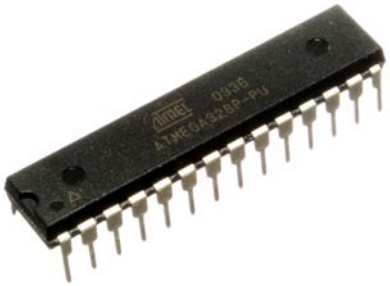
ΤΕΙ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ



Υλοποίηση αναπτύγματος μικροελεγκτή AVR και μελέτη εκπαιδευτικών παραδειγμάτων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Φοιτήτρια : Νικολαΐδου Ευανθία

ΑΕΜ : 2588

Επιβλέπων : Καλόμοιρος Ιωάννης

ΣΕΡΡΕΣ 2016

Πίνακας περιεχομένων

Ευρετήριο Σχημάτων / Εικόνων

Ευρετήριο Πινάκων

Αποποίηση ευθυνών

Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Μηχανικών Πληροφορικής

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον εισηγητή της πτυχιακής μου εργασίας κ. Ιωάννη Καλόμοιρο ο οποίος με εμπιστεύτηκε πως θα φέρω εις πέρας αυτό το δύσκολο έργο και ανέλαβε την επίβλεψη της εργασίας μου.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένεια μου για την στήριξη καθ' όλη τη διάρκεια των σπουδών μου, τους φίλους και τα αγαπημένα μου πρόσωπα που με στήριξαν κατά τη διάρκεια εκπόνησης της εργασίας αυτής.

Περίληψη

Ο σημερινός άνθρωπος ζητά όλο και περισσότερες ολοκληρωμένες ενσωματωμένες εφαρμογές για την χρησιμοποίηση τους στην καθημερινή ζωή. Κύριο συστατικό στην κατασκευή αυτών των εφαρμογών παίζουν οι μικροελεγκτές, οι οποίοι με τις τεράστιες δυνατότητες που προσφέρουν σήμερα, σε συνδυασμό με τον ελάχιστο όγκο που καταλαμβάνουν αλλά και την μικρή κατανάλωση τους, αποτελούν ιδανικά στοιχεία στην κατασκευή ενός ολοκληρωμένου φορητού ενσωματωμένου συστήματος. Στόχος της συγκεκριμένης πτυχιακής εργασίας είναι η γνωριμία με τους μικροελεγκτές AVR της εταιρίας atmel, η υλοποίηση ενός πλήρους αναπτύγματος με τον μικροελεγκτή ATmega328P,για να δοκιμαστούν απλές εφαρμογές I/O,χρονισμού και σημάτων διακοπής και να μελετηθεί η διαδικασία προγραμματισμού του μικροελεγκτή ώστε να παρατεθούν εκπαιδευτικά παραδείγματα που θα αποτελέσουν υλικό αναφοράς και για άλλους σπουδαστές.

Τέλος, μετατρέπεται ο μικροελεγκτής AVR σε Arduino με την διαδικασία εγκατάστασης του bootloader και επιβεβαιώνεται η διαδικασία μεσω προγραμματισμού του μικροελεγκτή(απλή εφαρμογη σε γλώσσα Wiring) από την θύρα USB του υπολογιστή.

1 Εισαγωγή στους μικροελεγκτές

1.1 Ορισμός του « Μικροελεγκτή»

Ο Μικροελεγκτής είναι ένα είδος επεξεργαστή, αποτελεί ουσιαστικά μια παραλλαγή του μικροεπεξεργαστή. Η λειτουργία του μπορεί να πραγματοποιηθεί με ελάχιστα εξωτερικά εξαρτήματα, λόγω των πολλών υποσυστημάτων που διαθέτει.

Οι περισσότεροι μικροελεγκτές βασίζονται στην αρχιτεκτονική Von-Neuman[1], η οποία σαφώς καθόρισε τα τέσσερα βασικά συστατικά που απαιτούνται για ένα ψηφιακό σύστημα. Αυτά περιλαμβάνουν έναν επεξεργαστικό πυρήνα (CPU), τη μνήμη για το πρόγραμμα και τα δεδομένα (RAM), χώρο μόνιμης αποθήκευσης(Flash σε έναν μικροελεγκτή), καθώς επίσης και τις θύρες I/O για επικοινωνία με εξωτερικές περιφερειακές μονάδες.

Τέλος, όπως ένας μικροϋπολογιστής έχει την δυνατότητα να εκτελεί διάφορα προγράμματα και να διαθέτει περιφερειακές συσκευές, επεξεργαστή και μνήμη, έτσι και ο μικροελεγκτής διαθέτει τα χαρακτηριστικά που προαναφέρθηκαν και, μάλιστα, σε ένα μόνο chip. Ενώ η αποθήκευση των προγραμμάτων που εκτελούν οι μικροελεγκτές γίνεται πάντα στη μνήμη προγράμματος.

1.2 Δομή του « Μικροελεγκτή»

Η οργάνωση των μικροελεγκτών είναι παρόμοια με εκείνη των κλασικών υπολογιστικών συστημάτων (main frame, mini). Αποτελούνται, όπως μπορείτε να παρατηρήσετε στο σχήμα 1.1, από τις παρακάτω λειτουργικές μονάδες:

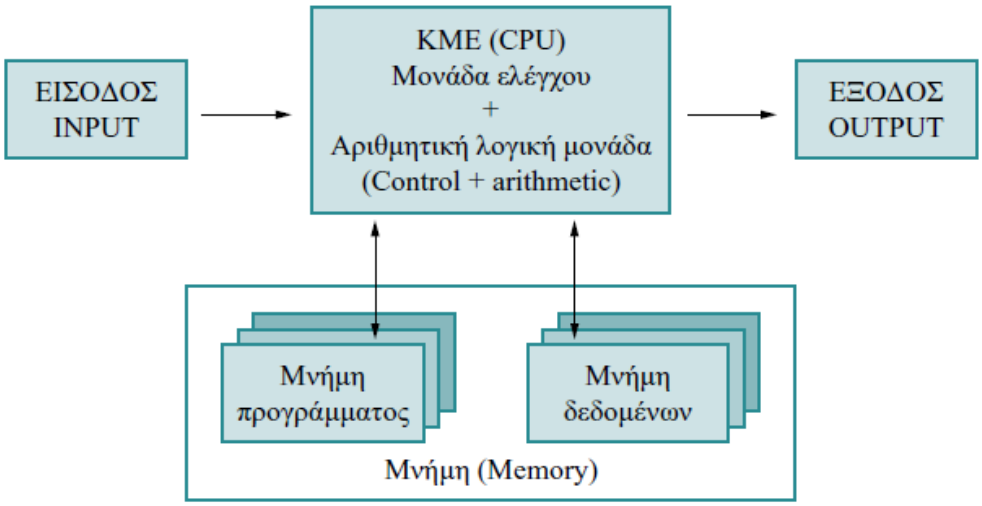
[1] Περισσότερες πληροφορίες στην παράγραφο 2.1.1

Μονάδες εισόδου/εξόδου, με τις οποίες το σύστημα επικοινωνεί με το εξωτερικό του περιβάλλον.

Το *μικροεπεξεργαστή* (ή αλλιώς κεντρική μονάδα επεξεργασίας), η οποίαπεριλαμβάνει την αριθμητική και λογική μονάδα, που επεξεργάζεται ταδεδομένα, τη μονάδα ελέγχου, που είναι υπεύθυνη για τον έλεγχο και το συντονισμό όλων των μονάδων του συστήματος, και τους καταχωρητές, που χρησιμεύουν για προσωρινή αποθήκευση.

Την *κύρια μνήμη,* που χρησιμεύει για την αποθήκευση των εντολών του προγράμματος, των αρχικών δεδομένων και των ενδιάμεσων αποτελεσμάτων.

Το ιδιαίτερο χαρακτηριστικό σε σχέση με άλλα υπολογιστικά συστήματα είναι ότι

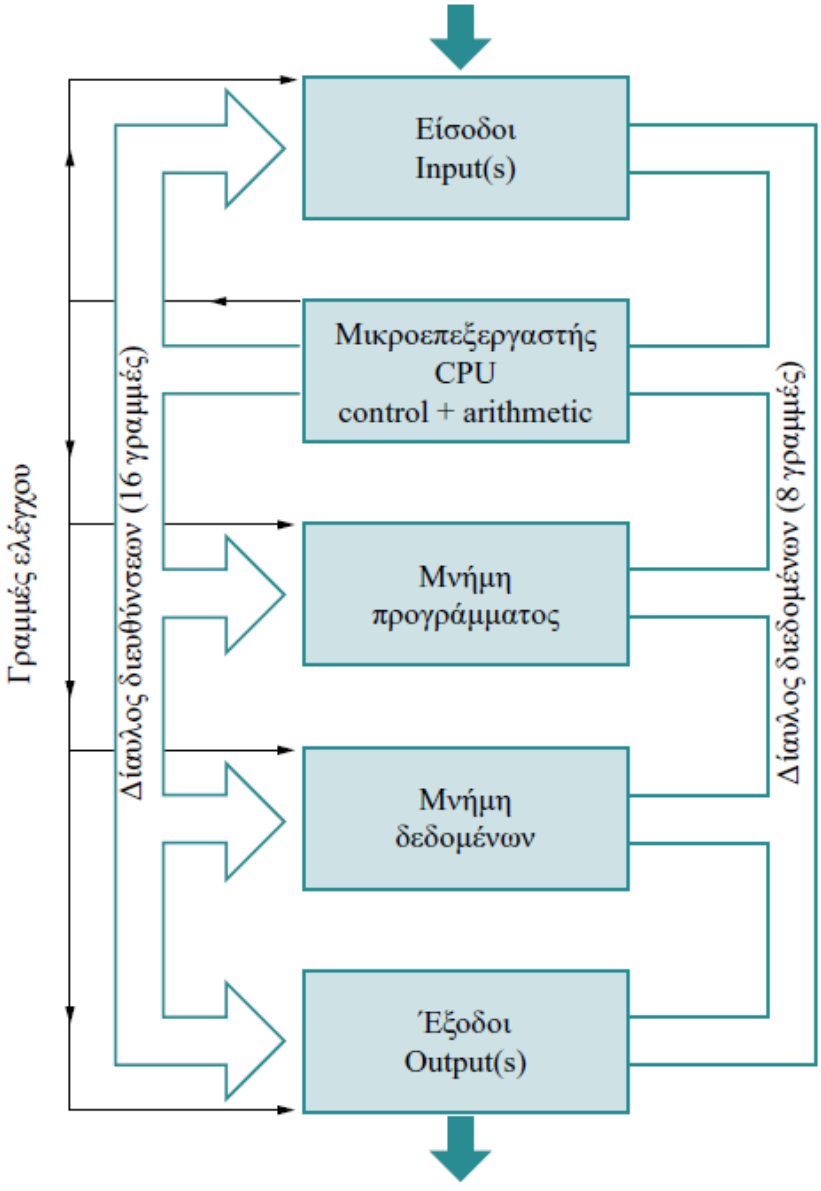


Σχήμα 1 - Ένα τυπικό σχήμα μικροελεγκτή

ολόκληρη η μονάδα επεξεργασίας περιέχεται σε ένα ολοκληρωμένο κύκλωμα, που κατασκευάζεται σε ένα μικρό κομμάτι πυριτίου και αναφέρεται σαν μικροεπεξεργαστής. Η ενσωμάτωση όλων των στοιχείων της κεντρικής μονάδας επεξεργασίας σε ένα μόνο ολοκληρωμένο κύκλωμα, συνδυάζει τα πλεονεκτήματα του μικρού μεγέθους, της υψηλής αξιοπιστίας και του χαμηλού κόστους. Ο μικροεπεξεργαστής συνδέεται κατάλληλα με τα ολοκληρωμένα κυκλώματα της μνήμης και των μονάδων εισόδου/εξόδου, για να αποτελέσει το υπολογιστικό σύστημα που

ονομάζουμε μικροελεγκτή. Στο παρακάτω Σχήμα 1.2, παρουσιάζονται οι δίαυλοι επικοινωνίας του μικροελεγκτή.

