότερες από τις εντολές που περιλαμβάνει το ρεπερτόριό τους εκτελούνται στη διάρκεια μιας περιόδου του κεντρικού ρολογιού.

Ä Λειτουργούν σε συχνότητες χρονισμού έως 10 MHz.

Ä Διαθέτουν μεγάλη ποικιλία σε ό,τι αφορά ενσωματωμένες περιφερειακές μονάδες όπως, ψηφιακές εισόδους – εξόδους (I/O), μετατροπείς αναλογικού σήματος σε ψηφιακό ή ADC, μνήμη τύπου EEPROM, χρονιστές, μονάδες ασύγχρονης σειριακής επικοινωνίας ή UART (Universal Asynchronous Receiver Transmitter), ρολόγια πραγματικού χρόνου (RTC – Real Time Clock), μονάδες διαμόρφωσης εύρους παλμών (PWM – Pulse Width Modulation) και άλλα.

Ä Ενσωματωμένες μνήμες προγράμματος και δεδομένων.

Ä Δυνατότητα προγραμματισμού εντός του συστήματος (ISP – In-System Programmable).

Ä Διατίθενται σε συσκευασίες των 8 έως 64 ακροδεκτών οπότε κρίνονται κατάλληλοι για έναν μεγάλο αριθμό διαφορετικών εφαρμογών.

Ä Είναι περίπου 12 φορές ταχύτεροι και πιο αποδοτικοί σε σχέση με τους ελεγκτές κλασικής αρχιτεκτονικής CISC (Complex Instruction Set Computer).

Ä Ευρεία περιοχή τάσεων λειτουργίας από 2.7 V έως 6.0V.

Ä Η σχετικά απλή αρχιτεκτονική τους δίνει το πλεονέκτημα του μικρού σε απαίτηση κύκλου εκμάθησης στους αρχάριους.

Η οικογένεια των μικροελεγκτών AVR διαθέτει τα ακόλουθα:

1. Μία ενσωματωμένη μνήμη flash ή αλλιώς ταχείας αποθήκευσης, η οποία έχει τη δυνατότητα προγραμματισμού εντός του συστήματος (ISP, In System Programmable), ως μνήμη προγράμματος. Η δυνατότητα αυτή καθιστά περιττή την ύπαρξη εξωτερικών μνημών τύπου ROM ή EPROM οι οποίες να περιέχουν τον κώδικα του προγράμματος χρήσης.
2. 32 καταχωρητές εργασίας των 8 bit. Αυτό έχει ως αποτέλεσμα οι διάφορες μεταβλητές να μπορούν να αποθηκεύονται εντός της CPU και όχι σε κάποια μνήμη, όπου η διαδικασία πρόσβασης θεωρείται χρονοβόρα.
3. Μία ενσωματωμένη μνήμη δεδομένων (data memory), τύπου EEPROM και RAM στις περισσότερες διατάξεις της σειράς τους. Σε αυτές τις μνήμες αποθηκεύονται σταθερές τιμές και μεταβλητές.
4. Η λειτουργία τους γίνεται σε συχνότητες χρονισμού από 0 έως 10 MHz. οι περισσότερες εντολές της γλώσσας των μικροελεγκτών AVR, σε μία μόνο περίοδο του κεντρικού σήματος χρονισμού, γεγονός που οδηγεί σε μία βελτιωμένη κατά 10 φορές περίπου επίδοση σε σχέση με τους κλασικούς μικροελεγκτές που λειτουργούν στην ίδια συχνότητα χρονισμού.
5. Εσωτερικό κύκλωμα που επανατοποθετεί το σύστημα κατά την εφαρμογή της τάσης τροφοδοσίας (Power On Reset – POR).

6. Ενσωματωμένο προγραμματιζόμενο χρονιστή με μονάδα διαίρεσης συχνότητας (prescaler). Η μονάδα αυτή χρησιμοποιείται στις περιπτώσεις αυτές όπου απαιτούνται εφαρμογές κρίσιμου χρονισμού.

7. Περιλαμβάνει πηγές εσωτερικών και εξωτερικών διακοπών.

8. Προγραμματιζόμενο χρονιστή επιτήρησης (Watch Dog Timer - WDT). Ο ρόλος του είναι η αποφυγή εσφαλμένων λειτουργιών σε περίπτωση κάποιας πιθανής κατάρρευσης του προγράμματος χρήσης.

9. Λειτουργίες αποκοπής (Power Down) και ηρεμίας (Sleep). Όταν δεν υπάρχει δραστηριότητα ο μικροεπεξεργαστής τίθεται σε αυτές τις καταστάσεις προκειμένου να καταναλώνεται λιγότερη ισχύς.

10. Διάφορα μοντέλα της σειράς AVR διαθέτουν ενσωματωμένο ταλαντωτή τύπου RC, ο οποίος όταν ενεργοποιείται συμβάλει σημαντικά στη μείωση του συνολικού αριθμού των απαιτούμενων εξωτερικών εξαρτημάτων.

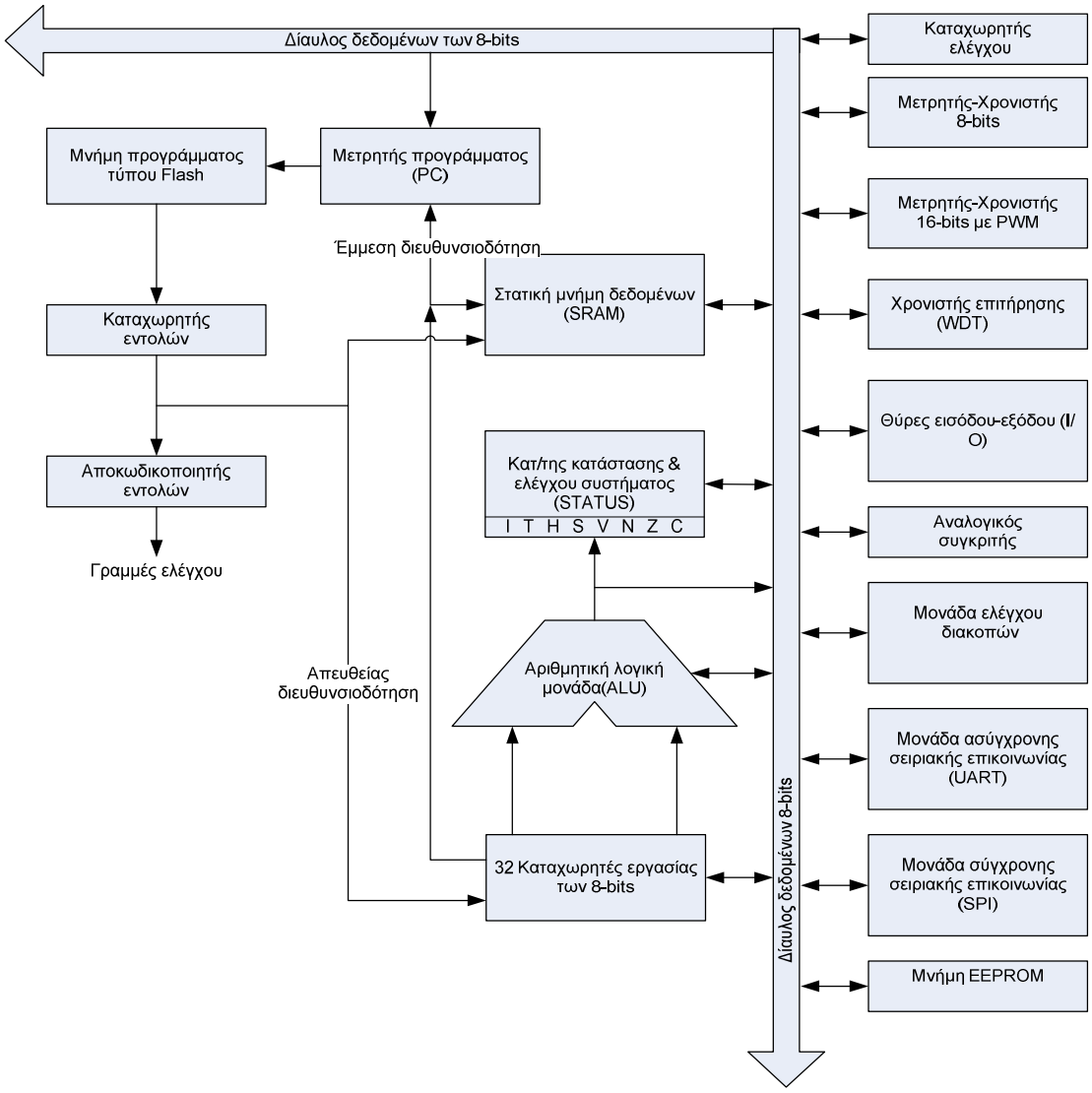
11. Τέλος, οι μικροελεγκτές αυτής της σειράς ποικίλουν (από τους 8 στους 68 ακροδέκτες), οπότε καθένας μπορεί να επιλέξει εκείνον που θεωρεί ως τον καταλληλότερο για την εργασία του, ενώ παρέχεται και μία πληθώρα περιφερειακών (UART, SPI, αναλογικός συγκριτής, μετατροπέας adc).

Στο σχήμα που ακολουθεί εμφανίζονται διάφορα μοντέλα μικροελεγκτών και τα βασικά χαρακτηριστικά τους.

Μοντέλο	Ακροδέκτες	Flash	EEPROM	RAM	UART	ADC
90S1200	20	1K	64 bytes	0	Όχι	Όχι
90S2313	20	2K	128	128	Ναι	Όχι
90S2323	8	2K	128	128	Όχι	Όχι
90S2333	28	2K	128	128	Ναι	Ναι
90S4433	28	4K	256	128	Ναι	Ναι
90S4414	40	4K	256	256	Ναι	Όχι
90S8515	40	8K	512	512	Ναι	Όχι
90S4434	40	4K	256	256	Ναι	Ναι
90S2343	8	2K	128	128	Όχι	Όχι

Mega103	64	128K	4096	4096	Ναι	Ναι
Mega603	64	64K	2048	4096	Ναι	Ναι
Tiny10	8	1K	64	0	Όχι	Όχι
Tiny12	8	1K	64	0	Όχι	Όχι
Tiny13	8	2K	128	128	Όχι	Όχι

Επιπλέον, στο ακόλουθο σχήμα μπορεί να δει κανείς την αρχιτεκτονική διάταξη των μικροεπεξεργαστών AVR και των λειτουργικών μονάδων τους. Το σημαντικό σε όλους αυτούς του τύπους μικροελεγκτών είναι ότι ο πυρήνας της αρχιτεκτονικής τους είναι ίδιος και έχουν το ίδιο σύνολο εντολών, διαφέρουν όμως ως προς τα περιφερειακά τους και το μέγεθος της μνήμης που παρέχουν. Όπως είναι σύνηθες σε αυτού του τύπου τους ελεγκτές, υπάρχουν διαφορετικοί εσωτερικοί δίαυλοι για τη μνήμη προγράμματος και τη μνήμη δεδομένων.

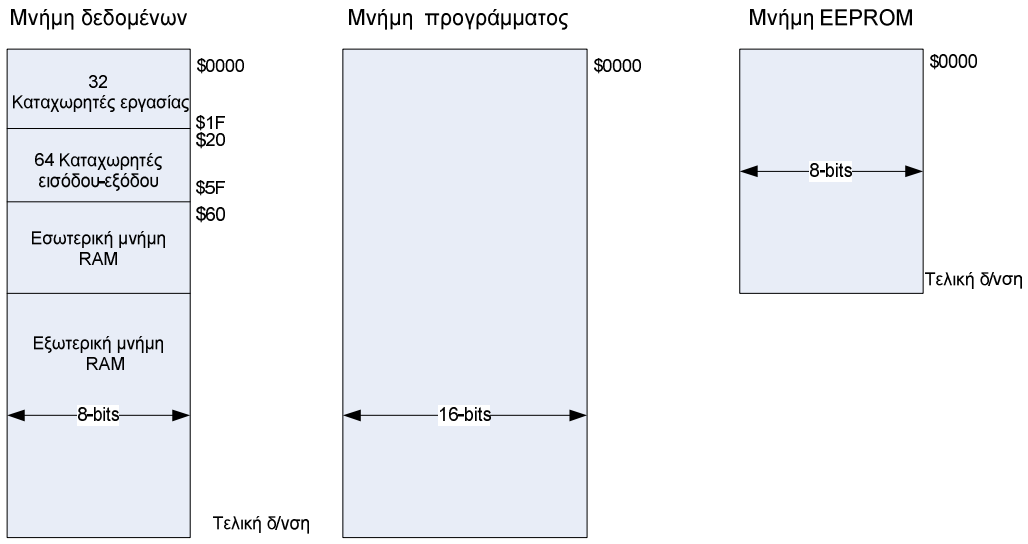


Σχήμα 2.2: Αρχιτεκτονική διατάξη μικροεπεξεργαστών AVR και περιφερειακών μονάδων τους.

2.3 Μνήμη δεδομένων και μνήμη προγράμματος

Ένας μικροελεγκτής διαθέτει τα εξής είδη μνήμης:

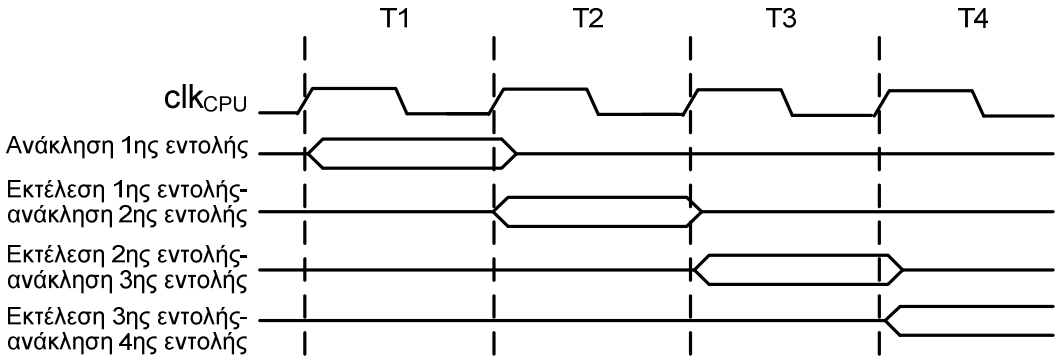
- μνήμη δεδομένων
- μνήμη προγράμματος τύπου flash
- μνήμη EEPROM με αρχική διεύθυνση $0000 και χωρητικότητα από 64bytes ως 4Kbytes.



Σχήμα 2.3 : Τύποι μνημών μικροελεγκτών AVR.

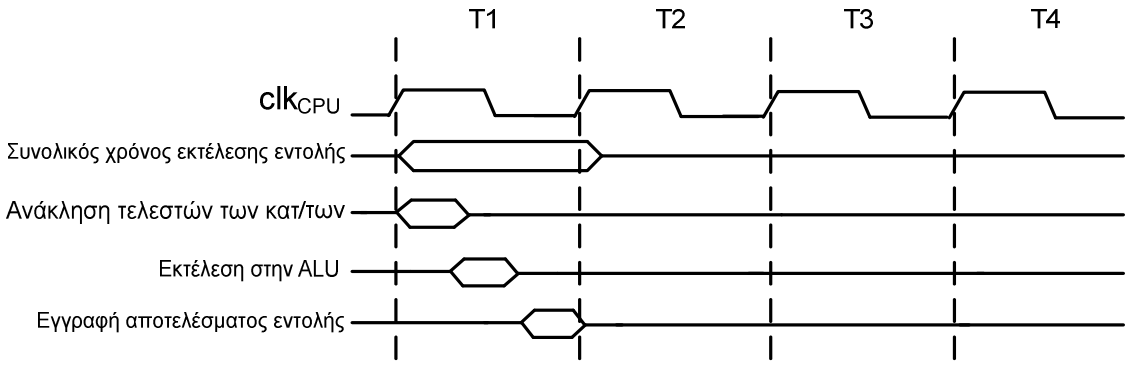
2.3.1 Μνήμη Προγράμματος

Η μνήμη προγράμματος αποτελεί έναν συνεχή χώρο μιας μνήμης τύπου flash, επιτυγχάνει ταχεία αποθήκευση-φόρτωση δεδομένων και συνδέεται με δίαυλο των 16-bit για την τροφοδοσία του καταχωρητή εντολών. Ωστόσο, η ακριβής χωρητικότητα της μνήμης αυτής διαφέρει από μικροελεγκτή σε μικροελεγκτή. Στην μνήμη προγράμματος αποθηκεύονται οι εντολές και τα διανύσματα διακοπών (interrupt vectors). Όλες οι εντολές έχουν μήκος 16 bits ή 32 bits (δηλαδή 2 bytes=1 λέξη), άρα καταλαμβάνουν μία ή δύο θέσεις της μνήμης προγράμματος, αφού η μνήμη flash, είναι οργανωμένη σε διάταξη 8K x 16. Η πρόσβαση στη μνήμη προγράμματος γίνεται σε μια περίοδο του σήματος χρονισμού. Επειδή συντελείται μία συνεχής διοχέτευση εντολών (instruction pipeline), η εκτέλεση των περισσότερων εντολών γίνεται σε μια περίοδο του κεντρικού ρολογιού. Αυτό σημαίνει ότι κατά την εκτέλεση μιας εντολής γίνεται ταυτόχρονα ανάκληση από τη μνήμη της επόμενης εντολής μέσω μιας παράλληλης διαδικασίας, όπως είναι προφανές στο ακόλουθο σχήμα. Σε αυτό απεικονίζεται η παράλληλη διαδικασία ανάκλησης (από τη μνήμη προγράμματος) και εκτέλεσης εντολών (με τη χρήση της μνήμης δεδομένων), η οποία καθίσταται δυνατή χάρη στην αρχιτεκτονική Harvard και το ταχείας πρόσβασης Register File.



Σχήμα 2.4: Παράλληλη ανάκληση και εκτέλεση εντολών

Κάθε εντολή η οποία ανακαλείται, φορτώνεται στον καταχωρητή εντολών, ο οποίος με τη σειρά του συνδέεται με την ομάδα των καταχωρητών εργασίας και επιλέγει τους κατάλληλους καταχωρητές προκειμένου να χρησιμοποιηθούν από την αριθμητική μονάδα ALU για την εκτέλεση της συγκεκριμένης εντολής. Επιπρόσθετα, η έξοδος του καταχωρητή εντολών αποκωδικοποιείται μέσα από τη μονάδα του αποκωδικοποιητή εντολών, όπου προκύπτουν τα απαραίτητα σήματα ελέγχου για την ολοκλήρωση της εκτέλεσης της τρέχουσας εντολής. Η διάρκεια της μνήμης είναι τουλάχιστον 1000 κύκλοι εγγραφής / διαγραφής. Προκειμένου να προστατεύεται το λογισμικό, η μνήμη χωρίζεται στο τμήμα εκκίνησης και το τμήμα εφαρμογών, τα οποία διαθέτουν bits κλειδώματος για ασφάλεια εγγραφής και ανάγνωσης / εγγραφής.



Σχήμα 2.5: Διαδικασία εκτέλεσης εντολής στην ALU

2.3.2 Μνήμη Δεδομένων

Η μνήμη δεδομένων του συστήματος διαιρείται σε πολλούς διαφορετικούς τύπους και συνδέεται με έναν διάδρομο των 8 bit για την επικοινωνία των περιφερειακών μονάδων με τους καταχωρητές ελέγχου. Χωρίζεται στα ακόλουθα πέντε τμήματα:

1. Ένα τμήμα που αντιστοιχεί στην ομάδα καταχωρητών εργασίας (register file) των 8 bits. Η ομάδα αυτή των καταχωρητών υπάρχει σε όλους τους μικροελεγκτές της οικογένειας των AVR.
2. Ένα τμήμα 64 καταχωρητών εισόδου – εξόδου των 8 bits. Δε διαθέτουν όλοι οι μικροελεγκτές της σειράς AVR και τους 64 παραπάνω καταχωρητές. Ανάλογα με το πλήθος των εσωτερικών περιφερειακών μονάδων κάποιοι διαθέτουν περισσότερους.
3. Ένα τμήμα εσωτερικής στατικής μνήμης SRAM.
4. Ένα τμήμα εξωτερικής στατικής μνήμης SRAM.
5. Ένα τμήμα εσωτερικής μνήμης τύπου EEPROM. Ο τύπος αυτός εσωτερικής μνήμης είναι διαθέσιμος σχεδόν σε όλους τους μικροελεγκτές AVR και η πρόσβασή του επιτυγχάνεται σε ιδιαίτερο χώρο διευθύνσεων μνήμης. Η μνήμη αυτή μπορεί να διαβαστεί και να γραφεί από οποιοδήποτε πρόγραμμα.

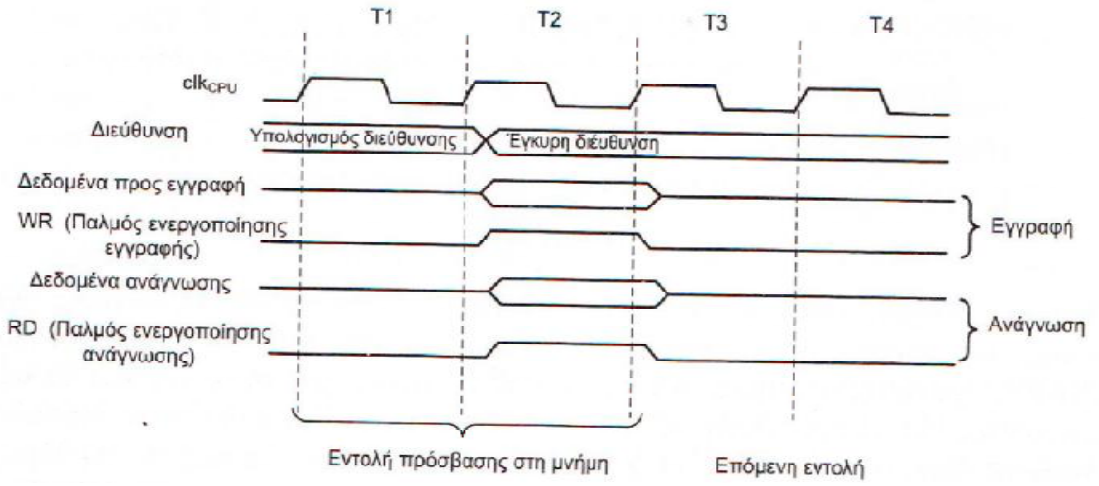
2.4 Μνήμη SRAM

Η στατική RAM (Static RAM - SRAM) είναι ένας τύπος μνήμης RAM που έχει την ικανότητα να διατηρεί αναλλοίωτα τα περιεχόμενά της για όσο χρονικό διάστημα τροφοδοτείται με ρεύμα, χωρίς να απαιτείται κάποια επιπλέον εξωτερική επέμβαση. Στις SRAMs χρησιμοποιούνται ειδικοί διακόπτες (switches), που μπορεί να είναι ανοικτοί ή κλειστοί (on/off). Ο τρόπος κατασκευής της SRAM μοιάζει περισσότερο με την τεχνολογία που εφαρμόζεται στους επεξεργαστές: πολλά ολοκληρωμένα κυκλώματα, που έχουν τοποθετηθεί πάνω σε μια μικρή πλακέτα πυριτίου. Για την αποθήκευση κάθε bit στη μνήμη SRAM, απαιτούνται τέσσερα με έξι transistors, γι’ αυτό και το μέγεθος της SRAM είναι μεγαλύτερο από το μέγεθος της DRAM (κάθε bit χρειάζεται μόνο έναν πυκνωτή). Η πολυπλοκότητα και ο μεγάλος αριθμός transistors που χρησιμοποιούνται για τη μνήμη SRAM, είναι ο βασικότερος λόγος, εξαιτίας του οποίου η μνήμη SRAM έχει μεγαλύτερο κόστος από την DRAM (Dynamic RAM[1]).

[1] Η δυναμική RAM (Dynamic RAM - DRAM) είναι ένας τύπος RAM, που μπορεί να διατηρήσει αναλλοίωτα τα περιεχόμενά της, μόνο αν γίνεται συνεχής αναζωογόνηση σε αυτά από ένα ειδικό

2.4.1. Εσωτερική μνήμη SRAM:

Η μνήμη δεδομένων SRAM (STATIC RAM ή αλλιώς στατική RAM), διατίθενται στα περισσότερα μοντέλα μικροελεγκτών. Στις περιπτώσεις κατά τις οποίες τα δεδομένα μπορούν να αποθηκευτούν σε καταχωρητές, η χρήση της είναι δυνατόν να αποφευχθεί. Πρόκειται για μία μνήμη η οποία δεν είναι απευθείας προσβάσιμη από την Κεντρική Μονάδα Επεξεργασίας (ALU), σε αντίθεση με τους καταχωρητές. Το μέγεθός της κυμαίνεται από 128 bytes σε 4096 bytes, ενώ για την πρόσβαση σε θέσεις μνήμης συνήθως χρησιμοποιούμε έναν καταχωρητή για προσωρινή ενδιάμεση αποθήκευση. Η μνήμη αυτή χρησιμοποιείται εκτός από την αποθήκευση μεταβλητών και ως στοίβα (Stack) του συστήματος, καθώς εάν εμφανιστεί μία διακοπή, η τρέχουσα κάθε φορά τιμή του μετρητή προγράμματος αποθηκεύεται στη στοίβα. Προφανώς οι λειτουργίες μέσω της SRAM είναι πιο αργές σε σχέση με τους καταχωρητές, γεγονός το οποίο οφείλεται στο ότι ο χρόνος πρόσβασης σε αυτή απαιτεί δύο περιόδους του σήματος χρονισμού. Κατά την πρώτη περίοδο εκτελούνται οι απαιτούμενες λειτουργίες των καταχωρητών για τον καθορισμό της διεύθυνσης και στη δεύτερη περίοδο λαμβάνει χώρα η πρόσβαση στη μνήμη (εγγραφή ή ανάγνωση). Οι φάσεις αυτές της εγγραφής και ανάγνωσης απεικονίζονται στο σχήμα που ακολουθεί.



κύκλωμα που ονομάζεται refresh circuit (κύκλωμα αναζωογόνησης). Το κύκλωμα αναζωογόνησης διαβάζει με μεγάλη συχνότητα τα περιεχόμενα κάθε πυκνωτή (σε κάθε πυκνωτή μπορούμε να αποθηκεύσουμε ένα bit), ανεξάρτητα αν τα περιεχόμενα του πυκνωτή χρησιμοποιούνται ή όχι εκείνη τη στιγμή. Εξαιτίας του τρόπου κατασκευής του κυκλώματος αναζωογόνησης, το διάβασμα των περιεχομένων των πυκνωτών έχει ως αποτέλεσμα και τη διατήρηση τους. Αν το κύκλωμα αναζωογόνησης δεν υπήρχε, τότε τα περιεχόμενα της μνήμης θα χάνονταν ακόμα και όταν ο υπολογιστής τροφοδοτούνταν με ρεύμα. Η DRAM ονομάζεται δυναμική RAM, εξαιτίας της διαδικασίας αναζωογόνησης των περιεχομένων της.

Σχήμα 2.6: Φάσεις εγγραφής – ανάγνωσης SRAM

Η πιο σημαντική χρήση της μνήμης SRAM είναι η λειτουργία στοίβας, αφού εκεί μπορούμε να αποθηκεύουμε (εντολή push) και μετά να ανακτούμε (εντολή pop) πληροφορίες, είτε πρόκειται για τιμές είτε για το περιεχόμενο ενός καταχωρητή ή τη διεύθυνση επιστροφής υπορουτίνας ή διακοπής του υλικού.

2.4.2. Εξωτερική μνήμη SRAM

Η μνήμη αυτή χρησιμοποιείται μόνο στους μεγαλύτερους επεξεργαστές της σειράς AVR και σε εφαρμογές υψηλών απαιτήσεων. Στους επεξεργαστές εκείνους που διαθέτουν θύρες πρόσβασης σε εξωτερικό δίαυλο δεδομένων και διευθύνσεων, μπορεί να χρησιμοποιηθεί οποιαδήποτε ποσότητα εξωτερικής μνήμης SRAM επιθυμεί ο χρήστης. Ο χρόνος πρόσβασης στην εξωτερική μνήμη διαρκεί τρεις περιόδους χρονισμού, όμως υπάρχουν περιπτώσεις που μπορεί να διαρκέσει τέσσερις.

2.5. Καταχωρητές και Καταχωρητές Εργασίας (Register File)

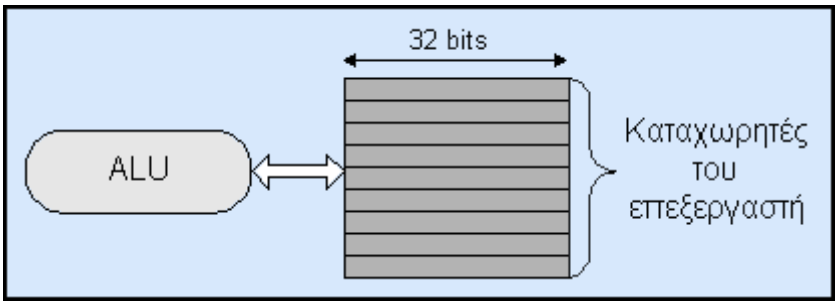
Οι καταχωρητές είναι τοπικές περιοχές ή αλλιώς μικρά κύτταρα μνήμης μέσα στον επεξεργαστή που χρησιμοποιούνται για να κρατούν δεδομένα, τα οποία επεξεργάζονται εκείνη τη στιγμή. Κάθε επεξεργαστής έχει μερικούς καταχωρητές, κάποιους αφοσιωμένους σε μια συγκεκριμένη λειτουργία και άλλους διαθέσιμους για γενική χρήση από τους προγραμματιστές. Μερικοί καταχωρητές έχουν ειδική λειτουργία:

- Απαριθμητής προγράμματος (program counter): περιέχει τη διεύθυνση της επόμενης εντολής που θα ανακτηθεί από την μνήμη για να εκτελεστεί.
- Καταχωρητής εντολών (Instruction register): αποθηκεύει τον κωδικό λειτουργίας της εντολής πριν αποκωδικοποιηθεί από την ΚΜΕ.
- Συσσωρευτής (accumulator): καταχωρητής που συνήθως χρησιμοποιείται για τις αριθμητικές και λογικές πράξεις.

Οι καταχωρητές είναι η ταχύτερη μνήμη, διαθέσιμη για χρήση στους υπολογιστές, ακόμη πιο γρήγορη και από την L1 Cache. Στους καταχωρητές συντελούνται οι περισσότερες λειτουργίες, καθώς ο επεξεργαστής δεν μπορεί να εκτελεί πράξεις στη μνήμη. Για παράδειγμα, αν θέλουμε να προσθέσουμε τον αριθμό 1 σε μία περιοχή μνήμης, ο επεξεργαστής κανονικά θα το κάνει αυτό, φορτώνοντας την αρχική τιμή από τη μνήμη στον καταχωρητή, προσθέτοντας τον αριθμό

1 στον καταχωρητή και σώζοντας μετά την τιμή πίσω στη μνήμη. Αυτό φυσικά γίνεται με τέτοια ταχύτητα που ο χρήστης δεν το καταλαβαίνει.

Το πλάτος (σε bits) των καταχωρητών του επεξεργαστή καθορίζει πόσα δεδομένα μπορεί να επεξεργαστή σε μια χρονική στιγμή. Αυτό ορισμένες φορές χρησιμοποιείται για να προσδιορίσει το «μέγεθος» του επεξεργαστή. Για παράδειγμα, μπορεί να ακούσετε να μιλούν για «16 bit επεξεργαστή» ή για «32 bit επεξεργαστή». Αυτός ο όρος συνήθως αναφέρεται στο μέγεθος του καταχωρητή μέσα στον επεξεργαστή. Ωστόσο, αυτός ο όρος κάποιες φορές χρησιμοποιείται λανθασμένα και κάποιοι αναφέροντας το μέγεθος του επεξεργαστή, εννοούν για παράδειγμα το εύρος του διαύλου του.



Σχήμα 2.7: Καταχωρητές μικροεπεξεργαστή

Όσους περισσότερους καταχωρητές έχει ο επεξεργαστής, τόσο μεγαλύτερη ευκινησία έχουν οι προγραμματιστές να γράψουν καλύτερο κώδικα. Ωστόσο, κάτι τέτοιο αυξάνει την πολυπλοκότητα του επεξεργαστή. Όλοι οι μικροελεγκτές της οικογένειας AVR διαθέτουν 32 καταχωρητές γενικής χρήσης, σε μερικούς από τους οποίους προβλέπονται και επιπλέον ειδικές λειτουργίες. Η αναγνώριση των καταχωρητών εργασίας γίνεται με τα σύμβολα R0 έως R31 ενώ κάθε ομάδα χωρίζεται σε δύο τμήματα, καθένα από τα οποία διεθέτει συνολικά 16 καταχωρητές, τους R0 και R15 το ένα και τους R16 έως R31 το άλλο. Σχέδον όλες οι εντολές που χρησιμοποιούν ως τελεστές τους, καταχωρητές εργασίας, έχουν άμεση πρόσβαση σε όλους τους καταχωρητές και ολοκληρώνονται σε διάρκεια μίας και μόνο περιόδου του ρολογιού.

2.6 Η Αριθμητική Λογική Μονάδα (ALU)

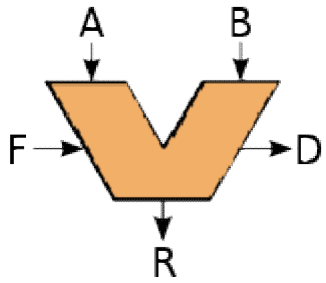
Στην επιστήμη των υπολογιστών η Αριθμητική Λογική Μονάδα (ALU) αποτελεί την Κεντρική Μονάδα εός επεξεργαστή και είναι ένα ψηφιακό κύκλωμα το οποίο εκτελεί αριθμητικούς και λογικούς υπολογισμούς, καθώς επίσης και εντολές σε επίπεδο bit, σε σχέση με

τα περιεχόμενα των καταχωρητών, ενώ αποθηκεύει τα αποτελέσματα των πράξεων σε συγκεκριμένο καταχωρητή που υποδεικνύεται από την ίδια την εντολή. Οι εντολές εκτελούνται στη διάρκεια μίας και μόνο περιόδου του σήματος χρονισμού. Κάθε πράξη που εκτελείται από την ALU, επηρεάζει τις σημαίες (flags) που αποτελούν τα bits του καταχωρητή κατάστασης (Status), ανάλογα με την εκάστοτε εντολή. Όλα τα άλλα στοιχεία του συστήματος του υπολογιστή – η μονάδα ελέγχου, οι καταχωρητές, η μνήμη, η λειτουργία I/O – υπάρχουν κυρίως για να προσκομίζουν δεδομένα στην ALU για να τα επεξεργαστεί, και κατόπιν εξάγουν τα αποτελέσματα.

Η ALU είναι θεμελιώδες δομικό στοιχείο της Κ.Μ.Ε. του υπολογιστή. Ακόμα και οι πιο απλοί μικροεπεξεργαστές έχουν μία για σκοπούς όπως η διαχείριση μετρητών. Οι σύγχρονοι επεξεργαστές και οι μονάδες επεξεργασίας γραφικών, φιλοξενούν ισχυρές και πολύ πολύπλοκες ALU. Ο μαθηματικός Τζον Φον Νόιμαν πρότεινε τον όρο ALU το 1945, όταν έγραψε μια αναφορά για την θεμέλια κατασκευή ενός καινούριου τότε υπολογιστή, του EDVAC. Το 1946, ο Φον Νόιμαν συνεργάστηκε με τους συναδέλφους του στο σχεδιασμό ενός υπολογιστή για λογαριασμό του Institute for Advanced Study του Princeton του New Jersey. Ο συγκεκριμένος υπολογιστής αποτέλεσε το πρότυπο για όλους τους μεταγενέστερους υπολογιστές. Σε αυτή του την αναφορά, ο Φον Νόιμαν υπογράμμιζε τι πίστευε ότι θα χρειάζονταν για την υλοποίηση της μηχανής του, συμπεριλαμβανομένης και της ALU. Ο Φον Νόιμαν αναφέρει ότι η ALU είναι απολύτως σημαντική για έναν υπολογιστή, διότι εγγυάται ότι ένας υπολογιστής μπορεί να υπολογίζει βασικούς μαθηματικούς υπολογισμούς, όπως πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση. Ως εκ τούτου ο υπολογιστής πρέπει να περιέχει εξειδικευμένα όργανα για να εκτελεί αυτές τις πράξεις. Η έρευνα γύρω από τις ALU εξακολουθεί να αποτελεί σημαντικό μέρος της επιστήμης των υπολογιστών.

Μια ALU, πρέπει να επεξεργάζεται αριθμούς χρησιμοποιώντας την ίδια μορφή όπως και το υπόλοιπο ψηφιακό κύκλωμα. Ως γνωστόν, η μορφή με την οποία αναπαριστώνται όλοι οι αριθμοί (και χαρακτήρες) στους σύγχρονους υπολογιστές είναι με την χρησιμοποίηση του δυαδικού συστήματος (με 0 και 1). Οι πρώτοι υπολογιστές χρησιμοποιούσαν ένα πλήθος από διαφορετικά συστήματα. Για κάθε ένα σύστημα, οι ALU είχαν διαφορετικούς σχεδιασμούς συχνά αρκετά πολύπλοκους. Έτσι καθιερώθηκε το αριθμητικό σύστημα του συμπληρώματος ως προς 2, μια αριθμητική παράσταση που καθιστά ευκολότερους τους υπολογισμούς για την ALU. Με αυτό το αριθμητικό σύστημα, η αφαίρεση επιτυγχάνεται με την πρόσθεση του αρνητικού αριθμού (π.χ. αντί για 5-3 , 5+(-3)), και έτσι εξουδετερώνεται η ανάγκη για εξειδικευμένα κυκλώματα για την αφαίρεση. Οι περισσότερες από τις πράξεις ενός επεξεργαστή εκτελούνται

από μία ή περισσότερες ALU. Μια ALU, φορτώνει δεδομένα από καταχωρητές εισόδου, μια εξωτερική μονάδα λέει στην ALU, τι πράξεις να κάνει στα δεδομένα, και στο τέλος η ALU αποθηκεύει το αποτέλεσμα σε έναν εξωτερικό καταχωρητή. Τέλος, ειδικοί μηχανισμοί μετακινούν τα δεδομένα από τους καταχωρητές στη μνήμη.



Σχήμα 2.8: Σχηματικός συμβολισμός της Αριθμητικής Λογικής Μονάδας (ALU).

Οι περισσότερες ALU, μπορούν να εκτελέσουν τις παρακάτω πράξεις :

• Ακέραιες αριθμητικές πράξεις (πρόσθεση, αφαίρεση και μερικές φορές πολ/σμό και διαίρεση, αν και η υλοποίηση τους είναι ακριβή).

• Λογικές πράξεις (και, ή , όχι)

• Μετατόπιση η περιστροφή μιας λέξης τόσο όσο ενός καθορισμένου αριθμού bits, προς τα αριστερά ή δεξιά προσημασμένους ή μη. Οι μετατοπίσεις μπορούν να υλοποιηθούν σαν πολλαπλασιασμοί με το 2 και ως διαιρέσεις με 2.

Ένας μηχανικός μπορεί να σχεδιάσει μια ALU για τον υπολογισμό οποιασδήποτε πράξης, άσχετα με το πόσο πολύπλοκο μπορεί να είναι. Το πρόβλημα έγκειται στο ότι όσο πιο πολύπλοκος είναι ένας υπολογισμός, τόσο πιο ακριβή γίνεται η ALU, καταλαμβάνει περισσότερο χώρο μέσα στον επεξεργαστή και καταναλώνει περισσότερη ενέργεια. Ως εκ τούτου, οι μηχανικοί πάντοτε θέτουν ένα συμβιβασμό, έτσι ώστε να παρέχουν στον επεξεργαστή μια ALU αρκετά δυνατή για να κάνει τον επεξεργαστεί γρήγορο, αλλά και όχι τόσο πολύπλοκο για να μην γίνεται το κόστος και το μέγεθος του απαγορευτικό. Φανταστείτε ότι πρέπει να υπολογίσετε την τετραγωνική ρίζα ενός αριθμού. Ο μηχανικός θα εξετάσει τις ακόλουθες επιλογές για την υλοποίηση αυτού του υπολογισμού :

1. Σχεδίαση μιας υπερβολικά σύνθετης ALU για τον υπολογισμό της τετραγωνικής ρίζας οποιουδήποτε αριθμού σε ένα μόνο βήμα. Αυτό ονομάζεται υπολογισμός σε έναν κύκλο ρολογιού.
2. Σχεδίαση μιας πολύ σύνθετης ALU, που υπολογίζει την τετραγωνική ρίζα οποιουδήποτε αριθμού σε διάφορα στάδια. Τα ενδιάμεσα στάδια όμως περνούν μέσα από μια σειρά κυκλωμάτων διατεταγμένα σε μια γραμμή, σαν μια γραμμή παραγωγής εργοστασίου. Αυτό επιτρέπει στην ALU να δέχεται νέους αριθμούς για υπολογισμό ακόμη και πριν ολοκληρώσει τον υπολογισμό των προηγούμενων αριθμών.
3. Σχεδίαση μιας σύνθετης ALU, που υπολογίζει την τετραγωνική ρίζα με πολλά βήματα. Αυτό ονομάζεται διαδραστικός υπολογισμός, και συνήθως ελέγχεται από μια σύνθετη μονάδα ελέγχου με ενσωματωμένο μικροκώδικα.
4. Σχεδίαση μιας απλής ALU στον επεξεργαστή, και πώληση ενός ξεχωριστού, εξειδικευμένου και πιο δαπανηρού επεξεργαστή που μπορεί να εγκαταστήσει ο πελάτης ακριβώς δίπλα, και ο οποίος υλοποιεί μία από τις παραπάνω εντολές. Αυτός ο επεξεργαστής ονομάζεται συν-επεξεργαστής.
5. Λέμε στους προγραμματιστές ότι δεν υπάρχει συν-επεξεργαστής, έτσι ώστε να σχεδιάσουν τους δικούς τους αλγορίθμους για τον υπολογισμό των τετραγωνικών ριζών σε λογισμικό. Αυτό γίνεται με τη χρήση βιβλιοθηκών λογισμικού.
6. Μίμηση της ύπαρξης ενός συν-επεξεργαστή, δηλαδή κάθε φορά που ένα πρόγραμμα επιχειρεί να εκτελέσει τον υπολογισμό μιας τετραγωνικής ρίζας, κάνε τον επεξεργαστή να ελέγξει την ύπαρξη ενός συν-επεξεργαστή, και κάνε τον να τον χρησιμοποιήσει εάν υπάρχει. Εάν δεν υπάρχει, διέκοψε την επεξεργασία του προγράμματος και επικαλέσου το λειτουργικό σύστημα να εκτελέσει τον υπολογισμό της ρίζας μέσω ενός λογισμικού αλγορίθμου.

Οι παραπάνω επιλογές πάνε από την γρηγορότερη και πιο ακριβή προς την πιο αργή και φθηνότερη. Ως εκ τούτου, ενώ ακόμα και οι πιο απλοί υπολογιστές μπορούν να υπολογίσουν ακόμα και τις πιο πολύπλοκες μαθηματικές συναρτήσεις, οι απλούστεροι υπολογιστές θα κάνουν περισσότερο χρόνο να το κάνουν εξαιτίας των πολλών βημάτων που πρέπει να διεκπεραιώσει για τον υπολογισμό της συνάρτησης. Οι είσοδοι στην ALU είναι τα δεδομένα πάνω στα οποία θα εφαρμοστούν οι διεργασίες (ονομάζονται τελεστές), και έναν κώδικα από την μονάδα ελέγχου που αναφέρει ποια λειτουργία θα εκτελεστεί. Σε πολλά σχέδια η ALU λαμβάνει επίσης ως εισόδους και τα σύνολα των κωδικών κατάστασης από τον καταχωρητή κατάστασης. Οι

κωδικοί αυτοί χρησιμοποιούνται για να δηλώσουν καταστάσεις όπως η υπερχείλιση, διαίρεση με μηδέν κλπ.

2.7 Εντολές μικροελεγκτών AVR

Η βασική λειτουργία των περισσότερων επεξεργαστών, ανεξάρτητα από τη φυσική μορφή τους, είναι να εκτελούν ακολουθίες αποθηκευμένων εντολών. Η εντολή στην πραγματικότητα, είναι ένας αριθμός ή μια ακολουθία αριθμών που αντιστοιχεί σε μια ενέργεια. Η λειτουργία της CPU καθορίζεται από τις εντολές που εκτελεί, και που ονομάζονται *εντολές μηχανής (machine instructions)* ή *εντολές υπολογιστή (computer instructions)*. Η συλλογή των διαφορετικών εντολών που μπορεί να εκτελέσει η CPU ονομαζεται *ομάδα εντολών (instruction set)*. Κάθε εντολή πρέπει να περιέχει τις πληροφορίες που χρειάζεται η CPU για την εκτέλεση. Στο παρακάτω σχήμα φαίνονται τα βήματα που αφορούν στην εκτέλεση εντολής και συνεπώς, ορίζονται τα στοιχεία μιας εντολής μηχανής. Τα στοιχεία αυτά είναι τα παρακάτω:

• **Κώδικας λειτουργίας:** Καθορίζει την πράξη που θα εκτελεστεί. Η πράξη καθορίζεται με έναν ψηφιακό κώδικα, γνωστό σαν κώδικα λειτουργίας ή ***opcode.*** Οι opcode παριστάνονται με συντμήσεις, που ονομάζονται *μνημονικοί (κανόνες)*, οι οποίοι δείχνουν την λειτουργία. Μερικά παραδείγματα είναι:

SUB Αφαίρεση, MPY Πολλαπλασιασμός, DIV Διαίρεση, LOAD Φόρτωση δεδομένων από την μνήμη, STOR Απομνημόνευση δεδομένων στην μνήμη.

• **Αναφορά τελεστέου πηγής:** Η πράξη μπορεί να αφορά σε ένα ή περισσότερους τελεστέους πηγής, δηλαδή, τελεστέους που είναι είσοδοι για την πράξη.

• **Αναφορά τελεστέου αποτελέσματος:** Η πράξη μπορεί να δώσει αποτέλεσμα.

• **Αναφορά επόμενης εντολής:** Αυτή λέει στην CPU που μπορεί να προσάγει την επόμενη εντολή μετά την πλήρη εκτέλεση αυτής της εντολής. Η επόμενη εντολή που θα προσαχθεί βρίσκεται στην κύρια μνήμη, ή, στην περίπτωση συστήματος με εικονική μνήμη, είτε σε κύρια μνήμη είτε σε δευτερεύουσα μνήμη. Στις περισσότερες περιπτώσεις, η επόμενη εντολή που θα προσαχθεί ακολουθεί αμέσως μετά την τρέχουσα εντολή. Στις περιπτώσεις εκείνες, δεν υπάρχει συγκεκριμένη αναφορά στην επόμενη εντολή. Όταν χρειάζεται σαφής αναφορά, τότε θα πρέπει να δοθεί η διεύθυνση κύριας μνήμης ή εικονικής μνήμης.

Οι τελεστέοι πηγής και αποτελέσματος μπορούν να βρίσκονται σε μια από τρεις περιοχές:

• **Κύρια ή εικονική μνήμη:** Όπως συμβαίνει και με τις αναφορές επόμενης εντολής θα πρέπει να δοθεί η διεύθυνση κύριας ή εικονικής μνήμης.

• **Καταχωρητής CPU:** Μια CPU περιέχει έναν ή περισσότερους καταχωρητές στους οποίους μπορούν να αναφερθούν οι εντολές μηχανής. Αν υπάρχει μόνο ένας καταχωρητής, η αναφορά σ'αυτόν μπορεί να εννοείται. Αν υπάρχουν περισσότεροι από έναν καταχωρητές, τότε σε κάθε καταχωρητή ανατίθεται ένας μοναδικός αριθμός, και η εντολή θα πρέπει να περιέχει τον αριθμό του επιθυμητού καταχωρητή.
• **Συσκευή I/O:** Η εντολή θα πρέπει να καθορίζει τη μονάδα και συσκευή λειτουργίας I/O. Αν χρησιμοποιηθεί I/O με απεικόνιση μνήμης, αυτή θα είναι απλά μια ακόμη διεύθυνση κύριας μνήμης, αυτή θα είναι απλά μια ακόμη διεύθυνση κύριας ή εικονικής μνήμης.

2.7.1. Μορφή και κωδικοποίηση

Μια εντολή μπορεί να επεξεργαστεί δεδομένα ή να μεταβάλλει καταστάσεις στο εσωτερικό της ΚΜΕ. Αν είναι εντολή επεξεργασίας μπορεί να μεταφέρει, να προσθέσει, ή να συγκρίνει δεδομένα, ενώ αν είναι εντολή μεταβολής μπορεί να τροποποιήσει τον απαριθμητή προγράμματος και τον καταχωρητή ενδείξεων. Μεταβάλλοντας τον απαριθμητή προγράμματος μπορεί να αλλάξει η ροή εκτέλεσης του προγράμματος. Τέτοιες εντολές είναι γνωστές ως "άλματα" (jumps) και είναι ο βασικός τρόπος υλοποίησης των βρόχων επιλογής και επανάληψης. Ο καταχωρητής ενδείξεων περιέχει διακριτές καταστάσεις ελέγχου, όπως ο ενδείκτης υπερχείλισης. Οι ενδείκτες επηρεάζουν τον τρόπο με τον οποίο συμπεριφέρεται ένα πρόγραμμα, δεδομένου ότι συχνά αποτελούν αναφορά στα αποτελέσματα διαφόρων εργασιών.

Κάθε εντολή περιλαμβάνει την λειτουργία που θα εκτελέσει, την πηγή των δεδομένων και τη διεύθυνση της επόμενης εντολής. Όλες αυτές οι πληροφορίες κωδικοποιούνται σε ένα ή περισσότερα bytes ανάλογα με τον τύπο της εντολής. Για να προκύψουν όσο το δυνατόν λιγότερα bytes δεχόμαστε ότι η επόμενη εντολή ακολουθεί αμέσως μετά την τρέχουσα εντολή. Έτσι συνήθως οι εντολές αποτελούνται από δύο τμήματα, οο πρώτο τμήμα περιέχει τον κωδικό εντολής (operation code) και υποδηλώνει τη λειτουργία προς εκτέλεση, και το δεύτερο τμήμα περιέχει πληροφορίες που απαιτούνται για την εν λόγω εντολή, για παράδειγμα οι τελεστές για την λειτουργία της πρόσθεσης ή η διεύθυνση της επόμενης εντολής αν είναι εντολή διακλάδωσης.

Για κάθε εντολή υπάρχει διαφορετική κωδικοποίηση. Ακόμη και στην ίδια εντολή, ανάλογα με τον τύπο τελεστών που χρησιμοποιεί, δηλαδή αν είναι καταχωρητές, θέσεις μνήμης ή απευθείας δεδομένα, υπάρχει διαφορετική κωδικοποίηση. Στην πραγματικότητα, όλες οι εντολές είναι δυαδικοί αριθμοί που έχουν αντιστοιχηθεί σε μια λειτουργία. Για να γίνονται πιο

κατανοητές οι εντολές συνήθως αναπαρίστανται είτε σε δεκαεξαδική μορφή είτε σε συμβολική γλώσσα. Για παράδειγμα η εντολή *ADD C* δηλώνει ότι θα προστεθεί το περιεχόμενο του καταχωρητή C με το περιεχόμενο του συσσωρευτή και το αποτέλεσμα θα αποθηκευτεί στον συσσωρευτή. Η αντίστοιχη δεκαεξαδική αναπαράσταση της εντολής αυτής είναι *81h*, σε αρχιτεκτονικές x86.

2.7.2. Κύκλος εντολής

Κύκλος εντολής είναι το διάστημα που απαιτείται για την αποπεράτωση μιας εντολής και την έναρξη της επόμενης. Υπάρχουν τέσσερα στάδια για την ολοκλήρωση ενός κύκλου: η ανάκληση (fetch), η αποκωδικοποίηση (decode), η εκτέλεση (execute) και η αποθήκευση του αποτελέσματος (store/writeback). Κατά την **ανάκληση**, ανακτάται η εντολή από τη θέση μνήμης που είναι αποθηκευμένη. Η θέση της εντολής στην μνήμη περιέχεται στον απαριθμητή προγράμματος. Όταν η εντολή μεταφερθεί από την μνήμη στον επεξεργαστή αποθηκεύεται στον καταχωρητή εντολών. Έπειτα αυξάνεται η τιμή του απαριθμητή προγράμματος, όσο είναι και το μήκος της εντολής σε μονάδες μνήμης, ώστε να υποδεικνύει την θέση της επόμενης εντολής ή την διεύθυνση των τελεστών σε περίπτωση που η τρέχουσα εντολή έχει τελεστές. Συχνά η εντολή προς ανάκληση καθυστερεί να μεταφερθεί από την μνήμη στον επεξεργαστή, λόγο ασύγχρονης λειτουργίας των δύο συσκευών, προκαλώντας παύση στην λειτουργία της ΚΜΕ. Για την αντιμετώπιση αυτού του προβλήματος, στους σύγχρονους επεξεργαστές γίνεται χρήση ενδιάμεσης μνήμης προσωρινής αποθήκευσης (cache) αλλά και τεχνικές σωλήνωσης (pipelining).

Στο στάδιο της **αποκωδικοποίησης**, η εντολή διασπάται και ερμηνεύεται από τον επεξεργαστή. Ανάλογα με τον κωδικό εντολής, κατά την αποκωδικοποίηση, ανακαλούνται και τυχόν τελεστές. Η τιμή των τελεστών ανακαλείται είτε άμεσα ως σταθερά, είτε έμμεσα ως μια διεύθυνση στην οποία βρίσκεται αποθηκευμένη η τιμή, σε κάποιο καταχωρητή ή μνήμη, όπως ορίζει το εκάστοτε πρότυπο διευθυνσιοδότησης. Σε παλιότερα σχέδια επεξεργαστών, η αποκωδικοποίηση της εντολής ήταν μια αμετάβλητη διαδικασία που πραγματοποιούσε το υλικό. Ωστόσο, σε πιο περίπλοκες αρχιτεκτονικές επεξεργαστών, για την ερμηνεία των εντολών χρησιμοποιείται ένα μικροπρόγραμμα. Το μικροπρόγραμμα συνήθως είναι επαναπρογραμματιζόμενο ώστε να μπορεί να μεταβληθεί, ακόμη και μετά την κατασκευή της ΚΜΕ, ο τρόπος που αποκωδικοποίησης των εντολών.

Μετά την ανάκληση και την αποκωδικοποίηση, ακολουθεί η **εκτέλεση** της εντολής. Σε αυτό το στάδιο, διάφορες μονάδες του επεξεργαστή συνδέονται ώστε να γίνει εφικτή η εκτέλεση της επιθυμητής λειτουργίας. Αν, για παράδειγμα, ζητήθηκε μια λειτουργία πρόσθεσης, η αριθμητική μονάδα (AU) θα συνδεθεί με ένα σύνολο εισόδων και εξόδων. Οι είσοδοι θα παρέχουν τους αριθμούς που πρέπει να προστεθούν και οι έξοδοι θα περιέχουν το άθροισμα. Εάν η πρόσθεση έχει ως αποτέλεσμα έναν υπερβολικά μεγάλο αριθμό για να χειριστεί η ΚΜΕ τότε θα ενεργοποιηθεί ο ενδείκτης αριθμητικής υπερχείλισης. Η αριθμητική λογική μονάδα (ALU) στο σύνολό της περιέχει κυκλώματα για την εκτέλεση απλών αριθμητικών και λογικών πράξεων, όπως η πρόσθεση και η σύγκριση αριθμών.

Στο τελικό στάδιο, την **αποθήκευση**, η ΚΜΕ στέλνει τα δεδομένα προς αποθήκευση στη μνήμη. Τα αποτελέσματα αρχικά αποθηκεύονται προσωρινά σε κάποιο καταχωρητή για ταχύτερη προσπέλαση από επόμενες εντολές και έπειτα αποθηκεύονται στην κύρια μνήμη του συστήματος. Εντολές που κάνουν άλματα αλλά και εντολές που μεταβάλλουν τον καταχωρητή ενδείξεων στην ουσία δεν παράγουν κάποιο αποτέλεσμα προς αποθήκευση. Μετά την αποθήκευση των αποτελεσμάτων που προέκυψαν, ο κύκλος εντολής ολοκληρώνεται και επαναλαμβάνεται με την επόμενη εντολή, αφού αυξήθηκε ο απαριθμητής προγράμματος. Σε επεξεργαστές με πιο περίπλοκη αρχιτεκτονική, περισσότερες εντολές μπορεί να ανακαλούνται, να αποκωδικοποιούνται και να εκτελούνται ταυτόχρονα.

