Ανώτατο Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Ηλεκτρολογίας

Θέμα: Έλεγχος και προγραμματισμός Μικροϋπολογιστών Atmel

Εισηγητής: Καγιαμπάκης Μανώλης

Σπουδαστές: Αλεβυζάκης Αριστοτέλης
Μαράκης Γιάννης

Ηράκλειο 2007

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Ο σκοπός του βιβλίου αυτού είναι να κατανοήσουν οι μαθητές τον τρόπο που λειτουργούν οι μικροελεγκτές, συγκεκριμένα ο μικροελεγκτής της Atmel AT90S8515 της οικογένειας AVR επίσης να κατανοήσουν τον τρόπο με τον οποίο προγραμματίζονται.

Στην σημερινή εποχή όλες οι ηλεκτρονικές συσκευές έχουν σαν βασικό εξάρτημα τους έναν μικροελεγκτή ο οποίος προγραμματίζεται και αναβαθμίζεται με την βοήθεια ενός ηλεκτρονικού υπολογιστή,ο οποίος διαθέτει θύρες κατάλληλες για την επικοινωνία του με τον μικροελεγκτή.

Ο μικροελεγκτής που θα περιγράψουμε σε αυτό το βιβλίο είναι ο AT90S8515 της Atmel με τον οποίο έχουν κατασκευαστεί οι αναπτυξιακές πλακέτες που είναι τοποθετημένες πάνω σε board για να γίνει η εξομοίωση των προγραμμάτων που υπάρχουν έτοιμα στο τέλος του βιβλίου.

Το βιβλίο αυτό είναι γραμμένο με απλό τρόπο ώστε να είναι κατανοητό από τους μαθητές.

Ευχαριστούμε για την
πολύτιμη βοήθεια, τους
συναδέλφους
Μισαργόπουλο Ιωάννη
Ανυφαντάκη Γεωργιο

1. ΕΙΣΑΓΩΓΗ

1.1 Ιστορία των μικροεπεξεργαστών

Τα τελευταία είκοσι χρονιά η επανάσταση στον τομές υπολογιστών συντέλεσε στην παραγωγή νέας γενιάς υπολογιστών που έχουν ταχύτητες και υπολογιστική ισχύ χιλιάδες φορές μεγαλύτερη από εκείνη των πρώτων εμπορικών υπολογιστων.αυτο ήταν αποτέλεσμα των ολοκληρωμένων κυκλωμάτων που συνδυάζουμε ένα μεγάλο αριθμό δυνατοτήτων πάνω σε μια φέτα πυριτίου(chip),ειδικότερα στην δημιουργία και αλματώδη εξέλιξη των μικροεπεξεργαστών (microprocessors).

Επεξεργαστής ονομάζεται γενικά ένα προγραμματιζόμενο ψηφιακό ολοκληρωμένο κυκλωμα,το οποίο αφού δεχτεί κάποια δεδομένα και ένα πρόγραμμα έχει την δυνατότητα να τα χρησιμοποιήσει για να εκτελέσει κάποιες εντολές ή λειτουργίες και να οδηγήσει στην λύση ενός προβλήματος .Με την λογική αυτή ,επεξεργαστές είχαν εφευρεθεί από τον 17^{ο} αιώνα ,όταν ο Γάλλος Blassé Pascal κατασκεύασε μια συσκευή που έκανε αφαιρέσεις και προσθέσεις βασιζομένη σε δέκα τροχούς .Όταν ο ένας τροχός έκανε ένα πλήρη κύκλο φτάνοντας από το μηδέν στο εννέα τότε ο αμέσως επόμενος τροχός άρχισε να γυρνά.

Το επόμενο βήμα έγινε από τον Gottfried Wilhelm Leibniz ο οποίος το 1694 κατασκεύασε ένα μηχανικό υπολογιστή που μπορούσε να κάνει πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση μέθοδος του συγκεκριμένου υπολογιστή χρησιμοποιήθηκε για αρκετά μετά χρονιά μέχρι τους πρώτους Η/Υ.

Το πιο σημαντικό όμως βήμα στους ηλεκτρονικούς επεξεργαστές έγινε από τον Άγγλο μαθηματικό Charles Babbage ο οποίος σχεδίασε την διαφορική και την αναλυτική μηχανή όπως τις ονόμασε. Διαφορική μηχανή είχε σκοπό να λύνει πολυωνυμίες εξισώσεις και να τυπώνει αυτόματα το αποτέλεσμα αναλυτική μηχανή από την άλλη είχε σκοπό να εκτελεί κάθε μαθηματική πράξη και δεν ήταν τίποτα άλλο από μηχανικό επεξεργαστή που έπρεπε να προγραμματιστεί μηχανή είχε «μαθηματικό συνεπεξεργάστη»,μνήμη καθώς και εισόδους και εξόδους. Δυστυχώς ποτέ δεν κατασκευάστηκε. Την σκυτάλη πηρέ η Adam Gordon ή Adam King,η οποία ενδιαφέρθηκε για την αναλυτική μηχανή. Στην συνεχεία η Adam King συνεργάστηκε με τον ίδιο τον Babbage και έφτασε σε σημαντικά συμπεράσματα, όπως ότι πολλές πράξεις επαναλαμβάνονται πολλές φορές κατά την διάρκεια ενός μαθηματικού προβλήματος και προγράμματος.

1.2 Τα μικροϋπολογιστικά συστήματα σήμερα

Αν συγκρίνει κανείς τις δυνατότητες των μικροεπεξεργαστών ή μικροελεγκτών που χρησιμοποιούνται σε μικροϋπολογιστικά συστήματα με τις αντίστοιχες των επεξεργαστών των μεγάλων συστημάτων (mainframes) ή ακόμα και των προσωπικών υπολογιστών (PC), η διαφορά είναι μεγάλη σε πολλά επίπεδα. Οι ταχύτητες λειτουργίας είναι χαμηλότερες, τα μεγέθη μνήμης που μπορούν να διαχειριστούν πολύ περιορισμένα κλπ. Όμως μια τέτοια σύγκριση είναι άδικη δεδομένου ότι δεν προορίζονται για τις ίδιες εφαρμογές. Παρότι πριν από 15-20 χρόνια επεξεργαστές 8-bit χρησιμοποιούνταν ευρέως σε προσωπικούς υπολογιστές, σήμερα η χρήση τους είναι τελείως διαφορετική. Υπάρχει μια σειρά από εφαρμογές ειδικού σκοπού που δεν έχουν την ανάγκη των επιδόσεων των σύγχρονων επεξεργαστών 32-bit. Σαν τέτοιες εφαρμογές μπορεί να αναφέρει κανείς τους ενσωματωμένους μικροϋπολογιστές σε οικιακές συσκευές, συστήματα ασφαλείας, συστήματα ελέγχου αυτοκινήτων, ρομποτικά συστήματα, παραγωγικές μονάδες όπως θερμοκήπια και βιοτεχνίες, όργανα μετρήσεων όπως ιατρικά και ηλεκτρονικά όργανα, μετεωρολογικοί σταθμοί, παιχνιδομηχανές κλπ. Σε τέτοιες εφαρμογές η χρήση ισχυρότερων επεξεργαστών από αυτούς που πραγματικά χρειάζονται προσθέτει όχι μόνο κόστος, που

μπορεί να είναι πολύ κρίσιμο, αλλά και ανεπιθύμητη αύξηση πολυπλοκότητας, κατανάλωσης, διαστάσεων κλπ.
Για το λόγο αυτό διατίθεται σήμερα μια πλειάδα μικροελεγκτών που ενσωματώνουν στο ίδιο ολοκληρωμένο κύκλωμα την ΚΜΕ μαζί με έναν αριθμό περιφερειακών (μνήμη, χρονιστές/μετρητές, ακροδέκτες γενικής χρήσεως , DAC και ADC, σειριακές και παράλληλες Θύρες επικοινωνίας κα). Διαλέγοντας το κατάλληλο μικροελεγκτή για μία εφαρμογή μπορεί να ελαχιστοποιηθεί το πλήθος των απαιτούμενων εξωτερικών εξαρτημάτων. Σε εφαρμογές που οι ανάγκες μνήμης ξεφεύγουν πολύ από τα μεγέθη που διαθέτουν οι ενσωματωμένες μνήμες σε μικροελεγκτές, είναι δυνατή η χρήση κάποιου μικροεπεξεργαστή στους διαύλους του οποίου μπορούν να συνδεθούν μνήμη κατάλληλου μεγέθους και τα απαραίτητα περιφερειακά. Αν ξεκινήσουμε από τα πιο στοιχειώδη τμήματα ενός υπολογιστή, μπορούμε να πούμε ότι ανεξάρτητα από τα μεγέθη όλα τα μικροϋπολογιστικά συστήματα καθώς και οι μεγαλύτεροι υπολογιστές αποτελούνται από τα ίδια βασικά τμήματα: ΚΜΕ, μονάδες I/O, μνήμη, και σύστημα χρονισμού (ρολόι). Η ΚΜΕ διαχειρίζεται πληροφορία σύμφωνα με τις οδηγίες ενός προγράμματος εντολών.
Διαχειρίζεται επίσης ένα πλήθος γραμμών ελέγχου που της δίνουν τη δυνατότητα να ελέγξει τα περιφερειακά και να επικοινωνήσει με τον έξω κόσμο. Μερικά περιφερειακά εισόδου χρειάζεται να μετατρέψουν αναλογικά σήματα σε δυαδικά ψηφία που σε επίπεδο κυκλώματος αντιστοιχούν σε 0 και 5V τάση (π.χ. αισθητήρας Θερμοκρασίας). Άλλες μονάδες εισόδου που στηρίζονται στη χρήση διακοπτών μπορούν να παράσχουν κατευθείαν τις δύο αυτές καταστάσεις όπως ένα πληκτρολόγιο. Τις καταστάσεις αυτές τις δέχεται η ΚΜΕ σαν είσοδο. Στις συσκευές εξόδου η ΚΜΕ αποστέλλει ψηφιακά δεδομένα τα οποία οι συσκευές αυτές μπορούν να μετατρέψουν σε άλλης μορφής σήματα, για να δώσουν στον έξω κόσμο τα αποτελέσματα της επεξεργασίας των δεδομένων εισόδου. Τέτοιες συσκευές μπορεί να είναι οθόνες, beeper, ρελέ κλπ. Ένα υπολογιστικό σύστημα διαθέτει συνήθως περισσότερες από μία μονάδες εισόδου / εξόδου. Υπάρχει συχνά σημαντική διαφορά μεταξύ των μονάδων εισόδου και εξόδου ενός μικροϋπολογιστικού συστήματος και των άλλων υπολογιστών όπως π.χ ενός προσωπικού υπολογιστή. Σε έναν προσωπικό υπολογιστή η βασική μονάδα εισόδου είναι το πληκτρολόγιο και το ποντίκι, ενώ συμπληρωματική είσοδος μπορεί να δοθεί από συσκευές όπως ο σαρωτής , το μικρόφωνο κλπ. Επίσης σε ένα PC η κύρια έξοδος είναι η οθόνη και ο εκτυπωτής. Δεδομένου όμως ότι ένα μικροϋπολογιστικό σύστημα προορίζεται σήμερα κυρίως για εφαρμογές ελέγχου, οι εισόδοι και έξοδοί του είναι σήματα από αισθητήρες (sensors) και ενεργοποιητές , (actuators) ή διακόπτες. Π.χ, σε ένα σύστημα συναγερμού ο ενσωματωμένος μικροεπεξεργαστής δέχεται είσοδο από αισθητήρες υπέρυθρων ακτίνων, θορύβου, καπνού, υγρασίας κλπ, για να ανιχνεύσει αν εισήλθε κάποιος ή αν έχει εκδηλωθεί πυρκαγιά / πλημμύρα στον προστατευόμενο χώρο. Η κύρια έξοδος ενός τέτοιου συστήματος είναι η σειρήνα, ο φάρος και η κλήση τηλεφώνου (dialer). Στα περισσότερα συστήματα συναγερμού υπάρχει ένα στοιχειώδες πληκτρολόγιο (keypad) και μια LCD οθόνη, των οποίων όμως η χρήση είναι περισσότερο βοηθητική.
Όλα τα chip μικροελεγκτών διαθέτουν δυνατότητα σύνδεσης με ένα τερματικό για να μπορούμε να το προγραμματίσουμε. Για τη σύνδεση αυτή απαιτούνται ορισμένα ηλεκτρονικά εξαρτήματα. Η συνήθης διαδικασία προγραμματισμού είναι:

- επιλογή του κατάλληλου μικροελεγκτή ή μικροελεγτική μοναδα ,(μικροελεγκτης με μνήμη RAM και ελαχιστο hardware).
- σύνδεση με έναν ηλεκτρονικό υπολογιστή.
- να τον προγραμματίζουμε να εκτελεί ένα συγκεκριμένο πρόγραμμα και
- να τον τοποθετήσουμε στη συγκεκριμένη εφαρμογή, για να εκτελέσει τη λειτουργία που του έχουμε ορίσει, όπως έλεγχος στροφών κινητήρα, σύστημα πυρασφάλειας, χρονοδιακόπτης και πολλές άλλες εφαρμογές.

Οι Μικροελεγκτές της Intel

Το 1976 η Intel παρουσίασε την οικογένεια MCS -48 με τους πρώτους μικροελεγκτές 8048,8748 και 8035 που περιλαμβάνουν μια CPU των 8 bits,μνήμη προγράμματος ROM 1 K η μνήμη EPROM, μνήμη δεδομένων RAM 64 Kbytes,πόρτες εισόδου-εξόδου και χρονιστή/μετρητή των 8 bits.Όλα τα παραπάνω είναι ενσωματωμένα σε ένα chip των 40 pins.

Πιο εξελιγμένα προϊόντα της Intel,επεκτείνανε την αρχιτεκτονική του MCS-48 σε διάφορες κατευθύνσεις, όπως το 8049,8050 και 8039 που έχουν διπλάσια μνήμη πάνω στο chip και είναι κατά 83 γρηγορότερα. Το 8022 έχει πάνω στο chip έναν A/D Converter 8 bits και μπορεί να συνδεθεί κατευθείαν με αναλογικά αισθητήρια.

Σήμερα οι νέοι μικροελεγκτές υψηλής αποδοτικότητας, επεκτείνουν κατά πολύ τα πλεονεκτήματα της ολοκληρωμένης ηλεκτρονικής. Είναι η σειρά MCS-51 σε τεχνολογία HMOS η HMOS 2 και με τετραπλάσια μνήμη προγράμματος και διπλάσια μνήμη δεδομένων. Επιπλέον, νέες πόρτες εισόδου/εξόδου και δυνατότητες περιφερειακών συνδέσεων, αυξάνουν τον αριθμό των εφαρμογών που μπορούν να πραγματοποιήσουν και παράλληλα μειώνουν το ολικό κόστος, καθώς πολλές λειτουργίες γίνονται με ένα και μόνο chip.

Παράλληλα, η ταχύτητα λειτουργίας(μέχρι και 16 MHz) είναι ανάλογη με τη χρήση από δυόμιση μέχρι πέντε φορές μεγαλύτερη. Το πρώτο μέλος της οικογένειας ο 8051, θεωρείται ο πυρήνας για όλα τα άλλα μέλη και είναι στην παραγωγή από το 1982.

Η οικογένεια MCS-51 των μικροελεγκτών σε ένα ολοκληρωμένο της Intel αποτελείται από τα μέλη που φαίνονται παρακάτω:

Part	On-Chip Program Memory	On-Chip Data Memory	
8031	None	128 Bytes	
80C31	None	128 Bytes	HCMOS
8051	4 K ROM	128 Bytes	HMOS
8051 AH	4 K ROM	128 Bytes	HMOS II
80C51	4 K ROM	128 Bytes	
8751	4 K EPROM	128 Bytes	
8032	None	256 Bytes	
8052	8 K	256 Bytes	
8052 AH	8 K ROM BASIC	256 Bytes	HMOS II
8752	8 K EPROM	256 Bytes	
80C51FA	None	256 Bytes	CHMOS
83C51FA	8 K ROM	256 Bytes	
87C51FA	8 K EPROM	256 Bytes	
80C51FB	None	256 Bytes	CHMOS
83C51FB	16 K ROM	256 Byte	
87C51CB	16 K EPROM	256 Byte	

Όπως φαίνεται πιο πάνω οι 805X διαθέτουν εσωτερική ROM ενώ οι 803X δεν διαθέτουν. Επίσης οι 80X1 διαθέτουν εσωτερική RAM 128 bytes και οι 80X2, 256 bytes

Η Intel διαθέτη επίσης μικροελεγκτές βασισμένους στον 8051, ειδικούς για τηλεπικοινωνιακές εφαρμογές, όπως ο 8044 και ο 83C152. Ακόμα υπάρχει και η σειρά των 16μπιτων MCS-96.

Σημειωτέων ότι και το 8052 AH είναι προγραμματιζόμενο μέσω μάσκας. Τούτο σημαίνει ότι διατίθεται μόνο σε μεγάλες ποσότητες, επειδή η ενσωματωμένη μνήμη προγράμματος (ROM) μπορεί να γραφτεί μόνο από τον κατασκευαστή του chip. Πρόκειται ουσιαστικά για για ένα ισχυρό και γρήγορο μεταφραστή BASIC φορτωμένο στην ενσωματωμένη ROM 8K. Ειδικά αυτό το ολοκληρωμένο έχει ιδιαίτερο ενδιαφέρον για μεμονωμένες εφαρμογές. Το 8031 όμως διαφέρει από το 8051 κατά το ότι δεν έχει ενσωματωμένη ROM. Αντ΄ αυτού, παίρνει τις εντολές του από μια εξωτερική μνήμη και συνεπώς είναι ιδανικό για την εγγραφή και τον έλεγχο προγραμμάτων που προορίζονται για το 8051.

Για την οικογένεια MCS-51 της Intel υπάρχουν και διάφοροι DOS-based compilers σε γλώσσες υψηλού επιπέδου όπως ASM-51, PL/M-51, C-51 και PASCAL που παίρνουν τον πηγαίο κώδικα προγράμματος , τον μεταφράζουν και παράγουν ένα αντικειμενικό κώδικα 8051 (object module format) ικανό να χρησιμοποιηθεί από τις άλλες εφαρμογές .

Σήμερα οι μικροελεγκτές τις Atmel MCS51 κατασκευάζονται από πολλές ακόμα εταιρίες με τον ίδιο υπολογιστικό πυρήνα, μεταξύ τον οποίων είναι η Philips (8XC750,8XC752), η Siemens (SAB80C535),η Dallas (DS 5000) και η Atmen που ενσωμάτωσε τη μνήμη Flash (AT89C51,89C52,889C2051).

Έτσι η οικογένεια 803X/805X έχει σήμερα το μεγαλύτερο πλήθος μελών.

Μικροελεγκτές της **SGS-THOMSON Microelectronics(ST2), (ST6)** διαθέτη διαφορετικά KIT γνωριμίας για κάθε ένα από τους δικούς της μικροελεγκτές .

Μικροελεγκτές της **ARIZONA MICROCHIP TECHNOLOGY (PIC)** (PIC16CXXseries) που είναι πολύ δημοφιλής τόσο μεταξύ τον ερασιτεχνών όσο και μεταξύ τον επαγγελματιών διότι προσφέρει αρκετά εργαλεία για την ανάπτυξη εφαρμογών. Η εταιρία parallax Inc. κατασκευάζει τον Basic Stamp που είναι ένα μικροσκοπικό chip μικροϋπολογιστή, μαζί με ένα σύνολο συναφών προϊόντων κατάλληλο για τους υπολογιστές PIC.

Μικροελεγκτές της **MOTOROLA (68HCXXX)**

Ο 68HC 11 υποστηρίζεται από ένα αρκετά μεγάλο πλήθος αναπτυξιακών εργαλείων που κατασκευάζονται από διάφορες εταιρείες.

Άλλοι μικροελεγκτές

- **NATIONAL Semiconductors (COP 800 series or COP8880**
- **TOSHIBA TLCS-870 και TLS-90**
- **Texas (TM320)**

Όλες οι εταιρείες προσφέρουν αναπτυξιακά συστήματα (μικρό Hardware) που κατά κανόνα περιλαμβάνουν και εργαλεία (λογισμικό) για την εκμάθηση της λειτουργίας των αντίστοιχων μικροελεγκτών .

Όλα τα αναπτυξιακά εργαλεία για μικροελεγκτές χρησιμοποιούν ένα πρόγραμμα που τρέχει είτε μόνο του είτε σε συνεργασία με κάποιο άλλο, ή και σε συνδυασμό με το κατάλληλο υλικό μέρος στο οποίο θα αναφερθούμε αργότερα. Το λογισμικό λοιπόν που θα ΄τρέχουν΄ υποχρεωτικά τα εργαλεία αυτά μπορεί να αποτελείται από :

- μεταφραστές σε γλώσσα μηχανής (cross assemblers)
- συμβολομεταφραστές (cross compilers)

- κειμενογράφους πλήρους οθόνης (full screen editors)
- διασυνδετές (linkers)
- προσομοιωτές (simulators)
- ανιχνευτές σφαλμάτων (debuggers)
- μετατροπείς μορφής αρχείων (format converters)

τα περισσότερα από τα προγράμματα αυτά παρέχονται στους χρήστες τους σε παραλλαγές που μπορούν να τρέξουν σε περιβάλλον DOS ή Windows.

Οι δυο ευρεία γνωστές γλώσσες που χρησιμοποιούνται για την συγγραφή των προγραμμάτων εφαρμογής είναι η C και η BASIC .Οι μικροελεγκτές της Atmen έχουν υιοθετήσει σαν βολικότερη γλωσσά την C.

Στις περισσότερες το λογισμικό συνοδεύει ένα αναπτυξιακό πακέτο ενός συγκεκριμένου μικροελεγκτη που εξασφαλίζει τις παρακάτω λειτουργίες και δυνατότητες .

- επικοινωνία με τον κύριο υπολογιστή .
- μεταφορά του προγράμματος από τον κύριο υπολογιστή προς τον μικροελεγκτη και το αντίστροφο .
- εύρεση σφαλμάτων σε επίπεδο συμβολών .
- εύρεση σφαλμάτων τρέχοντας το πρόγραμμα εφαρμογής εντολή προς εντολή .
- τοποθέτηση σημείων στάσης μέσα στο πρόγραμμα , καταγραφή ΄στιγμιοτυπων΄ , πάγωμα της τιμής ενός καταχώρηση ,τοποθέτηση σημείων παρατήρησης κ.λ.π.
- χρήση ενός on line assembler , η disassemble.

Πολλοί κατασκευαστές μικροελεγκτών αλλά και πολλοί προμηθευτές εργαλείων ανάπτυξης , διαθέτουν στην αγορά ένα προϊόν γνωστό με το όνομα IDE (Integrated Development Environment, ολοκληρωμένο περιβάλλον ανάπτυξης).Το IDE έχει σαν σκοπό να επιτρέπει στον χρηστή την άμεση πρόσβαση σε οποιοδήποτε αναπτυξιακό εργαλείο επιθυμεί να χρησιμοποιήσει. Ο αριθμός των IDE που τρέχουν κάτω από το περιβάλλον Windows αυξάνεται με γρήγορους ρυθμούς.

Παρακάτω αναφέρουμε τις δυνατότητες που πρέπει να προσφέρει ένα ικανοποιητικό περιβάλλον ανάπτυξης.

- εύκολη πρόσβαση μέσω ενός αποδοτικού προγράμματος επικοινωνίας με το χρηστή, το οποίο προϋποθέτει τη χρήση Α ενός editor που μπορεί να βλέπει πολλά διαφορετικά παράθυρα , Β ενός προγράμματος ελέγχου του ποντικιού, Γ ενός ημιαυτομάτου assembler ικανού να μετατρέπει τις εντολές του προγράμματος από πηγαίο κώδικα σε κώδικα μηχανής , Δ ενός αλληλεπιδρώντος προγράμματος βοηθείας , Ε ενός ‘σημειωματάριου’ στην οθόνη του υπολογιστή για την καταγραφή σημειώσεων κατά την διάρκεια της εκτέλεσης του προγράμματος εφαρμογής.
- Οι περισσότεροι assemblers διατίθενται σήμερα μαζί με ένα μετατροπία που είναι σε θέση να μετατρέπει τα δυαδικά αρχεία που παράγουν σε αρχεία μορφής IntelHex.
- Η μορφή αυτή αποτελεί πλέον ένα πρότυπο για τα περισσότερα συστήματα, όπως επίσης και για κλακέτες εκτίμησης συγκεκριμένων μικροελεγκτών. Ισχυροί cross assemblers όπως είναι ο cross-32 Met assembler της ASSET είναι σε θέση να πάρουν κώδικα κατάλληλο για διαφόρους τύπους μικροελεγκτών. Το πλεονέκτημα της χρήσης ενός assembler εντοπίζεται στο ότι χρησιμοποιώντας τον ίδιο υπολογιστή μπορείτε να εργαστείτε με πολλούς διαφορετικούς μικροελεγκτές.

Η σειριακή πόρτα

Όλοι οι μικροελεγκτές της MCS-51 διαθέτουν ενσωματωμένο ένα αμφίδρομο ενδιάμεσο κύκλωμα ασύγχρονης σειριακής επικοινωνίας (QUART) που μπορεί να εκπέμπει και να λαμβάνει δεδομένα ταυτόχρονα. Για τον δεκτή ασύγχρονων δεδομένων υπάρχει ένας ειδικός απομόνωσης δεδομένων, ο οποίος επιταχύνει την επικοινωνία με τις περιφερειακές συσκευές σειριακής επικοινωνίας.

Η σειριακή μπορεί να προγραμματιστή για λειτουργία σε 1 από 4 τρόπους ,με ταχύτητα επικοινωνίας (baud rate) και μορφή δεδομένων ελεγχόμενες μέσω προγράμματος . Μπορούν να επιλεγούν όλες οι συνηθισμένες ταχύτητες επικοινωνίας μέχρι τα 19200 baud (19200 bits/sec). Μπορούν επίσης να επιλεγούν ταχύτητες χρονισμού μέχρι 1MHz για χρήση σε δίκτυα και συστήματα επικοινωνίας πολλών επεξεργαστών. Η επιλογή της ταχύτητας χρονισμού επιτυγχάνεται με την βοήθεια των απαριθμητών / αχρόνιστων.

1.3 Διευθύνσεις μνήμης

Σε συνηθισμένες εφαρμογές των μικροεπεξεργαστών / μικροελεγκτών διατίθενται 16 ποδιά του chip για να δώσουν το σήμα διεύθυνσης στην μνήμη. Ο επεξεργαστής βγάζει στα 16 αυτά ποδιά 16 στάθμες (bit 0 ή 1) που ορίζουν την διεύθυνση μνήμης με την οποία επιθυμεί να ανταλλάξει πληροφορίες . Οι συνδυασμοί που προκύπτουν είναι 2**16=65536 διαφορετικές Διευθύνσεις (64K).Είναι όμως δύσκολο να συγκρατήσουμε μια τέτοια διεύθυνση, για αυτό το λόγω την μετατρέπουμε σε 16αδικο σύστημα (HEX) όπως θα δούμε παρακάτω.

Έτσι οι διευθύνσεις της μνήμης είναι από 0000 HEX μέχρι την FFFF HEX.
Η διεύθυνση λοιπόν 0000 είναι η πρωτη
Η διεύθυνση 000F είναι η 15η
Η διεύθυνση 0010 είναι η16η
Η διεύθυνση 00FF είναι η 255η
Η διεύθυνση 0100 είναι η 256η
Η διεύθυνση 1000 είναι η 4096η
Η διεύθυνση 48FD είναι η 4X 16**3 + 8X 16**2 + 15X16**1+13X16**0=18635
Η διεύθυνση FFFF είναι η $65535^{η}$

Επίσης η διεύθυνση 0110 έχει προηγούμενη την 010F και επόμενη την 0111
η διεύθυνση DEF9 έχει προηγούμενη την DEF8 και επόμενη την DEFA
η διεύθυνση 09BF έχει προηγούμενη την 09BE και επόμενη την 09C0

1.4 Μικροελεγκτές FLASH

.

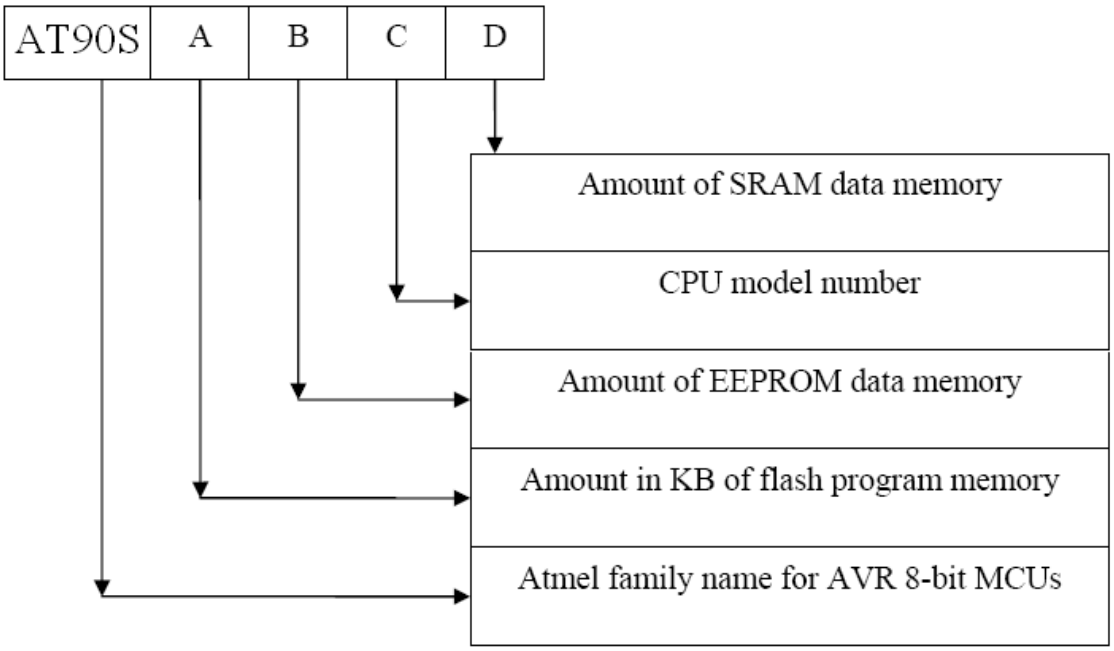
Τα προηγούμενα χρονιά αναπτύσσοντας προγράμματα σε γλωσσά 8051 διαπιστώσαμε ότι ήταν επίπονο και χρονοβόρο. Όταν μάλιστα απουσίαζε ένας εξωμότης ROM ή μια εξωτερική μνήμη που φιλοξενούσαμε το πρόγραμμα monitor ή οποιαδήποτε αλλά προγράμματα εύρεσης σφαλμάτων, τα προγράμματα ήταν ακόμα ποιο δύσκολα.
Στις περιπτώσεις αυτές χρησιμοποιηθήκαν EPROM's οπού γράφαμε και ξαναγράφαμε μέχρι να υλοποιήσουμε την τελική μορφή του προγράμματος.
Σήμερα μπορούμε να χρησιμοποιήσουμε ένα 8051 της Atmen με ενσωματωμένη FLASH RAM που απλοποιεί σημαντικά το έργο των σχεδιαστικών συστημάτων ελέγχου. Ένας μικροελεγκτής με τέτοια μνήμη μπορεί να σβηστεί μέσα σε μερικά δευτερόλεπτα χωρίς να πρέπει να εκτεθεί σε υπεριώδες φως όπως τα EPROM, ενώ το ίδιο μπορεί να επαναληφθεί για 1000 φορές έναντι μονό 5-10 των EPROM.

1.5 Μικροελεγκτές ATMEL AVR

Ένα χαρακτηριστικό παράδειγμα σύγχρονων μικροελεγκτών είναι εκείνοι της οικογένειας AVR της εταιρείας ATMEL. Οι μικροελεγκτές αυτοί προσφέρονται με ένα πλήθος εναλλακτικού αριθμού ακροδεκτών, ξεκινώντας από μικρά και φτηνά ολοκληρωμένα των 8 ακροδεκτών για εφαρμογές πολύ χαμηλού κόστους με περιορισμένες απαιτήσεις σε πλήθος προγραμματιζόμενων ακροδεκτών γενικού σκοπού. Οι πιο εξελιγμένοι μικροελεγκτές της οικογένειας διαθέτουν περισσότερους από 60 προγραμματιζόμενους ακροδέκτες γενικού σκοπού. Επίσης πολλά μέλη της σειράς διατίθενται σε τρεις παραλλαγές: τους απλούς μικροελεγκτές που λειτουργούν στα 5V, τους χαμηλής κατανάλωσης στα 2.7V ,(κατάληξη L) και τους πολύ χαμηλούς σε κατανάλωση στα 1.8V (κατάληξη V).
Συνήθως οι ακροδέκτες γενικού σκοπού έχουν πολυπλεγμένες περισσότερες από μία λειτουργίες, όπως για παράδειγμα είσοδοι με ικανότητα να προκαλούν διακοπή (interrupt) στον εσωτερικό επεξεργαστή, είσοδοι αναλογικών συγκριτών ή μετατροπέων αναλογικού σε ψηφιακό (ADC), είσοδοι κεντρικού u961 ρολογιού (oscillator) ή ασύγχρονης οδήγησης μετρητών (counters), ακροδέκτες για σύνδεση με διάφορες διεπαφές όπως USART. SPI κ.α. Στα πιο εξελιγμένα μέλη της οικογένειας διατίθενται ενσωματωμένα περιφερειακά ακόμα και για την οδήγηση LCD οθόνης ή τη σύνδεση με USB interface.
Στο εσωτερικό ενός μικροελεγκτή όπως ο AVR υπάρχει ένας αριθμός από διαφορετικούς τύπους μνήμης, όπως Flash για την εγγραφή του λογισμικού συστήματος (firmware), EEPROM για την αποθήκευση διαφόρων παραμέτρων, καθώς και κάποιος αριθμός θέσεων μνήμης Ram για τις μεταβλητές τον λογισμικού.
Για το λόγο αυτό οι AVR δεν βγάζουν σε ακροδέκτες την εσωτερική αρτηρία διευθύνσεων ή δεδομένων παρά μόνο ακροδέκτες γενικού σκοπού.
Με όλα τα παραπάνω περιφερειακά είναι φανερό ότι το πλήθος των εξωτερικών στοιχείων που απαιτούνται για τη δημιουργία ενός συστήματος με μικροελεγκτή AVR είναι ελάχιστο.Το βασικό μειονέκτημα μιας τέτοιας αρχιτεκτονικής μικροελεγκτή είναι η δυσκολία επεκτασιμότητας. Π.χ , αν οι απαιτήσεις σε μνήμη RAM είναι μεγάλες, ο μικροελεγκτής δεν είναι εύκολο να συνδεθεί με εξωτερική μνήμη, μια και δεν έχει αρτηρία διευθύνσεων και δεδομένων. Για να γίνει αυτό θα πρέπει να υλοποιηθούν τέτοιες αρτηρίες με τη χρήση ακροδεκτών γενικού σκοπού οι οποίες ωστόσο θα ήταν αδύνατο να επιτύχουν γρήγορους χρόνους προσπέλασης της μνήμης. Επίσης η συχνότητα ρολογιού στην οποία λειτουργούν τέτοιοι μικροελεγκτές δεν ξεπερνά τα 20 MHz στα πιο εξελιγμένα μοντέλα μιας σειράς όπως οι AVR mega.

1.6 Σύστημα ονομασίας μικροελεγκτών AVR

Η εταιρεία ATMEL χρησιμοποιεί ένα σύστημα ονομασίας για τους μικροελεγκτές που ανήκουν στην οικογένεια AVR. Το σύστημα αυτό φαίνεται παρακάτω:



Κωδικοποίηση χωρητικότητας για τη μνήμη EEPROM και SRAM												
Κωδικός	0	1	2	3	4	5	6	7	8	9	A	B
χωρητικότητα	0	32	64	128	256	512	1K	2K	4K	8K	16K	32K

Family Member	Flash (KB)	EEPROM (Bytes)	CPU Model	SRAM (Byte)	Counter Timer	UART	ADC	Pins	Avaliable
AT90S 1200	1	64	0	0	1	No	No	20	Q197
AT90S 1220	1	64	2	0	1	No	No	8	Q397
AT90S 2313	2	128	1	128	2	Yes	No	20	Q297
AT90S 4414	4	256	1	256	3	Yes	No	40/44	Q397
AT90S 4433	4	256	3	128	?	Yes	Yes	28	Q497
AT90S 8515	8	512	1	512	3	Yes	No	40/44	Q297
AT90S 68718	68	2048	1	4096					

2. Ο Μικροελεγκτής AT90S8515 της Atmel

2.1 Περιγραφή

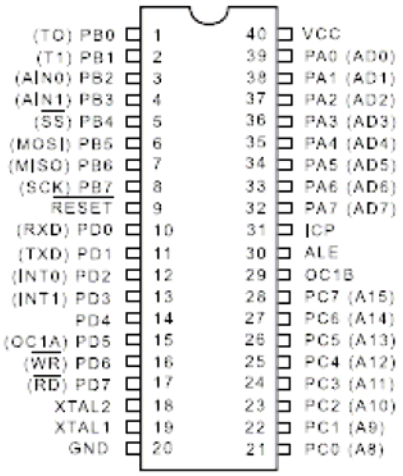
Ο AT90S8515 είναι ένας χαμηλής κατανάλωσης COMS, 8-bit μικροελεγκτής βασισμένος στην AVR τεχνολογία και στην αρχιτεκτονική. Εκτελώντας τις περισσότερες από τις 120 εντολές του σε ένα κύκλο ρολογιού ο AT9S08515 επιταχύνει να προσεγγίζει ταχύτητα 1 MIPS(Milions Instructions Per Second) για κάθε MHz επιτρέποντας στους σχεδιαστές να επιτυγχάνουν την καλύτερη δυνατή σχέση ταχύτητας/κατανάλωσης.
Διαθέτει 32 καταχώρησες γενικής χρήσης(general purpose registers). Όλοι αυτοί οι καταχωρητές είναι απευθείας συνδεμένοι με την αριθμητική λογική μονάδα(ALU-Arithmetic Logic Unit),επιτρέποντας την πρόσβαση σε δυο ανεξαρτήτους καταχωρητές με μια εντολή που εκτελείται σε ένα κύκλο ρολογιού .Το αποτέλεσμα εδώ είναι περισσότερος αποδοτικός κώδικας, επιτυγχάνοντας απόδοση μέχρι και δέκα φορές γρηγορότερη από τους συμβατικούς μικροελεγκτές τεχνολογίας CISK(Complex Instruction Set Compyter).

2.2 Ο AT90S8515 παρουσιάζει τα παρακάτω γενικά χαρακτηριστικά:

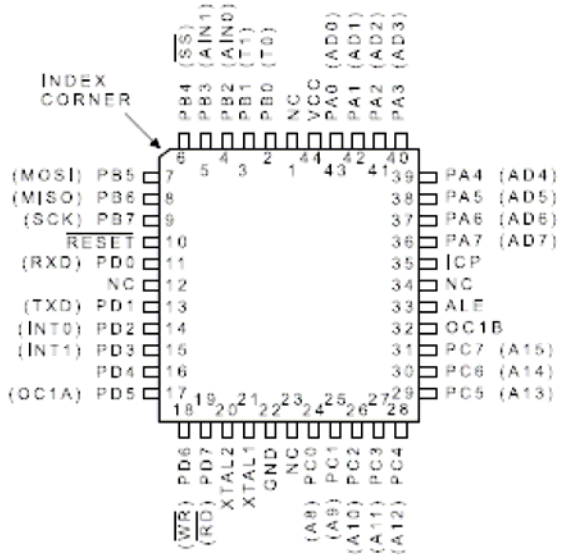
8 K μνήμη προγράμματος, 512 bytes μνήμη δεδομένων, μνήμη EEPROM 512 bytes 32 ακίδες εισόδου / εξόδου , 32 καταχωρητές γενικής χρήσης, δυο ευέλικτοι χρονιστές / μετρητές με δυνατότητα σύλληψης και σύγκρισης, διαχείρισης εσωτερικών και εξωτερικών σημάτων διακοπής (interrupts), προγραμματιζόμενο UART, χρονιστής επιτήρησης με δικό του ταλαντωτή και τέλος μια θύρα SPI για τον προγραμματισμό της Flash μνήμης .