Η μεταφορά της δυαδικής πληροφορίας ανάμεσα στις διάφορες μονάδες του μικροελεγκτή γίνεται παράλληλα από ένα σύνολο γραμμών, που αναφέρονται σαν *δίαυλος δεδομένων* (data bus). Οι γραμμές αυτές αναφέρονται σαν *γραμμές δεδομένων* (data lines). Ο δίαυλος δεδομένων δεν λύνει όλα τα προβλήματα μεταφοράς της



Σχήμα 2 - Οι δίαυλοι επικοινωνίας του μικροελεγκτή

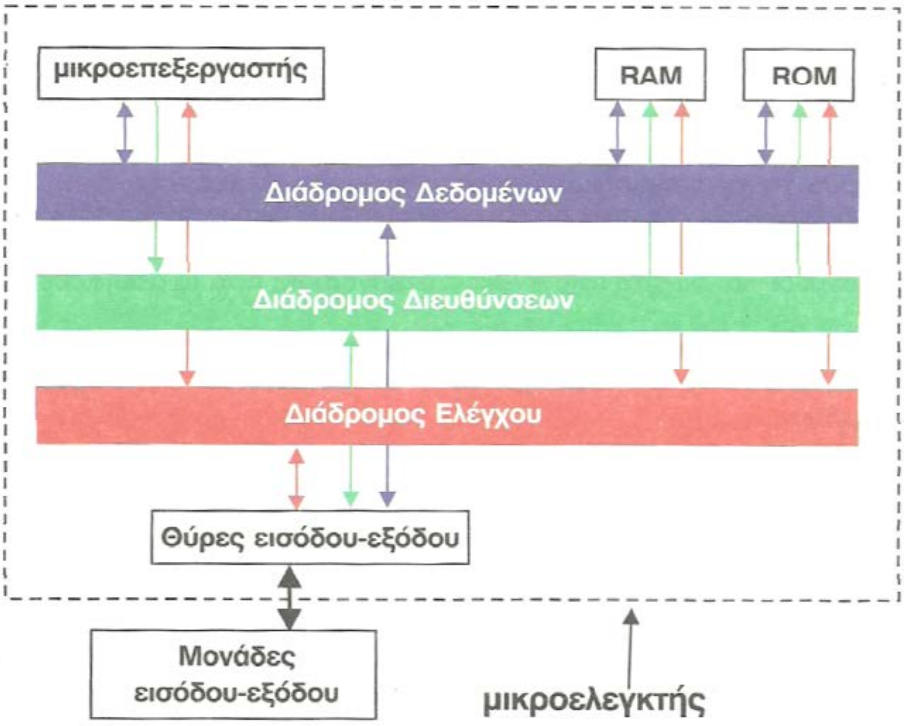
πληροφορίας. Ο μικροεπεξεργαστής θα πρέπει να έχει τη δυνατότητα επιλογής της μονάδας, με την οποία θα επικοινωνήσει, και να μπορεί να την ειδοποιήσει ότι θα στείλει ή θα πάρει δεδομένα από αυτή.

Για το λόγο αυτό διαθέτει δύο ακόμα διαύλους, το *δίαυλο διευθύνσεων* (addressbus) και το *δίαυλο ελέγχου* (controlbus). Οι γραμμές των διαύλων αυτών λέγονται αντίστοιχα *γραμμές διευθύνσεων* (address lines) και *γραμμές ελέγχου* (control lines). Με τις γραμμές διευθύνσεων ο μικροεπεξεργαστής στέλνει τη δυαδική διεύθυνση της θέσης μνήμης ή της μονάδας εισόδου/εξόδου, με την οποία θέλει να επικοινωνήσει, και με τις γραμμές ελέγχου τα κατάλληλα ηλεκτρικά σήματα για την ενεργοποίηση των επιθυμητών λειτουργιών της μνήμης ή των μονάδων εισόδου/εξόδου.

Τέλος, η απαίτηση της ενεργοποίησης στοιχειωδών λειτουργιών σε προ-καθορισμένα χρονικά διαστήματα, δημιουργεί την ανάγκη ύπαρξης μιας βάσης χρόνου, που αναφέρεται ως *κύκλωμα χρονισμού* (clock). Το κύκλωμα χρονισμού αποτελείται συνήθως από ένα κρυσταλλικό ταλαντωτή, που παράγει τετραγωνικούς παλμούς σταθερής συχνότητας. Η συχνότητα αυτή του ταλαντωτή καθορίζει και τη συχνότητα λειτουργίας του μικροεπεξεργαστή.

1.3 Σχέση Μικροελεγκτή και Μικροεπεξεργαστή

Ο μικροεπεξεργαστής είναι ένα ολοκληρωμένο κύκλωμα (IC) που έχει μόνο τη CPU στο εσωτερικό του δηλαδή μόνο τις εξουσίες επεξεργασίας, όπως το PentiumI, II, III, IV της Intel κλπ. Αυτοί οι μικροεπεξεργαστές δεν έχουν μνήμη RAM, ROM και άλλα περιφερειακά στο chip. Για να γίνει λειτουργικός ένας μικροεπεξεργαστής, ο σχε-διαστής του συστήματος πρέπει να τα προσθέσει στο εξωτερικό του. Σε αντίθεση ο μικροελεγκτής περιλαμβάνει CPU (μικροεπεξεργαστής), μνήμη (RAM,ROM), θύρες εισόδου/εξόδου καθώς και θύρες σειριακής και παράλληλης επικοινωνίας, μονάδα αναγνώρισης διακοπών, μετατροπείς αναλογικού σήματος σε ψηφιακό και αντί-στροφα, μετρητές χρόνου – χρονιστές (timers) όλα ενσωματωμένα σε ένα chip (σχήμα 3).



Σχήμα 3– Σχέση μικροελεγκτή - μικροεπεξεργαστή

Οι μικροελεγκτές είναι σχεδιασμένοι για να εκτελούν συγκεκριμένες διεργασίες, είναι το κύριο στοιχείο σε πολλά είδη ηλεκτρονικού εξοπλισμού. Ένα τυπικό σπίτι στο δυτικό κόσμο είναι πιθανό να περιλαμβάνει μόνο έναν ή δύο γενικού σκοπού μικροεπεξεργαστές, αλλά περισσότερους από είκοσι μικροελεγκτές. Μικροελεγκτές απαντώνται σε οποιονδήποτε τύπο ηλεκτρικής συσκευής, πλυντήρια ρούχων, φούρνους μικροκυμάτων, τηλέφωνα κ.λπ. Ενώ οι μικροεπεξεργαστές βρίσκουν εφαρμογές όπου τα καθήκοντα είναι αόριστα όπως η ανάπτυξη λογισμικού, παιχνίδια, ιστοσελίδες, επεξεργασία φωτογραφιών, δημιουργία εγγράφων κ.λπ.

Η ταχύτητα ρολογιού του μικροεπεξεργαστή είναι αρκετά υψηλή σε σύγκριση με του μικροελεγκτή. Ενώ οι μικροελεγκτές λειτουργούν από μερικά MHz έως 30 με 50 MHz, οι σημερινοί μικροεπεξεργαστές λειτουργούν πάνω από 1GHz, δεδομένου ότι εκτελούν πολύπλοκα καθήκοντα. Η σύγκριση του μικροελεγκτή και του μικρο-επεξεργαστή όσον αφορά το κόστος δεν είναι δικαιολογημένη. Αναμφίβολα ένας μικροελεγκτής είναι πολύ φθηνότερος από έναν μικροεπεξεργαστή. Ωστόσο ένας μικροελεγκτής δεν μπορεί να χρησιμοποιηθεί στη θέση ενός μικροεπεξεργαστή και

αντίστοιχα η χρήση ενός μικροεπεξεργαστή δεν συνίσταται στην θέση ενός μικροελεγκτή, καθώς κάνει την εφαρμογή αρκετά δαπανηρή. Ο μικροεπεξεργαστής δεν μπορεί να χρησιμοποιηθεί αυτόνομα, χρειάζεται άλλα περιφερειακά όπως RAM, ROM, buffer, θύρες I/O κ.λ.π. και ως εκ τούτου ένα σύστημα σχεδιασμένο γύρο από έναν επεξεργαστή είναι αρκετά δαπανηρό.

1.4 Πλεονεκτήματα – Μειονεκτήματα Μικροελεγκτή

Το "πακέτο" ενός μικροελεγκτή φέρει κάποια βασικά χαρακτηριστικά που τον καθιστά προτιμότερο για την χρήση του σε εφαρμογές έναντι της χρήσης των επιμέρους στοιχείων που τον απαρτίζουν ξεχωριστά (επεξεργαστής, μνήμες, συσκευές εισόδου- εξόδου, διεπαφές). Αυτά τα χαρακτηριστικά μπορούν να συνοψισθούν σε :

- **Χαμηλό κόστος.** Είναι ένα από τα βασικότερα χαρακτηριστικά που κάποιος σχεδιαστής λαμβάνει υπόψη. Η συνεχής απελευθέρωση στην αγορά μικροελεγκτών από διάφορες εταιρίες βελτίωσαν την ποιότητα αυτών και μείωσαν τις τιμές λόγω ανταγωνισμού.

- **Μικρότερο μέγεθος.** Η ολοκλήρωση των βασικών στοιχείων από τα οποία απαρτίζεται, μείωσε τις διαστάσεις σε σχέση με τη χρήση των επιμέρους στοιχείων ως σύνολο.

- **Χαμηλή κατανάλωση ισχύος.** Το γεγονός ότι οι μικροελεγκτές λειτουργούν σε συγκριτικά χαμηλές συχνότητες που φτάνουν τα 32KHz, οδηγεί στην κατανάλωση μικρών ποσών ισχύος της τάξης των mW ακόμα και μW. Επιπλέον έχουν τη δυνατότητα να εισέρχονται σε κατάσταση αναμονής – sleepmode, καταστέλλουν προσωρινά την λειτουργία της κεντρικής μονάδας επεξεργασίας και των περιφερειακών , οπότε αυτό μπορεί να γίνει μειώνοντας κατά πολύ την κατανάλωση ισχύος του μικροελεγκτή. Έτσι μπορούν να χρησιμοποιηθούν σε εφαρμογές με αυστηρές απαιτήσεις ως προς αυτήν την παράμετρο.

- **Αυτονομία.** Αυτό επιτυγχάνεται μέσω της ενσωμάτωσης σύνθετων περιφερειακών υποσυστημάτων όπως μνήμες και θύρες επικοινωνίας. Έτσι πολλοί μικροελεγκτές δεν χρειάζονται κανένα ολοκληρωμένο κύκλωμα για να λειτουργήσουν.

- **Επίτευξη ελέγχου ή μετρήσεων σε πραγματικό χρόνο.** Ενώ οι ηλεκτρονικοί υπολογιστές πρέπει να τρέχουν λειτουργικά συστήματα πραγματικού χρόνου (όπως RT-Linux, QNX κ.ά.) για να το επιτύχουν, οι μικροελεγκτές δεν απαιτούν επιπλέον λογισμικό.

- **Μειωμένες εκπομπές ηλεκτρομαγνητικών παρεμβολών και μειωμένη ευαισθησία σε αντίστοιχες παρεμβολές από άλλες ηλεκτρικές και ηλεκτρονικές συσκευές.** Το πλεονέκτημα αυτό προκύπτει από το μικρότερο αριθμό και μήκος εξωτερικών διασυνδέσεων καθώς και των χαμηλότερων ταχυτήτων λειτουργίας.

- **Περισσότεροι διαθέσιμοι ακροδέκτες για ψηφιακές εισόδους-εξόδους** (για δεδομένο μέγεθος ολοκληρωμένου κυκλώματος) λόγω της μη δέσμευσής τους για την σύνδεση εξωτερικών περιφερειακών.

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

- Η ενσωμάτωση περιφερειακών σημαίνει **ευκολότερη υλοποίηση εφαρμογών** λόγω των απλούστερων διασυνδέσεων. Επιπλέον έχουμε **μεγαλύτερη αξιοπιστία** λόγω των λιγότερων διασυνδέσεων και **μικρό μέγεθος συνο-λικού υπολογιστικού συστήματος**.

Παρόλο αυτά κάποια από μειονεκτήματα του μικροελεγκτή είναι :

- Η μη αλλαγή του προγράμματος για τον λόγω ότι είναι γραμμένο στην ROM.
- Η δυσκολία του προγραμματισμού του.
- Έχει μεγάλο χρόνο ανάπτυξης. Για να ολοκληρωθεί ένα προϊόν μπορεί να απαιτηθεί από 1 εβδομάδα μέχρι 1 χρόνο.