Οι περισσότερες από τις εντολές των μικροελεγκτών AVR έχουν μήκος μίας λέξης, δηλαδή 2 bytes, οπότε καταλαμβάνουν χώρο που αντιστοιχεί σε μία θέση της μνήμης προγράμματος. Επίσης, οι περισσότερες από τις εντολές εκτελούνται στη διάρκεια μίας περιόδου του κεντρικού σήματος χρονισμού, ενώ πολύ λίγες από αυτές απαιτούν για την εκτέλεσή τους 2 ή περισσότερες περιόδους του ίδιου σήματος. Αυτή η εκτέλεση των εντολών που διατηρείται ως επί το πλείστον στη διάρκεια μίας περιόδου του κεντρικού ρολογιού, οφείλεται κυρίως στο μηχανισμό της συνεχούς διοχέτευσης εντολών (instruction pipeline), η οποία λειτουργεί σε δύο στάδια. Η συνεχής διοχέτευση λειτουργεί ταυτόχρονα με ανάκληση της επόμενης εντολής από τη μνήμη προγράμματος κατά τη διάρκεια εκτέλεσης της προηγούμενης εντολής, η οποία λαμβάνει χώρα στο υπόλοιπο τμήμα του επεξεργαστή. Συνεπώς, σύμφωνα με το μηχανισμό αυτόν η ανάκληση και η αποκωδικοποίηση των εντολών, καθώς επίσης και η εκτέλεσή τους, αποτελούν διαδικασίες οι οποίες λαμβάνουν χώρα ταυτόχρονα κατά τη λειτουργία του επεξεργαστή.

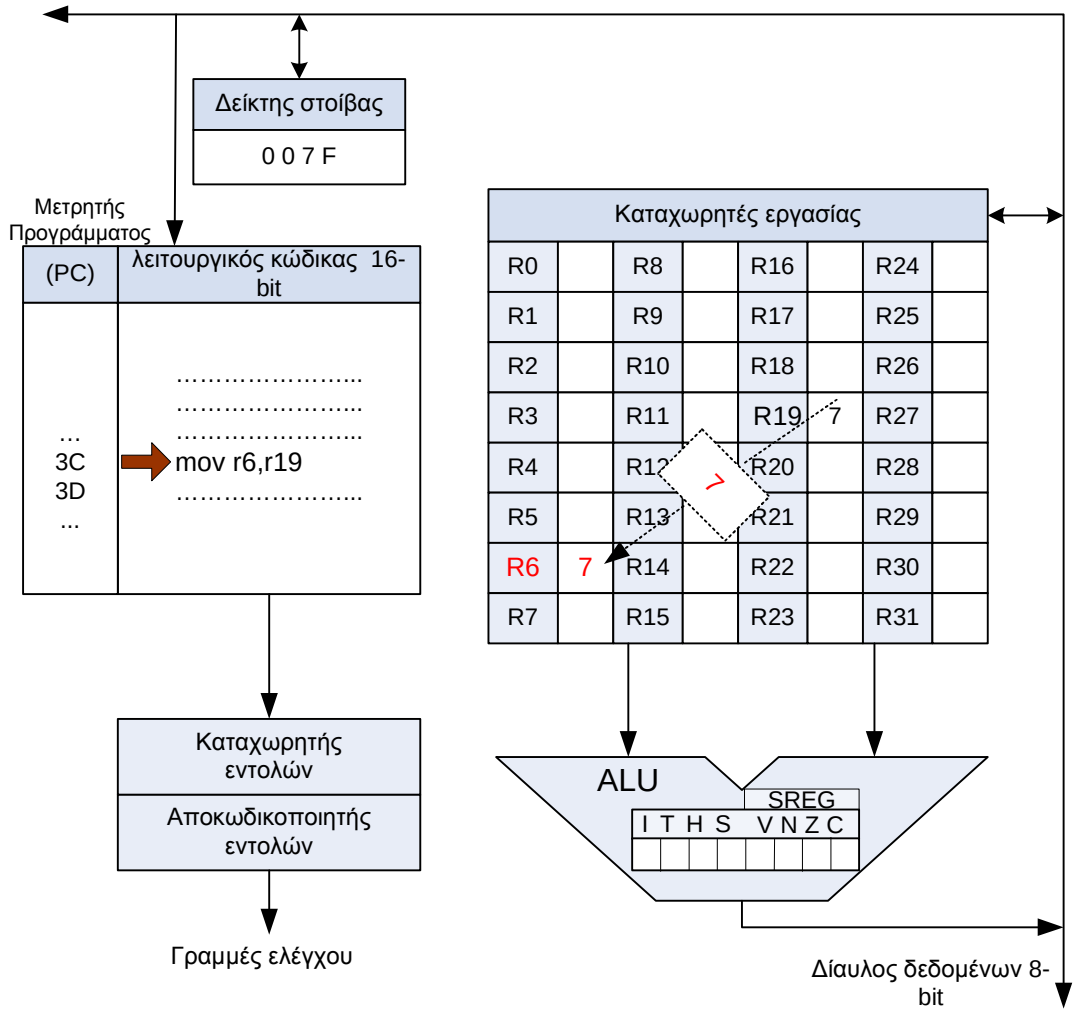
Οι εντολές της γλώσσας των μικροελεγκτών μπορούν να ομαδοποιηθούν στις παρακάτω τέσσερις κατηγορίες:

1. **Εντολές μεταφοράς δεδομένων μεταξύ καταχωρητών ή μεταξύ καταχωρητών και μνήμης ή μεταξύ καταχωρητή και σταθεράς**. Ο καταχωρητής είναι ένας από τους 32 καταχωρητές εργασίας (που αναγνωρίζονται με τα σύμβολα από R0 έως R31) και διαθέτουν το αντίστοιχο τμήμα μνήμης (register file). Η ομάδα των καταχωρητών εργασίας χωρίζεται σε δύο τμήματα καθένα από τα οποία συνολικά διαθέτει 16 καταχωρητές, από τον R0έως τον R15 και από τον R16 στον R31. Όλες σχεδόν οι εντολές που χρησιμοποιούν ως τελεστές τους τους καταχωρητές εργασίας, έχουν άμεση πρόσβαση σε όλους τους καταχωρητές και ολοκληρώνονται σε διάρκεια μίας και μόνο περιόδου του ρολογιού.

Εντολές μεταφοράς δεδομένων					
MOV	Rd, Rr	Move Between Registers	Rd ß Rr	Καμία	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ß Rr+1:Rr	Καμία	1
LDI	Rd, K	Load Immediate	Rd ß K	Καμία	1
LD	Rd, X	Load Indirect	Rd ß (X)	Καμία	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ß (X), X ß X+ 1	Καμία	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ß X – 1, Rd ß (X)	Καμία	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ß (Z+ q)	Καμία	2
LDS	Rd, k	Load Direct from SRAM	Rd ß (k)	Καμία	2
ST	X, Rr	Store Indirect	(X) ß Rr	Καμία	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ß Rr, X ß X+ 1	Καμία	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ß X – 1, (X) ß Rr	Καμία	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ß Rr	Καμία	2
STS	k, Rr	Store Direct to SRAM	(k) ß Rr	Καμία	2
LPM	Rd, Z	Load Program Memory	R0 ß (Z)	Καμία	3
SPM	Rr	Store Program Memory	(Z) ß R1:R0	Καμία	-
IN	Rd	In Port	Rd ß P	Καμία	1
OUT	Rd, Rr	Out Port	P ß Rr	Καμία	1
PUSH	Rd, Rr	Push Register on Stack	STACK ß Rr	Καμία	2
POP	Rd, K	Pop Register from Stack	Rd ß STACK	Καμία	2

Πίνακας 2.1: Εντολές μεταφοράς δεδομένων

Όπου Rr καταχωρητής προέλευσης και Rd καταχωρητής προορισμού του Register File, k μια σταθερή διεύθυνση και K μια σταθερά. Οι καταχωρητές X,Y,Z λέγονται καταχωρητές δείκτη και αντιστοιχούν στα ζευγη X=R27:R26, Y=R29:R28 και Z=R31:R30. Χρησιμοποιούνται ως 16-bit καταχωρητές δείκτη για πρόσβαση προς θέσεις μνήμης SRAM ή προς θέσεις μνήμης προγράμματος (μονάχα ο Z). Τέλος, q (6-bit) είναι η μετατόπιση για έμμεση διευθυνσιοδότηση. Οι εντολές LD, ST εφαρμόζονται και για τους τρεις καταχωρητές δείκτη.



Σχήμα 2.9: Παράδειγμα μεταφοράς μεταξύ των καταχωρητών

2. **Εντολές αριθμητικών και λογικών πράξεων**, όπου οι εντολές της ομάδας αυτής προϋποθέτουν τη χρήση της αριθμητικής λογικής μονάδας (ALU) σε αντίθεση με τις εντολές μεταφοράς δεδομένων. Εκτελούνται μόνο μεταξύ καταχωρητών εργασίας ή καταχωρητή εργασίας με σταθερά. Η εντολή διαβάζει τα περιεχόμενα του καταχωρητή, εκτελεί τις πράξεις

με αυτά και αποθηκεύει το αποτέλεσμα στον ίδιο ή σε άλλο καταχωρητή. Οι εντολές αριθμητικών πράξεων διακρίνονται σε εντολές πρόσθεσης και αφαίρεσης με ή χωρίς κρατούμενο μεταξύ δύο καταχωρητών ή ενός καταχωρητή και μιας σταθεράς ή ζεύγους καταχωρητών και μιας σταθεράς, και εντολές πολλαπλασιασμού προσημασμένων ή μη.

α. Εντολές πρόσθεσης:

ADD Rd,Rr : Rd ß Rd + Rr

Περιγραφή: Πρόσθεση 2 καταχωρητών χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.

ADC Rd, Rr **:** Rd R ßd + Rr + C

Περιγραφή: Πρόσθεση 2 καταχωρητών με χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.

ADIW Rd,K **:** Rd+1:Rd Rd ß+1:Rd + K

Περιγραφή: Πρόσθεση μιας τιμής K(0-63) σε ένα ζεύγος καταχωρητών και αποθήκευση στο ζεύγος καταχωρητών.

β. Εντολές αφαίρεσης:

SUB Rd,Rr **:** Rd ß Rd - Rr

Περιγραφή: Αφαίρεση 2 καταχωρητών χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd

SUBI Rd,K **:** Rd ßRd - K

Περιγραφή: Αφαίρεση μιας τιμής χωρίς χρήση κρατουμένου από έναν καταχωρητή και αποθήκευση στον καταχωρητή

SUBCI Rd,K **:** Rd ßRd – K-C

Περιγραφή: Αφαίρεση μιας τιμής με χρήση κρατουμένου από έναν καταχωρητή και ποθήκευση στον καταχωρητή

SBC Rd,Rr **:** Rd R ßd – Rr-C

Περιγραφή: Αφαίρεση 2 καταχωρητών με χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.

SBIWRd,K**:**Rd+1:Rd **ß** Rd+1:Rd-K

Περιγραφή: Αφαίρεση μιας τιμής από ένα ζεύγος καταχωρητών και αποθήκευση στο ζεύγος

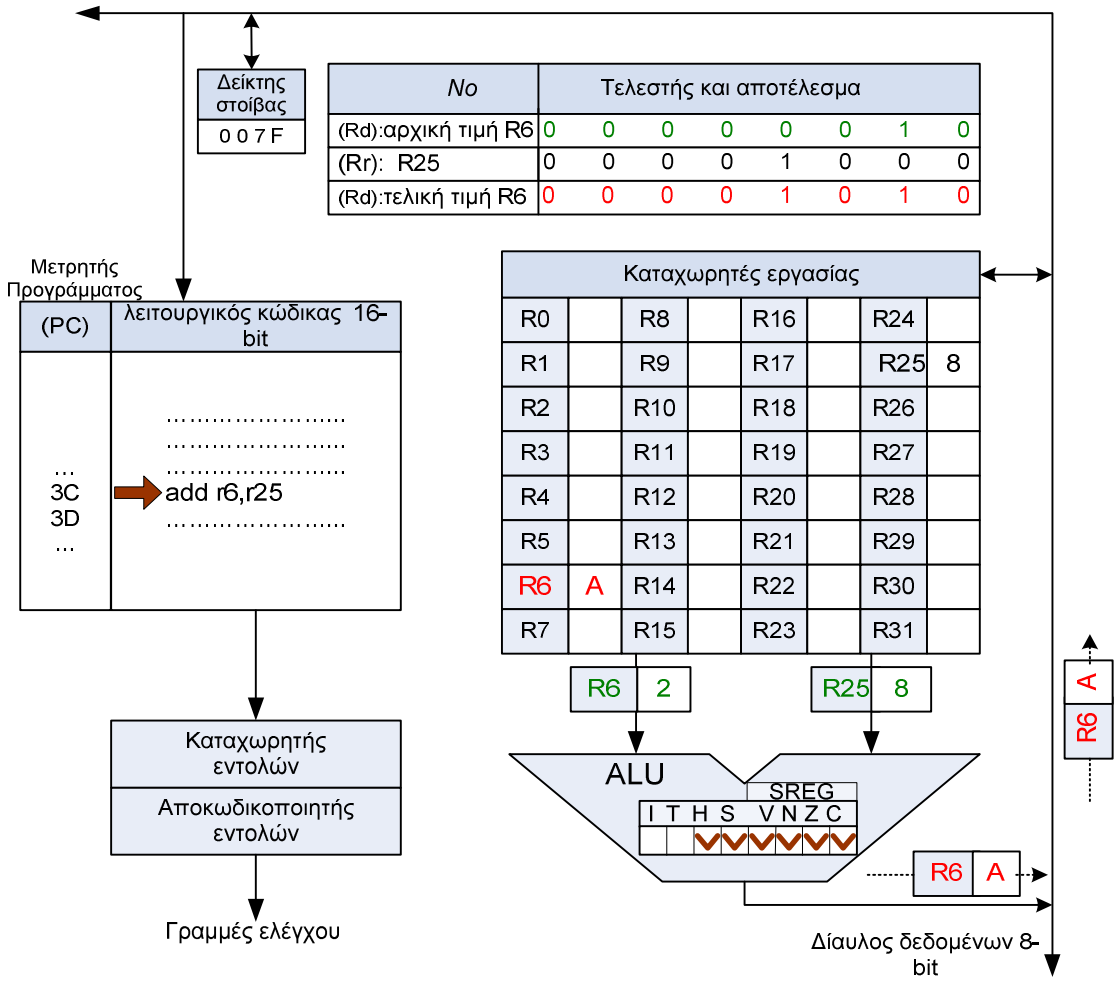
Καταχωρητών.

Υπάρχουν εντολές πολλαπλασιασμού μεταξύ προσημασμένων ή μη, περιλαμβανομένου του κλασματικού τμήματος ή όχι, αλλά δεν υποστηρίζονται από όλα τα μοντέλα μικροελεγκτή.

γ. Εντολές πολλαπλασιασμού:

MUL Rd,Rr **:** R1, R0 **ß** Rd **û**Rr

Περιγραφή: Μη προσημασμένος πολλαπλασιασμός μεταξύ των περιεχομένων των καταχωρητών Rd,Rr. Το αποτέλεσμα αποθηκεύεται στο ζεύγος καταχωρητών R1, R0.



Σχήμα 2.10: Παράδειγμα απευθείας διευθυνσιοδότησης δύο καταχωρητών.

Επίσης, εδώ ανήκουν οι εντολές αύξησης, ελάττωσης, μηδενισμού ή θεσίματος καταχωρητή ή bit καταχωρητή, συμπληρώματος του ένα και του δύο.

Οι εντολές λογικών πράξεων διακρίνονται σε εντολές λογικού ΚΑΙ, Η , ΑΠΟΚΛΕΙΣΤΙΚΟΥ Η:

AND Rd,Rr **:** Λογικό AND Rd ß Rd ŸRr

Περιγραφή: Λογικό AND μεταξύ των περιεχομένων των καταχωρητών Rd,Rr

OR Rd,Rr **:** Λογικό OR Rd ß Rd v Rr

Περιγραφή: Λογικό OR μεταξύ των περιεχομένων των καταχωρητών Rd,Rr

EOR Rd,Rr **:** Λογικό EXOR Rd ß Rd +Rr

Περιγραφή: Λογικό EXOR (αποκλειστικό ‘ή’) μεταξύ των περιεχομένων των καταχωρητών Rd,Rr.

Αντίστοιχα, υπάρχουν οι εντολές λογικού ΚΑΙ, Η μεταξύ κατ/τη και σταθεράς: ANDI Rd,K , ORI Rd,K.

Εντολές συμπληρώματος του 1 και 2:

COM Rd **:** Rd ß $FF - Rd

Περιγραφή: Αντιστροφή των περιεχομένων του καταχωρητή Rd με χρήση αριθμητικής συμπληρώματος του 1.

NEG Rd **:** Rd ß $00 - Rd

Περιγραφή: Αντιστροφή των περιεχομένων του καταχωρητή Rd με χρήση αριθμητικής συμπληρώματος του 2.

Τοποθέτηση λογικού ‘1’ ή ‘0’ στα bits καταχωρητή:

SBR Rd,K **:** Rd ß Rd v K

Περιγραφή: Τοποθέτηση λογικού ‘1’ στα bits του καταχωρητή Rd εκτελώντας λογικό ORI μεταξύ των περιεχομένων του καταχωρητή Rd και της σταθεράς K

CBR Rd,K **:** Rd ß Rd Ÿ($FFh -K)

Περιγραφή: Τοποθέτηση λογικού ‘0’ στα bits του καταχωρητή Rd εκτελώντας λογικό AND μεταξύ των περιεχομένων του καταχωρητή Rd και της σταθεράς K.

Αύξηση/ μείωση καταχωρητή:

INC Rd **:** Rd ß Rd + 1

Περιγραφή: Αύξηση των περιεχομένων του καταχωρητή Rd κατά μία μονάδα.

DEC Rd**:** Rd ß Rd - 1

Περιγραφή: Μείωση των περιεχομένων του καταχωρητή Rd κατά μία μονάδα.

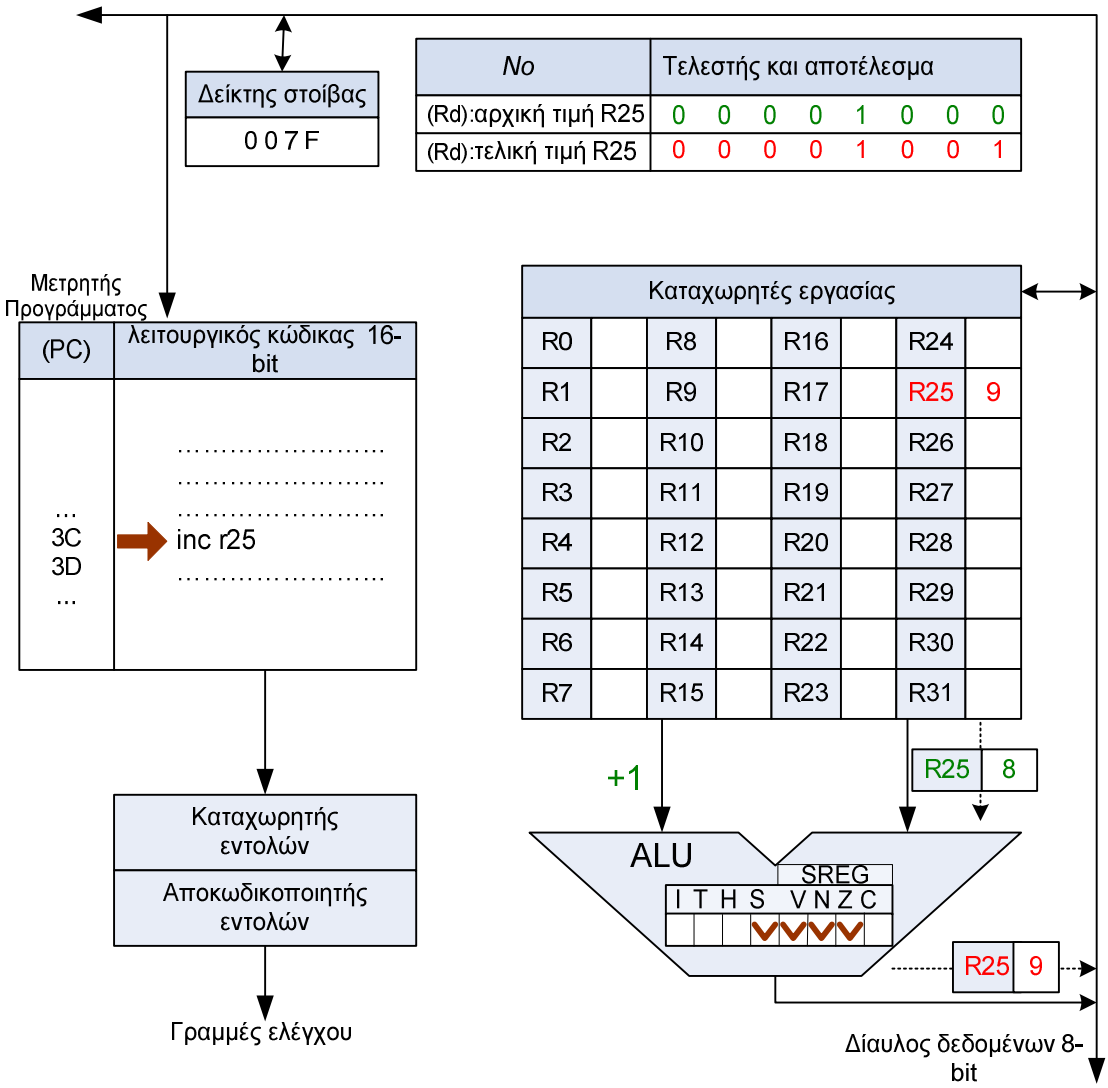
Μηδενισμός/ θέσιμο καταχωρητή:

CLR Rd **:** Rd ß Rd +Rd

Περιγραφή: Μηδενισμός του καταχωρητή Rd με εκτέλεση αποκλειστικού ‘ή’ μεταξύ του καταχωρητή και του εαυτού του.

SER Rd**:** Rd ß $FF

Περιγραφή: Τοποθέτηση ‘1’ σε όλα τα bits του καταχωρητή με φόρτωση της τιμής $FF.



Σχήμα 2.11: Παράδειγμα άμεσης διευθυνσιοδότησης ενός καταχωρητή.

2.7.3.Εντολές σε επίπεδο bit και ελέγχου bit

Οι εντολές αυτές μας παρέχουν τη δυνατότητα να θέσουμε ή να μηδενίσουμε σημαίες του καταχωρητή κατάστασης, διακοπές και συγκεκριμένα bit καταχωρητών ή θυρών. Επίσης, να ολισθήσουμε κάποιον καταχωρητή μέσω κρατουμένου ή όχι. Πρόκειται για τις εξής εντολές:

Εντολές σε επίπεδο bit και ελέγχου bit					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ß 1	Καμία	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ß 0	Καμία	2
LSL	Rd	Logical Shift Left	Rd(n+1) ß Rd(n), Rd(0) ß 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ß Rd(n+1), Rd(7) ß 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) ß C, Rd(n+1) ß Rd(n), C ß Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7) ß C, Rd(n) ß Rd(n+1), C ß Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ß Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3-0) ß Rd(7-4) Rd(7-4) ß Rd(3-0)	Καμία	1
BSET	S	Flag Set	SREG(s) ß 1	SREG(s)	1
BCLR	S	Flag Clear	SREG(s) ß 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ß Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ß T	Καμία	1
Sex		Set Flag	x ß 1	X	1
CLx		Clear Flag	x ß 0	X	1

Πίνακας 2.2: Εντολές σε επίπεδο bit και ελέγχου bit.

Όπου x μια από τις σημαίες του καταχωρητή κατάστασης C,N,Z,I,S,V,T,H και b το bit0-7 ενός καταχωρητή ή μιας θύρας (3-bit) , s το bit0-7 του καταχωρητή κατάστασης (3-bit) , k μια σταθερή διεύθυνση και P καταχωρητής εισόδου-εξόδου.

- **<u>Ολίσθηση καταχωρητή αριστερά ή δεξιά μέσω κρατουμένου ή όχι</u>**:

LSL Rd **:** Rd(n+1) □Ṛd(n) , Rd(0) □ 0 ,C □Rd(7)

Περιγραφή: Ολίσθηση των bits του καταχωρητή Rd μία θέση προς τα αριστερά..Το bit0 μηδενίζεται και το bit7 φορτώνεται στη σημαία C του SREG.

ROL Rd **:** Rd(0) □ C , Rd(n+1) □Ṛd(n) , C □Rd(7)

Περιγραφή: Περιστροφή των bits του καταχωρητή Rd μία θέση προς τα αριστερά μέσω της σημαίας κρατουμένου. Η σημαία C ολισθαίνει στο bit0 και το bit7 ολισθαίνει στη σημαία C.

Αντίστοιχα γίνεται η ολίσθηση προς τα δεξιά με τις εντολές **LSR** Rd, **ROR** Rd.

- **<u>Χειρισμός bit (σημαίας) καταχωρητή κατάστασης</u>**:

BSET s **:** SREG(s) ß 1

Περιγραφή: Ενεργοποίηση σημαίας του καταχωρητή κατάστασης.

BCLR s **:** SREG(s) ß 0

Περιγραφή: Μηδενισμός σημαίας του καταχωρητή κατάστασης.

- **<u>Χειρισμός bit καταχωρητή εισόδου-εξόδου</u>**:

SBI A,s **:** I/O(A,b) ß 1

Περιγραφή: Ενεργοποίηση του bit s ενός καταχωρητή εισόδου-εξόδου.

CBI A,s **:** I/O(A,b) ß 0

Περιγραφή: Μηδενισμός του bit s ενός καταχωρητή εισόδου-εξόδου.

- **<u>Χειρισμός bit κατ/τη μέσω σημαίας T</u>**:

BST Rd,b **:** T ß Rd(b)

Περιγραφή: Αποθήκευση του bit b του καταχωρητή Rd στη σημαία T του SREG.

BLD Rd,b **:** Rd(b) **ß** T

Περιγραφή: Αποθήκευση της σημαίας T του SREG στο bit b του καταχωρητή Rd.

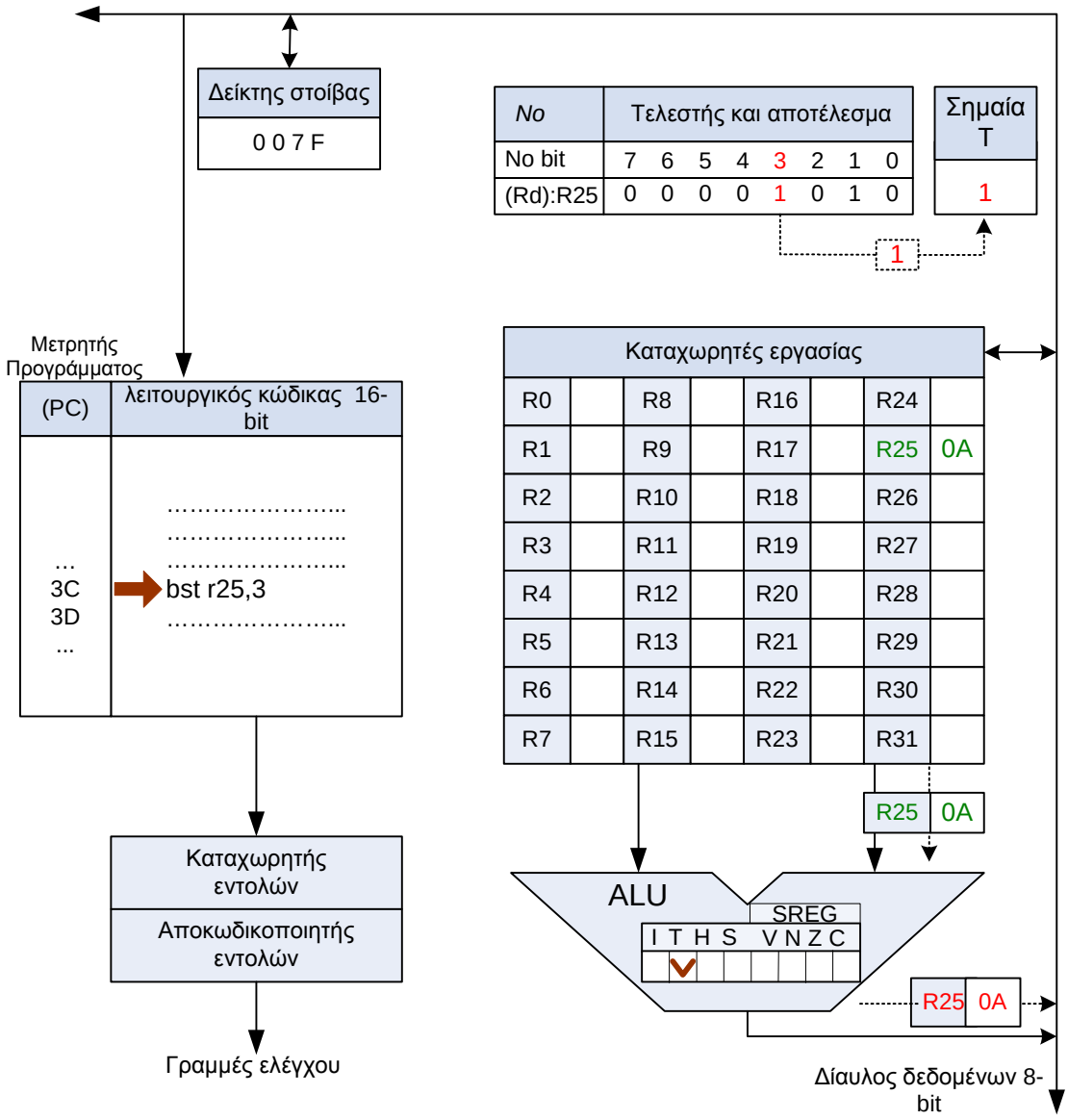
- **<u>Χειρισμός σημαίας καταχωρητή κατάστασης</u>**:

SEx : x **ß** 1

Περιγραφή: Ενεργοποίηση της σημαίας x του καταχωρητή κατάστασης.

CLx : x **ß** 0

Περιγραφή: Μηδενισμός της σημαίας x του καταχωρητή κατάστασης.



Σχήμα 2.12**:** Παράδειγμα εντολής bit.

2.7.4 Εντολές ελέγχου μικροελεγκτή

Πρόκειται για εντολές που ελέγχουν τη λειτουργία του μικροελεγκτή, π.χ. προκαλώντας επανεκκίνηση του watchdog timer, ή κάποια λειτουργία ηρεμίας, ανενεργής κατάστασης ή διακοπή. Οι σημαίες μένουν ανεπηρέαστες.

Εντολές ελέγχου MCU					
NOP		No Operation		Καμία	1
SLEEP		Sleep	Sleep function	Καμία	1
WDR		Watchdog Reset	WDR/timer function	Καμία	1

BREAK		Break	For On-chip Debug Only	Καμία	N/A

Πίνακας 2.3: Εντολές ελέγχου μικροελεγκτή

NOP: Δεν εκτελείτα καμία λειτουργία. Χρησιμοποιείται στη δημιουργία τεχνητών καθυστερήσεων εντός του προγράμματος. Οι καταχωρητές και οι σημαίες μένουν ανεπηρέαστες.

SLEEP: Θέτει το κύκλωμα σε ανενεργό κύκλο που προσδιορίζεται από τον καταχωρητή ελέγχου της MCU.

WDR: Επαναθέτει τον χρονιστή επιτήρησης.

BREAK: Θέτει τη CPU σε λειτουργία Stopped.

Οι εντολές αυτές μας παρέχουν τη δυνατότητα να θέσουμε ή να μηδενίσουμε σημαίες του καταχωρητή κατάστασης, διακοπές και συγκεκριμένα bit καταχωρητών ή θυρών. Επίσης, να ολισθήσουμε κάποιον καταχωρητή μέσω κρατουμένου ή όχι. Πρόκειται για τις εξής εντολές:

Εντολές σε επίπεδο bit και ελέγχου bit					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ß 1	Καμία	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ß 0	Καμία	2
LSL	Rd	Logical Shift Left	Rd(n+1) ß Rd(n), Rd(0) ß 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ß Rd(n+1), Rd(7) ß 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0) ß C, Rd(n+1) ß Rd(n), C ß Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7) ß C, Rd(n) ß Rd(n+1), C ß Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ß Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3-0) ß Rd(7-4) Rd(7-4) ß Rd(3-0)	Καμία	1
BSET	S	Flag Set	SREG(s) ß 1	SREG(s)	1
BCLR	S	Flag Clear	SREG(s) ß 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ß Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ß T	Καμία	1
SEx		Set Flag	x ß 1	X	1

CLx		Clear Flag	x ß 0	X	1

Πίνακας 2.4: Εντολές σε επίπεδο bit και ελέγχου bit

Όπου x μια από τις σημαίες του καταχωρητή κατάστασης C,N,Z,I,S,V,T,H και b το bit0-7 ενός καταχωρητή ή μιας θύρας (3-bit) , s το bit0-7 του καταχωρητή κατάστασης (3-bit) , k μια σταθερή διεύθυνση και P καταχωρητής εισόδου-εξόδου.

Ολίσθηση κατ/τη αριστερά ή δεξιά μέσω κρατουμένου ή όχι:

LSL Rd **:** Rd(n+1) ß Rd(n) , Rd(0) ß 0 ,C ß Rd(7)

Περιγραφή: Ολίσθηση των bits του καταχωρητή Rd μία θέση προς τα αριστερά..Το bit0 μηδενίζεται και το bit7 φορτώνεται στη σημαία C του SREG.

ROL Rd **:** Rd(0) ß C , Rd(n+1) ß Rd(n) , C ß Rd(7)

Περιγραφή: Περιστροφή των bits του καταχωρητή Rd μία θέση προς τα αριστερά μέσω της σημαίας κρατουμένου. Η σημαία C ολισθαίνει στο bit0 και το bit7 ολισθαίνει στη σημαία C.

Αντίστοιχα γίνεται η ολίσθηση προς τα δεξιά με τις εντολές **LSR** Rd, **ROR** Rd.

Χειρισμός bit (σημαίας) καταχωρητή κατάστασης:

BSET s **:** SREG(s) ß 1

Περιγραφή: Ενεργοποίηση σημαίας του καταχωρητή κατάστασης.

BCLR s **:** SREG(s) ß 0

Περιγραφή: Μηδενισμός σημαίας του καταχωρητή κατάστασης.

Χειρισμός bit καταχωρητή εισόδου-εξόδου:

SBI A,s **:** I/O(A,b) ß 1

Περιγραφή: Ενεργοποίηση του bit s ενός καταχωρητή εισόδου-εξόδου.

CBI A,s **:** I/O(A,b) ß 0

Περιγραφή: Μηδενισμός του bit s ενός καταχωρητή εισόδου-εξόδου.

Χειρισμός bit κατ/τη μέσω σημαίας T:

BST Rd,b **:** T ß Rd(b)

Περιγραφή: Αποθήκευση του bit b του καταχωρητή Rd στη σημαία T του SREG.

BLD Rd,b **:** Rd(b) ß T

Περιγραφή: Αποθήκευση της σημαίας T του SREG στο bit b του καταχωρητή Rd.

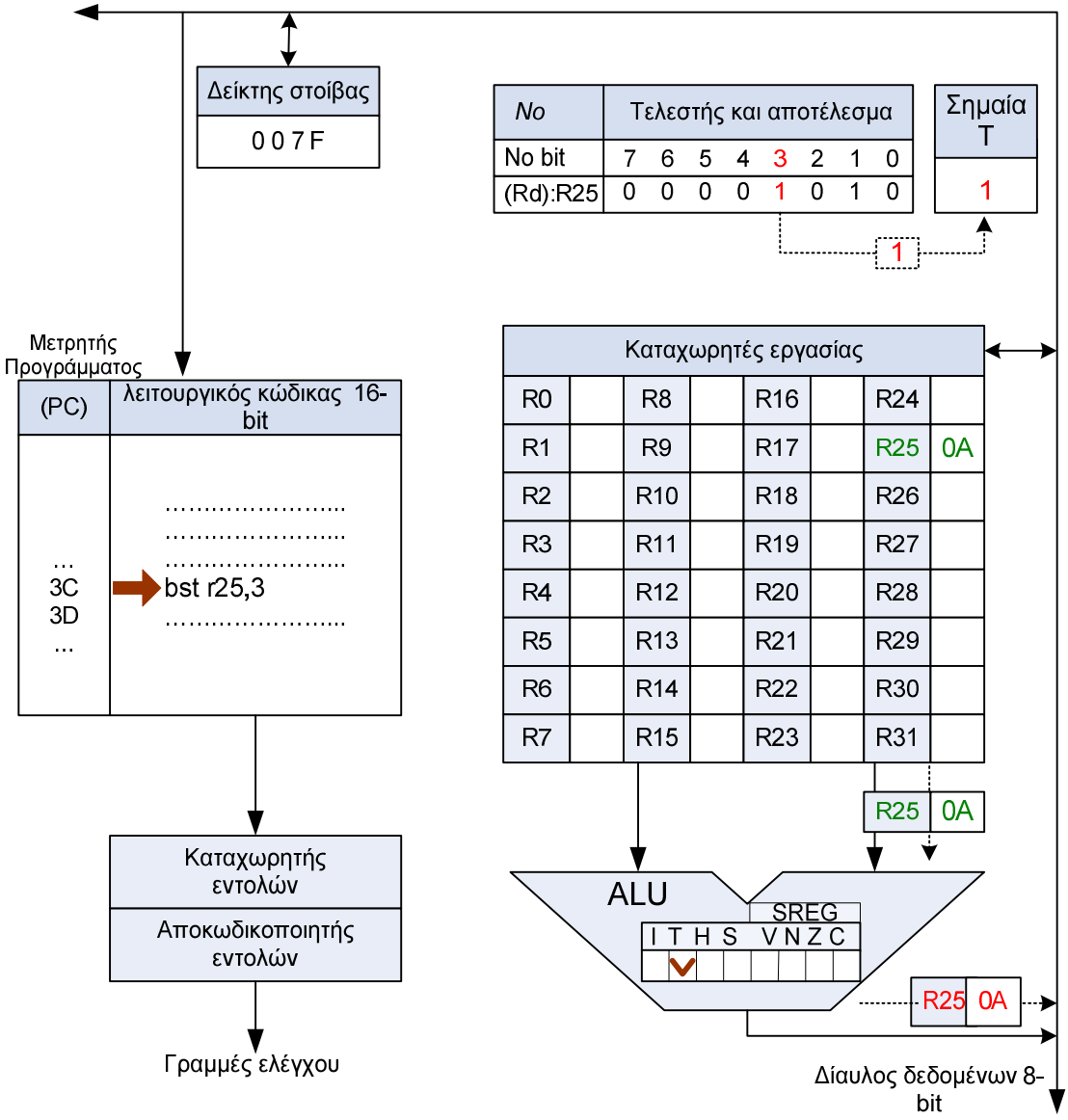
Χειρισμός σημαίας καταχωρητή κατάστασης:

SEx : x ß 1

Περιγραφή: Ενεργοποίηση της σημαίας x του καταχωρητή κατάστασης.

CLx : x ß 0

Περιγραφή: Μηδενισμός της σημαίας x τουκαταχωρητή κατάστασης.



Σχήμα 2.13**:** Παράδειγμα εντολής bit.

2.7.5 Εντολές ελέγχου ροής του προγράμματος και διακλάδωσης

Σε αυτήν την κατηγορία ανήκουν οι εντολές άλματος, παράκαμψης, διακλάδωσης και κλήσης ρουτινών. Πρόκειται για εντολές που αλλάζουν την κανονική ακολουθιακή ροή του προγράμματος. Δηλαδή, όταν εκτελείται μια τέτοια εντολή ο έλεγχος μεταφέρεται σε άλλο σημείο του προγράμματος, αντί να εκτελεστεί η επόμενη εντολή. Το σημείο αυτό μπορεί να είναι σε μια διεύθυνση της μνήμης προγράμματος, η οποία καθορίζεται από το όρισμα της εντολής. Συγκεκριμένα, ο μετρητής προγράμματος (PC-Program Counter) θα πάρει την επιθυμητή τιμή είτε από τον καταχωρητή Z είτε από μια ετικέτα (label).

Οι εντολές παράκαμψης και διακλάδωσης είναι υπό συνθήκη. Αυτό σημαίνει ότι εξετάζεται η τιμή κάποιου bit για να καθοριστεί αν πρέπει να μεταφερθεί ο έλεγχος ή όχι. Συγκεκριμένα, οι εντολές παράκαμψης εξετάζουν την τιμή ενός bit κάποιου καταχωρητή ή μιας θύρας εισόδου-εξόδου, ενώ οι εντολές διακλάδωσης εξετάζουν μία από τις επτά σημαίες του καταχωρητή κατάστασης. Ο μικροελεγκτής έχει μια πληθώρα εντολών διακλάδωσης, γεγονός που συμβάλλει στην αποδοτικότητα και ευελιξία του κώδικα. Παρακάτω παρατίθενται οι προαναφερθείσες εντολές ανά κατηγορία.

2.7.5.1 Εντολές Παράκαμψης

Εντολές παράκαμψης					
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	Καμία	1/ 2/ 3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	Καμία	1/ 2/ 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	Καμία	1/ 2/ 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	Καμία	1/ 2/ 3

Πίνακας 2.5: Εντολές παράκαμψης

Οι εντολές παράκαμψης εξετάζουν την τιμή ενός bit κάποιου καταχωρητή ή μιας θύρας εισόδου-εξόδου για να καθορίσουν αν πρέπει να μεταφερθεί ο έλεγχος ή όχι. Οι σημαίες δεν επηρεάζονται.

SBRC Rr,b **:** Αν Rr(b) = 0 τότε PC ß PC + 2 (ή 3) αλλιώς PC ß PC + 1

Περιγραφή: Αν το bit (b) του καταχωρητή είναι 0 (Rr(b) = 0) τότε παρακάμπτεται η επόμενη εντολή.

SBIC A,b **:** Αν I/O(A,b) = 0 τότε PC ß PC + 2 (or 3) αλλιώς PC ß PC + 1

Περιγραφή: Αν το bit (b) του καταχωρητή εισόδου-εξόδου είναι 0 (A(b) = 0) τότε παρακάμπτεται η επόμενη εντολή.

2.7.5.2 Εντολές διακλάδωσης

Αντίστοιχα, οι εντολές SBRS, SBIS παρακάμπτουν την επόμενη εντολή αν το bit (b) είναι μονάδα.

Εντολές διακλάδωσης					
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC□P̣C+k +1	Καμία	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC□P̣C+k +1	Καμία	1 / 2
BREQ	K	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	Καμία	1 / 2
BRNE	K	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	Καμία	1 / 2
BRCS	K	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	Καμία	1 / 2
BRCC	K	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	Καμία	1 / 2
BRSH	K	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	Καμία	1 / 2
BRLO	K	Branch if Lower	if (C = 1) then PC ← PC + k + 1	Καμία	1 / 2
BRMI	K	Branch if Minus	if (N = 1) then PC ← PC + k + 1	Καμία	1 / 2
BRPL	K	Branch if Plus	if (N = 0) then PC ← PC + k + 1	Καμία	1 / 2
BRGE	K	Branch if Greater or Equal, Signed	if (NV= 0) then PC ← PC + k + 1	Καμία	1 / 2
BRLT	K	Branch if Less Than Zero, Signed	if (NV= 1) then PC ← PC + k + 1	Καμία	1 / 2
BRHS	K	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	Καμία	1 / 2
BRHC	K	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	Καμία	1 / 2
BRTS	K	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	Καμία	1 / 2

BRTC	K	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	Καμία	1 / 2
BRVS	K	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	Καμία	1 / 2
BRVC	K	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	Καμία	1 / 2
BRIE	K	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	Καμία	1 / 2
BRID	K	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	Καμία	1 / 2

Πίνακας 2.6: Εντολές διακλάδωσης

Οι εντολές διακλάδωσης εξετάζουν μία από τις επτά σημαίες του καταχωρητή κατάστασης για να αποφασίσουν αν πρέπει να μεταφερθεί ο έλεγχος ή όχι. Οι σημαίες δεν επηρεάζονται.

Παράδειγμα : BRBC s,k **:** Αν SREG(s) = 0 τότε PC ß PC + k + 1, αλλιώς PC ß PC + 1

Περιγραφή: Σχετικό άλμα υπό συνθήκη. Αν το bit s του καταχωρητή κατάστασης είναι 0 (SREG(s) = 0) τότε εκτελείται άλμα στην διεύθυνση της μνήμης προγράμματος που βρίσκεται k θέσεις μετά σε σχέση με τον μετρητή προγράμματος.

2.7.5.3 Εντολές σύγκρισης

Εντολές σύγκρισης					
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	Καμία	1/ 2/ 3
CP	Rd,Rr	Compare	Rd < Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd < Rr +C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd < K	Z,N,V,C,H	1

Πίνακας 2.7: Εντολές σύγκρισης.

Οι εντολές σύγκρισης μας παρέχουν τη δυνατότητα να συγκρίνουμε δύο καταχωρητές ή έναν καταχωρητή και μια σταθερά.

Παράδειγμα: CPSE Rd,Rr **:** Αν Rd = Rr τότε PC□PC + 2 (ή 3) αλλιώς PC□PC + 1

Περιγραφή: Σύγκριση μεταξύ των καταχωρητών Rd,Rr. Σε περίπτωση ισότητας παρακάμπτεται η επόμενη εντολή.

2.7.5.4 Εντολές άλματος - ρουτινών

Εντολές άλματος- ρουτινών					
RJMP	K	Relative Jump	PC ← PC + k + 1	Καμία	2
IJMP		Indirect Jump to (Z)	PC ← Z	Καμία	2
JMP	K	Direct Jump	PC ← k	Καμία	3
RCALL	K	Relative Subroutine Call	PC ← PC + k + 1	Καμία	3
ICALL		Indirect Call to (Z)	PC ← Z	Καμία	3
CALL	K	Direct Subroutine Call	PC ← k	Καμία	4
RET		Subroutine Return	PC ← STACK	Καμία	4
RETI		Interrupt Return	PC ← STACK	I	4

Πίνακας 2.8: Εντολές άλματος ρουτινών.

Οι επόμενες εντολές μας παρέχουν τη δυνατότητα να εκτελέσουμε άλμα σε επιθυμητή διεύθυνση, να καλέσουμε μια ρουτίνα και να επιστρέψουμε από αυτήν.

Παράδειγμα: RJMP k **:** PC ß PC + k + 1

Περιγραφή: Σχετικό άλμα σε μια διεύθυνση της μνήμης προγράμματος μεταξύ PC - 2K +1 και PC + 2K .

2.8 ΟΜΑΔΕΣ ΕΝΤΟΛΩΝ: ΤΡΟΠΟΙ ΔΙΕΥΘΥΝΣΙΟΔΟΤΗΣΗΣ

Η τυπική μορφή (format) των εντολών του μικροελεγκτή AVR αποτελείται από τα παρακάτω τέσσερα τμήματα. Ενδέχεται ένα ή και περισσότερα να είναι κενό.

επιγραφή	λειτουργία	όρισμα(τα)	Σχόλια

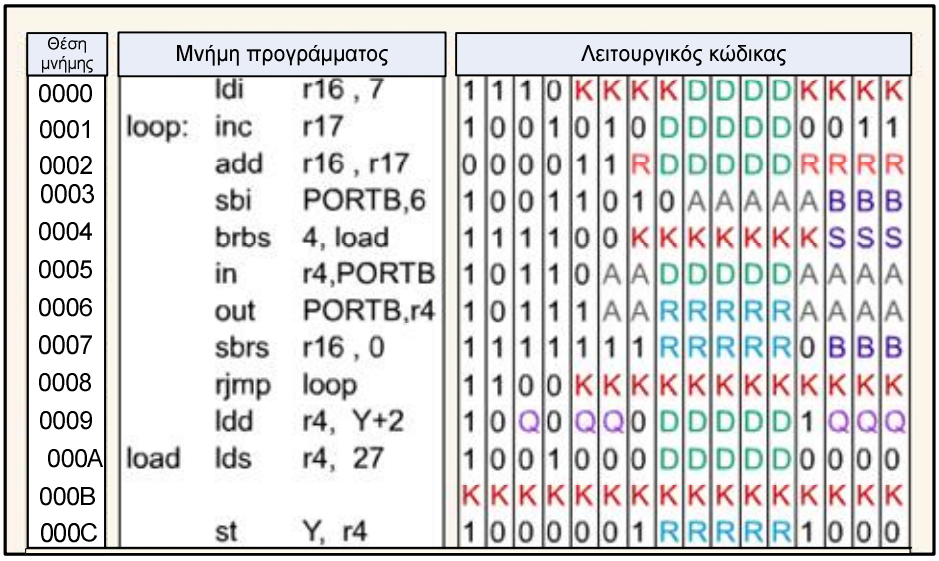
Τμήμα επιγραφής (label): Η επιγραφή αντιστοιχεί σε μια θέση του προγράμματος και είναι καθοριστικής σημασίας για τη ροή του προγράμματος. Κάθε ετικέτα πρέπει να αρχίζει με αλφαβητικό χαρακτήρα και να ακολουθείται από άνω και κάτω τελεία. Δεν επιτρέπεται να χρησιμοποιηθεί η ίδια ετικέτα πάνω από μία φορά.

Τμήμα λειτουργίας (mnemonic): Σε αυτό το πεδίο βρίσκεται το μνημονικό όνομα της εντολής (είτε της ψευδοεντολής). Τα γράμματα του μνημονικού ονόματος παραμπέμπουν στη λειτουργία της εντολής. Το όνομα της εντολής πρέπει να ακολουθείται από ένα τουλάχιστον κενό.

Τμήμα ορισμάτων (operands): Τα ορίσματα καθορίζουν την προέλευση και τον προορισμό των δεδομένων, τη ροή του προγράμματος,κλπ. Το πλήθος και η σειρά των ορισμάτων εξαρτάται από την εκάστοτε εντολή και διαχωρίζονται μεταξύ τους με κόμμα.

Τμήμα σχολίων : Η χρήση σχολίων βοηθά στην κατανόηση του προγράμματος και είναι προαιρετική. Τα σχόλια πρέπει να έπονται του ελληνικού ερωτηματικού(;) και δεν λαμβάνονται υπόψη από τον μεταφραστή.

Ένα παράδειγμα του format των εντολών είναι το εξής:



Θέση μνήμης	Μνήμη προγράμματος			Λειτουργικός κώδικας
0000		ldi	r16 , 7	1 1 1 0 K K K K D D D D K K K K
0001	loop:	inc	r17	1 0 0 1 0 1 0 D D D D D 0 0 1 1
0002		add	r16 , r17	0 0 0 0 1 1 R D D D D D R R R R
0003		sbi	PORTB,6	1 0 0 1 1 0 1 0 A A A A A B B B
0004		brbs	4, load	1 1 1 1 0 0 K K K K K K K S S S
0005		in	r4,PORTB	1 0 1 1 0 A A D D D D D A A A A
0006		out	PORTB,r4	1 0 1 1 1 A A R R R R R A A A A
0007		sbrs	r16 , 0	1 1 1 1 1 1 1 R R R R R 0 B B B
0008		rjmp	loop	1 1 0 0 K K K K K K K K K K K K
0009		ldd	r4, Y+2	1 0 Q 0 Q Q 0 D D D D D 1 Q Q Q
000A	load	lds	r4, 27	1 0 0 1 0 0 0 D D D D D 0 0 0 0
000B				K K K K K K K K K K K K K K K K
000C		st	Y, r4	1 0 0 0 0 0 1 R R R R R 1 0 0 0

Σχήμα 2.14**:** Τυπική μορφή (format) εντολών.

Όπου: K: ένας σταθερός αριθμός ή μια διεύθυνση (εύρος 7-,8-,12-,16- bit ανάλογα με εντολή)

R: καταχωρητής προέλευσης (4-,5-bit)

D: κατ/της προορισμού (4-,5-bit)

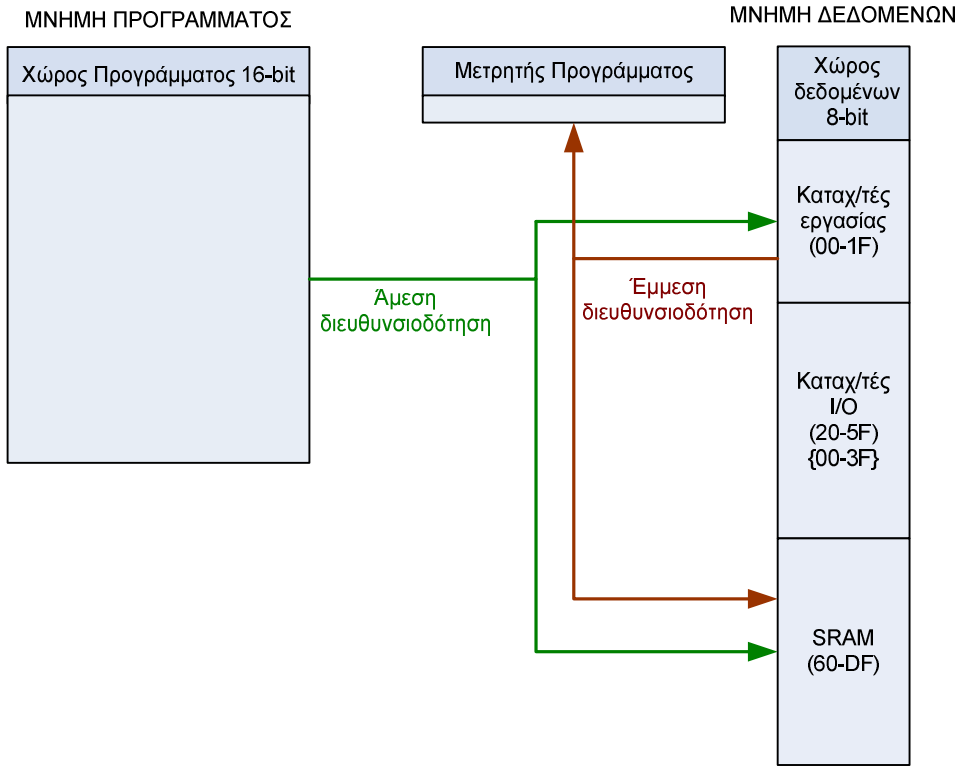
A: διεύθυνση I/O (5-,6- bit)

S: bit κατ/τη κατάστασης (3-bit)

B: bit κατ/τη εργασίας ή κατ/τη I/O (3-bit)

Q: μετατόπιση για άμεση διευθυνσιοδότηση (6-bit)

Οι εντολές της γλώσσας των μικροελεγκτών ταξινομούνται με βάση τον τρόπο που επιτυγχάνεται πρόσβαση στα δεδομένα και τις πράξεις που εκτελούνται με αυτά. Πρόκειται για τους τρόπους διευθυνσιοδότησης του προγράμματος και των δεδομένων. Η διευθυνσιοδότηση διακρίνεται σε άμεση και έμμεση, όπως φαίνεται και στο σχήμα που ακολουθεί.



Σχήμα 2.15: Τρόποι διευθυνσιοδότησης.