Στην κατάσταση αναμονής (Idle mode),η CPU διακόπτει την λειτουργία της διατηρώντας όμως σε ετοιμότητα την SRAM, την θύρα SPI, τους μετρητές / χρονιστές , το UART και την βαθμίδα διαχείρισης σημάτων διακοπής. Στην κατάσταση χαμηλής κατανάλωσης (Power Down mode) διατηρούνται μονό τα περιεχόμενα των καταχωρητών .Όλες οι υπόλοιπες βαθμίδες τίθενται εκτός λειτουργίας, έως ότου φθάσει ένα σήμα διακοπής ή σήμα εκκίνησης(Reset).
Ο AT90S8515 αποτελεί το προϊόν της συγκατοίκησης μιας CPU τύπου RISC των 8 και μιας μη πτητικής μνήμης Flash. Η κατασκευή του είναι δυνατή με την βοήθεια της τεχνολογίας ολοκλήρωσης υψηλής πυκνότητας μη πτητικών μνημών, που έχει επινοήσει η Atmel. Η τεχνολογία αυτή επιτρέπει την εγγραφή της μνήμης Flash πάνω στην πλακέτα που βρίσκεται ο μικροελεγκτής ή εναλλακτικά μέσω ενός οποιουδήποτε προγραμματιστή ικανού να εγγράψει μνήμες αυτής της κατηγορίας.
Το αποτέλεσμα του συνδυασμού είναι ένα πανίσχυρο ολοκληρωμένο κύκλωμα, ικανό να ανταποκριθεί σε ένα μεγάλο πλήθος εφαρμογών.
Ο μικροελεγκτής συνοδεύεται από αρκετά εργαλεία ανάπτυξης και προγραμματισμού, μεταξύ των οποίων συμπεριλαμβάνονται μεταφραστές από γλωσσά C(compilers), συμβολομεταφραστές με ικανότητα μακροεντολών , προσομοιωτές σε επίπεδο λογισμικού ,εξομοιωτές πραγματικού χρόνου και κιτ αξιολόγησης.
Παρακάτω βλέπουμε το pin-out του AT90S8515 και αμέσως μετά το block διάγραμμα του:

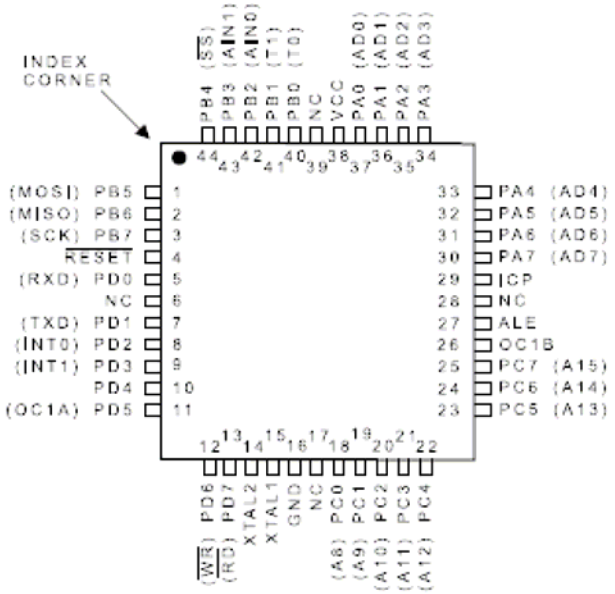
PDIP



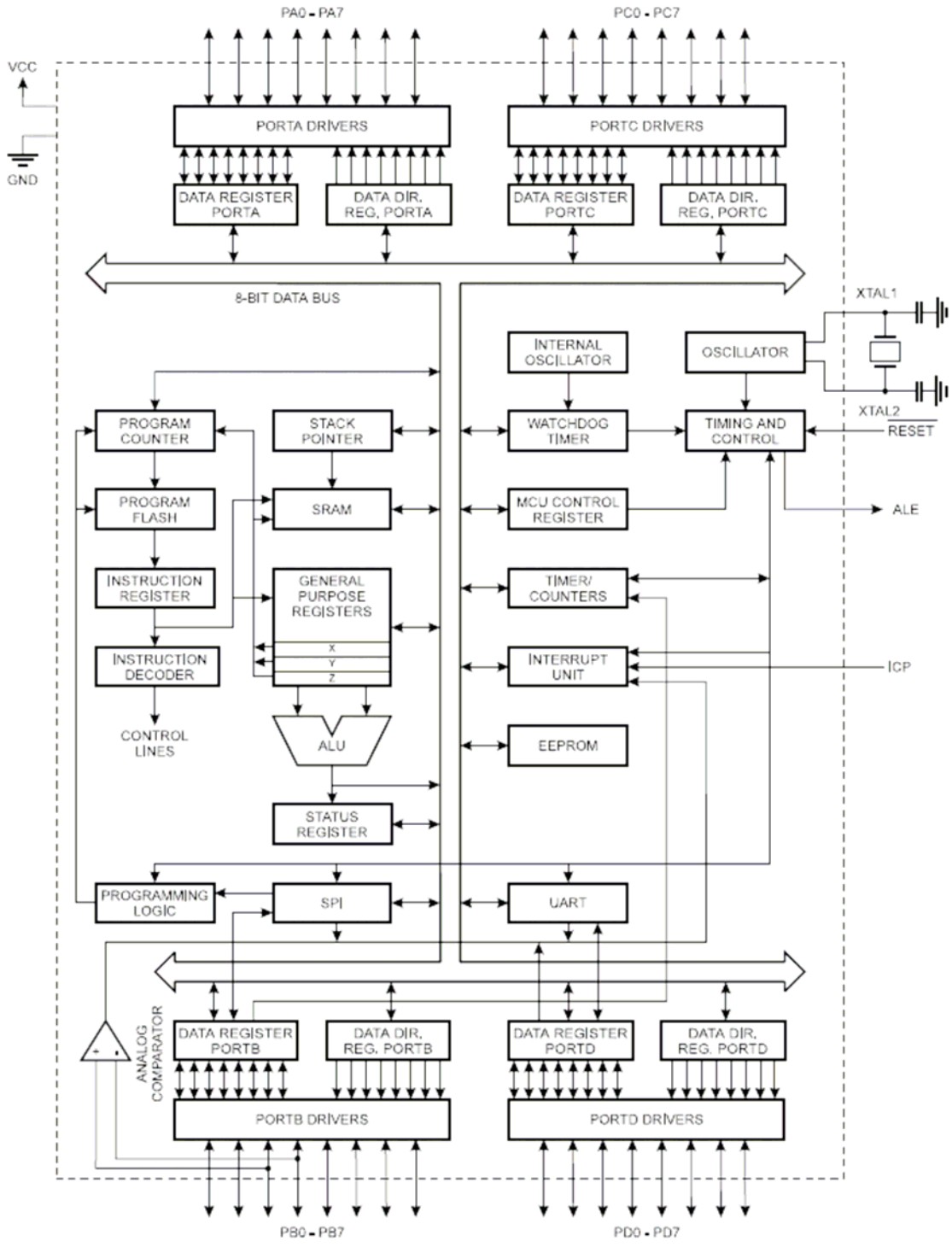
PLCC



TQFP



Μπλόκ διάγραμμα του AT90S8515:



Τεχνικά χαρακτηριστικά

- AVR Υψηλής απόδοσης και χαμηλής κατανάλωσης , αποτέλεσμα της αρχιτεκτονικής .
- 120 ισχυρές εντολές που οι περισσότερες εκτελούνται σε ένα κύκλο ρολογιού .
- 8K bytes flash μνήμης ,προγραμματιζόμενης πάνω στην πλακέτα .
  -SPI σειριακή διασύνδεση για τον προγραμματισμό της .
  -Διάρκεια: 1000 κύκλοι εγγραφής / διαγραφής.
- 512 bytes μνήμης EEPROM -Διάρκεια 100.000 κύκλοι εγγραφής / διαγραφής
- 512 bytes εσωτερικής SRAM (Static RAM)
- 32 γενικής χρήσης καταχωρητές 1-byte
- 32 προγραμματιζόμενες ακίδες εισόδου / εξόδου
- Προγραμματιζόμενο σειριακό UART (Universal Asynchronous Receiver Transmitter) πλήρως αμφίδρομο (full duplex)
- SPI σειριακή διασύνδεση
- Τροφοδοσία Vcc: 2.7-6.0 V
- Πλήρης στατική λειτουργία 0-20 MHz
- Ταχύτητα 4 MIPS με κρύσταλλο 4 MHz
- Ένα 8-bit χρονιστή / μετρητή με ανεξάρτητο προδιαιρέτη
- Ένα 16-bit χρονιστή /μετρητή με ανεξάρτητο προδιαιρέτη και λειτουργίες σύγκρισης και σύλληψης
- Διπλή γεννήτρια PWM με επιλεγόμενη ακρίβεια 8 , 9 ή 10 bit
- Υποστήριξη εσωτερικών και εξωτερικών σημάτων διακοπής (interrupt)
- Προγραμματιζόμενος χρονιστής επιτήρησης (Watchdog timer) με ενσωματωμένο ταλαντωτή
- Αναλογικός συγκριτής
- Μικρό ρεύμα σε κατάσταση αναμονής και χαμηλής ισχύος
- Προστασία περιεχομένων μνήμης προγράμματος

2.3 Σημασία των ακίδων του 8515:

VCC: **Ακίδα τροφοδοσίας** **GND**: **Ακίδα αναφοράς (γη)**

PortA (PA7……PA0):

Η θύρα αυτή περιλαμβάνει 8 αμφίδρομες (in/out) ακίδες. Οι ακίδες αυτές μπορούν κατά επιλογή να συνδεθούν εσωτερικά στην τροφοδοσία μέσω αντιστάσεων πρόσδεσης (pull-up resistor) όταν λειτουργούν σαν είσοδοι.

Όλες οι ακίδες όταν συμπεριφέρονται σαν έξοδοι έχουν την δυνατότητα να απορροφήσουν ρεύμα 20mA στην LOW κατάσταση και να δώσουν 10mA στην HIGH.

Όταν σεττάρονται σαν είσοδοι και πρέπει να δούν το LOW ενεργοποιούμε τις εσωτερικές αντιστάσεις .Ενώ για να δούν HIGH τοποθετούμε εξωτερικά αντιστάσεις PULL-DOWN .Τέλος η θύρα Α χρησιμοποιείτε σαν πολυπλέκτης διευθύνσεων / δεδομένων in / out όταν χρησιμοποιούμε εξωτερική μνήμη.

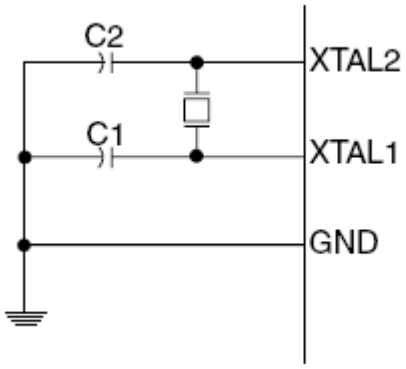
PORTB (PB7….PB0):

Η θύρα αυτή περιλαμβάνει 8 αμφίδρομες ακίδες. Οι ακίδες αυτές μπορούν κατά επιλογή να συνδεθούν εσωτερικά στην τροφοδοσία μέσω αντιστάσεων πρόσδεσης (pull-up resistor) όταν λειτουργούν σαν είσοδοι.

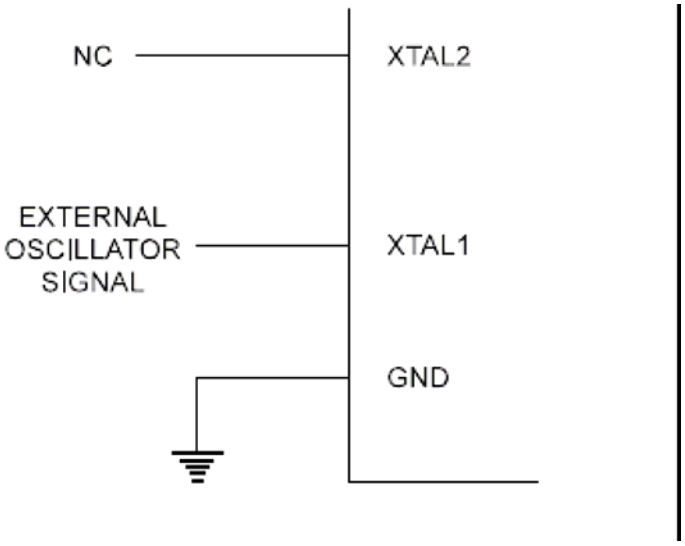
Όλες οι ακίδες όταν συμπεριφέρονται σαν έξοδοι έχουν την δυνατότητα να απορροφήσουν ρεύμα 20mA στην LOW κατάσταση και να δώσουν 10mA στην HIGH.
Όταν σεττάρονται σαν είσοδοι και πρέπει να δούν το LOW ενεργοποιούμε τις εσωτερικές αντιστάσεις .Ενω για να δούν HIGH τοποθετούμε εξωτερικά αντιστάσεις PULL-DOWN.Οι PB2 και PB3 οδηγούνται επίσης και στις εισόδους του ενσωματωμένου αναλογικού συγκριτή (μη αναστρέφουσα και αναστρέφουσα αντίστοιχα).Τέλος πολλές από τις ακίδες της θύρας B έχουν και άλλες λειτουργίες που αναφέρονται στο data sheet του μικροελεγκτή.
PORTC (PC7….PC0):
Η θύρα αυτή περιλαμβάνει 8 αμφίδρομες ακίδες. Οι ακίδες αυτές μπορούν κατά επιλογή να συνδεθούν εσωτερικά στην τροφοδοσία μέσω αντιστάσεων πρόσδεσης (pull-up resistor).
Όλες οι ακίδες όταν συμπεριφέρονται σαν έξοδοι, έχουν την δυνατότητα να απορροφήσουν ρεύμα 20mA στην LOW κατάσταση και να δώσουν 10mA στην HIGH.
Όταν σεττάρονται σαν είσοδοι και πρέπει να δούν το LOW ενεργοποιούμε τις εσωτερικές αντιστάσεις. Ενώ για να δούν HIGH τοποθετούμε εξωτερικά αντιστάσεις PULL-DOWN.Η θύρα C επίσης εξυπηρετεί λειτουργίες σαν έξοδος διευθύνσεων όταν χρησιμοποιείται εξωτερική SRAM.
PORTD (PD7….PD0):
Η θύρα αυτή περιλαμβάνει 8 αμφίδρομες ακίδες. Οι ακίδες αυτές μπορούν κατά επιλογή να συνδεθούν εσωτερικά στην τροφοδοσία μέσω αντιστάσεων πρόσδεσης (pull-up resistor).
Όλες οι ακίδες όταν συμπεριφέρονται σαν έξοδοι, έχουν την δυνατότητα να απορροφήσουν ρεύμα 20mA στην LOW κατάσταση και να δώσουν 10mA στην HIGH.
Όταν σεττάρονται σαν είσοδοι και πρέπει να δούν το LOW ενεργοποιούμε τις εσωτερικές αντιστάσεις. Ενώ για να δούν HIGH τοποθετούμε εξωτερικά αντιστάσεις PULL-DOWN.
Η θύρα D επίσης εξυπηρετεί τις λειτουργίες κάποιων ιδιαίτερων χαρακτηριστικών που αναφέρονται στο data sheet του μικροελεγκτή.
RESET:
Ακίδα επανεκκίνησης. Ο μικροελεγκτής αρχικοποιείται, όταν επιβληθεί σε αυτήν χαμηλή στάθμη για χρόνο ίσο με δύο κύκλους μηχανής τουλάχιστον, (ο ταλαντωτής της CPU οφείλει να βρίσκεται σε λειτουργία).
XTAL1:
Είσοδος στον εσωτερικό αναστρέφοντα ταλαντωτή και ταυτόχρονα στο σύστημα χρονισμού του μικροελεγκτή.
XTAL2:
Έξοδος του αναστρέφοντα ταλαντωτή.
ICP:
Ειναι η ακίδα εισόδου για την λειτουργία της σύλληψης για τον χρονιστή / μετρητή 1.
OC1B:
Ειναι η ακίδα εξόδου για την λειτουργία της σύγκρισης για τον χρονιστή / μετρητή 1.
ALE:
Χρησιμοποιείται όταν η εξωτερική μνήμη είναι ενεργοποιημένη. Η ενεργοποίηση του ALE για την μανδάλωση της χαμηλής (low order) στη σειρά διεύθυνσης (8 bits) μέσα σε ένα μανδαλωτή διεύθυνσης κατά τον πρώτο κύκλο προσπέλασης. Τα ίδια pins χρησιμοποιούνται για τα δεδομένα AD0-AD7 κατά την διάρκεια του δεύτερου κύκλου προσπέλασης.

2.4 Κρυσταλικός ταλαντωτής:

Ο αναστρέφων ταλαντωτής δέχεται στις δύο ακίδες του έναν εξωτερικό κρύσταλλο ή έναν κεραμικό συντονιστή (σχέδιο1).Στην περίπτωση αυτή η συχνότητα καθορίζεται από το χρονιστικό εξάρτημα. Μπορεί ακόμα να δεχθεί εξωτερικό σήμα χρονισμού από μια άλλη πηγή (σχέδιο 2).



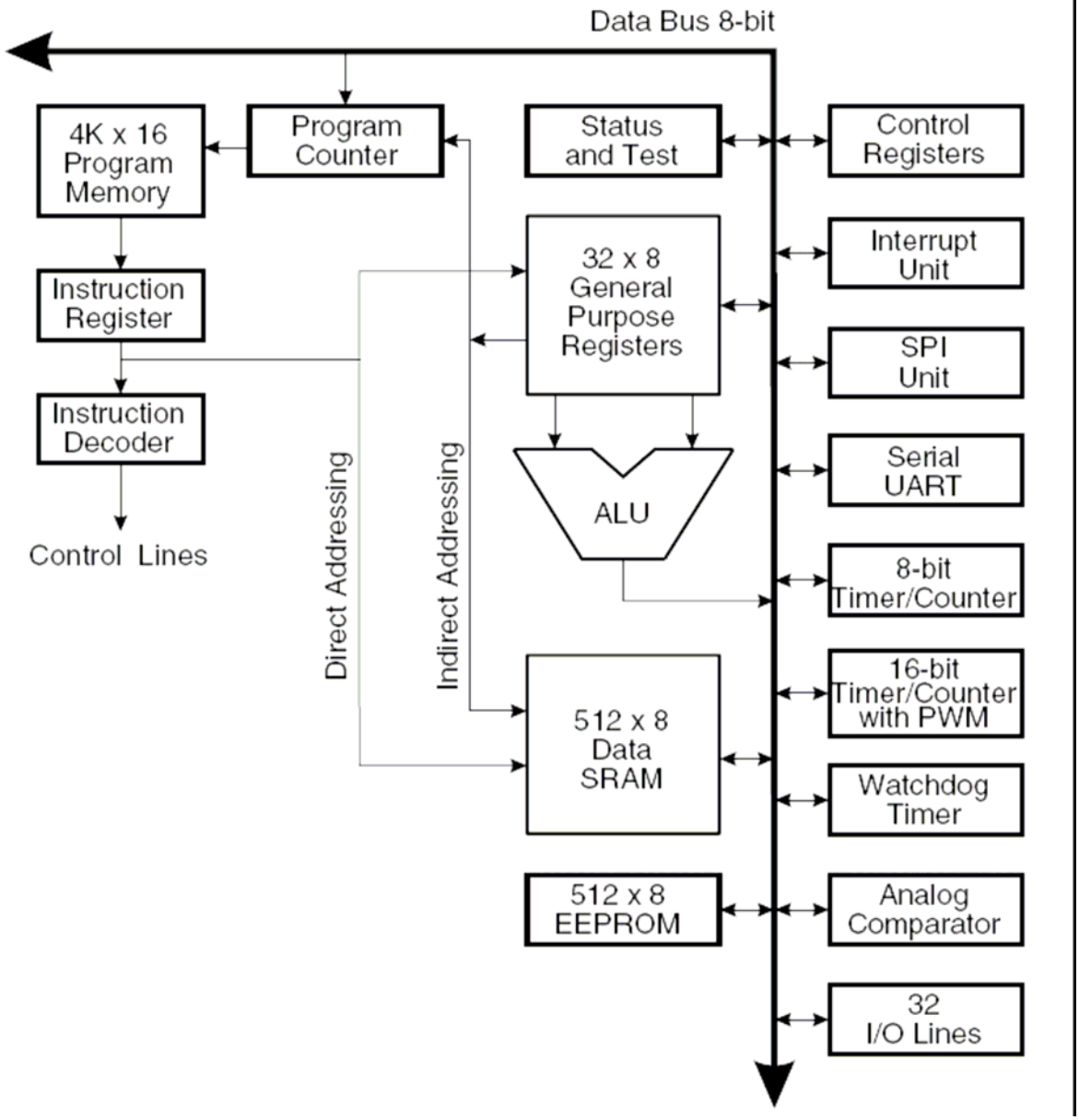
Σχέδιο1



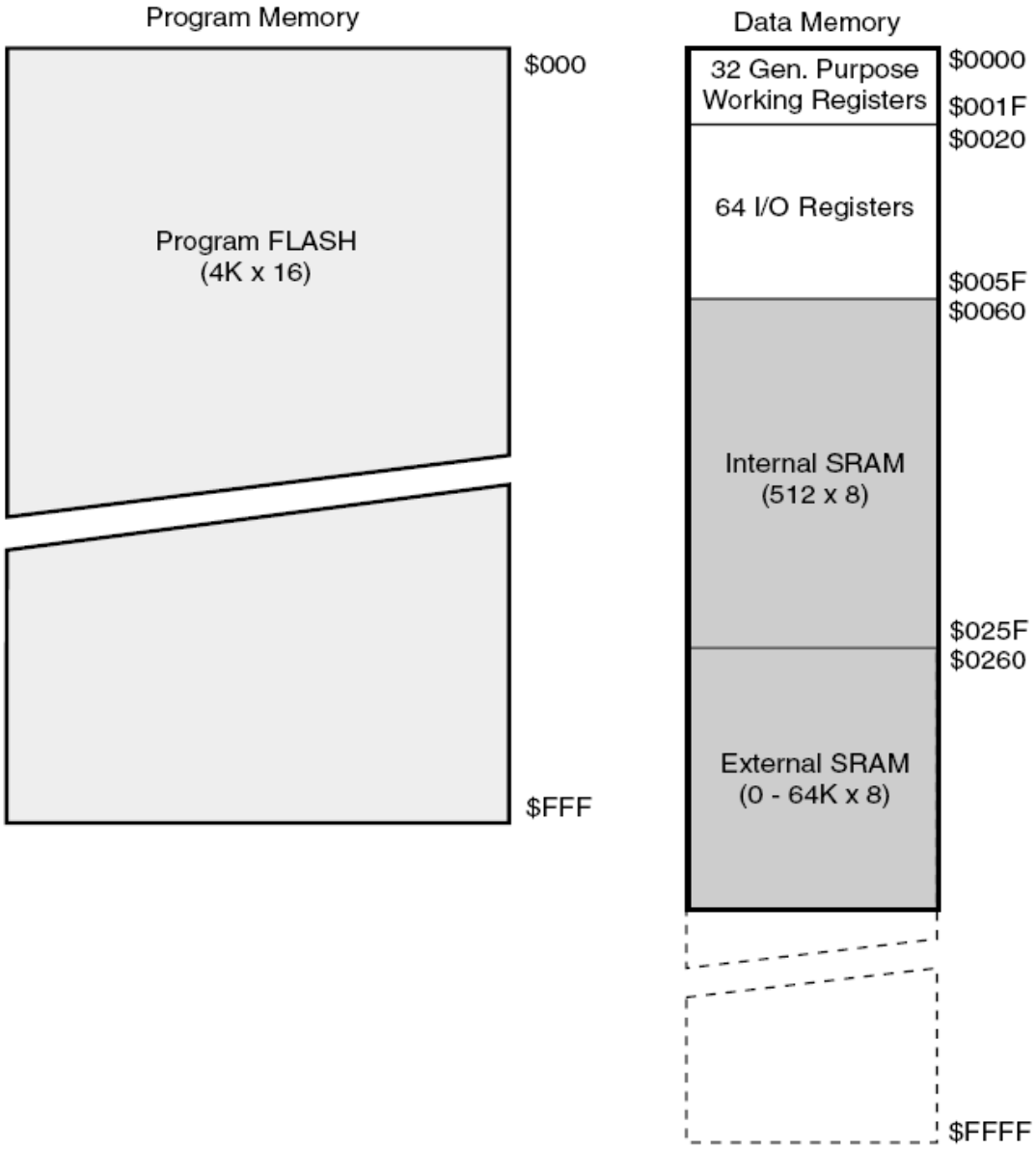
Σχέδιο 2

2.4 Αρχιτεκτονική RISC ΤΟΥ AT90S8515

Η ιδέα των καταχωρητών ταχείας – προσπέλασης στηρίζεται σε 32 (8 bit) καταχωρητές γενικής χρήσης με χρόνο πρόσβασης ένα κύκλο ρολογιού. Αυτό σημαίνει ότι σε ένα κύκλο ρολογιού , μόνο μια λειτουργία της ALU εκτελείται. Δύο τελεστές εξάγονται από τον τομέα των καταχωρητών, η λειτουργία εκτελείται, και το αποτέλεσμα αποθηκεύεται πίσω στον τομέα των καταχωρητών (σε ένα κύκλο ρολογιού).
Έξι από τους 32 καταχωρητές μπορούν να χρησιμοποιηθούν ως τρείς 16-bit καταχωρητές δείκτες (Index) για έμμεση διευθυνσιοδότηση. Αυτοί οι καταχωρητές με τις επιπρόσθετες λειτουργίες είναι οι καταχωρητές: **X-register, Y-register και Z-register**.
Η ALU υποστηρίζει αριθμητικές και λογικές πράξεις μεταξύ καταχωρητών ή μεταξύ μιας σταθεράς και ενός καταχωρητή. Το παρακάτω σχήμα δείχνει την AVR και RISC αρχιτεκτονική του AT90S8515.



Οι καταχωρητές αυτοί καταλαμβάνουν τις 32 πιο χαμηλές θέσεις μνήμης στην SRAM ($00-$1F) και έτσι μπορεί να γίνει πρόσβαση σε αυτούς σαν φυσιολογικές θέσεις μνήμης .
Ο χώρος στη μνήμη με το όνομα I/0 περιέχει 64 διευθύνσεις για τις λειτουργίες διάφορων περιφεριακών όπως καταχωρητές ελέγχου, χρονιστές / μετρητές, κ.λ.π .Εκτός της πρόσβασης με κάποιο συγκεκριμένο όνομα του καταχωρητή μπορεί να γίνει και πρόσβαση μέσω διεύθυνσης στην μνήμη δεδομένων. Ο χώρος που καταλαμβάνουν είναι από την διεύθυνση $20 μέχρι και την $5F, όπως φαίνεται και παρακάτω.
Η AVR τεχνολογία χρησιμοποιεί Harvard αρχιτεκτονική με ξεχωριστές περιοχές μνήμης(program and data memory) και ξεχωριστούς διαύλους για κάθε μία . Με την χρήση διασωλήνωσης ή αλλιώς διοχέτευσης (pipe-line) γίνεται ανάκληση από την μνήμη προγράμματος τύπου Flash , της επόμενης εντολής τη στιγμή που εκτελείται η πρώτη .
Στο παρακάτω σχήμα βλέπουμε πως είναι διαμορφωμένη η μνήμη προγράμματος και δεδομένων στον AT90S8515

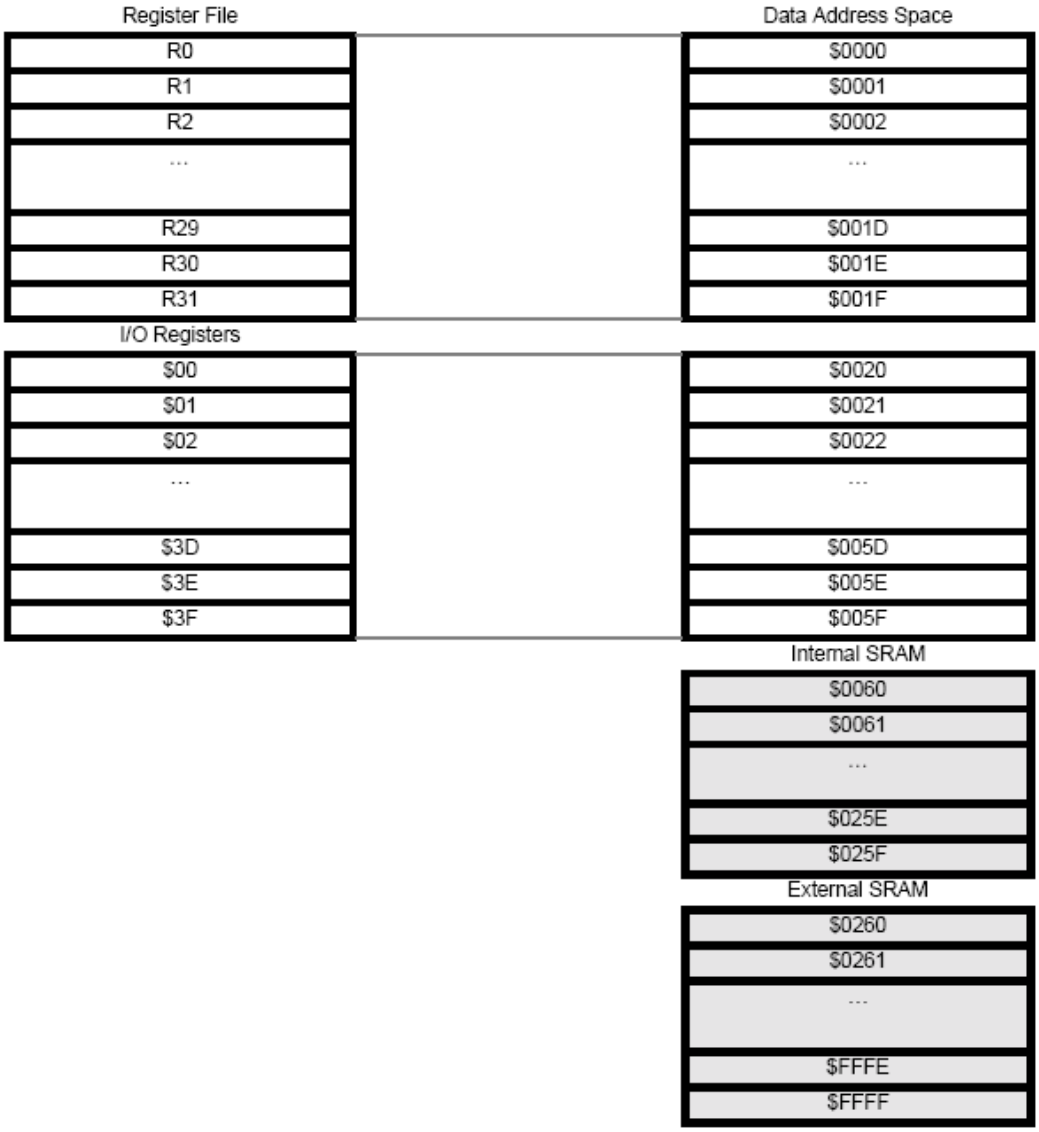


2.5 Η μνήμη προγράμματος τεχνολογίας Flash:

Από την στιγμή που όλες οι εντολές είναι 16 ή 32 bits λέξεις , η συγκεκριμένη μνήμη είναι οργανωμένη ως 4K x 16.Η διάρκεια ζωής της είναι το λιγότερο 1000 κύκλοι εγγραφής / σβησίματος Ο program counter του συγκεκριμένου μικροελεγκή έχει μέγεθος 12 bit.

Η μνήμη δεδομένων (SRAM):

Στο παρακάτω σχήμα φαίνεται η οργάνωση της εσωτερικής μνήμης RAM του AT90S8515



Υποστηρίζονται συνολικά 608 διευθύνσεις μέσω των οποίων προσπελαύνονται: οι καταχωρητές γενικής χρήσης οι εσωτερικές περιφερειακές μονάδες της CPU (μονάδες 1/0) και η μνήμη δεδομένων. Οι πρώτες 96 διευθύνσεις αποτείνονται στους 32 καταχωρητές και στις 64 (μονάδες 1/0) και αυτή καθ'αυτή η μνήμη δεδομένων. Οι πρώτες 96 διευθύνσεις αποτείνονται στους 32 καταχωρητές και στις 64 μονάδες I/O, ενώ οι τελευταίες 512 προορίζονται για την μνήμη δεδομένων.

Μια προαιρετική εξωτερική μνήμη SRAM μπορεί να τοποθετηθεί στον ίδιο χώρο της SRAM.Αυτή η μνήμη θα απασχολεί τον χώρο που βρίσκεται μετά την εσωτερική SRAM και μέχρι 64K-1 ανάλογα με το μέγεθος της SRAM.

Η CPU μπορεί να δει αυτόν τον χώρο μέσω πέντε διαφορετικών τρόπων διευθυνσιοδότησης:
Τον άμεσο
Τον έμμεσο
Τον έμμεσο με επιπλέον όρισμα,
Τον έμμεσο με μείωση διεύθυνσης πριν την προσπέλαση και
Τον έμμεσο με αύξηση της διεύθυνσης μετά την προσπέλαση.

Οι καταχωρητές R26-R31 συμπεριφέρονται σαν καταχωρητές δείκτη (X,Y,Z) μέσω των οποίων γίνονται οι έμμεσες προσπελάσεις.

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

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

- Ως μνήμη SRAM. Περιοχή διευθύσεων στην περίπτωση αυτή είναι $20-$5F.
- Ως I/0 καταχωρητές με τις διευθύνσεις να ξεκινούν από την διεύθυνση $00 και να φθάνουν ως και την διεύθυνση $3F.

Οι 64 καταχωρητές 1/0 που προαναφέραμε φαίνονται αναλυτικά στον παρακάτω πίνακα . Η διεύθυνση που είναι έξω από την παρένθεση είναι η διεύθυνση που έχει ο καταχωρητής . Η διεύθυνση που βρίσκεται μέσα στην παρένθεση είναι η διεύθυνση στην SRAM.

Register Summary

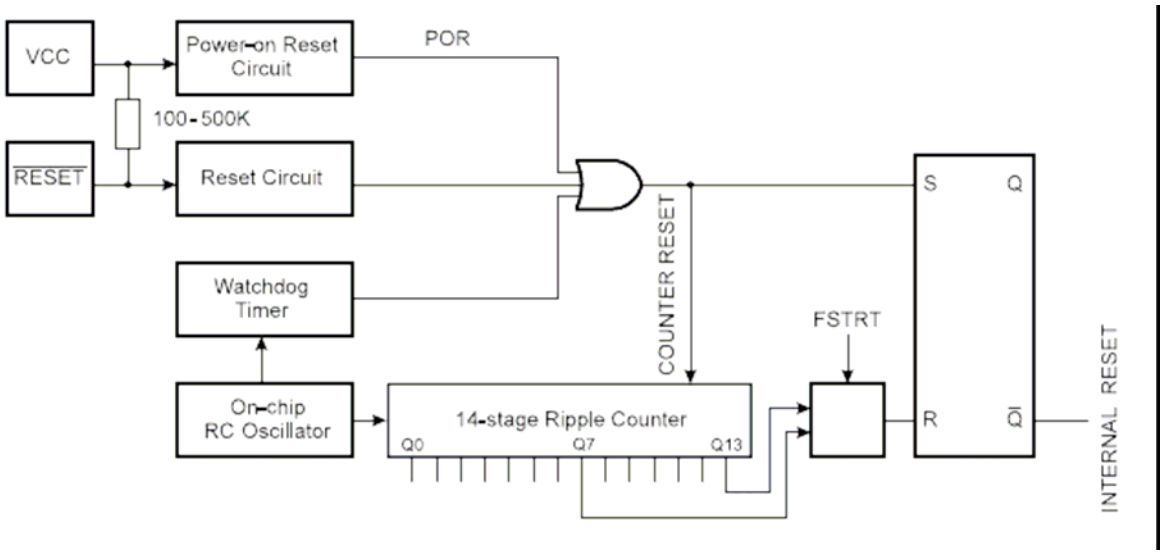
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	page 20
$3E ($5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	page 21
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 21
$3C ($5C)	Reserved									
$3B ($5B)	GIMSK	INT1	INT0	-	-	-	-	-	-	page 26
$3A ($5A)	GIFR	INTF1	INTF0							page 26
$39 ($59)	TIMSK	TOIE1	OCIE1A	OCIE1B	-	TICIE1	-	TOIE0	-	page 27
$38 ($58)	TIFR	TOV1	OCF1A	OCF1B	-	ICF1	-	TOV0	-	page 28
$37 ($57)	Reserved									
$36 ($56)	Reserved									
$35 ($55)	MCUCR	SRE	SRW	SE	SM	ISC11	ISC10	ISC01	ISC00	page 29
$34 ($54)	Reserved									
$33 ($53)	TCCR0	-	-	-	-	-	CS02	CS01	CS00	page 33
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								page 34
...	Reserved									
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	PWM11	PWM10	page 36
$2E ($4E)	TCCR1B	ICNC1	ICES1	-	-	CTC1	CS12	CS11	CS10	page 37
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								page 38
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								page 38
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								page 38
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								page 38
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								page 39
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								page 39
...	Reserved									
$25 ($45)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								page 39
$24 ($44)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								page 39
...	Reserved									
$21 ($41)	WDTCR	-	-	-	WDTOE	WDE	WDP2	WDP1	WDP0	page 42
$20 ($40)	Reserved									
$1F ($3F)	EEARH	-	-	-	-	-	-	-	EEAR8	page 44
$1E ($3E)	EEARL	EEPROM Address Register Low Byte								page 44
$1D ($3D)	EEDR	EEPROM Data Register								page 44
$1C ($3C)	EECR	-	-	-	-	-	EEMWE	EEWE	EERE	page 44
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	page 63
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	page 63
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	page 63
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	page 65
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	page 65
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	page 65
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	page 70
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	page 71
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	page 71
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	page 73
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	page 73
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	page 73
$0F ($2F)	SPDR	SPI Data Register								page 51
$0E ($2E)	SPSR	SPIF	WCOL	-	-	-	-	-	-	page 50
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	page 49
$0C ($2C)	UDR	UART I/O Data Register								page 55
$0B ($2B)	USR	RXC	TXC	UDRE	FE	OR	-	-	-	page 55
$0A ($2A)	UCR	RXCIE	TXCIE	UDRIE	RXEN	TXEN	CHR9	RXB8	TXB8	page 56
$09 ($29)	UBRR	UART Baud Rate Register								page 58
$08 ($28)	ACSR	ACD	-	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	page 59
...	Reserved									
$00 ($20)	Reserved									

Η μνήμη δεδομένων EEPROM (Electrical Erasable Programmable Read Only Memory)

Ο AT90S8515 περιέχει 512 bytes μνήμη δεδομένων EEPROM . Είναι οργανωμένη σε ξεχωριστό χώρο δεδομένων.Η διάρκεια ζωής της είναι περίπου το λιγότερο 100.000 κύκλοι εγγραφής / διαγραφής. Για την πρόσβαση της CPU στην μνήμη EEPROM χρησιμοποιούνται είδικοι καταχωρητές οι οποίοι περιγράφονται λεπτομερώς στο data sheet του μικροελεγκτή.

2.6 Πηγές αρχικοποίησης (RESET) του AT90S8515.

Κατά την διάρκεια ενός reset όλοι οι 1/0 καταχωρητές σεττάρονται στις αρχικές τους τιμές και το πρόγραμμα αρχίζει να εκτελείται από την διεύθυνση $000.
Παρακάτω φαίνεται το κύκλωμα που χρησιμοποιείται στον AT90S8515 για reset.