1.5 Κατασκευαστές Μικροελεγκτών

Οι περισσότερες εταιρείες παράγουν μεγάλη γκάμα μικροελεγκτών. Από πολύ μικρούς και φτηνούς για απλές εφαρμογές έως ιδιαίτερα προηγμένους για πολύ απαιτητικές εφαρμογές. Μερικοί από τους γνωστότερους κατασκευαστές μικρο-ελεγκτών είναι οι :

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

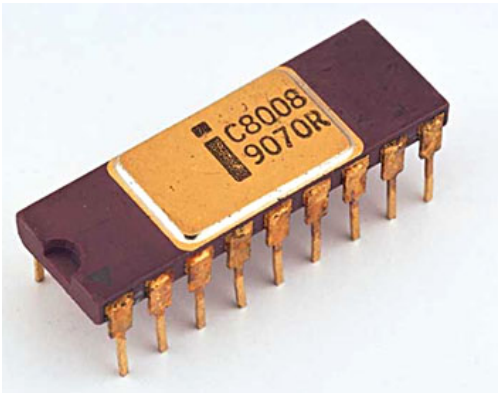
1.6 Ιστορική αναδρομή Μικροελεγκτών

Ο πρώτος μικροεπεξεργαστής αναπτύχθηκε από, (αυτό που ήταν τότε μια μικρή επιχείρηση), την Intel[2] στις αρχές της δεκαετίας του '70.

Εικόνα 1-1 – Ο πρώτος μικροεπεξεργαστής Intel 4004

Το ***1971*** έχουμε την παρουσίαση του πρώτου μικροεπεξεργαστή Intel 4004 από τον Ted Hoff και τον συνεργάτη του Stan Mazor. Ο Intel 4004 ήταν ένας 4bit επεξεργαστής ο οποίος αποτελούνταν από 2.300 τρανζίστορ με συχνότητα ρολογιού 108KHz, εκτελούσε 60.000 πράξεις το δευτερόλεπτο και μπορούσε να δεί 640 bytes μνήμης.

Αρχικά εφαρμόστηκε για τη δημιουργία αριθμομηχανών ενώ τον Νοέμβριο του 1971 η Intel ανακοίνωσε τον πρώτο μικροϋπολογιστή.



Εικόνα 1-2 – Ο πρώτος οκτάμπιτος επεξεργαστής 8008

Μέσα στην επόμενη χρονιά (***1972***) εμφανίζεται ο διάδοχός του ο 8008, ο παγκόσμια πρώτος οκτάμπιτος μικροεπεξεργαστής. Αυτοί οι επεξεργαστές είναι οι πρόδρομοι των πολύ επιτυχημένων Intel 8080, Zilog Z80, και τους ακόλουθους οκτάμπιτους επεξεργαστές της Intel.

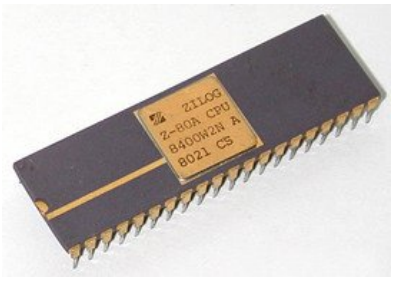
Το ***1974*** έχουμε την εμφάνιση του 8bit μικροεπεξεργαστή Intel 8080 που ήταν αποτέλεσμα εξέλιξης του 8008. Αποτελούνταν από 6.000 τρανζίστορ με συχνότητα λειτουργίας 2MHz.Την ίδια περίοδο η Motorola παρουσίασε τον 6800 που χρησιμοποιήθηκε σε υπολογιστές, σε όλα τα pinball -

[2]ΣυντόμευσηγιατοINTegratedELectronics (Ενσωματωμένη Ηλεκτρονική)

παχνίδια καθώς και σε βιομηχανικές συσκευές ελέγχου. Είχε 4.000 τρανζίστορ, 78 εντολές, σήμα χρονισμού στα 1 ή 2 MHz και 16bit πλάτος διαύλου διευθύνσεων.

Εικόνα 1-3 – Ο MOS 6502

Το ***1975*** η MOS Technology κατασκεύασε τον 6502 (MOS 6502), σαν ανταγωνιστικό επεξεργαστή για τον Intel 8080. Χρησιμοποιήθηκε σε πολύ γνωστά micros όπως τα, Acorn BBC, Apple II, Commodore PET κτλ.



Εικόνα 1-4 – Ο Z80 της Zilog

Το ***1976*** η Zilog φτιάχνει τον Z80 έναν 8bit μικροεπεξεργαστή βασισμένο στον 8080 του οποίου η γλώσσα μηχανής είναι υπερσύνολο αυτής της Intel 8080. Το σήμα χρονισμού του ήταν στα 3.5 MHz με 16 bit πλάτος διευθύνσεων ενώ μπορούσε να δει 64 Kbytes μνήμης. Κατά την δεκαετία του 80 είχε την μεγαλύτερη δημοτικότητα καθώς επικεντρώθηκε στο χαμηλό κόστος σε συνδυασμό με τη μικρή συσκευασία, τις χαμηλές απαιτήσεις και τον συνυπολογισμό των στοιχείων κυκλώματος που κανονικά θα έπρεπε να παρασχεθούν σε ένα χωριστό τσιπ.

Το ***1978*** εμφανίζονται οι πρώτοι 16–bit μικροεπεξεργαστές (δηλαδή ο δίαυλος δεδομένων τους έχει πλάτος 16 bit). Η Intel παρουσιάζει τον 8086/8088, του οποίου η συχνότητα λειτουργίας έχει ανέβει πλέον στα 10MHz και η κατασκευή του απαιτεί 29.000 τρανζίστορ. Αντίστοιχα η Motorola εμφανίζει τον 68000 με συχνότητα λειτουργίας 8 MHz, ο οποίος περιέχει 68.000 τρανζίστορ (από αυτό το γεγονός πήρε και το όνομά του). Ο συνδυασμός της υψηλής ταχύτητας, του μεγάλου χώρου αποθήκευσης (16Mbytes) και του αρκετού χαμηλού κόστους, τον έκανε τον δημοφιλέστερο μικροεπεξεργαστή με αποτέλεσμα να χρησιμοποιηθεί στους υπολογιστές Apple Lisa και Macintosh.

Εικόνα 1-5 – Ο MC68000 της Motorola

Εικόνα 1-6 - Ο 8086 της Intel

Το ***1982*** εμφανίζεται ο Intel 80286, ο οποίος περιέχει 134.000 τρανζίστορ και έχει συχνότητα λειτουργίας 12,5 MHz. Περιλάμβανε δίαυλο δεδομένων 16 bit, δίαυλο διευθύνσεων 24 bit και μπορούσε να δει μεχρι 16Mbytes μνήμης. Είχε δυνατότητα να λειτουργεί στην κατάσταση protectedmode (προστατευμένη κατάσταση λειτουργίας). Αντίστοιχα η Motorola εμφανίζει τον MC68010 με αποτέλεσμα την προσθήκη υποστήριξης της εικονικής μνήμης.

Το ***1985*** εμφανίζονται οι πρώτοι 32–bit μικροεπεξεργαστές. Από τη μια ο Intel 80386, ο οποίος περιέχει 275.000 τρανζίστορ και συχνότητα λειτουργίας 33 MHz και από την άλλη ο Motorola 68020 με 200.000 τρανζίστορ και 16 MHz,έγινε ιδιαίτερα δημοφιλής στην αγορά microcomputer Unix ενώ πολλές μικρές επιχειρήσεις παρή-γαγαν τα συστήματα desktop.

Το ***1989*** εμφανίζεται ο 32–bit μικροεπεξεργαστής Intel 80486, ο οποίος έχει 1.200.000 τρανζίστορ και συχνότητα λειτουργίας 50 MHz.

Το ***1990*** η IBM εισάγει τη τεχνολογία PowerPC επεξεργαστών στη σειρά RS/6000 servers, workstations και supercomputers. Αργότερα, οι PowerPC επεξεργαστές της IBM, θα χρησιμοποιηθούν σε υπολογιστές από την ίδια και την Apple.

Το ***1993*** εμφανίζεται ο Intel Pentium, ο οποίος περιέχει 3.100.000 τρανζίστορ και η συχνότητα λειτουργίας του έχει φτάσει στα 166 MHz.

Ο Pentium ήταν μια ριζική αναθεώρηση σχεδιασμού σε σχέση με την x86 σειρά, αφού εισήγαγε την superscalar επεξεργασία. Χρονιζόταν από τα 60MHz μέχρι τα 300MHz.

Το ***1997*** η Intel ανακοινώνει τον Pentium II. Η συχνότητα λειτουργίας του βρίσκεται στα 300 MHz και το ολοκληρωμένο κύκλωμά του αποτελείται από 7.700.000 τρανζίστορ. Χρονιζόταν από τα 233MHz έως τα 450MHz.

Το ***1999*** η Intel ανακοινώνει τον Pentium III με συχνότητα λειτουργίας 450MHz (σήμερα έχει φτάσει στο 1.13 GHz). Το ολοκληρωμένο κύκλωμα αποτελείται από 9.500.000 τρανσίστορς, ήταν ένας αναβαθμισμένος Pentium II και ο πρώτος που εισήγαγε τις εντολές SSE της Intel. Χρονιζόταν από τα 400MHz μέχρι τα 1.4GHz. Παράλληλα Ο Athlon της AMD, ήταν ο πρωτος επεξεργαστής ο οποίος ξεπέρασε τους Intel επεξεργαστές σε απόδοση. Κατασκευασμένος από 22 εκατομμύρια τρανζίστορ, χρονιζόταν από τα 500MHz μέχρι το 1GHz.

Το ***2000*** η Intel παρουσίασε τον Pentium 4, ένας ακόμα σημαντικός σταθμός στη σχεδίαση των επεξεργαστών, ήταν η εισαγωγή της Netburst αρχιτεκτονικής στους Intel Pentium 4. Κατασκευασμένος από 42 εκατομμύρια τρανζίστορ, χρονιζόταν από τα 1.4GHz μέχρι τα 3.8GHz.

Το ***2001*** η Intel παρουσίασε τον Itanium. Η συνεργασία της Intel και της HP, μας εδωσε τον Itanium ο οποίος ήταν ένας 64bit non-x86 αρχιτεκτονικής επεξεργαστής.

Εικόνα 1-7 – Είδη επεξεργαστών

Σχεδιασμένος για παράλληλη επεξεργασία, στόχευε στη χρήση του από τους enterprise servers. Η σειρά Itanium δεν κατάφερε να έχει μεγάλη επιτυχία.

Το ***2002*** η Intel παρουσίασε τον XScale ARM . Σε συνέχεια της StrongARM σειράς επεξεργαστών, η Intel σχεδίασε τη σειρά XScale ARM, οι οποίοι για πολλά χρόνια χρησιμοποιήθηκαν σε PDAs. Παρ'όλα αυτά η Intel πούλησε αργότερα την XScale στη Marvell το 2006. Παράλληλα η Texas Instruments (TI) έγινε από τις μεγαλύτερες κατασκευάστριες system-on-a-chip επεξεργαστών για χρήση στα smartphones και τα PDAs με τη σειρά Omap. Η σειρά Omap, συνδίαζε ARM επεξεργαστή μαζί με άλλα κυκλώματα όπως GSM επεξεργαστες.

Το ***2003*** η Intel παρουσίασε το Pentium-M που σχεδιάστηκε αποκλειστικά για χρήση σε laptop υπολογιστές. Αποτελούνταν από 77 εκατομμύρια τρανζίστορ και χρονιζόταν από τα 900MHz.Ενώ η AMD με τον Opteron, παρουσίασε τον πρώτο 64bit x86 αρχιτεκτονικής επεξεργαστή, ο οποίος σημείωσε επιτυχία στους servers και workstations. Ηταν κατασκευασμένος από 105 εκατομμύρια τρανζίστορ.

Το ***2005*** η Intel παρουσίασε τον Pentium D - Τη χρονιά αυτή η Intel παρουσίασε τον πρώτο dual-core επεξεργαστή, ξεκινώντας με την Pentium Extreme έκδοση.

Εικόνα 1-8 – Είδη επεξεργαστών

Το ***2006*** η AMD αγοράζει την ATI (σχεδιαστή και κατασκευαστή chipsets και καρτών γραφικών) και ανακοινώνει φιλόδοξα σχέδια για συνδυασμό x86 επεξεργαστών με ATI graphics επεξεργαστές. Παράλληλα η Intel σχεδιάζει τον Xeon 5300 . Οι πρώτοι τετραπλού πυρήνα επεξεργαστές ήταν της σειράς Xeon 5300 για servers και worksations. Στη πραγματικότητα το chip περιείχε δύο διπύρηνους πυρήνες συνδεμένους μεταξύ τους και αποτελούνταν από 582 εκατομμύρια τρανζίστορ.