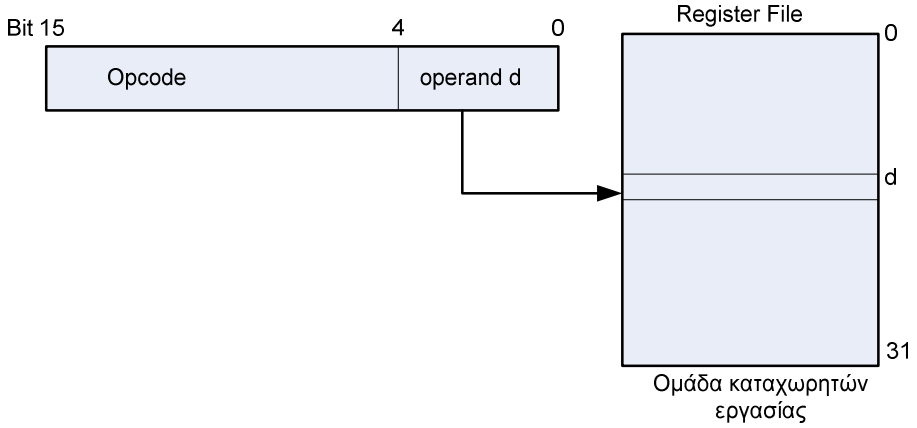
Οι εντολές περιλαμβάνουν 2 τμήματα. Το πρώτο (operation code ή opcode) αποτελεί το λειτουργικό κώδικα που πληροφορεί τον επεξεργαστή για τις ενέργειες που πρέπει να εκτελεστούν. Το δεύτερο τμήμα προσδιορίζει τον τελεστή (operand) για τον οποίο θα λειτουργήσει ο κώδικας. Η αναφορά τελεστέου σε μια εντολή είτε περιέχει την πραγματική τιμή του τελεστέου (άμεση) είτε μια αναφορά προς την διεύθυνση του τελεστέου. Στις διάφορες

ομάδες εντολών χρησιμοποιείται μια μεγάλη ποικιλία τρόπων διευθυνσιοδότησης. Το πεδίο ή πεδία διευθύνσεων σε μια συνηθισμένη μορφοποίηση εντολής είναι σχετικά μικρά. Θα θέλαμε να μπορούμε να αναφερόμαστε σε μια μεγάλη περιοχή θέσεων της κύριας μνήμης ή της ιδεατής μνήμης. Για να πετύχουμε αυτότον σκοπό, έχει χρησιμοποιηθεί μια ποικιλία τεχνικών διευθυνσιοδότησης. Οι πιο συνηθισμένες τεχνικες διευθυνσιοδότησης είναι:

1. Άμεση
2. Απευθείας
3. Έμμεση
4. Καταχωρητή
5. Έμμεση καταχωρητή
6. Μετατόπισης
7. Σωρού

2.8.1. Άμεση διευθυνσιοδότηση ενός καταχωρητή εργασίας

Η εντολή διαβάζει τα περιεχόμενα του καταχωρητή, εκτελεί τις πράξεις με αυτά και αποθηκεύει το αποτέλεσμα στον ίδιο καταχωρητή. Ο καταχωρητής είναι ένας από τους 32 καταχωρητές εργασίας R0-R31 που διαθέτουν το αντίστοιχο τμήμα μνήμης(register file). Στο σχήμα φαίνονται οι θέσεις προέλευσης και προορισμού των δεδομένων. Με Rd συμβολίζεται ο καταχωρητής εργασίας που αποτελεί τη θέση προέλευσης των δεδομένων και τη θέση προορισμού του αποτελέσματος.

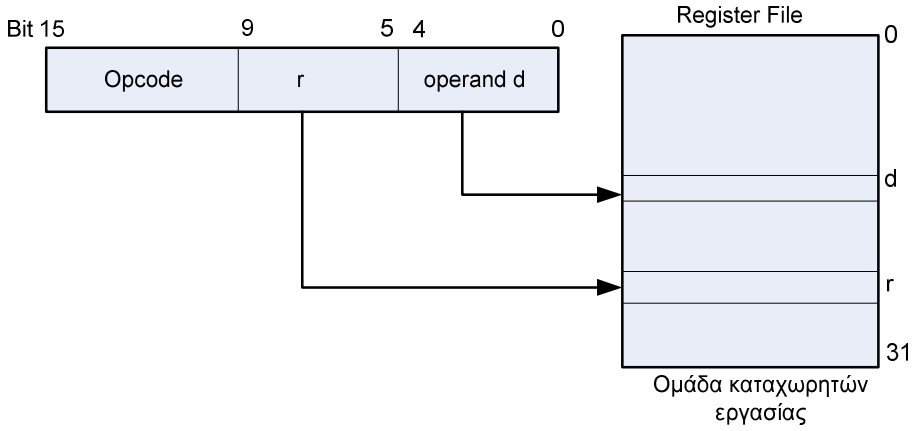


Σχήμα 2.16**:** Απευθείας διευθυνσιοδότηση καταχωρητή.

Πρόκειται για την απλούστερη μορφή διευθυνσιοδότησης, όπου ο τελεστέος στην πραγματικότητα βρίσκεται στην εντολή. Το πλεονέκτημα της άμεσης διευθυνσιοδότησης είναι ότι, για να πάρουμε τον τελεστέο, δεν χρειάζεται άλλη αναφορά μνήμης εκτός από την προσαγωγή της εντολής, πράγμα που εξοικονομεί μια περίοδο μνήμης ή μνήμης cache στον κύκλο εντολής Το μειονέκτημα είναι ότι το μέγεθος του αριθμού περιορίζεται στο μέγεθος του πεδίου διεύθυνσης το οποίο, στις περισσότερες ομάδες εντολών, είναι μικρό σε
σύγκριση με το μήκος λέξης.

2.8.2. Άμεση διευθυνσιοδότηση δύο καταχωρητών εργασίας

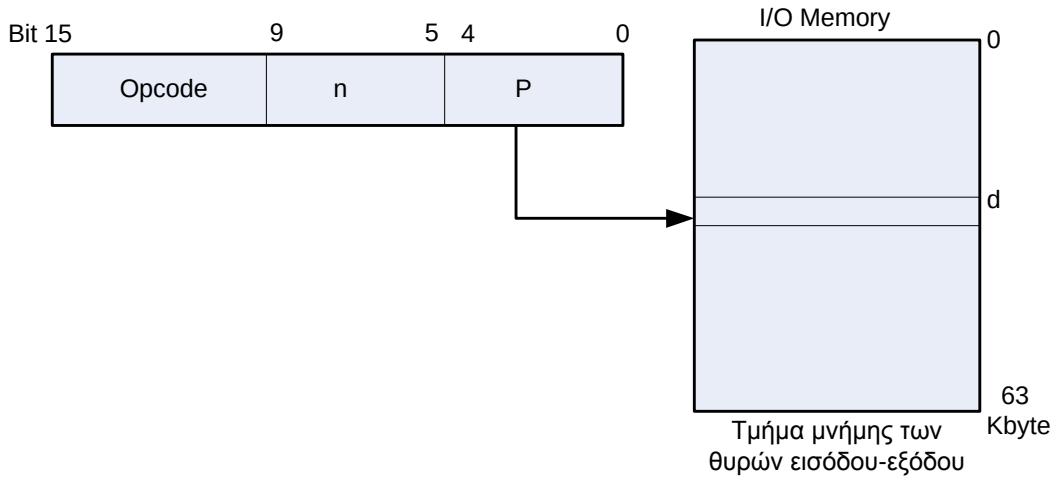
Η εντολή διαβάζει τα περιεχόμενα των δυο καταχωρητών, εκτελεί την πράξη μεταξύ των δεδομένων και αποθηκεύει το αποτέλεσμα στον καταχωρητή προορισμού Rd. Στο σχήμα φαίνονται οι θέσεις προέλευσης και προορισμού των δεδομένων. Με Rr συμβολίζεται ο καταχωρητής εργασίας που αποτελεί τη θέση προέλευσης των δεδομένων.



Σχήμα 2.17: Απευθείας διευθυνσιοδότηση δύο καταχωρητών

2.8.3. Απευθείας Διευθυνσιοδότηση των θυρών Εισόδου - Εξόδου

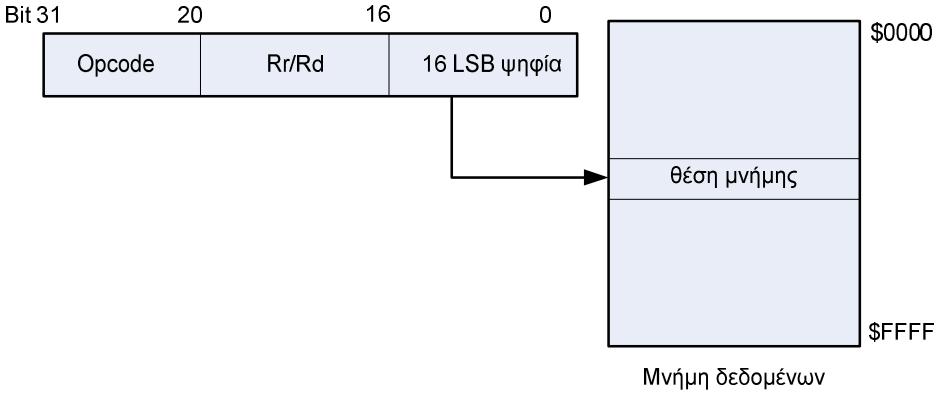
Μια πολύ απλή μορφή διευθυνσιοδότησης είναι η απευθείας διευθυνσιοδότηση, όπου το πεδίο διεύθυνσης περιέχει την ενεργό διεύθυνση του τελεστέου. Ο προφανής περιορισμός είναι ότι προσφέρει μόνο περιορισμένο χώρο διεύθυνσης. Η πρόσβαση στις θέσεις μνήμης που αντιστοιχούν οι θύρες εισόδου-εξόδου (I/O memory) επιτυγχάνεται με 2 εντολές: της IN Rd,PortAddress και της OUT PortAddress,Rr, όπου PortAddress είναι η διεύθυνση του καταχωρητή εισόδου-εξόδου.



Σχήμα 2.18: Απευθείας διευθυνσιοδότηση των θυρών εισόδου-εξόδου

2.8.4. Άμεση διευθυνσιοδότηση μνήμης δεδομένων

Η πρόσβαση στις θέσεις μνήμης επιτυγχάνεται με εντολές μήκους 2 λέξεων, όπου η μία λέξη (16 bits) αντιστοιχεί στη διεύθυνση της μνήμης δεδομένων. Άρα, ο χώρος μνήμης όπου επιτυγχάνεται πρόσβαση είναι 64Kbyte.

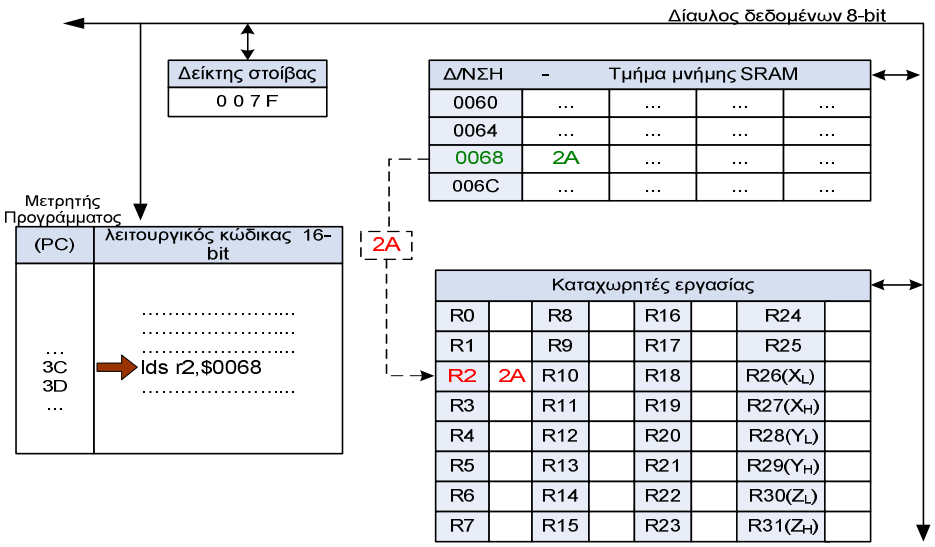


Σχήμα 2.19: Απευθείας διευθυνσιοδότηση της μνήμης δεδομένων

Παραδείγματα τέτοιων εντολών είναι:

STS K,Rr

LDS Rd,K, όπου K διεύθυνση μήκους 16 bits.

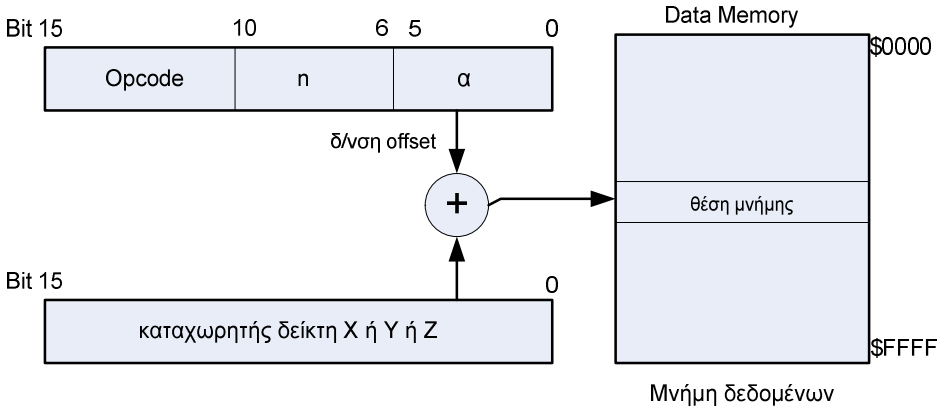


Σχήμα 2.20: Παράδειγμα απευθείας διευθυνσιοδότησης της μνήμης δεδομένων

Εκτός από την απευθείας πρόσβαση με χρήση διευθύνσεων, υπάρχει και η έμμεση πρόσβαση με χρήση κατ/τη δείκτη, όπως θα δούμε παρακάτω.

2.8.5. Έμμεση διευθυνσιοδότηση μνήμης δεδομένων

Οι εντολές αυτές έχουν μήκος μιας λέξης και χρησιμοποιούν έναν καταχωρητή δείκτη (έναν εκ των X,Y,Z), το περιεχόμενο του οποίου αντιστοιχεί στη βασική διεύθυνση της μνήμης δεδομένων. Για εξαρτήματα με SRAM, ο χώρος δεδομένων αποτελείται από το register file, τη μνήμη I/O, την εσωτερική SRAM (και την εξωτερική αν υπάρχει). Για εξαρτήματα χωρίς SRAM, ο χώρος δεδομένων αποτελείται μόνο από το register file. Η EEPROM έχει ξεχωριστό χώρο διευθύνσεων.



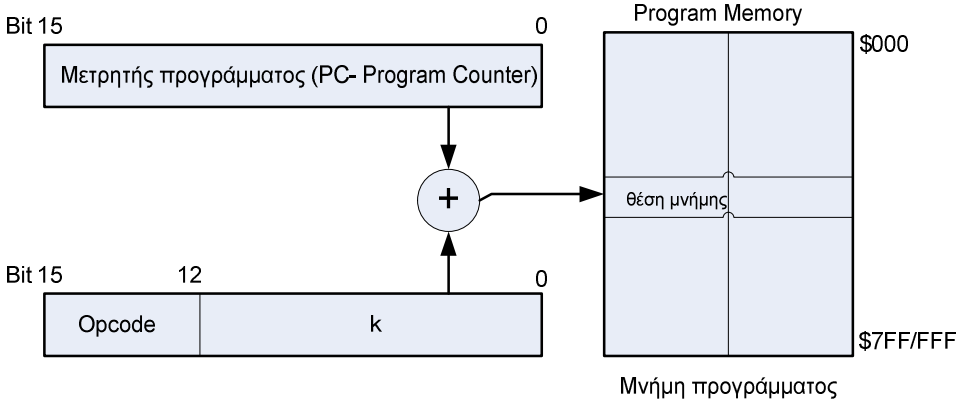
Σχήμα 2.21:Έμμεση διευθυνσιοδότηση της μνήμης δεδομένων.

2.8.6 Έμμεση διευθυνσιοδότηση του καταχωρητή

Τέλος, μπορούμε να έχουμε έμμεση φόρτωση του καταχωρητή Rd με τα περιεχόμενα της θέσης μνήμης (Y+q), ώστε να επιτύχουμε πρόσβαση σε παραπέρα θέση μνήμης. Πρόκειται δηλαδή για μετατόπιση q (εύρους 6-bit) για έμμεση διευθυνσιοδότηση. Έτσι με τον τρόπο αυτό επιτυγχάνουμε κυμαινόμενη πρόσβαση σε επόμενες θέσεις μνήμης, π.χ. σε πίνακα δεδομένων. Ακόμα πιο σημαντικό είναι η πρόσβαση χρησιμοποιώντας μια συγκεκριμένη διεύθυνση έναρξης (offset) σε έναν κατ/τη δείκτη. Στην περίπτωση αυτή, η διεύθυνση αποθηκεύεται στον κατ/τη δείκτη και μια σταθερή τιμή προστίθεται στη διεύθυνση ώστε να επιτυγχάνουμε πρόσβαση για εγγραφή/ ανάγνωση.

2.8.7. Σχετική διευθυνσιοδότηση μνήμης προγράμματος

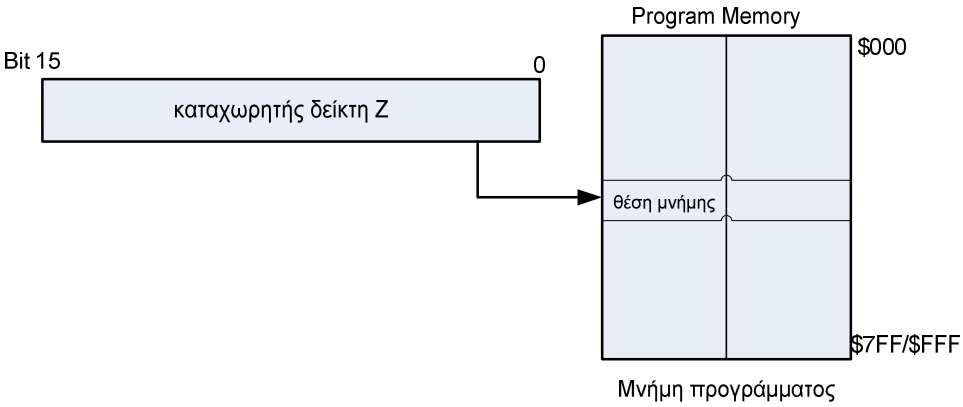
Οι εντολές αυτές επιτυγχάνουν πρόσβαση σε χώρο μνήμης προγράμματος και είναι του τύπου RCALL,RJMP, όπου χρησιμοποιείται μια μετατόπιση +/- 2K στο περιεχόμενο του μετρητή προγράμματος.



Σχήμα 2.22**:**Σχετική διευθυνσιοδότηση της μνήμης προγράμματος

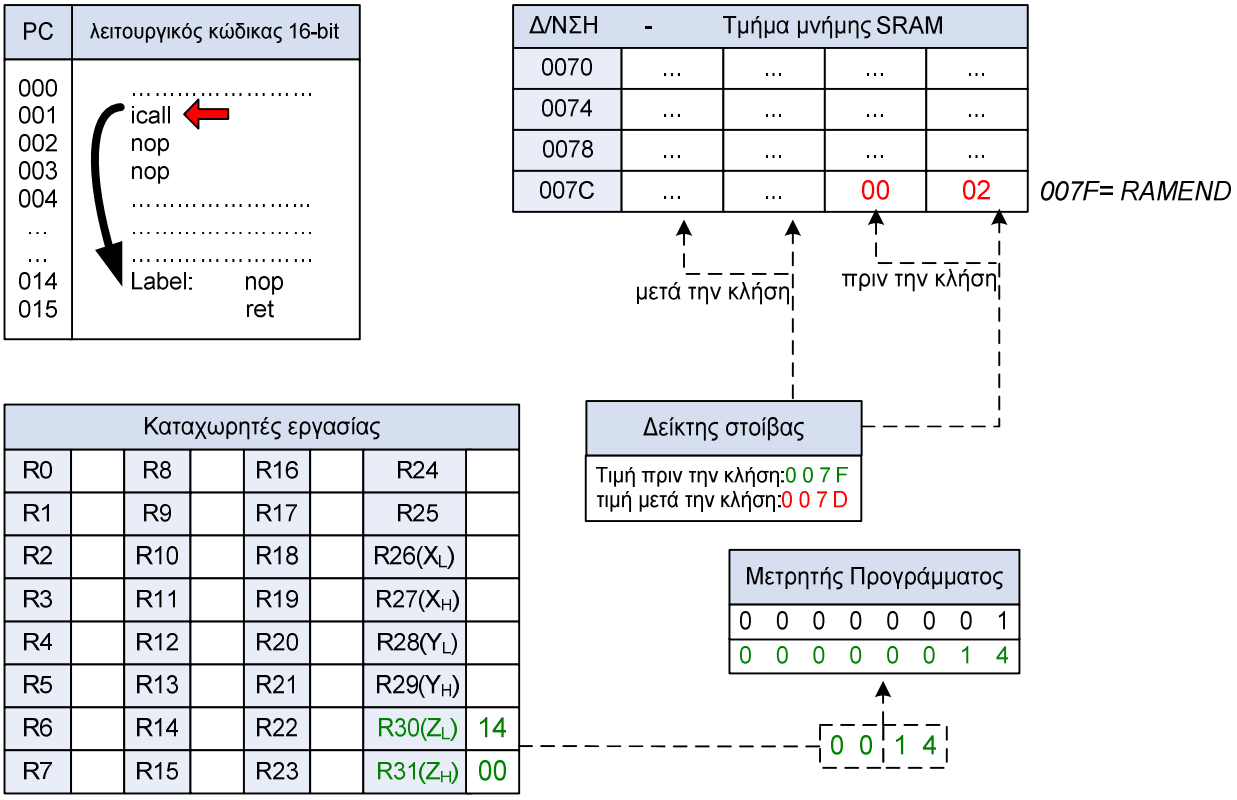
2.8.8 Έμμεση διευθυνσιοδότηση μνήμης προγράμματος

Οι εντολές αυτές επιτυγχάνουν πρόσβαση σε χώρο μνήμης προγράμματος έως 64Kbytes με χρήση του καταχωρητή Z, ως δείκτη μιας θέσης μνήμης προγράμματος. Παραδείγματα τέτοιων εντολών είναι οι ICALL,IJMP.



Σχήμα 2.23**:** Έμμεση διευθυνσιοδότηση της μνήμης προγράμματος

Η εντολή *icall* καλεί έμμεσα μια υπορουτίνα η διεύθυνση της οποίας αντιστοιχεί στο περιεχόμενο του καταχωρητή δείκτη Z . Δηλαδή, το περιεχόμενο του Z φορτώνεται στον μετρητή προγράμματος (PC Z(15:0)), όπως φαίνεται στο σχήμα:



Σχήμα 2.24**:**Παράδειγμα έμμεσης διευθυνσιοδότησης της μνήμης προγράμματος

2.8.9. Έμμεση Διευθυνσιοδότηση

Στην απευθείας διευθυνσιοδότηση, το μήκος του πεδίου διεύθυνσης είναι συνήθως μικρότερο από το μήκος λέξης, πράγμα που περιορίζει το πεδίο τιμών της διεύθυνσης. Μια λύση είναι να κάνουμε το πεδίο διεύθυνσης να αναφέρεται στην διεύθυνση μιας λέξης στην μνήμη, η οποία με την σειρά της θα περιέχει μια διεύθυνση πλήρους μήκους του τελεστέου. Αυτό είναι γνωστό ως *έμμεση διευθυνσιοδότηση* (indirect addressing).

2.8.10. Διευθυνσιοδότηση Καταχωρητή

Η διευθυνσιοδότηση καταχωρητή είναι παρόμοια με την απευθείας διευθυνσιοδότηση. Η μόνη διαφορά είναι ότι το πεδίο διεύθυνση αναφέρεται σε καταχωρητή αντί σε διεύθυνση κύριας μνήμης. Συνήθως, ένα πεδίο διεύθυνσης που αναφέρεται σε καταχωρητές θα έχει από 3 έως 5 bit, έτσι ώστε να μπορεί να γίνει αναφορά σε σύνολο από 8 μέχρι 32 καταχωρητών γενικής χρήσης. Τα πλεονεκτήματα της διευθυνσιοδότησης καταχωρητή είναι ότι:

1. στην εντολή χρειάζεται μόνο ένα μικρό πεδίο διεύθυνσης, και
2. δεν χρειάζεται αναφορά σε μνήμη.

2.8.11. Έμμεση Διευθυνσιοδότηση Καταχωρητή

Όπως η διευθυνσιοδότηση καταχωρητή είναι αντίστοιχη με την απευθείας διευθυνσιοδότηση, η έμμεση διευθυνσιοδότηση καταχωρητή είναι αντίστοιχη της έμμεσης διευθυνσιοδότησης. Και στις δύο περιπτώσεις, η μόνη διαφορά είναι αν το πεδίο διεύθυνσης αναφέρεται σε θέση μνήμης ή σε καταχωρητή.

2.8.12 Διευθυνσιοδότηση Μετατόπισης

Ένας πολύ ισχυρός τρόπος διευθυνσιοδότησης συνδυάζει τις δυνατότητες της απευθείας διευθυνσιοδότησης και την έμμεση διευθυνσιοδότησης καταχωρητή. Η διευθυνσιοδότηση μετατόπισης ζητά από την εντολή να έχει δύο πεδία διεύθυνσης, από τα οποία το ένα τουλάχιστον να είναι συγκεκριμενο. Η τιμή που περιέχεται στο ένα πεδίο διεύθυνσης (τιμή = A) χρησιμοποιειται απευθείας. Το άλλο πεδίο διεύθυνσης, ή εννοούμενη αναφορά που βασίζεται σε opcode, αναφέρεται σε καταχωρητή του οποίου τα περιεχόμενα προστίθενται στο A για να δημιουργηθεί μια ενεργός διεύθυνση.

2.8.13. Διευθυνσιοδότηση Σωρού

Ο τελευταίος τρόπος διευθυνσιοδότησης είναι η διευθυνσιοδότηση σωρού. Σωρός είναι μια γραμμική διάταξη θέσεων. Μερικές φορές ονομαζεται *κατάλογος ώθησης προς τα κάτω* (pushdown list) ή *ουρά τελευταίο μέσα πρώτο έξω* (last-in-first-out queue). Ο σωρός είναι ένα κρατημένο κομμάτι θέσεων. Αντικείμενα προσκολούνται στην κορυφή του σωρού έτσι ώστε, σε οποιαδήποτε δεδομένη στιγμή, το κομμάτι να είναι μερικώς γεμάτο. Με το σωρό σχετίζεται ένας ενδείκτης του οποίου η τιμή είναι η διεύθυνση της κορυφής του σωρού. Εναλλακτικά, τα επάνω δύο στοιχεία του σωρού μπορεί να βρίσκονται σε καταχωρητές της CPU, οπότε στην περίπτωση αυτή ο ενδείκτης σωρού θα ανφέρεται στο τρίτο στοιχείο του σωρού. Ο ενδείκτης σωρού κρατείται σε ένα καταχωρητή. Έτσι, οι αναφορές σε θέσεις του σωρού στην μνήμη θα είναι στην πραγματικότητα έμμεσες διευθύνσεις καταχωρητή.

2.9 Θύρες εισόδου-εξόδου

Οι θύρες ενός μικροελεγκτή είναι οι πύλες επικοινωνίας της κεντρικής μονάδας επεξεργασίας με εσωτερικές και εξωτερικές μονάδες υλικού και λογισμικού. Ο επεξεργαστής επικοινωνεί με τις διάφορες μονάδες γράφοντας ή διαβάζοντας στις αντίστοιχες θύρες-καταχωρητές. Όπως συμβαίνει π.χ. στους χρονιστές, στην σειριακή θύρα, στην παράλληλη θύρα και στο πλέον χρησιμοποιούμενο καταχωρητή κατάστασης.

Εκτός από τους 32 καταχωρητές εργασίας υπάρχουν 64 καταχωρητές, όχι όλοι διαθέσιμοι στους διάφορους τύπους μικροελεγκτών. Έχουν μια δεδομένη διεύθυνση (fixed address) μέσω της οποίας επικοινωνεί η κεντρική μονάδα επεξεργασίας. Η διεύθυνση είναι ανεξάρτητη του μοντέλου AVR, αλλά δεν είναι απαραίτητο να τη θυμόμαστε καθώς χρησιμοποιούνται ψευδώνυμα (π.χ. EQU, PORTx, 0x18). Η πληροφορία αυτή περιέχεται στα αρχεία (header files) του κατασκευαστή. Οι καταχωρητές των θυρών εισόδου-εξόδου είναι οι εξής:

<u>Καταχωρητής δεδομένων της θύρας PORTx</u>

Πρόκειται για έναν καταχωρητή των 8-bit με δυνατότητα εγγραφής και ανάγνωσης. Όταν η θύρα ρυθμιστεί ως έξοδος μπορούμε να γράψουμε στους ακροδέκτες της θύρας μέσω του καταχωρητή PORTx. Ενώ όταν η θύρα ρυθμιστεί ως είσοδος μπορούμε να διαβάσουμε τα δεδομένα του καταχωρητή PORTx. Η ρύθμιση των ακροδεκτών ως είσοδοι είτε ως έξοδοι γίνεται με τη βοήθεια του καταχωρητή DDRx. Η αρχική τιμή του καταχωρητή δεδομένων είναι $00.

Καταχωρητής κατεύθυνσης της θύρας PORTx

Ο καταχωρητής κατεύθυνσης δεδομένων(Data Direction Register of portx) ρυθμίζει την κατεύθυνση των ακροδεκτών της θύρας PORTx. Εάν γράψουμε την τιμή ‘0’ σε κάποια από τα bits του καταχωρητή DDRx, τότε οι αντίστοιχοι ακροδέκτες της θύρας PORTx γίνονται είσοδοι. Αντίστοιχα, αν γράψουμε την τιμή ‘1’, τότε οι αντίστοιχοι ακροδέκτες της θύρας PORTx γίνονται έξοδοι.

Η αρχική τιμή του καταχωρητή δεδομένων είναι $00. Επίσης, κάθε bit μπορεί να τροποποιηθεί με τις εντολές SBI,CBI.

Διεύθυνση ακροδεκτών εισόδου της θύρας PORTx

Η διεύθυνση ακροδεκτών εισόδου (PINx) δεν είναι καταχωρητής. Αυτή η διεύθυνση επιτρέπει την πρόσβαση στις λογικές στάθμες κάθε ακροδέκτη της θύρας PORTx. Δηλαδή όταν διαβάζουμε το περιεχόμενο του PINB διαβάζουμε τις λογικές στάθμες κάθε ακροδέκτη της θύρας PORTx

Καταχωρητής δεδομένων της θύρας PORTD (PORTD)

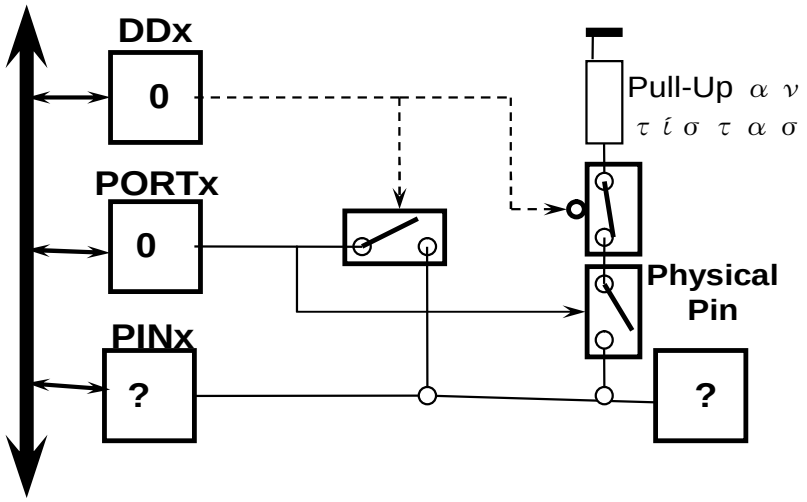
Ομοίως με τον καταχωρητή PORTx.

Καταχωρητής κατεύθυνσης της θύρας PORTD (DDRD)

Ομοίως με τον καταχωρητή DDRx.

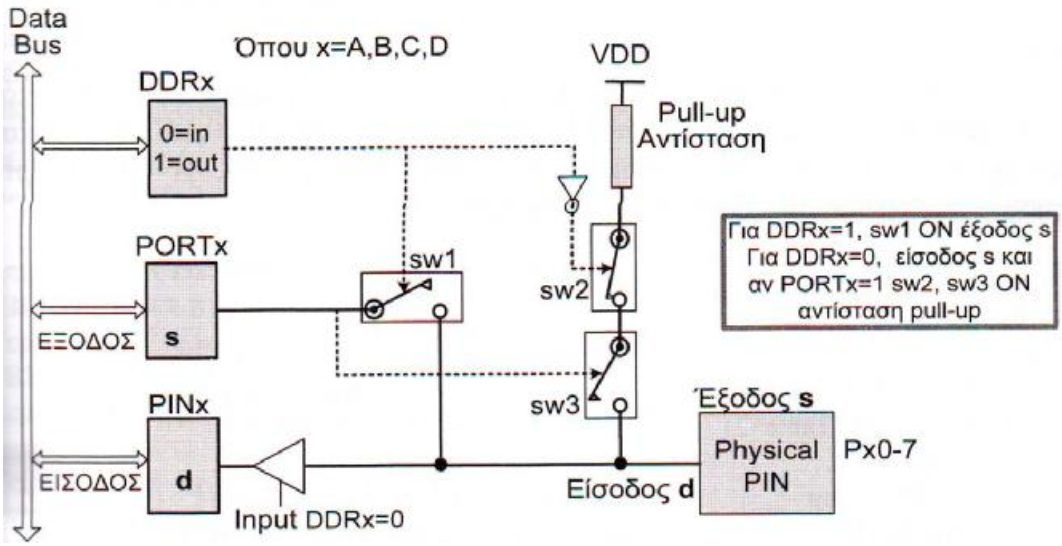
Διεύθυνση ακροδεκτών εισόδου της θύρας PORTD (PIND)

Ομοίως με τη θύρα PINx.



Σχήμα 2.25: Καταχωρητές θυρών ειδόδου-εξόδου.

Οι πιο απλές εντολές εισόδου-εξόδου είναι η ‘in’ και ‘out’. Η ‘in’ διαβάζει την τιμή μιας θύρας ή του κατ/τη ενός εσωτερικού περιφερειακού (χρονιστή, UART) και τη σώζει σε έναν καταχωρητή (in r16, PinD). Η ‘out’ γράφει την τιμή ενός κατ/τη σε μια θύρα ή στον κατ/τη ενός εσωτερικού περιφερειακού (out PortD, r16).



Σχήμα 2.26: Καταχωρητές Θυρών Εισόδου - Εξόδου

Επίσης, υπάρχουν οι εντολές ‘sbi’ και ‘cbi’ για χειρισμό μεμονωμένων ψηφίων μιας θύρας και οι sbic , sbis για τον έλεγχο των ψηφίων μιας θύρας I/O. Αυτές οι τέσσερις εντολές δεν εφαρμόζονται σε όλους τους κατ/τες εισόδου- εξόδου. Ο ακόλουθος κώδικας εξηγεί την εγγραφή και ανάγνωση σε μια θύρα με χρήση των παραπάνω εντολών:

1. **<u>Μετατροπή θύρας σε είσοδο</u>**

```
LDI  R12, 0b00000000        ; τα ψηφία της θύρας PORTD
OUT  DDRD, R12              ; γίνονται είσοδοι
```

Εναλλακτικά η PORTD μετατρέπεται σε είσοδο

```
LDI  R12, 0x00              ; τα ψηφία της θύρας PORTD
OUT  DDRD, R12              ; γίνονται είσοδοι
```

ή

```
CLR R21                     ; τα ψηφία της θύρας PORTD γίνονται είσοδοι
OUT  DDRD, R21
```

2. Μετατροπή θύρας σε έξοδο

```
LDI  R18, 0b11111111        ; τα ψηφία της θύρας PORTB
OUT  DDRB, R18              ; γίνονται έξοδοι
```

Εναλλακτικά η PORTB μετατρέπεται σε έξοδο

```
LDI  R22, 0xFF              ; τα ψηφία της θύρας PORTB
OUT  DDRB, R22              ; γίνονται έξοδοι
SER R19 ; τα ψηφία της θύρας PORTB γίνονται έξοδοι
OUT  DDRB, R19
```

- **Είσοδος δεδομένου: ανάγνωση από θύρα εισόδου- εξόδου με την εντολή 'in'.**

```
LDI  R12,  0b11111111       ; ενεργοποιούμε τις αντιστάσεις πρόσδεσης στη θύρα
OUT PORTD,R12               ; ανάγνωση των λογικών σταθμών στις εισόδους
IN R12, PIND                ; της θύρας PORTD και αποθήκευση αποτελέσματος στον
R12
```

- **Έξοδος δεδομένου: εγγραφή σε θύρα με την εντολή 'out'.**

```
LDI  R12, 0b00010100    ; το 2ο και 4ο ψηφίο της θύρας PORTD γίνονται έξοδοι
OUT  DDRD, R12
LDI  R12, 0b00010100          ;οδήγηση του 2ου και 4ου ψηφίου σε υψηλή λογική στάθμη(1)
OUT  PORTD, R12
LDI  R12, 0b00000000    ;οδήγηση του 2ου και 4ου ψηφίου σε χαμηλή λογική στάθμη(0)
OUT  PORTD, R12
```

- **Εγγραφή σε θύρα με τις εντολές 'sbi' και 'cbi'**:

```
SBI DDRD,2                    ; το 2ο ψηφίο της θύρας PORTD γίνεται έξοδος
SBI PORTD,2                   ; οδήγηση του 2ου ψηφίου σε υψηλή λογική στάθμη(1)
CBI PORTD,2                   ; οδήγηση του 2ου ψηφίου σε χαμηλή λογική στάθμη(0)
```

2.10 ΣΤΟΙΒΑ

Η στοίβα χρησιμοποιείται από την αριθμητική λογική μονάδα (ALU) για αποθήκευση διευθύνσεων επιστροφής από υπορουτίνες. Η στοίβα χρειάζεται έναν δείκτη στοίβας (SP) και χώρο μνήμης στην SRAM.

Καταχωρητής δείκτης στοίβας (stack pointer- SPH/SPL)

Σε περίπτωση εξυπηρέτησης μιας ρουτίνας διακοπής ή κατά την κλήση μιας υπορουτίνας, η διεύθυνση επιστροφής του κυρίου προγράμματος αποθηκεύεται στη στοίβα. Το περιεχόμενο του καταχωρητή δείκτη στοίβας είναι η διεύθυνση της μνήμης SRAM που αντιστοιχεί στην κορυφή της στοίβας. Ο καταχωρητής έχει μήκος 1 byte για μικροελεγκτές μεγέθους μνήμης ως 256 bytes και μήκος 2 bytes για μνήμες άνω των 256 bytes με συμβολισμό SPH-SPL. Κατά την εισαγωγή μιας τιμής στην στοίβα (διαδικασία push) η τιμή του δείκτη στοίβας ελαττώνεται κατά μια μονάδα. Αντίστροφα, η ανάκτηση μιας αποθηκευμένης τιμής από τη στοίβα (διαδικασία pop), αυξάνει κατά μια μονάδα το δείκτη στοίβας. Λειτουργίες στοίβας απαιτούνται όταν πρόκειται να κληθεί υπορουτίνα ή διακοπή. Τότε η πραγματική δ/νση φυλάσεται στη στοίβα για επιστροφή στο σημείο του προγράμματος μετά την εξυπηρέτηση της ρουτίνας.

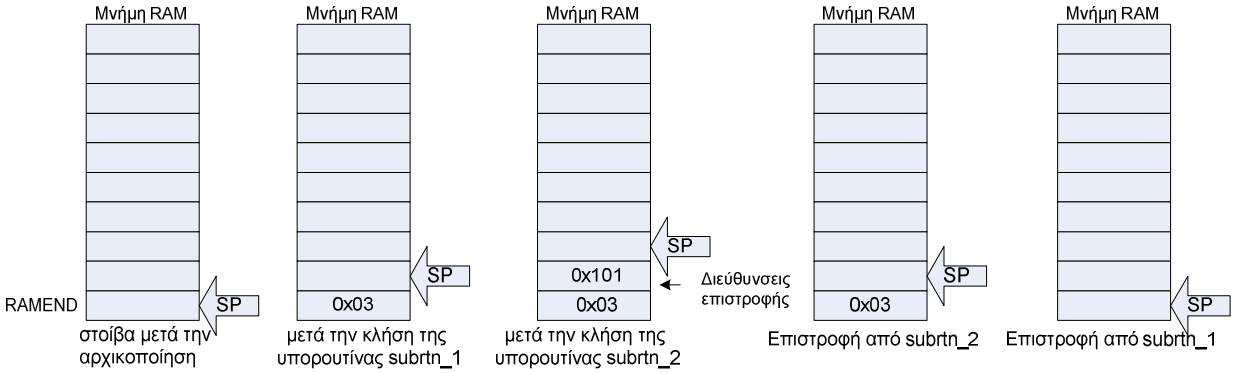
Η στοίβα βρίσκεται στο τέλος της ενσωματωμένης SRAM, ονόματι RAMEND και ορίζεται στο αρχείο "xxxxdef.inc" για τον αντίστοιχο τύπο επεξεργαστή, οπότε δεν μας απασχολεί η πραγματική της τιμή. Αν ένα byte σταλεί στη στοίβα, εγγράφεται σε θέση της μνήμης SRAM και ο 16-bit κατ/της δείκτη στοίβας SPH:SPL μειώνεται στην επόμενη χαμηλότερη θέση στοίβας. Περαιτέρω αποστολή bytes φέρνει το δείκτη πλησιέστερα προς την αρχή της SRAM. Αν ένα byte ληφθεί από τη στοίβα, τότε ο δείκτης αυξάνει προτού διαβαστεί η τιμή. Η τελευταία τιμή που αποθηκεύεται θα διαβαστεί πρώτη.Πρόκειται, λοιπόν, για δομή τύπου Last-In-First-Out (LIFO). Μια δ/νση SRAM έχει μήκος 16 bits, οπότε ο κατ/της SPL κρατά τα 8 LSB bits και ο SPH τα 8 MSB bits της δ/νσης. Η στοίβα αποτελεί την πλέον συνηθισμένη χρήση της SRAM. Αφού αρχικοποιήσουμε τη στοίβα, μένει να μάθουμε πως υλοποιούνται οι διαδικασίες αποθήκευσης και ανάκτησης δεδομένων. Το περιεχόμενο ενός κατ/τη σώζεται στη στοίβα με την εντολή:

PUSH Register

Δεν μας απασχολεί η δ/νση προορισμού και το γεγονός ότι ο κατ/της δείκτη μειώθηκε κατά μία μονάδα. Το μόνο που μας ενδιαφέρει είναι ότι η ανάκτηση των δεδομένων γίνεται με την εντολή: POP Register

(Σχόλιο: Η στοίβα είναι χρήσιμη όταν δεν υπάρχουν διαθέσιμοι κατ/τες, όταν τα δεδομένα ζητούνται επανειλημμένως στο πρόγραμμα και όταν εκτελούνται υπορουτίνες. Διαφορετικά η αποθήκευση κατ/των στη στοίβα είναι άχρηστη και σπαταλά πόρους του επεξεργαστή.)

Σε περίπτωση κλήσης μιας υπορουτίνας, η δ/νση επιστροφής στο κυρίως πρόγραμμα (δηλαδή η τιμή του μετρητή προγράμματος) σώζεται στη στοίβα (διαδικασία PUSH) και έπειτα εκτελείται το άλμα στην υπορουτίνα. Στο τέλος της εκτέλεση της υπορουτίνας, η δ/νση επιστροφής ανακτάται από τη στοίβα (διαδικασία POP) και φορτώνεται στον μετρητή προγράμματος. Τέλος, η στοίβα εξυπηρετεί τις διακοπές υλικού(hardware interrupts). Οι διακοπές σταματούν την κανονική εκτέλεση του προγράμματος και καλούνται ειδικές ρουτίνες για την εξυπηρέτηση τους. Έπειτα το πρόγραμμα επιστρέφει στη θέση όπου έγινε η διακοπή για την εκτέλεση του υπόλοιπου προγράμματος. Η δ/νση επιστροφής αποθηκεύεται στη στοίβα. Οι διευθύνσεις επιστροφής κατά την κλήση των ρουτινών αποθηκεύονται στη στοίβα όπως φαίνεται στο επόμενο σχήμα:



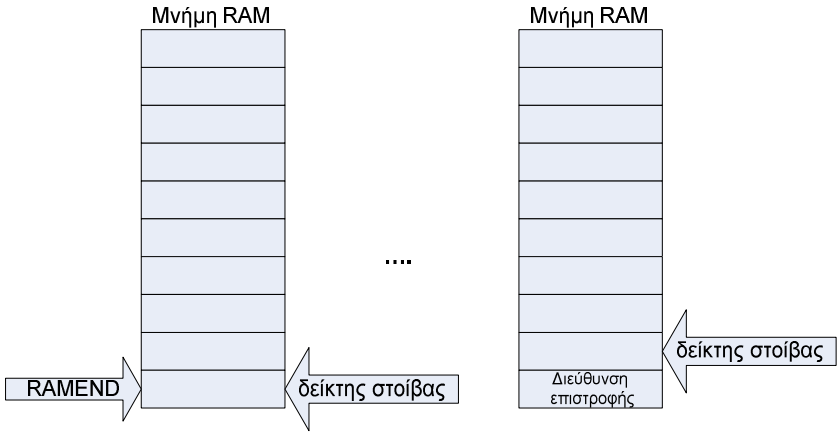
Σχήμα 2.27**:** Διαμόρφωση στοίβας

Όπου RAMEND αντιστοιχεί στη τελευταία διαθέσιμη θέση μνήμης SRAM και χρησιμοποιείται για την αρχικοποίηση της στοίβας. Οι εντολές push και pop πρέπει να χρησιμοποιούνται ισορροπημένα. Αυτό σημαίνει ότι κάθε push θα αντιστοιχεί σε μια pop. Διαφορετικά μετά την εντολή επιστροφής θα έχουμε απρόσμενη συμπεριφορά.

2.11 Υπορουτίνες

Οι υπορουτίνες είναι ανεξάρτητα τμήματα κώδικα τα οποία καλούμε κατά τη διάρκεια του κυρίου προγράμματος για την εκτέλεση κάποιας συγκεκριμένης λειτουργίας. Έπειτα επιστρέφουμε από αυτές, ενώ μπορούμε να τις καλούμε όποτε χρειάζεται. Έτσι κάνουμε οικονομία στη μνήμη προγράμματος και δίνουμε πιο απλή και κατανοητή μορφή στο πρόγραμμα μας. Μία υπορουτίνα ξεκινά με μια ετικέτα που αντιστοιχεί στο όνομα της και τελειώνει με την εντολή επιστροφής (ret). Η κλήση της γίνεται με την εντολή 'rcall'. Αυτή η εντολή εκτελεί άλμα σε μια σχετική δ/νση, έχει μήκος 2 bytes και απαιτεί 3 περιόδους ρολογιού. Το μειονέκτημα είναι ότι η υπορουτίνα πρέπει να βρίσκεται εντός +/- 2k λέξεων. Άλλη εντολή είναι η 'call'. Αυτή εκτελεί άλμα σε απόλυτη δ/νση, γι'αυτό έχει μήκος 4 bytes, και απαιτεί 4 περιόδους ρολογιού. Όμως επιτυγχάνει πρόσβαση σε όλο το χώρο κώδικα, γεγονός απαραίτητο σε μοντέλα με μνήμη προγράμματος άνω των 8kB. Οι 8k AVRs χρειάζονται μόνο τις 'rjmp' και 'rcall', αφού όλες οι δ/νσεις είναι προσβάσιμες με άλματα +/- 2k λέξεων. Επίσης, υπάρχει η 'icall' για έμμεση κλήση μέσω της δ/νσης στο ζεύγος κατ/των Z.

Για να γνωρίζουμε που βρίσκονται οι διευθύνσεις κλήσης και επιστροφής μιας υπορουτίνας πρέπει να θέσουμε το δείκτη στοίβας (SP). Ο SP συνήθως τίθεται στη τελευταία θέση μνήμης (RAMEND) κατά την αρχικοποίηση. Όταν η δ/νση επιστροφής αποθηκεύεται, ο SP δείχνει τη θέση μνήμης που προηγείται της αποθηκευμένης δ/νσης. Τα παραπάνω φαίνονται στο σχήμα:



Σχήμα 2.28: Λειτουργία στοίβας

Σε περίπτωση σύνδεσης εξωτερικής μνήμης SRAM, ο δείκτης τίθεται στη τελευταία θέση της εσωτερικής μνήμης, διότι η πρόσβαση στην εσωτερική μνήμη είναι ταχύτερη.

2.12 Μακροεντολές

Οι μακροεντολές είναι τμήματα κώδικα που γράφονται και ελέγχονται μια φορά και εισάγονται στο κυρίως πρόγραμμα μέσω του ονόματος τους. Οι μακροεντολές, όπως και οι υπορουτίνες, είναι χρήσιμες όταν χρειάζεται να γράψουμε πανομοιότυπα ή παρόμοια τμήματα κώδικα μέσα στο κύριο πρόγραμμα. Συνεπώς, γλιτώνουμε χρόνο, δίνουμε πιο απλή και κατανοητή μορφή στο πρόγραμμα μας και αποφεύγουμε τα άλματα που απαιτεί μια υπορουτίνα. Όμως, σε αντίθεση με τις υπορουτίνες, αυξάνει η απαίτηση σε μνήμη προγράμματος. Οπότε για μεγάλα τμήματα κώδικα ή όταν υπάρχει έλλειψη μνήμης προγράμματος προτιμούνται οι υπορουτίνες. Μια μακροεντολή δέχεται ως δέκα παραμέτρους, οι οποίες συμβολίζονται ως @0-@9 και βρίσκονται εντός του ορισμού της. Όταν καλούμε μια μακροεντολή οι παράμετροι διαχωρίζονται με κόμμα.

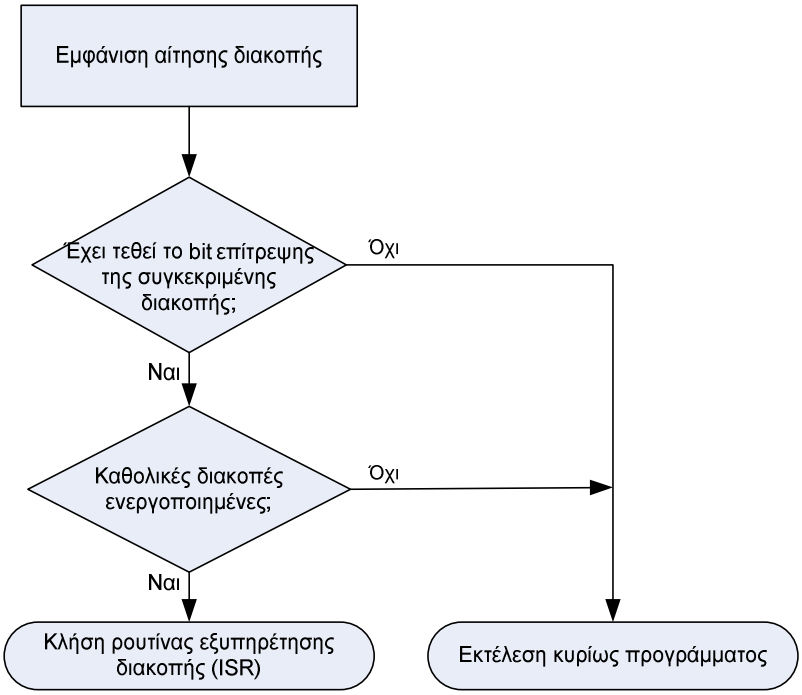
2.13 Εσωτερικές Διακοπές

Οι διακοπές, όπως δηλώνει το όνομα τους, διακόπτουν την κανονική ροή του προγράμματος. Η εμφάνιση μιας διακοπής μας πληροφορεί ότι προέκυψε ένα ιδιαίτερο γεγονός του επεξεργαστή ή σε κάποια περιφερειακή μονάδα, όπως η υπερχείλιση ενός καταχωρητή, το πάτημα ενός διακόπτη, η λήψη δεδομένων σε μια θύρα κλπ, που πρέπει να αντιμετωπιστεί μέσω της εκτέλεσης κάποιου τμήματος κώδικα. Αυτός ο κώδικας συνιστά την ρουτίνα εξυπηρέτησης διακοπής (ISR-Interrupt Service Routine). Τα βήματα κατά την εμφάνιση μιας διακοπής είναι τα εξής:

- Μια περιφερειακή μονάδα (μέσω μιας σημαίας) ειδοποιεί το σύστημα ότι προέκυψε διακοπή.
- Η εντολή που βρίσκεται υπό εκτέλεση στον καταχωρητή εντολών (IR) ολοκληρώνεται.
- Η διεύθυνση της επόμενης εντολής του κυρίου προγράμματος σώζεται στη στοίβα.
- Η διεύθυνση της ρουτίνας εξυπηρέτησης διακοπής φορτώνεται στον μετρητή προγράμματος (PC-Program Counter) μέσω μιας εντολής icall, rcall κλπ, ώστε να εκτελεστεί η $1^{η}$ εντολή της ISR.
- Οι εντολές της ρουτίνας εξυπηρέτησης διακοπής εκτελούνται διαδοχικά μέχρι και την τελευταία εντολή τύπου reti, όπου η διεύθυνση επιστροφής φορτώνεται από τη στοίβα και η σημαία ολικών διακοπών τίθεται.
- Η εκτέλεση του κυρίου προγράμματος συνεχίζεται κανονικά.

Με απλά λόγια αρκεί να θυμόμαστε ότι όταν προκύπτει μια διακοπή, καλείται το αντίστοιχο διάνυσμα διακοπής και ο επεξεργαστής εκτελεί τον απαραίτητο κώδικα από μια συγκεκριμένη διεύθυνση. Έπειτα, η ροή λειτουργίας επιστρέφει στο κύριο πρόγραμμα, συγκεκριμένα στο σημείο όπου εμφανίστηκε η διακοπή. Οι μικροελεγκτές AVR διαθέτουν ένα ολοκληρωμένο σύνολο διακοπών με δυνατότητα ενεργοποίησης-απενεργοποίησης. Αυτό σημαίνει ότι μπορούμε να (απ)ενεργοποιούμε κάποιες λειτουργίες των περιφερειακών μονάδων κατά βούληση.