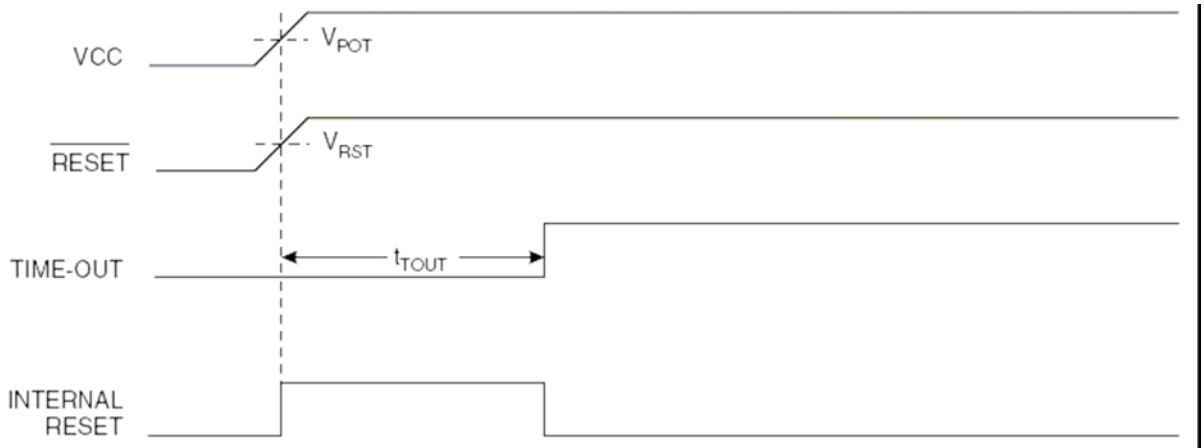
Ο AT90S8515 διαθέτει 3 διαφορετικές πηγές που μπορούν να κάνουν Reset στον μικροελεγκτή μας:

- **Power-on reset (POR):** Ο μικροελεγκτής αρχικοποιείται όταν τροφοδοτηθεί με τάση σωστή. Το κύκλωμα Power-on reset (POR) εξασφαλίζει ότι ο μικροελεγκτής δεν θα λειτουργήσει μέχρις ότου η τάση Vcc φτάσει σε ένα ασφαλές επίδεδο. Το όριο αυτό τάσης ονομάζεται Power-on reset Threshold voltage (Vpot) και φαίνεται στον πίνακα που ακολουθεί .Όπως φαίνεται από το προηγούμενο σχήμα γίνεται χρήση ενός timer ο οποίος χρονίζεται από τον ταλαντωτή του Watchdog timer , και ο οποίος δημιουργεί μια χρονική καθυστέρηση μετά από ένα Power-on reset και εμποδίζει τον μικροελεγκτή να λειτουργήσει για κάποιο χρονικό διάστημα μετά φυσικά που η τάση Vcc έχει φτάσει στην τιμή της Threshold Voltage – Vpot,ανεξάρτητα από τον ρυθμό ανόδου της Vcc.Το συνολικό χρονικό διάστημα καθυστέρησης που προαναφέραμε είναι ίσο με την περίοδο του Power on reset-tpor + χρονική καθυστέρηση που δημιουργεί ο timer-tout (timer out period).Το bit ασφαλείας με το όνομα FSTRT που βρίσκεται στην μνήμη Flash μπορεί να προγραμματιστεί ώστε να δίνει έναν πιο σύντομο χρόνο έναυσης λειτουργίας του μικροελεγκτή μειώνοντας το tout. Αυτό φαίνεται στον πίνακα που ακολουθεί . Ο ακροδέκτης του reset είναι εσωτερικά συνδεμένος μέσω μιας pull-up αντίστασης , στη Vcc. Έτσι το pin αυτό μπορούμε να μην το συνδέσουμε πουθενά εάν δεν χρειάζεται να χρησιμοποιήσουμε εξωτερικό reset. Κρατώντας το pin σε στάθμη low για ένα χρονικό διάστημα μετά που έχει εφαρμοστεί τάση Vcc, η περίοδος tpor μπορεί να μεγαλώσει . Όλα αυτά που προαναφέραμε φαίνονται παρακάτω με την βοήθεια πίνακα και διαγραμμάτων:

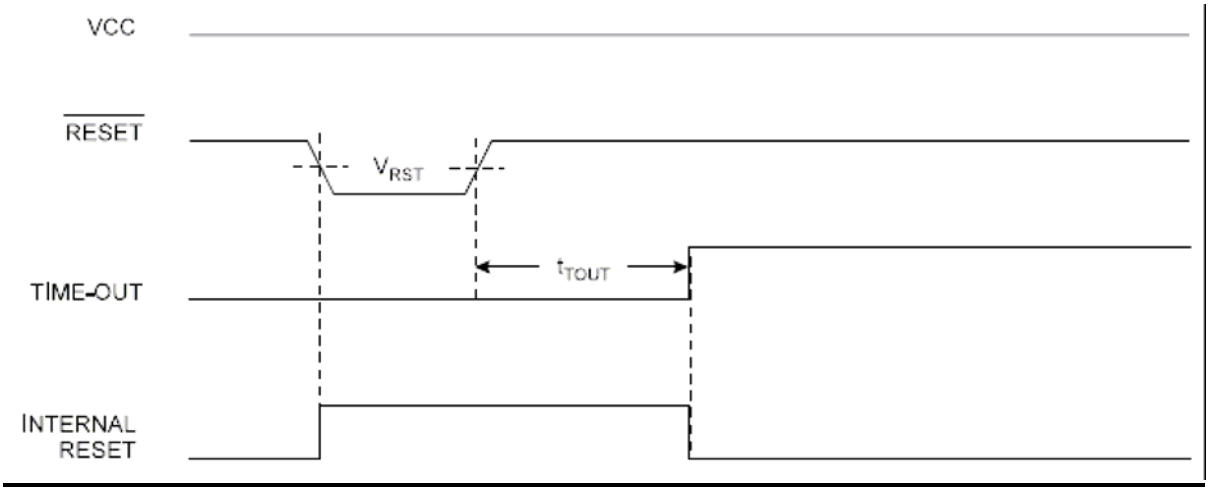
Table 3. Reset Characteristics

Symbol	Parameter	Min	Typ	Max	Units
V_{POT} (Note:)	Power-on Reset Threshold Voltage (rising)	0.8	1.2	1.6	V
	Power-on Reset Threshold Voltage (falling)	0.2	0.4	0.6	V
V_{RST}	$\overline{\text{RESET}}$ Pin Threshold Voltage	–	–	0.9 V_{CC}	V
t_{TOUT}	Reset Delay Time-out Period FSTRT Unprogrammed	11.0	16.0	21.0	ms
t_{TOUT}	Reset Delay Time-out Period FSTRT Programmed	0.25	0.28	0.31	ms

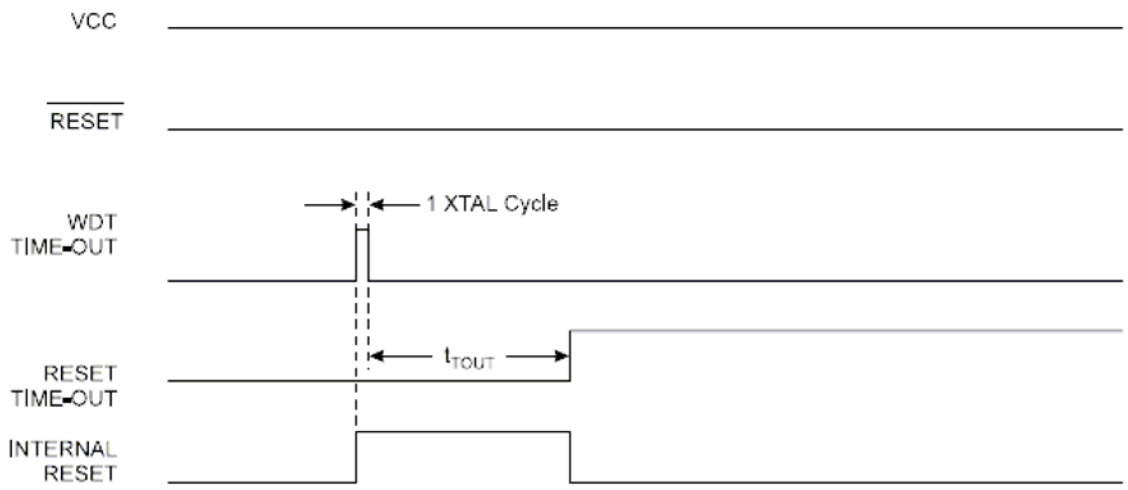
Note: The Power-on Reset will not work unless the supply voltage has been below V_{POT} (falling).



- **Εξωτερικό reset** Ένα εξωτερικό reset μπορεί να επιτευχθεί εάν δώσουμε Low στο reset pin του μικροελεγκτή, και το διατηρήσουμε εκεί για περίπου δύο κύκλους ρολογιού. Όταν ανερχόμενο στην θετική του ακμή φτάσει την τιμή της Threshold voltage-VRST, ο timer καθυστέρησης που είδαμε προηγουμένως, ενεργοποιείται και πάλι και ο μικροελεγκτής ενεργοποιείται μετά από χρονικό διάστημα tout. Βλέπουμε το παρακάτω σχήμα:



- **Watchdog reset** Όταν ο watchdog timer μετρήσει μέχρι το τέλος δημιουργεί έναν σύντομο παλμό διαρκείας περίπου 1XTAL.Στο falling edge του παλμού αυτού αρχίζει και μετρά ο timer καθυστέρησης και έτσι πάλι σε χρονικό διάστημα tout αρχίζει να λειτουργεί ο μικροελεγκτής. Βλέπουμε το παρακάτω σχήμα:



2.7 Χειρισμός Interrupt

Ο AT90S8515 παρέχει 12-διαφορετικές-περιπτώσεις interrupt.Καθένα από τα interrupts αυτά καθώς και το reset έχει ξεχωριστό τομέα (vector) στην μνήμη προγράμματος. Όλοι οι τομείς αυτοί, που βλέπουμε στον παρακάτω πίνακα, βρίσκονται στις χαμηλότερες διευθύνσεις στην μνήμη προγράμματος.

Vector No.	Program Address	Source	Interrupt Definition
1	$000	RESET	External Reset, Power-on Reset and Watchdog Reset
2	$001	INT0	External Interrupt Request 0
3	$002	INT1	External Interrupt Request 1
4	$003	TIMER1 CAPT	Timer/Counter1 Capture Event
5	$004	TIMER1 COMPA	Timer/Counter1 Compare Match A
6	$005	TIMER1 COMPB	Timer/Counter1 Compare Match B
7	$006	TIMER1 OVF	Timer/Counter1 Overflow
8	$007	TIMER0, OVF	Timer/Counter0 Overflow
9	$008	SPI, STC	Serial Transfer Complete
10	$009	UART, RX	UART, Rx Complete
11	$00A	UART, UDRE	UART Data Register Empty
12	$00B	UART, TX	UART, Tx Complete
13	$00C	ANA_COMP	Analog Comparator

Σε όσο πιο χαμηλή διεύθυνση βρίσκεται ο τομέας ενός interrupt τόσο μεγαλύτερη προτεραιότητα έχει. Για παράδειγμα το reset έχει την μεγαλύτερη προτεραιότητα και μετά ακολουθεί το εξωτερικό interrupt στον ακροδέκτη INTO, κ.ο.κ.

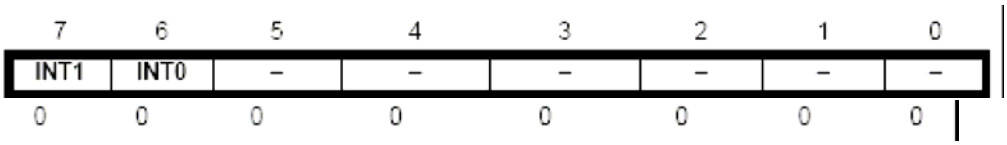
2.8 Εξωτερικά Interrupt του μικροελεγκτή AT90S8515 (INTI, INTO)

Ενδιαφέροντες Καταχωρητές: SREG, GIMSK, MCUCR, GIFR

SREG(STATUS REGISTER)

Bit 7-I (Global Interrupt Enable)
Τοποθετείται στο <<1>> αν θέλουμε να ενεργοποιήσουμε τα Interrupt του μικροελεγκτή .Ο έλεγχος τότε για τα εξωτερικά Interrupt εκτελείται στους Interrupt Mask Register: GIMSK (και αντίστοιχα TIMSK για τα χρονικά Interrupt των timers).
Το I bit γίνεται "0" από το Hardware μετά που θα συμβεί ένα Interrupt και σεττάρεται <<1>> με την εντολή RETI για να μπορεί να δεχθεί νέα Interrupts.
Για να μην πειράξουμε τα υπόλοιπα bit του SREG δίνουμε (SREG |=OX80;)

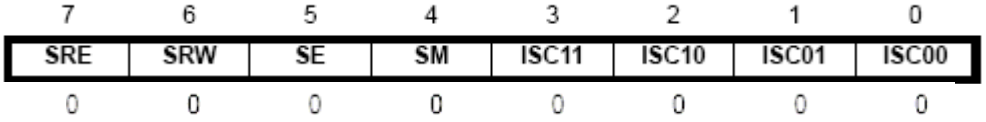
GIMSK (General Interrupt Mask R egister):



Bit 7-INT1 (External Interrupt Reguest 1 enable):
Όταν το INT 1 Bit τοποθετηθεί <<1>> και το 1 bit του SREG είναι και αυτό <<1>> τότε είναι ενεργοποιημένη η είσοδος εξωτερικού INT 1 που βρίσκεται στην πόρτα D (D3).
GIMSK \ = 0X80 or 0X40 for INTO Input.

MCUCR (MCU General control register):

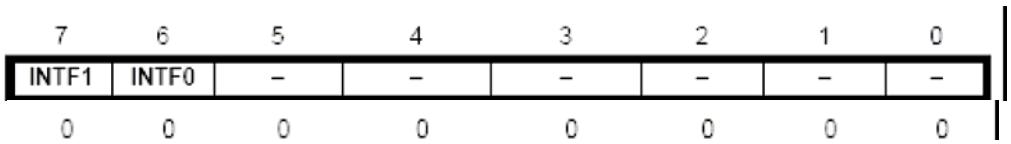
Ο παλμός που εισέρχεται στο πόδι αυτό μπορεί να επιδράσει στον μικροελεγκτή σαν διακοπή,με την αιχμή ανόδου ,καθόδου ή και με επίπεδο τάσης του. Για το λόγο αυτό σεττάρουμε με '1s'τον καταχωρητή **MCUCR** στα bit **ISC11** και **ISC10** για το **INT1** και στα **ISC01** και **ISC00** για το **INTO**.



ISC 11	ISC 1 0	
0	0	low level of INT 1 genarates an interrupt Request
0	0	
1	1	Falling edge of INT1
1	1	Rising edge of INT1

Στην πραγματικότητα μπορεί να συμβεί Interrupt σε ενα πόδι (INT1 , INT0) ακόμα και αν το pin έχει τοποθετηθεί σαν έξοδος.
Bit6-INT0 (External Interrupt Request 1 enable): Όμοια με το INT1 παραπάνω .

GIFR (General Interrupt FLAG Register):



Bit 7-INTF1 (External Interrupt Flag 1):
Όταν ένα γεγονός (μέτωπο παλμού) έλθει στο pin INT1 του μικροελεγκτή, προκαλεί ένα interrupt request και αμέσως το Bit INTF1 του GIFR γίνεται '1'. Αν το 1-bit του SREG και το INT1 B IT του GIMSK είναι σετταρισμένα στο '1' η MCU στέλνει (jump) την εκτέλεση στο Interrupt Vector που βρίσκεται στη διεύθυνση $002 της μνήμης Flash (4K X 16 bit program memory).
Η σημαία γίνεται «0» όταν η ρουτίνα του Interrupt εκτελεστεί. Σε άλλη περίπτωση η σημαία μπορεί να σβηστεί, ξαναγράφοντας ένα «1» στο αντίστοιχο Bit.

Bit6-INTF0 (External Interrupt Flag 0)

Όμοια με την προηγούμενη σημαία , αλλά για το INT0.

Συνάρτηση στην C (Compiler)

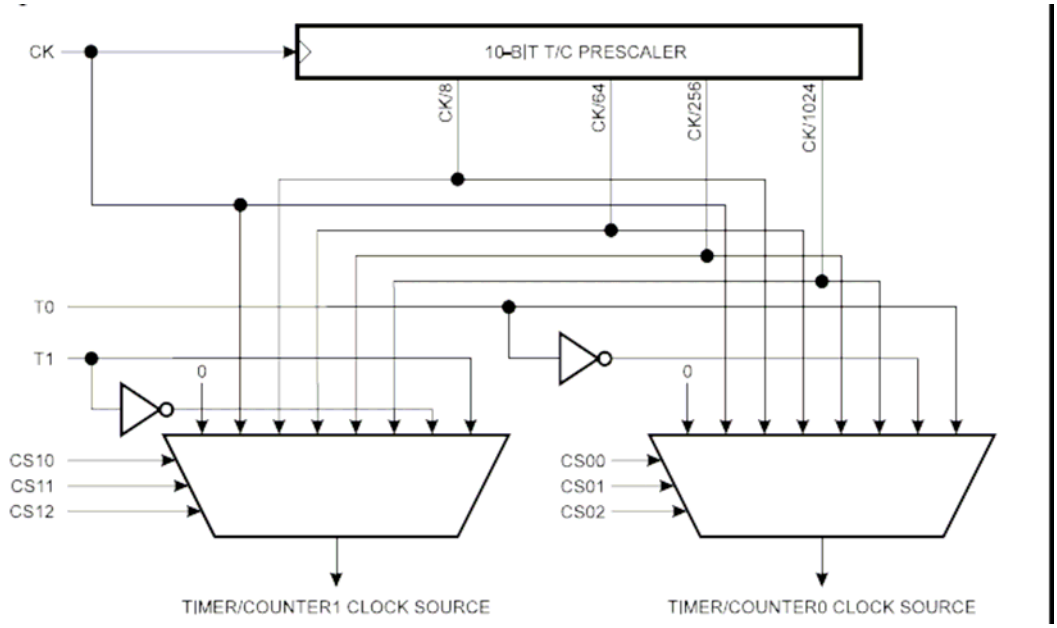
```
Interrupt [INT1_vect] void INT1_interrupt (void)  // για τον INT1
Interrupt [INTO_vect] void INT1_interrupt (void)  // για τον INTO


ή

pragma vector=INT1_vect   //νεος Compiler
- interrupt void Buton1(void)
```

Παράδειγμα για τους καταχωρητές των εξωτερικών Interrupts

```
MCUCR& =0X7F;   // Απενεργοποίηση λειτουργιών εξωτέρικης μνήμης

MCUCR| =0X0C;  // Rising edge of INT1 pulse
GIFR=0X80;       //   Δίνω τιμή 1 στο  INTF1 ώστε αν έχει ενεργοποιηθεί το
interrupt να απενεργοποιηθεί


SREG | =0X80;    // Γενική ενεργοποίηση των interrupt
GIMSK | =0X80;  // Ενεργοποίηση Εξωτερικών interrupt
```

2.9 Χρονιστές / Μετρητές (Timers / counters)

Ο AT90S8515 παρέχει δύο γενικής χρήσης χρονιστές / μετρητές (timers / counters). Έναν 8 bit και έναν 16 bit. Ο καθένας ξεχωριστά διαθέτει ρύθμιση με την χρήση του ίδιου 10 bit timer που λειτουργεί ως προδιαρέτης (prescaler). Έτσι ο χρονισμός με τον οποίο μετράνε ως timers μπορεί να είναι CK, CK/8, CK/64, CK/256, CK/1024.



Ο16-bit χρονιστής έχει επιπλέον λειτουργίες σύλληψης και σύγκρισης ενώ μπορεί να χρησιμοποιηθεί σαν 8, 9 και 10 bit διαμορφωτής εύρους παλμών (Pulse Width Modulator) και μάλιστα διπλός , δηλαδή έχουμε δύο διαμορφωτές εύρους παλμών .
Οι συχνότητες που μπορούμε να πάρουμε για την διαμόρφωση του εύρους παλμών είναι συγκεκριμένες και φαίνονται παρακάτω . Βέβαια με την βοήθεια προγραμματισμού μπορούμε να ρυθμίσουμε εμείς και λειτουργία , σε άλλες συχνότητες .

8-bit

	CK	CK/8	CK/64	CK/256	CK/1024
F pwm	7,843KHz	980,392Hz	122,549Hz	30,637Hz	7,659Hz
Ts	127,5μsec	1020μsec	8160μsec	32,64msec	3056msec

9-bit

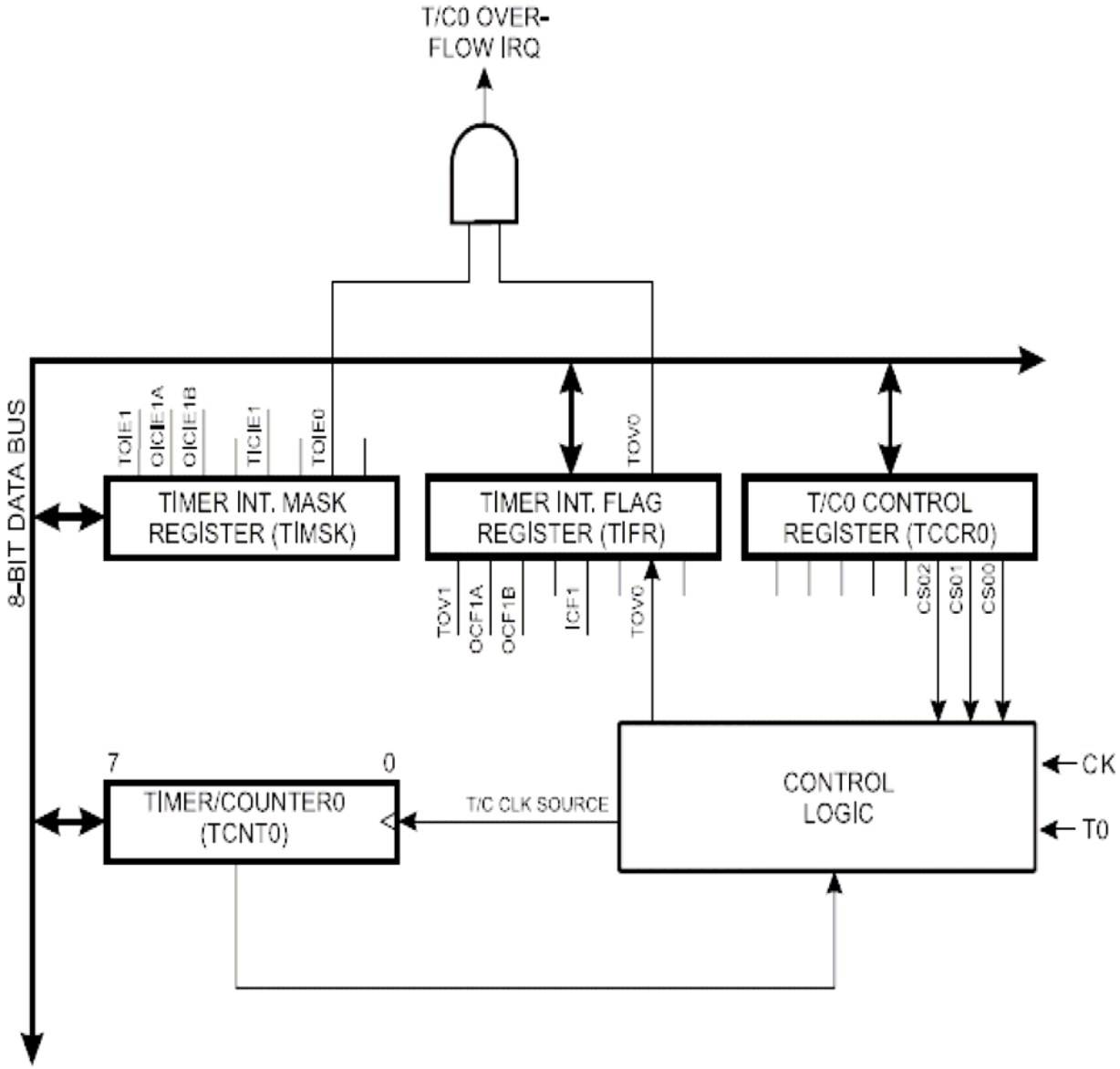
	CK	CK/8	CK/64	CK/256	CK/1024
F pwm	3,914KHz	489,237Hz	61,155Hz	15,289Hz	3,822Hz
Ts	255,433μsec	2043,999msec	16,352msec	65,407msec	261,643msec

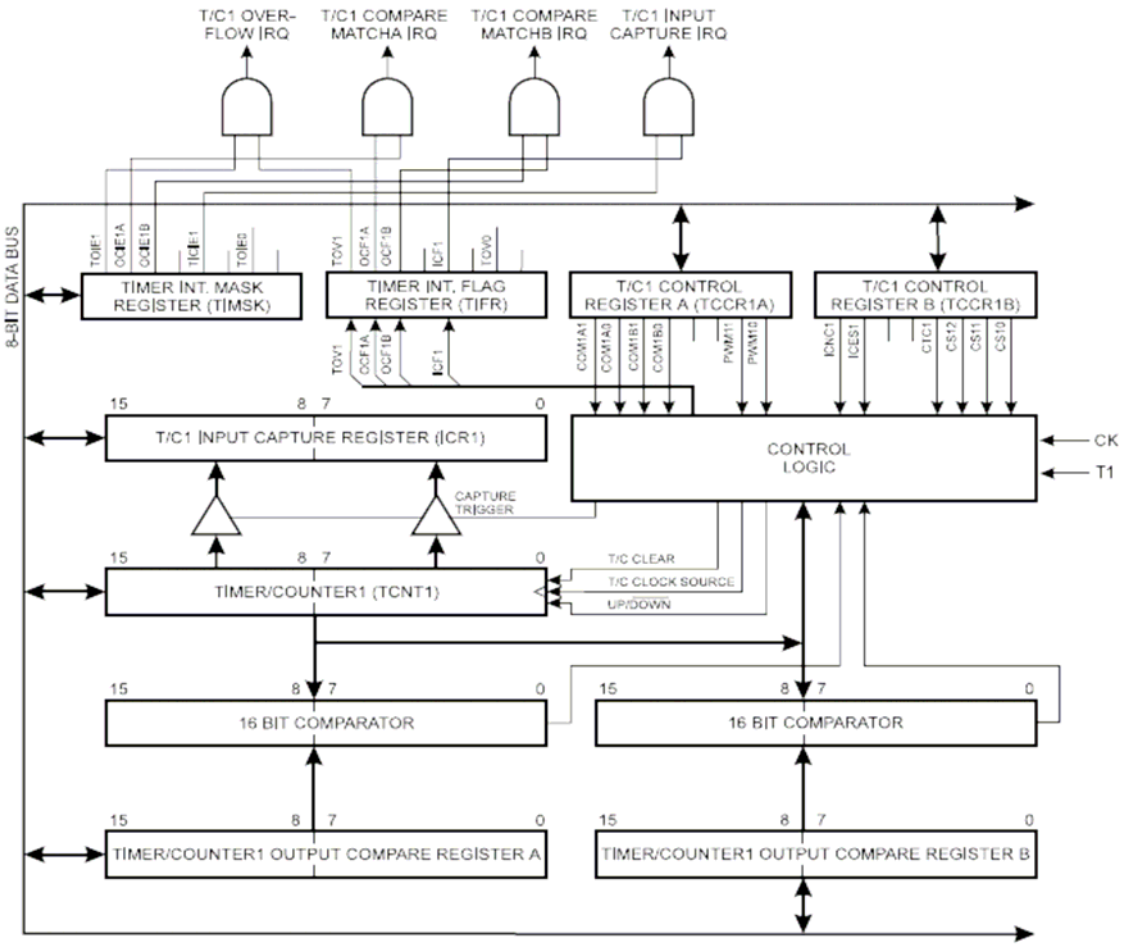
10-bit

	CK	CK/8	CK/64	CK/256	CK/1024
F pwm	1,955KHz	244,379Hz	30,547Hz	7,637Hz	1,909Hz
Ts	511,590μsec	4092,044μsec	32,736msec	135,74msec	523,834msec

Ts: η συνολική περίοδος του κάθε παλμού.

Ως counters μπορούν να μετρήσουν εξωτερικά γεγονότα , με την βοήθεια των εξωτερικών pin (T1, T0) που βρίσκονται στην θύρα B.
Εάν θέλουμε να μετρήσουμε εξωτερικούς παλμούς ή όπως αλλιώς λέμε «εξωτερικό ρολόϊ » τότε για να πετύχουμε σωστή δειγματοληψία θα πρέπει ο ελάχιστος χρόνος μεταξύ δύο μεταβάσεων του « εξωτερικού ρολογιού » να είναι το λιγότερο ίσος με μια περίοδο ρολογιού με το οποίο χρονίζεται η CPU του μικροελεγκτή μας.Το σήμα από το « εξωτερικό ρολόϊ » δειγματοληπτείται στην ανερχόμενη ακμή του ρολογιού που χρονίζει τον μικροελεγκτή μας .
Παρακάτω βλέπουμε τα block διαγράμματα των timers / counters :


T/C0 OVER-
FLOW IRQ
8-BIT DATA BUS
TOIE1
OICIE1A
OICIE1B
TICIE1
TOIE0
TOV0
TIMER INT. MASK
REGISTER (TIMSK)
TIMER INT. FLAG
REGISTER (TIFR)
T/C0 CONTROL
REGISTER (TCCR0)
TOV1
OCF1A
OCF1B
ICF1
TOV0
CS02
CS01
CS00
7
0
TIMER/COUNTER0
(TCNT0)
T/C CLK SOURCE
CONTROL
LOGIC
CK
T0



2.10 Χρονικά Interrupt του μικροελεγκτή AT90S8515 με TIMERS

Ο Timer/Counter1 TCNT1 (16-Bit)

Ο Timer/Counter1 μπορεί να μετρά χρόνο (λειτουργία timer) ή να απαριθμεί γεγονότα (παλμούς) που έρχονται στο εξωτερικό πόδι T1 του μικροελεγκτή (λειτουργία counter).
Όταν λειτουργεί σαν χρονιστής μπορεί να πάρει Clock από τον κρύσταλλο του μικροελεγκτή ή και μικρότερο Clock (prescaled CK) Μετά από 65535 βήματα αύξησης γεμίζει και λέμε ότι υπερχειλίζει . Τη στιγμή αυτή μπορεί να δώσει ένα Interrupt στον μικροελεγκτή , οπότε διαπιστώνουμε ότι έχει περάσει ένα υπολογισμένο χρονικό διάστημα . Βέβαια μπορεί να σταματήσει να μετρά (ή και να ξεκινήσει να μετρά από κάποια τιμή > 0) αν χρησιμοποιήσουμε τους καταχωρητές ελέγχου του TCCR1A και TCCR1B .Η ενεργοποίηση / απενεργοποίηση του timer / counter1 εξαρτάται από τον TIMSK.

Ενδιαφέροντες καταχωρητές : SREG, TIMSK, TCCR1A, TCCR1B, TIFR

SREG (Status Regisrer): Ενεργοποιούμε όμοια με τα εξωτερικά Interrupt

TIMSK (Timer/counter Interrupt Mask Register):

7	6	5	4	3	2	1	0
TOIE1	OCIE1A	OCIE1B	–	TICIE1	–	TOIE0	–
0	0	0	0	0	0	0	0

Bit7TOIE1 (Timer/counter1 Overflow Interrupt Enable)

Όταν το bit αυτό σεταριστεί στο «1» και το 1-bit του SRG είναι 1τοτε το Interrupt της υπερχείλισης του timer/counter1 υπερχειλίσει μετρώντας χρόνο ή γεγονότα από το πόδι τότε εκτελείται το Interrupt που βρίσκεται στη διεύθυνση $006 της Flash memory και σετάρεται αυτόματα η σημαία TOV1 (BIT7) του καταχωρητή σημαιών TIFR.

Bit1-TOIE0 (Timer/counter0 Overflow Interrupt Enable)

Παρόμοια με το προηγούμενο bit 7, αλλά για τον Timer/counter0.

TIFR (Timer/Counter Interrupt Flag Register):

7	6	5	4	3	2	1	0
TOV1	OCF1A	OCIFB	–	ICF1	–	TOV0	–
0	0	0	0	0	0	0	0

Bit 7-TOV1

Η σημαία αυτή του καταχωρητή σημαιών TIFR γίνεται «1» όταν αύτος υπερχείλιση και μηδενίζεται από το Hardware όταν εκτελεστεί η αντίστοιχη ρουτίνα εξυπηρέτησης του interrupt.
Εναλλακτικά μπορούμε να την απενεργοποιήσουμε γράφοντας ένα «1» στο αντίστοιχο bit .

Όταν το 1-bit του SREG, το TOIE1 του TIMSK και το TOV1 είναι «1» , το Overflow Interrupt του Timer / Counter1 εκτελείται.
Όταν ο Timer / Counter1 είναι σε λειτουργία PWM , η σημαία TOV1 γίνεται «1» ,το όταν ο counter αλλάζει φορά μέτρησης.

Bit 1-TOV0 : Overflow Flag αντίστοιχα για τον Timer/Counter 0.

TCCR1B (Timer / Counter1 Control Register B):

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

Bits 2,1,0-CS12, CS11, CS10: Επιλογή συχνότητας ρολογιού

Κάθε Timer είναι με βάση το CLK του ρολογιού του μικροελεγκτή .Αν το CLK του ρολογιού είναι 4MHz τότε το βήμα αύξησης του Timer είναι: 1/(4MHz)=0,25μsec.
Επομένως αν ο Timer είναι 16 bit, για να υπερχειλίσει θα κάνει 65535 βήματα αύξησης x
Στη περίπτωση που θέλουμε να υπερχειλίζει πιο αργά , επιλέγουμε χαμηλότερη συχνότητα ρολογιού οπότε με CK/64 το βήμα είναι πιο αργό , δηλαδή 16 μsec και θα υπερχειλίσει αργότερα , δηλαδή σε 1,048576 sec.

CS12	CS11	CS10	Description		Βήμα (μsec)	Υπερχείληση (sec)
0	0	0	Stop,the Timer/Counter 1 is stopped			
0	0	1	CK	(4MHz)	0,25	0.016384
0	1	0	CK/8	(500KHz)	2	0,131072
0	1	1	CK/64	(62,5 KHz)	16	1.048576
1	0	0	CK/256	(15,625KHz)	64	4,194307
1	0	1	CK/1024	(3,90625KHz)	256	16,77721
1	1	0	External pin T1, Falling edge			
1	1	1	External pin T1, Rising edge			

Η κατάσταση **Stop** δίνει τη δυνατότητα να κάνουμε Enable/Disable τον Timer 1 ενώ οι δύο τελευταίες περιπτώσεις του πίνακα αφορούν την εξωτερική πηγή παλμού ρολογιού που τροφοδοτεί τον Counter από το πόδι T1, με falling ή rising edge.

Συνάρηση:

```
Interrupt [TIMER1_OVEF1_vect] void TRM_1_OVF(void)  // για τον T1
Interrupt [TIMER0_OVEF0_vect] void TRM_0_OVF(void)  // για τον T0

ή pragma vector= TIMER1_OVF1_vect                   // για τον νέο compiler
  __interrupt void xronos1(void)
```

Καταχωρητές:

```
SREG | =0X80;   // Γενική ενεργοποίηση των Interrupt
TIMSK | =0X80;  // Ενεργοποίηση Overflow του Timer 1
                   ή TIMSK | = 0X02 για τον TIMER 0

TCCR1B = 0X03;   //   CK/64  ξεκίνημα μέτρησης
                              (υπερχείλιση σε 1,048sec)
         = 0X05;   //  CK/1024 (Υπερχείλιση σε 16,777sec)
```

Ο TCNT1 είναι ένας Up-Counter 16 Bit με δυνατοτότητα Read/Write.Αν του έχουμε γράψει μια τιμή και τον ενεργοποιήσουμε, τότε συνεχίζει από την τιμή α΄θτη να μετρά μέχρι να υπερχειλίσει .Στην πραγματικότητα αποτελείται από δύο 8bit READ/WRITE Registers ((TCNT1L και TCNT1H) που γράφονται ξεχωριστά από τον μικροελεγκτή με πρώτο πάντα τον TCNT1H ενώ διαβάζονται με πρώτο πάντα τον TCNT1L.

Όταν λειτουργεί σαν counter το εξωτερικό σήμα λαμβάνεται από το Pin T1 στην ανερχόμενη αιχμή του παλμού εισόδου.

Ο Timer/Counter0 (TCNT0) 8 -Bit

Ο Timer/counter 0 μπορεί να μετρά χρόνο (λειτουργία timer) ή να απαριθμεί γεγονότα (παλμούς) που έρχονται στο εξωτερικό πόδι T0 του μικροελεγκτή (λειτουργία counter).

Όταν λειτουργεί σαν χρονιστής μπορεί να πάρει Clock από τον κρύσταλλο του μικροελεγκτή, ή και μικρότερο (prescaled CK) .Μετα από 256 βήματα αύξησης γεμίζει και λέμε ότι υπερχειλίζει . Τη στιγμή αυτή μπορεί να δώσει ένα Interrupt στον μικροελεγκτή,οπότε διαπιστώνουμε ότι έχει περάσει ένα υπολογισμένο χρονικό διάστημα .

Βέβαια μπορεί να σταματήσει να μετρά (ή και να ξεκινήσει να μετρά από κάποια τιμή >0)

αν χρησιμοποιήσουμε τον καταχωρητή ελέγχου TCCR0.η σημαία Overflow βρίσκεται στον TIFR (TOV0) ενώ τα σήματα βρίσκονται στον TCCR0.

Η ενεργοποίηση / απενεργοποίηση του εξαρτώνται από την τοποθέτηση του TIMSK (TOIE0) .
Όταν λειτουργεί σαν counter το εξωτερικό σήμα λαμβάνεται στην ανερχόμενη αιχμή του παλμού ρολογιού.

TCCR0 (Timer / Counter0 Control Register)

7	6	5	4	3	2	1	0
–	–	–	–	–	CS02	CS01	CS00
0	0	0	0	0	0	0	0

Bits 2,1,0-CS02, CS01, CS00 : Επιλογή συχνότητας ρολογιού

CS 02	CS 01	CS 00	Description		Βήμα (μsec)	Υπερχείληση (msec)
0	0	0	Stop,the Timer/Counter 0 is stopped			
0	0	1	CK	(4MHz)	0,25	0.064
0	1	0	CK/8	(500KHz)	2	1,024
0	1	1	CK/64	(62,5 KHz)	16	4,096
1	0	0	CK/256	(15,625KHz)	64	16,384
1	0	1	CK/1024	(3,92625KHz)	256	65,536
1	1	0	External pin T0, Falling edge			
1	1	1	External pin T0, Rising edge			

Η κατάσταση **stop** δίνει τη δυνατότητα να κάνουμε Enable/Disable τον Timer 1 ενώ οι δύο τελευταίες περιπτώσεις του πίνακα αφορούν την εξωτερική πηγή παλμού ρολογιού που τροφοδοτεί τον Counter από το πόδι T0, με falling ή rising edge.

Ο TCNT0 είναι ένας Up-Counter 8 BIT με δυνατότητα Read/write. Αν του έχουμε γράψει αρχικά μια τιμή >0 και τον ενεργοποιήσουμε να μετρήσει, τότε συνεχίζει από την τιμή αυτή να μετρά .Το ίδιο ισχύει και για τον 16 bit TCNT1.

Παράδειγμα υπολογισμού χρόνου 1 sec υπερχείλισης του TIMER1:

Από τον πίνακα παίρνουμε την περίπτωση CK/64 οπότε:

4MHz/64 = 62,5KHz => 1/f = 1 / 62,5KHz = 16μsec
=> 16μsec x 65535 = 1,048576 sec (η μία υπερείλιση)

Για να επιτύχουμε ακριβώς 1 sec πρέπει να προφορτώσουμε τον Timer1 με κάποια τιμ'ή ώστε να μην ξεκινά από την τιμή 0 . Ο χρόνος λοιπόν των 0,048576 sec πρέπει να αφαιρεθεί. Ο χρόνος αυτός αντιστοιχεί σε βήματα αύξησης του TIMER1:

48576 / 16μsec = 3036 βήματα λιγότερα (αρχική τιμή φόρτωσης) ή 0BDC

Οπότε ο Timer θα κάνει 65536-3036 = 62500 βήματα .
(62500x16 = 1,000000 sec)
Ο Timer λοιπόν θα ξεκινάει από την τιμή 3036 και θα τελειώνει στην τιμή 65535 ή από 0BDC μέχρι την FFFF.
Προφορτώνουμε λοιπόν τον TCNTH=0x0B και τον TCNTL=0xDC

2.11 Έλεγχος του χρόνου υπερχείλισης του TIMER 1 με COMPARE

Μπορούμε ακόμα να χρησιμοποιήσουμε και την διαδικασία αλλά μόνο για υπερχείλιση του TIMER1 όχι στον TIMER 0.