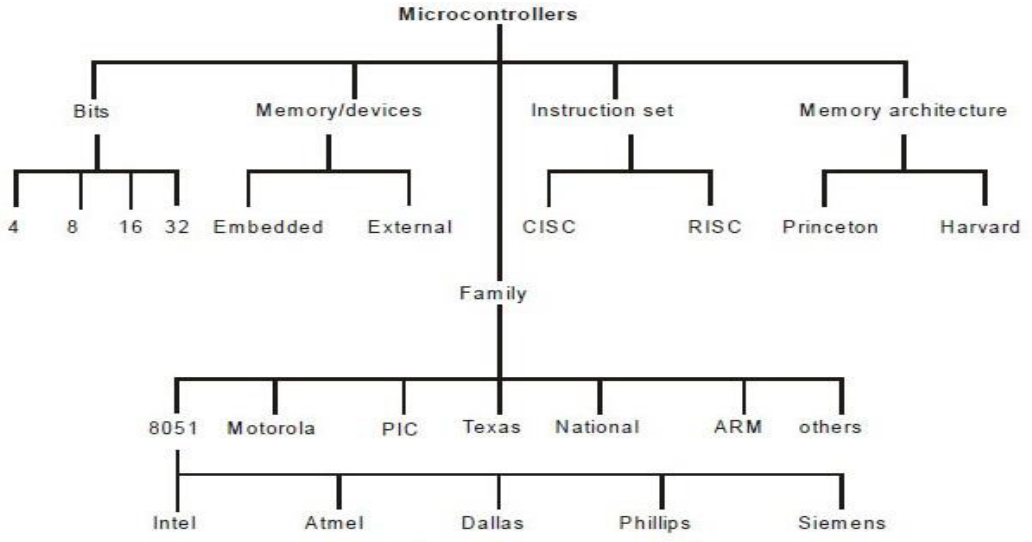
Το ***2008*** η εταιρεία ασύρματης τεχνολογίας Qualcomm, ξεκίνησε τη παραγωγή υψηλής απόδοσης επεξεργαστών για smart-phones, βασισμένους στην αρχιτεκτονική ARM. Ο SnapDragon επεξεργαστής είναι χρονισμένος στο 1GHz και είναι κατα-σκευασμένος από 200 εκατομμύρια τρανζί-στορ.

Εικόνα 1-9 – Ο Intel Xeon 5300

Τέλος από το ***2011*** και μετά η Intel παρουσιάζει τους Core i3, i5, i7. Οι τελευταίοι επεξεργαστές της Intel, είναι σχεδιασμένοι με την αρχιτεκτονική Sandy Bridge. Οι επεξεργαστές της desktop σειράς, έχουν μέχρι 8 πυρήνες σε ένα chip και μέχρι 1.45 δισεκατομμύρια τρανζίστορ!!! Ενώ η ARM ανακοινώνει την ARMv8 64-bit αρχιτεκτονική. Η ARM παρουσίασε τις προδιαγραφές για τα μελλοντικα 64bit chips. Με την αρχιτεκτονικη ARMv8, θα μπορούσαμε να έχουμε επεξεργαστές με έως και 128 πυρήνες!

1.7 Διαδεδομένες κατηγορίες Μικροελεγκτων

Οι μικροελεγκτές ταξινομούνται αναλόγως, το εύρος διαύλου, το σετ εντολών, την αρχιτεκτονική μνήμης, την γλώσσα προγραμματισμού που χρησιμοποιείται κ.α.



Σχήμα 4 - Κατηγορίες Μικροελεγκτών

Έτσι διακρίνονται οι εξής κυρίως κατηγορίες :

Μικροελεγκτές(καμία φορά 4-bit αλλά συνήθως 8-bit) πολύ χαμηλού κόστους, γενικής χρήσης , με πολύ μικρό αριθμό ακροδεκτών(ακόμη και λιγότερους από 8).

Εικόνα 1-11– Ο PIC της Microchip

Εικόνα 1-10 – Ο Intel 8051

Για να μη μπορεί να αντιγραφεί εύκολα το εσωτερικό λογισμικό τους, σχεδιάζονται με έμφαση στη χαμηλή κατανάλωση ισχύος και την αυτάρκεια. Απουσιάζει η δυνατότητα επέκτασης της μνήμης τους. Μερικά μοντέλα είναι ευρέως γνωστά στους ερασιτέχνες ηλεκτρονικούς όπως για παράδειγμα οι περισσότεροι μικροελεγκτές των σειρών PIC (Microchip), AVR (Atmel) και 8051 (Intel, Atmel,Dallas κα).

Μικροελεγκτές (συνήθως 8-bit αλλά και 16 ή 32-bit) χαμηλού κόστους, γενικής χρήσης , με μέτριο έως σχετικά μεγάλο αριθμό ακροδεκτών. Διαθέτουν μεγάλο αριθμό περιφερειακών , όπως θύρες UART, SPI ή CAN, μετατροπείς αναλογικού σε ψηφιακό και ψηφιακού σε αναλογικό. Στους κατασκευαστές της Άπω Ανατολής (Ιαπωνία, Κορέα), συνηθίζεται η ενσωμάτωση ελεγκτών οθόνης υγρών κρυστάλλων και πληκτρολογίου. Μερικές φορές παρέχουν δυνατότητα εξωτερικής επέκτασης της μνήμης τους. Παραδείγματα μικροελεγκτών 16-bit είναι το 8051XA, PIC2X, ο Intel 8096, η σειρά MC68HC12 της Motorolla.

Εικόνα1-12 – Ο Intel 8096

Μικροελεγκτές (κυρίως 32-bit) μέσου κόστους, γενικής χρήσης , με μεγάλο αριθμό ακροδεκτών. Χαρακτηρίζονται από έμφαση στην ταχύτητα εκτέλεσης εντολών, υψηλή αυτάρκεια περιφερειακών και μεγάλες δυνατότητες εσωτερικής ή εξωτερικής μνήμης προγράμματος (FLASH) και RAM. Στο χώρο έχουν ισχυρή παρουσία οι αρχιτεκτονικές με υψηλή μεταφερσιμότητα λογισμικού (portability) από τον έναν στον άλλον κατασκευαστή. Π.χ. μεταξύ των μικροελεγκτών τύπου ARM ή MIPS, το σύνολο των βασικών εντολών που αναγνωρίζει η ALU είναι ακριβώς το ίδιο, μειώνοντας έτσι τις μεγάλες αλλαγές στο λογισμικό, όταν στο μέλλον ο πελάτης υιοθετήσει ένα μικροελεγκτή άλλου κατασκευαστή (αρκεί, φυσικά, να υποστηρίζει και αυτός το σύνολο εντολών ARM ή MIPS,αντίστοιχα).

Μικροελεγκτές εξειδικευμένων εφαρμογών, οι οποίοι ενσωματώνουν συνήθως κάποιο εξειδικευμένο πρωτόκολλο επικοινωνίας το οποίο υλοποιείται πάντοτε σε hardware. Τέτοιοι μικροελεγκτές χρησιμοποιούνται σε τηλεπικοινωνιακές συσκευές όπως τα μόντεμ.

Η μεγάλη μερίδα πωλήσεων των μικροελεγκτών εξακολουθεί να αφορά αυτούς των 8- bit, καθώς είναι η κατηγορία με το χαμηλότερο κόστος και το μικρότερο μέγεθος λογισμικού για το ίδιο αποτέλεσμα, ιδίως επειδή οι σύγχρονες οικογένειες μικροελεγκτών 8- bit έχουν πολύ βελτιωμένες επιδόσεις σε σχέση με το παρελθόν.

1.8 Μικροελεγκτές Εσωτερικής και Εξωτερικής Μνήμης

Όταν ένα ενσωματωμένο σύστημα έχει μία μονάδα μικροελεγκτή που έχει όλα τα λειτουργικά τμήματα (συμπεριλαμβανομένου της μνήμης για την αποθήκευση των δεδομένων αλλά και του προγράμματος) διαθέσιμα σε ένα τσιπ, ονομάζεται ενσωματωμένος μικροελεγκτής. Για παράδειγμα, ο 8051 που έχει την μνήμη των δεδομένων και του προγράμματος, εισόδους και εξόδους, σειριακή επικοινωνία, μετρητές και timers και την ICL (Interrupt Control Logic) όλα σε ένα τσιπ αποτελεί παράδειγμα ενσωματωμένου μικροελεγκτή.

Όταν ένα ενσωματωμένο σύστημα έχει μία μονάδα μικροελεγκτή που δεν έχει όλα τα διαθέσιμα λειτουργικά τμήματα σε ένα chip, ονομάζεται μικροελεγκτής με εξωτερική μνήμη. Στους μικροελεγκτές με εξωτερική μνήμη, το σύνολο ή μέρος των μονάδων μνήμης είναι διασυνδεδεμένα εξωτερικά χρησιμοποιώντας το λεγόμενο «glue Circuit». Για παράδειγμα ο 8031 που δεν έχει ενσωματωμένη μνήμη για την αποθήκευση του προγράμματος, είναι μικροελεγκτής εξωτερικής μνήμης.

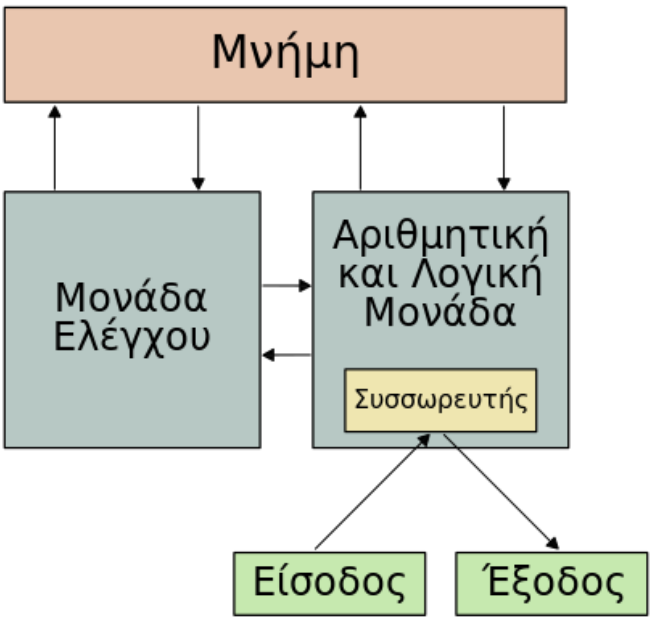
2 Εισαγωγή στους AVR

2.1 Τα Χαρακτηριστικά της Αρχιτεκτονικής ενός Μικροελεγκτή

Υπάρχουν κυρίως δύο κατηγορίες επεξεργαστών, με αρχιτεκτονική Von-Newman (ή Princeton) και αρχιτεκτονική Harvard. Αυτές οι δύο αρχιτεκτονικές διαφέρουν στον τρόπο που γίνεται η αποθήκευση και η προσπέλαση των δεδομένων και του προγράμματος.

2.1.1 Η Αρχιτεκτονική Von-Neumann

Χρησιμοποιήθηκε στον EDSAC, τον πρώτο υπολογιστή με αποθηκευμένο πρό-γραμμα, και εξακολουθεί να είναι η βάση όλων σχεδόν των ψηφιακών υπολο-γιστών, ακόμα και σήμερα. Η μηχανή του Von Neumann είχε πέντε βασικά τμήματα: την μνήμη, την αριθμητική λογική μονάδα, την μονάδα ελέγχου, και τον εξοπλισμό εισόδου και εξόδου. Η μνήμη αποτελούνταν από 4096 λέξεις, και η κάθε λέξη είχε 40-bit με τιμή 0 ή 1. Η κάθε λέξη περιείχε είτε δύο εντολές των 20-bit είτε ένα προσημασμένο ακέραιο των 40-bit. Οι εντολές χρησιμοποιούσαν 8-bit για τον προσδιο-ρισμό του τύπου της εντολής και 12-bit για τον προσδιορισμό μίας από τις 4096 λέξεις της μνήμης.