Κατά την εκτέλεση μιας ρουτίνας εξυπηρέτησης διακοπής ενδέχεται να αλλοιωθούν οι σημαίες του καταχωρητή κατάστασης (SREG) καθώς επίσης και κάποιοι κατ/τες εργασίας. Το γεγονός αυτό ίσως επηρεάσει την ομαλή λειτουργία του κυρίου προγράμματος. Για το λόγο αυτό οι προηγούμενοι κατ/τες αποθηκεύονται στην αρχή της ρουτίνας και ανακτώνται στο τέλος της (διαδικασίες PUSH και POP). Επίσης, ενδέχεται κατά τη διάρκεια εκτέλεσης μιας ρουτίνας εξυπηρέτησης διακοπής να προκύψει νέα διακοπή. Για να αποφύγουμε μια τέτοια κατάσταση απενεργοποιούμε τις καθολικές διακοπές του συστήματος μέσω της εντολής CLI στην αρχή της ρουτίνας και τις ενεργοποιούμε στο τέλος μέσω της εντολής SEI. Δηλαδή, μηδενίζουμε ή θέτουμε το bit GIE (Global Interrupt Enable) του SREG αντιστοίχως. Το επόμενο απλοποιημένο διάγραμμα ροής απεικονίζει την εξυπηρέτηση μιας διακοπής:



Σχήμα 2.29**:** Διάγραμμα ροής για ISR

Ο επόμενος πίνακας περιλαμβάνει τα διανύσματα διακοπών και τη λειτουργία τους για τρία διαφορετικά μοντέλα μικροελεγκτών:

Πίνακας 2.9**:** Διανύσματα διακοπών

Διανύσματα διακοπών στη μνήμη προγράμματος ενός μικροελεγκτή				
Όνομα	Διεύθυνση μνήμης προγράμματος			Σχόλια
	AT90S851	AT90S2313	AT90S2323	
Reset	$0000	$0000	$0000	Διαχείριση επανεκκίνησης
EXT_INT0	$0001	$0001	$0001	Διαχ/ση εξωτερικής διακοπής IRQ0
EXT_INT1	$0002	$0002	-	Διαχ/ση εξωτερικής διακοπής IRQ1
TIMER1_CAPT	$0003	$0003	-	Διαχ/ση διακοπής σε λειτουργία σύλληψης του χρονιστή timer1
TIMER1_COMPA	$0004	-	-	Διαχ/ση διακοπής σε λειτουργία συγκριτή Α του χρονιστή timer1
TIMER1_COMPB	$0005	-	-	Διαχ/ση διακοπής σε λειτουργία συγκριτή Β του χρονιστή timer1
TIMER1_OVF	$0006	$0005	-	Διαχ/ση διακοπής υπερχείλισης του χρονιστή timer1
TIMER0_OVF	$0007	$0006	$0002	Διαχ/ση διακοπής υπερχείλισης του χρονιστή timer0

SPI_STC	$0008	-	-	Διαχ/ση διακοπής κατά την ολοκλήρωση της μετάδοσης από τη μονάδα SPI
UART_RXC	$0009	$0007	-	Διαχ/ση διακοπής κατά την ολοκλήρωση της λήψης από τη μονάδα UART
UART_DRE	$000A	$0008	-	Διαχ/ση διακοπής κατά την εκκένωση του καταχ/τή UDR της μονάδας UART
UART_TXC	$000B	$0009	-	Διαχ/ση διακοπής κατά την ολοκλήρωση της εκπομπής από τη μονάδα UART
ANA_COMP	$000C	-	-	Διαχείριση διακοπής του αναλογικού συγκριτή

2.14 Χρονιστής επιτήρησης (WATCH DOG TIMER)

Πρόκειται για ένα ειδικά ελεγχόμενο χρονιστή που τροφοδοτείται από έναν ξεχωριστό ταλαντωτή (on-chip oscillator: που τρέχει στο 1Mhz με τυπική τιμή τάσης V_{CC} = 5V για το μοντέλο atmega16).Ο WDT είναι μια βοηθητική μονάδα ανάκτησης της κανονικής λειτουργίας του συστήματος. Δηλαδή, όταν ο έλεγχος του προγράμματος χαθεί εντός ενός ατέρμονα βρόχου ή σε περίπτωση εσφαλμένης εκτέλεσης του προγράμματος εφαρμογής (π.χ. αλλοίωση τιμής του μετρητή προγράμματος), τότε ο WDT προκαλεί επανατοποθέτηση (reset) του μικροελεγκτή. Το χρονικό διάστημα λειτουργίας του WDT ωσότου προκύψει επανατοποθέτηση, ρυθμίζεται θέτοντας κατάλληλες τιμές στον διαιρέτη συχνότητας Watchdog Timer prescaler για τα διάφορα επίπεδα τροφοδοσίας, όπως δείχνει ο παρακάτω πίνακας. Επιλέγοντας έναν από τους οκτώ

διαφορετικούς παλμούς ρολογιού καθορίζουμε την περίοδο επανατοποθέτησης. Εάν αυτή η περίοδος εκπνεύσει χωρίς να μηδενιστεί ο WDT, τότε ο ελεγκτής επανατοποθετείται και εκτελείται το διάνυσμα Reset. Για την αποφυγή άσκοπης απενεργοποίησης του χρονιστή επιτήρησης, εκτελείται μια διαδικασία απεμπλοκής, που περιγράφεται παρακάτω. Η εντολή WDR επανατοποθετεί τον Watchdog Timer. Επίσης επανατοποθετείται όταν είναι απενεργοποιημένος και όταν προκύψει ένα Chip Reset.

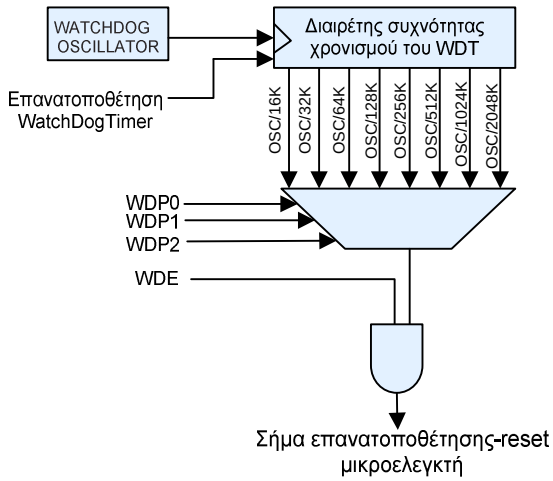
Καταχωρητής ελέγχου του Watchdog Timer (WDTCR)

Η σημασία των bits ελέγχου είναι η εξής:

- Bit 4 (WDTOE): Η λειτουργία του watchdog timer διακόπτεται, όταν το ψηφίο WDTOE(watchdog turn off enable) τίθεται ‘1’ και μηδενίζεται το ψηφίο WDE.
- Bit 3 (WDE):Ο watchdog timer τίθεται σε λειτουργία.
- Bit 2,Bit 1,Bit 0 (WDP2, WDP1, WDP0): Ο συνδυασμός αυτών των bits καθορίζει την τιμή μέτρησης του χρονιστή που οδηγεί σε υπέρβαση (timeout).Οι τιμές prescaling και οι αντίστοιχες τιμές υπέρβασης φαίνονται στο σχήμα.

Bits	7	6	5	4	3	2	1	0
Ελέγχου				WDTOE	WDE	WDP2	WDP1	WDP0

Πίνακας 2.10: Σημασία των bits



Σχήμα 2.30 **:** Δημιουργία επανατοποθέτησης από WDT

Πίνακας 2.11**:** Επιλογή Watchdog Timer Prescale

WDP 2	**WDP 1**	**WDP 0**	**Αριθμός κύκλων του WDT Oscillator**	**Τυπικό Time-out για VCC = 3.0V**	**Τυπικό Time-out για VCC = 5.0V**
0	0	0	16K (16,384)	17.1 ms	16.3 ms
0	0	1	32K (32,768)	34.3 ms	32.5 ms
0	1	0	64K (65,536)	68.5 ms	65 ms
0	1	1	128K (131,072)	0.14 s	0.13 s
1	0	0	256K (262,144)	0.27 s	0.26 s
1	0	1	512K (524,288)	0.55 s	0.52 s
1	1	0	1,024K (1,048,576)	1.1 s	1.0 s
1	1	1	2,048K (2,097,152)	2.2 s	2.1 s

Για απενεργοποίηση-απεμπλοκή του WDT εκτελούμε τον κώδικα (υποθέτοντας ότι ελέγχουμε τις διακοπές ,π.χ. απενεργοποιώντας καθολικές διακοπές, ώστε να μην προκύψει διακοπή κατά την εκτέλεση του κώδικα):

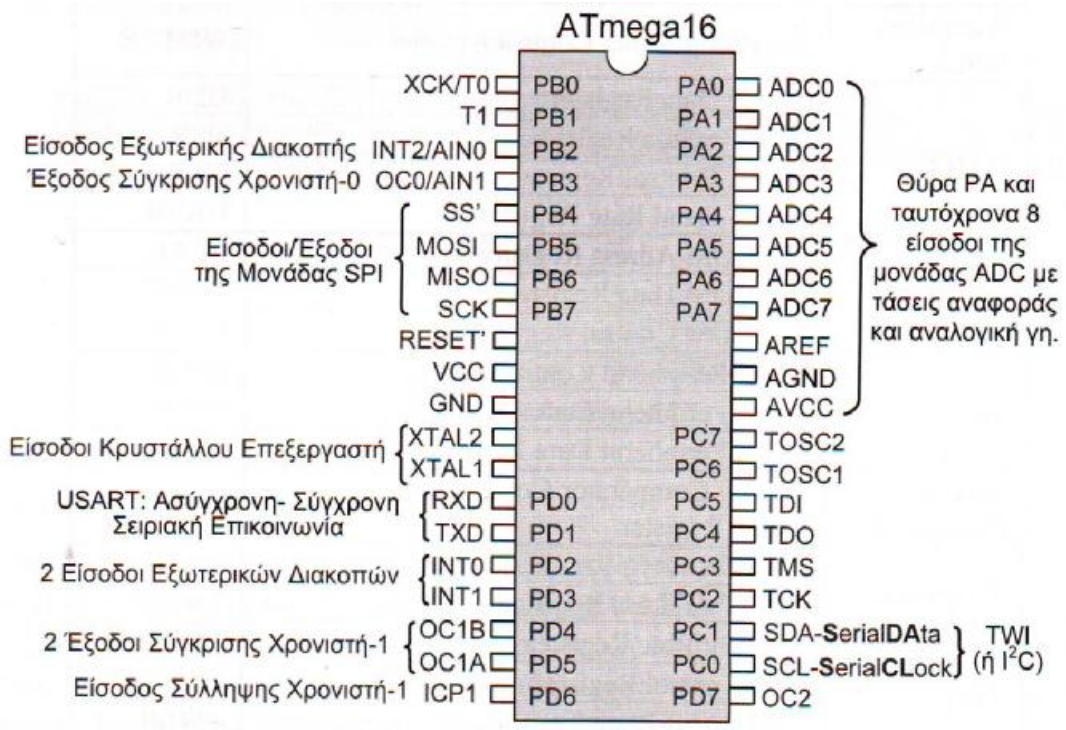
```
WDT_off:

WDR                          ; Reset WDT

in r16, WDTCR                ; Εγγραφή λογικού 1 στα  WDTOE και WDE
```

```
ori r16, (1<<WDTOE)|(1<<WDE)

out WDTCR, r16



ldi r16, (0<<WDE)                ; απενεργοποίηση WDT

out WDTCR, r16

ret
```

ΚΕΦΑΛΑΙΟ 3^{Ο} - ΠΕΡΙΦΕΡΕΙΑΚΑ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ AVR

Οι Μικροελεγκτές AVR διαθέτουν ένα μεγάλο πλήθος περιφερειακών μονάδων, το οποίο όμως πάντα εξαρτάται από τον ειδικότερο τύπο του μικροελεγκτή. Τα περιφερειακά στα οποία αναφερόμαστε είναι τα εξής:

- Μονάδα UART
- Χρονιστές (Timer 0, Timer 1, Timer 2) και Επιτήρησης (WatchDog Timer)
- Μονάδα EEPROM και μονάδα SPI
- Μονάδα Αναλογικού Συγκριτή
- Μονάδα Two – Wire Serial Interface
    - Μονάδα ADC (Analog to Digital Converter)



Σχήμα 3.1: Ακροδέκτες Περιφερειακών ATmega

Στο παραπάνω σχήμα εμφανίζονται οι ακροδέκτες που αντιστοιχούν στις εισόδους – εξόδους των περιφερειακών της νεότερης οικογένειας Mega των μικροελεγκτών AVR που περιλαμβάνει όλη την ποικιλία των περιφερειακών, ενώ ταυτόχρονα διαθέτει αρκετά μεγάλη on chip μνήμη προγράμματος και δεδομένων.

3.1 Η Μονάδα Ασύγχρονης Σειριακής Επικοινωνίας UART

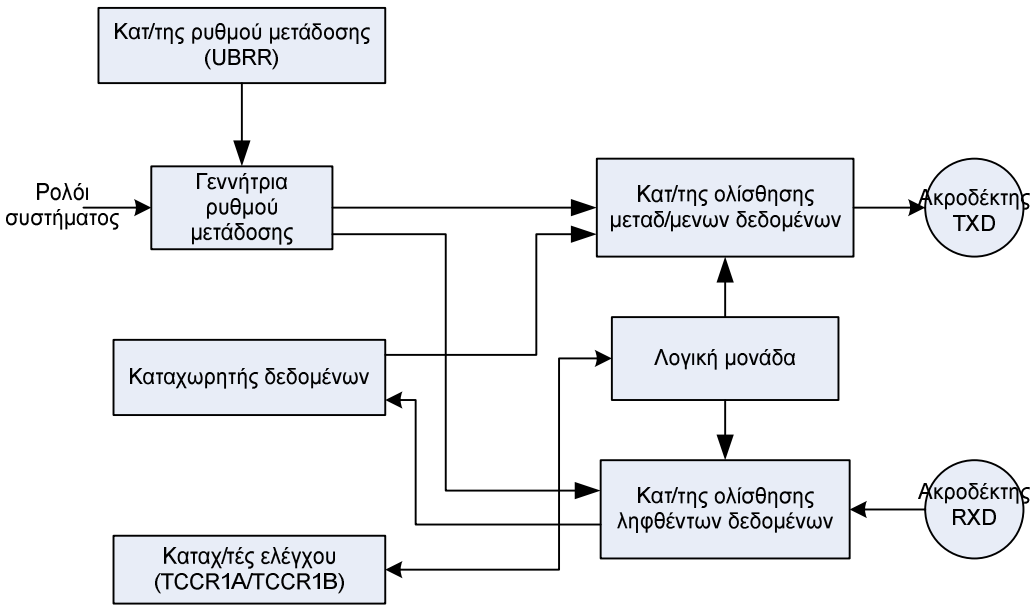
Πρόκειται για ένα πολύ ισχυρό και χρηστικό περιφερειακό, το οποίο χρησιμοποιείται προκειμένου να υλοποιηθεί με επιτυχία η αποστολή και λήψη δεδομένων από ένα PC, για επικοινωνία με το χρήστη, αποσφαλμάτωση κώδικα και άλλα.

Η μονάδα UART μπορεί να μεταδώσει τους εξής 30 συνδυασμούς πλαισίου:

- 1 bit εκκίνησης
- 5,6,7,8 ή 9 bits
- ένα ή δύο bit λήξης
- καμία, άρτια ή περιττή ισοτιμία.

Φιλτράρει τα ληφθέντα δεδομένα και ανιχνεύει σφάλματα πλαισίου και υπέρβασης. Διαθέτει τρία σήματα διακοπών και επιτρέπει μεγάλη ροή δεδομένων με απομονωτές υλοποιημένους από λογισμικό. Διαθέτει πομπό και δέκτη οι οποίοι μοιράζονται τη γεννήτρια του ρυθμού μετάδοσης και τους κατ/τες ελέγχου.

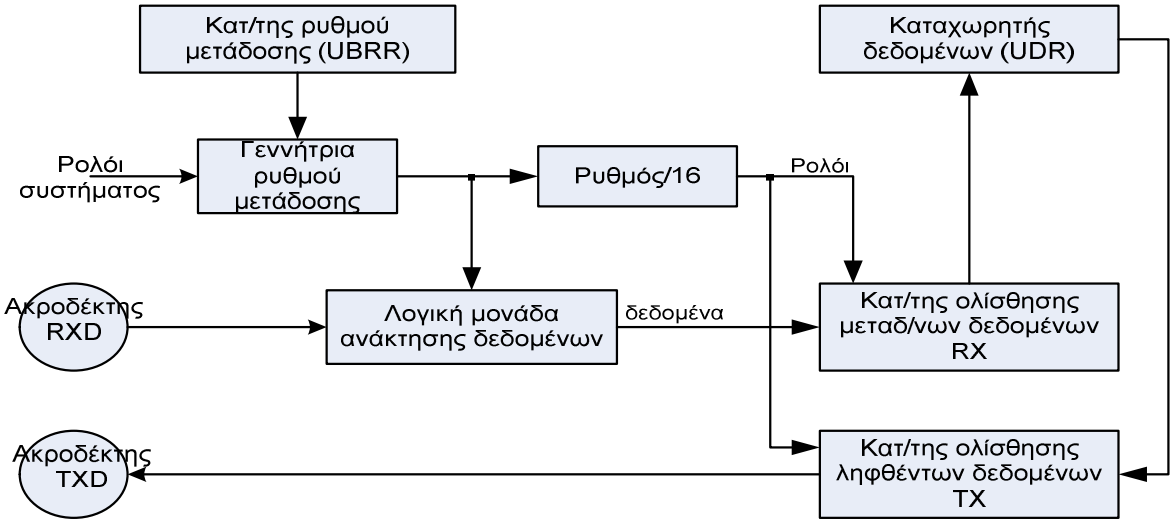
Η βασική δομή φαίνεται στο επόμενο σχήμα:



Σχήμα 3.2: Μονάδα UART

3.1.1 Γεννήτρια ρυθμού μετάδοσης

Η γεννήτρια ρυθμού μετάδοσης καθορίζει το σήμα ρολογιού που χρησιμοποιείται για μετάδοση και λήψη δεδομένων μέσω της UART. Το σήμα αυτό διαβαθμίζεται με μεγάλη ακρίβεια, με αποτέλεσμα μια σχεδόν αλάνθαστη μεταφορά δεδομένων.

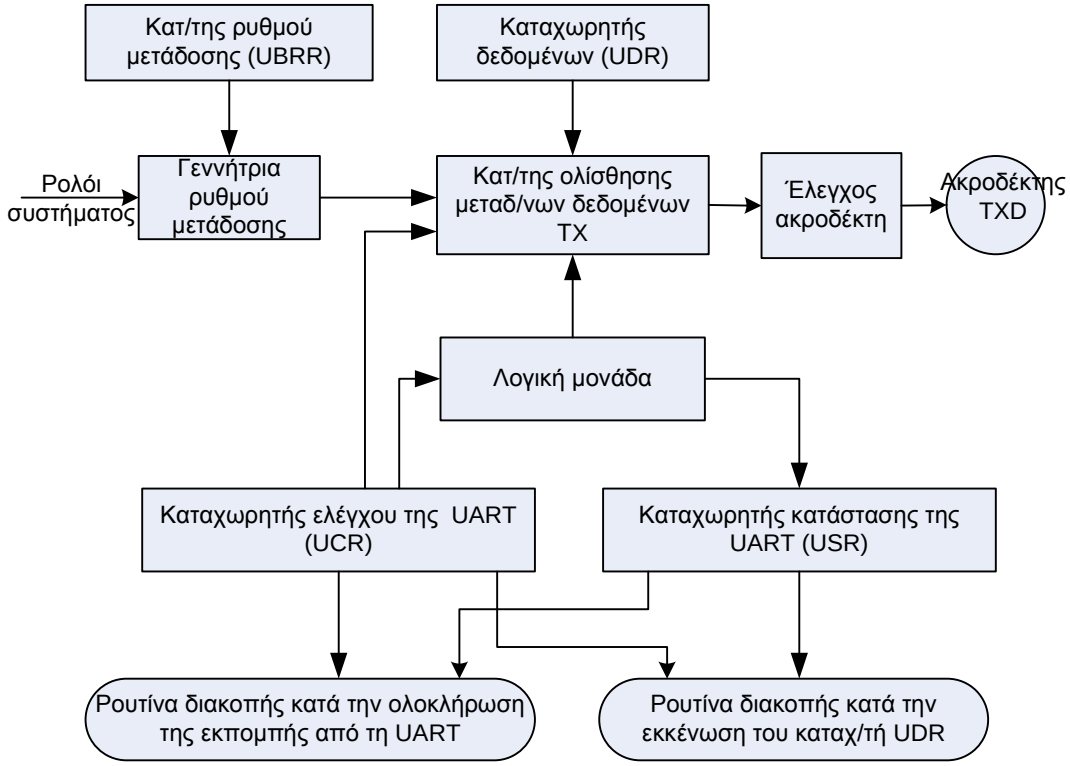


Σχήμα 3.3: Γεννήτρια ρυθμού μετάδοσης

Όπως φαίνεται στο σχήμα ο ρυθμός μετάδοσης διαιρείται με το 16 προτού τροφοδοτήσει τους καταχωρητές ολίσθησης εκπομπής και λήψης Rx/Tx. Διότι, το σήμα ρολογιού που παράγεται από τη γεννήτρια είναι 16πλάσιο του επιθυμητού ρυθμού μετάδοσης δεδομένων. Έτσι επιτυγχάνουμε δειγματοληψία με συχνότητα 16 φορές μεγαλύτερη από τον ρυθμό μετάδοσης. Επιπλέον το σήμα τροφοδοτεί το λογικό κύκλωμα ανάκτησης δεδομένων (Data Recovery Logic), το οποίο δειγματοληπτεί τα δεδομένα. Από την άλλη πλευρά ο κατ/της ολίσθησης εκπομπής δεν απαιτεί δειγματοληψία και συνδέεται απευθείας με το σήμα χρονισμού μετάδοσης.

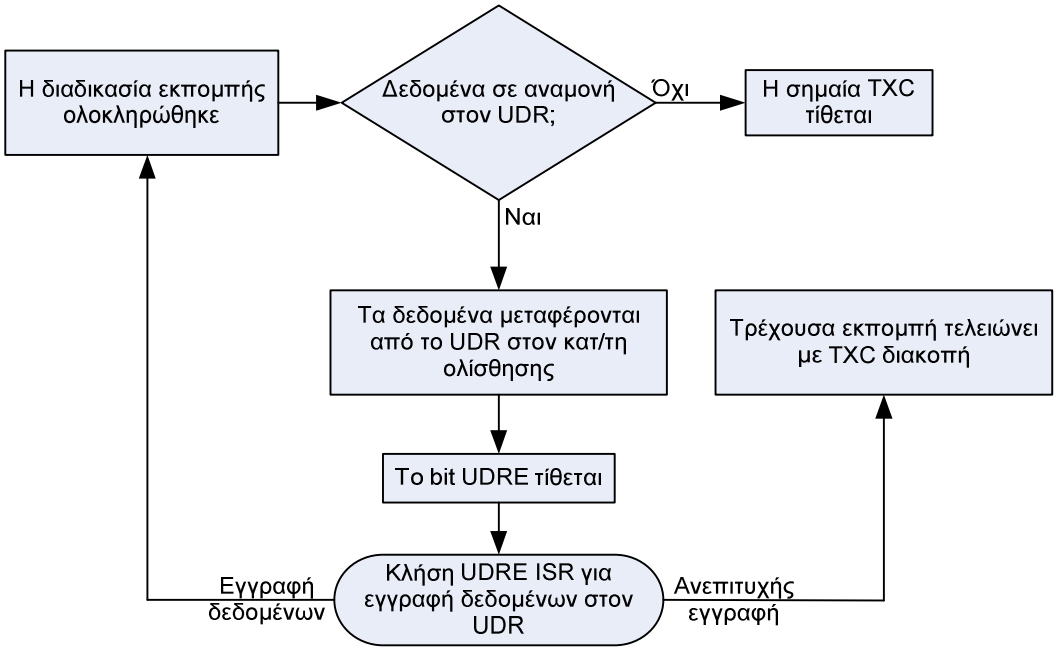
3.1.2 Πομπός UART

Ο πομπός UART στέλνει δεδομένα από τον μικροελεγκτή σε διάφορες συσκεύες (data logger, PC, κλπ) στον επιθυμητό ρυθμό μετάδοσης.



Σχήμα 3.4: Πομπός UART

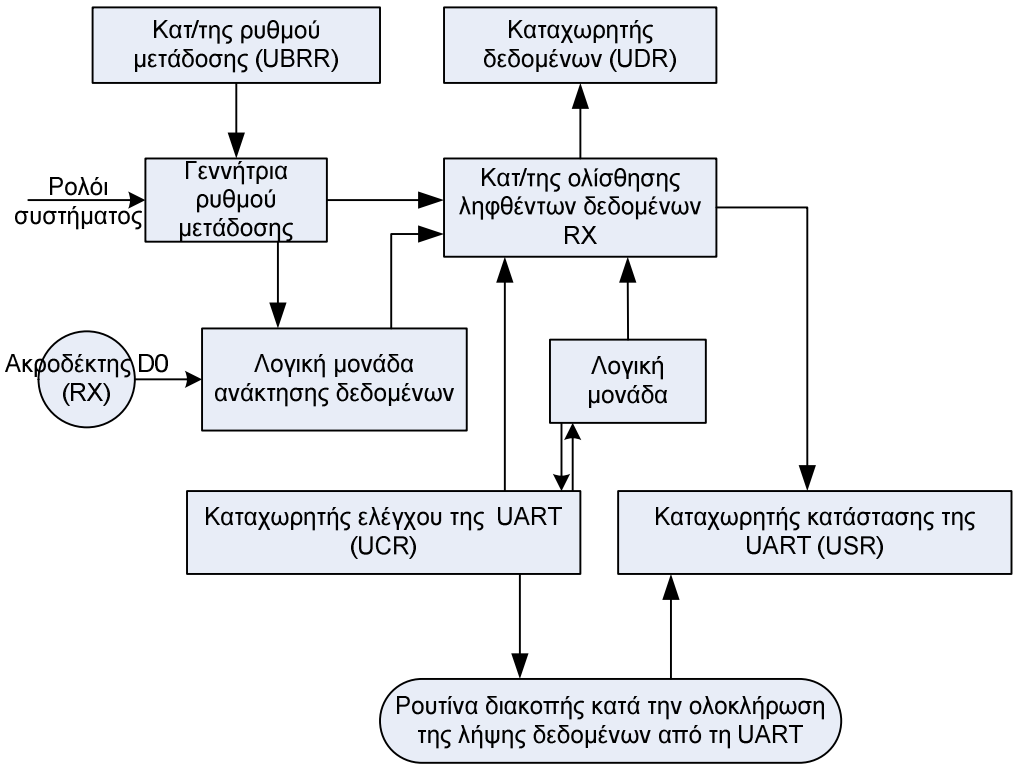
Η εκπομπή αρχικοποιείται εγγράφοντας δεδομένα στον κατ/τη UDR. Έπειτα, τα δεδομένα μεταφέρονται στον κατ/τη ολίσθησης εκπομπής TX εφόσον το προηγούμενο byte έχει ολισθήσει πλήρως. Όταν ένα byte μεταφέρεται στον κατ/τη ολίσθησης εκπομπής TX, η σημαία UDRE τίθεται. Η ρουτίνα εξυπηρέτησης διακοπής UDRE ISR μπορεί να γράψει το επόμενο byte στον κατ/τη UDR χωρίς να αλλοιώσει την παρούσα διαδικασία εκπομπής. Όταν ένα byte έχει ολισθήσει πλήρως και δεν έχουν γραφτεί δεδομένα στον κατ/τη UDR από τη ρουτίνα εξυπηρέτησης διακοπής UDRE ISR, τότε η σημαία TXC τίθεται. Το επόμενο διάγραμμα ροής δείχνει πως συνεργάζονται οι σημαίες διακοπών για την μετάδοση:



Σχήμα 3.5: Διάγραμμα ροής μετάδοσης

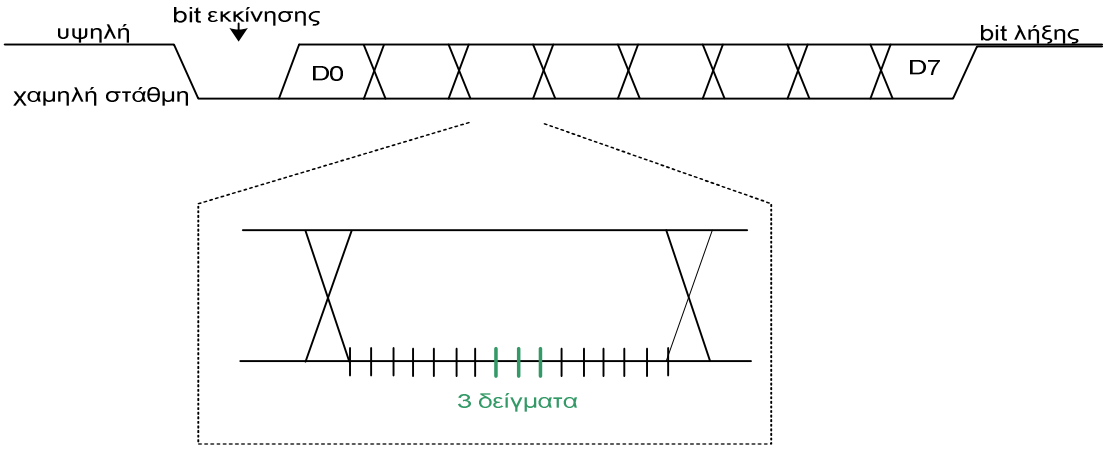
3.1.3 Ο δέκτης UART

Ο δέκτης UART έχει παρόμοια δομή με τον πομπό, αλλά επιπλέον διαθέτει τα απαραίτητα κυκλώματα για λήψη δεδομένων : το λογικό κύκλωμα ανάκτησης δεδομένων και μία διακοπή για την ολοκλήρωση της λήψης δεδομένου.



Σχήμα 3.6: Δέκτης UART

Το δεδομένο δειγματοληπτείται με τον τρόπο που περιγράψαμε στην ενότητα της γεννήτριας ρυθμού μετάδοσης και φαίνεται στο ακόλουθο σχήμα. Διακρίνουμε τις μικρές κάθετες γραμμές που αντιστοιχούν στους παλμούς ρολογιού που παράγει η γεννήτρια ρυθμού μετάδοσης. Οι τρεις κόκκινες αντιστοιχούν στο 8^{o} ,9^{o} και 10^{o} δείγμα. Συνεπώς, ο ρυθμός μετάδοσης αρχικά είναι 16πλάσιος ώστε να δειγματοληπτήσει τα δεδομένα και έπειτα διαιρείται με το 16 για να ολισθήσει τα δεδομένα.



Σχήμα 3.7: Δειγματοληψία δεδομένου

3.1.4 Καταχωρητής δεδομένων της ασύγχρονης σειριακής μονάδας UART(UDR)

Από το παραπάνω σχήμα φαίνεται ότι πομπός και δέκτης μοιράζονται τον κατ/τη δεδομένων UDR. Στην πραγματικότητα όμως πρόκειται για δυο διαφορετικούς καταχωρητές που έχουν κοινή φυσική διεύθυνση $0C. Ο ένας είναι ο καταχωρητής δεδομένων εκπομπής όπου μεταφέρονται τα δεδομένα που γράφουμε στη διεύθυνση $0C. Ο άλλος είναι ο καταχωρητής δεδομένων λήψης όπου μεταφέρονται τα δεδομένα που διαβάζουμε από τη διεύθυνση $0C.

3.1.5 Καταχωρητής κατάστασης της ασύγχρονης σειριακής μονάδας UART(USR)

Ο ρόλος του UART Status Register είναι η παρακολούθηση της θύρας αυτής. Διατηρεί τις σημαίες κατάστασης, όπως σημαίες διακοπών, σφάλματα υπέρβασης και πλαισίου. Η σημασία των bits ελέγχου φαίνεται στον πίνακα:

Πίνακας 3.1: Καταχωρητής κατάστασης της UART (USR)

Bit	Σύμβολο	Σημασία	Λειτουργία
7	RXC	Ολοκλήρωση διαδικασίας λήψης ενός byte δεδομένων. Μόλις διαβαστεί ο UDR η σημαία RXC μηδενίζεται.	1: Λήψη χαρακτήρα
6	TXC	Ολοκλήρωση διαδικασίας εκπομπής ενός byte δεδομένων. Η σημαία TXC μηδενίζεται μόλις εκτελεστεί η ρουτίνα εξυπηρέτησης διακοπής.	1:Άδειασμα κατ/τή ολίσθησης
5	UDRE	Ο κατ/τής UDR είναι έτοιμος να δεχθεί νέα δεδομένα.	1: Διαθέσιμος κατ/της δεδομένων
4	FE	Το εισερχόμενο bit σήμανσης λήξης μετάδοσης (stop bit) είναι ‘1’ αντί ‘0’.	1: Σφάλμα πλαισίου
3	OR	Ο κατ/της ολίσθησης εισάγει νέα δεδομένα ενώ τα	1: Σφάλμα

		προηγούμενα έγκυρα δεδομένα του UDR δεν έχουν διαβαστεί ακόμα.	υπέρβασης
2-0	-		

Bits	7	6	5	4	3	2	1	0
Κατάστασης	RXC	TXC	UDRE	FE	OR	-	-	-

3.1.6 Καταχωρητής ελέγχου της ασύγχρονης σειριακής μονάδας UART(UCR)

Ο κατ/της αυτός ελέγχει τον πομπό, το δέκτη και χειρίζεται τις διακοπές.

Πίνακας 3.2: Καταχωρητής ελέγχου της UART (UCR)

Bit	Σύμβολο	Σημασία	Λειτουργία
7	RXCIE	Ενεργοποίηση διακοπής μετά τη λήψη χαρακτήρα. Προϋπόθεση αποτελεί να έχουν τεθεί οι διακοπές και το bit RXC του USR.	1: διακοπή μετά τη λήψη χαρακτήρα
6	TXCIE	Ενεργοποίηση διακοπής μετά την εκπομπή χαρακτήρα. Προϋπόθεση αποτελεί να έχουν τεθεί οι διακοπές και το bit TXC του USR.	1: διακοπή μετά την εκπομπή χαρακτήρα
5	UDRIE	Ενεργοποίηση διακοπής κατά το άδειασμα του κατ/τή UDR. Προϋπόθεση αποτελεί να έχουν τεθεί οι διακοπές και το bit UDRE του USR.	1: διακοπή κατά το άδειασμα του UDR
4	RXEN	Ενεργοποίηση του τμήματος λήψης της μονάδας	1: Ενεργοποίηση

		UART.	του δέκτη
3	TXEN	Ενεργοποίηση του τμήματος εκπομπής της μονάδας UART.	1: Ενεργοποίηση του πομπού
2	CHR9	Το μήκος των χαρακτήρων εκπομπής - λήψης είναι 9-bit, χωρίς τα bit έναρξης και λήξης (start & stop bit).	1: χαρακτήρες των 9-bit
1	RXB8	Το RXB8 είναι το ένατο bit των ληφθέντων δεδομένων, εφόσον το bit CHR9 έχει τεθεί.	1: λήψη ένατου bit (bit8)
0	TXB8	Το TXB8 είναι το ένατο bit των σταλθέντων δεδομένων, εφόσον το bit CHR9 έχει τεθεί.	1: εκπομπή ένατου bit (bit8)

Bits Ελέγχου	7	6	5	4	3	2	1	0
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8

3.1.7 Καταχωρητής του ρυθμού μετάδοσης της μονάδας UART(UBRR)

Η μονάδα UART περιέχει τη γεννήτρια παραγωγής του ρυθμού μετάδοσης δεδομένων (baud rate generator). Ο κατ/της UBRR καθορίζει την τιμή του ρυθμού μετάδοσης που παράγει η γεννήτρια σύμφωνα με τον τύπο: Baud=Fck/[16*(UBRR+1)], όπου Fck η συχνότητα του ρολογιού και UBRR το περιεχόμενο του κατ/τη. Σε γρήγορους AVRs (megas) πρόκειται για ένα 16-bit κατ/τη που επιτρέπει χαμηλούς ρυθμούς μετάδοσης σε γρήγορους επεξεργαστές.

Παράδειγμα: Η συχνότητα ρολογιού του συστήματος είναι 8 MHz και επιθυμούμε ρυθμό στα 9600 bits/sec. Επιλύοντας ως προς την τιμή του κατ/τη UBRR (και αντικαθιστώντας fck=8 MHz και baud=9600):

$$UBRR = \frac{fck}{(16 * baud)} - 1$$

παίρνουμε την τιμή UBRR =51.08333333. Στρογγυλοποιούμε στην τιμή 51 που αντιστοιχεί σε 9615 baud, εισάγοντας ένα σφάλμα 0.16%. (Σημείωση: για το λόγο αυτό οι κρύσταλλοι έχουν περίεργες συχνότητες, όπως 7.3728 MHz. Με αυτή την συχνότητα παίρνουμε UBRR = 47 και ακριβώς 9600 baudrate).

Το παρακάτω παράδειγμα αρχικοποιεί, στέλνει και λαμβάνει δεδομένα των 8-bit από την USART με τη μέθοδο polling.

```
.include "m16def.inc"          ; δήλωση μικροελεγκτή atmega16
.def temp2 = r18               ; προσωρινός καταχωρητής
.def Delay = r19               ; μεταβλητή καθυστέρησης 1
.def Delay2 = r20              ; μεταβλητή καθυστέρησης 2
.def temp = r22


.equ rate = 25                 ; 4000000/(9600*16)-1=25 στα 4Mhz με σφάλμα=0,2%
   ldi temp, LOW(RAMEND)       ; αρχικοποίηση στοίβας
   out SPL, temp
   ldi temp, HIGH(RAMEND)
   out SPH, temp


   ldi r18,0xFF
   out DDRB,r18                ; ορισμός portB ως έξοδος
   rcall USART_Init
loop:
        rcall USART_Receive
```

```
        ;rcall DLY                          ; ρουτίνα καθυστέρησης
        ;inc r24                        ; επιστροφή επόμενου χαρακτήρα
        rcall USART_Transmit
        out PORTB,r24
rjmp loop


USART_Init:
   ldi temp, rate                ; επιλογή Baudrate 9600
   out UBRR,temp
   ldi temp, LOW(UBRR)
   out UBRRL, temp
   ldi temp, HIGH(UBRR)
   out UBRRH, temp
                                 ; Ενεργοποίηση πομπού και δέκτη
        ldi r16, (1<<RXEN)|(1<<TXEN)
        out UCSRB,r16
                                 ; Ορίζουμε τη μορφή πλαισίου 8-bit δεδομένων,
1 bit λήξης
        ldi r16, (1<<URSEL)|(0<<USBS)|(3<<UCSZ0)
        out UCSRC,r16
        ret


USART_Transmit:
```

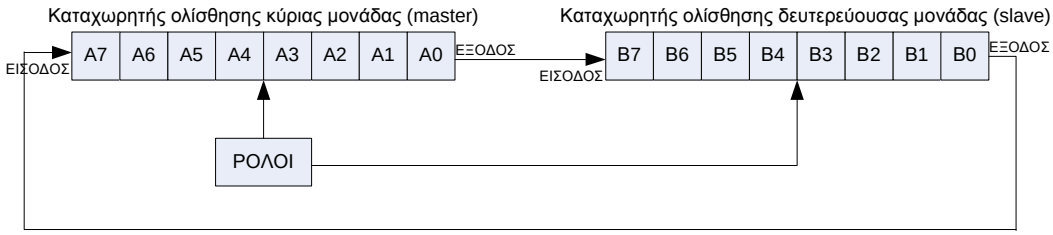
```
        sbis UCSRA,UDRE          ; Αναμονή ωσότου ο buffer μετάδοσης είναι
άδειος
        rjmp USART_Transmit
        out UDR,r24              ; Τοποθέτηση δεδομένων (r24) στον buffer,
        ret                      ; αποστολή δεδομένων


USART_Receive:
        sbis UCSRA, RXC          ; Αναμονή ωσότου ληφθούν τα δεδομένα
        rjmp USART_Receive
        in r24, UDR              ; Λήψη δεδομένων και επιστροφή
        ret


DLY: dec    Delay
        brne DLY
        dec    Delay2
        brne DLY
        ret
```

3.2 Μονάδα σύγχρονης σειριακής επικοινωνίας (SPI)

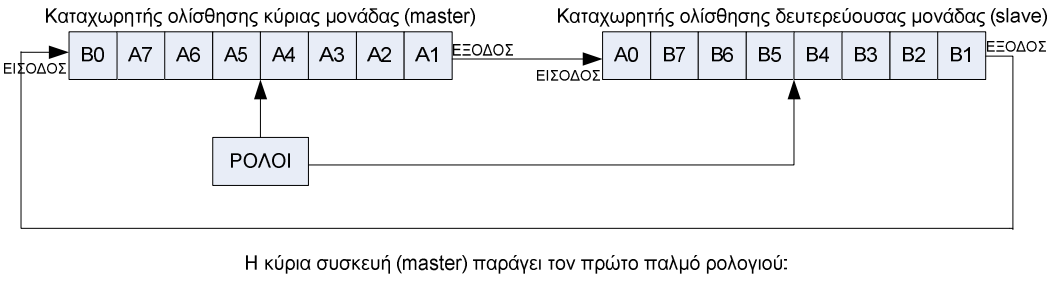
Η μονάδα σύγχρονης σειριακής επικοινωνίας SPI (Serial Peripheral Interface) είναι ένα περιφερειακό που χρησιμοποιείται για την επικοινωνία του μικροελεγκτή με διάφορες διατάξεις, όπως εξωτερικές μνήμες EEPROM, μετατροπείς DAC και ADC, άλλους μικροελεγκτές. Η μονάδα SPI δεν είναι διαθέσιμη σε όλα τα μοντέλα Παρόλα αυτά μπορούμε να χρησιμοποιήσουμε εναλλακτικά τις απλές θύρες εισόδου - εξόδου

και να μιμηθούμε τη λειτουργία μιας πραγματικής θύρας SPI με κατάλληλο λογισμικό. Συγκεκριμένα, έχουμε μια κύρια συσκευή (master) που αρχικοποιεί και ελέγχει την επικοινωνία, και μία ή περισσότερες δευτερεύουσες συσκευές (slaves) που λαμβάνουν και εκπέμπουν στην κύρια.

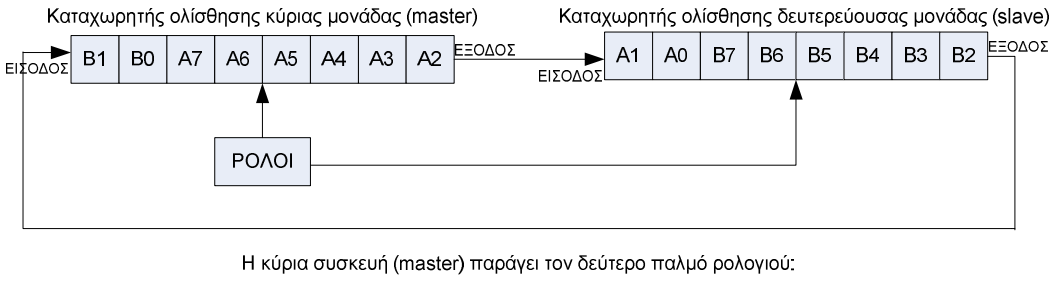
Ο πυρήνας της μονάδας SPI είναι ένας 8-μπιτος καταχωρητής ολίσθησης ευρισκόμενος σε κύρια και σε δευτερεύουσα συσκευή, καθώς επίσης και το σήμα χρονισμού παραγόμενο από την κύρια διάταξη. Έστω ότι η κύρια διάταξη (master) επιθυμεί να στείλει ένα byte δεδομένων στην δευτερεύουσα (slave) και συγχρόνως να λάβει δεδομένα από τη δευτερεύουσα. Προτού αρχίσει η επικοινωνία ο master και ο slave τοποθετούν τα δεδομένα τους στους κατ/τές ολίσθησης τους (σχήμα α). Τότε ο master παράγει 8 παλμούς ρολογιού και τα περιεχόμενα του κατ/τη ολίσθησης του μεταφέρονται στον κατ/τη ολίσθησης του slave και αντιστρόφως (σχήμα β ως ε). Η εκπομπή και η λήψη γίνονται συγχρόνως, οπότε έχουμε μια πλήρως αμφίδρομη μεταφορά δεδομένων. Η πρώτη εικόνα απεικονίζει τη κατάσταση των κατ/των πριν την μεταφορά, ενώ οι επόμενες απεικονίζουν τη διαδικασία μεταφοράς:



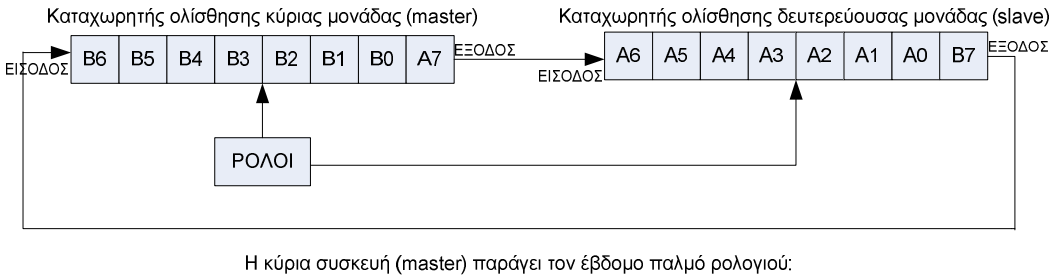
Σχήμα 3.8.1 (α): Μεταφορά δεδομένου μεταξύ master και slave διάταξης



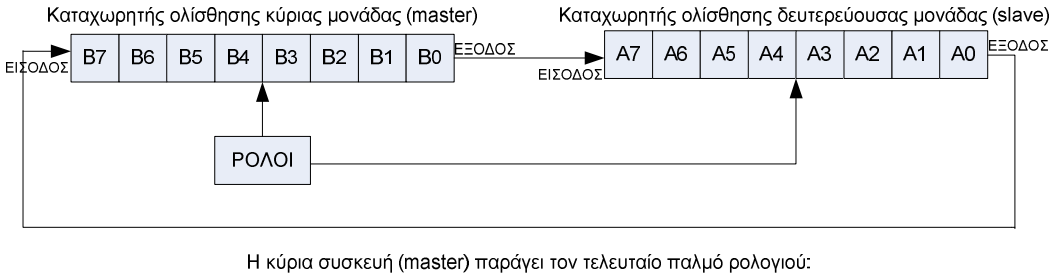
Σχήμα 3.8.1 (β): Μεταφορά δεδομένου μεταξύ master και slave διάταξης



Σχήμα 3.8.1 (γ): Μεταφορά δεδομένου μεταξύ master και slave διάταξης



Σχήμα 3.8.1 (δ): Μεταφορά δεδομένου μεταξύ master και slave διάταξης



Σχήμα 3.8.1 (ε): Μεταφορά δεδομένου μεταξύ master και slave διάταξης

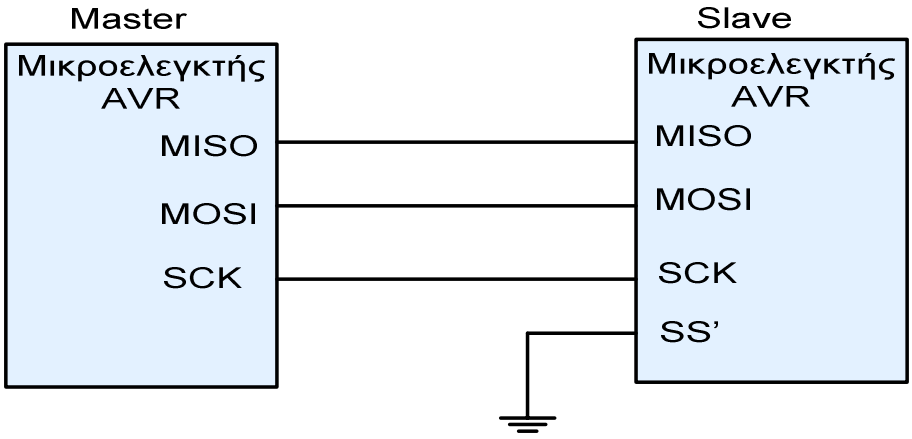
3.2.1 Περιγραφή διαύλου

Η επικοινωνία μέσω της μονάδας SPI προϋποθέτει τη συμφωνία κύριας και δευτερεύουσας διάταξης στις ρυθμίσεις σήματος χρονισμού. Τέσσερα σήματα (ακροδέκτες) χρειάζονται για την επικοινωνία: MISO, MOSI, SCK και SS'. Η λειτουργία κάθε σήματος είναι η εξής:

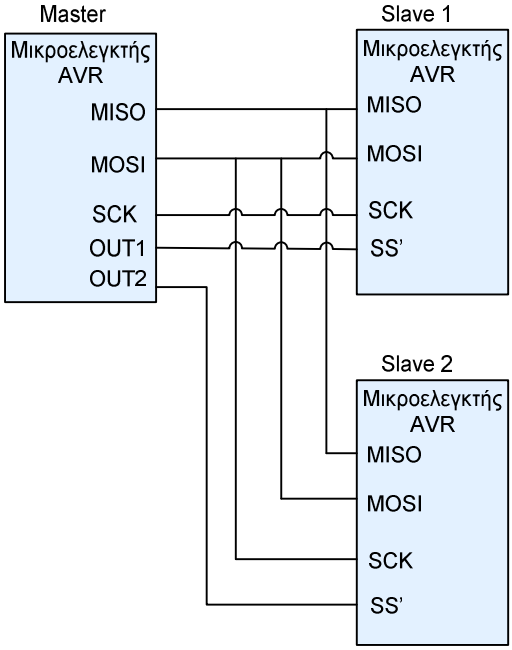
- MISO (Master In Slave Out): η είσοδος του κατ/τη ολίσθησης της κύριας διάταξης και η έξοδος του κατ/τη ολίσθησης της δευτερεύουσας διάταξης.

- MOSI (Master Out Slave In):): η έξοδος του κατ/τη ολίσθησης της κύριας διάταξης και η είσοδος του κατ/τη ολίσθησης της δευτερεύουσας διάταξης.
- SCK (Serial Clock): Στην κύρια διάταξη, το σήμα αυτό είναι η έξοδος της γεννήτριας χρονισμού. Στην δευτερεύουσα πρόκειται για την είσοδο του σήματος χρονισμού.
- SS' (Slave Select): Σε μια εγκατάσταση SPI μπορούν να συνδεθούν πολλές δευτερεύουσες διατάξεις. Το σήμα SS’ επιλέγει με ποια θα επικοινωνήσουμε. Όταν είναι σε υψηλή κατάσταση, όλοι οι ακροδέκτες των δευτερευουσών διατάξεων είναι είσοδοι και προφανώς δεν θα λάβουν δεδομένα μέσω της μονάδας SPI. Όταν είναι σε χαμηλή κατάσταση η μονάδα SPI ενεργοποιείται. Το λογισμικό της κύριας ελέγχει τη γραμμή SS’ κάθε δευτερεύουσας. Στη κύρια διάταξη, η συμπεριφορά του ακροδέκτη SS’ εξαρτάται από τη κατεύθυνση δεδομένων του ακροδέκτη. Αν ο SS’ ρυθμιστεί ως έξοδος, τότε δεν επιδρά στη μονάδα SPI. Αν ο SS’ ρυθμιστεί ως είσοδος, τότε πρέπει να διατηρηθεί σε υψηλή κατάσταση για να επιτύχουμε κύρια(master) λειτουργία. Σε χαμηλή κατάσταση το σύστημα ερμηνεύει το γεγονός ότι κάποια άλλη κύρια διάταξη χρησιμοποιεί την SPI για αποστολή δεδομένων σε δευτερεύουσα. Αυτή η κατάσταση (δηλαδή να υπάρχουν δυο SPI masters) είναι ασυνήθιστη και δεν θα μας απασχολήσει. Αυτό που μας ενδιαφέρει είναι να ρυθμίζουμε τον ακροδέκτη SS’ της κύριας διάταξης ως έξοδο.

Τα παρακάτω σχήματα δείχνουν μια τυπική εγκατάσταση μονάδας SPI:



Σχήμα 3.9: Εγκατάσταση μονάδας SPI με μια δευτερεύουσα διάταξη



Σχήμα 3.10: Εγκατάσταση μονάδας SPI με δύο δευτερεύουσες διατάξεις

Η κατεύθυνση των δεδομένων των ακροδεκτών της SPI εξαρτάται από το εκάστοτε ακροδέκτη και αν πρόκειται για κύρια ή δευτερεύουσα συσκευή. Γενικά υπάρχουν δυο δυνατότητες: ο ακροδέκτης να καθορίζεται ως είσοδος ανεξαρτήτως του καταχωρητή κατεύθυνσης δεδομένων της θύρας ή να καθορίζεται από τον χρήστη με βάση τη λειτουργία του. Ο ακόλουθος πίνακας συνοψίζει τα παραπάνω.

Πίνακας 3.3: κατεύθυνση των δεδομένων των ακροδεκτών της SPI

Ακροδέκτης	**κατεύθυνση Master SPI**	**κατεύθυνση Slave SPI**
MOSI	καθορίζεται από χρήστη	είσοδος
MISO	είσοδος	καθορίζεται από χρήστη
SCK	καθορίζεται από χρήστη	είσοδος
SS'	καθορίζεται από χρήστη	είσοδος

3.2.2 Καταχωρητής δεδομένων της σειριακής θύρας SPI(SPDR)

Ο καταχωρητής δεδομένων της θύρας SPI (Serial Peripheral Interface) είναι εγγραφής/ ανάγνωσης για την μεταφορά δεδομένων μεταξύ του καταχωρητή ολίσθησης της σειριακής μονάδας SPI και των καταχωρητών εργασίας. Δηλαδή πρόκειται για τον καταχωρητή στον οποίο γράφουμε τα δεδομένα προς μετάδοση και διαβάζουμε τα ληφθέντα δεδομένα από τον αντίστοιχο buffer. Η μονάδα SPI χρησιμοποιείται για την επικοινωνία του μικροελεγκτή με διάφορες περιφερειακές διατάξεις.

3.2.3 Καταχωρητής κατάστασης της σειριακής θύρας SPI(SPSR)

Η σημασία των bits ελέγχου είναι η εξής:

- Bit 7 (SPIF - SPI Interrupt Flag): Σημαία διακοπής της σειριακής μετάδοσης της θύρας SPI. Η σημαία αυτή τίθεται μόλις ολοκληρωθεί η σειριακή μετάδοση και εφόσον το bit SPIE του SPCR και οι διακοπές είναι ενεργοποιημένες. Μετά την εξυπηρέτηση της αντίστοιχης ρουτίνας, το bit μηδενίζεται. Επίσης, το bit μηδενίζεται μετά την ανάγνωση του καταχωρητή κατάστασης και πρόσβαση στον καταχωρητή δεδομένων της θύρας SPI.
- Bit 6 (WCOL - Write Collision Flag): Ενημέρωση σύγκρουσης δεδομένων. Αν κατά τη διάρκεια μετάδοσης δεδομένων μέσω της θύρας SPI εγγραφεί νέο δεδομένο, τότε προκύπτει σύγκρουση και το bit αυτό τίθεται. Το bit μηδενίζεται μετά την ανάγνωση του καταχωρητή κατάστασης και πρόσβαση στον καταχωρητή δεδομένων της θύρας SPI.

Bits	7	6	5	4	3	2	1	0
Ελέγχου	SPIF	WCOL						

3.2.4 Καταχωρητής ελέγχου της σειριακής θύρας SPI(SPCR)

Η σημασία των bits ελέγχου φαίνεται στον πίνακα:

Πίνακας 3.4: Καταχωρητής ελέγχου της σειριακής θύρας SPI (SPCR)

Bit	Σύμβολο	Σημασία	Λειτουργία
7	SPIE	Ενεργοποίηση διακοπών	0:απενεργοποίηση διακοπών 1: ενεργοποίηση διακοπών
6	SPE	Ενεργοποίηση σειριακής θύρας	0: απενεργοποίηση θύρας SPI 1: ενεργοποίηση θύρας SPI
5	DORD	Σειρά μετάδοσης δεδομένων	0: MSB πρώτο 1: LSB πρώτο
4	MSTR	Επιλογή master/ slave λειτουργίας	0: Slave 1: Master
3	CPOL	Πόλωση σήματος ρολογιού	0:Θετική στάθμη σήματος χρονισμού 1:Αρνητική στάθμη
2	CPHA	Φάση σήματος ρολογιού	0:δειγματοληψία στην αρχή της φάσης 1:δειγματοληψία στο τέλος της φάσης
1 0	SPR1 SPR0	Επιλογή συχνότητας ρολογιού	00: Clock / 4 01: Clock / 16 10: Clock / 64 11: Clock / 128

Bits Ελέγχου	7	6	5	4	3	2	1	0
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

Υπάρχουν οι εξής τέσσερις τρόποι να ρυθμίσεις τη γεννήτρια χρονισμού:

Πίνακας 3.5: ρυθμίσεις γεννήτριας χρονισμού θύρας SPI

Τρόπος SPI	**CPOL**	**CPHA**	**δείγμα**
0	0	0	ακμή ανόδου
1	0	1	ακμή καθόδου
2	1	0	ακμή καθόδου
3	1	1	ακμή ανόδου

Τέλος, παραθέτουμε ένα τμήμα κώδικα για την μεταφορά δεδομένων μεταξύ κύριας και δευτερεύουσας διάταξης. Η επικοινωνία γίνεται με τον τρόπο SPI 3, και αποστολή πρώτα του MSB ψηφίου. Η συχνότητα ρολογιού είναι fosc/16.Ο Master στέλνει το δεδομένο 0xAA, και ο Slave το 0x55.