Υπάρχουν δύο 16 bit καταχωρητές σύγκρισης **OCR1A** και **OCR1B** οι οποίοι σχετίζονται με τον Timer1.

Έστω ότι καταχωρούμε κάποια τιμή π.χ 3EFB στον OCR1A και στην συνέχεια ενεργοποιούμε τον Timer1.Οταν ο Timer1 αυξανόμενος φτάσει την τιμή 3EFB, παράγεται χρονικό interrupt:

```
#pragma vector=TIMER1_COMPA_vect
__interrupt void XRONOS1(void)
```

Για την καταχώρηση της 16 bit τιμής στον **OCR1A** χρησιμοποιούμε δύο 8 bit καταχωρητές: Τον OCR1AH και OCR1AL.
Καταχωρούμε το 3E στον OCR1AH πρώτα και μετά την τιμή FB στον OCR1AL.
Ο καταχωρητής TCCR1B πρέπει τότε να τοποθετηθεί με τι bit 3 (CTC1) ‘1’.

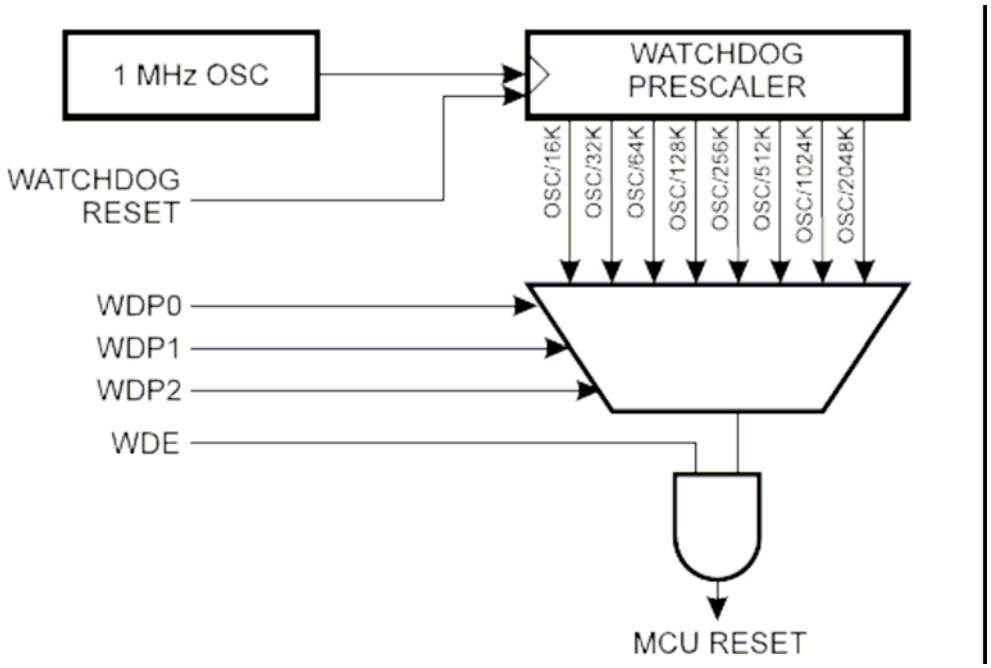
Για τον OCR1B οι καταχωρητές αυτοί είναι ο OCR1BH και OCR1BL αντίστοιχα, ενώ η αντίστοιχη συνάρτηση interrupt φαίνεται παρακάτω:

```
#pragma vector=TIMER1_COMPB_vect
__interrupt void XRONOS1(void)
```

Μπορούμε λοιπόν να ορίσουμε δύο τιμές για Compare του TIMER1 και να παράγουμε δύο καθορισμένους χρόνους υπερχείλισης.

2.12 χρονιστής επίβλεψης (Watchdog timer)

Ο χρονιστής επίβλεψης είναι ένας ειδικός χρονιστής με συγκεκριμένη λειτουργία .Χρησιμοποιείται συνήθως για να αποτρέπει κολλήματα του προγράμματος .Το block διάγραμμα του φαίνεται παρακάτω:



Λειτουργεί ως εξής :Όταν ενεργοποιηθεί , τότε αυξάνει ένας εσωτερικός μετρητής με κάποια συχνότητα . Εάν ο χειριστής του προγράμματος δεν κάνει reset στον μετρητή μέσω του προγράμματος , τότε ο μετρητής υπερχειλίζει και τότε γίνεται reset στον μικροελεγκτή .Γίνεται λοιπόν κατανοητό ότι ο δημιουργός του προγράμματος πρέπει να κάνει reset στον μετρητή αυτό σε τακτά χρονικά διαστήματα μέσα στο πρόγραμμα , διαστήματα που εξάρτιουνται από την περίοδο που έχει ρυθμιστεί ο μετρητής να κάνει υπερχείλιση (time-out period).
Το ζητούμενο και τελικώς και η χρησιμότητα του χρονιστή επιτήρησης είναι ότι είναι προτιμότερο Εάν κολλήσει το πρόγραμμα , που τρέχει σε έναν μικροελεγκτή που ελέγχει κάποιο σύστημα , να ξεκινήσει ξανά να τρέχει το πρόγραμμα από την αρχή παρά να εκτελεί απρόβλεπτες λειτουργίες .
Στον AT90S8515 ο χρονιστής αυτός χρονίζεται από έναν εσωτερικό ταλαντωτή που παράγει παλμούς συχνότητας 1MHz .Αυτή είναι η τυπική τιμή για τάση Vcc=5 V . Ο χρόνος time-out period μπορεί να πάρει τιμές 16 ,32 ,64 ,128 ,256 ,512 ,1024 καI 2048 ms .

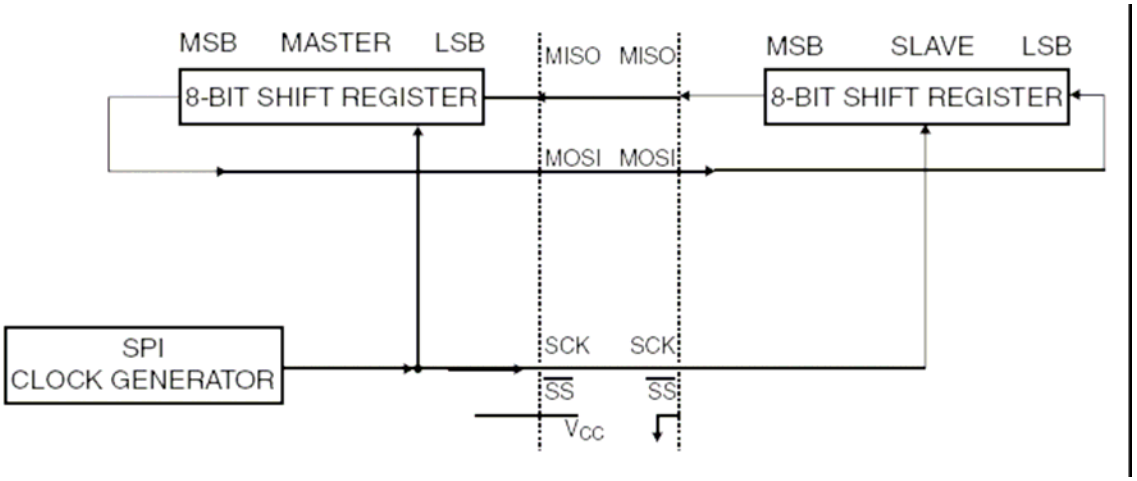
2.13 Λίγα λόγια για το SPI (Serial Peripheral Interface)

Το SPI είναι ένα πρωτόκολλο σύγχρονης σειριακής επικοινωνίας μεταξύ δύο συσκευών ή δύο ολοκληρωμένων. Έχει γίνει σήμερα τόσο δημοφιλές που αρκετοί μικροελεγκτές το υποστηρίζουν ,συμπεριλαμβανομένων και των AVR .Το SPI υποστηρίζει έναν μεγάλο ρυθμό μεταφοράς δεδομένων περίπου 3 MHz .Μικροελεγκτές με ενσωματωμένο SPI μπορούν να επικοινωνήσουν με περιφερειακές συσκευές που περιλαμβάνουν και αυτές SPI .Σε περίπτωση όμως που ο μικροελεγκτής δεν περιέχει το συγκεκριμένο πρωτόκολλο , μπορεί να χρησιμοποιήσει τις γραμμές I/O και να μιμηθεί τους ακροδέκτες που χρησιμοποιούνται για το SPI , με κάποιο κόστος φυσικά στην ταχύτητα μεταφοράς δεδομένων .

Μια θύρα που χρησιμοποιείται για το SPI διαθέτει τα εξής σήματα :

- MISO: Master In Slave Out
- MOSI: Master Out Slave In
- SCK: Serial Clock
- SS: Select Signal

Έστω ότι έχουμε δυο μικροελεγκτές που επικοινωνούν μεταξύ τους με το SPI .Τότε ,όταν ο ένας έχει τον χαρακτήρα Master ο άλλος έχει τον χαρακτήρα Slave .Η μόνη διαφορά μεταξύ του Master και του Slave είναι ποιος παράγει το ρολόι χρονισμού .Ο Master παράγει το ρολόι χρονισμού και τα δεδομένα που εξέρχονται από τον ακροδέκτη MISO του Slave εισέρχονται σε αυτόν (τον Master) από τον δικό του ακροδέκτη MISO , ενώ την ιδία στιγμή δεδομένα εξέρχονται από αυτόν από τον ακροδέκτη MOSI και εισέρχονται στον Slave από τον δικό του ακροδέκτη MOSI. Δηλαδή βασικά οι δυο καταχωρητές ολίσθησης Master και Slave μπορούν να παρομοιαστούν με ένα 16-bit κυκλικό καταχωρητή ολίσθησης .
Αυτά φαίνονται παραστατικότερα στο παρακάτω σχήμα



Στον AT90S8515 το SPI έχει μέγιστη ταχύτητα 5Mbit ανά δευτερόλεπτο .

2.14 **UART (Universal Asynchronous Receiver Transmitter)**

Επειδή τα δεδομένα σε ένα μικροεπεξεργαστή είναι σε παράλληλη μορφή , μια σειριακή μεταφορά I/O πρέπει να ξεκινήσει και να τελειώσει με παράλληλα δεδομένα .

Η μετατροπή παράλληλης σε σειριακή μεταφορά γίνεται ως εξής .Τα παράλληλα δεδομένα φορτώνονται σε ένα καταχωρητή ολίσθησης .Στην συνέχεια ,αυτός χρονίζεται και τα δεδομένα βγαίνουν από το LSB του καταχώρηση ένα κάθε φορά (1 bit για κάθε κύκλο του clock) .Το πρώτο bit της σειριακής μεταφοράς είναι το LSB (ελάχιστα σημαντικό ψηφίο) των δεδομένων .Το δεύτερο είναι το επόμενο LSB κ.ο.κ

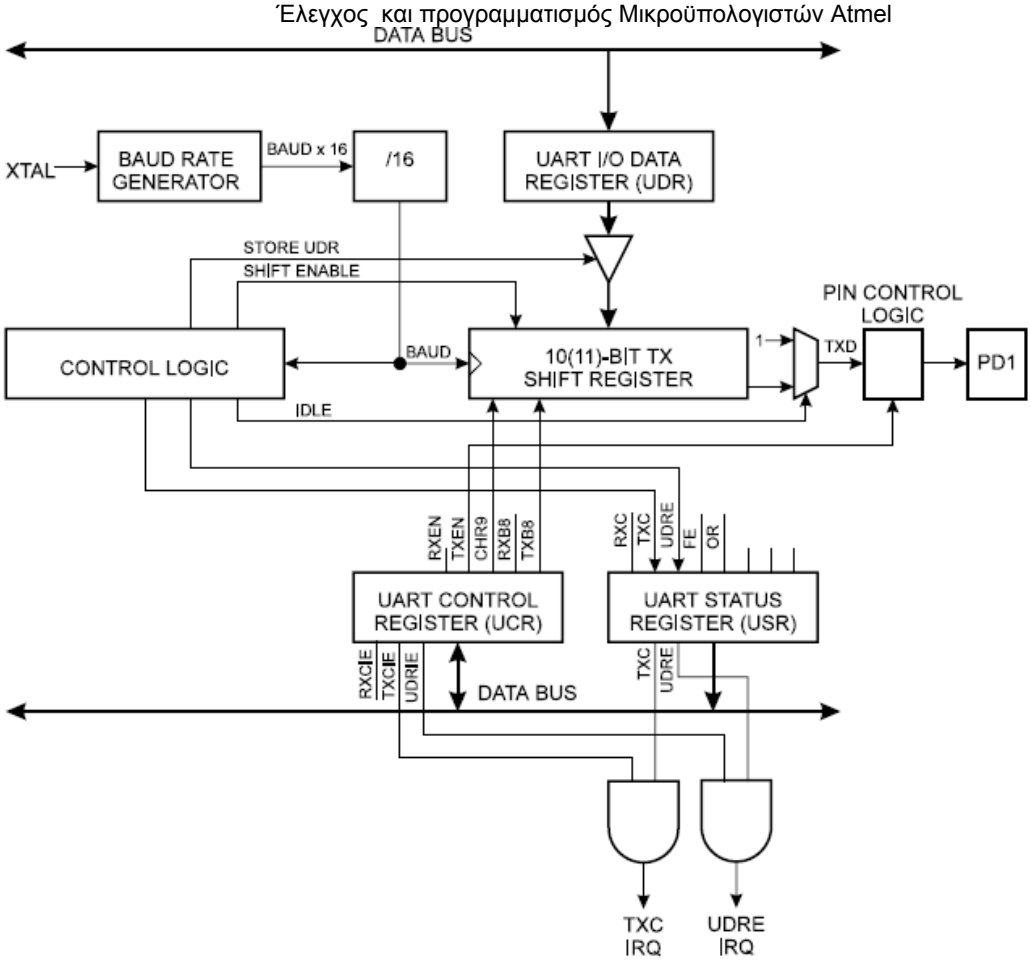
Η αντίθετη λειτουργία είναι η λήψη σειριακών δεδομένων και η μετατροπή τους σε παράλληλα .Τα σειριακά δεδομένα ολισθαίνουν σε έναν καταχώρηση ολίσθησης . Αφού όλα τα bits χρονιστούν στον καταχωρητή τα δεδομένα λαμβάνονται από αυτόν παράλληλα και μεταφέρονται στο σύστημα του μικροεπεξεργαστή.

Η συσκευή που εκτελεί αυτές τις μεταφορές καλείται UART .Η μονάδα αυτή είναι ένα ολοκληρωμένο κύκλωμα μεγάλης ολοκλήρωσης .Επιπλέον η μονάδα UART έχει λειτουργίες ελέγχου και ανίχνευσης .

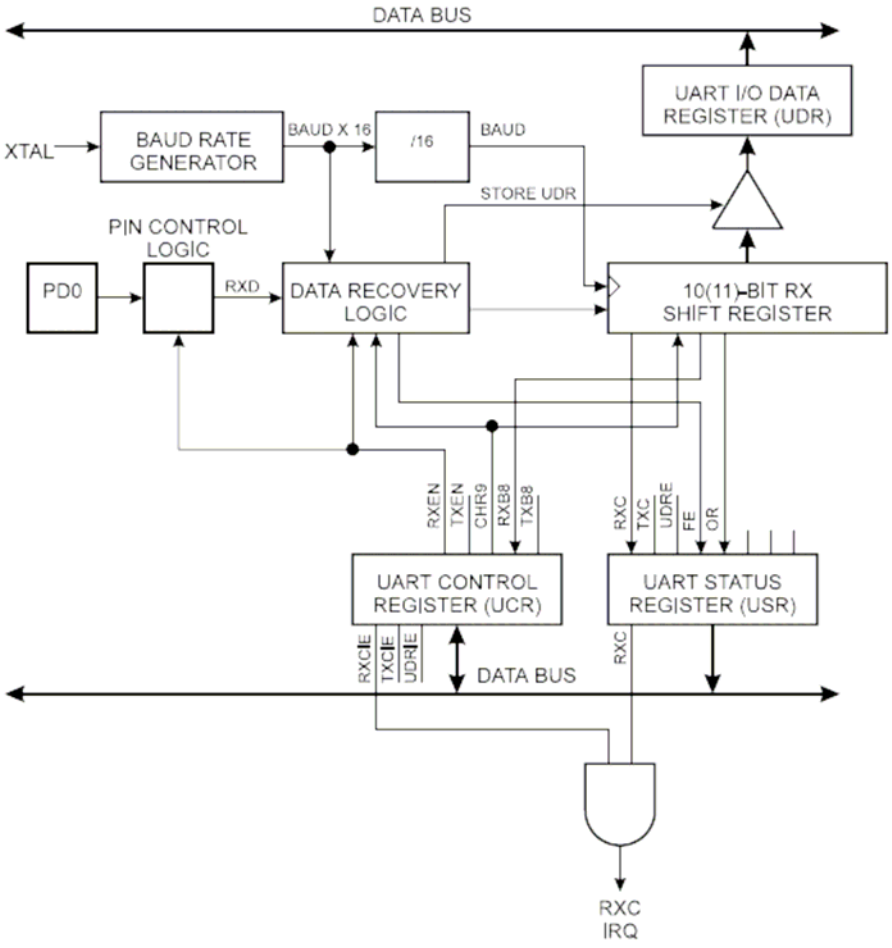
Η εκπομπή δεδομένων 8-bits ή 9-bits απαιτεί στην ουσία την εκπομπή 10 bits ή 11 bits αντίστοιχα .Το πρώτο bit λέει στην UART που λαμβάνει ότι έρχονται δεδομένα .Το bit εκκίνησης (start bit) είναι πάντα ένα λογικό μηδέν και το bit διακοπής (stop bit) .

Η ταχύτητα μεταφοράς της μονάδας UART καλείται baud-rate .Το baud-rate ή ρυθμός σηματοδότησης λέει ποσά bits εκπέμπονται ανά δευτερόλεπτο .Για παράδειγμα μια εκπομπή 1200 baud πραγματοποιείται με ρυθμό 120 χαρακτήρων των 10-bits ανά δευτερόλεπτο .
Οι μονάδες UART , συχνά χρησιμοποιούνται με ένα ειδικό ολοκληρωμένο γεννήτριας ρυθμού .Αυτό παράγει ένα σήμα 16x ή 24x για όλους τους standard ρυθμούς .Το βασικό σήμα δημιουργείται από ένα κρυσταλλικό ταλαντωτή ,έτσι ώστε ο ρυθμός να είναι πολύ σταθερός και ακριβής .
Το σήμα εισόδου για την γεννήτρια ρυθμών ,τις περισσότερες φορές ,παράγεται από το σήμα χρονισμού του μικροεπεξεργαστή ,χρησιμοποιώντας ένα είδος διαιρέτη .
Μια προγραμματοζόμενη γεννήτρια ρυθμών επιτρέπει στην UART να λειτουργήσει με πολλούς ρυθμούς κάτω από τον έλεγχο του προγράμματος .
Η UART λειτουργεί ασύγχρονα με τον μικροεπεξεργαστή .Αυτό σημαίνει ότι ο χρονισμός της εκπομπής των δεδομένων δεν έχει καμία σχέση με τον χρονισμό του μικροεπεξεργαστή .Κάθε μια μονάδα ,Λειτουργεί με τον δικό της χρονισμό .Αν οι δυο μονάδες χρησιμοποιούν την ίδια μονάδα χρονισμού ,είναι για μείωση των κυκλωμάτων χρονισμού και όχι για συγχρονισμό των λειτουργιών τους .

Καμιά φορά ακούμε ,σε σχέση με την UART ,τον ορό ασύγχρονη μεταφορά δεδομένων. Αυτό έχει διαφορετική έννοια .Σημαίνει ότι ο χρόνος μεταξύ δύο λέξεων είναι άγνωστος. Για παράδειγμα ,ας εξετάσουμε δεδομένα που παράγονται ένα πληκτρολόγιο .Κάθε ασύγχρονος χαρακτήρας παράγεται με το πάτημα ενός πλήκτρου.Το διάστημα μεταξύ δύο χαρακτήρων είναι τελείως απρόβλεπτο .Εφ΄ όσον όμως αρχίσει η εκπομπή ενός χαρακτήρα ASCII ,ο ρυθμός παραμένει ο ίδιος όσο χρόνο εκπέμπεται ο χαρακτήρας .Δηλαδή ,τα δεδομένα μέσα στον χαρακτήρα είναι σύγχρονα από bit σε bit ,αλλά ασύγχρονα από λέξη σε λέξη διότι ο σχετικός χρονισμός τους είναι άγνωστος .
Η UART που διαθέτει ο AT90S8515 είναι full-Duplex με λειτουργίες ελέγχου και ανίχνευσης .Τα δεδομένα που χειρίζεται δε ,είναι των 8 ή των 9 bit .Παρακάτω φαίνεται το απλοποιημένο – σε σχέση με το data sheet του μικροελεγκτή – σχηματικό διάγραμμα του πομπού (transmitter) και του δεκτή (receiver) της UART :



UART transmiter(simplified)



UART receiver (simplified)

. UBRR Settings at Various Crystal Frequencies

Baud Rate	1 MHz	%Error	1.8432 MHz	%Error	2 MHz	%Error	2.4576 MHz	%Error
2400	UBRR= 25	0.2	UBRR= 47	0.0	UBRR= 51	0.2	UBRR= 63	0.0
4800	UBRR= 12	0.2	UBRR= 23	0.0	UBRR= 25	0.2	UBRR= 31	0.0
9600	UBRR= 6	7.5	UBRR= 11	0.0	UBRR= 12	0.2	UBRR= 15	0.0
14400	UBRR= 3	7.8	UBRR= 7	0.0	UBRR= 8	3.7	UBRR= 10	3.1
19200	UBRR= 2	7.8	UBRR= 5	0.0	UBRR= 6	7.5	UBRR= 7	0.0
28800	UBRR= 1	7.8	UBRR= 3	0.0	UBRR= 3	7.8	UBRR= 4	6.3
38400	UBRR= 1	22.9	UBRR= 2	0.0	UBRR= 2	7.8	UBRR= 3	0.0
57600	UBRR= 0	7.8	UBRR= 1	0.0	UBRR= 1	7.8	UBRR= 2	12.5
76800	UBRR= 0	22.9	UBRR= 1	33.3	UBRR= 1	22.9	UBRR= 1	0.0
115200	UBRR= 0	84.3	UBRR= 0	0.0	UBRR= 0	7.8	UBRR= 0	25.0

Baud Rate	3.2768 MHz	%Error	3.6864 MHz	%Error	4 MHz	%Error	4.608 MHz	%Error
2400	UBRR= 84	0.4	UBRR= 95	0.0	UBRR= 103	0.2	UBRR= 119	0.0
4800	UBRR= 42	0.8	UBRR= 47	0.0	UBRR= 51	0.2	UBRR= 59	0.0
9600	UBRR= 20	1.6	UBRR= 23	0.0	UBRR= 25	0.2	UBRR= 29	0.0
14400	UBRR= 13	1.6	UBRR= 15	0.0	UBRR= 16	2.1	UBRR= 19	0.0
19200	UBRR= 10	3.1	UBRR= 11	0.0	UBRR= 12	0.2	UBRR= 14	0.0
28800	UBRR= 6	1.6	UBRR= 7	0.0	UBRR= 8	3.7	UBRR= 9	0.0
38400	UBRR= 4	6.3	UBRR= 5	0.0	UBRR= 6	7.5	UBRR= 7	6.7
57600	UBRR= 3	12.5	UBRR= 3	0.0	UBRR= 3	7.8	UBRR= 4	0.0
76800	UBRR= 2	12.5	UBRR= 2	0.0	UBRR= 2	7.8	UBRR= 3	6.7
115200	UBRR= 1	12.5	UBRR= 1	0.0	UBRR= 1	7.8	UBRR= 2	20.0

Baud Rate	7.3728 MHz	%Error	8 MHz	%Error	9.216 MHz	%Error	11.059 MHz	%Error
2400	UBRR= 191	0.0	UBRR= 207	0.2	UBRR= 239	0.0	UBRR= 287	-
4800	UBRR= 95	0.0	UBRR= 103	0.2	UBRR= 119	0.0	UBRR= 143	0.0
9600	UBRR= 47	0.0	UBRR= 51	0.2	UBRR= 59	0.0	UBRR= 71	0.0
14400	UBRR= 31	0.0	UBRR= 34	0.8	UBRR= 39	0.0	UBRR= 47	0.0
19200	UBRR= 23	0.0	UBRR= 25	0.2	UBRR= 29	0.0	UBRR= 35	0.0
28800	UBRR= 15	0.0	UBRR= 16	2.1	UBRR= 19	0.0	UBRR= 23	0.0
38400	UBRR= 11	0.0	UBRR= 12	0.2	UBRR= 14	0.0	UBRR= 17	0.0
57600	UBRR= 7	0.0	UBRR= 8	3.7	UBRR= 9	0.0	UBRR= 11	0.0
76800	UBRR= 5	0.0	UBRR= 6	7.5	UBRR= 7	6.7	UBRR= 8	0.0
115200	UBRR= 3	0.0	UBRR= 3	7.8	UBRR= 4	0.0	UBRR= 5	0.0

Ο ρυθμός σηματοδότησης BAUD στον μικροελεγκτή που εξετάζουμε δίνεται από την εξίσωση:

$$BAUD = \frac{f_{CK}}{16(UBRR + 1)}$$

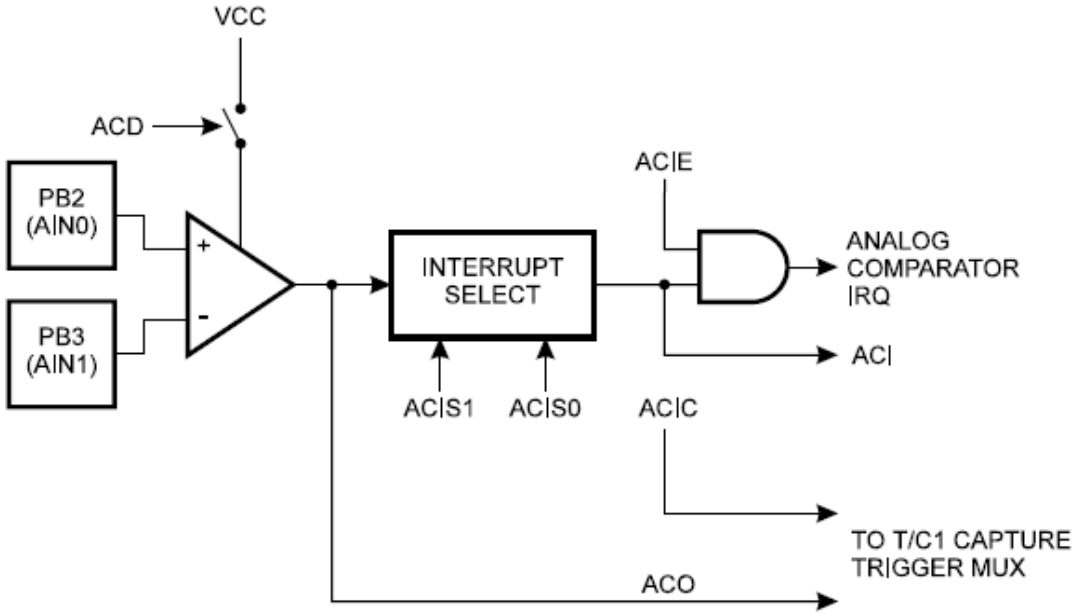
- fCK : συχνότητα του κρυστάλλου που διαθέτουμε .

- UBRR : τα περιεχόμενα του καταχωρητή UBRR(UART Baud Rate Register).

Παραπάνω βλέπουμε έναν πίνακα που δείχνει για διάφορες τιμές του UBRR και για ποικίλες συχνότητες κρυστάλλου, και τι Baud-Rate επιτυγχάνουμε.

2.15 Ο αναλογικός συγκριτής (Analog Comparator)

Ο AT90S8515 διαθέτει έναν αναλογικό συγκριτή .Ο αναλογικός αυτός συγκριτής συγκρίνει τις τιμές εισόδου στο θετικό ακροδέκτη PB2 (AIN0) και στον αρνητικό ακροδέκτη PB3 (AIN1) .Όταν η τάση στον θετικό ακροδέκτη PB2 (AIN0) είναι μεγαλύτερη από την τάση στον αρνητικό ακροδέκτη PB3 (AIN1) τότε η έξοδος του συγκριτή ,Δηλαδή το bit ACO στον καταχωρητή ASCR γίνεται 1 .Η έξοδος του συγκριτή μπορεί να ενεργοποιήσει την λειτουργία σύλληψης (input capture function) του timer/counter 1 .
Επιπροσθέτως μπορεί να ενεργοποιηθεί και ένα interrupt ,αποκλειστικό μόνο για τον συγκριτή .Ο χρήστης μπορεί να επιλέξει εάν το interrupt αυτό θα ενεργοποιηθεί στο rising ή στο falling edge της εξόδου του συγκριτή ή στην μεταβολή της κατάστασης της εξόδου (από 1 σε 0 και από 0 σε 1) .Η μεταβολή της τάσης πρέπει να είναι από 20mV και πάνω για να μπορέσει να γίνει αντιληπτή .Το block διάγραμμα του συγκριτή φαίνεται παρακάτω:



Analog Comparator Block Diagram.

2.16 Θύρες εισόδου – εξόδου (I/O Ports)

Ένας μικροελεγκτής χρησιμοποιεί τα ψηφιακά στοιχεία I/O που διαθέτει για να ανταλλάσσει ψηφιακά δεδομένα με τον εξωτερικό κόσμο .Σε σύγκριση με μια σειριακή θύρα ,η οποία μεταφέρει δεδομένα σειριακά ένα bit κάθε φορά ,τα δεδομένα σε μια παράλληλη ψηφιακή θύρα εισόδου / εξόδου ανταλλάσσονται με την μορφή bytes συνήθως .
Όλοι οι μικροελεγκτές της σειράς AVR διαθέτουν κάποια ποσότητα ακροδεκτών εισόδου / εξόδου που ξεκινούν από 3 στον μικροελεγκτή AT90S2323 και καταλήγουν σε 48 στον Mega 103 .Ο AT90S8515 διαθέτει 4 θύρες (Ports) των 8 bits έκαστη .Τα ονόματά τους είναι PortA ,PortB ,PortC και PortD .Όλοι οι ακροδέκτες στις θύρες του , σαν έξοδοι ,μπορούν να βυθίσουν (sink) ρεύμα περίπου 20mA και με αυτόν τον τρόπο μπορούν να οδηγήσουν κατευθείαν leds χωρίς την χρησιμοποίηση εξωτερικών απομονωτών όπως τρανζίστορ .Και αυτό στην περίπτωση που βρίσκονται σε κατάσταση low .

Όταν βρίσκονται σε κατάσταση high τότε το ρεύμα που άγουν (source) είναι 10mA .
Κάθε θύρα έχει τρεις διευθύνσεις στην μνήμη δεδομένων σχετικές με αυτήν .
Η μια διεύθυνση είναι για τον καταχωρητή δεδομένων PORTx ,όπου x το όνομα της θύρας .Π.χ για την θύρα A είναι PORTA .Η διεύθυνση αυτή χρησιμοποιείται για να τοποθετήσουμε δεδομένα σε εκείνα τα bit τα οποία έχουμε καθορίσει σαν έξοδο .
Η δεύτερη διεύθυνση είναι για τον καταχωρητή κατεύθυνσης δεδομένων DDRx .Η διεύθυνση αυτή χρησιμοποιείται για να καθορίσουμε ποιά pin σε μια θύρα θα χρησιμοποιηθούν σαν είσοδοι και ποια σαν έξοδοι .Γράφοντας ένα 1 σε ένα bit του συγκεκριμένου καταχωρητή DDRx το αντίστοιχο pin στον καταχωρητή PORTx γίνεται έξοδος ,ενώ εάν του γράψουμε ένα 0 γίνεται είσοδος .
Η τρίτη και τελευταία διεύθυνση έχει το όνομα PINx και η χρήση της είναι για ανάγνωση των ακροδεκτών της αντίστοιχης θύρας . Στις δύο πρώτες διευθύνσεις του καταχωρητή δεδομένων μπορούμε και να διαβάσουμε και να γράψουμε . Από την τρίτη διεύθυνση μπορούμε μονό να διαβάσουμε δεδομένα (read – only address) .
Επιπροσθέτως πρέπει να πούμε ότι στους ακροδέκτες μιας θύρας που χρησιμοποιούνται ως είσοδοι , υπάρχει η επιλογή να ενεργοποιηθούν εσωτερικές αντιστάσεις πρόσδεσης δηλ . Pull-up με τιμές μεταξύ 35 ΚΩ και 120 ΚΩ .Στην περίπτωση που δεν ενεργοποιήσουμε τις αντιστάσεις πρόσδεσης τότε οι ακροδέκτες εισόδου βρίσκονται στον <<αέρα >> ,σε κατάσταση Tri-state (Hi-Z) ή όπως αλλιώς το λεμέ σε κατάσταση υψηλής εμπέδησης . Τότε οι ακροδέκτες και τελικώς και το ολοκληρωμένο γίνονται ευαίσθητα σε εξωτερικά παράσιτα γεγονός που μπορεί να επηρεάσει την λειτουργία του ρολογιού και ίσως την γενική λειτουργία του προγράμματος .
Τέλος να αναφέρουμε όπως φαίνεται και από το Pin-out του AT90S8515 ότι ορισμένοι από τους ακροδέκτες των θυρών έχουν και άλλες λειτουργίες , οι οποίες ρυθμίζονται κατά επιλογή , όπως είσοδοι μετρητών , είσοδοι για εξωτερικά interrupt, SPI προγραμματισμό , διευθυνσιοδότηση εξωτερικής μνήμης κ.ο.κ. Το σχηματικό διάγραμμα κάθε θύρας φαίνεται στο data sheet του μικροελεγκτή .

2.17 Λειτουργίες χαμηλής κατανάλωσης

Όταν το σύστημα βρίσκεται σε κάποια από τις δυο λειτουργίες χαμηλής κατανάλωσης (sleep modes) και εμφανιστεί ένα interrupt , <<ξυπνάει>> ο μικροελεγκτής , εκτελεί την ρουτίνα του interrupt και συνεχίζει να εκτελεί το πρόγραμμα ,από την εντολή που βρίσκεται μετά από την εντολή εκείνη που ενεργοποίησε την λειτουργία sleep mode .Τα περιεχόμενα των καταχωρητών της μνήμης SRAM και της μνήμης I/O παραμένουν αμετάβλητα . Εάν παρουσιαστεί ένα reset κατά την λειτουργία sleep mode , ο μικροελεγκτής ενεργοποιείται και εκτελεί το πρόγραμμα αρχίζοντας από τον τομέα του reset (reset vector) .
Οι λειτουργίες χαμηλής κατανάλωσης είναι δύο :

- **Idle mode :** Όταν ενεργοποιηθεί η λειτουργία αυτή η CPU του μικροελεγκτή σταματάει να λειτουργεί , όμως οι χρονιστές / μετρητές ,ο χρονιστής επιτήρησης (watchdog timer) και το σύστημα των interrupt εξακολουθούν να λειτουργούν .Αυτό επιτρέπει την επανενεργοποίηση του μικροελεγκτή , την επαναφορά του δηλαδή από την λειτουργία idle mode , με την εμφάνιση ενός interrupt οφειλόμενου σε εσωτερικό ή και εξωτερικό αίτιο , καθώς και από ένα reset οφειλόμενο στον watchdog timer .
- **Power down mode :** Σε αυτήν εδώ την περίπτωση , όταν δηλαδή ενεργοποιηθεί η συγκεκριμένη λειτουργία ,σταματάει να χρονίζεται ο μικροελεγκτής από το εξωτερικό ρολόι . Εδώ είναι τρείς οι περιπτώσεις για επαναφορά του χρονισμού και

συνεπώς της επαναλειτουργίας του μικροελεγκτή. Η μία είναι να έχουμε ενεργοποιήσει τον watchdog timer .

Όταν η περίοδος time - out ,που είδαμε στο ειδικά για αυτόν κεφάλαιο , συμπληρωθεί τότε γίνεται reset στο σύστημα και εκείνο αρχίζει να λειτουργεί πλέον κανονικά . Στην περίπτωση όμως που δεν έχει ενεργοποιηθεί ο συγκεκριμένος χρονιστής τότε το σύστημα μπορεί να ξαναλειτουργήσει μόνο εάν γίνει ένα εξωτερικό reset στον μικροελεγκτή ή εμφανιστεί ένα εξωτερικό interrupt της μορφής level triggered δηλ .να δημιουργείται με μεταβολή επιπέδου τάσης .

Τα ρεύματα που καταναλώνονται στην διάρκεια λειτουργίας του μικροελεγκτή , κάτω υπό τις προαναφερθείσες λειτουργίες χαμηλής κατανάλωσης φαίνονται στον πίνακα που δείχνει τα D.C χαρακτηριστικά του μικροελεγκτή .

2.18 Bits κλειδώματος (lock bits)

Ο AT90S8515 διαθέτει δύο bit κλειδώματος , τα LB1 και LB2 τα οποία μπορούν να προγραμματιστούν ή να μείνουν απρογραμμάτιστα .Εάν προγραμματιστούν τότε , εάν το LB1 είναι 1 και το LB2 είναι 1 τότε δεν είναι ενεργοποιημένη κάποια λειτουργία κλειδώματος . Εάν το LB1 είναι 0 και το LB2 είναι 1 τότε κλειδώνει ο προγραμματισμός της μνήμης προγράμματος Flash δηλ . περαιτέρω προγραμματισμός δεν είναι εφικτός . Εάν το LB1 και το LB2 είναι 0 τότε και πάλι κλειδώνει ο προγραμματισμός αλλά αυτή την φορά δεν είναι εφικτή και η επαλήθευση .Ας σημειώσουμε ότι τα lock bits μπορούν μονό να σβηστούν από μια chip erase λειτουργία .

2.19 Bits ασφαλείας

Ο AT90S8515 έχει δύο bits ασφαλείας : το SPIEN και το FSTRT .Όταν έχει προγραμματιστεί (‘0’) (δηλ .γίνει 0) το SPIEN τότε ενεργοποιείται η άδεια για πραγματοποίηση σειριακού προγραμματισμού .Το bit αυτό είναι εξαρχής προγραμματισμένο (‘0’) .
Όταν έχει προγραμματιστεί (‘0’) το bit FSTRT , όπως είδαμε και στην παράγραφο του reset , γίνεται πιο σύντομος ο χρόνος tout . Το bit αυτό είναι εξαρχής απρογραμμάτιστο (‘1’) .Πρόσβαση σε αυτά τα bits δεν είναι δυνατή στον σειριακό προγραμματισμό και δεν επηρεάζονται από μια chip erase λειτουργία .

2.20 Τα bits υπογραφής (Signature bits)

Όλοι οι μικροελεγκτές τις Atmel έχουν μια <<υπογραφή>> με την μορφή κωδικού αποτελούμενου από τρία bytes , ο οποίος πιστοποιεί τον μικροελεγκτή. Αυτός ο κωδικός μπορεί να διαβαστεί και στην σειριακή και στη παράλληλη λειτουργία .Τα τρία αυτά bytes βρίσκονται σε ξεχωριστή διεύθυνση και για τον AT90S8515 είναι :

1. $000 : $1E (δείχνει ότι κατασκευαστής είναι η Atmel) .
2. $001 : $93 (δείχνει ότι η μνήμη είναι 8K)
3. $002 : $01 (δείχνει ότι ο μικροελεγκτής είναι ο AT90S8515 εάν το προηγούμενο είναι $93) .

2.21 Λειτουργία διαγραφής (Chip erase function)

Η λειτουργία διαγραφής θα σβήσει την μνήμη flash και την EEPROM καθώς και τα lock bits , τα οποία μάλιστα δεν αρχικοποιούνται (reset) μέχρι να σβήσει τελείως η μνήμη προγράμματος . Τα fuse bits δεν αλλάζουν . Πρίν προγραμματιστεί ο μικροελεγκτής πρέπει να προηγηθεί μια διαγραφή .