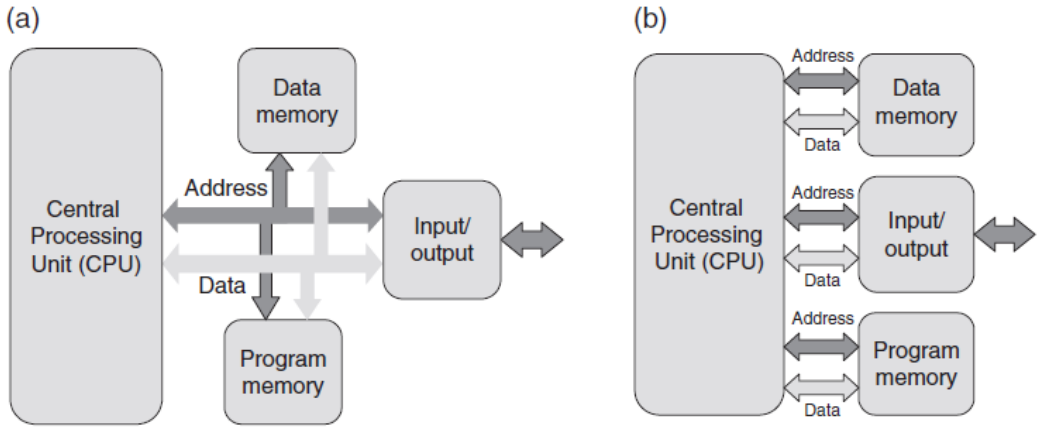


Σχήμα 5 - Η Αρχική μηχανή του Von Neumann

Στην αρχιτεκτονική Von Neumann η μνήμη έχει έναν ενιαίο χώρο φυσικών διευθύνσεων. Δηλαδή τα δεδομένα και οι εντολές των εκτελούμενων προγραμμάτων αποθηκεύονται σε μία και μοναδική μνήμη. Η σειριακή αυτή προσπέλαση των δεδομένων και των εντολών δημιουργεί συμφόρηση και καθιστά τον ελεγκτή πιο αργό. Για παράδειγμα, ένας μικροελεγκτής με δίαυλο διευθύνσεων 16 bit μπορεί να «δει» μέχρι 65536 θέσεις μνήμης με ίδιο μήκος λέξης. Κάποια τμήματα αυτού του χώρου διευθύνσεων μπορούν να χρησιμοποιηθούν από μνήμη RAM, ROM ή και περιφερειακά.

2.1.2 Η Αρχιτεκτονική Harvard

Βασικό χαρακτηριστικό της αρχιτεκτονικής πολλών μικροελεγκτών, που δεν απαντάται στους συνηθισμένους μικροεπεξεργαστές, είναι ότι οι εντολές βρίσκονται σε μία κρυφή μνήμη και τα δεδομένα σε άλλη, δηλαδή έχουν διαφορετικό **διάδρομο για τις εντολές** (*instructions bus*) και διαφορετικό **διάδρομο δεδομένων** (*data bus*), για όλα τα δεδομένα και αποτελέσματα της επεξεργασίας. Η αρχιτεκτονική αυτή αναφέρεται και ως αρχιτεκτονική Harvard. Αυτή η ονομασία παραπέμπει πίσω στον



Σχήμα 6 - Παρουσίαση των δυο τύπων αρχιτεκτονικής

α) Ενιαία μνήμη προγράμματος και δεδομένων (αρχιτεκτονική Von-Neumann)

β) Ξεχωριστή μνήμη προγράμματος και δεδομένων (αρχιτεκτονική Harvard).

υπολογιστή Mark III του Howard Aiken, που είχε διαφο-ρετικές μνήμες για τις εντολές και για τα δεδομένα.

Στο **σχήμα 6**, παρατηρούμε τον ξεχωριστό δίαυλο για τα δεδομένα και για τις εντολές, πράγμα που επιτρέπει την παράλληλη προσπέλαση. Το κίνητρο που ωθεί τους σχεδιαστές σε αυτή την κατεύθυνση είναι η διαδεδομένη χρήση των μικροεπεξεργαστών με διοχέτευση (pipelined CPU). Η μονάδα προσκόμισης εντολών χρειάζεται να προσπελάσει τις εντολές την ίδια ώρα που η μονάδα προσκόμισης τελεστέων χρειάζεται να προσπελάσει τα δεδομένα. Μία διαμερισμένη κρυφή μνήμη, επιτρέπει τις παράλληλες προσπελάσεις, ενώ μία ενοποιημένη δεν τις επιτρέπει. Επίσης αφού οι εντολές κανονικά δεν τροπο-ποιούνται κατά την εκτέλεση, το περιεχόμενο της κρυφής μνήμης εντολών δεν χρειάζεται ποτέ να ξαναγράφεται πίσω στην κύρια μνήμη. Τέλος, να αναφέρουμε πως το ολοκληρωμένο Atmega328P που χρησιμοποιείται στην πλατφόρμα του Arduino βασίζεται στην αρχιτεκτονική Harvard.

2.1.3 Αρχιτεκτονική CISC και RISC

Η καρδιά ενός AVR μικροελεγκτή είναι ένας επεξεργαστής που ανήκει στην κατηγορία RISC (Reduced Instruction Set Computer). Σε αντίθεση με τους επεξεργαστές της κατηγορίας CISC, όπως αναφέρεται παρακάτω, οι επεξεργαστές RISC διαθέτουν ένα μικρό και εύχρηστο σύνολο ομοιόμορφων εντολών που όλες μπορούν να εφαρμοστούν σε ένα μεγάλο πλήθος καταχωρητών . Ο χρόνος εκτέλεσης κάθε εντολής είναι συνήθως ένας κύκλος ρολογιού. Επίσης, το μήκος όλων των RISC εντολών είναι σταθερό πχ, 2 bytes. Έτσι αν για παράδειγμα σε ένα CISC επεξεργαστή μια λειτουργία μπορεί να εκτελεστεί από μία πολύπλοκη εντολή τεσσάρων bytes σε 4 κύκλους ρολογιού, η ίδια λειτουργία σε RISC επεξεργαστή μπορεί να εκτελεστεί από δύο απλούστερες εντολές των δύο bytes, που κάθε μία εκτελείται σε έναν ή δύο κύκλους ρολογιού. Υποστηρίζεται ότι τα RISC προγράμματα είναι συνήθως μικρότερα σε μέγεθος και γρηγορότερα εξαιτίας του ότι απαιτούν λιγότερες χρονοβόρες εντολές μετακίνησης και κάθε εντολή εκτελείται σε έναν συνήθως κύκλο ρολογιού. Οι λιγότερες μετακινήσεις οφείλονται στο ότι ο αριθμός των καταχωρητών στους

οποίους αποθηκεύονται τα ενδιάμεσα απο-τελέσματα πράξεων είναι πολύ μεγαλύτερος από εκείνους των CISC επεξεργαστών.

2.1.3.1 CISC

Η CISC αρχιτεκτονική περιέχει ένα μεγάλο σύνολο οδηγιών υπολογιστή που κυμαίνονται από πολύ απλές έως πολύ σύνθετες και εξειδικευμένες. Αν και το σχέδιο είχε ως στόχο να υπολογίσουμε πολύπλοκες οδηγίες με τον πιο απο-τελεσματικό τρόπο, βρέθηκε αργότερα ότι με πολλές μικρές, σύντομες οδηγίες μπορούσαμε να υπολογίσουμε πολύπλοκες οδηγίες πιο αποτελεσματικά. Αυτό οδήγησε σε ένα σχέδιο που ονομάζεται ReducedInstruction Set Computing (RISC), η οποία είναι πλέον το άλλο είδος της αρχιτεκτονικής μικροεπεξεργαστή.[3]

2.1.3.2 RISC

Είναι αναμφισβήτητα η ταχύτερη και πιο αποτελεσματική διαθέσιμη τεχνολογία μικρο-επεξεργαστή σήμερα. Η αρχιτεκτονική RISC είναι μια βελτίωση από την CISC (Complex Instruction Set Computing) αρχιτεκτονική που χρησιμοποιήθηκε στο αρχικό τσιπ της Intel Pentium. Το 1974, ο John Cocke της IBM Research όταν εργαζόταν για να κάνει μια ταχύτερη έκδοση του τσιπ CISC, έκανε ένα σχέδιο που μείωσε σημαντικά τον αριθμό των οδηγιών που χρειάζονται για την εκτέλεση υπολογισμών. Ο νέος σχεδιασμός δεν ήταν μόνο πιο γρήγορος από την αρχιτεκτονική CISC, αλλά και τα κυκλώματα ήταν επίσης μικρότερα και λιγότερο δαπανηρά στην κατασκευή.[4]

2.2 Οι μικροελεγκτές AVR

Οι μικροελεγκτές AVR χρησιμοποιούν τροποποιημένη Αρχιτεκτονική Χάρβαρντ 8-bit RISC (Reduced Instruction Set Computers) και αναπτύχθηκαν από την Atmel για πρώτη φορά το 1996. Η AVR ήταν μια από τις οικογένειες μικροελεγκτών που έκαναν χρήση της εσωτερικής μνήμης flash για την αποθήκευση του προγράμματος, σε αντίθεση με τις OTPROM (One Time Programmable Read Only Memory), EPROM (Erasable Programmable Read Only Memory) και EEPROM (Electrically Erasable Programmable Read Only Memory) που χρησιμοποιούνται από άλλους μικρο-

[3] http://techterms.com/definition/cisc
[4] http://techterms.com/definition/risc

ελεγκτές. Η βασική αρχιτεκτονική των AVR επινοήθηκε από δύο φοιτητές στο Νορβηγικό Ινστιτούτο Τεχνολογίας, τους Alf-Bogen EGIL και Vegard Wollan. Η θυγατρική της Atmel στη Νορβηγία ιδρύθηκε από τους δύο φοιτητές. Το όνομα AVR , σύμφωνα με την Atmel ,δεν αποτελεί κάτι ιδιαίτερο όσον αφορά την ερμηνεία του, ωστόσο είναι πιθανό να προέρχεται από το Advanced Virtual RISC ή από τα ονόματα των σχεδιαστών, δηλαδή Alfand Vegard RISC.

Υπάρχουν διάφορα είδη AVR μικροελεγκτών με διαφορετικές ιδιότητες ο καθένας. Εκτός από τον AVR32, ο οποίος είναι ένας 32-bit μικροεπεξεργαστής, όλοι οι άλλοι είναι 8-bit, πράγμα που σημαίνει ότι η CPU μπορεί να δουλέψει μόνο 8 bits δεδομένων την φορά. Δεδομένα μεγαλύτερα από 8-bit θα σπάσουν σε κομμάτια των 8-bit για να υποβληθούν σε επεξεργασία από την CPU του μικροελεγκτή. Ένα από τα προβλήματα με τους μικροελεγκτές AVR, από την άποψη του λογισμικού, είναι ότι δεν υπάρχει εκατό τις εκατό συμβατότητα μεταξύ τους. Για να τρέξουμε ένα πρόγραμμα γραμμένο για το ATtiny25 στον Atmega64, θα πρέπει να μεταγλωττίσουμε το πρόγραμμα ξανά και, ενδεχομένως, να αλλάξουμε θέση σε κάποιους registers πριν το φορτώσουμε στο Atmega64.

2.3 Οικογένεια των μικροελεγκτών AVR

Οι AVRs ταξινομούνται σε τέσσερις μεγάλες ομάδες οι οποίες είναι : ***tinyAVR, megaAVR, XMEGA, (Application Specific AVR), 32-bit AVRs.*** Ένα πράγμα που πρέπει να παρατηρήσουμε είναι ότι τα τσιπ της σειράς AVR, από το ATtiny15 μέχρι το Atmega328 περιλαμβάνουν το μέγεθος της μνήμης Flash στο όνομα τους. Δηλαδή, ο διαθέσιμος χώρος που έχουμε για το πρόγραμμα μας , μπορεί να είναι από 1KB έως 32KB. Για παράδειγμα το Arduino Uno R3 διαθέτει τον μικροελεγκτή Atmega 328, ο οποίος έχει 32 KB (0.5 KB χρησιμοποιούνται από τον Bootloader) μνήμη Flash, το Arduino Mega 2560 διαθέτει τον μικροελεγκτή Atmega2560, ο οποίος έχει 256 KB μνήμη flash (8 KB χρησιμοποιούνται από τον Bootloader).

Η διαφορά μεταξύ τους είναι τα διαθέσιμα χαρακτηριστικά κάθε μικροελεγκτή. Οι tinyAVR είναι συνήθως μικροελεγκτές με μικρότερο αριθμό ακροδεκτών (pin-count) ή περιορισμένες δυνατότητες συγκρινόμενοι με τους megaAVR's . Όλοι οι AVR έχουν

το ίδιο ρεπερτόριο εντολών (instruction set) και οργάνωση μνήμης έτσι ώστε να διευκολύνεται η αλλαγή του μικροελεγκτή μιας εφαρμογής όταν αυτό απαιτηθεί. Οι μικροελεγκτές AVR εκτός από την CPU περιλαμβάνουν ένα σύνολο περι-φερειακών μονάδων όπως στατική RAM (SRAM), EEPROM, διαύλους διασύνδεσης με εξωτερική SRAM μετατροπέα αναλογικού σήματος σε ψηφιακό (Analog to Digital Converter), μονάδα πολλαπλασιασμού (Hardware Multiplier), μονάδες σύγχρονης ή και ασύγχρονης επικοινωνίας (UART, USART) και πολλά άλλα περιφερειακά. Αν από οποιονδήποτε μικροελεγκτή AVR αφαιρέσουμε όλα τα περιφερειακά θα μείνει η κεντρική μονάδα επεξεργασίας (AVR Core). Η κεντρική αυτή μονάδα είναι ίδια για όλα τα μέλη της οικογένειας των AVR.