Κώδικας Master:

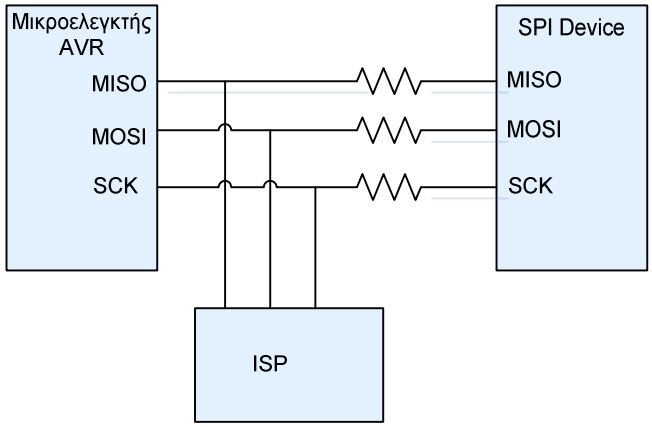
```
SPI_Init:
 sbi DDRB,DDB5          ; Θέτουμε MOSI ως έξοδο.
 sbi DDRB,DDB7          ; Θέτουμε SCK ως έξοδο.
 sbi DDRB,DDB4          ; Θέτουμε SS' ως έξοδο.
 ldi r16,01011101b      ; Θέτουμε SPI ως Master, με απενεργοποιημένες διακοπές
 out SPCR,r16           ; MSB πρώτο, SPI mode 3 και συχνότητα ρολογιού
fosc/16.


SPI_Send:
 ldi                                                    r16,0xAA
 out SPDR,r16           ; αρχικοποίηση μεταφοράς δεδομένων.
Wait:
 sbis SPSR,SPIF         ; αναμονή ωσότου ολοκληρωθεί η μετάδοση.
 rjmp                                                   Wait
 in SPDR,r16            ; τα ληφθέντα δεδομένα σώζονται στον r16.
```

Κώδικας Slave:

```
SPI_Init:
 sbi DDRB,DDB6          ; Θέτουμε MISO ως έξοδο
 ldi r16,01001100b      ; Θέτουμε SPI ως Slave, με απενεργοποιημένες διακοπές
 out SPCR,r16           ; MSB πρώτο, SPI mode 3.
 ldi                                              r16,0x55
 out SPDR,r16           ; αποστολή 0x55 με αίτηση Master.


SPI_Receive:
 sbis                                             SPSR,SPIF
 rjmp SPI_Receive       ; αναμονή ωσότου ολοκληρωθεί η λήψη.
  in r16,SPDR           ; τα ληφθέντα δεδομένα σώζονται στον r16.
```

3.2.5 SPI και In Circuit Programming (ISP)

Η μονάδα SPI χρησιμοποιείται επιπλέον για τον προγραμματισμό του μικροελεγκτή εντός κυκλώματος. Στην περίπτωση αυτή απαιτείται μια σειρά αντιστάσεων στις τρεις γραμμές όπως φαίνεται στο σχήμα:



Σχήμα 3.11: Προγραμματισμός εντός κυκλώματος

3.3 Χρονιστής

Υπάρχουν διάφοροι τύποι χρονιστών διαθέσιμοι στους μικροελεγκτές (ενώ μερικά μοντέλα δεν διαθέτουν καθόλου χρονιστή). Οι χρονιστές βασικά μετρούν παλμούς ρολογιού. Οι AVR διαθέτουν έναν απλό χρονιστή των 8-bits (timer/counter0) και έναν πολύπλοκο χρονιστή των 16-bits (timer/counter1) με τέσσερις τρόπους λειτουργίας. Ο χρονιστής δύναται να λειτουργεί ως χρονιστής ή ως μετρητής. Ως μετρητής δέχεται στην είσοδο του παλμούς ενός εξωτερικού σήματος, ενώ το ρολόι του χρονιστή μπορεί να είναι ίσο με το κεντρικό ρολόι τους συστήματος (από τον κρύσταλλο) ή να επιβραδυνθεί με τον διαιρέτη συχνότητας (prescaler). Ο prescaler υπάρχει στον timer0 , αλλά και στον timer1. Με τον prescaler επιτυγχάνουμε μεγαλύτερες τιμές μέτρησης, αλλά μικρότερη ακρίβεια. Ο prescaler παίρνει τις τιμές 8, 64, 256 ή 1024 σε σύγκριση με το ρολόι του συστήματος. Ένας AVR στα 8 MHz και ένας χρονιστής prescaler μπορεί να μετρήσει (όταν έχει 16-bit timer) : (0xFFFF + 1) * 1024 clock cycles = 67108864 clock cycles δηλαδή 8.388608 seconds.

3.3.1 Καταχωρητής μάσκας διακοπών χρονιστή (Timer Interrupt Mask Register -TIMSK)

Ο ρόλος του κατ/τη αυτού είναι να καθορίζει ποιες διακοπές είναι έγκυρες θέτοντας τα αντίστοιχα bits του.

Bit	7	6	5	4	3	2	1	0
Bits μάσκας	TOIE1	OCIE1A			TICIE1		TOIE0	

Η σημασία των bits του καταχωρητή είναι η εξής:

- Bit 7 (TOIE1- Timer Overflow Interrupt Enable/Timer 1): αν το bit αυτό έχει τεθεί και οι καθολικές διακοπές είναι ενεργοποιημένες, τότε εκτελείται άλμα στο διάνυσμα διακοπής υπερχείλισης του 16-bit χρονιστή timer1.

- Bit 6 (OCIE1A - Output Compare Interrupt Enable 1A): αν το bit αυτό έχει τεθεί και οι καθολικές διακοπές είναι ενεργοποιημένες, τότε εκτελείται άλμα στο διάνυσμα διακοπής του συγκριτή Α
- Bit 3 (TICIE1 - Timer 1 Input Capture Interrupt Enable): αν το bit αυτό έχει τεθεί και οι καθολικές διακοπές είναι ενεργοποιημένες, τότε εκτελείται άλμα στο διάνυσμα διακοπής εισόδου σύλληψης
- Bit 1 (TOIE0 - Timer Overflow Interrupt Enable/Timer 0): ομοίως με TOIE1, αλλά για τον 8-bit χρονιστή.

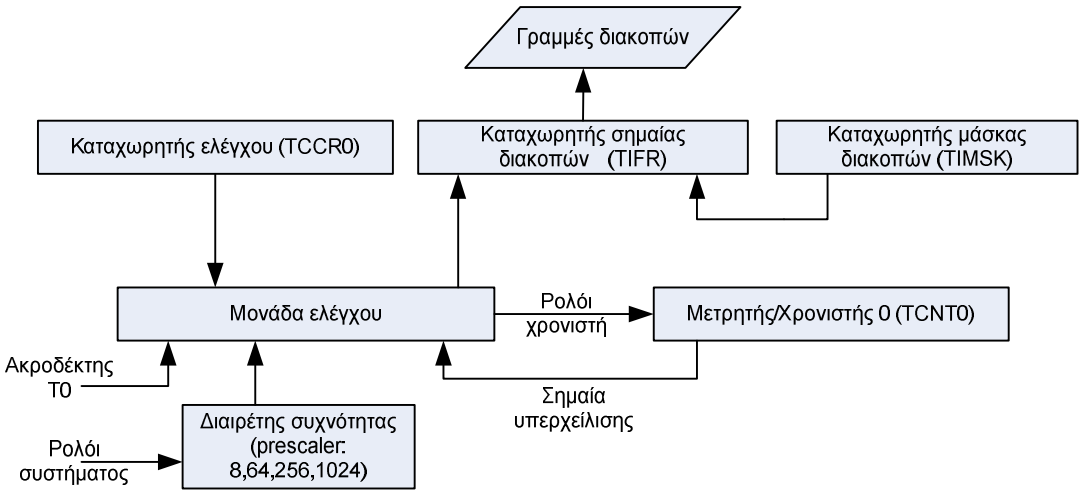
3.3.2 Καταχωρητής σημαίας διακοπών χρονιστή (Timer Interrupt Flag Register -TIFR)

Ο καταχωρητής αυτός μας ενημερώνει ποιες διακοπές εκκρεμούν. Διατηρεί τις σημαίες διακοπής του χρονιστή που αντιστοιχούν στα bits του TIMSK. Αν μια διακοπή δεν έχει ενεργοποιηθεί ελέγχουμε τον TIFR για να καθορίσουμε εάν συνέβη και να μηδενίσουμε τις σημαίες.

Bit	7	6	5	4	3	2	1	0
Bits σημαίας	TOV1	OCF1A			ICF1		TOV0	

3.3.3 Ο 8-bit Χρονιστής (Timer0):

Η είσοδος χρονισμού του χρονιστή των 8-bits συνδέεται είτε με το κεντρικό ρολόι, είτε με ένα σήμα υποπολλαπλάσιας συχνότητας του κεντρικού ρολογιού, είτε με κάποιο εξωτερικό σήμα σε κάποιον ακροδέκτη εισόδου-εξόδου (T0 Pin).Ακόμη, έχουμε τη δυνατότητα να σταματήσουμε τη λειτουργία του χρονιστή, τοποθετώντας τα σωστά bits στον κατ/τη ελέγχου TCCR0. Ο χρονιστής αποτελεί βασικά έναν μετρητή προς τα πάνω. Παρακάτω φαίνεται ένα διάγραμμα με το υλικό του χρονιστή:



Σχήμα 3.12: 8-bit χρονιστής και hardware

Η λειτουργία του είναι απλή: το ρολόι χρονισμού μετρά προς τα πάνω μέσω του κατ/τη Timer/Counter Register (TCNT0). Όταν αυτός υπερχειλίσει (από την τιμή 0xFF -> στην 0x00) τίθεται η σημαία υπερχείλισης του κατ/τη κατάστασης (Overflow Flag) και η σημαία διακοπής λόγω υπερχείλισης του χρονιστή. Αν το bit TOIE0 στον κατ/τη μάσκας διακοπών του χρονιστή TIMSK (Timer Interrupt Mask Register) είναι ‘1’ και οι καθολικές διακοπές είναι ενεργοποιημένες, τότε θα έχουμε άλμα στο αντίστοιχο διάνυσμα διακοπής.

3.3.4 Καταχωρητής ελέγχου του TIMER/COUNTER0 (TCCR0)

Αυτός ο καταχωρητής ελέγχει τη λειτουργία του εσωτερικού μετρητή/χρονιστή, ο οποίος μετρά μέχρι την τιμή που του έχει φορτωθεί. Κάθε βήμα μέτρησης αντιστοιχεί σε ένα παλμό του σήματος διέγερσης. Οι πηγές χρονισμού φαίνονται στον πίνακα:

Πίνακας: Πηγές χρονισμού του TIMER/COUNTER0

CS02	**CS01**	**CS00**	**Σχόλιο**
0	0	0	Διακοπή λειτουργίας του χρονιστή
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64

1	0	0	CK/256
1	0	1	CK/1024
1	1	0	Αρνητικό μέτωπο παλμού στον ακροδέκτη T0
1	1	1	Θετικό μέτωπο παλμού στον ακροδέκτη T0

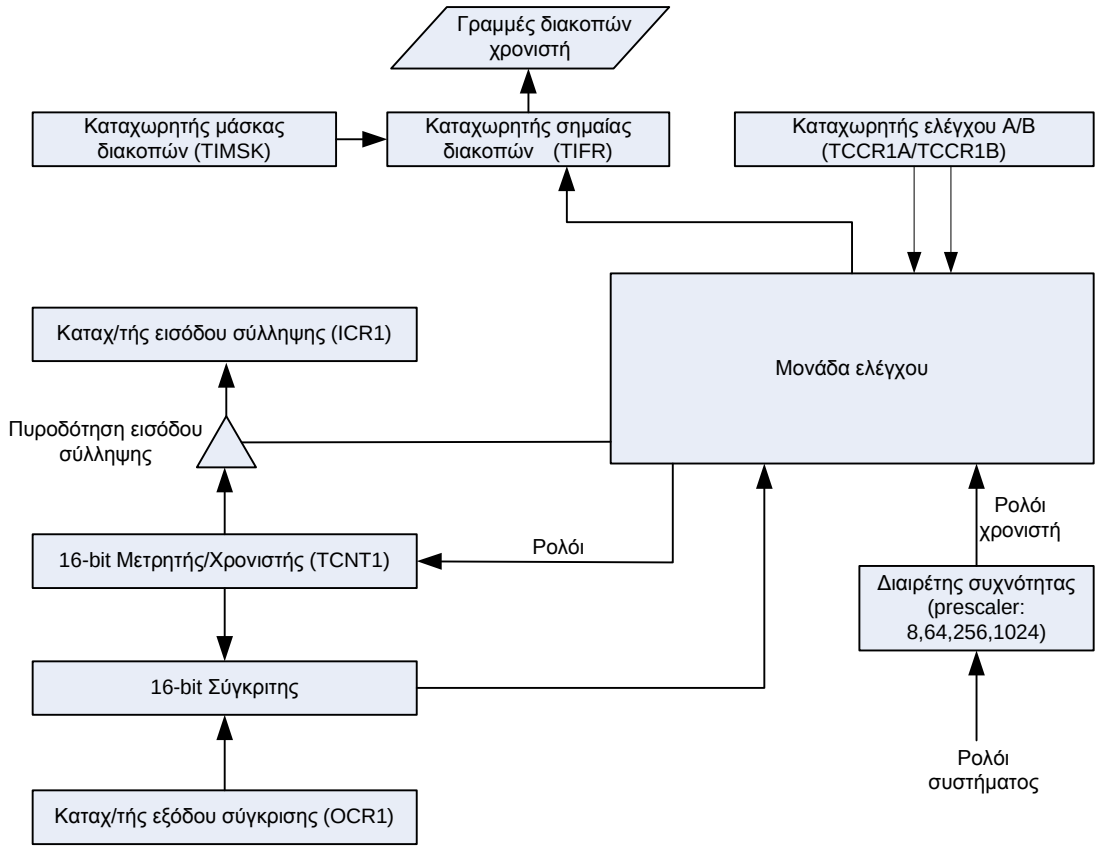
Bit	7	6	5	4	3	2	1	0
Bits ελέγχου						CS02	CS01	CS00

3.3.5 Καταχωρητής του TIMER/COUNTER0 (TCNT0)

Ο καταχωρητής αυτός μετρά τους παλμούς της εισόδου χρονισμού μέχρι να φτάσει την τιμή που του έχουμε φορτώσει. Μόλις φτάσει αυτή τη τιμή υπερχείλισης, τότε ο μετρητής μηδενίζεται και η μέτρηση ξεκινά από την αρχή.

3.3.6 Ο 16-bit Χρονιστής:

Ο 16-bit Χρονιστής είναι πιο σύνθετος:



Σχήμα 3.13: 16-bit χρονιστής και hardware

3.3.7 Καταχωρητής του TIMER/COUNTER1 (TCNT1H- TCNT1L)

Πρόκειται για έναν καταχωρητή των 16-bits που περιέχει την τιμή μέτρησης.

3.3.8 Καταχωρητής ελέγχου A του TIMER/COUNTER1 (TCCR1A)

Αυτός ο καταχωρητής ελέγχει τη λειτουργία του μετρητή/χρονιστή1.

Bits	7	6	5	4	3	2	1	0
Ελέγχου	COM1A 1	COM1A 0	COM1B 1	COM1B 0			PWM1 1	PWM1 0

Η σημασία των bits ελέγχου του καταχωρητή είναι η εξής:

§ Bits 7,6 (COM1A1, COM1A0): Είναι τα ψηφία για τον έλεγχο της λειτουργίας εξόδου σύγκρισης (Compare Output Mode). Ο ακροδέκτης εξόδου OC1(Output Compare) επηρεάζεται μετά από τη σύγκριση που έχει εκτελέσει ο μετρητής / χρονιστής1. Προϋπόθεση αποτελεί το ψηφίο ελέγχου κατεύθυνσης να έχει τεθεί 1, ώστε ο ακροδέκτης να είναι έξοδος.

Πίνακας : Λειτουργία εξόδου σύγκρισης

COM1A1	**COM1A0**	**Λειτουργία-τιμή της εξόδου OC1**
0	0	Αποσύνδεση από μετρητή/ χρονιστή1
0	1	Εναλλαγή επιπέδου
1	0	0
1	1	1

§ Bits 5,4 (COM1B1, COM1B0): Αντίστοιχα με bits 7,6.

§ Bits 1,0 (PWM11, PWM10): Είναι τα ψηφία επιλογής του τρόπου λειτουργίας της μονάδας του διαμορφωτή εύρους παλμών (PWM). Η σημασία φαίνεται στον πίνακα:

Πίνακας : Τρόπος λειτουργίας του διαμορφωτή εύρους παλμών (PWM)

PWM 11	**PWM 10**	**Λειτουργία**	**Μέγιστη τιμή**	**PWM συχνότητα ως προς ρολόι χρονιστή**
0	0	Απενεργοποίηση του PWM		
0	1	8-bit PWM	$00FF	fclock/510
1	0	9-bit PWM	$01FF	fclock/1022
1	1	10-bit	$03FF	fclock/2046

		PWM		

3.3.9 Καταχωρητής ελέγχου Β του TIMER/COUNTER1 (TCCR1B)

Ο μετρητής / χρονιστής1 έχει και δεύτερο καταχωρητή ελέγχου και η σημασία των bits ελέγχου είναι η εξής:

- Bit 7 (ICNC1): Ο μηχανισμός σύλληψης και απόρριψης θορύβου ενεργοποιείται.
- Bit 6 (ICES1): Καθορίζεται αν η είσοδος σύλληψης θα γίνει σε αρνητικό ή θετικό μέτωπο παλμού. Δηλαδή, όταν αυτό το bit είναι '1', τα περιεχόμενα του μετρητή/ χρονιστή1 μεταφέρονται στον καταχωρητή της εισόδου σύλληψης κατά το θετικό μέτωπο του σήματος στην είσοδο σύλληψης ICP.
- Bit 3 (CTC1): Όταν το ψηφίο αυτό είναι '1' ο μετρητής/ χρονιστής1 μηδενίζεται εάν η σύγκριση είναι επιτυχής. Ενώ όταν το ψηφίο αυτό είναι '0' η μέτρηση συνεχίζεται αγνοώντας τις επιτυχείς συγκρίσεις.
- Bit 2,1,0 (CS12,CS11,CS10): Επιλογή του σήματος χρονισμού του μετρητή/ χρονιστή1.

Bits	7	6	5	4	3	2	1	0
Ελέγχου	ICNC1	ICES1			CTC1	CS12	CS11	CS10

3.3.10 Καταχωρητής εξόδου σύγκρισης του TIMER/COUNTER1 (OCR1AH-OCR1AL)

Πρόκειται για έναν καταχωρητή των 16-bits με δυνατότητα εγγραφής και ανάγνωσης. Το περιεχόμενο του συγκρίνεται με την τιμή μέτρησης του μετρητή/ χρονιστή1 και σε περίπτωση επιτυχούς ελέγχου εκτελούνται οι λειτουργίες που καθορίζουν οι καταχωρητές ελέγχου του μετρητή/ χρονιστή1.

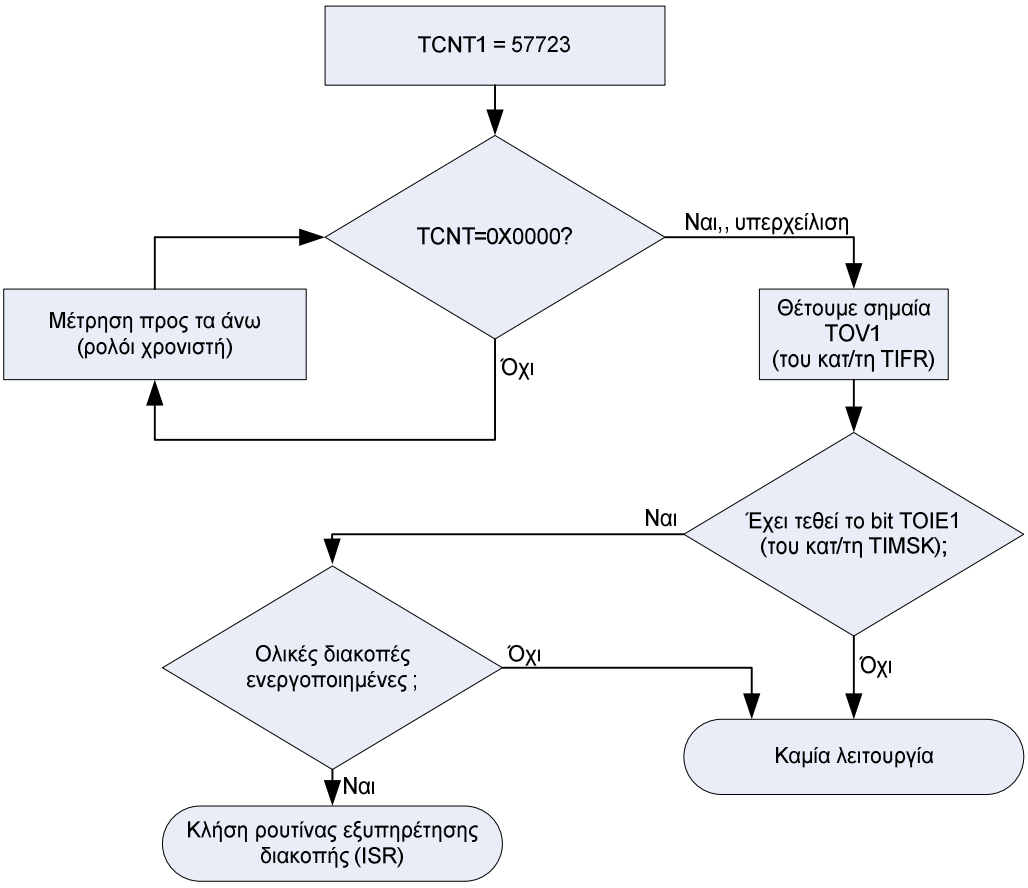
3.3.11 Καταχωρητής εισόδου σύλληψης του TIMER/COUNTER1 (ICR1H-ICR1L)

Πρόκειται για έναν καταχωρητή των 16-bits με δυνατότητα μόνο ανάγνωσης. Η τιμή μέτρησης του μετρητή/ χρονιστή1 μεταφέρεται στον καταχωρητή εισόδου σύλληψης (ICR1H- ICR1L), όταν στον ακροδέκτη εισόδου σύλληψης (ICP) ανιχνευθεί θετικό ή αρνητικό μέτωπο παλμού του ανάλογα με την τιμή του bit ICES1 του καταχωρητή ελέγχου TCCR1B. Παράλληλα ενεργοποιείται η σημαία ICF1. Κατά την ανάγνωση του, απαιτείται βοηθητικός καταχωρητής TEMP, αφού πρόκειται για 16-bit καταχωρητή. Τα ονόματα των καταχωρητών διαφέρουν μεταξύ των χρονιστών και των μικροελεγκτών. Η πρόσβαση στους 16-bit καταχωρητές γίνεται ένα byte τη φορά. Για να επιτύχουμε ακριβή χρονισμό, χρησιμοποιούμε ένα 16-bit προσωρινό καταχωρητή.

Εγγραφή: Όταν εγγράφεται το υψηλό byte (π.χ. TCNT1H), το δεδομένο τοποθετείται στον προσωρινό κατ/τη TEMP , ενώ όταν γράφεται το χαμηλό byte το δεδομένο μεταφέρεται άμεσα στον κατ/τη TCNT1 . Συνεπώς για μια ολοκληρωμένη διαδικασία εγγραφής σε καταχωρητή των 16-bits, το υψηλό byte πρέπει να γραφτεί πρώτο.

Ανάγνωση: Το υψηλό byte διαβάζεται άμεσα από τον καταχωρητή TCNT1, συνεπώς το χαμηλό byte πρέπει να διαβαστεί πρώτο (TCNT1L).

Κανονική λειτουργία: Ο καταχωρητής TCNT1 μετρά προς τα άνω και πυροδοτεί τη διακοπή υπερχείλισης όταν μεταβαίνει από την τιμή 0xFFFF στην 0x0000. Υποθέτοντας κεντρικό ρολόι στα 8 MHz και επιθυμητό χρονιστή 1 δευτερολέπτου, χρειαζόμαστε 8 εκατομμύρια παλμούς. Θέτοντας τον prescaler στα 1024 (ακόμα και 256) έχουμε 8,000,000/1024 = 7812.5 ~ 7813. Άρα η τιμή που πρέπει να φορτώσουμε στον TCNT1 ώστε να προκαλέσει υπερχείλιση σε 1 δευτερόλεπτο είναι 0x0000 - 7813 = 57723. Επιπλέον, θα πρέπει να έχουμε ενεργοποιήσει την διακοπή υπερχείλισης χρονιστή και τις καθολικές διακοπές, όπως φαίνεται στο διάγραμμα:



Σχήμα 3.14: Κανονική λειτουργία TCNT1

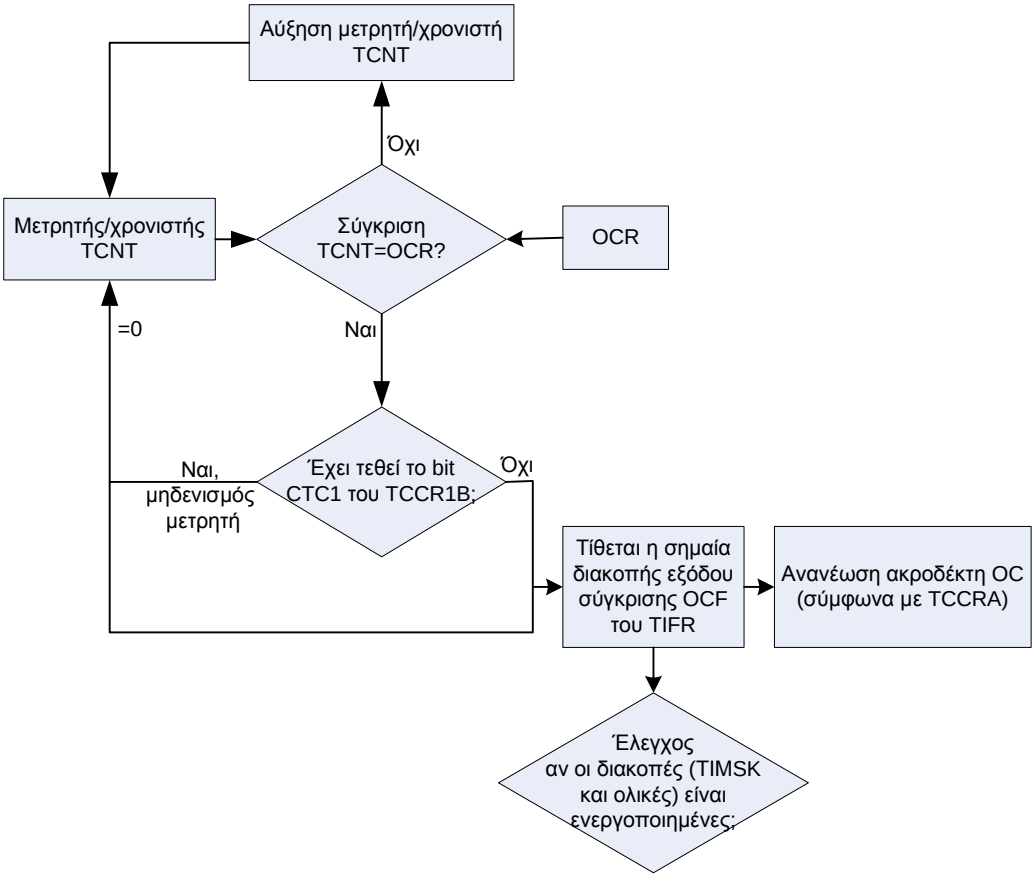
Με απλά λόγια τα βήματα είναι τα εξής:

- θέτουμε τον prescaler (π.χ. στο 1024 θέτοντας τα bits CS12 και CS10 του TCCR1B)
- γράφουμε την τιμή (π.χ. 57723) στον TCNT1
- ενεργοποιούμε το bit TOIE1 του TIMSK
- ενεργοποιούμε τις καθολικές διακοπές του SREG
- αναμονή

3.3.12 Λειτουργία εξόδου σύγκρισης

Ο χρονιστής timer /counter1 υποστηρίζει μια λειτουργία εξόδου σύγκρισης χρησιμοποιώντας τον καταχωρητή εξόδου σύγκρισης (OCR1A) ως την πηγή των δεδομένων που θα συγκριθούν με το περιεχόμενο του καταχωρητή TCNT1. Η τιμή του TCNT1 αυξάνει, αν δεν διακοπεί από τα bits επιλογής του prescaler. Όταν οι δύο τιμές εξισωθούν, η σημαία διακοπής εξόδου σύγκρισης (OCF του TIFR) τίθεται και

καλείται η ρουτίνα εξυπηρέτησης. Θέτοντας το bit CTC1 του TCCR1B, ο χρονιστής μηδενίζεται μετά την επιτυχή συσχέτιση τιμών.



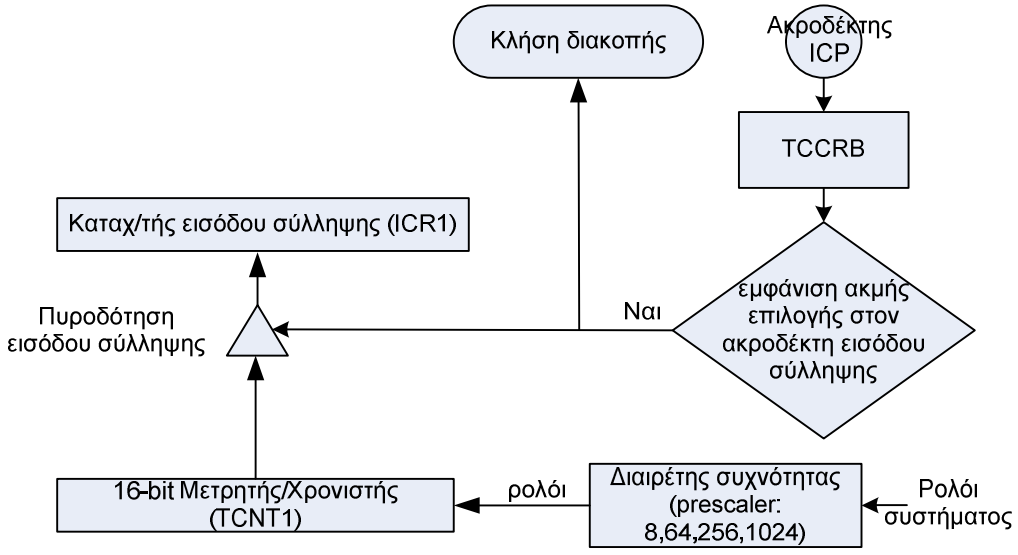
Σχήμα 3.15 Λειτουργία εξόδου σύγκρισης

Με απλά λόγια τα βήματα είναι τα εξής:

- θέτουμε τον prescaler (π.χ. στο 1024 θέτοντας τα bits CS12 και CS10 του TCCR1B)
- γράφουμε την τιμή εκκίνησης στον TCNT1
- γράφουμε τα δεδομένα προς σύγκριση με τον μετρητή στον κατ/τη εξόδου σύγκρισης (OCR1A)
- ενεργοποιούμε το bit OCIE1A του TIMSK
- ενεργοποιούμε τις καθολικές διακοπές του SREG
- αναμονή διακοπής.

3.3.13 Λειτουργία εισόδου σύλληψης

Η λειτουργία εισόδου σύλληψης μπορεί να χρησιμοποιηθεί για να μετρήσει το χρόνο μεταξύ δύο ακμών του ακροδέκτη εισόδου σύλληψης ICP (Input Capture Pin). Κάποια εξωτερικά κυκλώματα δημιουργούν παλμούς, έτσι μπορούμε να μετρήσουμε τις σ.α.λ. ενός κινητήρα, το μήκος ενός παλμού κλπ. Η βασική λειτουργία φαίνεται στο επόμενο διάγραμμα:



Σχήμα 3.16: Λειτουργία εισόδου σύλληψης

Το περιεχόμενο του TCNT1 μεταφέρεται στον ICR1, όταν η ακμή επιλογής εμφανίζεται στον ακροδέκτη εισόδου σύλληψης και η ρουτίνα εξυπηρέτησης μπορεί να κληθεί ώστε να μηδενίσει ή να φορτώσει μια τιμή στον TCNT1. Η ρουτίνα μπορεί ακόμη να μεταβάλλει την ακμή για την επόμενη διακοπή.

3.3.14 Λειτουργία PWM

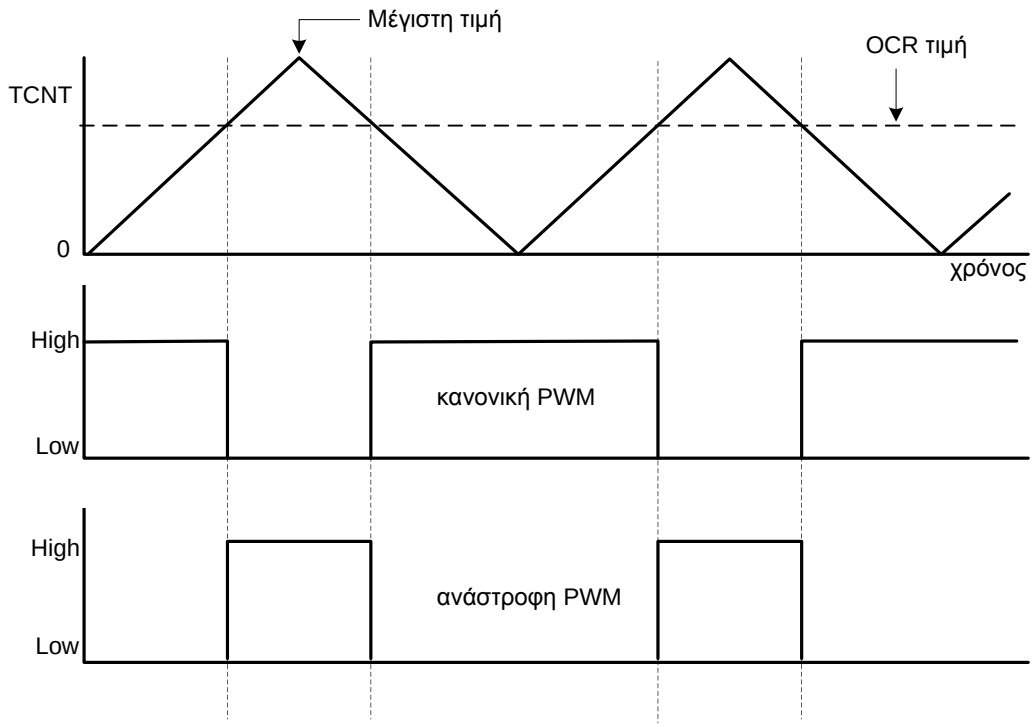
Ο PWM μπορεί να είναι των 8, 9 ή 10 bits. Αυτό επηρεάζει τη συχνότητα (τον χρόνο μεταξύ δύο PWM παλμών) και τη μέγιστη τιμή μέτρησης και επιλέγεται μέσω των bits PWM11,PWM10 του TCCR1A. Ο PWM είναι μια λειτουργία εξόδου σύγκρισης, που έχει τη δυνατότητα αύξουσας και φθίνουσας μέτρησης. Στη λειτουργία αυτή, ο μετρητής μετρά προς τα άνω ωσότου φθάσει τη μέγιστη τιμή. Όταν το περιεχόμενο του καταχωρητή TCNT1 γίνει ίσο με την τιμή του OCR1, ο

ακροδέκτης εξόδου OC1 τίθεται ή μηδενίζεται ανάλογα με τα bits COM1A1,COM1A0 του TCCR1A. Μπορούμε να επιλέξουμε κανονική ή ανάστροφη PWM μέσω αυτών των bits:

Πίνακας : Επιλογή PWM

COM1A1	**COM1A0**	**αποτέλεσμα**
0	0	PWM
0	1	
1	0	κανονική PWM
1	1	ανάστροφη PWM

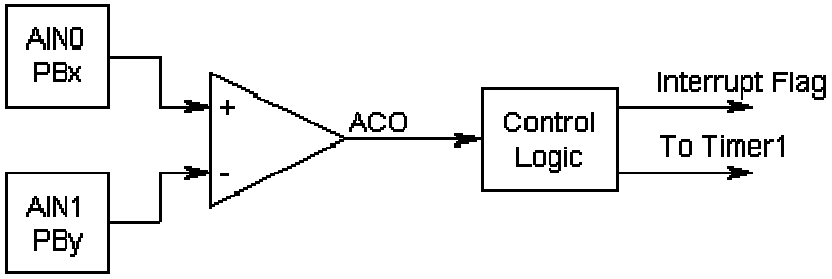
Κανονική PWM σημαίνει ότι ο ακροδέκτης εξόδου OC1 μηδενίζεται, όταν ο μετρητής μετρά προς τα άνω και φτάσει την τιμή OCR1. Ο μετρητής συνεχίζει τη μέτρηση και όταν φτάσει τη μέγιστη τιμή αρχίζει να μετρά προς τα κάτω. Μόλις ξανασυναντήσει την τιμή OCR1 ο ακροδέκτης τίθεται. Στην ανάστροφη μέτρηση, ο ακροδέκτης τίθεται αντί να μηδενίζεται, όπως φαίνεται στο διάγραμμα:



Σχήμα 3.17: Ανάστροφη και κανονική PWM

3.4 Αναλογικός συγκριτής

Ο αναλογικός συγκριτής είναι ένα χρήσιμο περιφερειακό για τη σύγκριση δύο αναλογικών σημάτων. Παραδείγματος χάριν, μπορούμε να συγκρίνουμε την έξοδο ενός αισθητήρα θερμοκρασίας με μια τάση αναφοράς, και να εκτελέσουμε κάποια λειτουργία όταν η θερμοκρασία υπερβαίνει το όριο που αντιστοιχεί στην τάση αυτή. Ο αναλογικός συγκριτής χωρίζεται σε 2 τμήματα. Το πρώτο είναι ο ίδιος ο συγκριτής, ο οποίος έχει δυο εισόδους: αναλογική είσοδο 0 (Analog Input 0 - AIN0) και αναλογική είσοδο 1 (Analog Input 1 - AIN0). Αν η είσοδος AIN0 είναι μεγαλύτερη από την AIN1, τότε η έξοδος του συγκριτή είναι υψηλή. Ενώ όταν η είσοδος AIN1 είναι μεγαλύτερη από την AIN0, τότε η έξοδος του συγκριτή είναι χαμηλή. Το δεύτερο τμήμα παίρνει την έξοδο του συγκριτή και θέτει τη σημαία διακοπής (Analog Comparator Interrupt Flag - ACI) και τη σημαία εξόδου του συγκριτή (Analog Comparator Output Flag - ACO). Ένα απλό σχέδιο του συγκριτή είναι:



Σχήμα 3.18: Σχέδιο συγκριτή

Πιο αναλυτικά, όπως θα δούμε στο παρακάω σχήμα, οι είσοδοι AIN0 και AIN1 αντιστοιχούν στους ακροδέκτες μιας θύρας (π.χ. PORTB), οπότε ο καταχωρητής κατεύθυνσης δεδομένων πρέπει να τεθεί κατάλληλα. Για το λόγο αυτό μηδενίζουμε τους κατ/τες κατεύθυνσης DDBx και DDBy ώστε οι ακροδέκτες να γίνουν είσοδοι. Ακόμη, μηδενίζουμε τους κατ/τες PORTx και PORTy ώστε να απενεργοποιήσουμε τις αντιστάσεις πρόσδεσης.

Σχήμα 3.19:Αναλογικός συγκριτής

Ο συγκριτής είναι σχετικά απλός και διαθέτει μόνο έναν καταχωρητή:

3.4.1 Καταχωρητής ελέγχου και κατάστασης της μονάδας του αναλογικού συγκριτή (ACSR)

Πρόκειται για τον κατ/τη που ελέγχει τη λειτουργία και την έξοδο του αναλογικού συγκριτή.

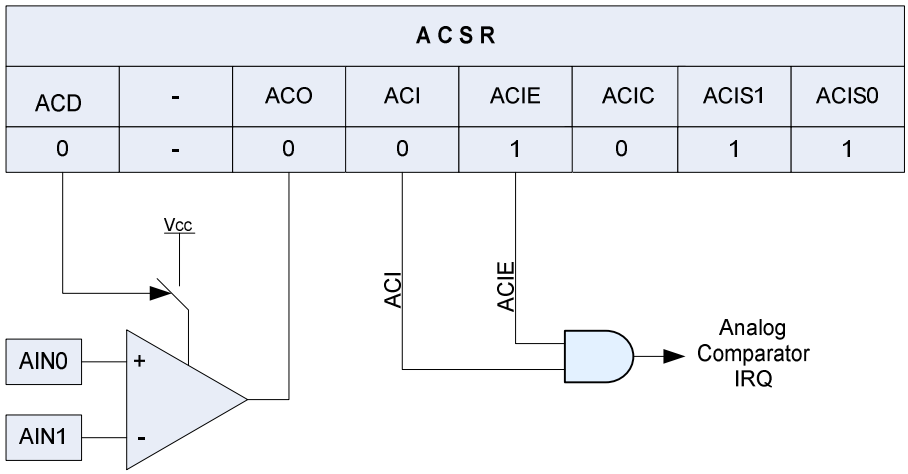
Πίνακας: Καταχωρητής ελέγχου-κατάστασης συγκριτή (ACSR)

Bit	Σύμβολο	Σημασία	Λειτουργία
7	ACD	Απενεργοποίηση του αναλογικού συγκριτή.	1: Απενεργοποίηση του συγκριτή
6	-	Δεν χρησιμοποιείται.	
5	ACO	Έξοδος της μονάδας του συγκριτή.	
4	ACI	Σημαία διακοπής. Εκτέλεση της αντίστοιχης ρουτίνας εξυπηρέτησης διακοπής εφόσον έχουν τεθεί οι διακοπές και το bit ACIE.	1: Αίτηση διακοπής
3	ACIE	Ενεργοποίηση διακοπής του συγκριτή.	1: Ενεργοποίηση διακοπής
2	ACIC	Σύνδεση της εξόδου του συγκριτή με την είσοδο σύλληψης του χρονιστή1.	1: σύνδεση με είσοδο σύλληψης του timer1
1	ACIS1	Καθορισμός του τρόπου διέγερσης της διακοπής.	00: διακοπή στην
			01: -
0	ACIS0		10: διακοπή στο
			11:διακοπή στο μέτωπο ανόδου

Bits	7	6	5	4	3	2	1	0
	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0

Παραθέτουμε τον κώδικα αρχικοποίησης του συγκριτή (με την προϋπόθεση ACIE=0):

```
sbi ACSR,ACI          ; μηδενισμός της σημαίας ACI εγγράφοντας τη μονάδα!
sei                   ; ενεργοποίηση καθολικών διακοπών
sbi ACSR, ACIS0       ; ενεργοποίηση διακοπών
sbi ACSR, ACIS1       ; στο μέτωπο ανόδου
sbi ACSR, ACIE        ; ενεργοποίηση διακοπής του συγκριτή
```



Σχήμα 3.20: Πυροδότηση διακοπής του συγκριτή

(Σχόλιο περί υλικού: Ο AVR έχει επίπεδα τάσης TTL(5V ή 3.3V για υψηλή λογική στάθμη και 0V για χαμηλή λογική στάθμη). Όμως σε περίπτωση σύνδεσης του μικροελεγκτή AVR με τον υπολογιστή πρέπει να χρησιμοποιηθούν επίπεδα τάσης RS-232. Για την μετατροπή στα επίπεδα RS-232 απαιτείται ένα μετατροπέας τάσης, όπως ο MAX232).

3.5 Η Μνήμη EEPROM

Οι μικροελεγκτές AVR διαθέτουν ενσωματωμένη μνήμη EEPROM. Το μέγεθος της κυμαίνεται από 64 bytes (στα μοντέλα AT Tiny10,AT Tiny12 και AT90S1200) έως 4096 bytes (στο μοντέλο AT Mega103). Υπάρχουν τρεις υπεύθυνοι καταχωρητές για την πρόσβαση στη μνήμη EEPROM. Αυτοί είναι: ο καταχωρητής δεδομένων (EEDR), ο καταχωρητής διευθύνσεων (EEAR) και ο καταχωρητής ελέγχου.

(EECR). Σε μικροελεγκτές με μνήμη άνω των 256 bytes,ο καταχωρητής διευθύνσεων χωρίζεται στους EEARH και EEARL. Η λειτουργία τους εξηγείται παρακάτω:

3.5.1 Καταχωρητής διευθύνσεων της μνήμης EEPROM (EEAR)

Ο καταχωρητής αυτός περιέχει τη διεύθυνση της μνήμης EEPROM. Σε περίπτωση επανεκκίνησης ο καταχωρητής διατηρεί τα δεδομένα του.

3.5.2 Καταχωρητής δεδομένων της μνήμης EEPROM (EEDR)

Για τη διαδικασία εγγραφής στη μνήμη EEPROM, ο καταχωρητής αυτός περιέχει τα δεδομένα που πρέπει να γραφτούν στη διεύθυνση που δίνεται από τον καταχωρητή EEAR. Για τη διαδικασία ανάγνωσης από τη μνήμη EEPROM, περιέχει τα δεδομένα που διαβάστηκαν από τη διεύθυνση που δίνεται από τον καταχωρητή EEAR. Έχει μήκος 8-bits.

3.5.3Καταχωρητής ελέγχου της μνήμης EEPROM (EECR)

Ο καταχωρητής αυτός ελέγχει τις διαδικασίες εγγραφής και ανάγνωσης στην EEPROM. Η σημασία των bits ελέγχου είναι η εξής:

- Bit 3 (EERIE- EEPROM Ready Interrupt Enable): Όταν το I-bit του SREG και το EERIE είναι ‘1’, τότε οι διακοπές της μνήμης είναι ενεργοποιημένες για την περίπτωση που το bit EEWE μηδενίζεται. Διαφορετικά απενεργοποιούνται.
- Bit 2 (EEMWE-EEPROM Master Write Enable): Όταν το bit αυτό τίθεται ίσο με ‘1’ και εφόσον θέσουμε και το bit EEWE τότε θα εγγραφούν

δεδομένα στη EEPROM. Μετά από 4 περιόδους ρολογιού το υλικό μηδενίζει το bit αυτό.

- Bit 1 (EEME- EEPROM Write Enable): Η διαδικασία εγγραφής ενεργοποιείται, όταν αυτό το bit τίθεται ίσο με ‘1’ και εφόσον έχουμε θέσει και το bit EEMWE. Το περιεχόμενο του καταχωρητή EEDR εγγράφεται στη διεύθυνση της μνήμης που δείχνει ο EEAR. Η διαδικασία εγγραφής στην EEPROM είναι:

  i. αναμονή ωσότου το bit EEWE μηδενιστεί
  ii. εγγραφή της διεύθυνσης των δεδομένων στον EEAR
  iii. εγγραφή των δεδομένων στον EEDR
  iv. θέτουμε το bit EEMWE του EECR
  v. μετά από 4 περιόδους ρολογιού θέτουμε και το bit EEWE

- Bit 0 (EERE: EEPROM Read Enable): Η διαδικασία ανάγνωσης ενεργοποιείται, όταν αυτό το bit τίθεται ίσο με ‘1’. Συγκεκριμένα, ελέγχουμε αρχικά το bit EEWE ώστε να μην βρίσκεται διαδικασία εγγραφής σε εξέλιξη, τοποθετούμε την επιθυμητή διεύθυνση στον καταχωρητή EEAR και θέτουμε το bit EERE. Έπειτα το υλικό μηδενίζει το bit EERE και τα δεδομένα μεταφέρονται αυτόματα στον καταχωρητή EEDR.

Bits ελέγχου	7	6	5	4	3 EERIE	2 EEMWE	1 EEWE	0 EERE

3.5.4 Κώδικας για τη διαχείριση της EEPROM.

Ανάγνωση και εγγραφή στη μνήμη EEPROM. Χρησιμοποιούμε ένα μετρητή των θέσεων μνήμης της EEPROM. Σε κάθε επανεκκίνηση του επεξεργαστή ο μετρητής αυξάνει και το περιεχόμενο απεικονίζεται σε δεκαεξαδική μορφή στα LEDs.

```
   .NOLIST
.INCLUDE "8515def.inc"      ; Δήλωση μικροελεγκτή
.LIST                                   ; Δήλωση σταθερών
```

```
   .equ   cnt=$0000          ; διεύθυνση του μετρητή στην EEPROM
.def   mpr=R16                    ; Δήλωση καταχωρητών

   .def   neu=R17                 ; κατ/της για προσωρινή αποθήκευση
                                  ; Διάνυσμα επανεκκίνησης- διακοπών

     RJMP   main                  ; επιστροφή στο κυρίως πρόγραμμα
main:                             ; κυρίως πρόγραμμα
  LDI   mpr,$FF           ; τα bits της θύρας Port B γίνονται έξοδοι

     OUT   DDRB,mpr

                          ; Ανάγνωση ενός byte από τη μνήμη EEPROM

     LDI   mpr,LOW(cnt)   ; Φόρτωση της διεύθυνσης  EEPROM
  OUT   EEARL,mpr         ; στον κατ/τη διευθύνσεων EEAR


  LDI   mpr,HIGH(cnt)     ; ξεχωριστά για το υψηλό και το χαμηλό byte
  OUT   EEARH,mpr               ;όταν η EEPROM έχει 512 Byte μνήμης
  SBI   EECR,EERE         ; Θέτουμε το Read-Enable-Bit EERE,
                                ; του κατ/τη ελέγχου της  EEPROM, EECR
  IN   neu,EEDR           ; ανάγνωση byte
  INC   neu               ; Αύξηση κατ/τη


wait:                     ; Αναμονή όταν η EEPROM είναι απασχολημένη

     SBIC   EECR,1        ; Παράκαμψη επόμενης εντολής όταν το bit EEWE του
EECR είναι1.
  RJMP   wait         ; βρόχος αναμονής. Έξοδος όταν η μνήμη είναι έτοιμη
  OUT   EEDR,neu    ; Φόρτωση νέων δεδομένων στον κατ/τη δεδομένων EEDR
                          ; Οι επόμενες 2 εντολές δεν πρέπει να διακοπούν ώστε
η διαδικασία                      ; εγγραφής να είναι έγκυρη. Για αυτό
απενεργοποιούμε τις διακοπές.
  CLI

                    ; Εγγραφή δεδομένων στη μνήμη
```

```
    SBI   EECR,EEMWE     ; Θέτουμε το bit EEMWE(Master Write Enable)
  SBI   EECR,EEWE        ; Θέτουμε το bit EEWE(Write Enable) του κατ/τη

                         ; ελέγχου της EEPROM για ενεργοποίηση της
διαδικασίας εγγραφής.
                         ; Το byte εγγράφεται εντός 1,5 milliseconds στη μνήμη.
                         ; Αναστρέφουμε το περιεχόμενο του μετρητή και το
μεταφέρουμε στα

                    ; LEDs της PortB
                         ; Το πρόγραμμα τελειώνει με ένα ατέρμονα βρόχο.
  COM   neu          ; αντιστροφή
  OUT   PORTB,neu    ; έξοδος στη θύρα Port B
loop:  RJMP   loop   ; ατέρμον βρόχος.

                          ; Αποθήκευση του κώδικα στο τμήμα μνήμης
EEPROM (και όχι στο

                               ; code  segment) και μηδενισμός του μετρητή

  .ESEG

  .DB   $00                ; αποθήκευση ενός μηδενικού byte στην EEPROM.

                         ; τέλος
```

ΚΕΦΑΛΑΙΟ 4^{Ο} - Η ΓΛΩΣΣΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ C

Για να κατανοήσουμε τι είναι μια γλώσσα προγραμματισμού θα πρέπει να τη δούμε σαν μία «τεχνητή γλώσσα», η οποία μπορεί να χρησιμοποιηθεί για τον έλεγχο μιας μηχανής, συνήθως ενός υπολογιστή. Οι γλώσσες προγραμματισμού (όπως άλλωστε και οι ανθρώπινες γλώσσες) ορίζονται από ένα σύνολο συντακτικών και εννοιολογικών κανόνων, που ορίζουν τη δομή και το νόημα, αντίστοιχα, των προτάσεων της γλώσσας. Οι γλώσσες προγραμματισμού παρέχουν ένα γενικό πλαίσιο για την οργάνωση των εντολών που μπορούν να μεταφραστούν σε κώδικα μηχανής, συνήθως μ' ένα ξεχωριστό πρόγραμμα που λέγεται μεταγλωττιστής. Για τον προγραμματισμό χρησιμοποιούνται διάφορες γλώσσες. Οι περισσότεροι χρησιμοποιούν γράμματα, λέξεις και σύμβολα. Η τοποθέτησή τους σε συγκεκριμένη σειρά καθορίζει το τι θα συμβεί όταν ο υπολογιστής διαβάσει κάθε γραμμή του προγράμματος. Ο προγραμματιστής πρέπει να διαλέξει την καλύτερη γλώσσα προγραμματισμού για τις απαιτήσεις του συγκεκριμένου λογισμικού. Οι πιο συνηθισμένες γλώσσες προγραμματισμού είναι η Basic, η Logo, η Cobol και η Fortran. Οι συμβολικές γλώσσες είναι συγκεκριμένες για κάθε τύπο CPU και μεταφράζουν εντολές στη βασικότερη γλώσσα που ονομάζεται γλώσσα μηχανής

Οι γλώσσες προγραμματισμού χρησιμοποιούνται για να διευκολύνουν την οργάνωση και διαχείριση πληροφοριών, αλλά και για την ακριβή διατύπωση αλγορίθμων. Ορισμένοι ειδικοί χρησιμοποιούν τον όρο *γλώσσα προγραμματισμού* μόνο για τυπικές γλώσσες που μπορούν να εκφράσουν όλους τους πιθανούς αλγορίθμους. Μη-υπολογιστικές γλώσσες όπως η HTML ή τυπικές γραμματικές όπως η BNF δεν λέγονται συνήθως γλώσσες προγραμματισμού. Υπάρχουν χιλιάδες διαφορετικές γλώσσες προγραμματισμού, και κάθε χρόνο δημιουργούνται περισσότερες.

Κάθε γλώσσα προγραμματισμού έχει το δικό της σύνολο τυπικών προδιαγραφών (ή κανόνων) που αφορούν το συντακτικό, το λεξιλόγιο και το νόημα της. Για τις περισσότερες γλώσσες που χρησιμοποιούνται ευρέως και έχουν χρησιμοποιηθεί για αρκετό χρονικό διάστημα, υπάρχουν ειδικοί οργανισμοί τυποποίησης οι οποίοι μέσα από τακτές συναντήσεις δημιουργούν, τροποποιούν ή επεκτείνουν τις τυπικές προδιαγραφές που διέπουν την χρήση μιας γλώσσας προγραμματισμού.

Δεν υπάρχει απλός τρόπος να κατηγοριοποιηθούν οι γλώσσες προγραμματισμού. Αυτό συμβαίνει γιατί συνήθως κάθε γλώσσα προγραμματισμού περιέχει επιρροές από πολλές προηγούμενες γλώσσες, συνδιάζοντας θετικά στοιχεία και προσθέτοντας νέα. Χαρακτηριστικά που εμφανίζονται σε μια γλώσσα και έχουν θετική αποδοχή, συνήθως υιοθετούνται από μεταγενέστερες γλώσσες ακόμα και αν πρόκειται για γλώσσες που ανήκουν σε διαφορετική κατηγορία. Η κατηγοριοποίηση είναι ακόμα πιο περίπλοκη για το λόγο ότι πολλές γλώσσες συνήθως ανήκουν σε παραπάνω από μία κατηγορίες. Για παράδειγμα, η Java είναι τόσο αντικειμενοστρεφής όσο και παράλληλη γλώσσα, δεδομένου ότι υποστηρίζει την οργάνωση των δεδομένων και υπολογισμών σε αντικείμενα, αλλά επιτρέπει επίσης και την δημιουργία προγραμμάτων με ταυτόχρονα threads που εκτελούνται παράλληλα.

Δεδομένης της δυσκολίας στην ομαδοποίηση, μπορούμε να κατηγοριοποιήσουμε τις γλώσσες προγραμματισμού με διάφορους τρόπους. Οι συνηθέστεροι τρόποι είναι:

- με βάση τον τρόπο οργάνωσης του προγράμματος,
- με βάση τον στόχο που έχει η γλώσσα,
- με βάση τον τρόπο που περιγράφουν το ζητούμενο αποτέλεσμα.