2.22 DC χαρακτηριστικά του AT90S8515

Τα DC χαρακτηριστικά του συγκεκριμένου μικροελεγκτή φαίνονται συγκεντρωτικά στον πίνακα που ακολουθεί :

DC Characteristics

T_A = -40°C to 85°C, V_{CC} = 2.7V to 6.0V (unless otherwise noted)

Symbol	Parameter	Condition	Min	Typ	Max	Units
V_{IL}	Input Low Voltage	(Except XTAL1)	-0.5		0.3 V_{CC}(1)	V
V_{IL1}	Input Low Voltage	(XTAL1)	-0.5		0.2 V_{CC}(1)	V
V_{IH}	Input High Voltage	(Except XTAL1, $\overline{RESET}$)	0.6 V_{CC}(2)		V_{CC} + 0.5	V
V_{IH1}	Input High Voltage	(XTAL1)	0.8 V_{CC}(2)		V_{CC} + 0.5	V
V_{IH2}	Input High Voltage	($\overline{RESET}$)	0.9 V_{CC}(2)		V_{CC} + 0.5	V
V_{OL}	Output Low Voltage(3) (Ports A, B, C, D)	I_{OL} = 20 mA, V_{CC} = 5V I_{OL} = 10 mA, V_{CC} = 3V			0.6 0.5	V V
V_{OH}	Output High Voltage(4) (Ports A, B, C, D)	I_{OH} = -3 mA, V_{CC} = 5V I_{OH} = -1.5 mA, V_{CC} = 3V	4.2 2.3			V V
I_{IL}	Input Leakage Current I/O Pin	V_{CC} = 6V, pin low (absolute value)			8.0	µA
I_{IH}	Input Leakage Current I/O Pin	V_{CC} = 6V, pin high (absolute value)			980.0	nA
RRST	Reset Pull-up Resistor		100.0		500.0	kΩ
$R_{I/O}$	I/O Pin Pull-up Resistor		35.0		120.0	kΩ
I_{CC}	Power Supply Current	Active Mode, V_{CC} = 3V, 4 MHz			3.0	mA
		Idle Mode V_{CC} = 3V, 4 MHz			1.2	mA
	Power-down mode(5)	WDT enabled, V_{CC} = 3V		9.0	15.0	µA
		WDT disabled, V_{CC} = 3V		<1.0	2.0	µA
V_{ACIO}	Analog Comparator Input Offset Voltage	V_{CC} = 5V V_{in} = V_{CC}/2			40.0	mV
I_{ACLK}	Analog Comparator Input Leakage Current	V_{CC} = 5V V_{in} = V_{CC}/2	-50.0		50.0	nA
t_{ACPD}	Analog Comparator Propagation Delay	V_{CC} = 2.7V V_{CC} = 4.0V		750.0 500.0		ns

3 Η γλώσσα προγραμματισμού C

3.1 Ιστορική αναδρομή

Μία ακόμη γλώσσα που γνώρισε μεγάλη διάδοση είναι η γλώσσα 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 εξελίχτηκε 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 πολύ μικρό κομμάτι του που αναγκαστικά θα γραφόταν σε γλώσσα assembly (τη γλώσσα που βρίσκεται στο χαμηλότερο σχεδόν επίπεδο, δηλ. 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 μία νέα μηχανή , αρκούσε να γράψει ένα μικρό κομμάτι του λειτουργικού συστήματος. Για το υπόλοιπο αρκούσε να έχει ένα C compiler για τη συγκεκριμένη μηχανή. (Είναι σημαντικό να ξεχωρίσουμε εδώ ότι ένας compiler για μία γλώσσα που τρέχει σε μια 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 οποίο είχε επινοηθεί. Ήταν μία 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 εμφάνισή της η C++ (αργότερα, όταν θα έχετε αποκτήσει λίγες γνώσεις για την γλώσσα C θα δείτε γιατί αυτή η περίεργη ονομασία) που είναι μια επέκταση της C προς την κατεύθυνση του object-oriented προγραμματισμού. Το μερίδιο της C++ στους προγραμματιστές αυξάνει διαρκώς, μιάς και προσφέρεται ιδιαίτερα για δουλειές όπως την κατασκευή φιλικών user interfaces (δηλ. 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 διαχείριση πολλών διαφορετικών τύπων δεδομένων πάνω στα οποία θέλουμε κάπως να κάνουμε ίδιες εργασίες.

3.2 Πλεονεκτήματα των γλωσσών υψηλού επιπέδου

Στα πλεονεκτήματα των γλωσσών προγραμματισμού υψηλού επιπέδου σε σχέση με τις συμβολικές μπορούν να αναφερθούν:
Ο φυσικότερος και πιο «ανθρώπινος» τρόπος έκφρασης των προβλημάτων. Τα προγράμματα σε γλώσσα υψηλού επιπέδου είναι πιο κοντά στα προβλήματα που επιλύουν. Η ανεξαρτησία από τον τύπο του υπολογιστή. Προγράμματα σε μία γλώσσα υψηλού επιπέδου μπορούν να εκτελεστούν σε οποιονδήποτε υπολογιστή με ελάχιστες ή καθόλου μετατροπές. Η δυνατότητα της μεταφερσιμότητας των προγραμμάτων είναι σημαντικό προσόν. Η ευκολία της εκμάθησης και εκπαίδευσης ως απόρροια των προηγουμένων. Η διόρθωση λαθών και η συντήρηση προγραμμάτων σε γλώσσα υψηλού επιπέδου είναι πολύ ευκολότερο έργο. Συνολικά οι γλώσσες υψηλού επιπέδου ελάττωσαν σημαντικά το χρόνο και το κόστος παραγωγής νέων προγραμμάτων, αφού λιγότεροι προγραμματιστές μπορούν σε μικρότερο χρόνο να αναπτύξουν προγράμματα που χρησιμοποιούνται σε περισσότερους υπολογιστές.
Τα τελευταία χρόνια χρησιμοποιείται ιδιαίτερα, ειδικά για προγραμματισμό στο Διαδίκτυο (Internet), η JAVA. Η JAVA είναι μια αντικειμενοστραφής γλώσσα που αναπτύχθηκε από την εταιρία SUN με σκοπό την ανάπτυξη εφαρμογών που θα εκτελούνται σε κατανεμημένα περιβάλλοντα.

3.3 Τι είναι ένα πρόγραμμα σε γλώσσα C

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 ποια συνάρτηση αναφερόμαστε όταν χρησιμοποιούμε κάποιο όνομα, ενώ μία από τις συναρτήσεις θα πρέπει να έχει το όνομα *main* και αντιπροσωπεύει το σημείο από το οποίο ξεκινάει και τελειώνει η εκτέλεση του προγράμματός 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.

3.4 Συγγραφή ενός προγράμματος C

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

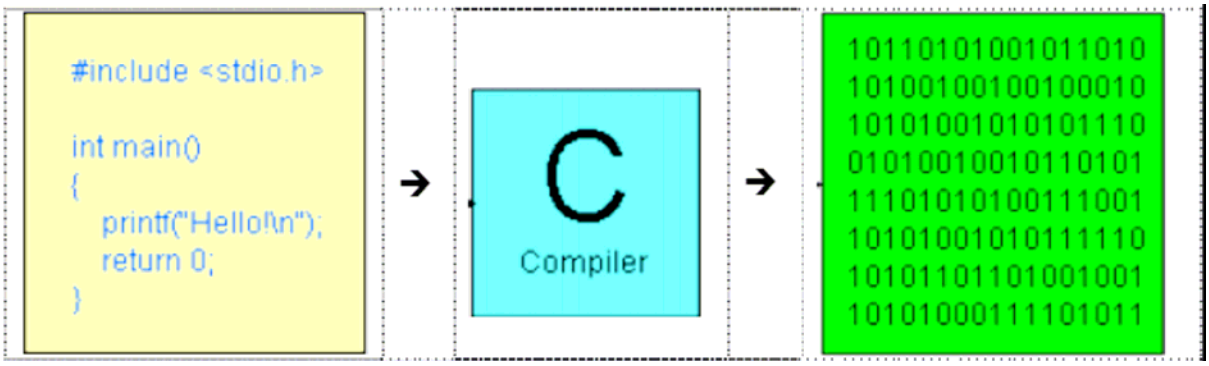
2) ο μεταγλωττιστής της γλώσσας C που θα αναλάβει να μεταφράσει το πρόγραμμά μας σε κώδικα μηχανής. Ο μεταγλωττιστής της C αποτελείται από τον C preprocessor και τον C-compiler. Στα επόμενα με τον όρο "compiler" θα αναφερόμαστε και στα δύο αυτά τμήματα, εκτός αν καθορίζεται διαφορετικά. Ο C-preprocessor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

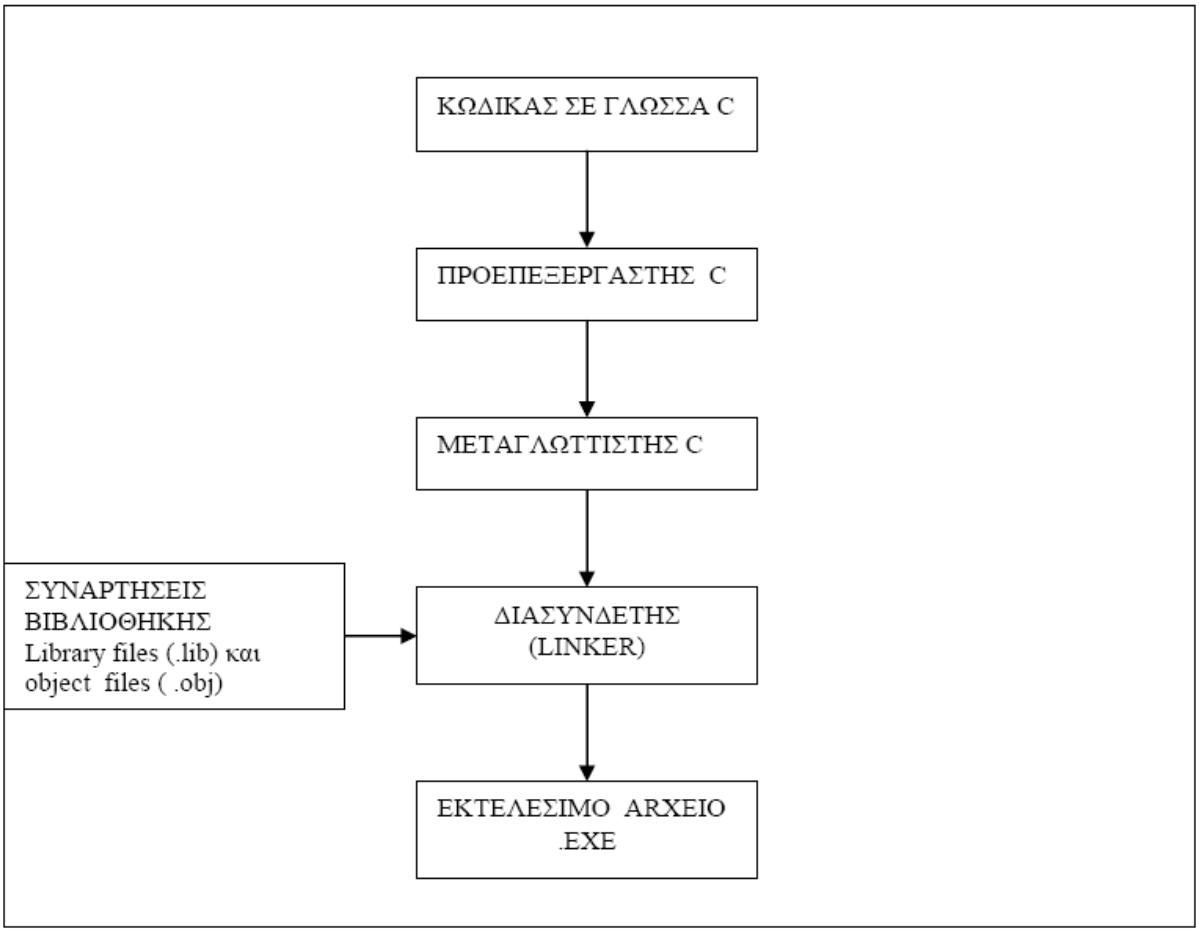
κώδικα C, με τρόπο που θα αναλυθεί σε επόμενο κεφάλαιο, και παράγει ενδιάμεσα αρχεία, από τα οποία ο C-compiler παράγει αρχεία με κώδικα μηχανής (.obj) αρχεία

3) ο linker, που είναι κομμάτ ι 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:





3.5 Ένα απλό παράδειγμα σε γλώσσα C

```
# include <stdio.h>
main()
{
int epiloge;
do
{
printf ("1:esagogi vathmologion spoudaston \n");
printf("2:ektiposi vathmologias spoudasti \n");
printf("3:ektiposi vathmologias olon ton spoudaston \n");
printf("4:diorthosi vathmologias spoudasti \n");
printf("opoiosdipote allos akereos gia exodo \n");
printf("doste tin epiloge sas \n");

scanf("%d" , &epiloge);
}
while((epiloge>=1)&&(epiloge<=4));
}
```

```
C:\Documents and Settings\aristos\Τα έγγραφά μου\παράδειγμα.exe
1:esagogi vathmologion spoudaston
2:ektiposi vathmologias spoudasti
3:ektiposi vathmologias olon ton spoudaston
4:diorthosi vathmologias spoudasti
opoiosdipote allos akereos gia exodo
doste tin epiloge sas
```

Το παραπάνω πρόγραμμα τυπώνει τα σχόλια που βρίσκονται μέσα στις παρενθέσεις της συνάρτησης printf, τοποθετώντας το νούμερο 10 στη θέση της μεταβλητής %d. Το πρόγραμμα της C ξεκινά με την κλήση της κύριας συνάρτησης με όνομα main. Οι αγκύλες

{και} δηλώνουν την αρχή και το τέλος του προγράμματος, αλλά και την αρχή και το τέλος μίας ενότητας του προγράμματος. Το ζευγάρι των παρενθέσεων που ακολουθεί τη λέξη-κλειδί main, υποδεικνύει στο μεταφραστή της γλώσσας προγραμματισμού C ότι πρόκειται για μία συνάρτηση. Η πρώτη εντολή (#include <stdio.h>) καλείται οδηγία προεπεξεργαστή και πληροφορεί τον compiler (μεταφραστή) της C ότι θα γίνει χρήση της βιβλιοθήκης stdio.h που πρόκειται για προκαθορισμένη standard βιβλιοθήκη και κατά βάση αφορά τις κύριες εντολές εισόδου / εξόδου. 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 . Η εντολή (που είναι συνάρτηση) printf καλείται από την κύρια συνάρτηση main( ), για να τυπώσει τη σειρά των χαρακτήρων που βρίσκονται μέσα στα εισαγωγικά). Παρατηρήστε το χαρακτήρα\n στο τέλος της εντολής printf. χρησιμοποιούνται για να αλλάξουμε γραμμή στη μονάδα εξόδου που δουλεύουμε (π.χ. στην οθόνη μας). Υπάρχουν και άλλοι χαρακτήρες ελέγχου του τρόπου
εμφάνισης του μηνύματος μας μέσω της συνάρτησης printf. Ο χαρακτήρας; (ελληνικό ερωτηματικό) δηλώνει ότι τελείωσε η περιγραφή της Εντολής και χρησιμοποιείται για αυτόν τον σκοπό. Δηλαδή κάθε εντολή της C τελειώνει με ένα ;.Τα σχόλια περικλείονται μεταξύ των χαρακτήρων/* (αρχή του σχολίου) και */ (τέλος σχολίου), μπορούμε να τα βάλουμε σε οποιοδήποτε σημείο του προγράμματος, αγνοούνται από το μεταφραστή και είναι δυνατόν να συνεχίζονται σε άλλη γραμμή.

3.6 Δεδομένα, σταθερές και μεταβλητές

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.

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 ";")

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.

3.7 Λέξεις κλειδιά που δεν χρησιμοποιούνται για ονόματα μεταβλητών στη γλώσσα C.

Η λέξεις κλειδιά αποτελούν το λεξιλόγιο της C. Γι' αυτό το λόγο δεν μπορούν να χρησιμοποιούνται για ονόματα μεταβλητών. Τέτοιες λέξεις είναι η if , int , printf.

Στον παρακάτω πίνακα δίνονται οι λέξεις κλειδιά.

Auto	Break	Case	Char	Const	Continue
Default	Do	Double	Else	Enum	Extern
Float	For	Goto	If	Int	Long
Register	Retun	Short	Signed	Sizeof	Static
Struct	Switch	Typedf	Union	Usigned	Void
Volalite	While				

3.8 Βασικοί τύποι δεδομένων

Η C έχει πέντε βασικούς τύπους δεδομένων:
char (character),

int (integer),

float (floating point),

double (double floating point),

void (no value).
Όλοι οι άλλοι τύποι της C βασίζονται σ' αυτούς. Όλοι οι βασικοί τύποι εκτός από τον τύπο void μπορεί ν' 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.Στο πίνακα δίνουμε τους τύπους δεδομένων όπως ορίζονται από το πρότυπο ANSI.

Τυπος	Μέγεθος σε bits	Διάστημα τιμων
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

3.9 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	= += -= *= /=

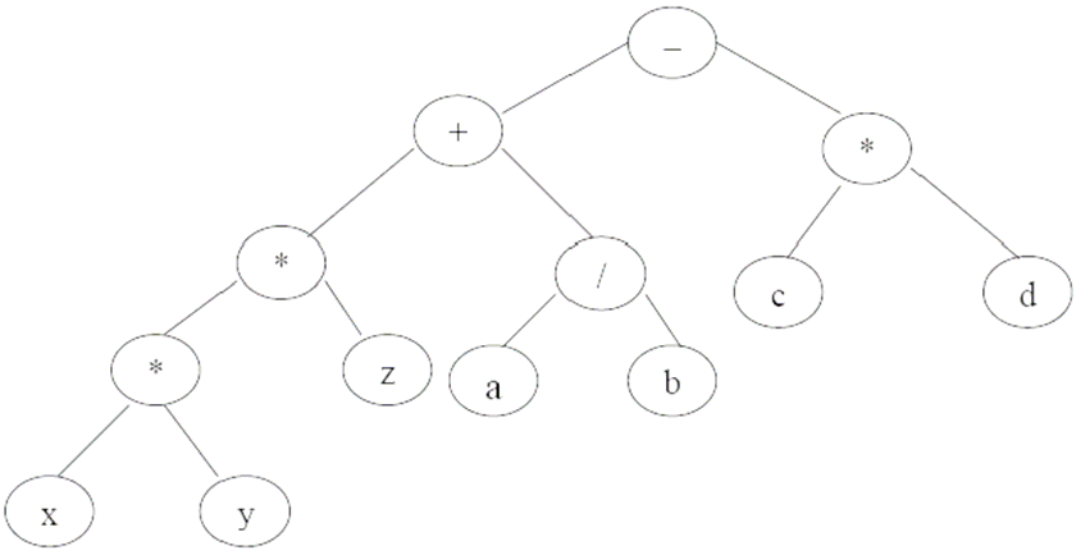
3.10 Αριθμητικοί τελεστές

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.

3.11 Οι τελεστές συσχετισμού

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)

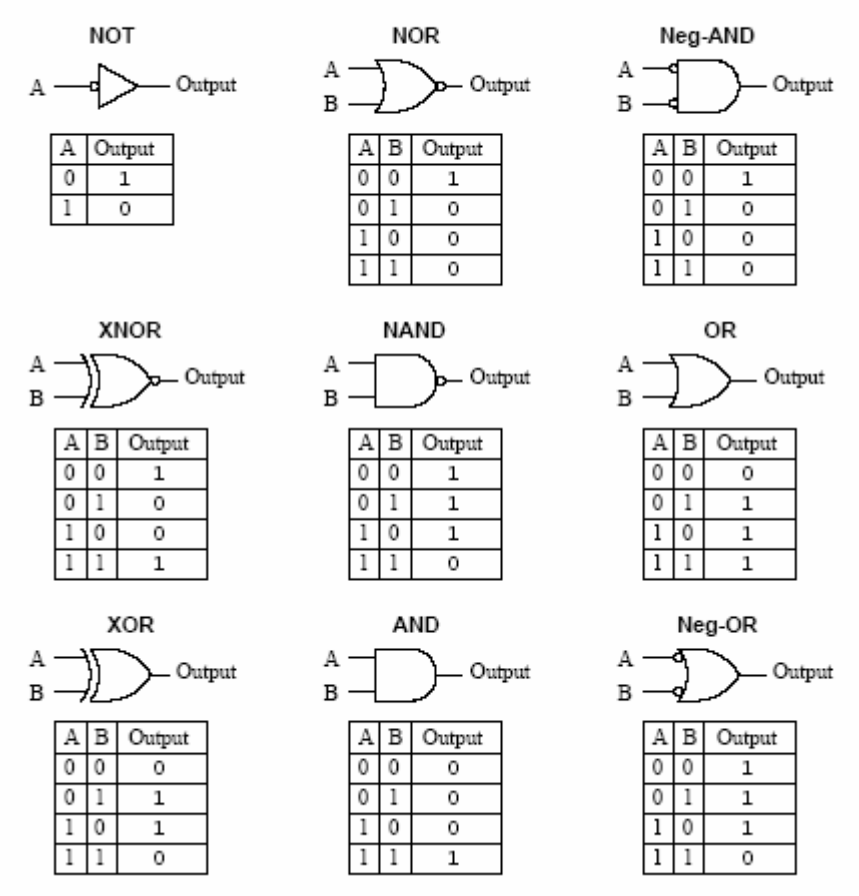
3.12 Λογικοί τελεστές

&&	Λογικό ‘και’ (AND) μας επιστρέφει 1 αν και δύο μεταβλητές είναι λογικό μηδέν , Διαφορετικά μηδέν
II	Λογικό διαζευκτικό ‘ή’ (είτε -OR)
I	Λογική άρνηση (NOT)

3.13 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

Πύλες και πίνακες αληθείας τους



A	Output
0	1
1	0

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	0

A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

A	B	Output
0	0	0
0	1	1
1	0	1
1	1	0

A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

3.14 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; }

3.15 Εντολή if-else

Οι εντολές if και if-else υπάρχουν σχεδόν σε όλες τις γλώσσες προγραμματισμού 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.

3.16 Εντολή switch

Η γενικευμένη μορφή 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) { case1: 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).

3.17 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.

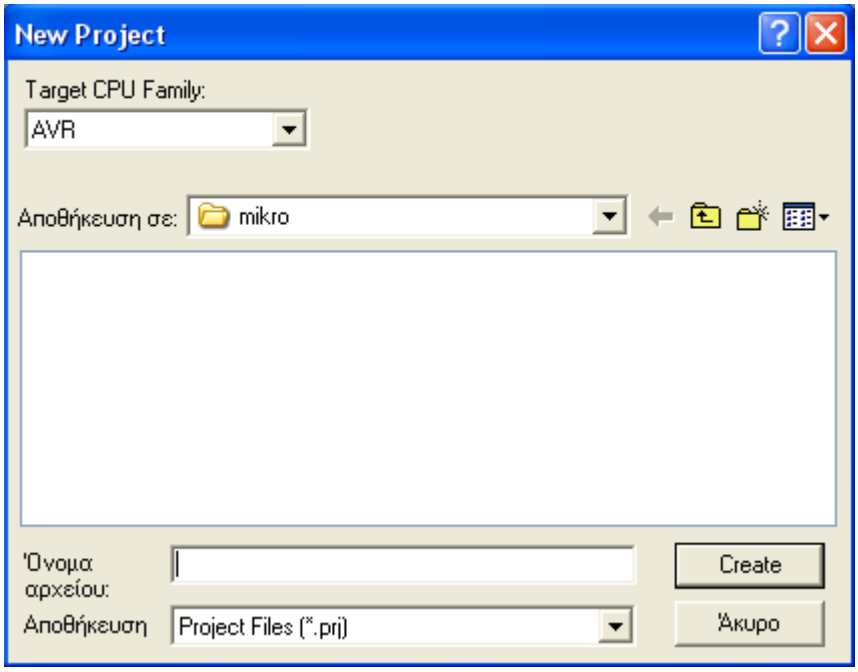
```
while(...)
{
. . .
break;
. . .
}

. . . /* Η εκτέλεση του προγράμματος συνεχίζεται μετά το break */
```

4 Compiler IAR Embedded Workbench 3.2

Σε αυτό το κεφαλαίο θα δούμε πώς γίνονται οι απαραίτητες ρυθμίσεις για τη σωστή χρήση του Compiler. Ο μεταγλωττιστής ,(Compiler), είναι ένα λογισμικό που έχει σκοπό να μεταγλωττίσει μια γλώσσα υψηλού επιπέδου όπως την C++ και να την μετατρέψει σε ένα εκτελέσιμο αρχείο, όπου μέσω του προγράμματος ponyprog ή του AVR ISP, να προγραμματίσουμε το μικροελεγκτή μας.

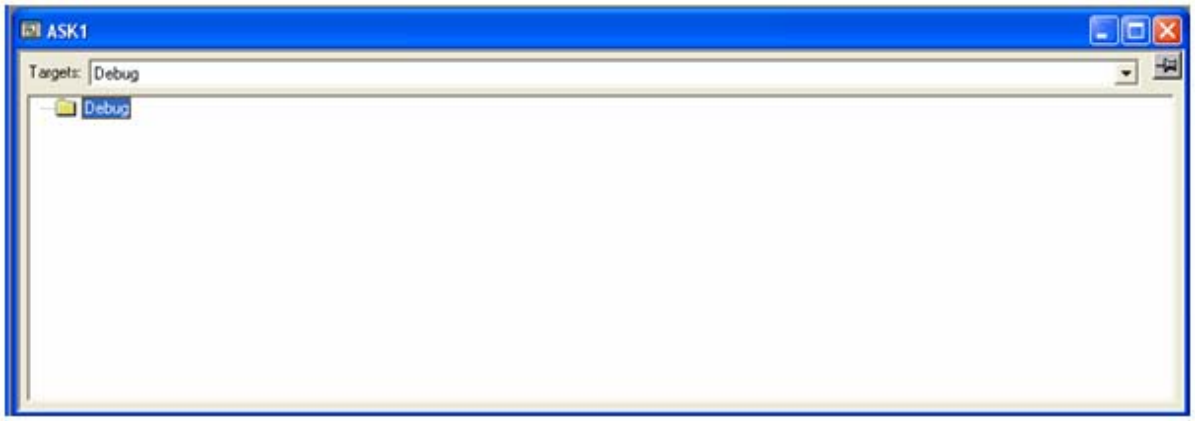
4.1 Δημιουργώντας ένα project

Ανοίγουμε το πρόγραμμα IAR Embedded Workbench και θέλουμε να γράψουμε τον κώδικά μας σε γλώσσα C++ και να προγραμματίσουμε ένα μικροελεγκτή της σειράς AVR της Atmel. Στις εργαστηριακές ασκήσεις θα προγραμματίσουμε το μικροελεγκτή AT90S8515. Ετσι πηγαίνουμε στο file→new→project. Τότε θα εμφανιστεί το παρακάτω παράθυρο:



Εδώ ονομάζουμε το project που έχουμε ανοίξει ( ) ,με το δικό μας όνομα ASK1a και επιλέγουμε στο save in: MICRO.
Το project είναι ένα αρχείο μέσα στο οποίο τοποθετούμε το αρχείο που περιλαμβάνει τον κώδικα σε C , του προγράμματος που θα δημιουργήσουμε .
Επίσης μπορούμε να τοποθετήσουμε και περισσότερα του ενός αρχεία με κώδικες που συνεργάζονται μεταξύ τους για να δημιουργήσουν ένα κοινό εκτελέσιμο αρχείο με το οποίο γίνεται ο προγραμματισμός του μικροελεγκτή. Για αυτόν ακριβώς τον λόγο το εκτελέσιμο αυτό αρχείο έχει το όνομα του project που έχουμε δημιουργήσει.
Όταν στο τέλος πατήσουμε create θα δημιουργηθεί ένα αρχείο τύπου project (.prj) στον φάκελο που εμείς έχουμε επιλέξει (συγκεκριμένα εδώ είναι ο φάκελος με το όνομα της ομάδας μας C : \Micro\Hlectro1).

Μόλις ενεργοποιήσουμε το πλήκτρο create τότε εμφανίζεται το παρακάτω παράθυρο:

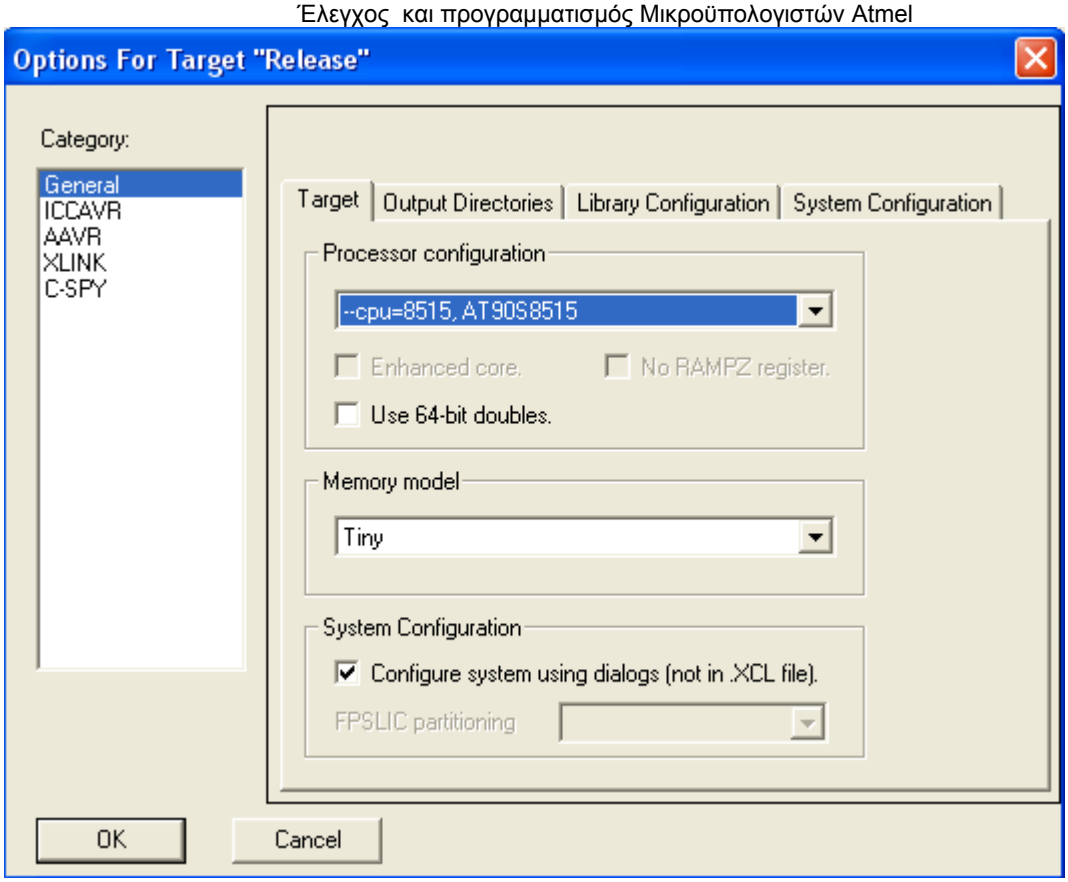


Εδώ μπορούμε να ακολουθήσουμε δυο κατευθύνσεις (Targets) .
Η πρώτη είναι να επιλέξουμε Debug .Με την επιλογή αυτή δημιουργείται στο τέλος της διαδικασίας , όταν δηλαδή έχει γίνει και η διασύνδεση (link) , ένα αρχείο το οποίο ανοίγει από κάποιο προσομοιωτή (simulator) , μέσα από τον οποίο μπορούμε να δούμε πως τρέχει το πρόγραμμα μέσα στον μικροελεγκτή ,
Εάν επιλεχθεί η δεύτερη κατεύθυνση Release τότε δημιουργείται ένα εκτελέσιμο αρχείο τύπου hex(intel standard) με το οποίο προγραμματίζουμε τον μικροελεγκτή ,εμείς Εδώ θα επιλέξουμε το **Release .**
αρχείο το οποίο ανοίγει από κάποιο προσομοιωτή (simulator) , μέσα από τον οποίο μπορούμε να δούμε πως τρέχει το πρόγραμμα μέσα στον μικροελεγκτή ,

Εάν επιλεχθεί η δεύτερη κατεύθυνση Release τότε δημιουργείται ένα εκτελέσιμο αρχείο τύπου hex(intel standard) με το οποίο προγραμματίζουμε τον μικροελεγκτή ,Εμείς Εδώ θα επιλέξουμε το **Release .**

4.2 Ρυθμίσεις

Πριν μεταγλωττίσουμε και διασυνδέσουμε κάποιο κώδικα ,οι επιλογές του compiler και του linker πρέπει να ρυθμιστούν σωστά .
Στην πάνω γραμμή των εργαλείων επιλέγουμε από το **Project to options.**
Τότε θα εμφανιστεί το παρακάτω παράθυρο:

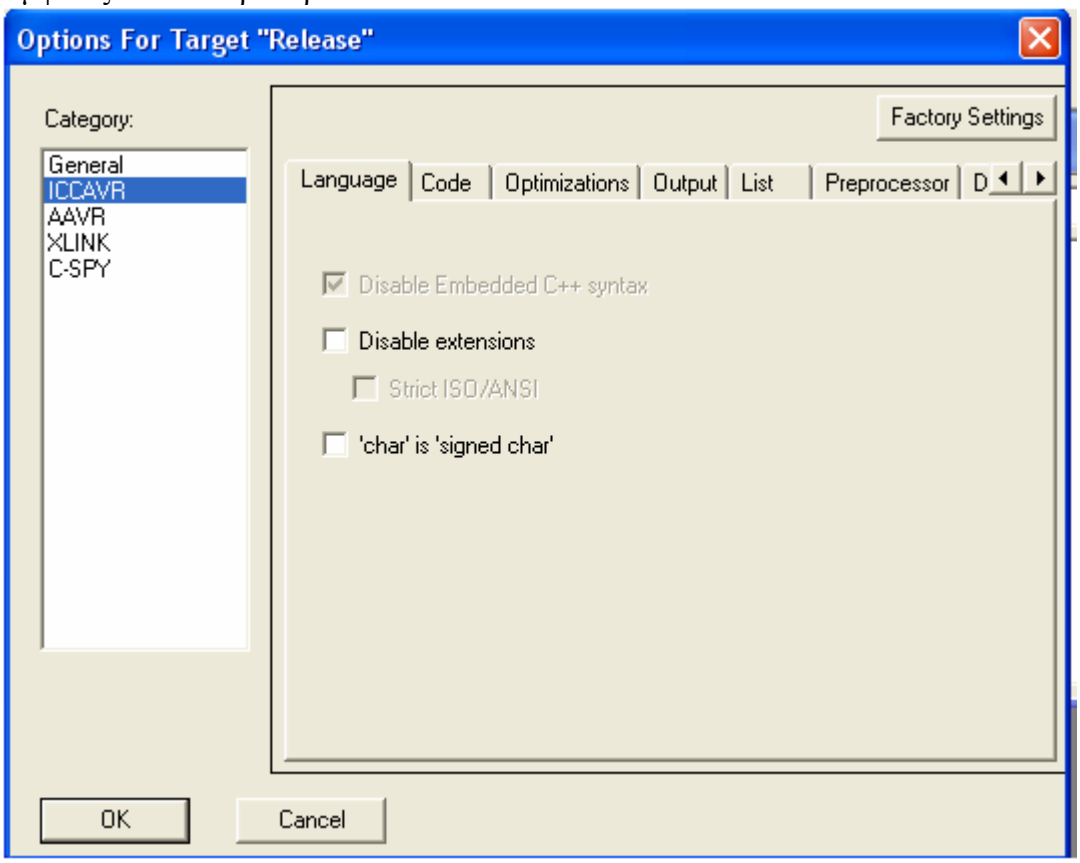


Γενικές ρυθμίσεις (GENERAL)

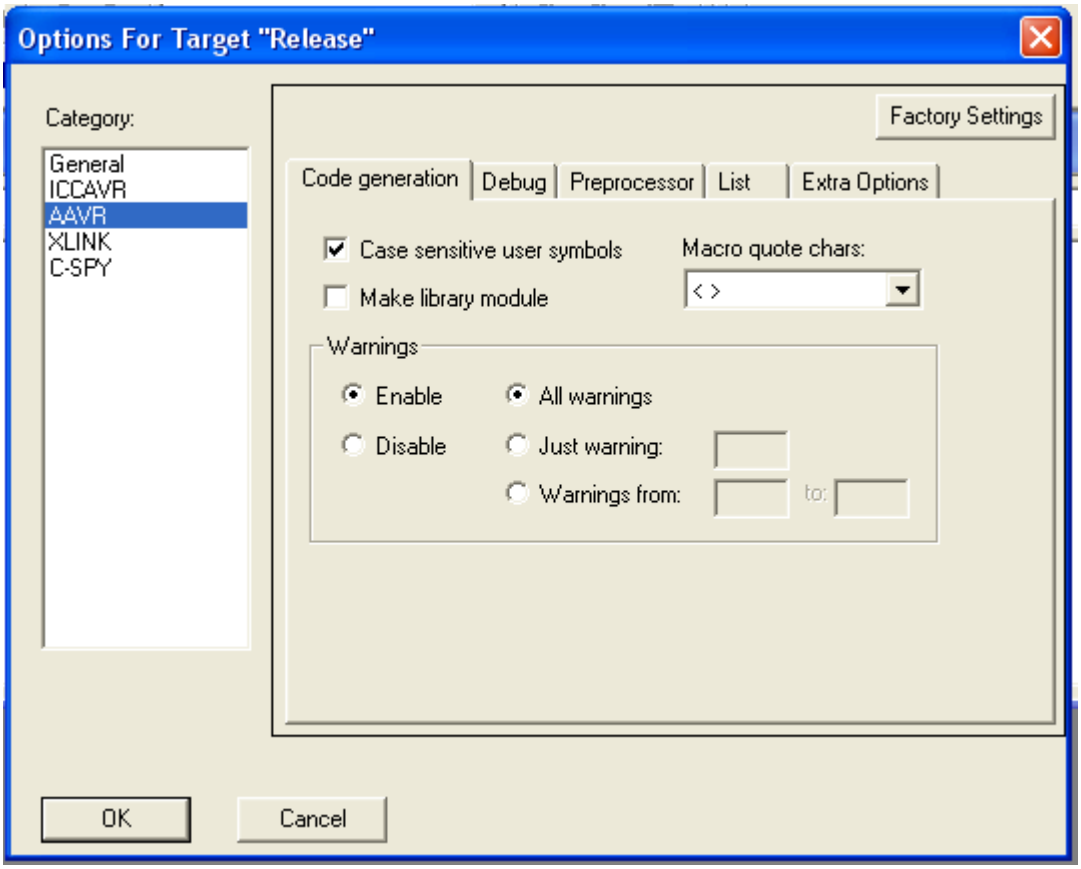
Από την επιλογή processor configuration , γίνεται η επιλογή του μικροελεγκτή .Από το Memory moodel η επιλογή **TINY** χρησιμοποιεί Data pointer 1-byte ενώ η επιλογή **Small** χρησιμοποιεί 2-byte Data pointer.
Η επιλογή για τον 8515 είναι η επιλογή TINY.