Όταν θέλουμε να επιλέξουμε τον κατάλληλο AVR για μία εφαρμογή πρέπει να έχουμε υπόψη ότι ο χαρακτηρισμός tinyAVR, megaAVR και XmegaAVR κ.α. δεν είναι χαρακτηρισμός επιδόσεων αλλά ένδειξη της πολυπλοκότητας του μικροελεγκτή: πολλά περιφερειακά = megaAVR, περιορισμένος αριθμός περιφερειακών = tinyAVR. Επομένως, αν πρέπει να επιλέξουμε έναν AVR με βάση το κόστος πρέπει να επιλέξουμε αυτόν με τα λιγότερα περιφερειακά που ικανοποιεί τις ανάγκες της εφαρμογής. Αν όμως δεν υπάρχει περιορισμός κόστους καλό είναι να χρησιμοποιήσουμε κάποιον με περισσότερες δυνατότητες, ώστε να μπορέσουμε ευκολότερα στο μέλλον να βελτιώσουμε τα χαρακτηριστικά της εφαρμογής.

2.3.1 Σειρά ATtiny

Οι συσκευές Atmel tinyAVR® έχουν σχεδιαστεί για εφαρμογές που απαιτούν απόδοση, εξοικονόμηση ενέργειας και ευκολία στη χρήση. Είναι ιδανικοί για συστήματα μικρού κόστους και δυνατοτήτων. Όπως το όνομα τους επιδεικνύει, οι μικροελεγκτές αυτής της κατηγορίας έχουν μικρότερο ρεπερτόριο εντολών και μικρότερες συσκευασίες συγκριτικά με εκείνους της κατηγορίας ATmega. Όλες οι συσκευές tinyAVR βασίζονται στην ίδια αρχιτεκτονική και είναι συμβατές με άλλες συσκευές AVR . Μερικά από τα χαρακτηριστικά τους είναι:

- Μνήμη προγράμματος (Program memory) 1K μέχρι 8Kbytes.
- Συσκευασία από 8 μέχρι 28 ακροδέκτες.

- Περιορισμένα περιφερειακά υποκυκλώματα.
- Περιορισμένο ρεπερτόριο εντολών.

2.3.2 Σειρά ATmega

Οι μικροελεγκτές Atmel® megaAVR® είναι η ιδανική επιλογή για εφαρμογές που απαιτούν μεγάλο κώδικα αφού, προσφέρουν σημαντικές μνήμες προγραμμάτων και δεδομένων με απόδοση έως και 20 MIPS. Εν τω μεταξύ, η πρωτοποριακή τεχνολογία Atmel picoPower® ελαχιστοποιεί την κατανάλωση ενέργειας. Τα μέλη αυτής της κατηγορίας είναι πανίσχυροι μικροελεγκτές που υποστηρίζουν μέχρι 120 εντολές και έχουν πολλά περιφερειακά κυκλώματα που μπορούν να χρησιμοποιηθούν σε μια ποικιλία συστημάτων. Μερικά από τα χαρακτηριστικά τους είναι :

- Μνήμη προγράμματος (Program memory) από 4K έως 256K.
- Συσκευασία από 28 έως 100 ακροδεκτών.
- Εκτεταμένα περιφερειακά υποκυκλώματα.
- Πλούσιο ρεπερτόριο εντολών.

2.3.3 Σειρά ATXmega

Οι μικροελεγκτές Atmel® AVR® XMEGA® διαθέτουν το καλύτερο δυνατό συνδυασμό απόδοσης σε πραγματικό χρόνο, υψηλής ενσωμάτωσης και χαμηλή κατανάλωση ενέργειας. Μερικά από τα χαρακτηριστικά τους είναι :

- Μνήμη προγράμματος (Program memory) από 16K έως 384K.
- Συσκευασία από 44-64 έως 100 ακροδεκτών.
- Εκτεταμένα χαρακτηριστικά επιδόσεων, όπως η DMA,”EventSystem”,καθώς και υποστήριξη κρυπτογράφησης.
- Εκτεταμένο σετ εντολών χειρισμού περιφερειακών και DAC.

2.3.4 Σειρά Application Specific AVR

Τα μέλη αυτής της κατηγορίας μπορούν να θεωρηθούν σαν υποκατηγορίες των προηγούμενων, αλλά τα ιδιαίτερα χαρακτηριστικά τους τα κάνουν ιδανικά για ειδικές

εφαρμογές. Μερικά χαρακτηριστικά τους είναι ο USB controller, CAN controller, LCD controller, Zigbee, Ethernet controller, FPGA, προχωρημένο PWM κ.ά.

2.3.5 Σειρά 32bit AVR'S

Το 2006 η Atmel κυκλοφόρησε μικροελεγκτές που βασίζονται στη νέα, 32-bit, AVR32 αρχιτεκτονική. Περιλαμβάνουν SIMD και DSP εντολές μαζί με χαρακτηριστικά επεξεργασίας βίντεο και ήχου. Αυτή η 32-bit οικογένεια των συσκευών κατασκευάστηκαν με σκοπό τον ανταγωνισμό με τους επεξεργαστές ARM. Το σετ εντολών τους είναι παρόμοιο με άλλους πυρήνες RISC, αλλά δεν είναι συμβατό με το αρχικό AVR.

Ο 32-bit μικροελεγκτής AVR® UC3 πηγαίνει την αποδοτικότητα σε ένα νέο επίπεδο, πέρα από την υψηλή απόδοση και χαμηλή κατανάλωση ενέργειας. Εγγενής σταθερό σημείο στήριξης DSP, διπλή είσοδο SRAM, πολλαπλών επιπέδων μεταφορά δεδομένων, περιφερική ελεγκτή DMA και ευφυή περιφερειακά πηγαίνουν την απόδοση και την κατανάλωση ηλεκτρικής ενέργειας στο επόμενο βήμα.

Ο ελεγκτής DMA περιφερειακών και η αρχιτεκτονική διαύλου υψηλής ταχύτητας κάνουν το 32-bit μικροελεγκτή ιδανικό για εφαρμογές υψηλής απόδοσης. Ευφυής περιφερειακά και ο δυναμικός έλεγχος της ισχύος καθιστούν τις συσκευές 32-bit τη προφανή επιλογή για φορητές και αυτόνομες (τροδοφοτούμενες από μπαταρίες) εφαρμογές.

3 Εισαγωγή στον Arduino

3.1 Ιστορία του Arduino

Το 2005 κάνει την εμφάνισή του το Arduino, όταν ο καθηγητής Massimo Banzi στο ινστιτούτο σχεδιασμού αλληλεπίδρασης στην πόλη της Ivrea, στην Ιταλία, θέλησε να καταστήσει ευκολότερη τη μάθηση των ηλεκτρονικών για τους μαθητές. Ήθελε να δώσει στους μαθητές την ευκαιρία να ανακαλύψουν οι ίδιοι, αντί να τα ακούν θεωρητικά χωρίς να κάνουν κάτι. Για τον σκοπό αυτό ζήτησε βοήθεια από τον David Cuatrielles, έναν μηχανικό από το Πανεπηστήμιο Malmö και μαζί αποφάσισαν να κάνουν ένα μικροελεγκτή που θα ήταν πιο προσιτός ως προς την χρήση του. Δύο φοιτητές επιλέχτηκαν να γράψουν το λογισμικό για την συσκευή. Ο ηλεκτρολόγος μηχανικός Gianluca Martino, κλήθηκε να κάνει μια αρχική παρτίδα των 200 μικροελεγκτών. Το όνομα Arduino δώθηκε από ένα ιστορικό χαρακλτήρα, τον Arduin της Ivrea. Το πρώτο Arduino που φτιάχτηκε ονομάστηκε "Serial Arduino» και περιλάμβανε μια ATmega8 με άμεση σύνδεση RS-232 με το μικροελεγκτή και όλα τα συστατικά του. Στη συνέχεια σχεδιάστηκε η έκδοση 2.0 και μια μονόπλευρη εκδοχή σαφέστερη για τους χομπίστες.

Οι εκδόσεις που ακολούθησαν ήταν όλες FTDI USB μετατροπέα. Μετά το USB v1.0 and v2.0, κυκλοφόρησε το Arduino Extreme το οποίο αύξησε την ποσότητα των επιφανειακών εξαρτημάτων. Το Arduino Nuova Generazione μεταβαίνει σε έναν απλούστερο μετατροπέα USB και μετατρέπεταιαπό το ATmega8 σε ATmega168. Οι βελτιώσεις συνεχίστηκαν με το Diecimila, το Duemilanove και το Uno αντικαθιστώντας τον μετατροπέα FTDI ATmega8U2, ο οποίος περιέχει έναν ελεγκτή USB και αναβαθμίζει τον μικροελεγκτή σε ATmega328.

Η τελευταία έκδοση του βασικού Arduino Leonardo, κινείτε εύκολα από εναλλάξιμη υποδοχή μικροελεγκτή σε μία επιφάνεια που έχει τοποθετηθεί συγκόλληση και απαιτεί να αλλαχθεί, καθώς εξαλείφει τον μετατροπέα πλήρως αναβαθμίζοντάς το σε ένα ATmega32U4, το οποίο περιέχει έναν ελεγκτή USB απλουστεύοντας έτσι τον σχεδιασμό σε μεγάλο βαθμό(Massimo 2008).

Έχουν φτιαχτεί επίσης εξειδικευμένες εκδόσεις. Οι μικρότερες εκδόσεις, όπως Mini και Nano επιτρέπου μικρότερες εγκαταστάσεις. Έχουμε μεγαλύτερες εκδόσεις με το ATmega1260 και το ATmega2560 για μεγαλύτερη επεκτασιμότητα. Μερικές εκδόσεις είναι επίσης διαθέσιμες σε μορφή Arduino BT για συνδεσιμότητα με Bluetooth καισε μορφή Arduino Ethernet για σύνδεση με LAN το οποίο εχει επίσης υποστίριξη για Power over Ethernet, εάν μια add-on μονάδα έχει εγκατασταθεί στο πλακίδιο. Το Arduino LilyPad είναι η σαφής υπόδειξη που έχει σχεδιαστεί για ευελιξία. Μπορούν να συνδεθούν 6 προστασίες στις πινέζες που είναι διαθέσιμες στο Arduino και περιλαμβάνει προστασία όπως Wi-Fi, Ethernet, Wireless, Motor και Proto για τη προτυποποίηση. Η Wireless ασπίδα έρχεται είτε με υποδοχή SD ή χώρο πρωτοτύπωνκαι έχει κεφαλίδες για την σύνδεση των μονάδων που χρησιμοποιούν το XBee αποτύπωμα. Το Motor επιτρέπει τον έλεγχο των 2 κινητήρων συνεχούς ρεύματος. Ο καθένας οπουδήποτε μπορεί να κάνει ασπίδες για αυτούς, εάν μπορούν να παράγουν τυπωμένα κυκλώματα.(Massimo 2008).

3.2 Τι είναι το Arduino

Το Arduino είναι μια ηλεκτρονική πλατφόρμα ανοιχτού λογισμικού. Βασίζεται στο ευέλικτο και εύκολο στη χρήση hardware και software που προορίζεται για οποιονδήποτε έχει λίγη προγραμματιστική εμπειρία, στοιχειώδεις γνώσεις ηλεκτρονικών και ενδιαφέρεται να δημιουργήσει διαδραστικά αντικείμενα ή περιβάλλοντα. Ουσιαστικά πρόκειται για ένα ηλεκτρονικό κύκλωμα που βασίζεται

Εικόνα 3-1 – Ο Arduino Uno (οι δύο όψεις του)

στον μικροελεγκτή ATmega της Atmel και του οποίου όλα τα σχέδια , καθώς και το software που χρειάζεται για τη λειτουργία του, διανέμονται ελεύθερα και δωρεάν

ώστε να μπορεί να κατασκευαστεί από τον καθένα. Αφού κατασκευαστεί, μπορεί να συμπεριφερθεί σαν ένας μικροσκοπικός υπολογιστής, αφού μπορείνα δεχτεί σαν είσοδο μια ποικιλία από αναλογικά ή ψηφιακά σήματα και να ελέγχει σύμφωνα με αυτά κάποιες περιφερειακές συσκευές που θα είναι συνδεδεμένες σε αυτόν, όπως LED's,διακόπτες, κινητήρες κ.τ.λ. Ο μικροελεγκτής του προγραμματίζεται χρησιμοποιώντας τη γλώσσα προγραμματισμού Arduino. Το αναπτυξιακό που επιλέχτηκε για την υλοποίηση του συστήματος και ενσωματώνει τον ATmega328p είναι το Arduino Uno.