Στην πρώτη περίπτωση προκύπτουν κατηγορίες όπως:

- **Διαδικαστικές γλώσσες** (procedural) όπου το πρόγραμμα είναι οργανωμένο σε διαδικασίες, που αποτελούνται απο σειρές εντολών που περιγράφουν αλγορίθμους. Παραδείγματα γλωσσών που ανήκουν σε αυτήν την κατηγορία είναι η Pascal ή η C.
- **Αντικειμενοστρεφείς γλώσσες** (object-oriented) όπου το πρόγραμμα είναι οργανωμένο σε αντικείμενα. Ένα αντικείμενο είναι μια μονάδα που αποτελείται από την περιγραφή κάποιων δεδομένων και την περιγραφή των αλγορίθμων που τα επεξεργάζονται. Ένα αντικειμενοστρεφές πρόγραμμα αποτελείται από διάφορα αντικείμενα που αλληλεπιδρούν μεταξύ τους. Παραδείγματα αντικειμενοστρεφών γλωσσών είναι η Java ή η C++.
- **Συναρτησιακές γλώσσες** (functional) όπου οι υπολογισμοί εκφράζονται ως εφαρμογές μαθηματικών συναρτήσεων, σε αντίθεση με τα άλλα είδη προγραμματισμού όπου οι υπολογισμοί εκφράζονται ως σειρές εντολών, όπου η κάθε μία αλλάζει με κάποιο τρόπο την κατάσταση του συστήματος.

Θεωρητικό τους υπόβαθρο είναι ο λ-λογισμός. Χαρακτηριστικές συναρτησιακές γλώσσες είναι η Lisp, η Haskell και η OCaml.

Στην περίπτωση που οι κατηγοριοποίηση των γλωσσών προγραμματισμού γίνει με βάση το στόχο που έχει η γλώσσα, υπάρχουν οι παρακάτω κατηγορίες:

- **Γλώσσες γενικής χρήσης**. Σ'αυτή την κατηγορία ταξινομούνται γλώσσες που δημιουργήθηκαν για τον προγραμματισμό γενικών εφαρμογών, καθώς και πολλές εκπαιδευτικές γλώσσες που αποδείκτηκαν χρήσιμες για την ανάπτυξη γενικών εφαρμογών, όπως η Pascal.
- **Γλώσσες προγραμματισμού συστημάτων**, που χρησιμοποιούνται συνήθως για τον προγραμματισμό λειτουργικών συστημάτων ή οδηγών (drivers) υλικού, όπου χρειάζεται πολλές φορές ο προγραμματιστής να έχει έλεγχο και γνώση του πως λειτουργεί το υλικό. Η πιο συχνά χρησιμοποιούμενη γλώσσα προγραμματισμού συστημάτων είναι η C.
- **Γλώσσες σεναρίων** (scripting). Αυτές οι γλώσσες χρησιμοποιούνται συνήθως για τη γρήγορη ανάπτυξη μικρών προγραμμάτων, σε περιπτώσεις που ο χρόνος του προγραμματιστή είναι πιο πολύτιμος από την ταχύτητα εκτέλεσης του προγράμματος, όπως για παράδειγμα συμβαίνει όταν το πρόγραμμα απλά αυτοματοποιεί απλές λειτουργίες. Παραδείγματα σεναριακών (scripting) γλωσσών είναι η Perl, η Python, η Ruby ή τα κελύφη του λειτουργικού συστήματος Unix (shells).
- **Γλώσσες ειδικών εφαρμογών**. Σε αυτή την κατηγορία ανήκουν γλώσσες που αναπτύχθηκαν ειδικά για μια συγκεκριμένη εφαρμογή. Για παράδειγμα, η γλώσσα PostScript είναι σχεδιασμένη ειδικά για να περιγράφονται με λεπτομέρεια κείμενα προς εκτύπωση, ή η γλώσσα Matlab είναι σχεδιασμένη για την επεξεργασία πινάκων από αριθμητικά δεδομένα.
- **Παράλληλες ή κατανεμημένες γλώσσες**. Στη συγκεκριμένη κατηγορία ταξινομούνται γλώσσες που επιτρέπουν τη ανάπτυξη παράλληλων προγραμμάτων, όπου πολλές εντολές εκτελούνται ταυτόχρονα σε πολλούς υπολογιστές, έτσι ώστε το τελικό αποτέλεσμα να προκύψει γρηγορότερα. Οι παράλληλες γλώσσες προσφέρουν συνήθως εύκολους τρόπους επικοινωνίας μεταξύ των νημάτων που εκτελούνται παράλληλα, καθώς και τρόπους ώστε να δημιουργούνται καινούριες παράλληλες εκτελέσεις. Παραδείγματα

γλωσσών που ανήκουν (και) σε αυτή την κατηγορία είναι η Java, η Erlang, η MultiLisp ή η Cilk.

Τέλος, στην περίπτωση που η κατηγοριοποίηση γίνεται με βάση τον τρόπο που περιγράφεται το ζητούμενο, υπάρχουν οι παρακάτω κατηγορίες:

- **Προστακτικές γλώσσες προγραμματισμού** (imperative) είναι οι γλώσσες που περιγράφουν το ζητούμενο αποτέλεσμα κατασκευαστικά, δίνοντας μια σειρά εντολών που όταν εκτελεστούν παράγουν το ζητούμενο αποτέλεσμα. Τέτοιες γλώσσες είναι η C, η Java αλλά και η OCaml.
- **Δηλωτικές γλώσσες προγραμματισμού** (declarative) είναι οι γλώσσες που περιγράφουν το ζητούμενο αποτέλεσμα χρησιμοποιώντας τις ιδιότητες που έχει, και όχι τον τρόπο με τον οποίο υπολογίζεται. Παραδείγματα δηλωτικών γλωσσών είναι η Haskell, η SQL και η Prolog.

Υπάρχουν οι ακόλουθες γλώσσες προγραμματισμού στην αγορά σήμερα, οι οποίες παρατίθενται με αλφαβητική σειρά:

- ActionScript
- Ada
- Algol
- APL
- AWK
- BASIC
- C
- C++
- C#
- Cilk
- Clojure
- COBOL
- Datalog
- Eiffel
- Erlang
- Forth
- FORTRAN
- Haskell
- Java
- JavaScript
- Lisp
- Logo
- Lua
- Lucid
- Mathematica
- Matlab
- Miranda
- ML
- Modula-2
- OBJ / Σύστημα Maude
- Objective-C
- OCaml
- Pascal
- Perl
- PHP
- PostScript
- Prolog
- Python
- RPG
- Ruby
- Scala
- Scheme
- Simula
- Smalltalk
- SQL
- Tcl
- Visual Basic

Η παραπάνω λίστα είναι ενδεικτική και σε καμία περίπτωση δεν εξαντλεί το εύρος και την ποικιλία των χιλιάδων γλωσσών που χρησιμοποιούνται στην πράξη. Στο κεφάλαιο αυτό θα επιχειρήσουμε να παρουσίασουμε τη γλώσσα προγραμματισμού C. Είναι φανερό ότι κάθε λογισμικό πρόγραμμα πλέον, εξαιτίας της πολυπλοκότητας και των σύνθετων εφαρμογών τα οποία υλοποιούν απαιτούν από τους επεξεργαστές εφαρμογής, τη χρήση γλωσσών προγραμματισμού υψηλού επιπέδου για τη διευκόλυνση κάθε προγραμματιστή. Από τις γλώσσες αυτές, η C είναι μια σχετικά μινιμαλιστική γλώσσα προγραμματισμού. Θεωρείται γενικά γλώσσα μέσου επιπέδου και αυτό γιατί συνδυάζει στοιχεία των γλωσσών υψηλού επιπέδου (high level languages), όπως είναι η Cobol και η Pascal και στοιχεία των γλωσσών χαμηλού επιπέδου (low level languages), όπως είναι η Assembly.

Για να διευκρινίσουμε, πρέπει να πούμε ότι σαν *γλώσσα υψηλού επιπέδου* θεωρείται η γλώσσα εκείνη που είναι αρκετά περιγραφική και που είναι έτσι πιο κοντά στην ανθρώπινη γραπτή γλώσσα και σαν *γλώσσα χαμηλού επιπέδου* θεωρείται εκείνη που είναι πιο κοντά στη μηχανή. Αυτός ο χαρακτηρισμός δεν έχει καμία απολύτως σχέση με τις δυνατότητες της γλώσσας. Σίγουρα, η κάθε γλώσσα προγραμματισμού έχει κάποιες προτεραιότητες να εκπληρώσει. Η ***Pascal***, για παράδειγμα, χρησιμοποιείται κυρίως για τη σωστή διδασκαλία των αρχών του προγραμματισμού, ενώ η ***Basic*** δημιουργήθηκε έτσι ώστε να δώσει τη δυνατότητα σε αρχάριους στον προγραμματισμό να κάνουν με άνεση και ευκολία τα πρώτα τους βήματα στον ιδιόμορφο αυτό χώρο. Η Clipper και η Cobol είναι καθαρά επαγγελματικές γλώσσες προγραμματισμού.

Η C, όμως, μπόρεσε να φέρει τον προγραμματιστή *πιο κοντά στο* ***hardware***, που με τις άλλες γνωστές γλώσσες προγραμματισμού κάτι τέτοιο θα ήταν πολύ δύσκολο να γίνει. Βέβαια, η κάθε γλώσσα προγραμματισμού κάνει και διαφορετική δουλειά και δεν θα ήταν σωστό να κάνουμε συγκρίσεις, για τον ίδιο λόγο που δεν μπορούμε να συγκρίνουμε ένα αεροπλάνο μ' ένα ποδήλατο καθώς το καθένα είναι προορισμένο να κάνει διαφορετική δουλειά. Ανάμεσα στους σχεδιαστικούς στόχους που έπρεπε να καλύψει η γλώσσα περιλαμβανόταν το ότι θα μπορούσε να μεταγλωττιστεί (να γίνεται compile) άμεσα με τη χρήση single-pass compiler — με άλλα λόγια, ότι θα απαιτούνταν μόνο ένας μικρός αριθμός από εντολές (instructions) σε γλώσσα μηχανής (machine language) για κάθε βασικό στοιχείο της, χωρίς εκτεταμένη run-

time υποστήριξη. Ως αποτέλεσμα, είναι δυνατό να γραφτεί κώδικας σε C σε low level επίπεδο προγραμματισμού με ακρίβεια ανάλογη της συμβολικής γλώσσας, στην πραγματικότητα η C ορισμένες φορές αποκαλείται (και χωρίς να υπάρχει πάντα αντιπαράθεση) "high-level assembly" ή "portable assembly." Επίσης, γίνονται αναφορές στη C ως mid-level γλώσσα προγραμματισμού.

4.1 Ιστορική αναδρομή

Η **C** είναι μια γενικής χρήσης διαδικαστική γλώσσα προγραμματισμού η οποία αναπτύχτηκε σε πρώτη φάση, στα AT&T Bell Labs ανάμεσα στο 1969 και το 1973. Σύμφωνα με τον D. Ritchie, η πιο δημιουργική περίοδος υπήρξε το 1972. Η νέα γλώσσα ονομάστηκε "C" λόγω του ότι πολλά από τα χαρακτηριστικά της προήλθαν από μια παλαιότερη γλώσσα, η οποία ονομαζόταν "B". Οι πηγές δεν επιτρέπουν την πλήρη εξακρίβωση για την προέλευση του ονόματος "B": ο Ken Thompson το παρουσιάζει ως απλούστευση μιας έκδοσης της γλώσσας προγραμματισμού BCPL, αλλά είχε επίσης δημιουργήσει μία γλώσσα που ονομαζόταν Bon προς τιμήν της συζύγου του Bonnie.

Μέχρι το 1973, η C είχε γίνει αρκετά ισχυρή και αποτελεσματική, ώστε το μεγαλύτερο μέρος του πυρήνα του UNIX (UNIX kernel), γραμμένο αρχικά σε PDP-11/20 assembly, επανεγγράφηκε σε C. Ήταν ένας από τους πρώτους πυρήνες που υλοποιήθηκε σε μια γλώσσα διαφορετική της assembly. (Προηγούμενα παραδείγματα περιλαμβάνουν το Multics system (γραμμένο σε PL/I), και το MCP (Master Control Program) για το Burroughs B5000 γραμμένο σε ALGOL το 1961).

K&R C

Το 1978, ο Dennis Ritchie και ο Brian Kernighan δημοσίευσαν την πρώτη έκδοση του *"The C Programming Language"*. Το συγκεκριμένο βιβλίο, γνωστό στους προγραμματιστές της C ως "K&R", χρησίμευσε πολλά χρόνια ως ένας ανεπίσημος ορισμός της γλώσσας. Η έκδοση της C που περιγράφει αναφέρεται συνήθως ως "K&R C." ή "Common C". (Η δεύτερη έκδοση του βιβλίου καλύπτει το μεταγενέστερο πρότυπο ANSI για τη C (ANSI C standard), βλ. συνέχεια.)

Το K&R εισήγαγε τα παρακάτω χαρακτηριστικά στη γλώσσα:

- struct data types
- long int data type
- unsigned int data type
- Ο τελεστής (operator) =+ αλλάχθηκε σε += για να αφαιρεθεί η αμφισβήτηση σημαντικότητας (semantic ambiguity) που δημιουργούνταν με κατασκευές όπως i=+10, που μπορούσε να μεταφραστεί είτε i =+ 10 είτε i = +10.

Η K&R C συχνά λογίζεται ως το βασικό μέρος της γλώσσας που πρέπει να υποστηρίζει ένας C compiler. Για αρκετά χρόνια, ακόμη και μετά την εισαγωγή της ANSI C, θεωρούνταν ο "ελάχιστος συνήθης παρονομαστής" στον οποίο έπρεπε να προσαρμοστούν οι προγραμματιστές της C σε περιπτώσεις κατά τις οποίες ήταν επιθυμητή η μέγιστη μεταφερσιμότητα (portability), καθώς δεν είχε γίνει ενημέρωση (update) σε όλους τους compilers για πλήρη υποστήριξη της ANSI C, και διότι με προσοχή, ο κώδικας σε K&R C μπορούσε να γραφεί ώστε να είναι σύμφωνος και με το πρότυπο ANSI.

ANSI C

Το 1983, το American National Standards Institute (ANSI) όρισε επιτροπή, τη X3J11, για να δώσει ένα σύγχρονο, πλήρη ορισμό της C. Μετά από μακρά και επίπονη επεξεργασία, το πρότυπο (standard) ολοκληρώθηκε το 1989 και επικυρώθηκε ως as ANSI X3.159-1989 "Programming Language C." Η συγκεκριμένη έκδοση της γλώσσας ονομάζεται συχνά ANSI C, ή ορισμένες φορές C89 (για να διαχωρίζεται από τη C99).

Το 1990, το πρότυπο ANSI για τη C (με ορισμένες μικρές τροποποιήσεις) υιοθετήθηκε από τον Οργανισμό Διεθνών Προτύπων (International Organization for Standardization (ISO)) ως ISO/IEC 9899:1990. Αυτή η έκδοση καλείται C90. Επομένως, οι όροι "C89" και "C90" αναφέρονται ουσιαστικά στην ίδια γλώσσα.

Ένας από τους στόχους της διαδικασίας δημιουργίας του προτύπου ANSI για τη C ήταν να δημιουργήσει ένα υπερσύνολο της K&R C, το οποίο θα απορροφούσε πολλά χαρακτηριστικά που είχαν εισαχθεί στην πορεία. Παρόλα αυτά, η επιτροπή συμπεριέλαβε και ορισμένα νέα χαρακτηριστικά, όπως function prototypes (δανεισμένα από τη C++), και ένα πιο ικανό προεπεξεργαστή (preprocessor). Η σύνταξη για τους ορισμούς παραμέτρων άλλαξε επίσης, ώστε να αντικατοπτρίζει το στυλ της C++.

C99

Μετά τη διαδικασία καθορισμού του προτύπου ANSI, ο ορισμός της γλώσσας C παρέμενε σχετικά σταθερός για ορισμένο καιρό, ενώ η C++ συνέχιζε να αναπτύσσεται. (Normative Amendment 1 δημιούργησε μία νέα έκδοση της γλώσσας C το 1995, αλλά σπάνια είναι γνωστή.) Ωστόσο, το πρότυπο επανεξετάστηκε προς το τέλος της δεκαετίας του '90, γεγονός που οδήγησε στην έκδοση του ISO 9899:1999 το 1999. Το πρότυπο αυτό συχνά αναφέρεται ως "C99". Υιοθετήθηκε ως πρότυπο ANSI το Μάρτιο του 2000.

Ο GCC και μερικοί άλλοι C compilers υποστηρίζουν πλέον τα περισσότερα χαρακτηριστικά του C99. Ωστόσο, υπάρχει μικρότερη υποστήριξη από εταιρίες όπως η Microsoft και η Borland που εστίασαν περισσότερο στη C++, καθώς η C++ παρέχει παρόμοια λειτουργικότητα και συχνά ασύμβατους τρόπους (π.χ., η complex template class). Ο Brandon Bray από τη Microsoft είπε "Σε γενικές γραμμές, έχουμε δει μικρές απαιτήσεις για πολλά χαρακτηριστικά του C99. Μερικά χαρακτηριστικά έχουν μεγαλύτερη ζήτηση από άλλα, και θα τη λάβουμε υπόψιν μας σε μελλοντικές releases εφόσον είναι συμβατά με τη C++." [1]

Ακόμη και ο GCC με την εκτεταμένη υποστήριξη του C99 ακόμη δεν προσεγγίζει μια καθ'όλα συμβατή υλοποίηση, ορισμένα χαρακτηριστικά-κλειδιά λείπουν ή δεν λειτουργούν σωστά.[2]

C

Η γλώσσα C χρησιμοποιήθηκε για την ανάπτυξη του λειτουργικού συστήματος Unix γλώσσα με ισχυρά χαρακτηριστικά, μερικά από αυτά κοινά με την Pascal

κατάλληλη για την ανάπτυξη δομημένων εφαρμογών αλλά και με πολλές δυνατότητες γλώσσας χαμηλού επιπέδου. Η C εξελίχθηκε στη γλώσσα C++, που είναι αντικειμενοστραφής. Η γλώσσα C είναι άρρηκτα συνδεδεμένη με το λειτουργικό σύστημα UNIX.

Δημιουργήθηκε στις αρχές της δεκαετίας του 1970 με σκοπό να είναι μια γλώσσα προγραμματισμού που θα διευκόλυνε ακριβώς τη δουλειά της δημιουργίας ενός νέου λειτουργικού συστήματος το οποίο θα ήταν γραμμένο σε μια portable (εύκολα μεταφερόμενη) γλώσσα υψηλού επιπέδου, εκτός από ένα μικρό κομμάτι του που αναγκαστικά θα γραφόταν σε γλώσσα assembly (τη γλώσσα που βρίσκεται στο χαμηλότερο σχεδόν επίπεδο, δηλαδή πλησιέστερα στη μηχανή, και όπου κάθε παραμικρό κομμάτι της μηχανής ελέγχεται πλήρως). Αυτό το χαρακτηριστικό του λειτουργικού συστήματος Unix βοήθησε τρομερά στο να λειτουργήσει αυτό σε όλων των ειδών τις μηχανές (υπολογιστές διαφορετικών κατασκευαστών) με σχετικά μεγάλη ευκολία, αφού το machine-dependent κομμάτι του ήταν περιορισμένο στο ελάχιστο δυνατό. Έτσι, όταν κάποιος ήθελε να μεταφέρει (port) το Unix σε μια μηχανή, αρκούσε να γράψει ένα μικρό κομμάτι του λειτουργικού συστήματος. Για το υπόλοιπο αρκούσε να έχει ένα C compiler για τη συσγεκριμένη μηχανή. (Είνια σημαντικό να ξεχωρίσουμε εδώ ότι ένας compiler για μια γλώσσα που τρέχει σε μαι μηχανή Α μπορεί κάλλιστα να παράγει κώδικα ο οποίος να είναι εκτελέσιμος σε μια μηχανή Β. Αυτοί λέγονται cross-compilers).

Από τη δεκαετία του 70 και πέρα η C έχει γίνει μια από τις πιο ευρέως διαδεδομένες γλώσσες, ξεφεύγοντας από τα στενά όρια του system programming για το οπίο είχε επινοηθεί. Ήταν μια γλώσσα θεμελιωδώς απλή, που παρείχε μεν τη δυνατότητα στον προγραμματιστή για το λεγόμενο δομημένο προγραμματισμό (structured programming), αλλά δεν του έδενε ταυτόχρονα τα χέρια (όπως κάνει π.χ μία γλώσσα όπως η Pascal, της οποίας οι compilers επιμένουν, κατά παράδοση, στην εφαρμογή ενός στυλ προγραμματισμού και δεν επιτρέπουν παρεκκλίσεις από αυτό).

Από τα μέσα της δεκαετίας του 1980 έχει κάνει την εφανισή της η C++, που είναι μια επέκταση της C προς την κατεύθυνση του object-oriented προγραμματισμού. Το μερίδιο της C++ στους προγραμματιστές αυξάνει διαρκώς, μίας και πρασφέρεται ιδιαίτερα για δουλειές, όπως την κατασκευή φιλικών user interfaces (δηλαδή περιβαλλόντων αλληλεπίδρασης με το χρήστη), και το object oriented programming

προσφέρει ένα πολύ καλό πρότυπο οργάνωσης μεγάλων προγραμμάτων που έχουν να κάνουν όχι τόσο με αριθμητικούς υπολογισμούς αλλά με διαχείρηση πολλών διαφορετικών τύπων δεδομένων πάνω στα οποία θέλουμε κάπως να κάνουμε εργασίες. Οι λόγοι της ραγδαίας ανάπτυξης της συγκεκριμένης γλώσσας προγραμματισμού είναι η ταχύτητά της, καθώς και το γεγονός ότι είναι διαθέσιμη στα περισσότερα σημερινά λειτουργικά συστήματα.

Στην επιστήμη υπολογιστών **δομημένος προγραμματισμός** (structured programming) ή **διαδικαστικός προγραμματισμός** (procedural programming), όπως αναφέρθηκε παραπάνω είναι μία προσέγγιση στον προγραμματισμό, η οποία βασίζεται στην έννοια της *κλήσης διαδικασίας*. Η διαδικασία, γνωστή επίσης και ως ρουτίνα, υπορουτίνα, μέθοδος ή συνάρτηση (δεν σχετίζεται άμεσα με τη μαθηματική έννοια της συνάρτησης), είναι απλά ένα αυτοτελές σύνολο εντολών προς εκτέλεση. Ο δομημένος προγραμματισμός βασίζεται στην αρχή του διαίρει και βασίλευε, καθώς διασπά το βασικό πρόβλημα σε μικρότερα υποπροβλήματα (γνωστά επίσης και ως εργασίες). Κάθε εργασία με πολύπλοκη περιγραφή διαιρείται σε μικρότερες, έως ότου οι εργασίες να είναι αρκετά μικρές, περιεκτικές και εύκολες προς κατανόηση.

Ιστορικά ο δομημένος προγραμματισμός αναπτύχθηκε ύστερα από έρευνα κατά τη δεκαετία του 1960, ως βελτίωση του ήδη υπάρχοντος διαδικαστικού προγραμματισμού. Ένα από τα πιο σημαντικά αποτελέσματα αυτής της έρευνας ήταν η ανάπτυξη της γλώσσας Pascal (γλώσσα προγραμματισμού), από τον Niklaus Wirth το 1971, η οποία σύντομα έγινε η προτιμώμενη γλώσσα διδασκαλίας σε πολλά πανεπιστήμια. Η έννοια της διαδικασίας επομένως ήταν προϋπάρχουσα αλλά δεν έπαιζε τόσο σημαντικό ρόλο στη δομή των υπό συγγραφή εφαρμογών, καθώς τα δεδομένα ήταν αρκετά διαχωρισμένα από τις διαδικασίες και έπρεπε ο προγραμματιστής να θυμάται για κάθε διαδικασία ποια άλλη καλούσε, αλλά και ποια δεδομένα διαφοροποιούνταν. Καθώς όμως οι περισσότερες διαδικαστικές γλώσσες γρήγορα υιοθέτησαν στοιχεία ώστε να υποστηρίζουν δομημένο προγραμματισμό, οι δύο όροι σήμερα έχουν πρακτικώς ταυτιστεί. Με τον καιρό οι δομημένες γλώσσες έφτασαν να μην επαρκούν για τη συγγραφή προγραμμάτων, επεκτάθηκαν και ως λύση υιοθετήθηκε ο αντικειμενοστρεφής προγραμματισμός.

4.2 Τι είναι ένα πρόγραμμα σε γλώσσα C

Η Γλώσσα Προγραμματισμού C είναι μία γενικού σκοπού γλώσσα προγραμματισμού, που υποστηρίζει πλήρως το προγραμματιστικό πρότυπο διαδικασιών (procedural programming paradigm). Με λίγα λόγια υποστηρίζει τον προγραμματισμό μέσω διαδικασιών (με τη μορφή συναρτήσεων) πλήρως. Επίσης είναι γλώσσα μέσου και χαμηλού επιπέδου. Το τελευταίο σημαίνει ότι βρίσκεται κοντά στο υλικό (hardware). Δεν υπάρχει χώρος για άλλη γλώσσα προγραμματισμού χαμηλότερου επιπέδου από την C (και την C++), εκτός από τη συμβολική γλώσσα (assembly language).

Ένα πρόγραμμα C αποτελείται από οδηγίες προς τον preprocessor (preprocessor directives), από δηλώσεις δεδομένων (data declarations) και από συναρτήσεις (functions). Οι συναρτήσεις δεν πρέπει να συγχέονται με τις συναρτήσεις που συναντάμε στα μαθηματικά. Στην πραγματικότητα πρόκειται για κομμάτια κώδικα, δηλαδή για ακολουθίες εντολών της C και πιθανώς, κάποιες δηλώσεις δεδομένων, τα οποία έχουν όνομα, μπορούν να δέχονται παραμέτρους και να επιστρέφουν αποτελέσματα. Τα ονόματα των συναρτήσεων θα πρέπει να είναι διαφορετικά μεταξύ τους, έτσι ώστε να είναι σαφές σε ποιά συνάρτηση αναφερόμαστε όταν χρησιμοποιούμε κάποιο όνομα main και αντιπροσωπεύει το σημείο από το οποίο ξεκινάει και τελειώνει η εκτέλεση του προγραμματός μας. Με άλλα λόγια, η πρώτη εντολή στη συνάρτηση main είναι η πρώτη εντολή του προγράμματος που θα εκτελεστεί, ενώ με την εκτέλεση της τελευταίας εντολής της συνάρτησης main τελειώνει και η εκτέλεση του προγράμματος. Σε ένα πρόγραμμα C μπορούμε να έχουμε και σχόλια:Η αρχή των σχολίων δηλώνεται με ΄΄/*΄΄ και το τέλος τους με ΄΄/*΄΄. Οτιδήποτε βρίσκεται ανάμεσα στην αρχή και το τέλος των σχολίων αγνοείται.

Η C χρησιμοποιείται για την δημιουργία Λειτουργικών Συστημάτων, προγραμμάτων που τρέχουν απευθείας στο hardware, αλλά και συνηθισμένων προγραμμάτων χρηστών. Ένα πρόγραμμα που είναι γραμμένο σύμφωνα με το επίσημο Πρότυπο, είναι φορητό (δηλαδή γίνεται compiled σε μηχανήματα διαφορετικών αρχιτεκτονικών με compilers που υποστηρίζουν το επίσημο Πρότυπο), χωρίς ή με ελάχιστες αλλαγές. Για παράδειγμα, η χρήση σταθερών χαρακτήρων (π.χ. 'A', '\n', κ.λ.π.) αντί συγκεκριμένων ακέραιων τιμών που αντιστοιχούν σε συγκεκριμένο κώδικα χαρακτήρων ενός συγκεκριμένου συστήματος (π.χ. ASCII),

κάνει ένα πρόγραμμα να δουλεύει χωρίς αλλαγές σε σύστημα με διαφορετικό κώδικα χαρακτήρων (π.χ. EBCDIC).

Άλλα παραδείγματα φορητότητας αποτελούν τα μεγέθη σε Bytes και το εύρος τιμών των ενσωματωμένων τύπων, που μπορούν να διαφέρουν από μηχάνημα σε μηχάνημα (εκτός του μεγέθους σε Bytes των τύπων char, signed char και unsigned char, που καταλαμβάνουν πάντοτε 1Byte). Για παράδειγμα, τα εύρη τιμών των ενσωματωμένων τύπων περιέχονται στις τυπικές κεφαλίδες *limits.h* και *float.h*. Έτσι, η μέγιστη και η ελάχιστη τιμή του ενσωματωμένου τύπου int σε ένα σύστημα δίνονται από τις σταθερές INT_MAX και INT_MIN της τυπικής κεφαλίδας *limits.h*. Ο αριθμός των bits ενός Byte σε οποιοδήποτε σύστημα, δίνεται από την σταθερά CHAR_BIT της *limits.h* (υπάρχουν και συστήματα όπου ένα Byte περιέχει περισσότερα από 8 bits).

Τέλος, από άποψη υλικού (hardware), η Γλώσσα Προγραμματισμού C χρησιμοποιείται για τον προγραμματισμό μικρών εμφυτευμένων συσκευών (π.χ. BIOSes γραμμένα σε C, αυτόματα ποτιστικά συστήματα, συστήματα συναγερμού) μέχρι τεράστια Mainframes.

4.3 Πλεονεκτήματα της Γλώσσας Προγραμματισμού C

Τα τελευταία χρόνια η C έχει καθιερωθεί ως μια από τις σημαντικότερες και δημοφιλέστερες γλώσσες προγραμματισμού. Τα σημαντικότερα πλεονεκτήματα που εξηγούν αυτήν την προτίμησή της, αναφέρονται παρακάτω.

- *Χαρακτηριστικά Σχεδίασης*, Η C έχει μοντέρνες δομές ελέγχου για να μπορούμε να κάνουμε επαναληπτικές εργασίες και για εύκολη επιλογή εναλλακτικών τρόπων δράσης. Με το πλήθος των δομών δεδομένων που διαθέτει, μπορεί να αναπαραστήσει ένα μεγάλο σύνολο από διαφορετικούς τύπους πληροφοριών. Έχει και το μεγάλο πλεονέκτημα ότι επιβάλλει τη διάσπαση του προγράμματος σε αυτοδύναμες ενότητες, τις συναρτήσεις.
- *Αποτελεσματική*, Η C είναι μια αποτελεσματική γλώσσα προγραμματισμού, που είναι τόσο συμπεριεκτική, ώστε να χρησιμοποιούμε σ' αυτήν πολύ λιγότερες λέξεις σε σχέση με άλλες γλώσσες. Έχει έναν συμπαγή και γρήγορο κώδικα.

- *Φορητή Γλώσσα*, Η C είναι μια φορητή γλώσσα, δηλ. τα προγράμματά της μπορούν να τρέξουν με λίγες ή και με καθόλου τροποποιήσεις και σε ένα άλλο σύστημα. Μάλιστα θεωρείται σαν η πιο φορητή γλώσσα.
- *Δυναμικότητα και Ευελιξία*, Η C είναι δυναμική και ευέλικτη, δύο ιδιότητες που είναι αρκετά δημοφιλείς στους υπολογιστές. Όπως ξέρουμε, το μεγαλύτερο μέρος του δυναμικού και ευέλικτου λειτουργικού συστήματος Unix είναι γραμμένο σε C. Αυτό ισχύει και για επεξεργαστές κειμένων, μεταγλωττιστές (compilers) και ερμηνευτές (interpreters) γλωσσών προγραμματισμού. Η C διαθέτει μερικά από τα χαρακτηριστικά ελέγχου που συνήθως τα συναντάμε στη συμβολική γλώσσα (assembly language).
- *Προσανατολισμός προς τον Προγραμματιστή*, Η C είναι προσανατολισμένη προς τις ανάγκες του προγραμματιστή, ο οποίος και έχει άμεση πρόσβαση στο υλικό. Με τη C έχουμε τη σπουδαία δυνατότητα να μπορούμε να χειριζόμαστε μεμονωμένα τα *δυαδικά ψηφία* ***(bits)*** της μνήμης. Γενικά η C είναι πολύ λιγότερο περιοριστική στο να μας αφήνει να κάνουμε ό,τι θέλουμε σε σχέση με την Pascal για παράδειγμα.

Αυτή η ελευθερία είναι και πλεονέκτημα, αλλά είναι και επικίνδυνη όπως είναι φυσικό. Στη C τα πάντα (σχεδόν) επιτρέπονται. Δεν γίνεται έλεγχος των τύπων, άρα μπορεί κανείς να ανακατέψει ό,τι δεδομένα θέλει, κάτι που είναι πολύ χρήσιμο όταν προγραμματίζουμε σε επίπεδο συστήματος. Ακόμη, η C έχει μια τεράστια βιβλιοθήκη από χρήσιμες συναρτήσεις.

4.4 Τα Μειονεκτήματα της C

Η C έχει και μειονεκτήματα, γιατί όπως πολύ καλά ξέρουμε η πολύ ελευθερία βλάπτει. Για παράδειγμα, η *ελευθερία έκφρασης* που αναφέραμε παραπάνω ότι έχει η C, απαιτεί από τον προγραμματιστή μια αυξημένη επαγρύπνηση και υπευθυνότητα. Ακόμη, η λακωνικότητα της C σε συνδυασμό με τον πλούτο των τελεστών που έχει, έχει σαν αποτέλεσμα τη δημιουργία προγραμμάτων που είναι τόσο *δυσανάγνωστα*, ώστε να είναι δύσκολο να τα κατανοήσει κάποιος με την πρώτη ματιά και πολλές φορές ακόμα και αυτός που τα έγραψε. Επιπλέον, συχνά είναι πολύ δύσκολο να ανιχνευθούν και τα λογικά λάθη σ' ένα πρόγραμμα της C. Η C έχει τελικά τόσες

πολλές δυνατότητες έκφρασης, ώστε να χρειαστεί πολύς καιρός για να μπορεί να πει κανείς με βεβαιότητα ότι την έμαθε καλά.

4.5 Δομή - Γράψιμο ενός προγράμματος στη γλώσσα C

Όπως είπαμε στα προηγούμενα, η C επιβάλλει τον καταμερισμό του προγράμματος σε ενότητες, που ονομάζονται *συναρτήσεις (functions)*. Εάν είναι απαραίτητο, οι συναρτήσεις μπορούν να χωριστούν και σε μικρότερες συναρτήσεις. Επίσης, στη C το κύριο πρόγραμμα είναι κι αυτό μια συνάρτηση, που ονομάζεται ***main()***. Μια μέθοδος για το γράψιμο ενός προγράμματος στη C είναι να ξεκινήσουμε γράφοντας τη συνάρτηση main(), την ενότητα του πιο πάνω επιπέδου και μετά να ασχοληθούμε με τις συναρτήσεις των πιο κάτω επιπέδων. Η διαδικασία αυτή ονομάζεται *πάνω-προς-τα-κάτω προγραμματισμός (top-down programming)*.

Η αντίστροφη διαδικασία, δηλαδή το να ασχοληθούμε πρώτα με τις συναρτήσεις των κατώτερων επιπέδων και μετά να ανεβαίνουμε προς τα πάνω, ονομάζεται *κάτω-προς-τα-πάνω προγραμματισμός (bottom-up programming)*. Ένα πλεονέκτημα του πάνω-προς-τα-κάτω προγραμματισμού είναι ότι μπορούμε να χαράξουμε καλύτερα τη ροή του προγράμματος, μια και δεν ασχολούμαστε από την αρχή με τις λεπτομέρειες των επί μέρους συναρτήσεων.

Για τη μεταγλώττιση και σύνδεση ενός προγράμματος στη γλώσσα C, ακολουθείται η εξής διαδικασία: ο *μεταγλωττιστής (compiler)* της C μετατρέπει τον *πηγαίο κώδικα (source program)*, δηλαδή το πρόγραμμα που γράφουμε σε C, σ' έναν *αντικειμενικό κώδικα (object program)* και το *πρόγραμμα σύνδεσης (linker)* συνδυάζει αυτόν τον κώδικα με άλλους κώδικες και δημιουργείται έτσι το *εκτελέσιμο αρχείο (executable file)*. Τα προγράμματα της C έχουν την επέκταση ***.c***.

Ο ρόλος του προγράμματος σύνδεσης είναι να ενώσει τον τελικό κώδικα, τον *κώδικα εκκίνησης (start-up code)* του συστήματός μας και τον *κώδικα βιβλιοθήκης (library code)* στο εκτελέσιμο αρχείο. Ο κώδικας εκκίνησης έχει σχέση με την επικοινωνία μεταξύ του προγράμματος και του λειτουργικού συστήματος και ο κώδικας βιβλιοθήκης περιέχει τον τελικό κώδικα για πολλές συναρτήσεις. Σε μερικά συστήματα πρέπει να τρέξουμε τα προγράμματα μεταγλώττισης και σύνδεσης

ξεχωριστά, ενώ σε άλλα ο μεταγλωττιστής ενεργοποιεί το πρόγραμμα σύνδεσης αυτόματα μόνος του.

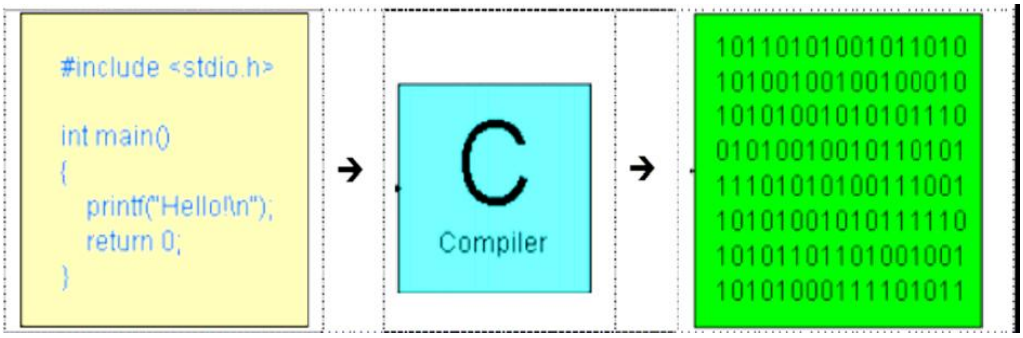
Η γλώσσα προγραμματισμού "C" είναι σχεδιασμένη για να τρέχει κάτω από όλα τα λειτουργικά συστήματα, και έτσι δεν βασίζεται σε κάποιες ειδικές ευκολίες που μπορεί να παρέχει το λειτουργικό σύστημα. Για να γράψουμε έτσι ένα πρόγραμμα C απαιτούνται:

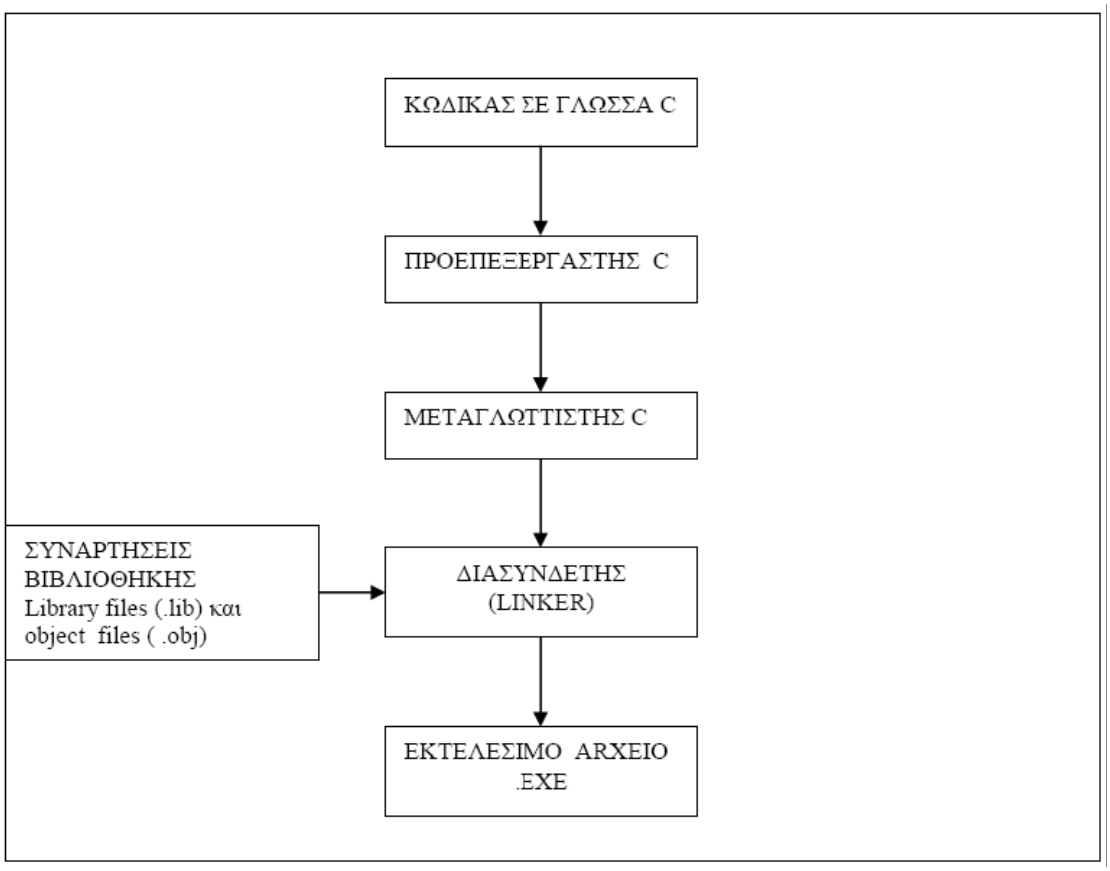
1) ένας text editor στον οποίο θα γράψουμε το πρόγραμμά μας. Τέτοιοι editors υπάρχουν σε όλα τα λειτουργικά συστήματα,

2) ο μεταγλωττιστής της γλώσσας C που θα αναλάβει να μεταφράσει το πρόγραμμά μας σε κώδικα μηχανής. Ο μεταγλωττιστής της C αποτελείται από τον C preprocessor και τον C-compiler. Στα επόμενα με τον όρο "compiler" θα αναφερόμαστε και στα δύο αυτά τμήματα, εκτός αν καθορίζεται διαφορετικά. Ο C-preprocessor επεξεργάζεται τα αρχεία που περιέχουν κώδικα C, με τρόπο που θα αναλυθεί σε επόμενο κεφάλαιο, και παράγει ενδιάμεσα αρχεία, από τα οποία ο C-compiler παράγει αρχεία με κώδικα μηχανής (.obj) αρχεία και

3) ο linker, που είναι κομμάτι του λειτουργικού συστήματος, αν και κάποια υλοποίηση ενός C-compiler μπορεί να παρέχει τον δικό της. Ο linker αναλαμβάνει να συνδέσει τον κώδικα μηχανής του προγράμματος μας (αυτόν που βρίσκεται στα . obj αρχεία) με κώδικα που υπάρχει σε βιβλιοθήκες (αρχεία .lib) και άλλα αρχεία με κώδικα μηχανής (.obj), παράγοντας τελικά ένα εκτελέσιμο αρχείο. Ο κώδικας που βρίσκεται στις βιβλιοθήκες συμπεριλαμβάνει κώδικα που υλοποιεί υψηλού επιπέδου λειτουργίες μέσα από διαδικασίες πιο χαμηλού επιπέδου, απαλλάσσοντας μας έτσι από την υποχρέωση να κάνουμε εμείς την υλοποίηση αυτή.

Στα περισσότερα λειτουργικά συστήματα δεν είναι απαραίτητο να ασχοληθούμε με τη διαδικασία αυτή μια και υπάρχει κάποιο πρόγραμμα που την αναλαμβάνει . Σε μερικά μάλιστα συστήματα και ο text editor παρέχεται σε ένα ολοκληρωμένο περιβάλλον, στο οποίο μπορούμε να γράψουμε τα προγράμματά μας, να τα μεταφράσουμε και να τα συνδέσουμε παράγοντας έτσι το εκτελέσιμο αρχείο χωρίς να χρειαστεί να ασχοληθούμε καθόλου με την ενδιάμεση διαδικασία. Η όλη διαδικασία παραγωγής ενός εκτελέσιμου αρχείου από τα c, .obj και .lib αρχεία φαίνεται στο πιο κάτω σχήμα:





4.5 Ένα Απλό Πρόγραμμα στη C

/ prog01.c – αυτό είναι ένα απλό πρόγραμμα στη γλώσσα C */*

#include <stdio.h>

main()

{

```
    int num;        /* ορίζεται μια ακέραια μεταβλητή με το όνομα num */

    num = 10 ;   /* καταχώρηση τιμής στη μεταβλητή num */

    printf("Ένα πολύ απλό πρόγραμμα σε C.\n"); /* η συνάρτηση printf() */

    printf("Ο αγαπημένος μου αριθμός είναι ο %d.\n", num);

}
```

Αφού μεταγλωττίσουμε και τρέξουμε το παραπάνω πρόγραμμα, τότε θα πρέπει να εμφανισθούν στην οθόνη τα παρακάτω :

Ένα πολύ απλό πρόγραμμα σε C.

Ο αγαπημένος μου αριθμός είναι ο 10.

ν Σύντομη Ανάλυση του Προγράμματος

Η πρώτη γραμμή του προγράμματος χρησιμοποιεί τα σύμβολα /* και */ για να συμπεριλάβουμε εκεί κάποια *σχόλια (comments),* που θα μας βοηθήσουν να κάνουμε το πρόγραμμά μας πιο ευανάγνωστο. Αυτά τα σχόλια αγνοούνται από τον υπολογιστή. Η δεύτερη γραμμή λέει στον υπολογιστή να συμπεριλάβει (include) τις πληροφορίες που υπάρχουν στο αρχείο stdio.h, το οποίο αποτελεί μέρος του πακέτου της γλώσσας C.

Τα προγράμματα της C αποτελούνται από μία ή περισσότερες συναρτήσεις, που είναι και οι βασικές ενότητες ενός προγράμματος C. Αυτό το πρόγραμμα αποτελείται από μία μόνο συνάρτηση που καλείται *main()*. Οι παρενθέσεις υποδηλώνουν ότι το main() είναι ένα όνομα μιας συνάρτησης.

Η αγκύλη { δηλώνει την αρχή των προτάσεων που αποτελούν τη συνάρτηση και ο ορισμός της συνάρτησης τελειώνει με την αντίστοιχη αγκύλη }. Οι αγκύλες { και } είναι αντίστοιχες με τα begin και end της Pascal. Η πρόταση δήλωσης μάς λέει ότι θα χρησιμοποιήσουμε μια μεταβλητή με το όνομα num και ότι η num είναι ακέραια μεταβλητή (integer). Η πρόταση καταχώρησης δίνει την τιμή 10 στη μεταβλητή num.

Η επόμενη γραμμή είναι για την εκτύπωση της φράσης που βρίσκεται μεταξύ των εισαγωγικών (" "). Το *\n* λέει στον υπολογιστή να ξεκινήσει μια νέα γραμμή (σαν να πατούσαμε το πλήκτρο <enter>). Η επόμενη γραμμή είναι για την εκτύπωση της τιμής της num (που είναι το 10) ανάμεσα στη φράση που βρίσκεται μεταξύ των " ". Η εντολή %d λέει στον υπολογιστή πού και με ποια μορφή να εκτυπώσει την τιμή της num. Το πρόγραμμα τελειώνει με την αγκύλη }.

v Λεπτομερής Ανάλυση του Προγράμματος

1. Τα Αρχεία Επικεφαλίδας #include

Το αρχείο stdio.h αποτελεί μέρος του μεταγλωττιστή της C και περιέχει πληροφορίες σχετικές με συναρτήσεις εισόδου και εξόδου, όπως είναι η ***printf()***, που χρησιμοποιεί ο μεταγλωττιστής. Το όνομά του προέρχεται από τις λέξεις ***standard input/output header***. Οι χρήστες της C αναφέρουν σαν επικεφαλίδα μια συλλογή πληροφοριών που βρίσκεται στην αρχή ενός αρχείου. Το αποτέλεσμα της εντολής ***#include <stdio.h>*** θα ήταν το ίδιο με το να αντιγράφαμε όλο το περιεχόμενο του αρχείου stdio.h στο δικό μας αρχείο, στη θέση όπου εμφανίζεται αυτή η γραμμή προγράμματος.

Στην πραγματικότητα, αυτή η γραμμή προγράμματος δεν είναι καν μια πρόταση της γλώσσας C. Το σύμβολο # σημαίνει ότι τη γραμμή αυτή τη διαχειρίζεται ο *προεπεξεργαστής* ***(preprocessor)*** της C, ο οποίος διαχειρίζεται κάποιες εργασίες πριν από τον μεταγλωττιστή.

2. Η Συνάρτηση main()

Η εκτέλεση ενός προγράμματος σε C αρχίζει πάντα με μια συνάρτηση που αποκαλείται main(). Είμαστε ελεύθεροι να επιλέξουμε τα ονόματα των άλλων συναρτήσεων που ίσως χρησιμοποιήσουμε, αλλά θα πρέπει υποχρεωτικά να υπάρχει η συνάρτηση main() στην αρχή του προγράμματος. Οι παρενθέσεις δηλώνουν ότι η main() είναι μια συνάρτηση. Γενικά, οι παρενθέσεις περιέχουν πληροφορίες (ορίσματα) που θα περάσουν μέσα στη συνάρτηση. Όταν, βέβαια, δεν υπάρχουν πληροφορίες για να περάσουν (μεταβιβασθούν), τότε οι παρενθέσεις είναι άδειες.

3. Τα Σχόλια

Όταν στο πρόγραμμά μας έχουμε σχόλια, τότε είναι πολύ ευκολότερο σε κάποιον άλλον, αλλά ακόμα και σε μας, να καταλάβει τι κάνει το πρόγραμμά μας. Ο,τιδήποτε υπάρχει ανάμεσα στο σύμβολο ανοίγματος /* και στο σύμβολο κλεισίματος */ των σχολίων αγνοείται από τον υπολογιστή και τα σχόλια στη C μπορούν να τοποθετηθούν οπουδήποτε, ακόμα και σε πολλές συνεχόμενες γραμμές.

4. Αγκύλες, Σώματα, Μπλοκ

Οι αγκύλες { και } δηλώνουν την αρχή και το τέλος του σώματος μιας συνάρτησης και μπορούν ακόμη να χρησιμοποιηθούν για να συμπεριλάβουν μαζί προτάσεις μέσα σε μια ομάδα ή σ' ένα μπλοκ του προγράμματος, κάτι δηλαδή παρόμοιο με τα begin και end της Pascal.

5. Οι Προτάσεις Δήλωσης

Η πρόταση δήλωσης είναι μια από τις σημαντικότερες προτάσεις της C. Η δήλωση αυτή σημαίνει ότι κάπου μέσα στη συνάρτηση χρησιμοποιούμε τη μεταβλητή που δηλώνουμε και ότι ο τύπος της είναι αυτός που δείχνουμε, π.χ. ακέραιος. Η λέξη ***int*** είναι μια λέξη-κλειδί της C, δηλ. δεν μπορεί να χρησιμοποιηθεί αλλού μέσα στο πρόγραμμα σαν όνομα μιας συνάρτησης ή μιας μεταβλητής. Το ερωτηματικό στο τέλος της γραμμής δηλώνει ότι η γραμμή αυτή αποτελεί μια πρόταση ή εντολή της C.

4.6 Οι Τύποι Δεδομένων και οι Μεταβλητές

Η C έχει διάφορα είδη (τύπους) δεδομένων : ακέραιους, χαρακτήρες, κινητής υποδιαστολής, αριθμούς κ.ά. Για το όνομα μιας μεταβλητής μπορούμε να χρησιμοποιήσουμε μικρά γράμματα, κεφαλαία γράμματα, ψηφία αριθμών και τον χαρακτήρα της υπογράμμισης (_). Ο πρώτος χαρακτήρας, όμως, πρέπει να είναι πάντα γράμμα. Πρέπει να είναι όλες οι μεταβλητές του προγράμματος συγκεντρωμένες μαζί για να είναι ευκολότερο για τον αναγνώστη να καταλάβει τι κάνει το πρόγραμμα και ακόμη πρέπει να υπάρχουν και σχόλια δίπλα στην κάθε μεταβλητή που να εξηγούν την αποστολή της. Ακόμη, η ονομασία που δίνουμε στις μεταβλητές πρέπει να μας βοηθάει να καταλαβαίνουμε και τη χρήση τους.

4.7 Δεδομένα, σταθερές και μεταβλητές

4.7.1 Δεδομένα

Τα δεδομένα (πληροφορίες-data) είναι απαραίτητα στοιχεία ενός προγράμματος , καθώς οι βασικές λειτουργίες ενός προγράμματος είναι η επεξεργασία αυτών των δεδομένων και η εξαγωγή αποτελεσμάτων (δηλαδή άλλα δεδομένα). Απαιτείται λοιπόν η δέσμευση κάποιων χώρων μνήμης για να αποθηκευτούν αυτά τα δεδομένα κατά τη διάρκεια της εκτέλεσης του προγράμματος. Αυτοί οι χώροι μνήμης που τους χρησιμοποιούμε για τη φύλαξη δεδομένων, όταν εκτελείται ένα πρόγραμμα, ονομάζονται ανάλογα με τη χρήση τους σταθερές ή μεταβλητές και κάθε γλώσσα προγραμματισμού μας δίνει τη δυνατότητα με διάφορους τρόπους να δηλώσουμε (καθορίσουμε) κάποιους τύπους μεταβλητών / σταθερών, τις οποίες θα χρησιμοποιήσουμε για την αποθήκευση δεδομένων, όταν εκτελείται ένα πρόγραμμα.

4.7.2 Οι σταθερές

Αρκετά προγράμματα απαιτούν ορισμένα δεδομένα που δεν αλλάζουν ποτέ κατά τη διάρκεια της εκτέλεσής τους. Αυτά τα δεδομένα συνήθως καθορίζονται μια φορά και χρησιμοποιούνται όσο συχνά επιθυμούμε κατά την διάρκεια λειτουργίας του προγράμματος. Για παράδειγμα, όταν γράφετε ένα πρόγραμμα, μπορείτε να καθορίσετε ότι η τιμή του π είναι 3.14159 και να χρησιμοποιείτε αυτή την τιμή όταν την χρειάζεστε, ξέροντας ότι είναι διαθέσιμη και σωστή. Οι σταθερές ορίζονται με εντολές του προεπεξεργαστή με την οδηγία # define και συνήθως χρησιμοποιούμε κεφαλαία γράμματα για να τις περιγράψουμε: # define PI 3.14159 (σημειώνουμε ότι αυτή η εντολή δεν τελειώνει με το ερωτηματικό ";")

4.7.3 Οι μεταβλητές

Αντίθετα με τις σταθερές, οι τιμές των μεταβλητών είναι δυνατόν να αλλάξουν κατά τη διάρκεια της εκτέλεσης ενός προγράμματος. Οι γλώσσες προγραμματισμού μας δίνουν τη δυνατότητα να καθορίσουμε μεταβλητές και στη συνέχεια να τους δώσουμε όποια τιμή επιθυμούμε, που έχει βέβαια σχέση με το πρόγραμμα. Η δήλωση των μεταβλητών γίνεται συνήθως στην αρχή του προγράμματος.

4.8 Τεχνικές για πιο Ευανάγνωστα Προγράμματα

Δύο τέτοιες τεχνικές που είδαμε στα προηγούμενα ήταν η χρήση σχολίων και η σωστή ονομασία των μεταβλητών. Μια άλλη τεχνική είναι η χρήση κενών γραμμών για να ξεχωρίζουν τα τμήματα του προγράμματος. Ακόμη, στη C μπορούμε να τοποθετήσουμε πολλές προτάσεις σε μια γραμμή ή και να χωρίσουμε μια πρόταση σε πολλές γραμμές. Το ελληνικό ερωτηματικό λέει στον μεταγλωττιστή πού τελειώνει μια πρόταση και πού αρχίζει η επόμενη. Πιο σωστό είναι πάντως να γράφουμε μια πρόταση ανά γραμμή.

4.9 Ένα Ακόμη Παράδειγμα στη C

```
/* prog02.c – το πρόγραμμα αυτό μετατρέπει τα μέτρα σε εκατοστά */

#include <stdio.h>

main()

{

        int cm, metres;



        metres = 2;

        cm = 100 * metres;

        printf("Υπάρχουν %d εκατοστά σε %d μέτρα. \n", cm, metres);



}
```

Όπως βλέπουμε, το πρώτο σχόλιο του προγράμματος περιέχει το όνομά του και το τι ακριβώς κάνει. Ακόμη, το πρόγραμμα δηλώνει δύο ακέραιες μεταβλητές μαζί, τις οποίες και διαχωρίζει με κόμμα. Χρησιμοποιεί τον τελεστή του πολλαπλασιασμού,

που είναι το * και εκτυπώνει πολλές μεταβλητές μαζί στη συνάρτηση printf(). Όταν τρέξει το πρόγραμμα, θα δώσει το εξής αποτέλεσμα :

Υπάρχουν 200 εκατοστά σε 2 μέτρα.