Ρυθμίσεις για τον compiler(ICCAVR)

Εμφανίζεται το παράθυρο:



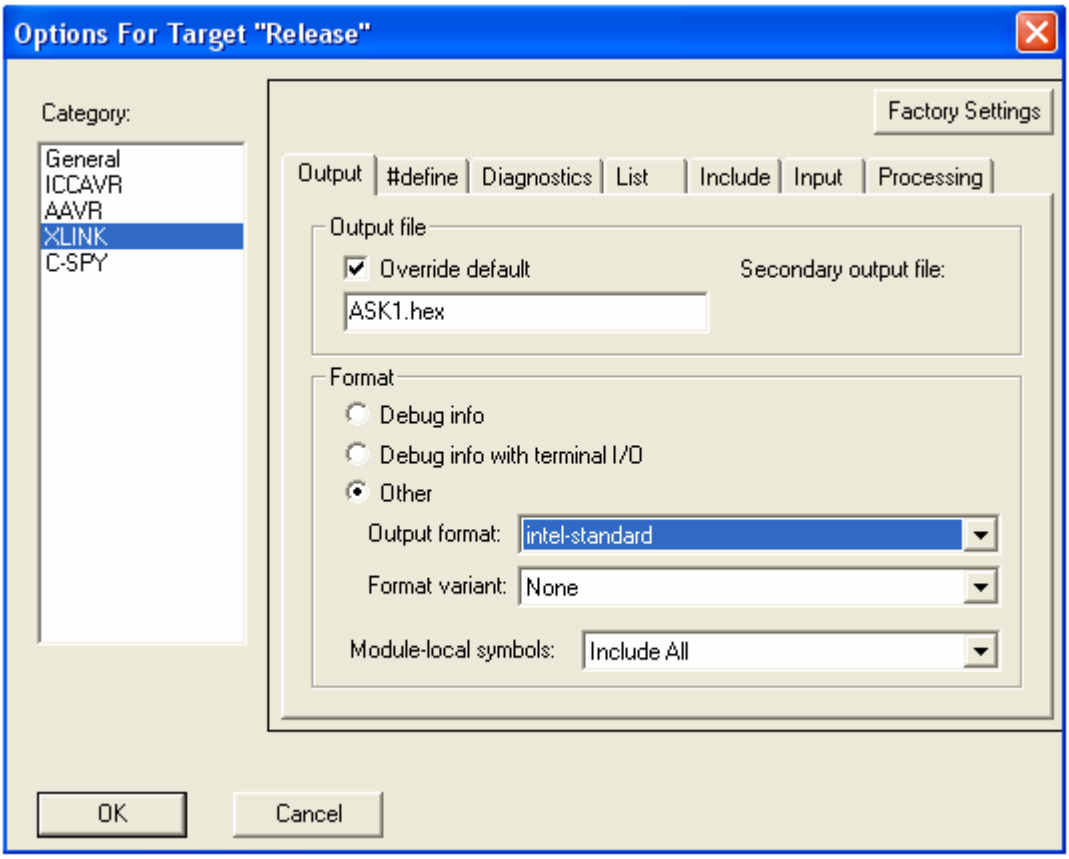
Ρυθμίσεις Assembler(AAVR) (συνήθως είναι έτοιμες)



Τις ρυθμίσεις εδώ τις αφήνουμε όπως έχουν.

Ρυθμίσεις του διασυνδετή (XLINK)

Επιλέγουμε XLINK και εμφανίζεται το παρακάτω παράθυρο :



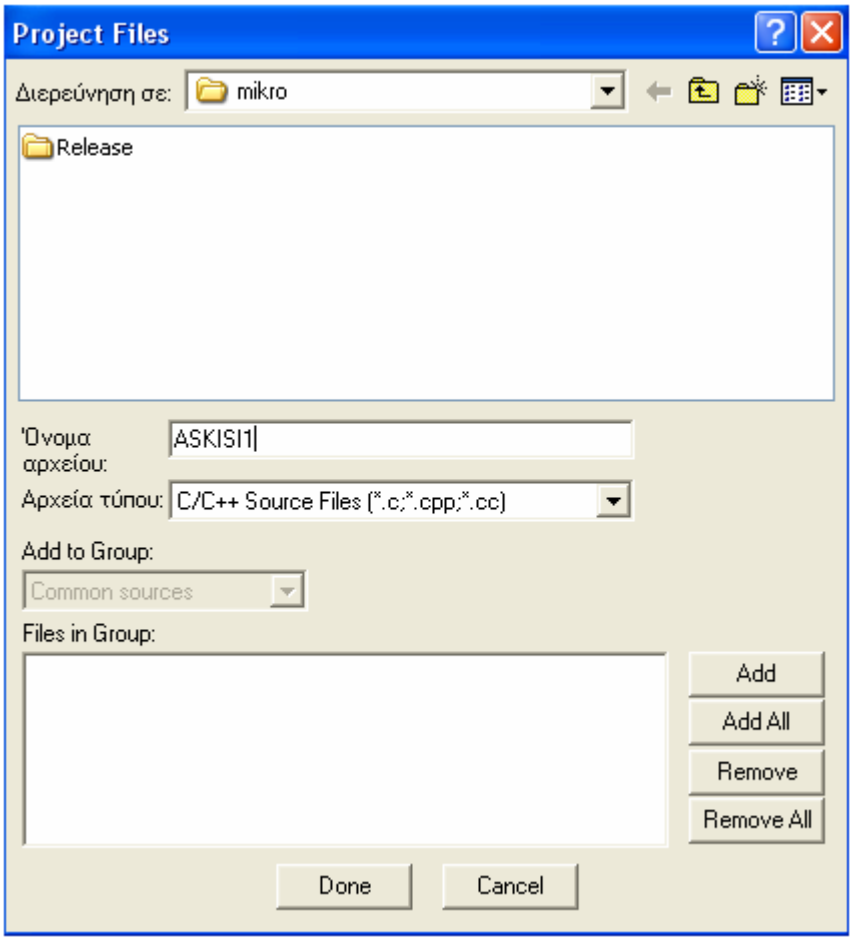
Εδώ ενεργοποιούμε την επιλογή other στο κάτω μέρος ,στον τομέα του Format και από το Output format **επιλέγουμε Intel-standard** .
Έπειτα ενεργοποιούμε το Override default και βάζουμε στο όνομα του project που εμφανίζεται την **κατάληξη .hex.**
Εάν θέλουμε να δημιουργήσουμε ένα αρχείο για να κάνουμε προσομοίωση με κάποιο πρόγραμμα , επιλέγουμε από το Format την επιλογή Debug info with terminal I/O .
Ακόμα από το Output format την επιλογή debug (ubrof) .
Τέλος πατάμε OK

4.3 Γράψιμο του κώδικα σε C .

Στην συνεχεία κάνουμε **new file ➔ source code**
Στο πλαίσιο που εμφανίζεται γράφουμε τον κώδικα . Ένα παράδειγμα κώδικα φαίνεται παρακάτω:

```
#include<io8515.h>    //σύνδεση βιβλιοθήκης της C

void main(void)        // συνάρτηση main
{
DDRB=0XFF;             // βρόγχος χωρις τέλος

while(1)
{
PORTB=0x00;            // τιμή εξόδου 00000000 στην πορτα B για να ανάβουν τα 8 led
}}
```

Αφού γραφεί ο κώδικας κάνουμε save as και του δίνουμε ένα όνομα .
Στην συγκεκριμένη περίπτωση το ονομάζω Askisi1.c (και βάζω την κατάληξη .c)Έπειτα πηγαίνω στο πάνω μέρος στο project **και επιλέγω Files** .
Τότε εμφανίζεται το παρακάτω παράθυρο :



Επιλέγουμε το αρχείο και **πατάμε Add** . Τότε το αρχείο προστίθεται στο χώρο που γράφει Files in Group .Εδώ εάν υπάρχουν κώδικες που συνεργάζονται τότε τους προσθέτουμε και αυτούς .
Τέλος πατάμε Done και κλείνει το παράθυρο .

4.4 Μεταγλώτιση (compile) και διασύνδεση (link)

Από την μπάρα εργαλείων επιλέγουμε **Project➔compile** και γίνεται η μεταγλώτιση. Για να κάνουμε διασύνδεση επιλέγουμε **Project➔link** και γίνεται η διασύνδεση. Αυτό μπορεί να γίνει κατευθείαν από το πληκτρο. 
Βέβαια υπάρχει η επιλογή **Build all** που κάνει και τα δυο μαζί και βρίσκεται στο Project➔Build all
Στο τέλος εμφανίζεται ένα παράθυρο που μας ενημερώνει εάν έγιναν όλα σωστά ή έγιναν λάθη και ποια είναι αυτά .
Τότε αναλόγως με το αν εμείς έχουμε επιλέξει εξ αρχής από την ρύθμιση του project, Release ή Debug δημιουργούνται ένα αντίστοιχο αρχείο στον φάκελο AVRprograms Test .
Το πρώτο ονομάζεται **Release** και μέσα εκεί υπάρχουν τρία άλλα αρχεία. Το **Exe,List,Obj** .Μέσα στο Exe βρίσκονται τα εκτελέσιμα αρχεία που ανοίγουμε από τον **AvrProg ή από άλλο προγραμματιστή** για να προγραμματίσουμε με τον μικροελεγκτή.

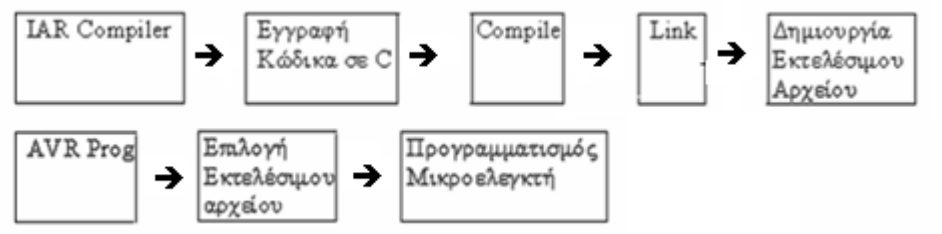
Το δεύτερο ονομάζεται Debug και εκεί υπάρχουν τρία αρχεία . Το Exe,List,Obj .

Μέσα στο Exe βρίσκονται τα αρχεία τύπου .d90 τα οποία ανοίγουν από το Avr Studio για να γίνει η προσομοίωση της λειτουργίας του προγράμματος στον μικροελεγκτή που διαθέτουμε

4.5 Διαγράμματα Προγραμματισμού

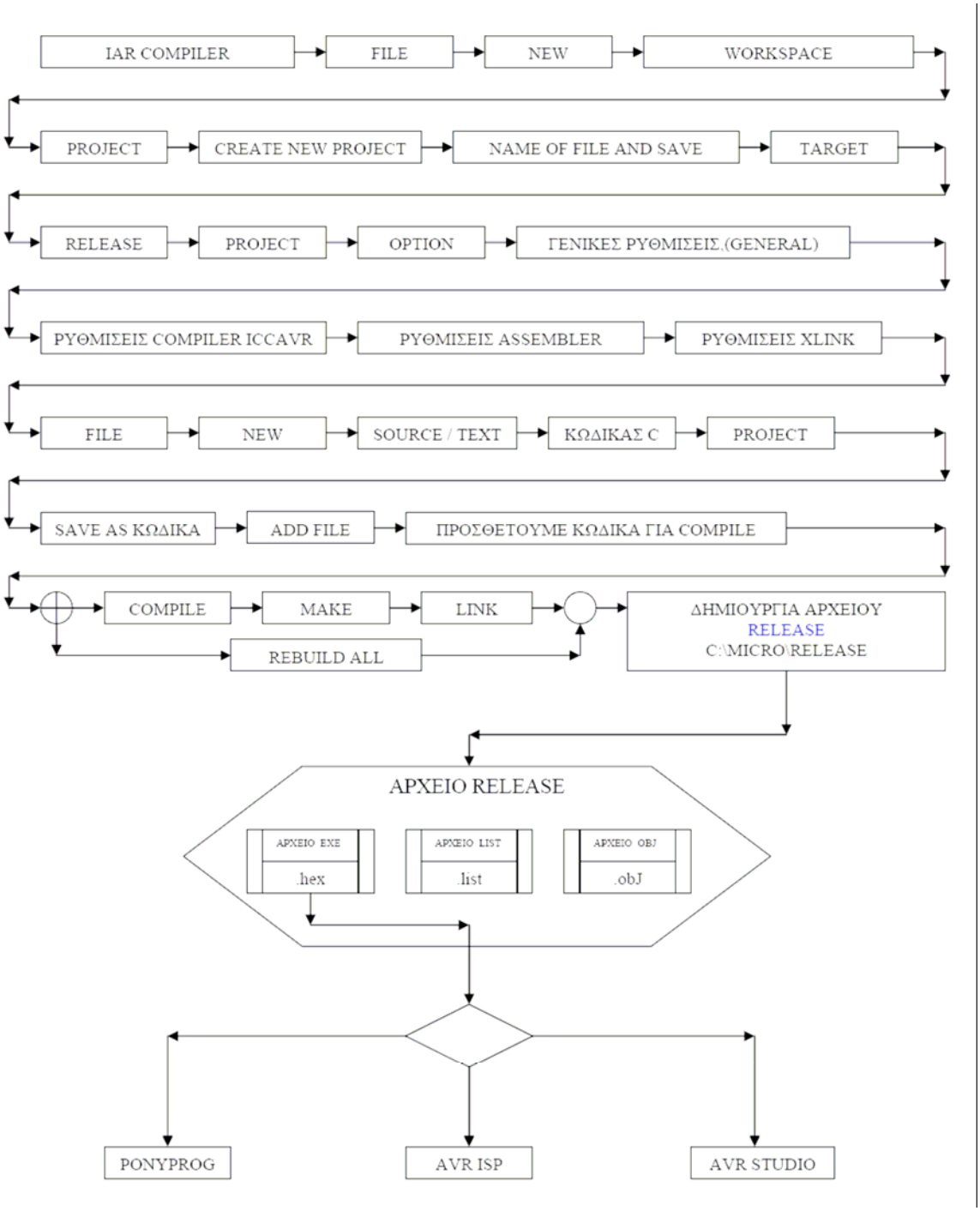
Παρακάτω φαίνεται ένα διάγραμμα που περιγράφει τη διαδικασία προγραμματισμού ενός μικροελεγκτή με τα εργαλεία που διαθέτουμε και άλλο ένα διάγραμμα που δείχνει ακριβώς τα ίδια αλλά με μεγαλύτερη λεπτομέρεια.

Γενικό διάγραμα προγραμματισμού μικροελεγκτή



Ο προγραμματισμός του μικροελεγκή μπορεί να γίνει και με άλλα προγράμματα όπως το ISP Programmer ή το Ponyprog .

Αναλυτικό διάγραμμα συνολικής διαδικασίας προγραμματισμού ενός μικροελεγκτή

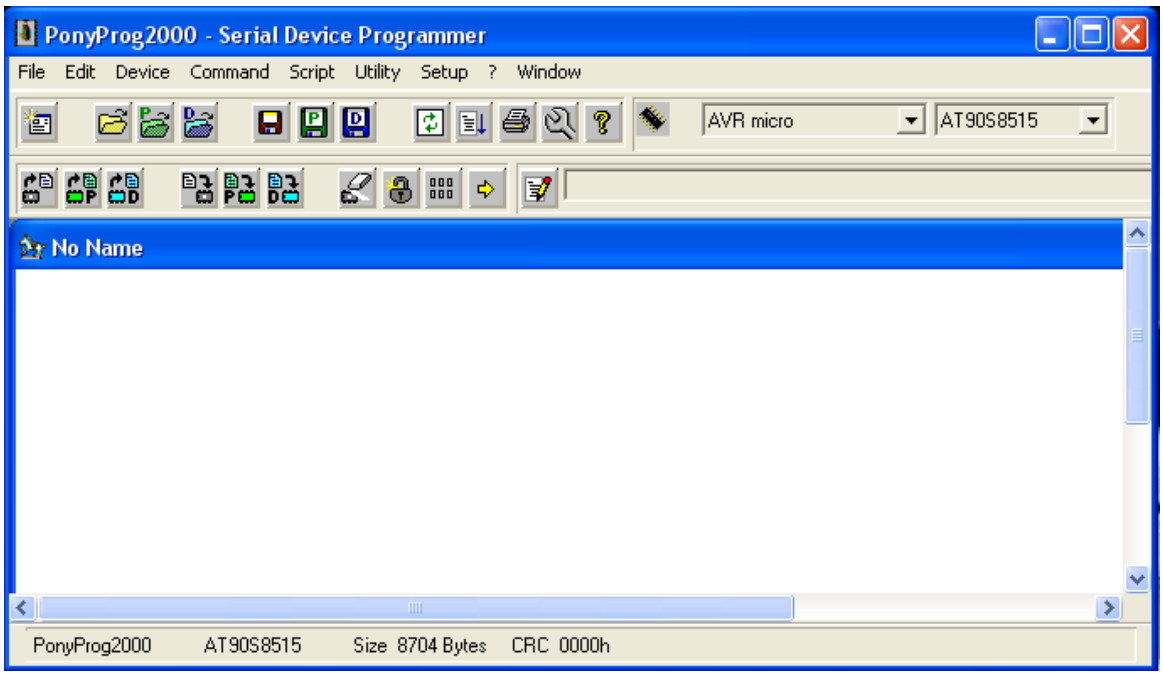


5.PONYPROG

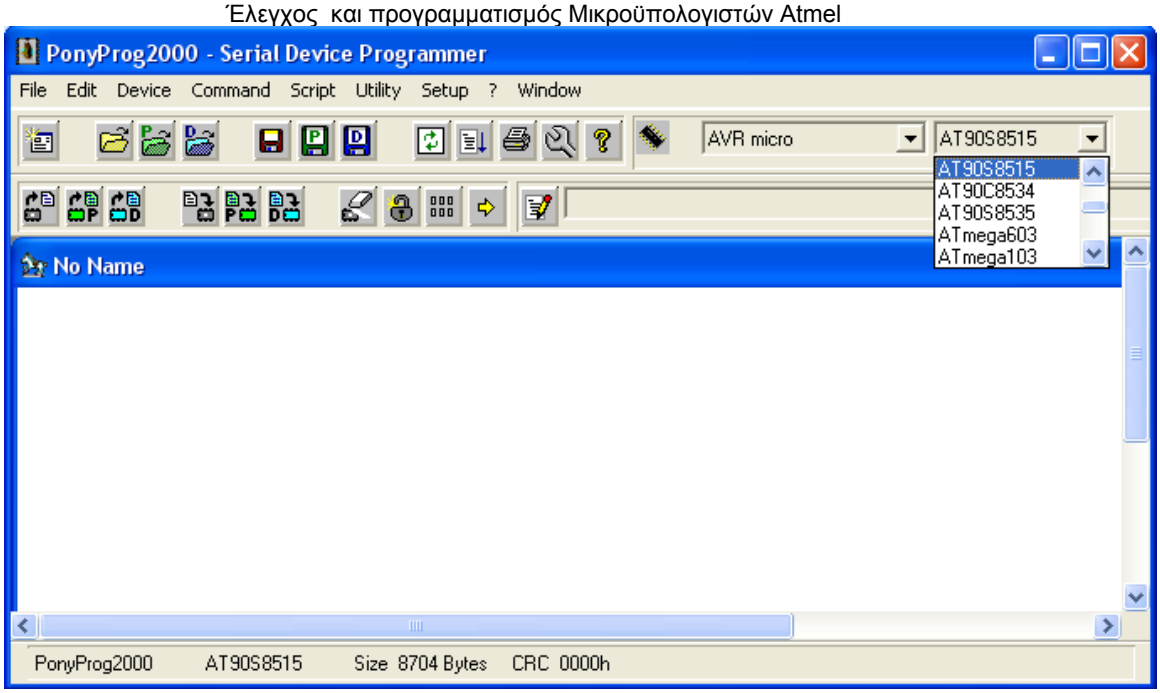
5.1 Προγραμματίζοντας με το Ponyprog 2000

Το πρόγραμμα αυτό χρησιμοποιείται (εφόσον έχει γραφτεί ο κώδικας και έχει γίνει compile και link) για να κατεβάσει το εκτελέσιμο αρχείο *.hex στον μικροελεγκτή .
Ο programmer αυτός μπορεί να προγραμματίσει όλους τους μικροελεγκτές της σειράς AT90S καθώς και αρκετούς της σειράς AT89S .Στην περίπτωση μας ο προγραμματισμός γίνεται στην μνήμη Flash του μικροελεγκτή .

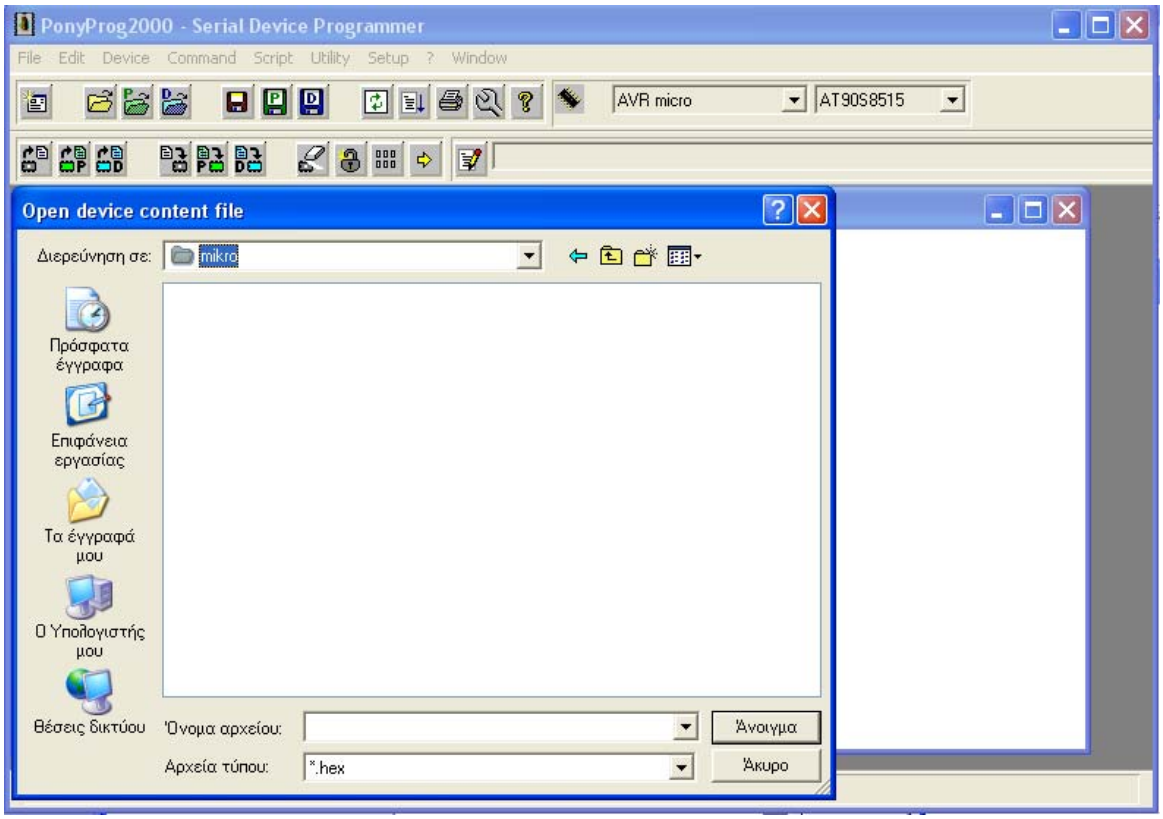
Ανοίγουμε το πρόγραμμα Ponyprog 2000 κλικάροντας το εικονίδιο :



- Επιλέγουμε τον μικροελεγκτή μας AT90S8515



Στη συνέχεια επιλέγουμε **File ➔ Open device file** και φορτώνουμε το αρχείο ***.hex** που αντιστοιχεί στο project μας.



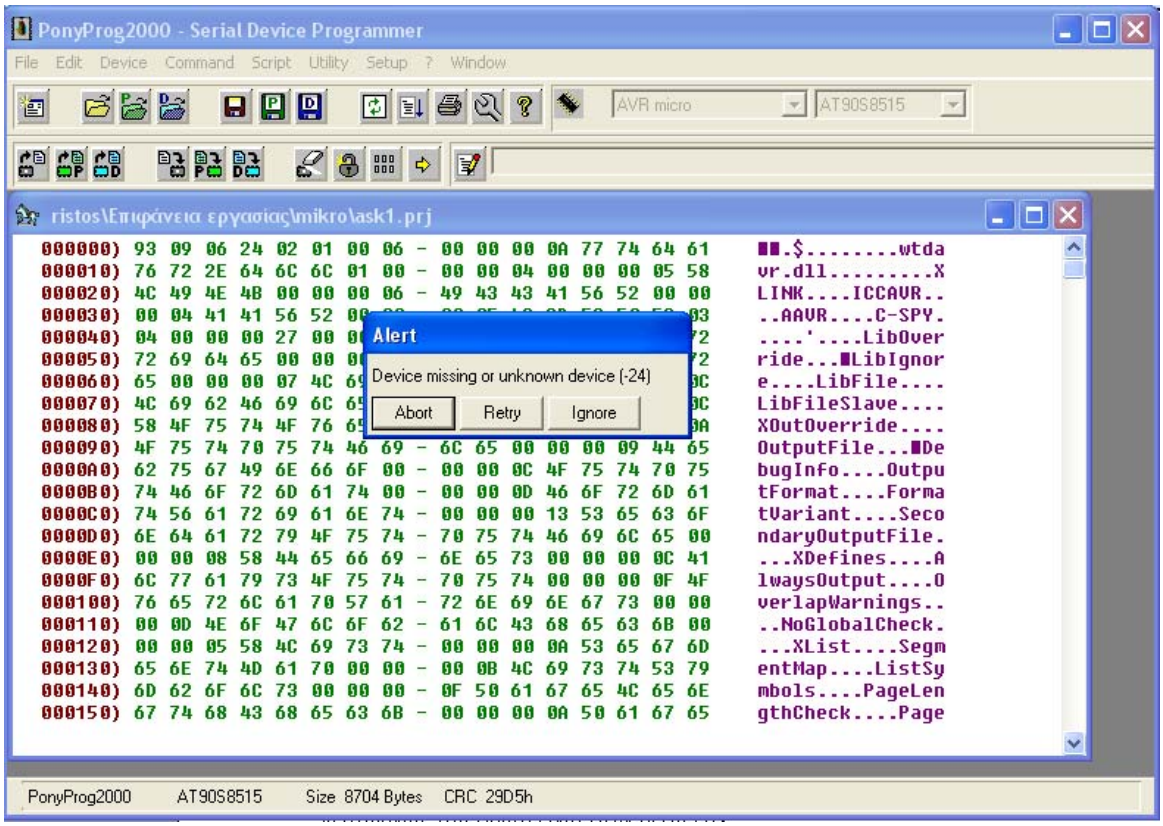
Το αρχείο αυτό βρίσκεται στο directory που σώνουμε τα projects στο φάκελο Release και μέσα σε εκείνο που το ονομάζουμε Exe
Τέλος επιλεγούμε **command➔write all .**

Αυτό το κάνουμε με το εικονίδιο **write device** 

Αν θέλουμε να ανακτήσουμε το πρόγραμμα από το μικροελεγκτή πατάμε το εικονίδιο **read device** 

Για να διαγράψουμε το περιεχόμενο του μικροελεγκτή πατάμε το εικονίδιο **Erase all the device** 

Σε περίπτωση που εμφανιστεί το παρακάτω μήνυμα ελέγξτε εάν Board(αναπτυξιακή πλακέτα) είναι συνδεμένο σε κάποια θύρα του υπολογιστή και αν ο διακόπτης λειτουργίας του Board είναι στην θέση ON.



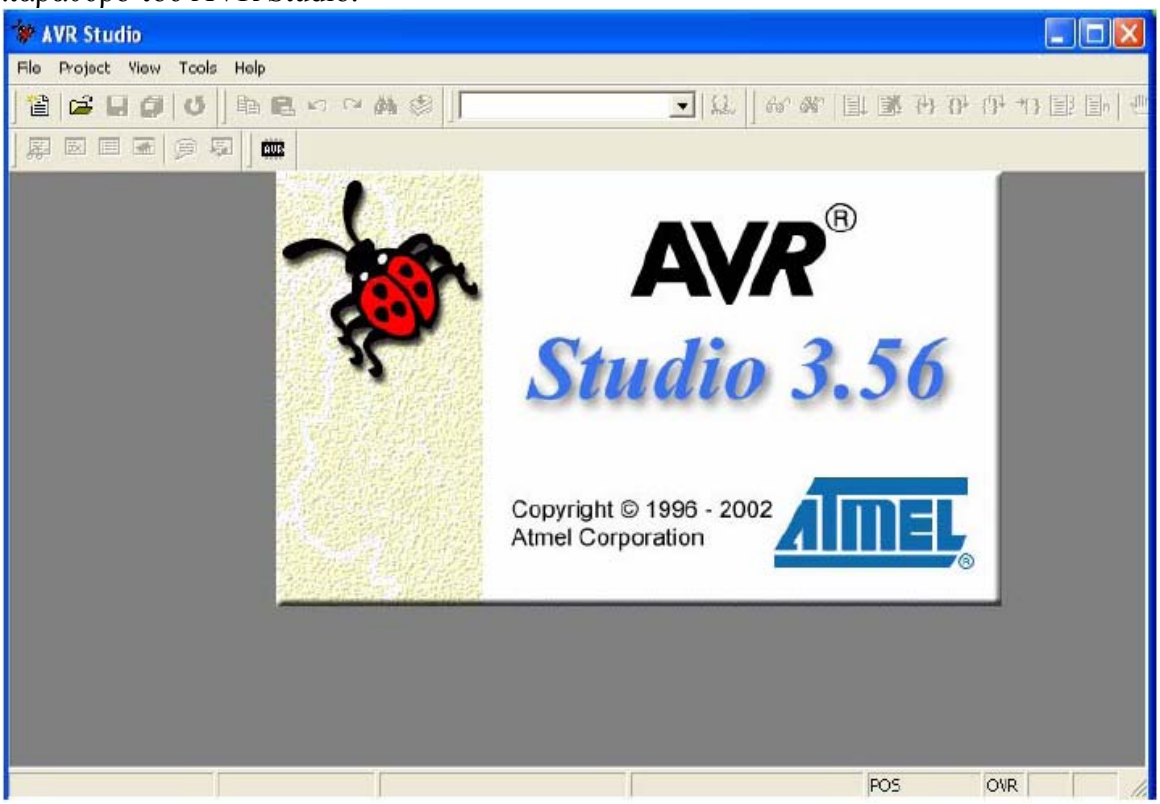
5.2 AVR studio 3.56

Το κεφάλαιο αυτό αποτελεί μια περιγραφή του AVR Studio.Το AVR Studio είναι το πρόγραμμα εκείνο που χρησιμοποιείται τελικώς (αφού έχει γραφτεί ο κώδικας και έχει γίνει compile και link), για να κατεβάσει το εκτελέσιμο αρχείο (exe) (που στην περίπτωση μας έχει τη μορφή .hex), στο μικροελεγκτή.
Το AVR Studio μπορεί να προγραμματίσει όλους τους μικροελεγκτές των σειρών AT90S, AT tiny και AT mega. Το παραπάνω πρόγραμμα λειτουργεί μέσα από τα Microsoft Windows 9x\2000\NT\XP αλλά και μέσα από το περιβάλλον του DOS. Εμείς θα ασχοληθούμε με την έκδοση 3.56. Το AVR Studio μπορεί να χρησιμοποιηθεί με το σύστημα STK 500, όπου τέτοια αναπτυξιακά κιτ βρίσκονται στο εργαστήριο σας.

5.3 Χρησιμοποιώντας το AVR Studio

Το πρόγραμμα ξεκινά με διπλό κλικ στο εικονίδιο 

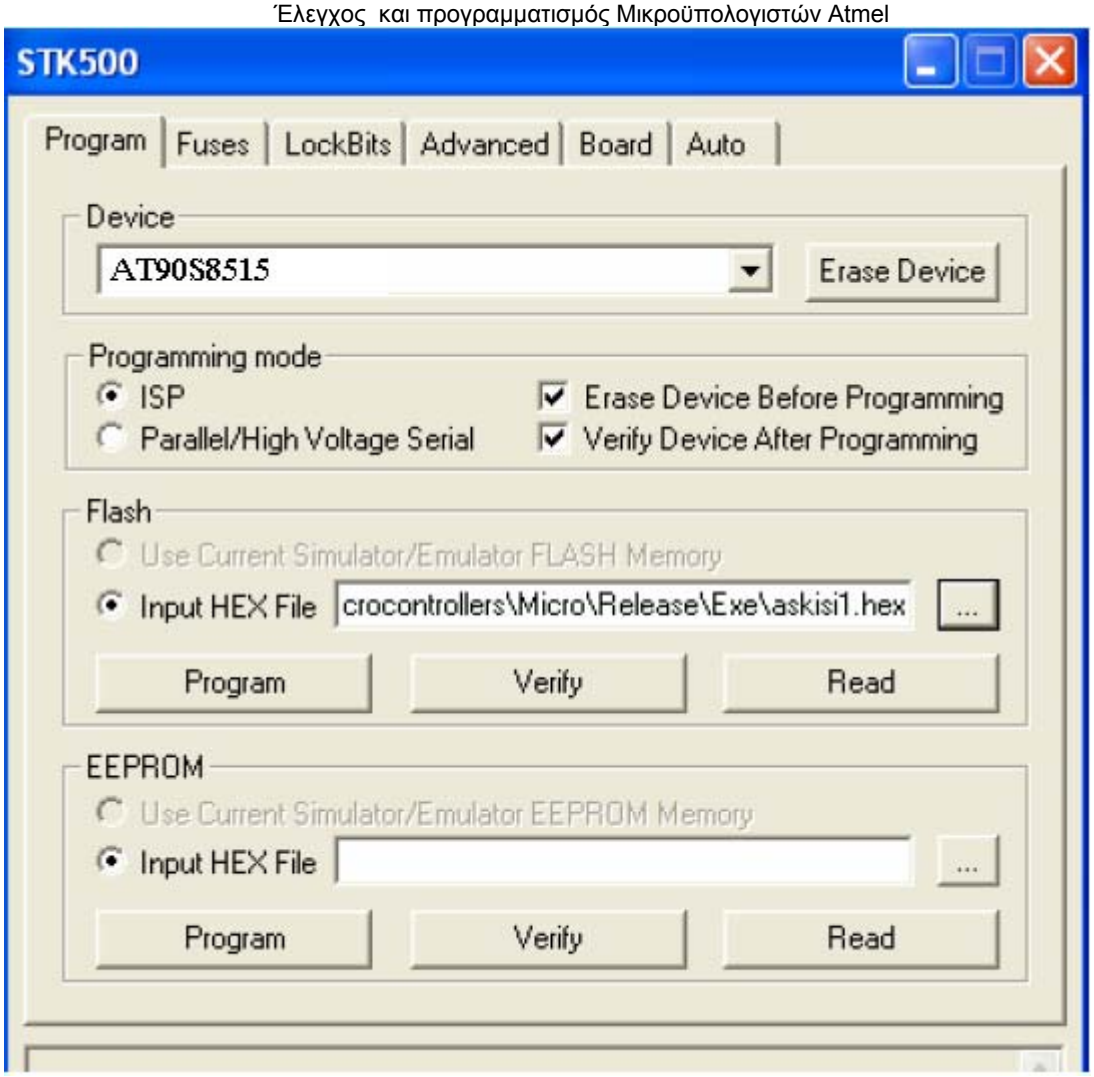
.Τότε θα εμφανιστεί το παράθυρο του AVR Studio.



Έπειτα επιλέγουμε ‘ Tools’□□ ‘STK 500’ και εμφανίζεται το παρακάτω παράθυρο:

(Είτε πατώντας το εικονίδιο 

όπου βρίσκεται στην μπάρα εργαλείων).



Αυτό το παράθυρο αποτελείται από έξι σελίδες. Η πρώτη σελίδα με το όνομα 'Program' είναι υπεύθυνη για τον προγραμματισμό του μικροελεγκτή. Θα αναφερθούμε σε αυτή αργότερα με λεπτομέρεια. Η δεύτερη σελίδα με το όνομα 'fuses' είναι υπεύθυνη για τα fuses του μικροεπεξεργαστή. Αυτή, καθώς και η τρίτη σελίδα με το όνομα lock bits (ελέγχει τα bit κλειδώματος του μικροελεγκτή) και δεν είναι διαθέσιμες όταν προγραμματίζουμε το STK500 σειριακά, διότι δεν υποστηρίζεται η σειριακή ανάγνωση των fuses και των lock bits. Στη σελίδα με όνομα 'advanced' , μπορούμε να διαβάσουμε τα signature bits του μικροελεγκτή που χρησιμοποιούμε. Η επόμενη σελίδα 'board' μας επιτρέπει να δούμε πληροφορίες σχετικά με την πλακέτα που χρησιμοποιούμε και τα ηλεκτρικά χαρακτηριστικά της, όπως την τάση αναφοράς (δηλ. το λογικό 1) και τη συχνότητα του κρυστάλλου. Επίσης μπορούμε να δούμε την έκδοση του Hardware και του Software που χρησιμοποιούμε. Τέλος στη σελίδα με όνομα 'Auto' μπορούμε να προγραμματίσουμε κάποιες ενέργειες ώστε να γίνουν αυτόματα.

Η σελίδα που θα χρησιμοποιούμε κάθε φορά που θα θέλουμε να κατεβάσουμε ένα πρόγραμμα στο μικροελεγκτή είναι η σελίδα με όνομα 'program' και χωρίζεται σε τέσσερα τμήματα.

• Device: Από εδώ διαλέγουμε το μικροελεγκτή που θέλουμε να προγραμματίσουμε. Στο εργαστήριο χρησιμοποιούμε τον AT90S8515

• Programming mode: Επιλέγουμε τον τρόπο με τον οποίο θα προγραμματίσουμε το μικροελεγκτή. (Συνήθως επιλεγουμε ISP με τσεκαρισμένα τα ***Erase device before programming*** και ***verify device after programming***.

• Flash: Από εδώ μπορούμε να χειριστούμε την Flash του μικροελεγκτή. Η πιο συχνή

λειτουργία είναι να διαλέγουμε ένα αρχείο για κατέβασμα (hex) και να το φορτώνουμε στο μικροελεγκτή. Εναλλακτικά έχουμε την επιλογή να διαβάσουμε την Flash και να αποθηκεύσουμε τα δεδομένα σε κάποιο αρχείο.
• EEPROM: Όπως και με το τμήμα που αναφέρεται στην Flash, έτσι από εδώ μπορούμε να ελέγχουμε την EEPROM του μικροελεγκτή.

5.4 Επιλέγοντας μικροελεγκτή

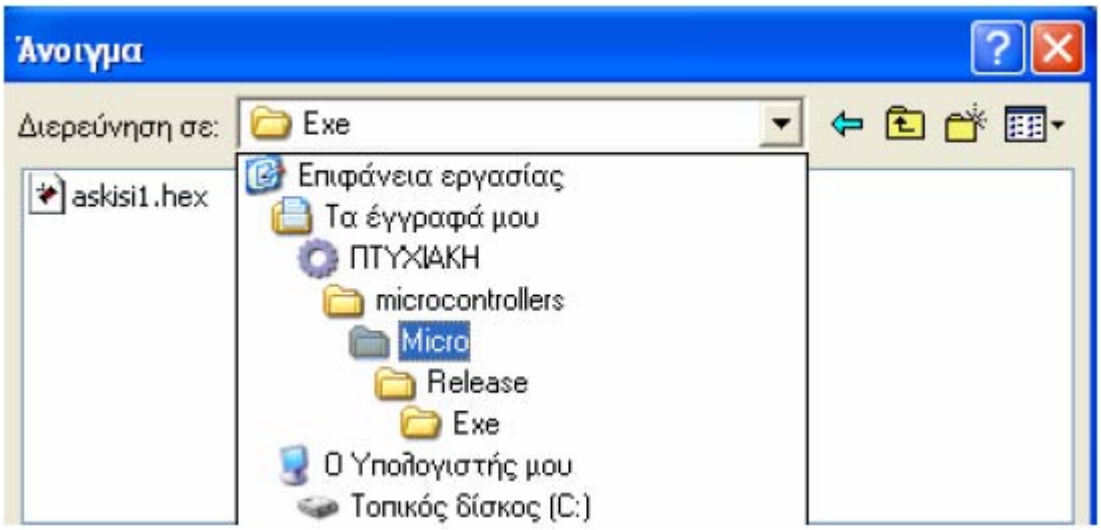
Το πρώτο βήμα για τον προγραμματισμό ενός μικροελεγκτή είναι η επιλογή του από το μενού με το όνομα Device.Το AVR Studio είναι φτιαγμένο και για το STK 500, οπότε απλώς διαλέγουμε το μικροελεγκτή που θέλουμε να προγραμματίσουμε. Το επόμενο βήμα είναι να διαλέξουμε ένα αρχείο για να το 'κατεβάσουμε' στο μικροελεγκτή μας.