3.3 Η γλώσσα προγραμματισμού που χρησιμοποιείται

Η γλώσσα προγραμματισμού που χρησιμοποιεί είναι η Wiring, η οποία είναι αρκετά εύκολη στη σύνταξη και διατίθεται σε πλατφόρμες Linux, MAC και Windows με άδεια χρήσης GPL. Αυτό όμως που κάνει το Arduino ακόμα πιο σημαντικό είναι ότι όλο το κύκλωμα της πλακέτας διατίθεται με άδεια χρήσης Creative Commons, πράγμα που σημαίνει ότι ο καθένας μπορεί να κατασκευάσει την δική του πλακέτα όπως αυτός επιθυμεί. Μάλιστα κάποιος θα μπορούσε να ισχυριστεί – και θα ήταν ένας αρκετά πετυχημένος παραλληλισμός- ότι λειτουργικά το Arduino μοιάζει πολύ με το NXTBrick των Lego Mindstorms NXT. Άλλωστε το Arduino διαπρέπει στις εφαρμογές της ρομποτικής. Το Arduino βέβαια, δεν είναι ούτε ο μοναδικός, ούτε και ο καλύτερος δυνατός τρόπος για την δημιουργία μιας οποιασδήποτε διαδραστικής ηλεκτρονικής συσκευής.

3.4 Πλεονεκτήματα του Arduino

Το κύριο πλεονέκτημα του Arduino είναι η τεράστια κοινότητα που το υποστηρίζει και η οποία έχει δημιουργήσει, επεκτείνει και συντηρήσει μια ανάλογου μεγέθους online γνωστική βάση. Έτσι , ενώ ένας έμπειρος ηλεκτρονικός μπορεί να προτιμήσει μια διαφορετική πλατφόρμα ή εξαρτήματα ανάλογα με την εφαρμογή που έχει στο νου του, το Arduino, με την εκτενές τεκμηρίωση, καταφέρνει να κερδίσει όλους

αυτούς των οποίων οι γνώσεις στα ηλεκτρονικά περιορίζονται στα όσα λίγα έμαθαν στο σχολείο με ένα ξεκάθαρο προγραμματιστικό περιβάλλον. Ακριβώς επειδή απευθύνεται κυρίως σε αρχάριους των ηλεκτρονικών και επειδή, παρά τις αναλυτικότατες οδηγίες που υπάρχουν, δεν έχουν όλοι τις γνώσεις και τα μέσα να κατασκευάσουν μια ηλεκτρονική πλακέτα, κυκλοφορούν έτοιμες, προκατασκευασμένες πλακέτες Arduino στο διαδίκτυο σε προσιτές τιμές. Επίσης οι περισσότεροι προμηθευτές με λίγα χρήματα παραπάνω μπορούν να διαθέτουν το Arduino StarterKit, το οποίο, εκτός από το ίδιο το Arduino, περιέχει διάφορα αλλα εξαρτήματα και εργαλεία που μπορεί να χρειαστούν για τις εφαρμογές (όπως το απαραίτητο καλώδιο USB για την σύνδεση με τον υπολογιστή, καλώδια, διακόπτες, LED, ποτενσιόμετρα, αντιστάσεις, διόδους, τρανζίστορ κ.λπ.). Επιπλέον βλέπουμε ότι τρέχει σε διάφορα λειτουργικά συστήματα. Οι μηχανικοί λογισμικού, ανέπτυξαν το περιβάλλον προγραμματισμού του Arduino για Windows, MachinstohOSX και για λειτουργικά συστήματα Linux. Τα περισσότερα συστήματα Μικροελεγκτών περιορίζονται στα Windows.

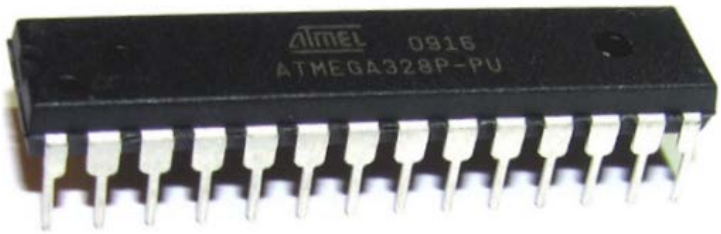
3.5 Δυνατότητες του Arduino

Αν και μικροσκοπικό (7x5 cm) οι δυνατότητες που προσφέρει είναι πάρα πολλές. Μπορούμε να το χρησιμοποιήσουμε σε εφαρμογές ρομποτικής και γενικότερα σε αυτοματισμούς καταφέρνοντας έτσι πάρα πολλά όπως: την κίνηση servo, stepper και DC κινητήρων, τη λήψη πληροφοριών από διάφορους αισθητήρες (θερμο-κρασίας, υγρασίας, υπέρυθρων κ.α.), την αμφίδρομη σειριακή επικοινωνία μεταξύ Arduino και PC χρησιμοποιώντας γλώσσες προγραμματισμού (όπως Java, python), όπως επίσης την αναπαραγωγή και αντίληψη ήχων. Η πλακέτα Arduino μέχρι αυτή την στιγμή διατίθεται σε 12 βασικές παραλλαγές οι οποίες αναφέρονται σε διαφορετικές χρήσεις η κάθε μία, ανάλογα με τις ανάγκες της εφαρμογής μας.

4 Ο Μικροελεγκτής AVR ATmega328p

4.1 Γενική περιγραφή του ATmega328P

Ο μικροελεγκτής ATmega328P βασίζεται στην αρχιτεκτονική RISC. Εκτελώντας ισχυρές εντολές σε έναν απλό κύκλο ρολογιού, ο ATmega328P επιτυγχάνει διαμεταγωγή, η οποία πλησιάζει το 1 MIPS ανά MHz, επιτρέποντας, έτσι, στο σχε-διαστή του συστήματος να βελτιστοποιήσει την κατανάλωση ενέργειας σε σχέση με την ταχύτητα επεξεργασίας.



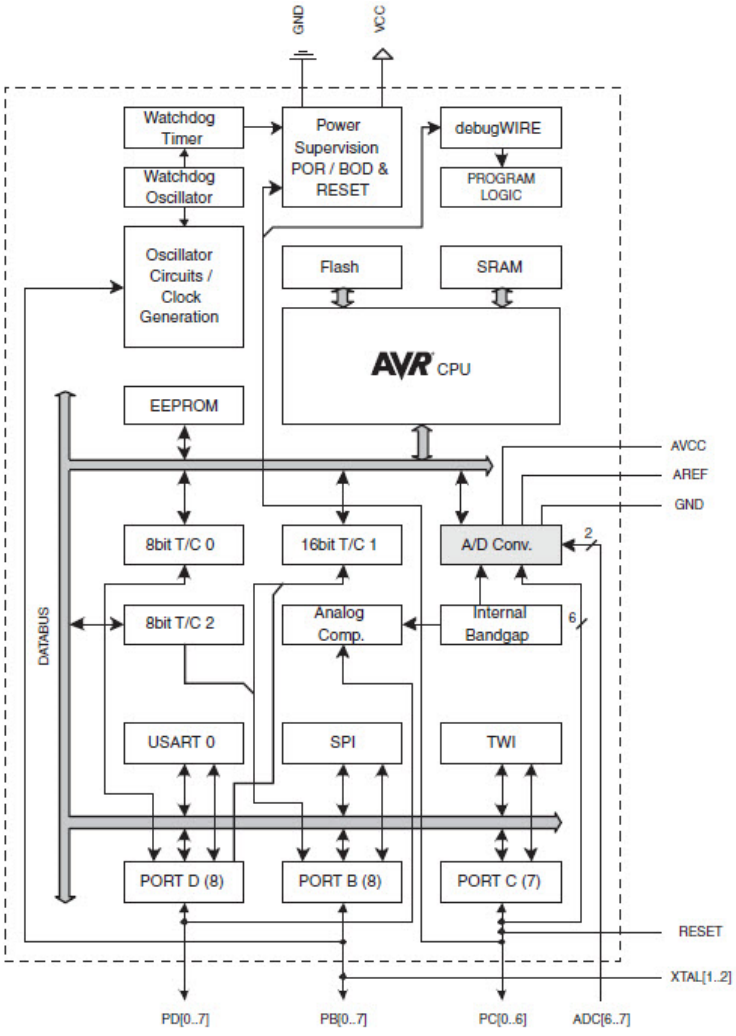
Εικόνα 4-1 – Ο μικροελεγκτής ATmega328P

Ο πυρήνας του μικροελεγκτή AVR συνδυάζει ένα πλούσιο σετ εντολών με 32 καταχωρητές γενικού σκοπού. Και οι 32 καταχωρητές συνδέονται άμεσα με τη Λογική και Αριθμητική Μονάδα (ALU), επιτρέποντας σε δύο ανεξάρτητους κατα-χωρητές να είναι προσβάσιμοι με μία μόνο εντολή, η οποία εκτελείται σε έναν κύκλο ρολογιού. Η αρχιτεκτονική που προκύπτει είναι περισσότερο αποτελεσματική σε ό,τι αφορά τον κώδικα, επιτυγχάνοντας, παράλληλα, διαμεταγωγή η οποία είναι μέχρι και δέκα φορές πιο γρήγορη σε σχέση με τους συμβατικούς μικροελεγκτές CISC.

Ο μικροελεγκτής ATmega328P παρέχει τις ακόλουθες δυνατότητες : 8Kbytes μιας προγραμματιζόμενης μνήμης Flash η οποία έχει δυνατότητες ανάγνωσης και εγγραφής, 1Kb EEPROM μνήμη, 2Kb SRAM μνήμη, 23 γραμμές εισόδου – εξόδου (I/O) γενικής χρήσεως, 32 καταχωρητές γενικού σκοπού, 3 Timer /Counters και PMW, 1 USART, μία σειριακή διεπαφή προσανατολισμένη σε byte 2 καλωδίων, έναν 8κάναλο μετατροπέα σημάτων ADC των 10 bits με προαιρετικό το στάδιο διαφο-ρικής εισόδου

με προγραμματιζόμενο κέρδος, προγραμματιζόμενο WatchdogTimer με εσωτερικό Oscillator, μία σειριακή θύρα SPI, και για τον προγραμματισμό του λογισμικού χρησιμοποιούνται πέντε τρόποι εξοικονόμησης ενέργειας. Η κατάσταση αναμονής σταματά τη CPU, καθώς επιτρέπει τη SRAM, το Timer/counters, τη θύρα SPI, τη σειριακή διεπαφή προσανατολισμένη σε byte 2 καλωδίων, τη USART και διακόπτει το σύστημα, προκειμένου να συνεχίσουν να λειτουργούν.

Τέλος, ο μικροελεγκτής ATmega328P υποστηρίζει ένα πλήρες σετ εργαλείων ανάπτυξης και προγραμματισμού. Συγκεκριμένα, περιλαμβάνει μεταγλωττιστές της C, macro assemblers, προγράμματα εντοπισμού σφαλμάτων και προσομοίωσης,

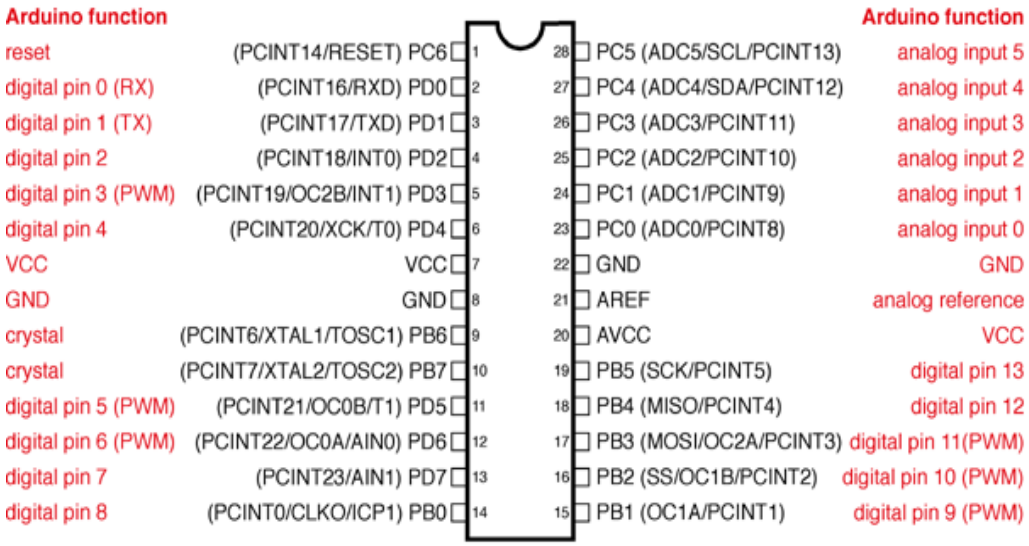


Σχήμα 7- Block διάγραμμα του ATmega328P

κύκλωμα εξομοίωσης και kit αξιολόγησης.