Κλήση Συνάρτησης

Στο επόμενο παράδειγμα θα δούμε πώς μπορούμε να συμπεριλάβουμε και να καλούμε και μια δική μας συνάρτηση.

```
/* prog03.c – ένα πρόγραμμα που καλεί μια συνάρτηση */

#include <stdio.h>

main()

{

        printf("Θα καλέσω τη συνάρτηση της Κεφαλονιάς.\n");

        kefalonia();

        printf("Η κλήση της συνάρτησης έγινε.\n");

}



kefalonia()

{

        printf("Γεια σας από την Κεφαλονιά.\n");

}
```

Το αποτέλεσμα του προγράμματος θα είναι το παρακάτω :

Θα καλέσω τη συνάρτηση της Κεφαλονιάς.

Γεια σας από την Κεφαλονιά.

Η κλήση της συνάρτησης έγινε.

Η συνάρτηση kefalonia) ορίζεται με τον ίδιο τρόπο, όπως και η main(), με το σώμα της να βρίσκεται ανάμεσα σε αγκύλες. Η συνάρτηση καλείται δίνοντας απλά το όνομά της μαζί με τις παρενθέσεις. Όταν η συνάρτηση kefalonia() τελειώσει τη δουλειά της, το πρόγραμμα προχωράει στην επόμενη πρόταση της main().

Πρέπει να έχουμε υπόψη μας ότι οι συναρτήσεις μπορούν να γραφούν είτε πριν ή μετά από την κύρια συνάρτηση main(), αλλά εκτελούνται μόνο όταν και όπου η main() τις καλεί.

4.10 Έλεγχος της Ορθότητας των Προγραμμάτων

1. Συντακτικά Λάθη

Το συντακτικό λάθος στη C είναι κάτι ανάλογο με το γραμματικό λάθος στη γλώσσα που μιλάμε. Συντακτικά λάθη στη C μπορούν να γίνουν και με τη χρήση επιτρεπτών συμβόλων σε λανθασμένες θέσεις. Παραδείγματα τέτοιων λαθών μπορεί να είναι η μη σωστή χρήση των αγκυλών { και } ή ακόμη το να ανοίγουμε μια αγκύλη και να μην την κλείνουμε, το να ανοίγουμε κάπου σχόλια και να ξεχνάμε να τα κλείσουμε και τα λοιπά. Όπως ξέρουμε, μέρος της δουλειάς του μεταγλωττιστή είναι και η ανακάλυψη των συντακτικών λαθών του προγράμματος. Υπάρχουν, όμως, και περιπτώσεις όπου ένα λάθος παράγει, άθελά μας, και άλλα λάθη.

2. Εννοιολογικά Λάθη

Το εννοιολογικό λάθος είναι το λάθος στο νόημα των προτάσεων. Στη C εννοιολογικά λάθη μπορούμε να κάνουμε, όταν ακολουθούμε μεν σωστά τους κανόνες της γλώσσας, αλλά με λανθασμένο αποτέλεσμα. Τέτοιο λάθος μπορεί να γίνει, όταν π.χ. αντί να προσθέσουμε δύο μεταβλητές, τις πολλαπλασιάζουμε. Με τα λάθη αυτά βέβαια δεν έχει καμία σχέση ο μεταγλωττιστής. Είναι δική μας δουλειά να

τα ανακαλύψουμε και να τα διορθώσουμε. Ο καλύτερος τρόπος για να ανακαλύψουμε τέτοια λάθη είναι να εξετάσουμε το πρόγραμμα βήμα-βήμα.

Μπορούμε ακόμα να χρησιμοποιούμε επιλεκτικά και τη συνάρτηση printf() μέσα στο πρόγραμμα, ώστε να ελέγχουμε τις τιμές κάποιων μεταβλητών του προγράμματος. Τις εντολές printf() τις απομακρύνουμε μετά όταν το πρόγραμμά μας λειτουργήσει κανονικά. Και η χρήση των σχολίων μπορεί να αποδειχθεί χρήσιμη εδώ, γιατί με τη βοήθειά τους μπορούμε να απομονώσουμε κάποιο κομμάτι του προγράμματος προσωρινά και να ελέγξουμε έτσι την ορθότητα του υπόλοιπου προγράμματος. Υπάρχουν και ειδικά προγράμματα που λέγονται *αποσφαλματωτές* ***(debuggers)*** και που μας επιτρέπουν να βλέπουμε τις τιμές των μεταβλητών του προγράμματος και ποια γραμμή του προγράμματος εκτελείται.

4.11 Οι Λέξεις-Κλειδιά της ANSI C

Οι *λέξεις-κλειδιά* ή *δεσμευμένες λέξεις* ***(reserved words)*** αποτελούν το λεξιλόγιο της C και γι' αυτόν τον λόγο δεν μπορούμε να τις χρησιμοποιούμε για να δηλώσουμε ονόματα μεταβλητών ή συναρτήσεων στα προγράμματά μας.

Οι λέξεις αυτές είναι οι εξής :

auto	break	case	char	const
continue	default	do	double	else
enum	Extern	float	for	goto
If	int	long	register	return
short	signed	sizeof	static	struct
switch	typedef	union	unsigned	void
volatile	While			

4.12 Οι Τύποι Δεδομένων

Ο τύπος μιας μεταβλητής μπορεί να καθοριστεί σε μια πρόταση δήλωσης. Η C χρησιμοποιεί τις εξής πέντε λέξεις-κλειδιά για τους τύπους δεδομένων της :

Int (integar), char (character), float (floating point), void (no value) και double (double floating point). Όλοι οι άλλοι τύποι βασίζονται σε αυτούς. Η int δηλώνει τον βασικό τύπο ακεραίων και η char χρησιμοποιείται γενικά για τους χαρακτήρες, ενώ οι float και double χρησιμοποιούνται για την παράσταση αριθμών κινητής υποδιαστολής. Όλοι οι βασικοί τύποι εκτός από τον τύπο void, μπορούν να αλλάξουν, γράφοντας πριν από τον τύπο τον κατάλληλο μετασχηματισμό. Οι μετασχηματισμοί αυτοί είναι οι: signed, unsigned, long και short. Το μέγεθος και τα διαστήματα τιμών των τύπων της C εξαρτώνται από τον επεξεργαστή. Στον πίνακα που ακολουθεί δίνουμε τους τύπους δεδομένων, όπως ορίζονται από το πρότυπο ANSI.

Τύπος	Μέγεθος σε bits	Διάστημα τιμών
Char	8	-128...127
Unsigned char	8	0...255
Signed char	8	-127...127
Int	16	-32767...32767
Unsigned Int	16	0...65535
Signed Int	16	-32767...32767
Short Int	16	-32767...32767
Unsigned Short Int	8	0...65535
Signed short Int	8	-32767...32767
Long Int	32	-2147483647...2147483647
Signed Long Int	32	0...424967295

4.12.1 Οι Ακέραιοι Τύποι Δεδομένων

Οι ακέραιοι τύποι δεδομένων αποθηκεύονται σε μορφή ψηφιακών αριθμών. Ο τύπος int που ήδη ξέρουμε είναι ένας *ακέραιος με πρόσημο*, δηλ. πρέπει να είναι ένας ολόκληρος ακέραιος και μπορεί να είναι θετικός, αρνητικός ή και μηδέν. Συνήθως χρησιμοποιούνται δύο ψηφιολέξεις (bytes) για να αποθηκευθεί ένας ακέραιος της μορφής αυτής και έτσι το εύρος τιμών του είναι από -32768 έως και +32767. Η λέξη-

κλειδί int χρησιμοποιείται για τη δήλωση μεταβλητών αυτού του τύπου. Για να δηλώσουμε περισσότερες μεταβλητές αυτού του τύπου, μπορούμε είτε να τις δηλώσουμε χωριστά, ή να τις δηλώσουμε όλες μαζί, αλλά χωρισμένες με κόμμα.

Οι δηλώσεις δημιουργούν μεταβλητές, αλλά δεν καταχωρούν τιμές σ' αυτές. Η καταχώρηση των τιμών γίνεται μέσα στο πρόγραμμα με τις εντολές καταχώρησης ή και μέσω της συνάρτησης scanf(). *Απόδοση αρχικής τιμής* σε μια μεταβλητή σημαίνει να ορίσουμε την τιμή εκκίνησης της μεταβλητής. Στη C αυτό μπορεί να γίνει σε μια πρόταση δήλωσης, όπου μετά το όνομα της μεταβλητής υπάρχει ο τελεστής καταχώρησης (=) και μετά η αρχική τιμή της μεταβλητής, ως εξής :

```
int a=10;

int b, cs=20;
```

Στη C ειδικά προθέματα δηλώνουν ποια βάση αρίθμησης χρησιμοποιούμε. Το πρόθεμα *0 (μηδέν)* σημαίνει ότι γράφουμε στο *οκταδικό* και το πρόθεμα *0x* ή *0X* σημαίνει ότι ο αριθμός είναι *δεκαεξαδικός*. Έτσι, ο δεκαδικός αριθμός 16 γράφεται στο οκταδικό σύστημα σαν 020 και στο δεκαεξαδικό σύστημα σαν 0x10 ή 0X10. Σ' όλες τις περιπτώσεις χρησιμοποιείται ο δυαδικός κώδικας από τον υπολογιστή για την αποθήκευση του αριθμού.

Το σύμβολο %d που είδαμε στα προηγούμενα παραδείγματα αντιστοιχεί σε μια ακέραια τιμή αριθμού, που μπορεί να είναι μια μεταβλητή τύπου int, μια ακέραια σταθερά τύπου int ή και οποιαδήποτε άλλη έκφραση, που έχει όμως μια ακέραια τιμή int. Ακολουθεί ένα απλό πρόγραμμα που επεξηγεί τα παραπάνω :

```
/* prog05.c – μερικές ιδιότητες της printf() */

#include <stdio.h>

main()

{

        int ten = 10;
```

```
        printf("%d μείον %d ίσον %d\n", ten, 2, ten-2);
}
```

Το αποτέλεσμα του προγράμματος θα είναι :

10 μείον 2 ίσον 8

Μπορούμε, όμως, να εκτυπώσουμε τους ακέραιους αριθμούς στο οκταδικό σύστημα με το σύμβολο *%o* και στο δεκαεξαδικό με το σύμβολο *%x*. Ακολουθεί ένα απλό παράδειγμα :

```
/* prog06.c – εκτύπωση του αριθμού 100 σε τρία αριθμητικά συστήματα */

#include <stdio.h>

main()

{

        int x = 100;

        printf("δεκαδ = %d; οκταδ = %o; δεκαεξ = %x\n", x, x, x);

}
```

Το αποτέλεσμα του προγράμματος θα είναι :

δεκαδ = 100; οκταδ = 144; δεκαεξ = 64

Παρατηρούμε ότι τα προθέματα 0 και 0x δεν παρουσιάζονται στην έξοδο.

4.12.2 Άλλοι Τύποι Ακεραίων

Η C προσφέρει τρεις τύπους λέξεις-κλειδιά για την τροποποίηση του βασικού τύπου ακεραίου : ***unsigned***, ***long*** και ***short***. Ο τύπος ***short int*** ή ***short*** είναι ένας τύπος με πρόσημο και χρησιμοποιεί λιγότερο χώρο αποθήκευσης, απ' ό,τι ο int και έτσι μπορεί να χρησιμοποιηθεί για μικρούς αριθμούς. Ο τύπος ***long int*** ή ***long*** είναι ένας

τύπος με πρόσημο που χρησιμοποιεί περισσότερο χώρο αποθήκευσης απ' ό,τι ο int και συνεπώς επιτρέπει τη χρήση μεγάλων ακεραίων. Ο τύπος ***unsigned*** επιτρέπει τη χρήση της κλίμακας από το 0 μέχρι το 65535 αντί για την κλίμακα από το -32768 μέχρι το +32767.

Όσον αφορά τώρα τη χρήση αυτών των τύπων ακεραίων, πρέπει να έχουμε υπόψη μας ότι ο τύπος unsigned μπορεί να χρησιμοποιηθεί για μέτρημα, αφού δεν υπάρχουν σ' αυτόν τον τύπο αρνητικοί αριθμοί. Ο τύπος long χρησιμοποιείται αν έχουμε πολύ μεγάλους αριθμούς, που δεν μπορεί να τους διαχειριστεί ο τύπος int. Δεν πρέπει, όμως, να χρησιμοποιείται ο τύπος long αν αυτό δεν είναι απαραίτητο, γιατί επιβραδύνει τους υπολογισμούς και καταλαμβάνει πολύ μνήμη. Μπορούμε να χρησιμοποιήσουμε τον τύπο short όταν χρησιμοποιούμε μεγάλους πίνακες ακεραίων στο πρόγραμμά μας.

Για να αποθηκευθεί μια σταθερά σαν τύπου long, θα πρέπει να προσθέσουμε την κατάληξη l ή L και οι καταλήξεις αυτές μπορούν να χρησιμοποιηθούν ακόμα και με τους οκταδικούς ή τους δεκαεξαδικούς ακέραιους. Για την εκτύπωση ενός αριθμού τύπου unsigned χρησιμοποιούμε το σύμβολο *%u*, ενός αριθμού τύπου long το *%ld* και ακόμη μπορούμε να συνδυάσουμε το l με τα o και x. Ακόμη, το *%h* χρησιμοποιείται για τον τύπο short.

4.12.3 Οι Χαρακτήρες

Όπως ήδη ξέρουμε, ο τύπος ***char*** χρησιμοποιείται για την αποθήκευση χαρακτήρων, αν και στην πραγματικότητα αποθηκεύει ακέραιους. Για τον χειρισμό των χαρακτήρων ο υπολογιστής χρησιμοποιεί έναν αριθμητικό κώδικα, όπου συγκεκριμένοι ακέραιοι παριστάνουν συγκεκριμένους χαρακτήρες. Ο πιο συχνά χρησιμοποιούμενος κώδικας είναι ο ASCII και για παράδειγμα ο ακέραιος 65 παριστάνει το γράμμα A. Οι μεταβλητές τύπου char δηλώνονται όπως όλες οι μεταβλητές :

char name, city;

Υπάρχουν εκδόσεις της C που έχουν τον τύπο char με πρόσημο και με περιοχή τιμών από -128 έως +127 και άλλες εκδόσεις της C έχουν τον τύπο char χωρίς πρόσημο και με περιοχή τιμών από 0 έως 255. Μπορούμε να δώσουμε αρχικές τιμές στις μεταβλητές τύπου char με δύο τρόπους :

```
char grade = 65;    /* εδώ χρησιμοποιούμε τον ASCII κώδικα του A */
```

ή

```
char grade = 'A';    /* εδώ τα πράγματα είναι πιο απλά για μας */
```

Τα απλά εισαγωγικά ' ' δηλώνουν στη C μια *σταθερά χαρακτήρα*, ενώ τα διπλά εισαγωγικά " " θεωρούνται μια *συμβολοσειρά*, που θα την δούμε αναλυτικά παρακάτω.

4.12.4 Οι μη Εκτυπούμενοι Χαρακτήρες

Υπάρχουν μερικοί ASCII χαρακτήρες που είναι μη εκτυπούμενοι, όπως το διάστημα προς τα πίσω (backspace), το enter και το ηχητικό σήμα (beep). Η C μάς παρέχει τρεις τρόπους για την αναπαράσταση τέτοιων χαρακτήρων. Ο πρώτος είναι χρησιμοποιώντας τον κώδικα ASCII του χαρακτήρα, ως εξής :

```
char beep = 7;
```

Ο δεύτερος είναι χρησιμοποιώντας μια ειδική μορφή του κώδικα ASCII, όπου τοποθετούμε ανάμεσα σε απλά εισαγωγικά τον οκταδικό κώδικα ASCII μαζί με μια πλάγια κάθετο \ πριν απ’ αυτόν, ως εξής :

```
beep = '\007';
```

Οι αριθμοί ερμηνεύονται σαν οκταδικοί, ακόμη και αν δεν υπάρχει μπροστά ένα 0. Η ANSI C και πολλές νέες εκδόσεις δέχονται μια δεκαεξαδική μορφή για τις σταθερές χαρακτήρα, όπου η πλάγια κάθετος \ ακολουθείται από ένα x ή X και από 1 μέχρι 3 δεκαεξαδικά ψηφία. Έτσι, ο χαρακτήρας Control-P θα μπορεί να παρασταθεί

σαν '\x10' ή '\X010'. Ο τρίτος τρόπος είναι να χρησιμοποιήσουμε μια ειδική ακολουθία συμβόλων, που ονομάζεται *ακολουθία διαφυγής* ***(escape sequence)*** :

\a ειδοποίηση - beep (ANSI C)

\b ένα διάστημα προς τα πίσω (προσοχή δεν σβήνει τους χαρακτήρες)

\f τροφοδότηση σελίδας

\n νέα γραμμή (enter)

\r επιστροφή στην αρχή της τρέχουσας γραμμής

\t οριζόντια στηλοθέτηση (tab)

\v κάθετη στηλοθέτηση (ANSI C)

\\ πλάγια κάθετος (\)

\' απλά εισαγωγικά (')

\" διπλά εισαγωγικά (") (ANSI C)

Όλοι αυτοί οι χαρακτήρες κλείνονται σε απλά εισαγωγικά όταν καταχωρούνται σε μια μεταβλητή χαρακτήρα :

next = '\n';

Η εκτύπωση της μεταβλητής next προχωράει την εκτύπωση κατά μια γραμμή στην οθόνη ή στον εκτυπωτή. Η συνάρτηση printf() χρησιμοποιεί το %c για να δείξει ότι πρέπει να τυπωθεί ένας χαρακτήρας. Θα πρέπει να θυμόμαστε ότι οι χαρακτήρες είναι αποθηκευμένοι σαν ακέραιοι, οπότε αν τυπώσουμε την τιμή μιας μεταβλητής τύπου char, θα πάρουμε έναν ακέραιο.

Ο προσδιορισμός %c λέει στην printf() να μετατρέψει τον ακέραιο στον αντίστοιχο χαρακτήρα. Αυτά φαίνονται καθαρά στο παρακάτω πρόγραμμα :

/* prog07.c - εμφανίζει τον κωδικό αριθμό κάποιου χαρακτήρα */

```
#include <stdio.h>

main()

{

        char ch;

        printf("Δώστε έναν χαρακτήρα : \n");

        scanf("%c", &ch);

        printf("Ο κώδικας του %c είναι το %d. \n", ch, ch);

}
```

Το αποτέλεσμα θα είναι :

```
Δώστε έναν χαρακτήρα :

C

Ο κώδικας του C είναι το 67.
```

Η printf() τυπώνει την τιμή της ch δύο φορές, την πρώτη σαν χαρακτήρα (προτρεπόμενη από τον κώδικα %c) και τη δεύτερη σαν ακέραιο (προτρεπόμενη από τον κώδικα %d). Οι προσδιοριστές της printf() αποφασίζουν για το πώς θα εμφανιστούν τα δεδομένα και όχι για το πώς θα αποθηκευθούν.

4.12.5 Οι Τύποι float και double

Σε προγράμματα μαθηματικής φύσης συχνά χρησιμοποιούμε αριθμούς κινητής υποδιαστολής που στη C λέγονται τύπου ***float*** και είναι αντίστοιχοι με τους τύπους real της Pascal. Έτσι, μπορούμε να παραστήσουμε ένα πολύ μεγαλύτερο εύρος αριθμών, μεγάλων και μικρών. Υπάρχουν ακόμα και οι τύποι ***double*** (για διπλή ακρίβεια) και ***long double***. Οι μεταβλητές αυτές δηλώνονται όπως και οι άλλες :

```
float a=10.25e-3;

double b;

long double c;
```

Ένα πρόθεμα f ή F στην ANSI C σ' έναν αριθμό κινητής υποδιαστολής τον κάνει τύπου float, π.χ. 2.3f και 9.11E9F, ενώ μια κατάληξη l ή L τον κάνει τύπου long double, π.χ. 54.31l και 4.32e4L.

Η συνάρτηση printf() χρησιμοποιεί τον προσδιοριστή μορφής *%f* για να τυπώσει τύπου float και double αριθμούς με δεκαδικό συμβολισμό και τη μορφή *%e* για να τους τυπώσει σε εκθετικό συμβολισμό. Για τους τύπους long double έχουμε αντίστοιχα τους προσδιοριστές %Lf και %Le.

4.12.6 Ο Τελεστής sizeof

Η C έχει έναν εσωτερικό τελεστή, τον *sizeof*, που δίνει το μέγεθος κάποιων πραγμάτων σε bytes. Αυτό θα το δούμε καλύτερα μ' ένα παράδειγμα :

```
/* prog08.c – τυπώνει τα μεγέθη των τύπων δεδομένων */

#include <stdio.h>

main()

{

        printf("Ο τύπος int έχει μέγεθος %d bytes. \n", sizeof(int));

        printf("Ο τύπος char έχει μέγεθος %d bytes. \n", sizeof(char));

        printf("Ο τύπος long έχει μέγεθος %d bytes. \n", sizeof(long));

        printf("Ο τύπος double έχει μέγεθος %d bytes. \n", sizeof(double));

}
```

Το αποτέλεσμα θα είναι :

Ο τύπος ***int*** *έχει μέγεθος* ***2 bytes.***

Ο τύπος ***char*** *έχει μέγεθος* ***1 bytes.***

Ο τύπος ***long*** *έχει μέγεθος* ***4 bytes.***

Ο τύπος ***double*** *έχει μέγεθος* ***8 bytes.***

4.13 Τελεστές

Η C έχει τέσσερις τύπους τελεστών: **αριθμητικοί, σύγκρισης (συσχεσιακοί), λογικοί και τελεστές χειρισμού bits (bitwise operators).** Παρακάτω δίνεται ένας πίνακας με όλους τους τελεστές διατεταγμένους σύμφωνα με την προτεραιότητά τους.

προτεραιότητα	τελεστές
υψηλή	( ) [ ] ->
	! ~ ++ -- -(type) * & sizeof
	* / %
	- +
	<< >>
	< <= > >=
	== !=
	&
	^
	|
	&&
	||
	?
χαμηλή	= += -= *= /=

4.13.1 Αριθμητικοί τελεστές

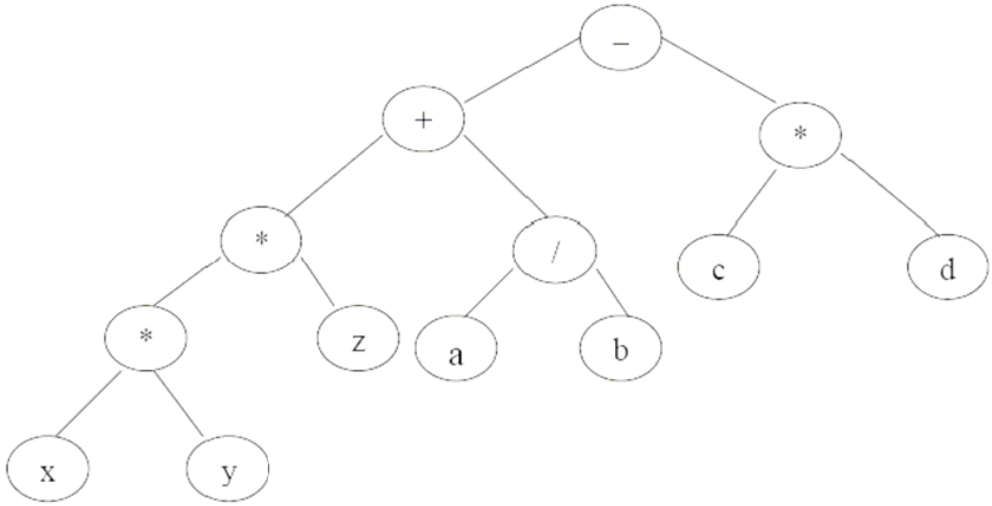
Η C ακολουθεί την αλγεβρική προτεραιότητα στους αριθμητικούς τελεστές. Δηλαδή, μεγαλύτερη προτεραιότητα έχουν οι παρενθέσεις, μετά το πρόσημο (δηλαδή το μείον), ακολουθούν ο πολλαπλασιασμός, η διαίρεση και το modulo και τέλος οι τελεστές πρόσθεσης και αφαίρεσης. Η εντολή καταχώρησης έχει μικρότερη προτεραιότητα από όλες τις πράξεις. Η φορά των πράξεων είναι από αριστερά προς δεξιά για όλους τους αριθμητικούς τελεστές, εκτός αν χρησιμοποιείται το μείον σαν πρόσημο, οπότε η φορά είναι από δεξιά προς αριστερά!

-	αφαίρεση, πρόσημο
+	πρόσθεση
*	πολλαπλασιασμός
/	διαίρεση
%	(mod)
--	ελάττωση μεταβλητής κατά 1
++	αύξηση μεταβλητής κατά 1

Οι τελεστές -- και ++ μπορεί να τοποθετηθούν μπροστά ή μετά ένα τελεσταίο. Η εντολή --x; ισοδυναμεί με την x:=x-1 αλλά η αφαίρεση εκτελείται πριν χρησιμοποιήσουμε την τιμή της x. Όμοια και η εντολή ++x; Η εντολή x--; Ισοδυναμεί με την x:=x-1 αλλά η αφαίρεση εκτελείται αφού χρησιμοποιήσουμε την τιμή της. x.x=3; x=3;y=++x; y=x++; αποτέλεσμα: y=4 (x=4) αποτέλεσμα: y=3 (x=4). Δυαδικοί τελεστέοι στην ίδια υποέκφραση και με την ίδια προτεραιότητα αποτιμούνται από τα αριστερά προς τα δεξιά – *αριστερή εταιρικότητα.*

Παράδειγμα

Η έκφραση x * y * z + a / b – c * d ερμηνεύεται ως: (((x * y) * z) + (a / b)) – (c * d)



Συντακτικά Ορθές	Συντακτικά Λανθασμένες
x + y / 2	((3 + z) / 2
(–x / y) * 2	–2x / y
((3))	5 +* 9
+(9 + z)	6 ^ 4
5 – –x	

Σημείωση: Δεν υπάρχει τελεστής που υψώνει στη δύναμη. Η λειτουργία παρέχεται από τη συνάρτηση pow που ορίζεται στη βιβλιοθήκη math.

4.13.2 Οι τελεστές συσχετισμού

Πρόκειται για τελεστές που τους χρησιμοποιούμε για τη δημιουργία απλών λογικών εκφράσεων συσχετισμού (σύγκρισης). Αυτοί είναι:

<	Μικρότερο από (πχ. αν i<j επιστρέφει 1 αντο i είναι μικρότερο από το j)
>	Μεγαλύτερο από
<=	Μικρότερο από ή ίσο
>=	Μεγαλύτερο από ή ίσο
==	Λογικό ίσον
!=	Διάφορο (όχι ίσο)

4.13.3 Λογικοί τελεστές

&&	Λογικό «και» (AND) μας επιστρέφει 1 αν και δύο μεταβλητές είναι λογικό μηδέν, διαφορετικά μηδέν
II	Λογικό διαζευτικό «ή» (είτε -OR)
I	Λογική άρνηση (NOT)

4.13.4 Ψηφιακοί τελεστές

τελεστής	περιγραφή
&	Ψηφιακό AND
|	Ψηφιακό OR
^	Ψηφιακό αποκλειστικό OR
~	Συμπλήρωμα ως προς ένα
&=	Σύνθετο ψηφιακό AND
|=	Σύνθετο ψηφιακό OR
^=	Σύνθετο ψηφιακό αποκλειστικό OR
<<	Ολίσθηση κατά ένα ψηφίο αριστερά
>>	Ολίσθηση κατά ένα ψηφίο δεξιά

4.13.5 Εντολή while

Η εντολή while χρησιμοποιείται αντί της for όταν δε μπορούμε να προβλέψουμε εύκολα πόσες φορές θέλουμε να εκτελεστούν οι εντολές ή όταν δεν έχει σημασία ο αριθμός των επαναλήψεων αλλά η ικανοποιήση ή όχι της συνθήκης: while (συνθήκη) { εντολές; }

4.13.6 Εντολή if-else

Οι εντολές if και if-else υπάρχουν σχεδόν σε όλες τις γλώσσες προγραμματισμού και χρησιμοποιούνται για να ελέγξουν αν ισχύει ή όχι κάποια συνθήκη:

Γενική σύνταξη της εντολής	παράδειγμα

if
(<ΣΥΝΘΗΚΗ>)
Ενότητα -Α
else
Ενότητα-Β
	If (a= = 0)
	Printf ("η μεταβλητή α είναι ίση με μηδέν");
	else
	Printf (η μεταβλητή α διάφορη του μηδενός);

Η <ΣΥΝΘΗΚΗ> μπορεί να είναι λογική έκφραση, έκφραση συσχετισμού, αποτέλεσμα κάποιας πράξης, είτε ακόμα και κάποια μεταβλητή. Η Ενότητα-Α και η ενότητα-Β μπορεί να περιλαμβάνουν μία εντολή ή πολλές εντολές (block) που περικλείονται σε άγκιστρα ({,}). Οι εντολές (ή η εντολή) της Ενότητας-Α εκτελούνται αν η <ΣΥΝΘΗΚΗ> είναι αληθής (όχι 0), ενώ οι εντολές (ή η εντολή) της ενότητας-Β εκτελείται αν η <ΣΥΝΘΗΚΗ> είναι ψευδής (0-μηδέν). Το else είναι προαιρετικό και όταν υφίσταται αναφέρεται στο πλησιέστερο πριν από αυτό if που δεν έχει else. Κάθε ενότητα είναι δυνατόν να περικλείει και άλλες if-else ενότητες (blocks) εντολών.

4.13.7 Εντολή switch

Η γενικευμένη μερφή της εντολής	Παράδειγμα
switch (έκφραση) { case σταθερά 1: εντολή 1; break ; case σταθερά 2: εντολή 2; break; default: εντολή N; break;	Printf ("δώσε ακέραια τιμή στη μεταβλητή α"); Scanf ("%d &a"); Switch (a) { case 1; printf ("η μεταβλητή α είναι 1"); break; case2: printf ("η μεταβλητή α είναι 2"); break ;

Εκτελείται η εντολή της οποίας η σταθερά ταιριάζει με την τιμή της έκφρασης. Η εκτέλεση του προγράμματος μέσα σε ένα **switch** συνεχίζεται στο επόμενο **case**, εκτός αν δεν μεσολαβεί κάποια από τις εντολές **break**, **exit** ή **return**. Αν η τιμή δεν ταιριάζει με καμία σταθερά, τότε εκτελείται η εντολή στο block της **default** (που είναι προαιρετικό).

4.13.8 Εντολή break

Η εντολή **break** προκαλεί τον τερματισμό μιας εντολής **switch** ή και μιας επαναληπτικής διαδικασίας (που προκλήθηκε με εντολή **for** ή την **while** – δείτε την επόμενη ενότητα). Η εκτέλεση του προγράμματος συνεχίζεται μετά την εντολή **switch** ή την εντολή επανάληψης.

```
while(...)
{
. . .
break;
. . .
}
. . . /* Η εκτέλεση του προγράμματος συνεχίζεται μετά το break */
```

ΚΕΦΑΛΑΙΟ 5^{Ο} - ΕΦΑΡΜΟΓΕΣ ΤΩΝ ΜΙΚΡΟΕΛΕΓΚΤΩΝ AVR

1^{Η} ΕΦΑΡΜΟΓΗ: ΣΥΣΤΗΜΑ ΔΙΑΣΦΑΛΙΣΗΣ ΛΟΓΙΣΜΙΚΟΥ (SECURITY DONGLE)

5.1 Εισαγωγή - Τι είναι τα συστήματα διασφάλισης λογισμικού

Τί αντιπροσωπεύουν άραγε τα συστήματα διασφάλισης λογισμικού; Στο χώρο των ηλεκτρονικών υπολογιστών και των διαφόρων εφαρμογών λογισμικού, αυτά τα συστήματα διασφάλισης, γνωστά επίσης και ως «Κλειδιά Προγραμμάτων», αναφέρονται σε ένα σχήμα εξουσιοδότησης σύμφωνα με το οποίο κάποιο συγκεκριμένο λογισμικό, ενδεχόμενα σε συνεργασία και με κάποιο αντίστοιχο υλικό, επιτρέπεται να χρησιμοποιηθεί από εξουσιοδοτημένους χρήστες και μόνον. Με βάση ένα τέτοιο σύστημα, ζητείται από τον χρήστη ενός υπολογιστή ένα είδος αναγνώρισης εισόδου στο σύστημα η οποία συνοδεύεται από την απαραίτητη κωδική λέξη. Σε περίπτωση που δεν δοθεί η σωστή κωδική λέξη που αντιστοιχεί σε μια αναγνώριση εισόδου, ο υπολογιστής δεν επιτρέπει στον χρήστη την είσοδο στο σύστημά του. Πρόκειται για ένα απλό παράδειγμα ενός λογισμικού συστήματος προστασίας.

Παρόμοια, όταν αγοράζουμε ένα ακριβό πρόγραμμα ο σχεδιαστής του προγράμματος αυτού ή ακόμη και ο προμηθευτής του, επιθυμώντας να επιβάλλουν κάποιον έλεγχο στην χρήση προγράμματος, υιοθετούν έναν κατάλληλο μηχανισμό προστασίας με σκοπό να περιορίσουν την διάδοση του συγκεκριμένου λογισμικού σε μη εξουσιοδοτημένους χρήστες, εφόσον κάτι τέτοιο θα οδηγήσει σε μείωση των εσόδων τους από το προϊόν αυτό. Για να είναι ένας τέτοιος έλεγχος εφικτός, ο σχεδιαστής του προγράμματος παρέχει συνήθως μαζί με το πρόγραμμα και ένα κατάλληλο συνοδευτικό κύκλωμα που συνδέεται στον υπολογιστή στον οποίο πρόκειται να λειτουργήσει τος συγκεκριμένο πρόγραμμα. Όταν εκτελέσουμε το πρόγραμμα αυτό στον συγκεκριμένο υπολογιστή, εκτελείται αρχικά μια αναζήτηση του κυκλώματος προστασίας και σε περίπτωση που το πρόγραμμα αποτύχει να εντοπίσει το κύκλωμα αυτό τότε η εκτέλεση του ματαιώνεται αυτόματα. Αντίθετα, όταν το πρόγραμμα εντοπίσει το κύκλωμα τότε φορτώνεται και εκτελείται κανονικά.

Το συνοδευτικό αυτό κύκλωμα προστασίας φιλοξενείται συνήθως σε μια μικρή πλακέτα τοποθετημένη σε συμπαγή συσκευασία η οποία συνδέεται στον υπολογιστή.

Για την διευκόλυνση της διαδικασίας σύνδεσης και αποσύνδεσης του συστήματος αυτού στον υπολογιστή, το αντίστοιχο κύκλωμα σχεδιάζεται συνήθως με τέτοιον τρόπο ώστε να μπορεί να συνδεθεί είτε στην παράλληλη, είτε στη σειριακή θύρα επικοινωνίας του υπολογιστή, επειδή η προσπέλαση των θυρών αυτών από τον χρήστη είναι αρκετά εύκολη. Παρόμοια κύκλωματα προστασίας λογισμικού διατίθενται και με δυνατότητα σύνδεσης στην θύρα USB. Ένα τυπικό σύστημα διασφάλισης λογισμικού λειτουργεί με τον ακόλουθο τρόπο:

1. Ο υπολογιστής αποστέλλει πληροφορίες στο κύκλωμα προστασίας και στη συνέχεια αναμένει από αυτό να αποστείλει με τη σειρά του, επίσης κάποιες πληροφορίες.
2. Εφόσον το κύκλωμα διασφάλισης έχει εγκατασταθεί και λειτουργεί σωστά θα επιστρέψει στον υπολογιστή τις αναμενόμενες πληροφορίες, οπότε το αντίστοιχο πρόγραμμα θα εκτελεστεί κανονικά.
3. Αν το ίδιο κύκλωμα δεν έχει εγκατασταθεί σωστά ή έχει απομακρυνθεί για κάποιον λόγο από το σύστημα, τότε το πρόγραμμα δεν λαμβάνει τις ζητούμενες πληροφορίες και η εκτέλεσή του ματαιώνεται. Το ίδιο το πρόγραμμα σε κανονικές συνθήκες λειτουργίας, αναζητά σε τακτά χρονικά διαστήματα το εγκατεστημένο σύστημα προστασίας.

5.1.1 Διάφορες Τεχνικές Υλοποίησης Συστημάτων Διασφάλισης Λογισμικού

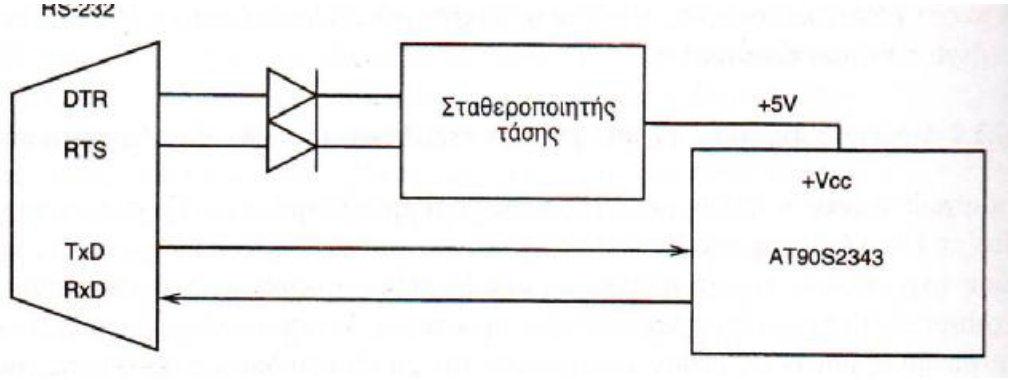
Υπάρχουν διάφοροι τρόποι με τους οποίους ένα πρόγραμμα αποστέλλει πληροφορίες σε ένα κύκλωμα προστασίας αναμένοντας την σωστή ανταπόκρισή του, με βάση τους οποίους γίνεται η διάκριση μεταξύ ποικίλων κυκλωμάτων προστασίας λογισμικού. Ένα ολοκληρωμένο σύστημα προστασίας λογισμικού δεν χρησιμοποιείται συνήθως μόνον ως μέσον περιορισμού της μη εξουσιοδοτημένης χρήσης του υπό προστασία λογισμικού, αλλά ταυτόχρονα πρέπει να προστατεύει το ίδιο το πρόγραμμα από οποιαδήποτε απόπειρα παράνομης διείσδυσης ή και αντιγραφής. Κατι τέτοιο μπορεί να είναι εφικτό σε περίπτωση που το αντίστοιχο σύστημα προστασίας διαθέτει έναν απεριόριστο αριθμό συνδυασμών. Αν σε ένα τέτοιο σύστημα προστασίας αποστέλλονται συνεχώς οι ίδιες πληροφορίες στο πρόγραμμα το οποίο αναμένει ένα συγκεκριμένο είδος απόκρισης, τότε η διαδικασία αυτή δεν θεωρείται αξιόπιστη εφόσον είναι δυνατό μια τέτοια συγκεκριμένη ακολουθία να αντιγραφεί. Στην ιδανική περίπτωση το κύκυλωμα προστασίας θα

πρέπει να διαθέτει απεριόριστους συνδυασμούς, αν και κάτι τέτοιο δεν είναι πρακτικά εφικτό.

Ένα καλό φορητό σύστημα διασφάλισης λογισμικού θα πρέπει να διαθέτει τα ακόλουθα χαρακτηριστικά:

1. Να συνδέεται σε εύκολα προσπελάσιμες θύρες του υπολογιστή όπως στην παράλληλη θύρα ή την θύρα RS232.
2. Να λαμβάνει τροφοδοσία από το κύκλωμα των θυρών επικοινωνίας με τις οποίες συνδέεται.
3. Να διαθέτει έναν ικανοποιητικά μεγάλο αριθμό πιθανών συνδυασμών έτσι ώστε να αποτρέπεται κάθε πιθανή απόπειρα αντιγραφής.
4. Για τα κυκλώματα προστασίας που συνδέονται στην παράλλληλη θύρα θα πρέπει επιπλέον να υπάρχει και η δυνατότητα διακλάδωσης της σύνδεσης στην θύρα αυτή, δηλαδή να μπορεί ο χρήστης του συστήματος να συνδέει στην ίδια θύρα και άλλες περιφερειακές συσκευές εκτός του κυκλώματος προστασίας.
5. Το πρόγραμμα της εφαρμογής του οποίου η χρήση διασφαλίζεται, θα πρέπει να εξετάζει περιοδικά το σύστημα με σκοπό να ανιχνεύει την παρουσία του κυκλώματος προστασίας, επικοινωνώντας μαζί του.

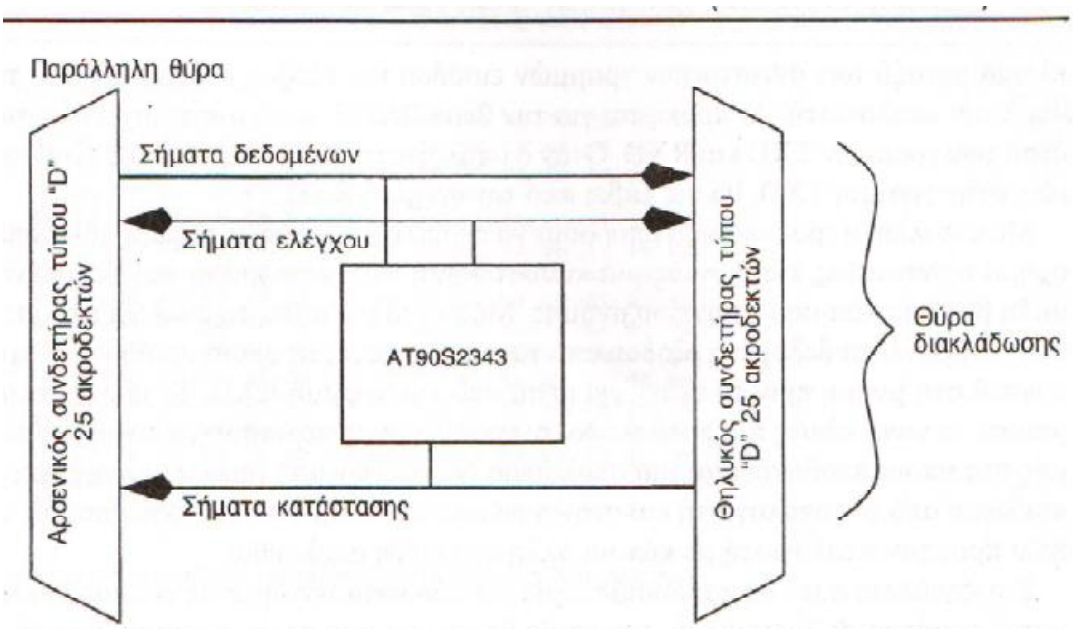
Στο παρακάτω σχήμα φαίνεται το διάγραμμα ενός τέτοιου συστήματος το οποίο συνδέεται στην θύρα RS232 ενός υπολογιστή. Το διάγραμμα αυτό εμφανίζει ένα κύκλωμα το οποίο περιλαμβάνει τον μικροελεγκτή AT90S2343 με σκοπό να φανερώσει ότι μια τέτοια διάταξη μπορεί να υλοποιηθεί κατά βάση χρησιμοποιώντας τον συγκεκριμένο μικροελεγκτή. Οι γραμμές των σημάτων της θύρας RS232 δε χρησιμοποιούνται μόνο για την διαδικασία της επικοινωνίας του υπολογιστή με το κύκλωμα προστασίας, αλλά και για την τροφοδοσία του. Οι γραμμές αυτές φέρουν τάσεις των +12 V ή των – 12 V. Το πρόγραμμα της εφαρμογής θα πρέπει να οδηγήσει τις γραμμές αυτές στα + 12 V, των οποίων η τάση υποβιβάζεται στο επιθυμητό επίπεδο με σκοπό να τροφοδοτηθεί το κύκλωμα. Οι άλλες δύο γραμμές που φαίνονται στο ίδιο σχήμα, δηλαδή οι γραμμές των σημάτων TXD και RXD, χρησιμοποιούνται από το κύκλωμα προστασίας για την εκπομπή και τη λήψη δεδομένων αντίστοιχα.



Σχήμα 5.1: Διάγραμα ενός κυκλώματος προστασίας λογισμικού που συνδέεται στη θύρα RS232 του υπολογιστή.

Η παραπάνω σύνθεση συνιστά μια βασική υποδομή στην οποία βασίζεται ένας συγκεκριμένος αλγόριθμος υλοποίησης ενός συστήματος προστασίας λογισμικού. Βέβαια, υποτίθεται ότι το συγκεκριμένο κύκλωμα υποστηρίζει τις ανάγκες που προκύπτουν από τον αντίστοιχο αλγόριθμο.

Ένας άλλος τρόπος σχεδίασης ενός συστήματος προστασίας λογισμικού, με δυνατότητα σύνδεσης αυτή τη φορά στην παράλληλη θύρα του υπολογιστή, παρουσιάζεται στο ακόλουθο σχήμα. Η μέθοδος αυτή είναι κάπως πιο απαιτητική από την προηγούμενη, στην οποία πραγματοποιείται σύνδεση με την θύρα RS232. Ο λόγος βασίζεται στο γεγονός ότι οι περισσότεροι υπολογιστές διαθέτουν μόνο μια παράλληλη θύρα την οποία δεν επιθυμούμε να δεσμεύουμε αποκλειστικά για την χρήση του κυκλώματος προστασίας. Για να ξεπεραστεί το πρόβλημα αυτό εφαρμόζεται η τεχνική της διακλάδωσης των γραμμών της παράλληλης θύρας, γεγονός που σημαίνει ότι οι γραμμές της θύρας αυτής μπορούν να διαμοιραστούν κατάλληλα ώστε να χρησιμοποιούνται ταυτόχρονα από αρκετές περιφερειακές συσκευές.



Σχήμα 5.2: Διάγραμμα ενός κυκλώματος προστασίας που συνδέεται στην παράλληλη θύρα του υπολογιστή με δυνατότητα διακλάδωσης.

5.1.2 Τρόποι Κατασκεύης Συστημάτων Διασφάλισης Λογισμικού

Έχοντας ολοκληρώσει την παρουσίαση των βασικών απαιτήσεων από ένα κύκλωμα προστασίας λογισμικού, θα προχωρήσουμε με την εξέταση του τρόπου κατασκευής του. Αναφέρθηκε παραπάνω ότι ένα σύστημα διασφάλισης λογισμικού βασίζεται σε έναν μηχανισμό ο οποίος καλείται αλγόριθμος προστασίας. Συνεπώς, ένας τέτοιος αλγόριθμος θα πρέπει να υλοποιηθεί με κάποιο συγκεκριμένο κύκλωμα. Το κύκλωμα αυτό θα πρέπει στη συνέχεια να συνδεθεί με κάποια από τις θύρες επικοινωνίας του υπολογιστή. Επίσης, το ίδιο κύκλωμα θα πρέπει και να τροφοδοτηθεί από τις γραμμές της αντίστοιχης θύρας. Το κύκλωμα θα πρέπει να χρησιμοποιεί την θύρα για να επικοινωνεί με το πρόγραμμα της εφαρμογής και κυρίως θα πρέπει να διαθέτει και δυνατότητα διακλάδωσης.

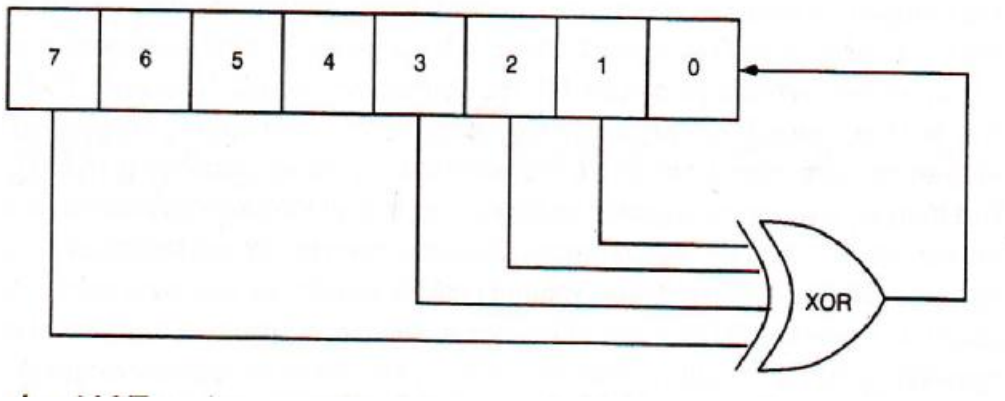
Αν επιθυμούμε να κατασκευάσουμε ένα κύκλωμα προστασίας θα πρέπει να διαθέτουμε τον αντίστοιχο αλγόριθμο προστασίας. Τέτοιοι αλγόριθμοι υπάρχουν σε μεγάλη αφθονία από πολύ βασικούς και απλούς, μέχρι εξαιρετικά πολύπλοκους. Το απλούστερο δυνατό σχήμα προστασίας θα μπορούσε να αποτελέσει ένα βραχυκύκλωμα μεταξύ των αντίστοιχων γραμμών εισόδου και εξόδου δεδομένων από την θύρα του υπολογιστή. Αν πρόκειται για την θύρα RS232, αυτό αντιστοιχεί στην

σύνδεση των γραμμών TXD και RXD. Όταν ο υπολογιστής εκπέμπει κάποιες πληροφορίες στην γραμμή TXD, θα τις λάβει από την γραμμή RXD.

Με κατάλληλη τροποποίηση μπορούμε να περιπλέξουμε κάπως το παραπάνω απλό σχήμα προστασίας, εισάγοντας μια καθυστέρηση στην επιστροφή των δεδομένων με τη βοήθεια κάποιου στοιχείου μνήμης. Με τον τρόπο αυτό, το κύκλωμα προστασίας λαμβάνει τα δεδομένα εξόδου από τον υπολογιστή, τα οποία αποθηκεύει προσωρινά στη μνήμη πριν τα εκπέμψει μέσα από την γραμμή RXD. Το ίδιο σύστημα μπορεί να γίνει κάπως πιο πολύπλοκο αν προσθέσουμε περισσότερα στοιχεία μνήμης στα οποία αποθηκεύεται μια ακολουθία δεδομένων από αυτά που λαμβάνει το κύκλωμα από τον υπολογιστή και στην συνέχεια τα δεδομένα αυτά εκπέμπονται και πάλι προς τον υπολογιστή με κάποια άλλη δεδομένη ακολουθία.

Στο κεφάλαιο αυτό θα παρουσιάσουμε μια μέθοδο κατασκεύης ενός τέτοιου κυκλώματος προστασίας λογισμικού, την οποία θεωρούμε αρκετά πρωτοποριακή. Το βασικότερο τμήμα του αλγόριθμου που πρόκειται να παρουσιάσουμε αντιστοιχεί στην υλοποίηση ενός κυκλικού καταχωρητή ολίσθησης. Το σύστημα προστασίας το οποίο βασίζεται στην τεχνική του κυκλικού καταχωρητή ολίσθησης, εκμεταλλεύεται τον μεγάλο αριθμό των συνδυασμών που αντιστοιχούν στους μακρούς κύκλους επανάληψης ενός τέτοιου καταχωρητή. Ένας κυκλικός καταχωρητής χωρητικότητας 8-bits διαθέτει 255 μοναδικούς συνδυασμούς. Παρόμοια, ένας κυκλικός καταχωρητής ολίσθησης, χωητικότητας 20- bits, διαθέτει ένα εκατομμύριο συνδυασμούς.

Στο σχήμα φαίνεται ένας κυκλικός καταχωρητής ολίσθησης χωρητικότητας 8-bits. Στον καταχωρητή αυτόν φορτώνεται αρχικά ένας αριθμός ο οποίος στη συνέχεια ολισθαίνει διαμέσου του συστήματος καταχωρητή. Κάθε μια διαδικασία ολίσθησης έχει ως αποτέλεσμα την εμφάνιση ενός νέου αριθμού, ο οποίος φαίνεται να μην έχει καμία σχέση με τον αρχικό αριθμό, δηλαδή την γενέτειρα. Η διαδικασία της ολίσθησης μπορεί να επαναληφθεί έως 255 φορές μέχρι να αρχίσει να επαναλαμβάνεται η ίδια ακολουθία αριθμών. Βέβαια, από την άποψη του ίδιου του συστήματος προστασίας, ένας κυκλικός καταχωρητής μεγαλύτερης χωρητικότητας θα ήταν ακόμη πιο χρήσιμος. Από την πλευρά ενός επίδοξου «πειρατή λογισμικού», η όλη διαδικασία αντιστοιχεί σε μια συνεχή αναζήτηση του σωστού εκείνου τυχαίου αριθμού ο οποίος μπορεί να σπάσει όπως λέγεται το πρόγραμμα, μια διαδικασία ωστόσο που εμπεριέχει πολλές απογοητευτικές προσπάθειες.



Σχήμα 5.3: Ένα σύστημα κυκλικού καταχωρητή ολίσθησης με ενδιάμεσες λήψεις από τις θέσεις που αντιστοιχούν στα bits 1,2,3 και 7.

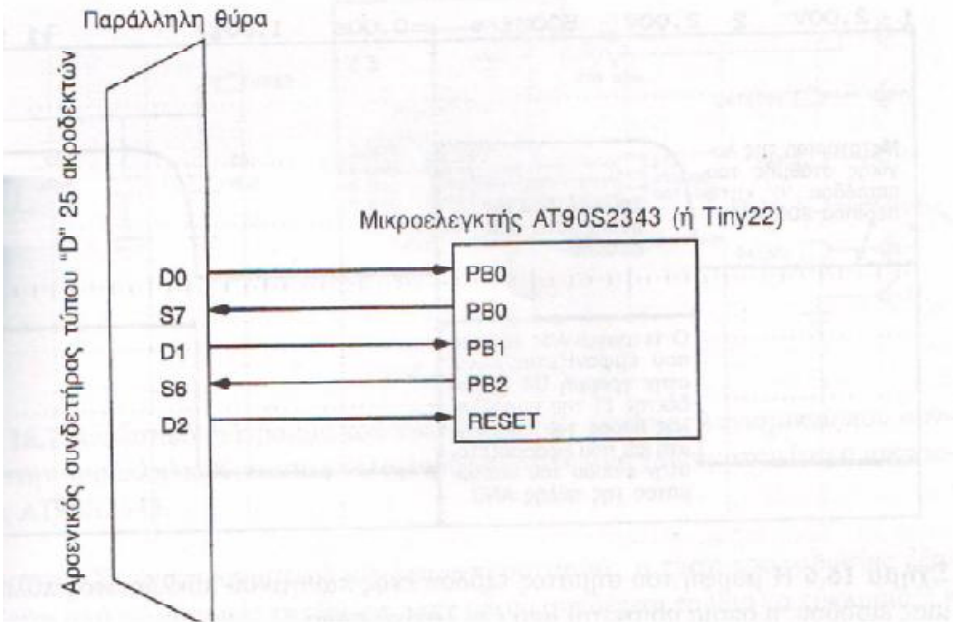
Η χωρητικότητα ενός τέτοιου κυκλικού καταχωρητή μπορεί να αυξηθεί στα 16-bits ή στα 20-bits ή ακόμη και περισσότερα, με σκοπό να παρέχει μεγαλύτερο πλήθος συνδυασμών κάτω από πραγματικές συνθήκες. Προς το παρόν θα περιοριστούμε στην κατασκευή ενός κυκλώματος προστασίας με βάση το απλό αυτό σχήμα του κυκλικού καταχωρητή των 8bits. Η προτεινόμενη λειτουργία του κυκλώματος αυτού έχει ως εξής:

1. Το κύκλωμα προστασίας θα πρέπει να επανατοποθετείται αμέσως μετά από κάθε διαδικασία επικοινωνίας με το πρόγραμμα. Η λειτουργία αυτή εξασφαλίζει τον συγχρονισμό μεταξύ του μικροελεγκτή του κυκλώματος και του υπολογιστή.
2. Ο υπολογιστής αποστέλλει δύο bytes δεδομένων. Το πρώτο από αυτά αντιστοιχεί στην γενέτειρα της ακολουθίας των τυχαίων αριθμών.
3. Το σύστημα του κυκλώματος προστασίας υπολογίζει το αποτέλεσμα και το επιστρέφει πίσω στον υπολογιστή.