5.5 Φορτώνοντας ένα Intel Hex αρχείο

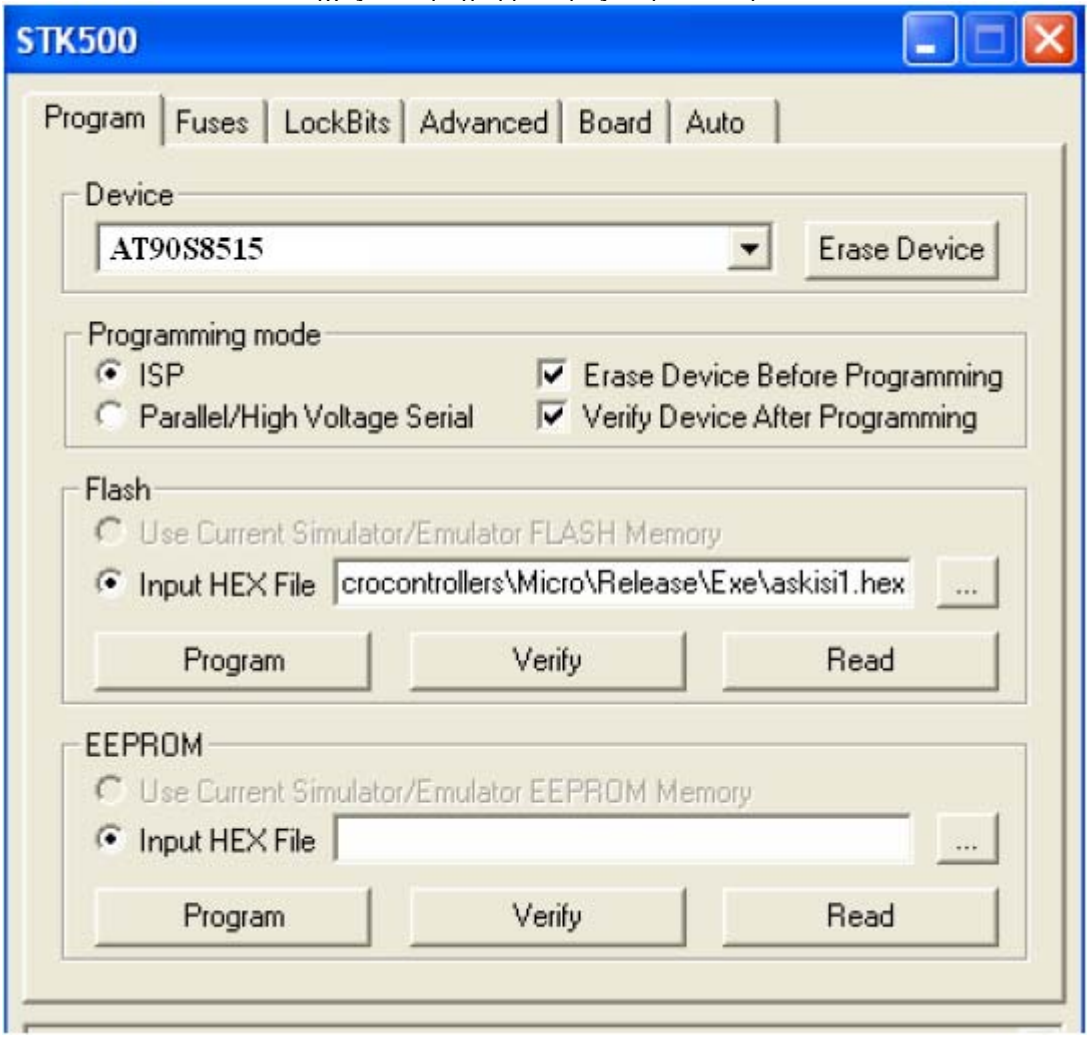
Το επόμενο βήμα είναι να φορτώσουμε ένα αρχείο Intel hex για τον προγραμματισμό της μνήμης flash ή της EEPROM του μικροελεγκτή.

Επιλέγουμε το κουμπί  και εμφανίζεται ένα παράθυρο, μέσα στο οποίο βρίσκονται τα αρχεία Hex που έχουν δημιουργηθεί από τον C compiler (μετά από compile και link) και μετά επιλέγουμε το επιθυμητό αρχείο.

Τα αρχεία αυτά βρίσκονται στο φάκελο που έχει σωθεί το αντίστοιχο project (βλέπουμε το φάκελο που γράφει Release και μέσα σε αυτό, εκείνο που ονομάζεται Exe).



Μόλις γίνει open σε ένα αρχείο τότε το παράθυρο του AVR Studio έχει την παρακάτω μορφή:

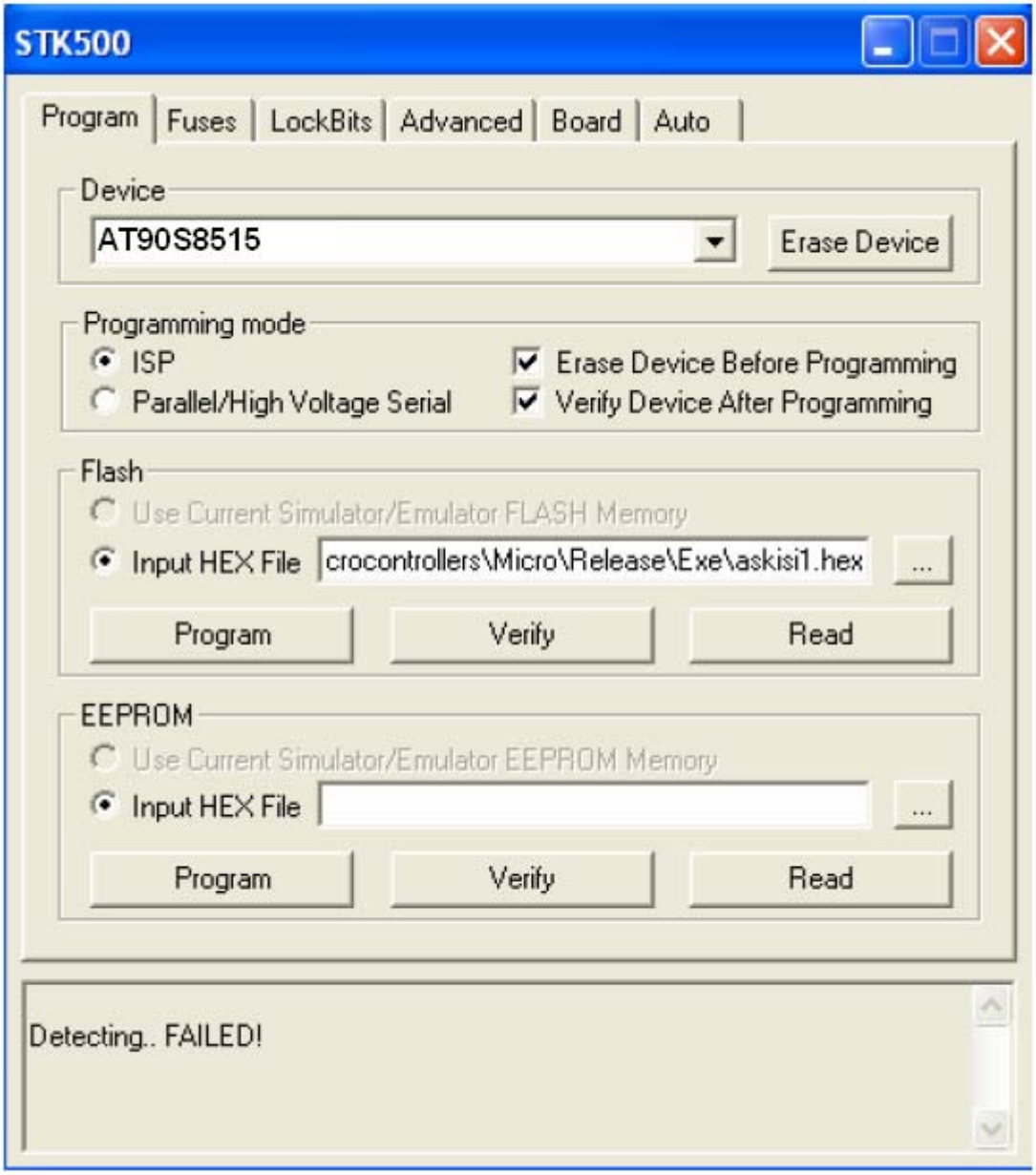


Τώρα πλέον μπορούμε να προγραμματίσουμε το μικροελεγκτή με την παρακάτω διαδικασία.:

5.6 Προγραμματισμός του μικροελεγκτή

Το τελευταίο βήμα στη διαδικασία του προγραμματισμού είναι να φορτωθεί το επιλεγμένο αρχείο hex στο μικροελεγκτή.

Με το πλήκτρο Program που βρίσκεται στον τομέα της μνήμης Flash, προγραμματίζεται η μνήμη Flash, ενώ αν επιλεχθεί το πλήκτρο Program που βρίσκεται στον τομέα της EEPROM προγραμματίζεται η EEPROM του μικροελεγκτή. Όταν ξεκινάει η διαδικασία προγραμματισμού, τότε στο κάτω τμήμα του AVR Studio φαίνονται κάποια μηνύματα ενδεικτικά της κατάστασης του προγραμματισμού.
Αν ο προγραμματισμός πετύχει τότε βλέπουμε όλα τα μηνύματα να τελειώνουν με ένα ***οκ***. Αλλιώς παρατηρούμε ότι κάποια μηνύματα έχουν failed, άρα και ο προγραμματισμός απέτυχε μερικώς ή ολικώς. Παρακάτω φαίνονται τα μηνύματα που παράγει ένας επιτυχής προγραμματισμός.



Εδώ ας σημειώσουμε ότι εάν θελήσουμε να προγραμματίσουμε και τη flash μνήμη και την EEPROM τότε θα πρέπει πρώτα να προγραμματίσουμε τη Flash και μετά την EEPROM. Και αυτό γιατί ο προγραμματισμός της Flash πάντα ξεκινάει με ολική διαγραφή όλων των μνημών του clip (chip erase) συμπεριλαμβανομένης και της EEPROM. Έτσι εάν υπάρχει πρόγραμμα στην EEPROM θα σβηστεί. Δεν συμβαίνει όμως το ίδιο και με την EEPROM.

5.7 Επαλήθευση

Για την επαλήθευση ότι τα περιεχόμενα που προγραμμάτισαν τη μνήμη flash ή την EEPROM είναι εκείνα που βρίσκονται στο αρχείο hex που έχει επιλεχθεί και δεν έχει γίνει κάποιο λάθος κατά τον προγραμματισμό, πατάμε το πλήκτρο ***Verify*** που βρίσκεται στον τομέα της flash ή της EEPROM αντίστοιχα, αναλόγως με το ποια

προγραμματίζουμε. Εάν η επαλήθευση είναι σωστή, τότε στο κάτω τμήμα του AVR Studio θα δούμε το μήνυμα: ***FLASH contents is equal to file….OK***
Ο πιο συνηθισμένος λόγος που η επαλήθευση δεν έγινε σωστά, είναι ότι έχουν προγραμματιστεί τα bit κλειδώματος (lock bits) του μικροελεγκτή. Αυτό ακυρώνει την επιλογή επαλήθευσης του μικροελεγκτή.
Το μήνυμα αυτό μπορεί επίσης να εμφανιστεί όταν κατά τον προγραμματισμό είναι συνδεδεμένα στα Ports του μικροελεγκτή διάφορα κυκλώματα εξωτερικά. Γι'αυτό κατά τον προγραμματισμό δεν πρέπει να είναι συνδεδεμένο τίποτα εκεί. Προσοχή αν έχουμε χρησιμοποιήσει εξωτερικό κρύσταλλο και έχουμε αλλάξει τα jumpers στο αναπτυξιακό STK500, να υπάρχει πάνω ο κρύσταλλος.
Ανάγνωση των περιεχομένων της μνήμης του μικροελεγκτή (flash ή EEPROM) για να διαβαστούν τα περιεχόμενα που βρίσκονται στη μνήμη flash ή EEPROM πατάμε το πλήκτρο ***Read*** στον τομέα της αντίστοιχης μνήμης και εμφανίζεται αυτόματα ένα παράθυρο που μας ρωτάει πού θέλουμε να αποθηκεύσουμε τη μνήμη του μικροελεγκτή. Αν επιλέξουμε ένα αρχείο που ήδη υπάρχει, τότε μας ρωτάει αν θέλουμε να το αντικαταστήσουμε.

5.8 Χρησιμοποιώντας τα προχωρημένα χαρακτηριστικά του STK500 Programmer

Για να χρησιμοποιήσουμε τα προχωρημένα χαρακτηριστικά του STK 500 programmer απλώς αλλάζουμε τη σελίδα, είτε στη σελίδα με τα fuses ή σε αυτήν με τα lock bits. Από εδώ μπορούμε να διαβάσουμε και να προγραμματίσουμε τα lock bits και τα fuses. Αν βέβαια το δοκιμάσουμε θα πάρουμε ένα μήνυμα λάθους καθώς δεν γίνεται να προγραμματίσουμε τα lock bits και τα fuses από τη σειριακή (ISP) επικοινωνία.

5.9 Προγραμματισμός των bits κλειδώματος (lock bits)

Για να προγραμματιστούν τα lock bits, επιλέγουμε την επιθυμητή ρύθμιση από το πάνω τμήμα των επιλογών και πατάμε program. Αν δεν βγει κάποιο μήνυμα λάθους τότε τα lock bits έχουν προγραμματιστεί. Για να είμαστε σίγουροι, μπορούμε να πατήσουμε το πλήκτρο verify ώστε να επιβεβαιώσουμε ότι τα lock bits έχουν προγραμματιστεί σωστά.

5.10 Προγραμματισμός των bits ασφάλειας (fuse bits)

Για να προγραμματίσουμε τα fuse bits απαιτείται παράλληλος προγραμματισμός. Κατά τα αλλά ισχύει ότι ισχύει και για τα lock bits, δηλαδή επιλέγουμε τις ρυθμίσεις και πατάμε program.

5.11 Το πλήκτρο chip erase

Όταν πατηθεί το πλήκτρο erase device στην πρώτη σελίδα , τότε ο μικροελεγκτής καθαρίζεται. Αυτό σημαίνει ότι σβήνονται τα περιεχόμενα της μνήμης flash και EEPROM, καθώς και τα lock bits. Τα fuse bits δεν επηρεάζονται από το chip erase.

5.12 Δυσλειτουργίες αναπτυξιακού συστήματος

Παρακάτω φαίνονται μερικά από τα προβλήματα που ενδέχεται να παρουσιαστούν κατά τη χρήση του προγράμματος:
Προβλήματα στην επικοινωνία με το Board. Ο πρώτος έλεγχος που γίνεται από το AVR Studio όποτε του ζητάμε να επικοινωνήσει με το μικροελεγκτή, είναι ο έλεγχος ύπαρξης

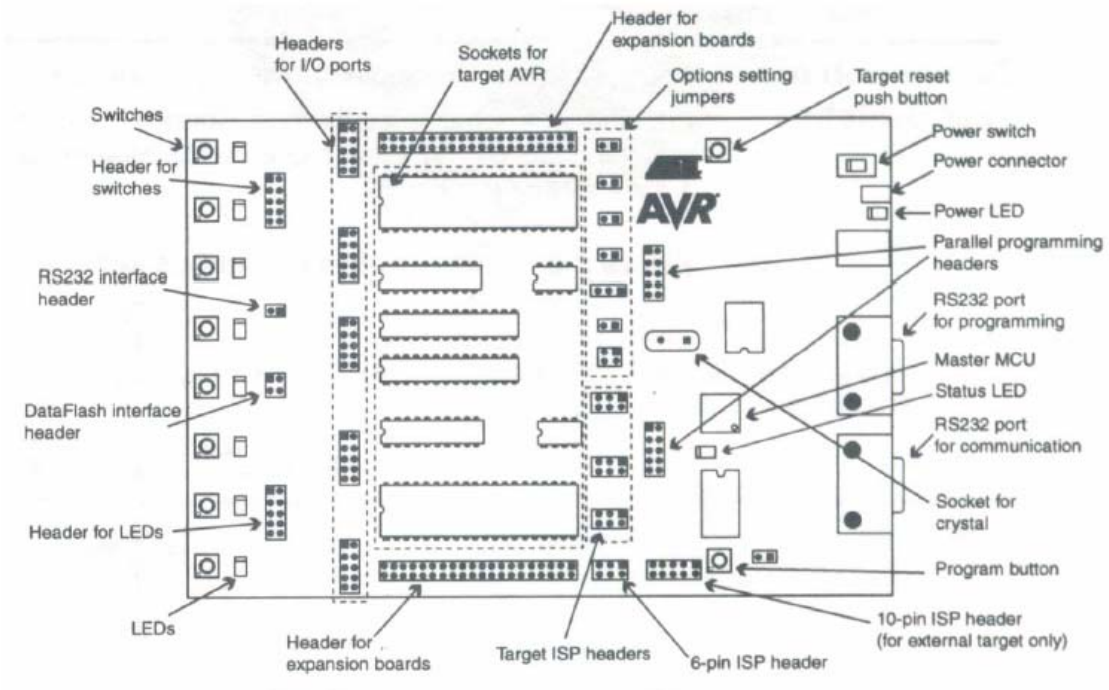
του σωστού board. Αν λοιπόν κατά την εκκίνηση προγραμματισμού με το board εμφανιστεί το μήνυμα στο κάτω μέρος του παραθύρου του STK500 - ***Detecting ..FAILED!.-***Ελέγχουμε τα παρακάτω:
✓ Υπάρχει τροφοδοσία;
✓ Είναι ο διακόπτης ανοικτός ;(αν η πλακέτα έχει σωστή τροφοδοσία και ο διακόπτης είναι ανοικτός τότε θα είναι αναμμένη το κόκκινη φωτοδίοδος δίπλα στον διακόπτη).
✓ Είναι το καλώδιο σωστά συνδεδεμένο στην υποδοχή δίπλα στην τροφοδοσία;
✓ Το ίδιο το καλώδιο είναι σωστό; (Μόνο σειριακό ένα προς ένα)
✓ Μήπως κάποιο άλλο πρόγραμμα των windows χρησιμοποιεί τη σειριακή πόρτα;
Προβλήματα κατά τον προγραμματισμό και την επαλήθευση:
Εάν εμφανιστούν τα μηνύματα 'Programming failed' ή 'Verified failed' τότε ελέγξτε τα παρακάτω:
✓ Έχουμε τοποθετήσει το μικροελεγκτή στη βάση του;
✓ Είναι τοποθετημένος με το σωστό τρόπο (προσανατολισμό);
✓ Βρίσκονται όλα τα pins μέσα στη βάση;
✓ Έχουμε τοποθετήσει το σωστό μικροελεγκτή στην πλακέτα;
✓ Μήπως έχουμε συνδέσει κάποιο κύκλωμα στα Ports (Port B);
✓ Έχουμε επιλέξει σωστό μικροελεγκτή από τις επιλογές στο παράθυρο του AVR Studio.
✓ Έχουμε βάλει την καλωδιοταινία στη σωστή θέση;

Σημείωση: Εάν προσπαθήσουμε να προγραμματίσουμε έναν μικροελεγκτή με λάθος επιλογή από τον τομέα Device στο παράθυρο του AvrStudio, υπάρχει περίπτωση το πρόγραμμα να κολλήσει. Στην περίπτωση αυτή το κίτρινο LED πάνω στην πλακέτα θα είναι συνεχώς αναμμένο. Για να αντιμετωπίσουμε την κατάσταση αυτή είτε πατάμε το πλήκτρο RESET που βρίσκεται στην πλακέτα είτε κλείνουμε την τροφοδοσία σε αυτήν. Κλείνουμε το πρόγραμμα Avr Studio και μετά ανοίγουμε πάλι την τροφοδοσία και το πρόγραμμα..

Σημείωση: Τα διάφορα εξαρτήματα μπορεί να καταστραφούν εξαιτίας κακού ESD περιβάλλοντος (ESD: Electrostatic Discharge ➔ Ηλεκτροστατική εκφόρτιση) ή κακών χειρισμών.
Εάν υπάρχει η υποψία για χαλασμένο μικροελεγκτή τότε δοκιμάζουμε και άλλους όμοιους, για να δούμε εάν συμπεριφέρεται το ίδιο. Εάν όχι, είναι πιθανόν να είναι κατεστραμμένος. Εάν ναι, τότε υπάρχει περίπτωση να έχει υποστεί ζημία η πλακέτα.

Οι χειρισμοί μας πρέπει να είναι ιδιαίτερα προσεκτικοί και να γίνονται με ιδιαίτερες προφυλάξεις:

✓ Δεν πρέπει στην πλακέτα η τάση τροφοδοσίας να ξεπεράσει το επιτρεπόμενο όριο (10-15V DC) και πρέπει να προσέχουμε να μην προκαλούμε βραχυκυκλώματα. Και αυτό γιατί υπάρχει ο κίνδυνος να προκληθεί μόνιμη βλάβη σε αυτή.
✓ Δεν αλλάζουμε επεξεργαστή στην πλακέτα όταν βρίσκεται υπό τάση, γιατί μπορεί να καταστραφεί ο επεξεργαστής.
✓ Για εξωτερικό κρύσταλλο οι ρυθμίσεις στα Fuse bits γίνονται μόνες τους αρκεί να έχουμε βάλει τα σωστά jumpers πάνω στην πλακέτα , TARGET, RESET, XTALI, DSCSEL


Header for expansion boards
Headers for I/O ports
Sockets for target AVR
Options setting jumpers
Target reset push button
Switches
Header for switches
Power switch
Power connector
Power LED
Parallel programming headers
RS232 interface header
RS232 port for programming
Master MCU
Status LED
DataFlash interface header
RS232 port for communication
Socket for crystal
Header for LEDs
Program button
LEDs
10-pin ISP header (for external target only)
Header for expansion boards
Target ISP headers
6-pin ISP header

6 AVR Studio User Manual

Εισαγωγή

Το AVR Studio είναι ένα αναπτυξιακό εργαλείο για τους μικροελεγκτές της οικογένειας . Εδώ περιγράφεται πώς να εγκαταστήσετε και να χρησιμοποιήσετε το AVR Studio .
Το AVR Studio επιτρέπει στον χρηστή να ελέγχει πλήρως την εκτέλεση των προγραμμάτων στο εξωτερικό (hardware)προσομοιωτή . Το AVR Studio επιτρέπει την εκτέλεση σε επίπεδο πηγαίου κώδικα της assembly που έχει δημιουργηθεί με τον assembler της Atmel και προγράμματα γραμμένα σε C αφού γίνουν compile με τον compiler ICCA90 της IAR Systems για τους μικροελεγκτές της οικογένειας AVR .

6.1 Εγκατάσταση του AVR Studio

Για να εγκαταστήσουμε το AVR Studio Windows 9X ή Windows NT 4.0:
1.Εισάγουμε την AVR Studio Diskette 1 στο drive A
2.Κάνουμε κλικ στο Start και επιλέγουμε Run
3.Γράφουμε A SET UPστο πεδίο με τίτλο Open και πατάμε OK
4.Ακολουθούμε τις οδηγίες του προγράμματος εγκατάστασης

Για να εγκαταστήσουμε το AVR Studio σε Windows NT 3.51:
1. Εισάγουμε την δισκέτα AVR Studio Diskette 1 στο drive A
2. Επιλέγουμε Run από το μενού file
3. Γράφουμε A SET UP στο πεδίο με τίτλο Open και πατάμε OK
4. Ακολουθούμε τις οδηγίες του προγράμματος εγκατάστασης

Για να εγκαταστήσουμε το AVR Studio από το WEB

1.Εντοπίζουμε το ASTUDIO.EXE στο τμήμα της software τηςAtmel
2.Κατεβάζουμε το ASTUDIO.EXE σε ένα προσωρινό κατάλογο
3.Τρέχουμε το ASTUDIO.EXE από τον τοπικό δίσκο . Θα αποσυμπιέσει το πρόγραμμα εγκατάστασης
4.Τρέχουμε το SET UP.EXE
5.Ακολουθούμε τις οδηγίες του προγράμματος εγκατάστασης

Μόλις εγκαταστήσουμε το AVR Studio , μπορούμε να το τρέχουμε από το εικονίδιο που θα δημιουργήσει . Αν ο στόχος μας είναι να χρησιμοποιήσουμε έναν hardware προσομοίωση τότε πρέπει να έχουμε συνδέσει τον προσομοίωση και να έχουμε ανοίξει την τροφοδοσία του.(σημείωση: Εάν το AVR Studio δεν βρει τον εξομοιωτή όταν ξεκινάει τότε αυτόματα μπαίνει σε λειτουργία προσομοίωσης .Δηλ. Μέσω software).

6.2 Περιγραφή

Αυτό το τμήμα δίνει μια σύντομη περιγραφή των βασικών χαρακτηριστικών του AVR Studio . Για να εκτελέσουμε ένα πρόγραμμα με το AVR Studio πρέπει πρώτα να έχει γίνει compile με τον compiler της IAR Systems ή να έχει γίνει με τον assembler της Atmel ώστε να δημιουργηθεί ένα αρχείο object που να μπορεί να διαβαστεί από το AVR Studio
Ενα παράδειγμα του πως μπορεί να είναι το AVR Studio κατά την εκτέλεση ενός προγράμματος δίνεται παρακάτω .Μαζί με το παράθυρο του πηγαίου κώδικα το AVR

Studio ορίζει και έναν αριθμό άλλων παραθύρων που μπορούν να χρησιμοποιηθούν για να δούμε τα διάφορα resources του μικροελεγκτή .

Το βασικό παράθυρο στο AVR Studio είναι το παράθυρο που φαίνεται ο πηγαίος κώδικας . Όταν ανοίγουμε ένα object αρχείο τότε το παράθυρο του πηγαίου κώδικα δημιουργείται από το AVR Studio . Το παράθυρο του πηγαίου προβάλει τον κώδικα που εκτελείται αυτήν την στιγμή και το σήμα επιλογής κειμένου δείχνει πάντα στην εντολή που πρόκειται να εκτελεστεί αμέσως μετά .

Η προεπιλογή είναι να εκτελείται ο κώδικας σε επίπεδο πηγαίου κώδικα , έτσι αν υπάρχουν πληροφορίες για τον πηγαίο κώδικα AVR Studio ξεκινάει να δουλεύει με τον πηγαίο . Επιπρόσθετα εκτός από την λειτουργία με πηγαίο κώδικα είτε σε C είτε σε assembly το AVR Studio βλέπει και εκτελεί προγράμματα σε disassembly επίπεδο . Ο χρήστης μπορεί να αλλάξει τον τρόπο λειτουργίας του AVR Studio από πηγαίο κώδικα σε disassembly,όταν η εκτέλεση του προγράμματος σταματάει .

Όλες οι απαραίτητες εκτελούμενες εντολές είναι διαθέσιμες στο AVR Studio και σε επίπεδο πηγαίου κώδικα (C assembley) και σε επίπεδο κώδικα μηχανής (disassembly code) .Ο χρήστης μπορεί να εκτελεί το πρόγραμμα με απλά βήματα είτε ακολουθώντας τον κώδικα μέσα στις συναρτήσεις είτε εκτελώντας τις συναρτήσεις χωρίς να μας δείχνει τον κώδικα τους . Επίσης μπορεί να εκτελεί μονοκόματα τον (υπόλοιπο) κώδικα μιας συνάρτησης μέχρι το τέλος της, ακόμα μπορεί να εκτελέσει τον κώδικα από το σημείο που έχει σταματήσει μέχρι όποιο σημείο αποφασίσει ο χρήστης .Τέλος μπορεί να σταματήσει την εκτέλεση του κώδικα ή να κάνει reset στον στόχο του προγράμματος . Επίσης ο χρήστης μπορεί να ορίσει έναν άπειρο αριθμό από σημεία διακοπής καθένα από τα οποία μπορεί να είναι είτε ενεργό είτε όχι. Τα σημεία διακοπής δεν διαγράφονται όταν κλείνουμε το πρόγραμμα .

Το παράθυρο του πηγαίου δίνει πληροφορίες για την ροή της εκτέλεσης του προγράμματος . Επιπρόσθετα το AVR προσφέρει έναν αριθμό από διάφορα παράθυρα τα οποία επιτρέπουν στον χρήστη να έχει πλήρη έλεγχο της κατάστασης κάθε στοιχείου που περιλαμβάνει ο υποθετικός μικροελεγκτής που εξομοιώνεται.

Τα διαθέσιμα παράθυρα είναι:
- Watch window: Δείχνει τις τιμές των διάφορων συμβόλων που έχω ορίσει . Σε αυτό το παράθυρο ο χρήστης μπορεί να «δεί» της τιμές των μεταβλητών ενός προγράμματος σε C που τρέχει.
- Register window: Δείχνει τις τιμές των καταχωρητών .Οι τιμές των καταχωρητών μπορούν να αλλάξουν όταν σταματάει η εκτέλεση του προγράμματος .
- Memory window: Δείχνει τα περιεχόμενα της μνήμης προγράμματος δεδομένων ή εισόδου / εξόδου. Τα περιεχόμενα μπορούν να φανούν είτε σε δεκαεξαδική μορφή είτε σε ASCKII.Τα περιεχόμενα της μνήμης μπορούν να αλλάξουν όταν σταματάει η εκτέλεση του προγράμματος .
- Message window:Δείχνει μηνύματα του AVR Studio προς τον χρήστη.
- Processor window:Δείχνει ζωτικές για τον μικροελεγκτή πληροφορίες όπως : program counter ,stack pointer, status register, ,cycle counter, X ,Y&Z pointer, rampd register and eind register. Όλες αυτές οι πληροφορίες μπορούν να αλλαχθούν όταν έχει σταματήσει η εκτέλεση του προγράμματος.

Την πρώτη φορά που ένα αρχείο object εκτελείται ο χρήστης πρέπει να ρυθμίσει τα παράθυρα που είναι βολικά για την παρατήρηση της εκτέλεσης του προγράμματος ώστε να βλέπει τις πληροφορίες που θέλει για το συγκεκριμένο πρόγραμμα. Την επόμενη φορά που θα ξαναφορτωθεί το ίδιο project οι ρυθμίσεις παραμένουν ως είχαν.

6.3 Τα παράθυρα του AVR Studio

6.3.1 Το παράθυρο του πηγαίου κώδικα

Το παράθυρο του πηγαίου κώδικα είναι το κεντρικό παράθυρο του AVR Studio. Δημιουργείται όταν ανοίγει ένα αρχείο object και είναι παρόν καθ΄ όλη την διάρκεια χρήσης του AVR Studio .Αν ο χρήστης κλείσει το παράθυρο του πηγαίου τότε κλείνει και το αρχείο object .
Το παράθυρο του πηγαίου δείχνει τον κώδικα που εκτελείται αυτή την στιγμή .Η επόμενη προς εκτέλεση εντολή είναι πάντα μαρκαρισμένη από το AVR Studio. Αν το μαρκάρισμα μετακινηθεί από τον χρηστή τότε μπορούμε ακόμα να αναγνωρίσουμε πια είναι η επόμενη προς εκτέλεση εντολή καθώς το προηγουμένως μπαρκαρισμένο κείμενο γίνεται μπλε .
Ένα σημείο διακοπής αναγνωρίζεται στο παράθυρο του πηγαίου κώδικα σαν μια τελεία στα αριστερά της εντολής οπού έχουμε θέση το σημείο διακοπής .
Αν τοποθετήσουμε το δρομέα (cursor) σε ένα σημείο και επιλέξουμε την εντολή ‘‘run to cursor’’ τότε το πρόγραμμα θα εκτελεστεί μέχρι να φτάσει στην εντολή που έχουμε τοποθετήσει το δρομέα. Τα σημεία διακοπής τοποθετούνται με παρόμοιο τρόπο. Τοποθετούμε το δρομέα στη εντολή που θέλουμε να θέσουμε το σημείο διακοπής και επιλέγουμε το toggle break point.Αν υπάρχει ένα σημείο διακοπής τότε αυτό διαγράφεται αλλιώς τοποθετείτε ένα σημείο διακοπής.
Ένα αρχείο object μπορεί να αποτελείτε από πολλά τμήματα. Ένα τμήμα εμφανίζεται κάθε φορά αλλά ο χρήστης μπορεί να αλλάξει το τμήμα που εμφανίζεται διαλέγοντας αυτό που θέλει να δει στο πλαίσιο επιλογής, επάνω αριστερά στο παράθυρο του πηγαίου κώδικα. Αυτό είναι ένα αρκετά χρήσιμο χαρακτηριστικό καθώς μας επιτρέπει να ορίσουμε σημεία διακοπής και σε άλλα τμήματα του κώδικα εκτός αυτού που εκτελείται .
Αν το κουμπί δεξιά από το πλαίσιο επιλογής τμήματος πατηθεί ,τότε το παράθυρο αλλάζει μεταξύ εκτέλεσης πηγαίου κώδικα και κώδικα μηχανής.
Όταν το AVR STUDIO δουλεύει με κώδικα μηχανής . Σε ορισμένες περιπτώσεις όταν δεν είναι διαθέσιμος ο πηγαίος κώδικας , όπως π.χ. όταν φορτώνω ένα αρχείο τύπου Intel-hex .Τότε η εκτέλεση γίνεται σε επίπεδο κώδικα μηχανής .
Το παράθυρο του πηγαίου κώδικα υποστηρίζει τον clipboard τον windows.Ο χρήστης μπορεί να επιλέξει τμήμα (ή ολόκληρο) του περιεχομένου του παραθύρου και να τα αντιγράψει στο clipboard επιλέγοντας ‘copy’ στο μενού ‘edit’.
Οι εντολές Toggle Breakpoint, Run to cursor, και εντολές αντιγραφής κειμένου είναι επίσης διαθέσιμες στο μενού που εμφανίζεται κάνοντας δεξί κλίκ με το ποντίκι. Μόλις πατηθεί το δεξί πλήκτρο του ποντικιού ένα μενού εμφανίζεται.

6.3.2 watch window

Το watch window μπορεί να εμφανίσει τον τύπο και την τιμή συμβόλων όπως π.χ. μια μεταβλητή σε ένα πρόγραμμα γραμμένο σε C . Καθώς το AVR Studio δεν μπορεί να δημιουργήσει πληροφορίες σχετικά με τα σύμβολα , αυτό το παράθυρο έχει νόημα μόνο όταν έχουμε τροφοδοτήσει το AVR Studio με πηγαίο κώδικα σε C .
Το watch window έχει τρία πεδία . Το πρώτο πεδίο περιέχει το όνομα του συμβόλου το οποίο είναι υπό παρακολούθηση. Το επόμενο έχει τον τύπο του συμβόλου και το τρίτο την τιμή . Εξ ορισμού το παράθυρο των watches είναι άδειο

,οπότε όλα τα σύμβολα που θέλει ο χρήστης να παρακολουθεί πρέπει να τοποθετηθούν στο watch window .Μόλις ένα σύμβολο τεθεί υπό παρακολούθηση παραμένει όσες φορές και να ξανά-εκτελέσουμε τον κώδικα. Ακόμα και αν κλείσουμε το παράθυρο με τα watches αυτά δεν διαγράφονται.

Υπάρχουν εντολές για να προσθέτουμε και να αφαιρούμε watches καθώς και για να τα διαγράψουμε όλα . Ένα watch προστίθεται με την εντολή add watch από το μενού watch ή από την μπάρα εργαλείων απασφαλμάτωσης .Επίσης μπορούμε να προσθέσουμε ένα watch αν πατήσουμε το πλήκτρο INS όταν είναι ενεργό παράθυρο το watch window. Οταν εκτελεστεί μια εντολή add watch χρήστης πρέπει να δώσει το όνομα του συμβόλου που θέλει να παρακολουθεί. Μπορεί να δώσει, είτε να μην δώσει πληροφορίες για την εμβέλεια του συμβόλου (SCOPE).

Το AVR Studio πρώτα θα ψάξει για το σύμβολο σαν να περιέχει πληροφορίες εμβέλειας. Αν αποτύχει αυτό το AVR Studio ψάχνει αν υπάρχει αυτό το σύμβολο στην εμβέλεια του κώδικα που εκτελείται αυτήν την στιγμή. Αν το σύμβολο, το πεδίο του τύπου δείχνει '???' και το πεδίο τιμής είναι άδειο. Κάθε φορά που η εκτέλεση του προγράμματος σταματάει το AVR Studio ψάχνει στην τρέχουσα εμβέλεια και προσπαθεί να βρεί το σύμβολο στο οποίο αναφέρεται το όνομα που δώσαμε. Αν το βρει τότε στο παράθυρο με τα watches δείχνει, εκτός του ονόματος, την εμβέλεια του συμβόλου.

Δεν είναι δυνατόν να έχουμε σύμβολα που αλλάζουν εμβέλειες. Αν ένα σύμβολο βρεθεί, δεν πρόκειται να αλλάξει εμβέλεια. Τα watches παραμένουν αν κλείσει το πρόγραμμα. Το αν ένα σύμβολο είναι εντός εμβέλειας ή όχι αποτελεί μέρος της πληροφορίας που προσφέρει το παράθυρο παρακολούθησης. Αν κάποιο σύμβολο είναι εκτός εμβέλειας τότε το πεδίο που φαίνεται η τιμή του αναφέρει 'out of scope'.

Για να διαγράψουμε έναν watch πρέπει πρώτα να κάνουμε κλίκ στο σύμβολο με το αριστερό κουμπί του ποντικιού. Όταν ένα σύμβολο με αυτόν τον τρόπο το AVR Studio δέχεται την επιλογή Delete Watch από το μενού watch. Αν το παράθυρο παρακολούθησης είναι το ενεργό παράθυρο τότε μπορούμε να διαγράψουμε ένα watch πατώντας το πλήκτρο DEL.

Το παράθυρο παρακολούθησης μπορεί να χρησιμοποιηθεί για να παρακολουθήσουμε πίνακες ή δομές της C. Η σύνταξη είναι ίδια με αυτήν της C, (αγκύλες ([,]) για πίνακες και την τελεία (.) για τις δομές). Όταν παρακολουθούμε πίνακες μπορούμε να χρησιμοποιήσουμε μεταβλητές σαν δυναμικούς δείκτες στα στοιχεία του πίνακα. Είναι δυνατόν δηλαδή να παρακολουθούμε το σύμβολο my array[i] αν το i είναι μια ακέραια μεταβλητή και έχουν κοινή εμβέλεια με το my array[].

Δεν γίνεται να υπάρχουν παραπάνω από ένα ενεργά παράθυρα παρακολούθησης ταυτόχρονα. Τα σύμβολα που είναι υπό παρακολούθηση δεν χάνονται αν κλείσουμε το πρόγραμμα. Το παράθυρο παρακολούθησης μπορεί να απενεργοποιηθεί και να ενεργοποιηθεί και πάλι τα σύμβολα υπό επιτήρηση δεν χάνονται.

6.5 Watches

Όταν η εκτέλεση γίνεται σε επίπεδο πηγαίου κώδικα τότε το παράθυρο με τα watches μπορεί να χρησιμοποιηθεί για να παρατηρούμε την τιμή κάποιων μεταβλητών. Όταν χρησιμοποιούμε object αρχεία που δημιουργήθηκαν από τον AVR Assembler τότε δεν υπάρχουν διαθέσιμες οι απαραίτητες πληροφορίες και δεν μπορούμε να χρησιμοποιήσουμε τα watches.

6.5.1 Προσθήκη watches

Για να προσθέσει ο χρήστης ένα νέο watch πρέπει να επιλέξει το add watch από το μενού debug ή να πατήσει το κουμπί add watch από την μπάρα εργαλείων. Αν το παράθυρο με τα watches δεν είναι ενεργό τότε μπορούμε να χρησιμοποιήσουμε το INS

8.5.2 Διαγραφή watches

Ο χρήστης μπορεί να διαγράψει ένα watch μαρκάροντας τον στο watch window και έπειτα δίνοντας μια εντολή delet watch είτε από το μενού debug είτε από την μπάρας εργαλείων .Εάν το παράθυρο των watches είναι ενεργό τότε μπορούμε να χρησιμοποιήσουμε το DEL.

6.5.3 Διαγραφή όλων των watches

Η εντολή Delete all watches είναι διαθέσιμη από το μενού debug. Όταν κληθεί αυτή η εντολή τότε διαγράφονται όλα τα watches.

6.6 Σημεία Διακοπής

Ο χρήστης μπορεί να ορίσει απεριόριστο αριθμό από σημεία διακοπής .Τα σημεία διακοπής δεν χάνονται όταν κλείνουμε το πρόγραμμα εκτός αν χρησιμοποιήσουμε πιο καινούργιο αρχείο object.
Όταν ένα σημείο τεθεί ,φαίνεται στα αριστερά της εντολής σαν μια τελεία .

6.6.1 Toggle Breakpoint

Η εντολή Toggle Breakpoint ενεργοποιεί ή απενεργοποιεί το σημείο διακοπής για την εντολή στην οποία βρισκόμαστε. Η επιλογή αυτή είναι διαθέσιμη μόνο όταν το ενεργό παράθυρο είναι αυτό του κώδικα .

6.6.2 Clear all Breakpoint

Αυτή η εντολή διαγράφει όλα τα σημεία διακοπής .Η επιλογή αυτή είναι διαθέσιμη μονό όταν το ενεργό παράθυρο είναι αυτό του κώδικα.

6.6.3 Show List

Εμφανίζεται ένα παράθυρο στο οποίο ο χρήστης μπορεί να δει που έχει βάλει σημεία διακοπής , να προσθέσει ένα καινούργιο να διαγράψει να απενεργοποιήσει ή να απενεργοποιήσει και να δει τον κώδικα όπου έχει βάλει το σημείο διακοπής.

6.7 Μπάρα Εργαλείων

Το AVR Studio περιέχει 3 μπάρες εργαλείων:

- Γενικά εργαλεία
- Εργαλεία απασφαλμάτωσης (debug)
- Εργαλεία views

6.8 Execution Target

Το AVR Studio μπορεί να δουλέψει με έναν V3 In –Circuit Emulator. Όταν ο χρήστης φορτώνει ένα αρχείο τότε αυτομάτως ο AVR Studio ελέγχει αν υπάρχει ο Emulator και τότε τον θέτεις ως στόχο για εκτέλεση του προγράμματος .
Ο εξομοιωτής πρέπει να είναι συνδεμένος σε κάποια σειριακή θύρα .

6.8.1 Επιλογές εξομοιωτή

Οι επιλογές του εξομοιωτή εμφανίζονται όταν ξεκινάει ένα καινούργιο project. θα εμφανιστεί ένα παράθυρο που έχει διαθέσιμα 4 φύλλα :μνήμη, Ρολόι, Για προχωρημένους ,και LCD Display (το LCD Display δεν είναι διαθέσιμο ακόμη).

6.8.2 Ρυθμίσεις Ρολογιού

Ο χρήστης μπορεί να επιλέξει αν ο εξομοιωτής θα χρονίζεται από το δικό του προγραμματιζόμενο κρύσταλλο ή αν θα χρονίζετε από κάποια εξωτερική πηγή .Αν χρησιμοποιηθεί ο εσωτερικός κρύσταλλος τότε ο χρήστης μπορεί να επιλέξει οποιαδήποτε συχνότητα μεταξύ 400 KHz- 20MHz.Ο χρήστης μπορεί να επιλέξει οποιαδήποτε συχνότητα.από κάποιες προκαθορισμένες ή να γράψει οποιαδήποτε συχνότητα θέλει. Δεν είναι δυνατό να δημιουργηθεί η κάθε συχνότητα. Η πραγματική συχνότητα του ρολογιού είναι αυτή που τυπώνεται στο μήνυμα.

ΑΣΚΗΣΗ 1

1.1 Φωτοβολία όλων των led συνεχώς

Συνδέουμε το PORT B με τα 8 LED που υπάρχουν πάνω στην πλακέτα .Χρησιμοποιούμε το PORT B (Εάν θέλουμε να χρησιμοποιήσουμε κάποιο άλλο PORT π.χ το PORT C πρέπει να το δηλώσουμε στο πρόγραμμα αντί του PORT B) σαν έξοδο και θέλουμε να ανάβουμε όλα τα LED (LED0-LED7) συνεχώς.

Τα θετικά άκρα των LED συνδέονται μέσω αντιστάσεων pull-up (1 ΚΩ) με την τροφοδοσία Vcc.Τα αρνητικά τους άκρα συνδέονται με τους ακροδέκτες της πόρτας η οποία θα δόση 0 Volts για να ανάψουν.

Τα υλικά που θα χρησιμοποιήσουμε είναι:

- Αναπτυξιακό σύστημα
- Καλώδιο επίπεδο10 αγωγών με ακροδέκτες idc10-pin.Τα pin 8 και 9 είναι GND.

ΠΡΟΓΡΑΜΜΑΤΑ

Κώδικας 1.1

```
 # include<io8515.h>  // σύνδεση βιβλιοθήκης

void main(void)
{
DDRB=0xff;              // PORT B έξοδος
while(1)                //  βρόγχος χωρίς τέλος
{
PORTB=0x00;          //  τιμή του PORT B ώστε να ανάβουν όλα τα LED
}}
```

1.1.1 Φωτοβολία των led ένα παρά ένα

Κώδικας 1.1.1

```
#include<io8515.h>  // σύνδεση βιβλιοθήκης

void main(void)
{
DDRB=0xff;              // PORT B έξοδος
while(1)                //  βρόγχος χωρίς τέλος
{
PORTB=0XAA;          //  τιμή του PORT B ώστε να ανάβουν όλα τα LED
}}
```

1.1.2 Φωτοβολία των led 4 δεξιά ή 4 αριστερά

Κώδικας 1.1.2

```
 # include <io8515.h>  // σύνδεση βιβλιοθήκης

void main(void)
{
DDRB=0xff;              // PORT B έξοδος
while(1)                //  βρόγχος χωρίς τέλος
{
PORTB=0X0f;          //  τιμή του PORT B ώστε να ανάβουν όλα τα LED
}}
```

1.2 Πρόσθεση δυο μεταβλητών.

Χρησιμοποιούμε το PORT B σαν έξοδο, δηλαδή φαίνεται το αποτέλεσμα της πρόσθεσης των δύο μεταβλητών a,b. Το αποτέλεσμα της πρόσθεσης είναι σε δυαδική μορφή επίσης λόγω ότι η πόρτα είναι 8 bit το μέγιστο άθροισμα που μπορεί να απεικονιστεί είναι το 255. Τα αναμένα LED εκφράζουν 1 και τα σβηστά τα 0 δυαδικά ψηφία.

Τα υλικά που θα χρησιμοποιήσουμε είναι:

- Αναπτυξιακό σύστημα
- Καλώδιο επίπεδο10 αγωγών με ακροδέκτες idc10-pin.Τα pin 8 και 9 είναι GND

Κώδικας 1.2

```
#include <io8515.h>      // σύνδεση βιβλιοθήκης

unsigned char a=25;      // δήλωση μεταβλητών
unsigned char b=3;       //  δήλωση μεταβλητών
unsigned char sum;       //   δήλωση μεταβλητών

void main(void)
 {
DDRB=0Xff;              // PORT B έξοδος
sum=a+b;                // υλοποίηση της άθροισης

while(1)
{                 // βρόγχος χωρίς τέλος
PORTB=~sum;       // το PORT B παίρνει την τιμή του αθροίσματος sum
}}
```

1.3 Φωτοβολία των led με το αντίστοιχο μπουτόν (οποιοδήποτε μπουτόν πατάω ανάβει το αντίστοιχο led.

Σε αυτό το πρόγραμμα χρησιμοποιούμε ως είσοδο το PORTD και ως έξοδο το PORTB .Του δίνουμε μια τιμή με την βοήθεια των πλήκτρων που έχουμε συνδέσει στην είσοδο και την τιμή αυτή την βγάζει στην έξοδο(led) .

Κώδικας 1.3

```
#include<io8515.h>    // σύνδεση βιβλιοθήκης
unsigned char a;        // δήλωση μεταβλητής

void main(void)
{
DDRB=0XFF;          // PORT B έξοδος
DDRC=0X00;          //  PORT D είσοδος
PORTC=0xFF;       //  ενεργοποίηση pull-ups στο PORT C

while(1)
{
a=PINC;                 // ότι έρχεται στο PORT C καταχωρείται στο a
PORTB=a;                // η τιμή της a καταχωρείται στο PORTB
}}
```

1.4 Φωτοβολία των LED με ένα μόνο πλήκτρο

Εδώ χρησιμοποιούμε δύο πόρτες , η PORT D χρησιμοποιείται σαν είσοδο και την συνδέουμε με τα πλήκτρα της αναπτυξιακής πλακέτας . Η PORT B χρησιμοποιείται σαν έξοδο και την συνδέουμε με τα 8 LED. Στήν αρχή τα LED ανάβουν ένα παρά ένα και παραμένουν έτσι πατώντας οποιοδήποτε πλήκτρο εκτός του SW1.Οταν τώρα πατηθεί το SW1 τότε ανάβουν τα σβηστά LED και σβήνουν αυτά που άναβαν , εάν αφήσουμε το SW1 τότε τα LED επανέρχονται στην αρχική κατάσταση . Τα πλήκτρα όταν πατηθούν μας δίνουν LOW (0 volts).

Τα υλικά που θα χρησιμοποιήσουμε είναι:

- Αναπτυξιακό σύστημα
- Καλώδιο επίπεδο10 αγωγών με ακροδέκτες idc10-pin.Τα pin 8 και 9 είναι GND.

Κώδικας 1.4

```
#include<io8515.h>    // σύνδεση βιβλιοθήκης
unsigned char a;        // δήλωση μεταβλητής

void main(void)
{
DDRB=0XFF;          // PORT B έξοδος
DDRD=0X00;         //  PORT D είσοδος
PORTD=0xFF;       //  ενεργοποίηση pull-ups στο PORT D

while(1)
{
a=PIND;                 // ότι έρχεται στο PORT D καταχωρείται στο a
if(a==0xfe)             // αν πατηθεί το πλήκτρο SW1 θα συμβεί η προηγούμενη σειρά του
{                          κώδικα.
PORTB=~0x55;       // ανάβουν ένα παρά ένα τα LED
}
else
{PORTB=0x55;     // ανάβουν τα σβηστά και σβήνουν αυτά που ήταν ανάμενα
}
}
}
```

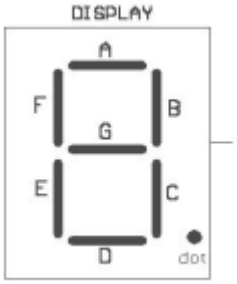
ΑΣΚΗΣΗ 2

2.1 οδήγηση ενός 'seven segment display'

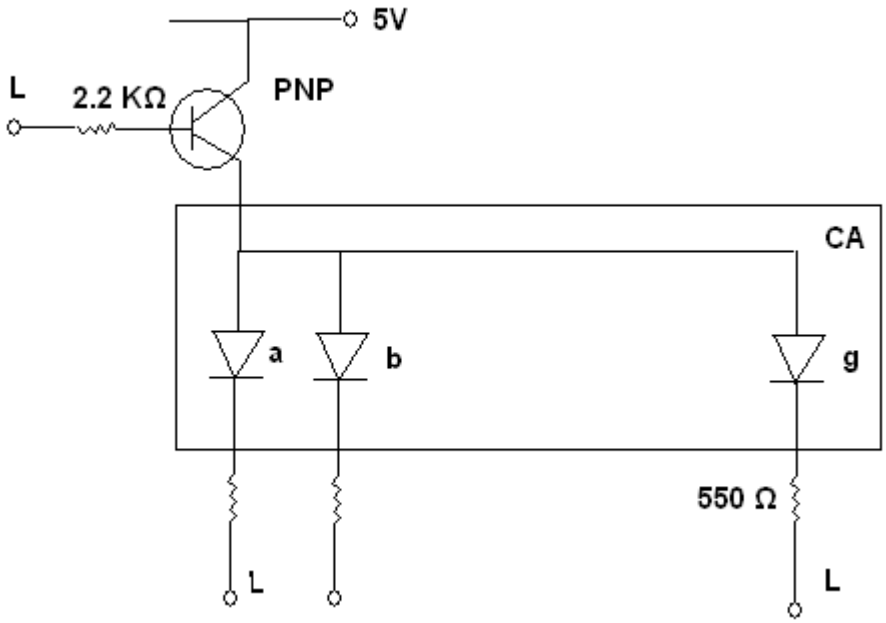
Πατώντας ένα πλήκτρο (ένα από 16 και όση ώρα το πατάμε) μας δείχνει τον αριθμό του πλήκτρου σε ένα display.Εδώ χρησιμοποιούμε δυο πόρτες την D σαν είσοδο και την B σαν έξοδο. Την PORT D την συνδέουμε στα πλήκτρα και την PORT B στο display.

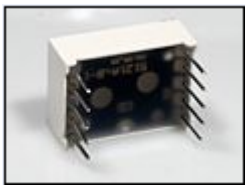
Τα υλικά που θα χρησιμοποιήσουμε είναι:

- Αναπτυξιακό σύστημα
- Καλώδιο για τροφοδοσία του display



Port: 7 6 5 4 3 2 1 0
Sev-seg : h g f e d c b a





Κώδικας 2.1

```
#include<io8515.h>                    //   σύνδεση βιβλιοθήκης
void main(void)
{
DDRB=0xff;                           // PORT B έξοδος
DDRA=0x00;                           // PORT A είσοδος
PORTA=0xff;                          // ενεργοποίηση pull-ups στο PORT A

while(1)                             // βρόγχος χωρίς τέλος
{
switch(PINA)                         // επιλογή ανάλογα με τα PINA
{
case 0xfe:PORTB=~0x3f;break;   //  το display δείχνει 0
case 0xfd:PORTB=~0x06;break; // το display δείχνει 1
```

```
case 0xfb:PORTB=~0x5b;break; // το display δείχνει 2
case 0xf7:PORTB=~0x4f;break; // το display δείχνει 3
case 0xef:PORTB=~0x66;break; // το display δείχνει 4
case 0xdf:PORTB=~0x6d;break; // το display δείχνει 5
case 0xbf:PORTB=~0x7d;break; // το display δείχνει 6
case 0x7f:PORTB=~0x07;break; // το display δείχνει 7

default:PORTB=~0x00;break;    // το display δεν δείχνει τίποτα

}
}
}
```

2.2 Οδήγηση seven segment display-με συγκράτηση της τιμής

Στο πρόγραμμα αυτό πατάμε ένα πλήκτρο από αυτά που βρίσκονται στο board, και το display δείχνει τον αριθμό του πλήκτρου .
Τα πλήκτρα τα έχουμε συνδέσει στο PORTA ενώ το dislay στο PORTB
Στην άσκηση αυτή το κρατάει το νούμερο του πλήκτρου και το αλλάζει μονό αν πατηθεί άλλο πλήκτρο.

Υλικά

- Αναπτυξιακό σύστημα
- Δυο καλώδια με ακροδέκτες idc 10-pin
- Πλακέτα με seven segment display κοινής ανόδου.
- Καλώδιο για τροφοδοσία του display

Κώδικας 2.2

```
#include<io8515.h>
unsigned char b;

void main(void)
{
b=0;
DDRB=0xff;
DDRA=0x00;
PORTA=0xff;

while(1)
{
switch(PINA)
{
case 0xfe:{PORTB=~0x3f;b=PORTB;break;}
case 0xfd:{PORTB=~0x06;b=PORTB;break;}
case 0xfb:{PORTB=~0x5b;b=PORTB;break;}
case 0xf7:{PORTB=~0x4f; b=PORTB;break;}
case 0xef:{PORTB=~0x66;b=PORTB;break;}
case 0xdf:{PORTB=~0x6d;b=PORTB;break;}
case 0xbf:{PORTB=~0x7d;b=PORTB;break; }
```

```
case 0x7f:{PORTB=~0x07;b=PORTB;break; }

default:PORTB=b;break;
}
}
}
```

2.3 Οδήγηση seven segment display με μια καθυστέρηση στο πλήκτρο

Κώδικας 2.3

```
#include<io8515.h>
unsigned char b;
void delay(void)
{
int i,j;
for(i=0;i<1000;i++)
for(j=0;j<1000;j++);
  }
void main(void)
{
b=0;
DDRB=0xff;
DDRD=0x00;
PORTD=0xff;
while (1)
{
switch(PIND)
{
case 0xfe:{delay();PORTB=~0x3f;b=PORTB;break;}
case 0xfd:{delay();PORTB=~0x06;b=PORTB;break;}
case 0xfb:{delay();PORTB=~0x5b;b=PORTB;break;}
case 0xf7:{delay();PORTB=~0x4f;b=PORTB;break;}
case 0xef:{delay();PORTB=~0x66;b=PORTB;break;}
case 0xdf:{delay();PORTB=~0x6d;b=PORTB;break;}
case 0xbf:{delay();PORTB=~0x7d;b=PORTB;break;}
case 0x7f:{delay();PORTB=~0x07;b=PORTB;break;}
default:PORTB=b;break;
  }
  }
  }
```

ΑΣΚΗΣΗ 3

ΣΥΝΔΕΣΗ ΠΛΗΚΤΡΟΛΟΓΙΟΥ 4Χ4 ΣΤΟΝ ΜΙΚΡΟΕΛΕΓΚΤΗ (AT90S8515)

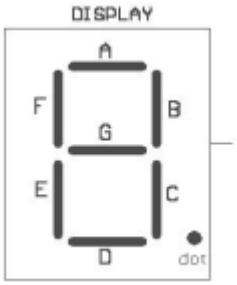
Τα PORT του AT90S8515 είναι διπλής κατεύθυνσης.Μπορούμε δηλαδή να οδηγήσουμε 8 φορτία TTL, ή να διαβάσουμε 8 ψηφιακά σήματα (0 ή 5V) που έρχονται στην πόρτα από εξωτερική συσκευή ή πλήκτρα . Ένας απλός τρόπος ελέγχου ενός πληκτρολογίου με 16 πλήκτρα φαίνεται στο παρακάτω σχήμα όπου μπορούμε να γράψουμε ένα «0» σε κάθε γραμμή RAW, και μετά να διαβάσουμε την πόρτα ψάχνοντας για πλήκτρο που πιθανόν πατήθηκε (που είναι «0»).

3.1 Οδήγηση ενός seven segement display από ένα πληκτρολόγιο 4χ4.Πατώντας ένα πλήκτρο να φαίνεται ο αντίστοιχος αριθμός στο display.

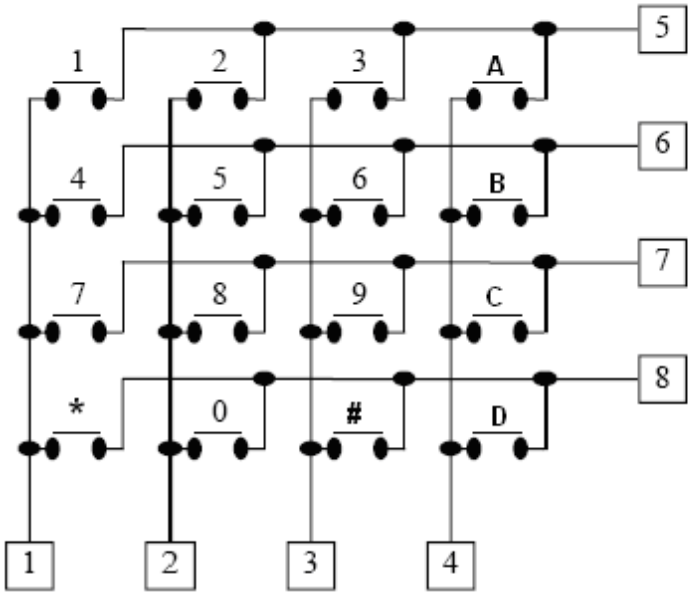
Για την αποκωδικοποίηση του πληκτρολογίου χρησιμοποιούμε την θύρα Α, ενώ στην θύρα C να την συνδέσετε στο display.Το πρόγραμμα να είναι έτσι κατασκευασμένο ώστε το display να κρατάει την τιμή που έχει μέχρι να πατηθεί το επόμενο πλήκτρο. Αρχικά το display να ανάβει (και τα 8 led).

Υλικά

- Αναπτυξιακό σύστημα
- Δυο καλώδια με ακροδέκτες idc 10-pin,Τα pin 8και 9 είναι GND.
- Πλακέτα με seven segment display κοινής ανόδου.
- Καλώδιο για τροφοδοσία του display.
- Το πληκτρολόγιο 4Χ4



Port: 7 6 5 4 3 2 1 0
Sev-seg : h g f e d c b a



Raw1 (A4)
D

Raw2 (A5)
E

Raw3 (A6)
7

Raw4 (A7)
B

Column1 (A0)
D

Column2 (A1)
E

Column3 (A2)
7

Column4 (A3)
B

Πίνακας αποκωδικοποίησης πληκτρολογίου :

Πλήκτρο	Κωδικός	Hex
1	11011101	DD
2	11101101	ED
3	01111101	7D
4	11011110	DE
5	11101110	EE
6	01111110	7E
7	11010111	D7
8	11100111	E7
9	01110111	77
0	11101011	EB
*	11011011	DB
#	01111011	7B
A	10111101	BD
B	10111110	BE
C	10110111	B7
D	10111011	BB

Κώδικας 3.1

```
#include <io8515.h>
unsigned char a,b,c;
void main(void)
{
a=0;
b=0;
c=0;
MCUCR&=0x7f;
DDRC=0xff;
while(1)
{
DDRA=0xff;
PORTA=0x0f;
DDRA=0x00;
a=PINA;
DDRA=0xff;
PORTA=0xf0;
DDRA=0X00;
b=PINA;
c=a|b;
PORTB=~c;
```

```
switch(c)
{
case 0xDD:PORTC=~0x06;break;
case 0xED:PORTC=~0x5b;break;
case 0x7D:PORTC=~0x4f;break;
case 0xBD:PORTC=~0x77;break;
case 0xDE:PORTC=~0x66;break;
case 0xEE:PORTC=~0x6d;break;
case 0x7E:PORTC=~0x7d;break;
case 0xBE:PORTC=~0x7f;break;
case 0xD7:PORTC=~0x07;break;
case 0xE7:PORTC=~0x7f;break;
case 0x77:PORTC=~0x6f;break;
case 0xB7:PORTC=~0x39;break;
case 0xDB:PORTC=~0x80;break;
case 0xEB:PORTC=~0x3f;break;
case 0x7B:PORTC=~0x76;break;
case 0xBB:PORTC=~0x3f;break;

}
}
}
```

ΑΣΚΗΣΗ 4

Χρονικά interrupt του μικροελεγκτή AT90S8515 με TIMERS

Το πρόγραμμα αυτό ανάβει και σβήνει τα led του αναπτυξιακού συστήματος διαδοχικά το ένα μετά το άλλο. Το port B χρησιμοποιείται σαν έξοδος και είναι συνδεμένο στα led.

Υλικά:

- Αναπτυξιακό σύστημα
- Δυο καλώδια με ακροδέκτες idc 10-pin,Τα pin 8και 9 είναι GND.

Κώδικας 4.1

```
#include<io8515.h>
unsigned char i,j;
#pragma vector=TIMER1_OVF1_vect
__interrupt void XRONOS1 (void)                    // interrupt TIMER 1
{
if(j==0)
{
i++;                                                // εάν είναι true αυξανεται το i
if(i==7)j=1;                                // μόλις το i γινει 7 τότε ανάβει και το τελευταίο
led
}                                                    και το j γίνεται 1 έτσι ώστε οταν έρθει interrupt
να
else                                                πραγματοποιηθούν οι εντολές του else
{
i--;
if(i==0)j=0;            //αφού ανάψει και το τελευταίο led τότε το j γίνεται 0
}                       και η συνθήκη της if είναι ξανά true οπότε τα led ανάβουν
}                        ξανά από την αρχή και ου το κάθε εξής
void main(void)
{
j=0;
i=0;
DDRB=0xff;
SREG|=0x80;              // Γενική άδεια ενεργοποίησης των interrupt
TIMSK|=0x80;
TCCR1B=0x02;                  // CK/8
while(1)
{
switch(i)
{
```

```
case 0:PORTB=~0x01;break;
case 1:PORTB=~0x02;break;
case 2:PORTB=~0x04;break;
case 3:PORTB=~0x08;break;
case 4:PORTB=~0x10;break;
case 5:PORTB=~0x20;break;
case 6:PORTB=~0x40;break;
case 7:PORTB=~0x80;break;
}
}
}
```

Σε φάρο λιμανιού οι αναλαμπές είναι σε κανονική λειτουργία ανά 3 δευτερόλεπτα και έκτακτη ανάγκη ανά 0,5 δευτερόλεπτα .

Κώδικας 4.2

```
#include<io8515.h>
unsigned char i,M,N;


#pragma vector=INT0_vect
__interrupt void button1(void)
{
M=1;
TCNT1H=0x85;
TCNT1L=0xEE;
TCCR1B=0x03;
}
#pragma vector=INT1_vect
__interrupt void button2(void)
{
N=1;
TCNT1H=0x48;
TCNT1L=0xE5;
TCCR1B=0x04;
}

#pragma vector=TIMER1_OVF1_vect
__interrupt void xronos1(void)
{
i++;
if(i==2)i=0;
}
void main(void)
{
i=0;
```

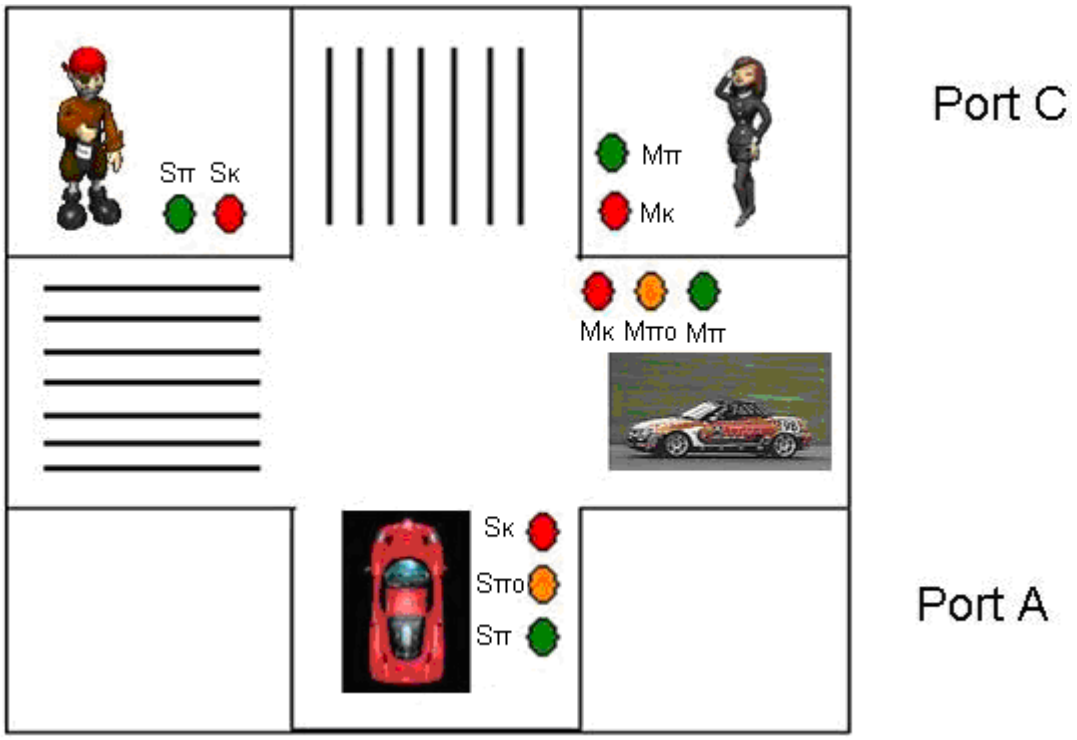
```
DDRB=0xFF;
MCUCR&=0x80;
SREG|=0x80;
GIMSK|=0xC0;
MCUCR|=0x0F;
GIFR|=0xC0;
TIMSK|=0x80;
while(1)
{


if(N==1)
{
if(i==0)
PORTB=0xFE;
else
PORTB=0xFF;
}
if(M==1)
{
if(i==0)
PORTB=0xFE;
else
PORTB=0xFF;

}
```

ΑΣΚΗΣΗ 5

Φανάρια κυκλοφορίας οχημάτων:

Πίνακας λειτουργίας φαναριών:

ΟΧΗΜΑΤΑ(PORTA)									ΠΕΖΟΙ(PORTC)								
	Side				Main									Side		Main	
		Sπ	Sπο	Sκ	Mπ	Mπο	Mκ							Sκ	Sπ	Mκ	Mπ
A7	A6	A5	A4	A3	A2	A1	A0	HEX	HEX	C7	C6	C5	C4	C3	C2	C1	C0
X	X	0	0	1	1	0	0	**0C**	**09**	X	X	X	X	1	0	0	1
X	X	0	0	1	0	1	0	**0A**	**09**	X	X	X	X	1	0	0	1
X	X	1	0	0	0	0	1	**21**	**06**	X	X	X	X	0	1	1	0
X	X	0	1	0	0	0	1	**11**	**06**	X	X	X	X	0	1	1	0
0	0	0	1	0	0	1	0	**12**	**0A**	0	0	0	0	1	0	1	0
0	0	0	0	1	0	0	1	**09**	**05**	0	0	0	0	0	1	0	1



Συνδέουμε την πλακέτα με τα φανάρια .Επιτρέπεται μόνο ευθεία ροή αυτοκινήτων στην διασταύρωση. Η πόρτα Α έξοδος για της διαβάσεις και η πόρτα C για τα φανάρια των αυτοκινήτων.

Κώδικας: 5.1

```
#include<io8515.h>
unsigned long int i;
#pragma vector=INT1_vect
__interrupt void button(void)
{
PORTC=~0x0A;
PORTA=~0x12;
for(i=0;i<200000;i++);
PORTC=~0x05;
PORTA=~0x09;
for(i=0;i<200000;i++);
}
void delay(void)
{



for(i=0;i<200000;i++);
}
```

```
void main(void)
{
DDRC=0xFF;
DDRA=0xFF;
DDRD|=0x08;
PORTD|=0x08;
SREG=0x80;
GIMSK|=0x80;
GIFR|=0x7F;
MCUCR&=0x7F;
MCUCR&=0xFB;
MCUCR|=0x08;
while(1)
{
PORTC=~0x09;
PORTA=~0x0C;
delay();

PORTC=~0x09;
PORTA=~0x0A;
delay();

PORTC=~0X06;
PORTA=~0X21;
delay();

PORTC=~0X06;
PORTA=~0X11;
delay();
delay();
}}
```

Φανάρια με μπουτόν στο INT0,πατώντας το PD2 ανάβουν οι σηματοδότες πράσινο γία τους πεζούς ανεξάρτητα ποιός σηματοδότης ανάβει.

Κωδικας 5.2

```
#include<io8515.h>
long int i,j;
#pragma vector=INT0_vect
__interrupt void pliktro0(void)
{
for (i=0;i<300000;i++){PORTC=~0X0a;PORTA=~0X0c;}

for (i=0;i<600000;i++){PORTC=~0X05;PORTA=~0X09;}
}

void main(void)
```

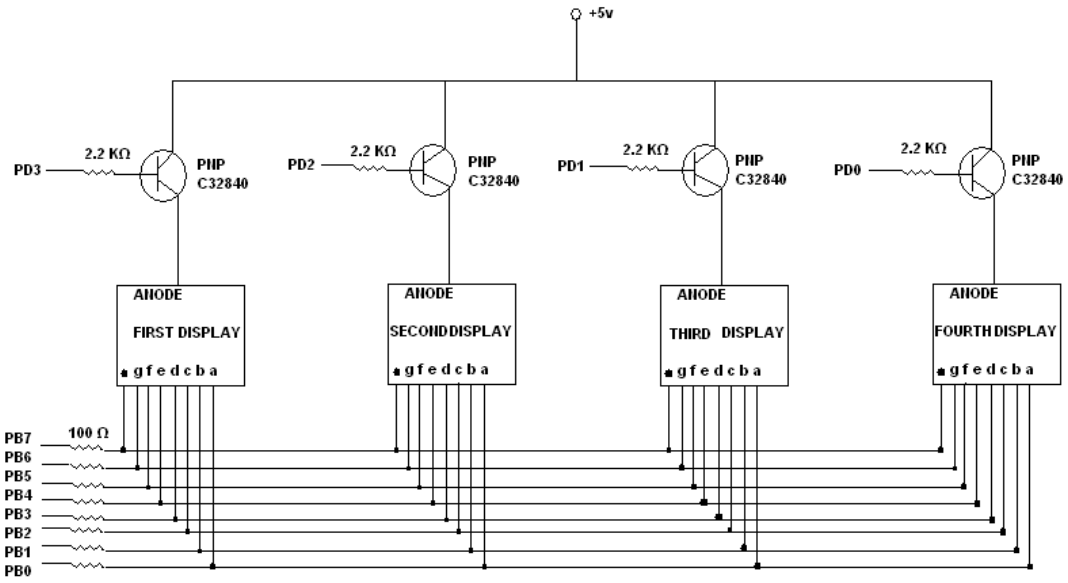
```
{
MCUCR&=0X7f;
DDRA=0XFF;
DDRC=0XFF;
GIFR=0X80;
DDRD=0X00;
PORTD=0XFF;
SREG|=0X80;
GIMSK|=0X40;
while(1)
{
j=0;

if (j==1) goto Fasi1;
else if (j==2) goto Fasi2;
else if (j==3) goto Fasi3;
else if (j==4) goto Fasi4;

Fasi1:
for (i=0;i<500000;i++)
{
PORTC=~0x09;
PORTA=~0x0c;
j=1;
}

Fasi2:
for (i=0;i<500000;i++)
{
PORTC=~0x09;
PORTA=~0x0a;
j=2;
}

Fasi3:
for (i=0;i<500000;i++)
{
PORTC=~0x06;
PORTA=~0x21;
j=3;
}

Fasi4:
for (i=0;i<500000;i++)
{
PORTC=~0x06;
PORTA=~0x11;
j=4;
}}}
```

ΑΣΚΗΣΗ 6

Οδήγηση τετραπλού display με σάρωση:

Χρησιμοποιούμε display αποτελούμενο από 4 seven-segment κοινής ανόδου το οποίο θα μας αποτυπώνει ένα τετραψήφιο αριθμό.

Υλικά:
Χρησιμοποιούμε 4 seven-segment display κοινής ανόδου , 4 τρανζίστορ PNP .Η βάση του κάθε τρανζίστορ είναι συνδεμένη σε ένα από τα 4 πρώτα pins της θύρας D.Ο εκπομπός του κάθε τρανζίστορ έχει συνδεθεί σε τάση +5V ενώ ο συλλέκτης σε κάθε ένα από τα seven-segment και συγκεκριμένα στην άνοδο.
Επίσης χρησιμοποιούμε καλώδια με ακροδέκτες idc 10-pin,Τα pin 8και 9 είναι GND.



Μόνιμο τύπωμα του αριθμού 2087

Κώδικας 6.1

```
#include<io8515.h>
unsigned char i;
unsigned                                                    char
data[]={~0x3F,~0x06,~0x5B,~0x4f,~0x66,~0x6d,~0x7d,~0x07,~0x7f,~0x6f};
unsigned char disp[]={~0x01,~0x02,~0x04,~0x08};
void main(void)
{
DDRD=0xff;
DDRA=0xff;
MCUCR&=0x7f;
while(1)
{
PORTD=disp[0];
```

```
PORTA=data[2];
for(i=0;i<100;i++);

PORTD=disp[1];
PORTA=data[0];
for(i=0;i<100;i++);

PORTD=disp[2];
PORTA=data[8];
for(i=0;i<100;i++);

PORTD=disp[3];
PORTA=data[7];
for(i=0;i<100;i++);
}
}
```

Μόνιμο τύπωμα του αριθμού 1234

Κώδικας 6.2

```
#include<io8515.h>
unsigned char i;
unsigned int t;

unsigned                                                    char
data[]={~0x3F,~0x06,~0x5B,~0x4f,~0x66,~0x6d,~0x7d,~0x07,~0x7f,~0x6f};
unsigned char disp[]={~0x01,~0x02,~0x04,~0x08};
void main(void)
{
MCUCR&=0x7f;
DDRD=0xff;
DDRA=0xff;
while(1)
{
for(i=0;i<10;i++)
{
PORTD=disp[i-1];
PORTA=data[i];
for(t=0;t<500;t++)
if(i==4)i=0;
}
}

}
```

Μετρητής από 0 εως 9999 με το πρώτο μπουτόν κάνει reset.

Κώδικας 6.3

```
#include<io8515.h>
unsigned                                                    char
data[]={~0x3f,~0x06,~0x5b,~0x4f,~0x66,~0x6d,~0x7d,~0x07,~0x7f,~0x6f};
```

```
unsigned char disp[]={~0x01,~0x02,~0x04,~0x08};
long int i,j;
unsigned int t;
#pragma vector=TIMER1_OVF1_vect
__interrupt void Xronos1(void)
{
j++;
if(j==1)
{
i++;
if(i==10000)i=0;
j=0;
}}
void delay (void)
{
for(t=0;t<2000;t++);
}
void main(void)
{
i=0;
DDRA=0x00;
PORTA=0xff;
DDRC=0xff;
DDRB=0xff;
MCUCR&=0x7f;
SREG|=0x80;
TIFR=0x80;
TIMSK|=0x80;
TCCR1B=0x03;
TCNT1H=0x0B;
TCNT1L=0XDC;
while(1)
{
{
if(PINA==0xfe)
i=0;
}
PORTC=disp[3];
PORTB=data[(i%10)];
delay();
PORTC=disp[2];
PORTB=data[(i%100)/10];
delay();
PORTC=disp[1];
PORTB=data[(i%1000)/100];
delay();
PORTC=disp[0];
PORTB=data[(i%10000)/1000];
delay();}
```

ΑΣΚΗΣΗ 7

Παραγωγή παλμών με μεταβλητό duty cycle PWM

Χρησιμοποιούμε τον TIMER/COUNTER1 σαν PWM .Αυτό γίνεται με την βοήθεια ενός καταχωρητή 16-bit,που χρησιμοποιείται για σύγκριση και που ονομάζεται OCR1B.Ο TIMER1 χρησιμοποιείται σε συνδυασμό με τον OCR1B, για την παραγωγή παλμού PWM με χρήση 8,9 και 10 bit (στον OCR1B).Στην άσκηση χρησιμοποιούμε τον OCR1B ,με 8-bit.Σε αυτόν καταχωρούμε την τιμή που καθορίζει το ton, το χρόνο που το pin OCR1B, θα είναι high,(στο pin 29 παίρνουμε το PWM).Έτσι ρυθμίζεται το
Duty cycle που είναι **ton/Ts** (Ts η συνολική περίοδος του παλμού και 1/Ts η συχνότητα του).

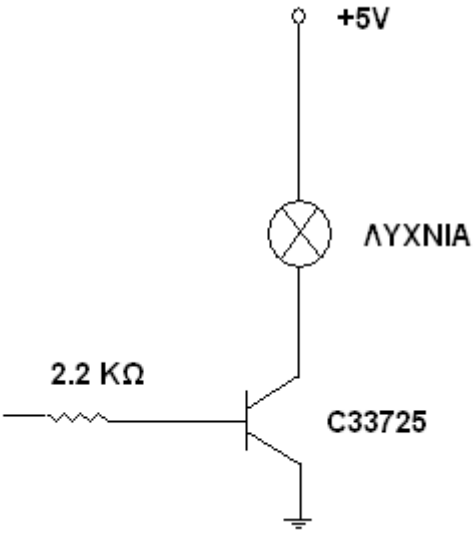
Πίνακας συχνοτητων και περιοδων του PWM

	CK	CK/8	CK/64	CK/256	CK/1024
f pwm	7,843KHz	980,392Hz	122,549Hz	30,637Hz	7,659Hz
Ts	127,5μsec	1020μsec	8160μsec	32,64msec	3056msec

	CK	CK/8	CK/64	CK/256	CK/1024
f pwm	3,914KHz	489,237Hz	61,155Hz	15,289Hz	3,822Hz
Ts	255,433μsec	2043,999μsec	16,352msec	65,407msec	261,643msec

	CK	CK/8	CK/64	CK/256	CK/1024
f pwm	1,955KHz	244,379Hz	30,547Hz	7,637Hz	1,909Hz
Ts	511,509μsec	4092,044μsec	32,736msec	135,74msec	523,834msec

Ο TIMER/COUNTER 1 :
θα μετράει UP από το 00 έως το 127(8 bit) και όταν συναντήσει την τιμή που τοποθετήσαμε στον OCR1BL τότε το pin OC1B θα γίνεται High . Αυξανόμενος μόλις πάρει την τιμή 127 (7f) αρχίζει να μετρά ανάποδα και όταν ελατούμενος συναντήσει ξανά το θα γίνει low 7f



Παραγωγή παλμού PWM συχνότητας 7,843 KHz και duty cycle 50%. Ανάβει η λάμπα ή το BAZER.

Κώδικας7.1

```
# include<io8515.h>
# define value 0x7f
void main(void)
{
TCCR1A|=0x31;
OCR1BH|=0x00;
TCCR1B|=0x01;
OCR1BL=value;
while(1)
{
}
}
```

Αυξομοιώνεται το λαμπάκι ή BAZER μόνο του.

Κώδικας 7.2

```
#include<io8515.h>
```

```
char value=0;
void delay(int d)
{
int i;
for(i=0;i<=d;i++);
}
void main (void)
{
TCCR1A|=0X31;
TCCR1B=0X01;
while(1)
{
value++;
OCR1BL=value;
delay(20000);
}

}
```