Τον ίδιο υπολογισμό εκτελεί και ο υπολογιστής, ο οποίος στη συνέχεια συγκρίνει τον αριθμό που έλαβε από το κύκλωμα με το αποτέλεσμα του θπολογισμού. Εφόσον διαπιστωθεί ότι οι παραπάνω δύο αριθμοί είναι ίσοι μεταξύ τους ο υπολογιστής χαρακτηρίζει έγκυρη την παρουσία του κυκλώματος προστασίας και συνεπώς συνεχίζει να εκτελεί κανονικά το πρόγραμμα.

Η μετάδοση των δεδομένων μεταξύ του κυκλώματος και του υπολογιστή εκτελείται σειριακά και υποστηρίζεται από σήματα ενεργοποίησης και επιβεβαίωσης.

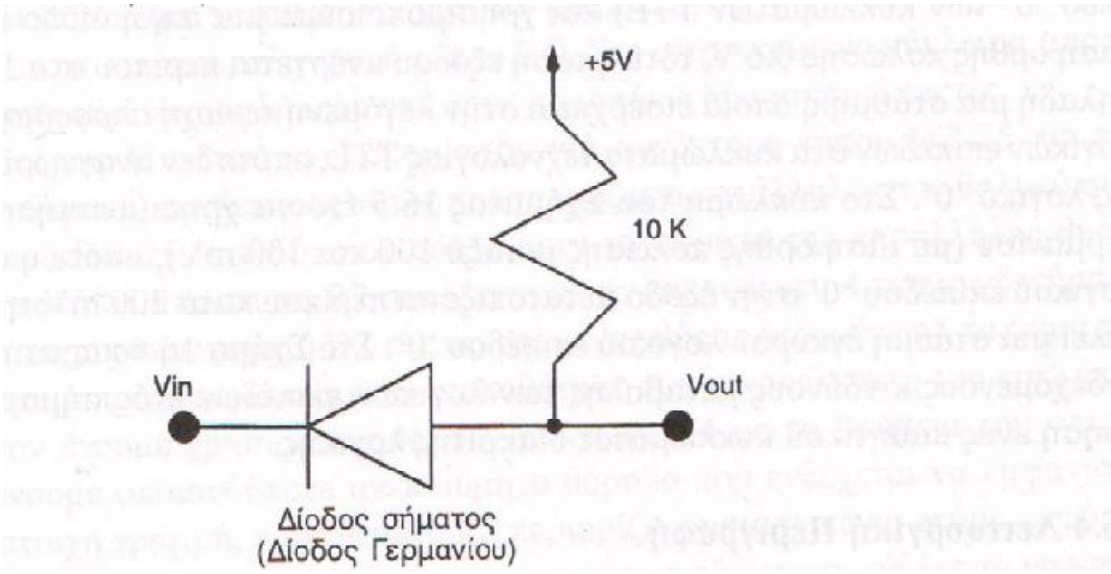
Η παράλληλη θύρα επικοινωνίας του υπολογιστή διαθέτει ουσιαστικά τρεις υφιστάμενες θύρες. Οι γραμμές σημάτων της παράλληλης θύρας που χρησιμοποιούνται για την διαδικασία επικοινωνίας με τον υπολογιστή είναι αυτή που αντιστοιχεί στο bit D0 της υφιστάμενης θύρας δεδομένων (DATA Port bit0) ως σειριακή έξοδος της πληροφορίας, εκείνη που αντιστοιχεί στο bit S7 της υφιστάμενης θύρας κατάστασης(STATUS Port bit7) ως είσοδος λήψης δεδομένων από το κύκλωμα προστασίας, η γραμμή που αντιστοιχεί στο bit D1 της υφιστάμενης θύρας δεδομένων (DATA Port bit1) ως γραμμή του σήματος ενεργοποίησης του κυκλώματος και η γραμμή που αντιστοιχεί στο bit S6 της υφιστάμενης θύρας κατάστασης (STATUS Port bit6) ως γραμμή του σήματος επιβεβαίωσης από το κύκλωμα προστασίας προς τον υπολογιστή. Από την πλευρά του κυκλώματος προστασίας χρησιμοποιούμε τη γραμμή εισόδου-εξόδου PB0 ως γραμμή εισόδου και εξόδου των σειριακών δεδομένων, τη γραμμή PB1 ως γραμμή λήψης του σήματος ενεργοποίησης (strobe) και τη γραμμή PB2 ως γραμμή εξόδου του σήματος επιβεβαίωσης (acknowledge). Από την πλευρά της παράλληλης θύρας του θπολογιστή χρησιμοποιείται μια επιπλέον γραμμή που αντιστοιχεί στο bit D2 της υφιστάμενης θύρας δεδομένων (DATA Port bit2) για την επανατοποθέτηση του μικροελεγκτή του κυκλώματος προστασίας. Στο σχήμα φαίνεται η παραπάνω διάταξη των γραμμών επικοινωνίας μεταξύ του υπολογιστή και του κυκλώματος προστασίας.



Σχήμα 5.4: Διάγραμμα των συνδέσεων του κυκλώματος προστασίας με την παράλληλη θύρα του υπολογιστή.

Παρατηρούμε ότι η γραμμή εισόδου-εξόδου PB0 του μικροελεγκτή συνδέεται τόσο στη γραμμή D0, όσο και στην γραμή S7 της παράλληλης θύρας. Βέβαια, οι συνδέσεις αυτές δεν είναι δυνατό να πραγματοποιηθούν απευθείας. Επομένως, θα πρέπει η γραμμή D0 να απομονωθεί από την γραμμή S7 με τέτοιον τρόπο, ώστε όταν ο μικροελεγκτής αποστέλλει δεδομένα στην γραμμή PB0 προς τη γραμμή S7, το επίπεδο της λογικής στάθμης στη γραμμή DO δεν θα επηρεάζει το αντίστοιχο λογικό επίπεδο που έχει τεθεί από τη γραμμή PB0.

Για να πετύχουμε την απομόνωση αυτή χρησιμοποιούμε μια διάταξη η οποία αντιστοιχεί σε λογική πύλης AND μιας εισόδου, όπως φαίνεται στο σχήμα.

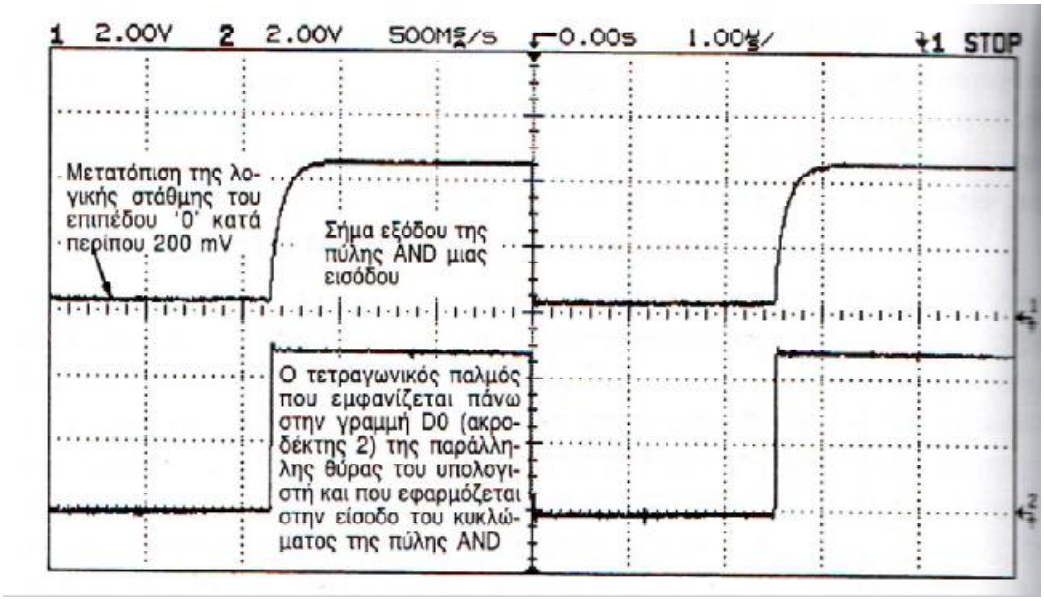


Σχήμα 5.5: Ένα απλό παθητικό κύκλωμα πύλης AND μίας εισόδου ως απομονωτής λογικών επιπέδων.

Η είσοδος της απλής αυτής πύλης AND συνδέεται στην γραμμή D0, ενώ η έξοδος της συνδέεται στις γραμμές PB0 και S7. Με τον τρόπο αυτό, όταν η είσοδος της πύλης AND βρεθεί σε λογικό ‘1’, η δίοδος αποκόπτεται οπότε το αντίστοιχο λογικό επίπεδο της γραμμής D0 απομονώνεται από το λογικό επίπεδο που έχει τεθεί από την γραμμή PB0.

Στο σχήμα, παρατηρούμε την επίδραση του παραπάνω παθητικού κυκλώματος της πύλης AND μιας εισόδου, στα λογικά επίπεδα του αντίστοιχου σήματος. Όταν η είσοδος του κυκλώματος της πύλης βρίσκεται σε λογικό ‘0’ τότε η δίοδος του κυκλώματος άγει και η τάση εξόδου ισούται με την τάση εισόδου μετατοπισμένη

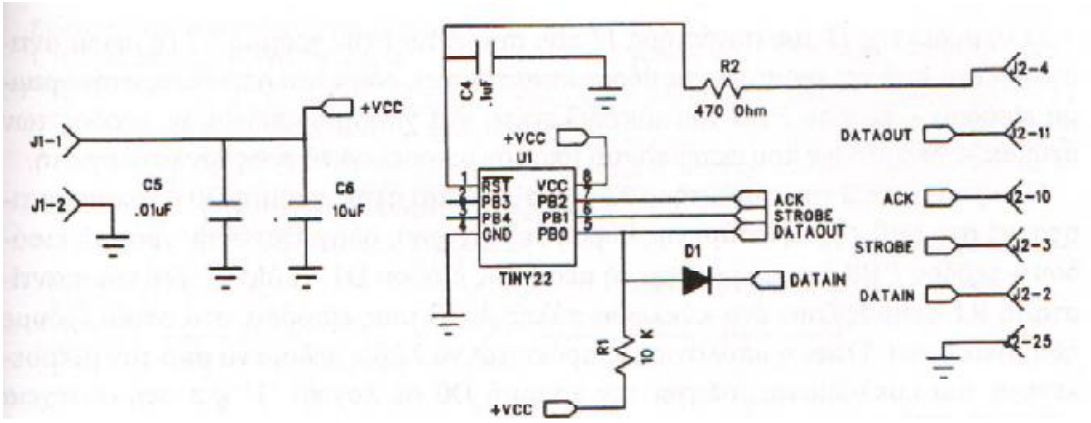
κατά την τιμή της τάσης ορθής πόλωσης της διόδου. Αν η αντίστοιχη στάθμη του λογικού επιπέδου ‘0’ ισούται με 0.5 V και χρησιμοποιούμε μια απλή δίοδο πυριτίου με τάση ορθής πόλωσης 0.6 V, τότε η τάση εξόδου ανέρχεται περίπου στα 1.1 με 1.2 V, δηλαδή μια στάθμη η οποία εισέρχεται στην λεγόμενη περιοχή απροσδιοριστίας των λογικών επιπέδων στα κυκλώματα τεχνολογίας TTL, οπότε δεν αναγνωρίζεται πλέον ως λογικό ‘0’. Στο κύκλωμα του σχήματος έχουμε χρησιμοποιήσει μια δίοδο γερμανίου, οπότε η στάθμη του λογικού επιπέδου ‘0’ στην έξοδο μετατοπίζεται περίπου κατά 200mV, η οποία αποτελεί μια στάθμη έγκυρου λογικού επιπέδου ‘0’. Στο σχήμα παρατηρούμε τους ενδεχόμενους κινδύνους μεταβολής των λογικών επιπέδων ενός σήματος από τη χρήση ενός κυκλώματος διακριτής λογικής.



Σχήμα 5.6: Η μορφή του σήματος εξόδου ενός παθητικού κυκλώματος πύλης AND μίας εισόδου, η οποία οδηγείται από ένα λογικό σήμα.

5.1.3 Λειτουργική Περιγραφή

Έχοντας ολοκληρώσει την αναφορά στην διαδικασία σχεδίασης του κυκλώματος που αντιστοιχεί στην έκδοση του δικού μας συστήματος διασφάλισης λογισμικού, θα προχωρήσουμε στη συνέχεια σε μια περιγραφή του πραγματικού κυκλώματος. Στο παρακάτω σχήμα φαίνεται το αναλυτικό διάγραμμα του κυκλώματος που χρησιμοποιεί το δικό μας σύστημα.



Σχήμα 5.7: Αναλυτικό διάγραμμα του κυκλώματος προστασίας λογισμικού που συνδέεται στηνμ παράλληλη θύρα του υπολογιστή και στο οποίο χρησιμοποιείται ο μικροελεγκτής AT90S2343.

Στο σχήμα αυτό φαίνεται ο μικροελεγκτής Tinny2 στην θέση του οποίου μπορεί να χρησιμοποιηθεί ισοδύναμα και ο μικροελεγκτής AT90S2343 με ενεργοποιημένο τον εσωτερικό ταλαντωτή χρονισμού τύπου RC.

Ας εξετάσουμε το κύκλωμα με λεπτομέρεια. Οι ακροδέκτες 1 και 2 του συνδετήρα j1(J1-1 και J1-2) χρησιμοποιούνται για την παροχή τροφοδοσίας (+5V) του κυκλώματος. Σε ένα πραγματικό κύκλωμα προστασίας, η τάση τροφοδοσίας εξασφαλίζεται από τις γραμμές της θύρας στην οποία συνδέεται το ίδιο το κύκλωμα. Οι πυκνωτές C5 και C6 χρησιμοποιούνται ως φίλτρο για την γραμμή της τάσης τροφοδοσίας. Το ολοκληρωμένο κύκλωμα U1 αντιστοιχεί στον μικροελεγκτή Tiny22, αλλά στο ίδιο κύκλωμα έχουμε τοποθετήσει και τον μικροελεγκτή AT90S2343 με ενεργοποιημένο τον εσωτερικό ταλαντωτή τύπου RC.Στο συγκεκριμένο κύκλωμα μπορεί να χρησιμοποιηθεί οποιοσδήποτε από τους παραπάνω δύο μκροελεγκτές.

Ο συνδετήρας j2 αντιστοιχεί στον αρσενικό συνδετήρα τύπου DB-25 για την παράλληλη θύρα. Ο συνδετήρας αυτός προσαρμόζεται κατάλληλα στον θηλυκό συνδετήρα τύπου DB-25, στον οποίον οδηγούνται τα σήματα της παράλληλης θύρας του Υπολογιστή. Η αντίσταση R2 συνδέεται με τον ακροδέκτη 4 του συνδετήρα J2 (που αντιστοιχεί στην γραμμή D2 της υφιστάμενης θύρας δεδομένων), το σήμα του οποίου χρησιμοποιείται για την επανατοποθέτηση του μικροελεγκτή του κυκλώματος. Εφόσον έχουμε χρησιμοποιήσει τον πυκνωτή C4 με τη βοήθεια του οποίου απομακρύνουμε οποιονδήποτε ανεπιθύμητο θόρυβο που ενδέχεται να εμφανιστεί στην αντίστοιχη γραμμή, η αντίσταση R2 περιορίζει ουσιαστικά το ρεύμα

εκφόρτισης του πυκνωτή, η τιμή του οποίου αν παραμείνει ανεξέλεγκτη ενδέχεται να καταστρέψει το κύκλωμα του ακροδέκτη που συνδέεται στην γραμμή D2. Τα ρεύματα εκφόρτισης των πυκνωτών μπορούν να λάβουν στιγμιαία μεγάλες τιμές (που περιορίζονται από την αντίστοιχη αντίσταση φορτίου), οπότε για τον περιορισμό του ρεύματος εκφόρτισης του πυκνωτή στο συγκεκριμένο κύκλωμα έχουμε χρησιμοποιήσει μια αντίσταση των 470 Ω. Η τιμή του ρεύματος εκφόρτισης που προκύπτει (στην χειρότερη περίπτωση 10mA) παρέχεται χωρίς πρόβλημα από τον αντίστοιχο ακροδέκτη της παράλληλης θύρας. Για την επανατοποθέτηση του μικροελεγκτή η γραμμή D2 τίθεται σε λογικό ‘0’ για ένα μικρό χρονικό διάστημα(0.001 s) και στη συνέχεια επανέρχεται σε κατάσταση λογικού ‘1’.

Ο ακροδέκτης 11 του συνδετήρα J2 που συνδέεται στη γραμμή S7 (η οποία αντιστοιχεί στο bit7 της υφιστάμενης θύρας κατάστασης), οδηγείται απευθείας στη γραμμή εισόδου-εξόδου PB0 του μικροελεγκτή και χρησιμοποιείται ως έξοδος των σειριακών δεδομένων που εκπέμπονται από τον μικροελεγκτή προς τον υπολογιστή. Ο ακροδέκτης 2 του συνδετήρα J2 που συνδέεται στην γραμμή D0(η οποία αντιστοιχεί στο bit0 της υφιστάμενης θύρας δεδομένων), οδηγείται στην γραμμή εισόδου-εξόδου PB0 του μικροελεγκτή μέσω της διόδου D1. Η δίοδος D1 και η αντίσταση R1 σχηματίζουν ένα κύκλωμα πύλης AND μιας εισόδου στο οποίο έχουμε ήδη αναφερθεί. Όταν ο υπολογιστής πρόκειται να λάβει τα δεδομένα από τον μικροελεγκτή του κυκλώματος, οδηγεί την γραμμή D0 σε λογικό ‘1’ και στη συνέχεια ενημερώνει τον μικροελεγκτή ότι μπορεί να δεχθεί δεδομένα.

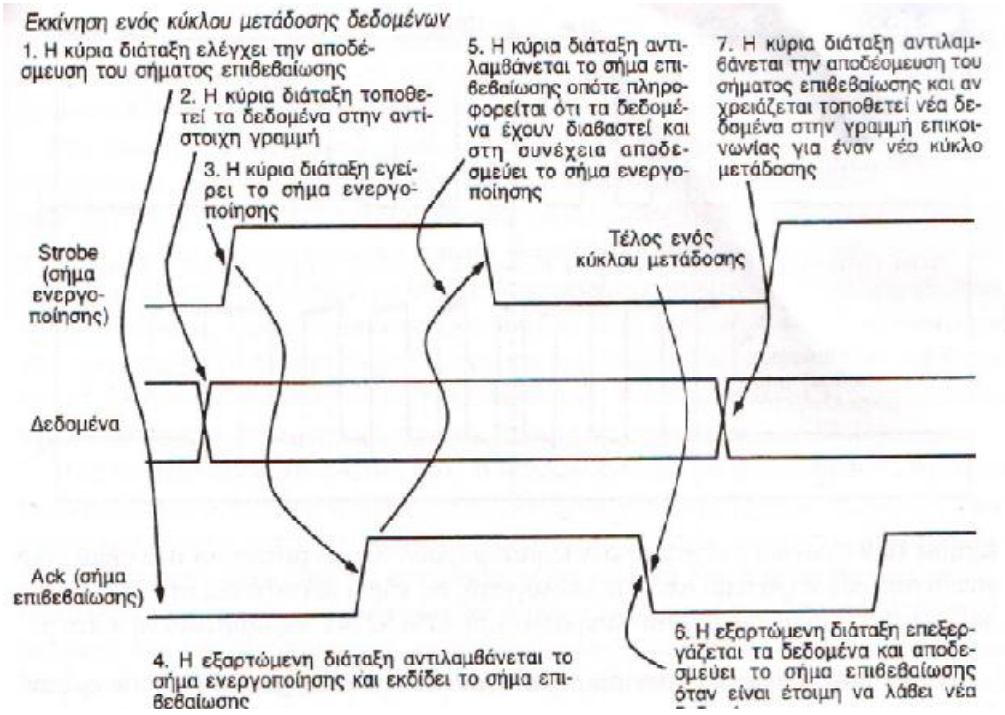
Ο ακροδέκτης 3 του συνδετήρα J2 που συνδέεται στη γραμμή S6 (η οποία αντιστοιχεί στο bit1 της υφιστάμενης θύρας δεδομένων) αποτελεί μια γραμμή εξόδου από την παράλληλη θύρα και χρησιμοποιείται για την έκδοση του σήματος ενεργοποίησης(strobe). Η γραμμή αυτή συνδέεται με τον ακροδέκτη εισόδου-εξόδου PB1 του μικροελεγκτή.

Τελος, ο ακροδέκτης 10 του συνδετήρα J2 που συνδέεται στη γραμμή S6 (η οποία αντιστοιχεί στο bit6 της υφιστάμενης θύρας κατάστασης) αποτελεί μια γραμμή εισόδου της παράλληλης θύρας και χρησιμοποιείται για την λήψη του σήματος επιβεβαίωσης(acknowledge) από τον ακροδέκτη PB2 του μικροελεγκτή.

Στη συνέχεια θα εξετάσουμε τη δυνατότητα ανταλλαγής δεδομένων του κτκλώματος. Η σειριακή μετάδοση δεδομένων πραγματοποιείται με την εκπομπή

λέξεων των 8-bits. Πρόκειται για μια σύγχρονη σειριακή μετάδοση δεδομένων όπου το σήμα ενεργοποίησης παίζει τον ρόλο του σήματος χρονισμού κστ΄σ την εκμπομπή δεδομένων από τον υπολογιστή προς τον μικροελεγκτή του κυκλώματος, ενώ όταν ο μικροελεγκτής εκπέμπει δεδομένα προς τον υπολογιστή, τον ρόλο του σήματος χρονισμού αναλαμβάνει το σήμα επιβεβαίωσης.

Οποιαδήποτε μορφή μετάδοσης δεδομένων μεταξύ δύο διατάξεων θα πρέπει να εξασφαλίσει μια αξιόπιστη ανταλλαγή δεδομένων χωρίς το ενδεχόμενο απωλειών.Για τον λόγο αυτό χρησιμοποιούνται τα λεγόμενα σήματα χειραψίας (handshake signals). Τα σήματα αυτά καλούνται αντίστοιχα, σήμα ενεργοποίησης(Strobe signal) και σήμα επιβεβαίωσης (Acknowledge signal). Στο σχήμα παρουσιάζεται η μορφή με την οποία χρησιμοποιούνται τα σήματα αυτά με σκοπό την ανταλλαγή δεδομένων μεταξύ μιας κύριας (Master) και μιας εξαρτώμενης διάταξης (Slave). Μία από τις διατάξεις αυτές καλείται Κύρια διάταξη, εφόσον επιβάλλει όλες τις απαραίτητες αρχικές συνθήκες της αντίστοιχης μετάδοσης. Η δεύτερη διάταξη καλείται Εξαρτώμενη Διάταξη, εφόσον η λειτουργία της εξαρτάται από την παραπάνω κύρια διάταξη. Στο σχήμα παρουσιάζεται η διαδικασία ανταλλαγής πληροφοριών πάνω σε έναν δίαυλο δεδομένων, ωστόσο η λογική αλληλουχία ισχύει και στην περίπτωση ανταλλαγής δεδομένων σε επίπεδο ενός απλού bit μεταξύ κύριας και εξαρτώμενης διάταξης.



Σχήμα 5.8: Διαδικασία μετάδοσης δεδεομένων μεταξύ μιας κύριας και μιας εξαρτώμενης διάταξης με τη βοήθεια των σημάτων χειραψίας Ack και Strobe (επιβεβαίωσης και ενεργοποίησης αντίστοιχα).

Η ίδια διαδικασία χρησιμοποιείται και από την κύρια διάταξη όταν πρόκειται να λάβει δεδομένα από την αντίστοιχη εξαρτώμενη διάταξη. Στην περίπτωση αυτή, η κύρια διάταξη πρικειμένου να λάβει δεδομένα, ελέγχει κατά πόσο έχει αποδεσμεύσει το σήμα επιβεβαίωσης και στη συνέχεια εκδίδει το σήμα ενεργοποίησης. Η εξαρτώμενη διάταξη ανταποκρίνεται τοποθετώντας τα δεδομένα στην αντίστοιχη γραμμή και στη συνέχεια εκδίδει το σήμα επιβεβαίωσης. Το σήμα αυτό πληροφορεί την κύρια διάταξη ότι τα δεδομένα είναι έτοιμα προς ανάγνωση. Η κύρια διάταξη διαβάζει τα δεδομένα και αμέσως μετά αποδεσμεύει το σήμα ενεργοποίησης. Όταν η εξαρτώμενη διάταξη αντιληφθεί την αποδέσμευση του σήματος αυτού πληροφορείται ότι η κύρια διάταξη έχει ήδη λάβει τα δεδομένα, οπότε μπορεί να αποδεσμεύσει το αντίστοιχο σήμα επιβεβαίωσης. Η διαδικασία που περιγράφηκε αντιστοιχεί σε έναν πλήρη κύκλο μετάδοσης δεδομένων μεταξύ μίας κύριας και μίας εξαρτώμενης διάταξης.

2^{Η} ΕΦΑΡΜΟΓΗ: ΣΥΣΤΗΜΑ ΧΕΙΡΙΣΜΟΥ ΤΟΥ ΚΩΔΙΚΑ MORSE

5.2.1 Εισαγωγή

Ένας μεγάλος αριθμός ραδιοερασιτεχνών (radio amateur) σε όλο τον κόσμο κάνει χρήση του γνωστού κώδικα Morse στις διάφορες επικοινωνίες. Ο κώδικας Morse αποτελεί ένα διεθνές σύστημα σημάτων που χρησιμοποιούνται στην ραδιο-τηλε-γραφία σε συστήματα επίγειων επικοινωνιών σε όλες τις χώρες.

Για το λόγο αυτό, οι ενδιαφερόμενοι θα πρέπει να εξοικειωθούν με τον κώδικα Morse. Ο κώδικας Morse αποτελεί ένα τυποποιημένο σύστημα σημάτων στο οποίο χρησιμοποιούνται κατάλληλοι ηχητικοί τόνοι. Η αρχή του κώδικα αυτού βασίζεται στην διάρκεια του εκάστοτε ηχητικού τόνου με την οποία κωδικοποιείται οποιοδήποτε σύμβολο. Στον κώδικα Morse ορίζονται δύο τύποι διάρκειας τόνου. Έτσι έχουμε τον βραχύ καί τον μακρύ τόνο. Η διάρκεια του μακριού τόνου είναι τριπλάσια από την αντίστοιχη διάρκεια του βραχέως τόνου. Η τιμή της χρονικής διάρκειας του ίδιου του τόνου είναι μεταβλητή και καθορίζει ουσιαστικά την ταχύτητα μετάδοσης των μηνυμάτων. Τα διάφορα σύμβολα που κωδικοποιούνται κατά Morse, αποτελούνται από μία ακολουθία βραχέων και μακρών τόνων. Ακόμη, οι διάφοροι τόνοι που συνθέτουν ένα σύμβολο χωρίζονται μεταξύ τους με ένα χρονικό διάστημα παύσης (δηλαδή απουσίας τόνου). Τα διαστήματα παύσης διαρκούν τον ίδιο χρόνο με εκείνον των βραχέων τόνων. Τα σύμβολα που χωρίζονται με ένα διάστημα απουσίας τόνου, ισοδυναμούν με αυτά που αντιστοιχούν σε έναν μακρύ τόνο. Τα σύμβολα αυτά καλούνται ‘’τελείες’’ και ‘’παύλες΄΄. Αν η διάρκεια μιας τελείας ισούται με Τ, τότε η διάρκεια μιας παύλας ισούται με 3Τ.

Η τυποποιημένη μορφή ενός κώδικα Morse, η οποία είναι αποδεκτή σε οποιαδήποτε χώρα, καλείται Διεθνής Κώδικας Morse. Για την εκμάθηση του κώδικα Morse θα πρέπει να εξασκηθούμε στο άκουσμα των τόνων που αντιστοιχούν σε κάθε σύμβολο και να τους απομνημονεύουμε. Ένας από τους τρόπους εκμάθησης του κώδικα Morse είναι να προμηθευτούμε ηχογραφημένες κασέτες στις οποίες ακούγονται με την μορφή τόνων, όλα τα σύμβολα του κώδικα. Θα πρέπει να ακούσουμε πολλές φορές τους ήχους αυτούς που συνδέονται με τα σύμβολα, έτσι

ώστε να τους απομνημονεύσουμε. Το επόμενο βήμα είναι να χειριστούμε εμείς οι ίδιοι τον κώδικα Morse πριν τον χρησιμοποιήσουμε σε κάποια ραδιοεπικοινωνία.

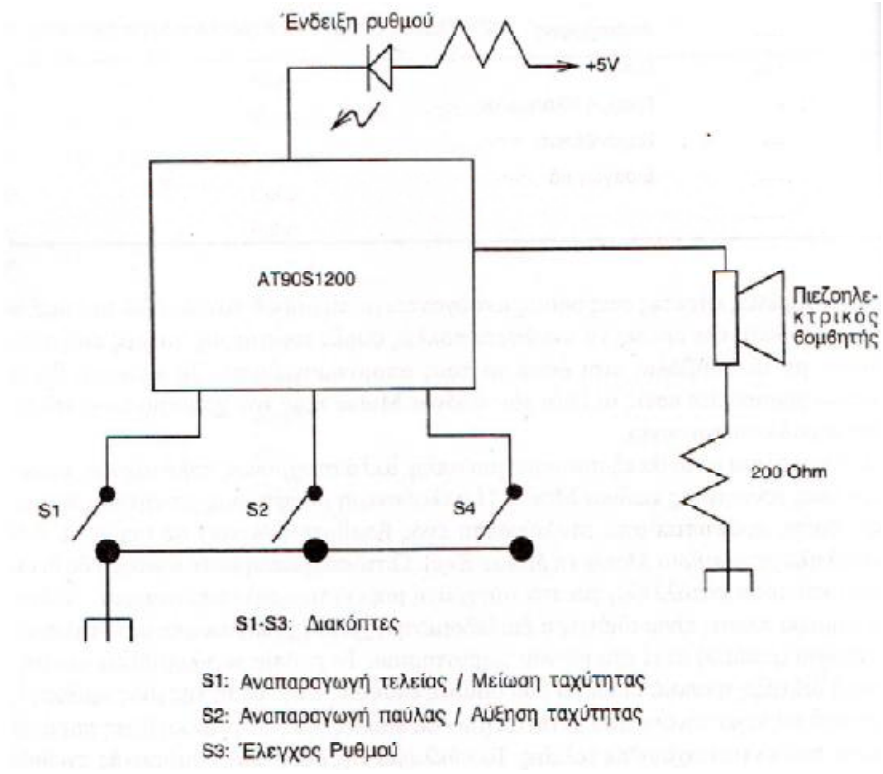
Στο κεφάλαιο αυτό θα εξετάσουμε μια απλή, αλλά συγχρόνως πολύ ισχυρή, εφαρμογή μιας γεννήτριας κώδικα Morse. Η απλούστερη μορφή μιας γεννήτριας σημάτων Morse, προκύπτει από την σύνδεση ενός βομβητή (Buzzer) σε σειρά με ένα κατάλληλο χειριστήριο Morse (ή Morse Key). Ωστόσο η απλή αυτή διάταξη δε θεωρείται και τόσο κατάλληλη για την σύγχρονη μορφή των τηλεπικοινωνιών. Άλλωστε σήμερα πλέον, είναι ιδιαίτερα διαδεδομένη η χρήση του ηλεκτρονικού τηλεχειριστηρίου (paddle) αντί του κοινού χειριστηρίου. Το paddle περιλαμβάνει μια μηχανική διάταξη η οποία διαθέτει δύο ομάδες επαφών. Στη θέση της μιας ομάδας η συσκεύη παράγει τόνους που αντιστοιχούν σε παύλες, ενώ στην άλλη θέση παράγει τόνους που αντιστοιχούν σε τελείες. Το κύκλωμα της δική μας εφαρμογής σχεδιάστηκε για να λειτουργεί όπως μια συσκεύη paddle.

5.2.2 Προδιαγραφές της Σχεδίασης

Για την εφαρμογή αυτή χρειαζόμαστε ένα απλό κύκλωμα το οποίο να μπορεί να συνδεθεί με μια συσκευή paddle, εφόσον είναι διαθέσιμη, ή ακόμη και με έναν κοινό πιεστικό διακόπτη για την παραγωγή των σημάτων παύλας και τελείας του κώδικα Morse. Επίσης, όπως πάντα είναι επιθυμητό, η κατανάλωση ρεύματος θα πρέπει να διατηρείται σε χαμηλά επίπεδα έτσι ώστε να εξασφαλίζεται μεγαλύτερη διάρκεια ζωής της μπαταρίας. Τέλος, θεωρείται απαραίτητη η δυνατότητα ελέγχου της ταχύτητας παραγωγής των σημάτων Morse, καθώς επίσης ιδιαίτερα σημαντικό θεωρείται να διατηρήσουμε το μέγεθος του κυκλώματος όσο το δυνατό μικρότερο για να είναι εύχρηστο ως φορητό εξάρτημα.

5.2.3 Λειτουργική Περιγραφή

Για να ικανοποιήσουμε όλες τις παραπάνω προδιαγραφές, καταλήξαμε σε ένα κύκλωμα βασισμένο στον μικροεγκτή AT90S1200. Στο Σχήμα που ακολουθεί φαίνεται το λειτουργικό διάγραμμα του κυκλώματος χειρισμού του κώδικα Morse. Πρόκειται για ένα πολύ απλό κύκλωμα για το οποίο απαιτούνται μερικοί πιεστικοί διακόπτες, μια δίοδος LED, ένας βομβητής και μερικά άλλα εξαρτήματα, εκτός βέβαια και του ίδιου του μικροελεγκτή AVR.



Σχήμα5.9: Λειτουργικό διάγραμμα του κυκλώματος χειρισμού του κώδικά Morse.

Ο λόγος για τον οποίο αποφασίσαμε να χρησιμοποιήσουμε τον ελεγκτή AT90S1200, τον απλούστερο ελεγκτή της οικογένειας AVR, βασίστηκε στο γεγονός ότι η συγκεκριμένη εφαρμογή δεν απαιτεί περισσότερες δυνατότητες από αυτές που μπορεί ήδη να προσφέρει ο μικροελεγκτής AT90S1200.

Στο παρακάτω Σχήμα φαίνεται το αναλυτικό διάγραμμα του κυκλώματος χειρισμού του κώδικα Morse. Ένας μικρός πιεζοηλεκτρικός βομβητής χρησιμοποιείται για την αναπαραγωγή των ηχητικών τόνων. Το κύκλωμα λειτουργεί με συχνότητα χρονισμού 4 MHz, που παράγεται με τη βοήθεια ενός εξωτερικού κρυστάλλου χαλαζία. Για να μειώσουμε τον αριθμό των απαραίτητων εξαρτημάτων του κυκλώματος μπορούμε να ενεργοποιήσουμε τον εσωτερικό ταλαντωτή τύπου RC του μικροελεγκτή 1200. Για να χρησιμοποιήσουμε έναν κοινό μικροελεγκτή 1200 με τον εσωτερικό ταλαντωτή τύπου RC, θα πρέπει πρώτα να ενεργοποιήσουμε τον ταλαντωτή αυτό θέτοντας το bit ελέγχου RCEN, σε λογικό ‘0’. Ο τύπος του ολοκληρωμένου αυτού διατίθεται με το bit RCEN τοποθετημένο σε λογικό΄1΄, οπότε για να μηδενίσουμε χρειάζεται να εκτελέσουμε αποκλειστικά την διαδικασία του παράλληλου προγραμματισμού. Βέβαια, χρησιμοποιώντας την έκδοση AT90S1200A του ίδιου μικροελεγκτή, στην οποία το bit RCEN είναι μηδενισμένο, ο εσωτερικός

ταλαντωτής τύπου RC είναι ενεργοποιημένος και μπορεί να χρησιμοποιηθεί άμεσα ως πηγή χρονισμού του μικροελεγκτή.

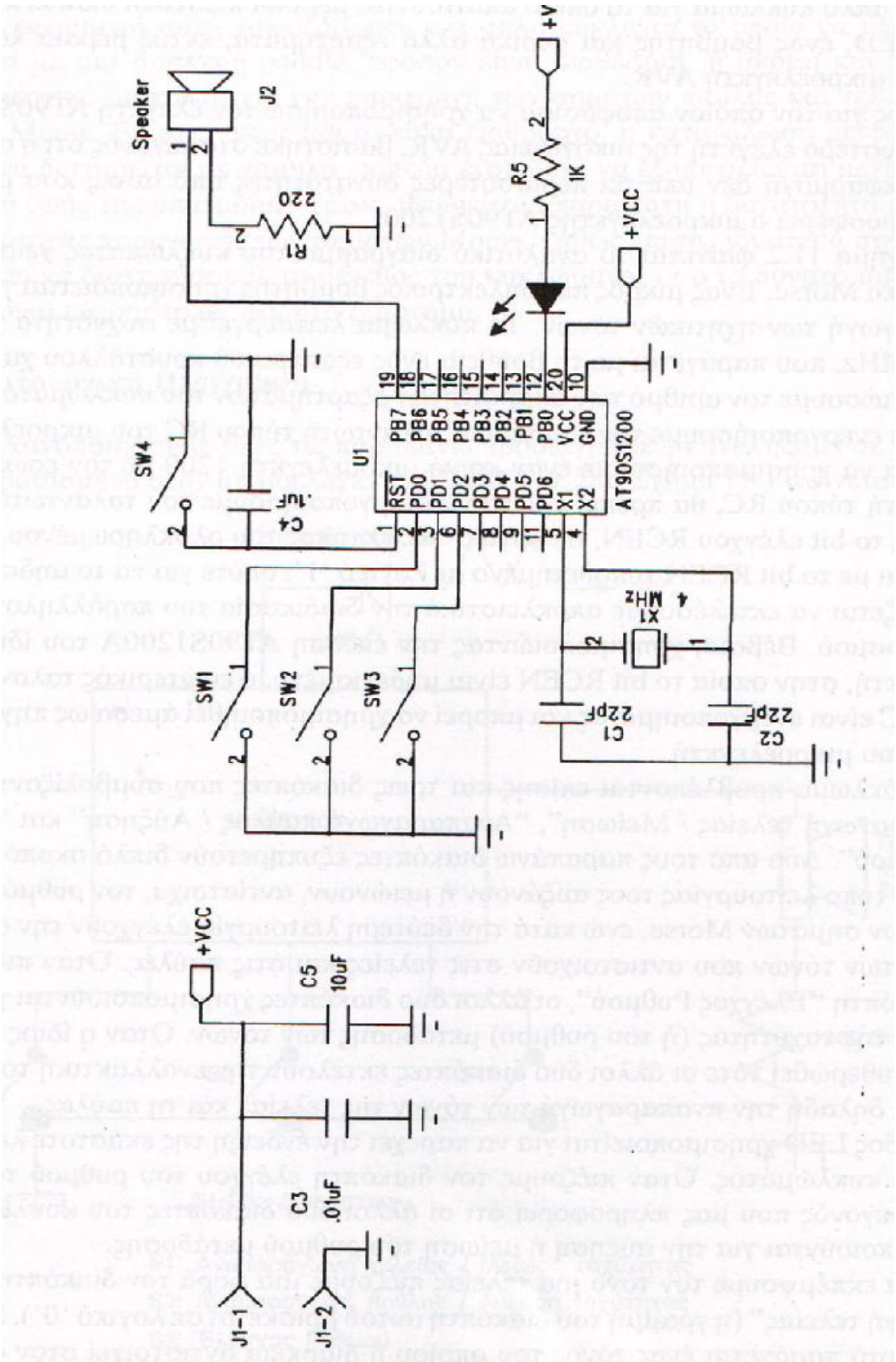
Στο κύκλωμα προβλέπονται επίσης και τρεις διακόπτες που συμβολίζονται ως, ‘’Αναπαραγωγή τελείας / Μείωση’’, ΄΄Αναπαραγωγή παύλας / Αύξηση’’ και ΄΄Έλεγχος Ρυθμού’’. Δυο από τους παραπάνω διακόπτες εξυπηρετούν διπλό σκοπό. Κατά τον ένα τύπο λειτουργίας τους αυξάνουν ή μειώνουν, αντίστοιχα, τον ρυθμό παραγωγής των σημάτων Morse, ενώ κατά την δεύτερη λειτουργία ελέγχουν την αναπαραγωγή των τόνων που αντιστοιχούν στις τελείες και τις παύλες. Όταν πιέζουμε τον διακόπτη ‘’Έλεγχος Ρυθμού’’, οι άλλοι δύο διακόπτες χρησιμοποιούνται για την ρύθμιση της ταχύτητας (ή του ρυθμού) μετάδοσης των τόνων. Όταν ο ίδιος διακόπτης ελευθερωθεί τότε οι άλλοι δύο διακόπτες εκτελούν την εναλλακτική τους λειτουργία, δηλαδή την αναπαραγωγή των τόνων της τελείας και της παύλας.

Η δίοδος LED χρησιμοποιείται για να παρέχει την ένδειξη της εκάστοτε λειτουργίας του κυκλώματος. Όταν πιέζουμε τον διακόπτη ελέγχου του ρυθμού το LED ανάβει γεγονός που μας πληροφορεί ότι οι άλλοι δύο διακόπτες του κυκλώματος χρησιμοποιούνται για την αύξηση ή μείωση του ρυθμού μετάδοσης. Για να εκπέμψουμε τον τόνο μιας τελείας πιέζουμε μία φορά τον διακόπτη «αναπαραγωγή τελείας» (η γραμμή του διακόπτη αυτού βρίσκεται σε λογικό ‘0’). Με τον τρόπο αυτό παράγεται ένας τόνος του οποίου η διάρκεια αντιστοιχεί στον ορισμό της τελείας. Κρατώντας συνεχώς πατημένο τον ίδιο διακόπτη παράγονται διαδοχικοί τόνοι τελείας μεταξύ των οποίων παρεμβάλλονται διαστήματα σιγής της ίδιας χρονικής διάρκειας του ήχου της τελείας. Παρόμοια, για την παραγωγή του τόνου μιας παύλας πιέζουμε μια φορά τον διακόπτη παράγονται διαδιχικοί τόνοι παύλας, μεταξύ των οποίων παρεμβάλλονται διαστήματα σιγής της ίδιας χρονικής διάρκειας του ήχου της τελείας.

Με κατάλληλο χειρισμό των δύο διακοπτών παράγονται ακολουθίες τόνων για κάθε διακόπτη. Το κύριο τμήμα του προγράμματος της εφαρμογής αφορά στον έλεγχο του Timer0, με τη βοήθεια της ρουτίνας διακοπής του οποίου παράγεται ο ζητούμενος ήχος από τις διαδοχικές αλλαγές της λογικής κατάστασης του ακροδέκτη PB7. Στον ακροδέκτη αυτόν συνδέεται ο πιεζοηλεκτρικός βομβητής. Ο χρονιστής Timer0 προγραμματίζεται έτσι ώστε να παράγει μια διακοπή κάθε 1.12ms (δηλαδή μια συχνότητα διακοπών ίση με 892 Hz). Καθε φορά που εκδηλώνεται η

συγκεκριμένη διακοπή αλλάζει η λογική κατάσταση του ακροδέκτη PB7(toggling), οπότε το σήμα που προκύπτει συνολικά στην έξοδο αυτή, έχει συχνότητα 446Hz. Η ταχύτητα αναπαραγωγής του κώδικα Morse πραγματοποιείται με την μέτρηση του πλήθους των τόνων που παράγονται. Για τον σκοπό της μέτρησης αυτής χρησιμοποιείται ένα τμήμα του προγράμματος στο οποίο επιτελείται η απαρίθμηση. Αν το πλήθος των παλμών που αντιστοιχούν στον ήχο μιας τελείας (ή μιας παύλας) ισούται με τον παραπάνω αριθμό, τότε η αναπαραγωγή ήχου απενεργοποιείται για ένα χρονικό διάστημα ίσο με τον χρόνο διάρκειας του τόνου μιας τελείας. Η ταχύτητα μετάδοσης μπορεί να ρυθμιστεί μεταξύ 5 και 40 λέξεων το λεπτό.

α



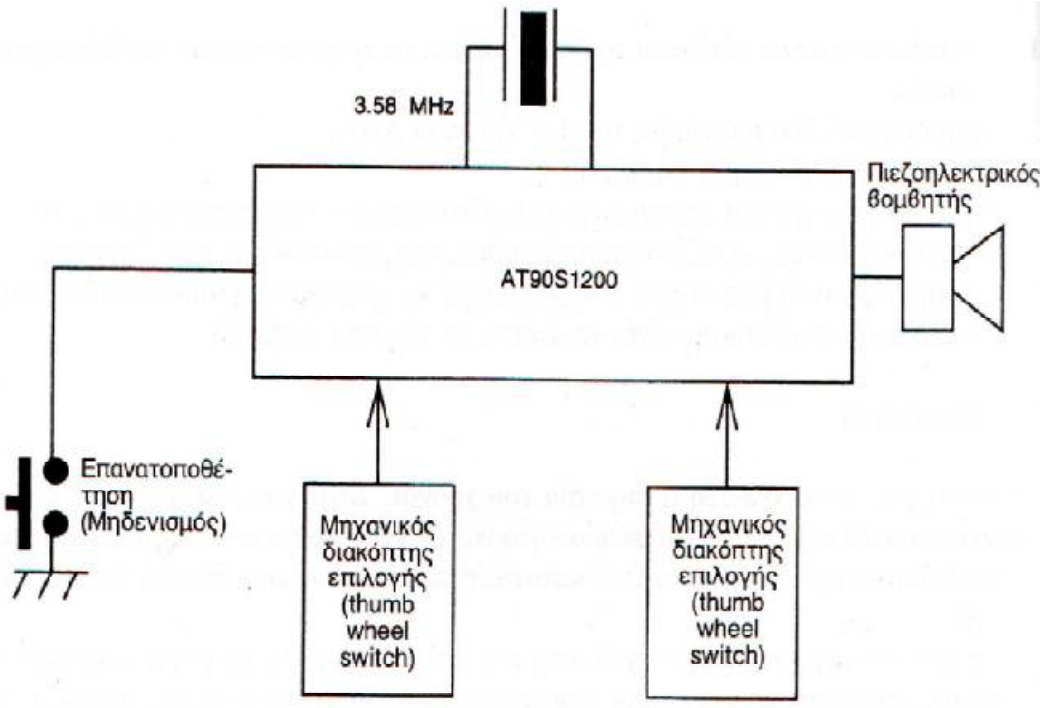
Σχήμα 5.10: Αναλυτικό διάγραμμα του κυκλώματος χειρισμού του κώδικα Morse

3Η ΕΦΑΡΜΟΓΗ: ΧΡΟΝΟΔΙΑΚΌΠΤΗΣ ΚΟΥΖΙΝΑΣ

5.3.1 Εισαγωγή

Σε όλους μας είναι γνωστή η σημασία του χρόνου. Στην κουζίνα ο χρόνος παίζει έναν σημαντικό ρόλο και είναι απαραίτητο να τον γνωρίζουμε ανά πάσα στιγμή έτσι ώστε να αποφεύγουμε δυσάρεστες καταστάσεις, όπως για παράδειγμα ένα καμμένο φαγητό. Για το λόγο αυτόν έχουμε σχεδιάσει ένα απλό αλλά πολύ χρήσιμο κύκλωμα το οποίο μας επιτρέπει να ελέγχουμε τον χρόνο. Εκτός από την κουζίνα, το κύκλωμα της εφαρμογής αυτής μπορούμε να το χρησιμοποιήσουμε και ως ένα ξυπνητήρι.

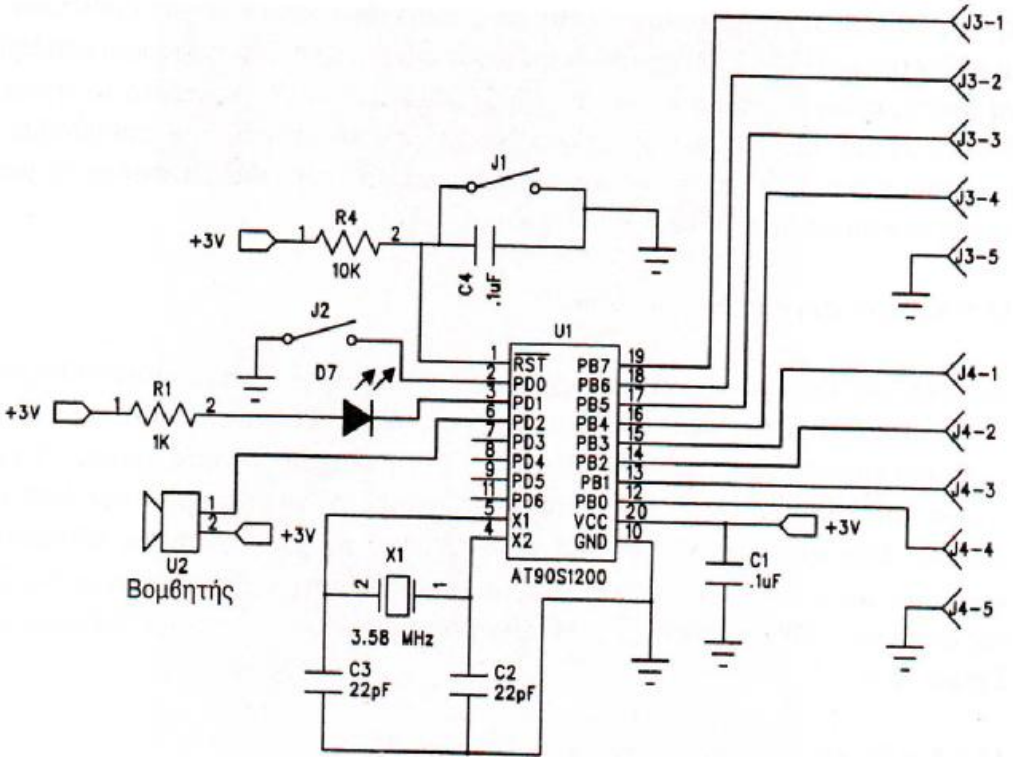
Η απλή αυτή εφαρμογή βασίζεται στη μονάδα του ενσωματωμένου χρονιστή ενός μικροελεγκτή. Πρόκειται για μια εξαιρετικά απλή εφαρμογή. Στο σχήμα που ακολουθεί, μπορούμε να δούμε το λειτουργικό διάγραμμα του κυκλώματος της εφαρμογής. Με τη βοήθεια μηχανικών διακοπτών επιλογής (thumb wheel switches) εισάγουμε το επιθυμητό χρονικό διάστημα σε μονάδες λεπτών. Εφόσων ρυθμίσουμε τον χρόνο μηδενίζουμε τον χρονοδιακόπτη, ο οποίος αμέσως μετά ξεκινά από την αρχή. Όταν παρέλθει το προκαθορισμένο χρονικό διάστημα, το κύκλωμα ενεργοποιεί έναν βομβητή.



Σχήμα 5.11: Λειτουργικό διάγραμμα ενός απλού χρονοδιακόπτη κουζίνας.

5.3.2 Λειτουργική Περιγραφή

Η λειτουργία του χρονοδιακόπτη είναι πολύ απλή. Στο σχήμα, μπορούμε να δούμε το αναλυτικό διάγραμμα του αντίστοιχου κυκλώματος. Επιπλέον, για την κατασκευή του κυκλώματος αυτού δεν απαιτούνται παρά μόνον ελάχιστα εξαρτήματα. Για να διατηρήσουμε την απλότητα της κατασκευής χρησιμοποιήσαμε απλούς μηχανικούς διακόπτες επιλογής για την ρύθμιση του χρόνου. Ωστόσο μπορούμε να προτείνουμε και μια πιο κομψή λύση κατά την οποία συνδυάζουμε ένα πληκτρολόγιο και μια οθόνη LCD, γεγονός όμως που οδηγεί σε ένα κύκλωμα μεγαλύτερων διαστάσεων ενώ ταυτόχρονα καταναλώνει και παρισσότερο ρεύμα.



Σχήμα 5.12: Αναλυτικό διάγραμμα του κυκλώματος του χρονοδιακόπτη κουζίνας.

Οι μηχανικοί διακόπτες συνδέονται στους ακροδέκτες της θύρας PORTB. Κάθε ένας από τους διακόπτες αυτούς απαιτεί για την σύνδεση του στο κύκλωμα τέσσερις ακροδέκτες. Έτσι με δύο μηχανικούς διακόπτες καταλαμβάνεται ολόκληρη η θύρα PORTB. Η θύρα αυτή προγραμματίζεται για να λειτουργήσει αποκλειστικά ως θύρα εισόδου με ενεργοποιημένες τις αντίστοιχες εσωτερικές αντιστάσεις πρόσδεσης. Αυτό σημαίνει βέβαια ότι δεν απαιτούνται εξωτερικές αντιστάσεις.

Η δίοδος LED και ο πιεζοηλεκτρικός βομβητής συνδέονται στους ακροδέκτες της θύρας PORTB. Συγκεκριμένα, η δίοδος LED συνδέεται έτσι ώστε να απορροφά ρεύμα από το αντίστοιχο κύκλωμα του ακροδέκτη PD1. Ο βομβητής συνδέεται στον ακροδέκτη PD2, επίσης με τέτοιον τρόπο ώστε να απορροφά ρεύμα από το αντίστοιχο κύκλωμα εξόδου. Το κύκλωμα του χρονοδιακόπτη λειτουργεί με συχνότητα χρονισμού ίση με 3.58MHz. Στη συγκεκριμένη εφαρμογή δεν μπορούμε να χρησιμοποιήσουμε τον εσωτερικό ταλαντωτή τύπου RC αντί του εξωτερικού κρυστάλλου, διότι η συχνότητα του ταλαντωτή RC εξαρτάται από την τιμή της τάσης τροφοδοσίας γεγονός που καθιστά μη αποδεκτή τη λειτουργία του χρονοδιακόπτη. Σε διάφορες άλλες εφαρμογές όπου τυχόν διακυμάνσεις της συχνότητας του ταλαντωτή δε θεωρούνται κρίσιμες, η παραπάνω λύση αποδεικνύεται ιδιαίτερα ενδιαφέρουσα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. *Επιστημονικό Λεξικό Πληροφορικής*, Κλειδάριθμος, Αθήνα, 1998
2. Harvey M. Deitel, *C++ How to Program: how to program*, Pearson Prentice.
3. Gadre, Dhananjay V. *Programming and customizing the AVR microcontroller*, McGraw-Hill, 2000.
4. Giovino, Bill. *Microcontrollers and DSPs – Will They Converge?*, www.microcontroller.com
5. Ken, Arnold. *Embedded controller hardware design*, Newnes, 2001. Melear, Charles, Kilbane, Kevin. *Problems solved by FLASH microcontrollers*, *www.microcontroller.com (Αγγλικά).*
6. Κόγιας, Γεωργίου Δ. *Εισαγωγή στους Μικροεπεξεργαστές*, Εκόσεις Σύγχρονη Εκδοτική (Β' Έκδοση), Αθήνα 1994.
7. Κουζανίδης Αντώνιος, *Δημιουργία Εκπαιδευτικών Ασκήσεων και Εφαρμογών στον Μικροελεγκτή AVR*, Αθήνα 2005.
8. Melear, Charles, Kilbane, Kevin. *Problems solved by FLASH microcontrollers*, *www.microcontroller.com (Αγγλικά).*
9. Πεσμετζή Π., *Συστήματα Μικροϋπολογιστών I (1995) &II*, Εκδόσεις Συμμετρία, Αθήνα 2009.
10. Ritchie Dennis & Brian Kernighan *Η γλώσσα προγραμματισμού C*, Εκδόσεις Κλειδάριθμος.
11. Stallings William, *Οργάνωση & Αρχιτεκτονική Υπολογιστών*, Εκδόσεις Τζιόλα, 2003
12. Σιδερίδης Α.Β., *Μικροϋπολογιστές*, Αθήνα 1996
13. Τσουρόπλης Αθ. & Κ. Κλημόπουλος, *Εισαγωγή στην Πληροφορική*, Εκδόσεις Νέων Τεχνολογιών, 2005
14. Tanenbaum Andrew S., *Σύγχρονα Λειτουργικά Συστήματα*, Εκδόσεις Κλειδάριθμος, 2007
15. http://www.atmel.com
16. http://www.google.com
17. http://www.mikrocontroller.net/articles/AVR
18. http://de.wikipedia.org/wiki/Atmel_AVR
19. www.lis.upatras.gr