4.1.1 Περιγραφή ακροδεκτών

Παρακάτω παρουσιάζεται το pin mapping του μικροεπεξεργαστή ATmega 328P της



Σχήμα 8 – Διάγραμμα ακροδεκτών του ATmega328P

πλακέτας Arduino Uno.

Κάθε pin στον μικροελεγκτή AVR έχει ένα όνομα. Αν για παράδειγμα συνδέσουμε ένα Led στο pin 14, μπορούμε μετά να κάνουμε το pin αυτό HIGH ή LOW, να περάσουμε δηλαδή μία τάση, μιλώντας σε αυτό ως PB0. Τα περισσότερα pin σε ένα AVR έχουν και δευτερεύουσες λειτουργίες, αυτές παρατίθενται ως μνημονικά μέσα σε παρένθεση. Για παράδειγμα, τα RXD και TXD που χρησιμοποιούνται στην σει-ριακή επικοινωνία, βρίσκονται στα pins PD0 και PD1 αντίστοιχα.

Εσωτερικά, ας πούμε αναλόγως την λειτουργία τους, τα pins είναι χωρισμένα σε ομάδες το πολύ των οκτώ pins, όπως βλέπουμε στην παραπάνω εικόνα. Επειδή λοιπόν κάθε ομάδα περιλαμβάνει το πολύ 8 pin, μπορούμε να αναφερόμαστε σε αυτά χρησιμοποιώντας ένα δυαδικό αριθμό των 8 bit, ώστε να ενεργοποιήσουμε ή όχι την τάση τους. Αναλυτικότερα :

- **V_{cc}** Ψηφιακή Παροχή Τάσης (τροφοδοσία)
- **GND** Γείωση

- **Port B (PB7:0)**

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

  Ανάλογα με τις ρυθμίσεις ασφάλειας ρολογιού, ο ακροδέκτης PB6 μπορεί να χρησιμοποιηθεί σαν είσοδος στον ανάστροφο ενισχυτή ταλαντωτή και είσοδος στο εσωτερικό κύκλωμα ρολογιού.

  Ανάλογα με τις ρυθμίσεις ασφάλειας ρολογιού ,ο ακροδέκτης PB7 μπορεί να χρησιμοποιηθεί σαν έξοδος από τον ανάστροφο ενισχυτή ταλαντωτή.

- **Port C (PC5:0)**

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

- **PC6/RESET**

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

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

- **Port D (PD7:0)**

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

- **AVCC**

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

- **AREF**

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

- **ADC7:6 (Μόνο στη συσκευασία TQFP and QFN/MLF)**

  Στη συσκευασία TQFP and QFN/MLF, το ADC7:6 λειτουργεί σαν αναλογική είσοδο στον A/D μετατροπέα. Αυτά τα pin τροφοδοτούνται από την αναλογική τροφοδοσία και λειτουργούν σαν 10-bitADC κανάλια.

4.1.2 Χαρακτηριστικά του ATmega328P

Ο μικροελεγκτής AVR ATmega328P κατασκευάζεται από την εταιρεία ATMEL και έχει τα εξής χαρακτηριστικά:

- **AVR 8 bit μικροελεγκτή υψηλής απόδοσης και χαμηλής κατανάλωσης ισχύος**.
- **Προηγμένη αρχιτεκτονική RISC.**
  - 131 εντολές, εκ των οποίων οι περισσότερες απαιτούν ένα μόνο κύκλο του ρολογιού για την εκτέλεση τους.
  - 32 καταχωρητές μεγέθους 8-bit γενικής χρήσης.
  - Απόδοση μέχρι 20 MIPS στα 20MHz.
  - Ενσωματωμένο πολλαπλασιαστή δύο κύκλων ρολογιού.
- **Μη πτητική μνήμη προγράμματος και δεδομένων**.
  - 32 KB επαναπρογραμματιζόμενη μνήμη flash και πάνω στην πλακέτα, 10.000 κύκλων εγγραφών/διαγραφών.
  - 1 KB μνήμη EEPROM διάρκειας 100.000 κύκλων εγγραφών/διαγραφών.
  - 2 KB εσωτερική SRAM.
  - Προγραμματιζόμενο κλείδωμα για την ασφάλεια του προγράμματος.

- **Χαρακτηριστικά περιφερειακών**
  - Δύο 8 bit χρονιστές/Απαριθμητές με ανεξάρτητο προδιαιρέτη.
  - Έναν 16 bit χρονιστή/Απαριθμητή με ανεξάρτητο προδιαιρέτη.
  - Απαριθμητή πραγματικού χρόνου με ξεχωριστό ταλαντωτή.
  - Έξι κανάλια PWM.
  - Οκτώ κανάλια ADC στην συσκευασία TQFP και QFN/MLF με διακριτότητα 10 bit.
  - Έξι κανάλια ADC στην συσκευασία PDIP με διακριτότητα 10 bit.
  - Προγραμματιζόμενη Serial USART.
  - Master/Slave SPI λειτουργία.
  - Σειριακή θύρα με δυνατότητες σύγχρονης και ασύγχρονης λειτουργίας.
  - Προγραμματιζόμενο WatchdogTimer με ξεχωριστό on-chip ταλαντωτή.
  - On-chip αναλογικό συγκριτή.
- **Ειδικά χαρακτηριστικά του μικροελεγκτή.**
  - Power on reset (σύστημα επανεκκίνησης του μικροελεγκτή κατά την έναρξη της τροφοδοσίας).
  - Εσωτερικό ρολόι.
  - Εσωτερικές και εξωτερικές διακοπές.
  - Έξι sleep modes για εξοικονόμηση ενέργειας.
- **Μέγιστη συχνότητα λειτουργίας 20MHz.**
- **Θερμοκρασία λειτουργίας -40 έως 85 βαθμούς Κελσίου.**
- **Τάση λειτουργίας 1.8 – 5.5V.**

4.2 Αρχιτεκτονική του ATmega328P

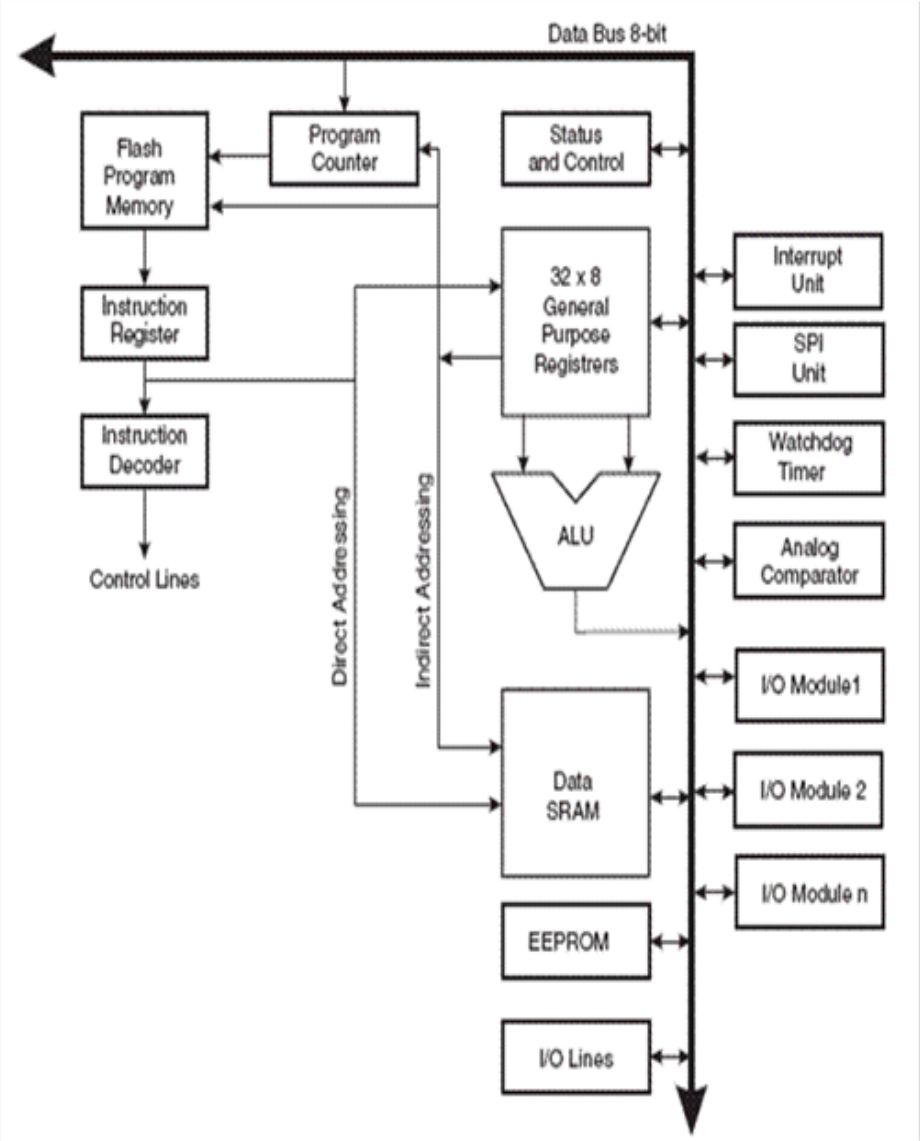
4.2.1 Βασική αρχιτεκτονική

Μία από τις διακρίσεις που μπορούμε να κάνουμε για την αρχιτεκτονική ενός υπολογιστικού συστήματος είναι σε Von Neumann και σε Harvard. Η διαφοροποίηση αυτών των δύο σχετίζεται με την θέση αποθήκευσης των εντολών και των δεδομένων του προγράμματος(όπως έχουμε προαναφέρει στο Κεφάλαιο 2). Στην μεν Von Neumann αρχιτεκτονική, οι εντολές και τα δεδομένα βρίσκονται στο ίδιο σύστημα μνήμης, ενώ στην Harvard σε διαφορετικό. Στην Von Neumann αρχιτεκτονική κάθε διεύθυνση είναι δυνατόν να αναφέρεται είτε σε εντολή, είτε σε δεδομένο καθώς τα 2 τελευταία καταλαμβάνουν τον ίδιο χώρο διευθύνσεων, σε αντίθεση με την Harvard αρχιτεκτονική, οπού υπάρχουν δύο διαφορετικοί χώροι διευθύνσεων.

Οι μικροελεγκτές AVR της σειράς ATMega , έχουν 8bit τροποποιημένη Harvard RISC αρχιτεκτονική. Στην τροποποιημένη Harvard αρχιτεκτονική υπάρχουν πράγματι ξεχωριστοί χώροι διευθύνσεων για το πρόγραμμα και για τα δεδομένα, με μία διαφορά, ο προγραμματιστής έχει πρόσβαση στην μνήμη του προγράμματος και μπορεί να διαβάσει ή και να γράψει σε αυτήν. Αυτό μας δίνει την δυνατότητα να αποθηκεύουμε δεδομένα μαζί με κώδικά στην μνήμη προγράμματος, καθώς και να τροποποιούμε τα περιεχόμενα της μνήμης. Όταν μία εντολή εκτελείται, η επόμενη εντολή έχει ήδη αναζητηθεί από τη μνήμη του προγράμματος. Αυτή η έννοια επιτρέπει τις εντολές να εκτελούνται σε κάθε κύκλο ρολογιού. Η μνήμη προγράμματος είναι In – System Reprogrammable Flash memory (επαναπρο-γραμματιζόμενη μνήμη Flash).

4.2.2 Ο πυρήνας της CPU του AVR

Η βασική λειτουργία της CPU είναι να εξασφαλίζει τη σωστή εκτέλεση των



Σχήμα 9 – Διάγραμμα αρχιτεκτονικής AVR

προγραμμάτων. Συγκεκριμένα, η CPU θα πρέπει να προσπελαύνει μνήμες, να εκτελεί υπολογισμούς, να ελέγχει περιφερειακές συσκευές και να διαχειρίζεται διακοπές (Interrupts).

Ο μικροελεγκτής διαθέτει 32 καταχωρητές γενικής χρήσης των 8 –bit, με χρόνο πρόσβασης έναν μόνο κύκλο ρολογιού. Αυτό σημαίνει πως σ ’ένα κύκλο ρολογιού μια μόνο λειτουργία της αριθμητικής – λογικής μονάδας εκτελείται.

Δύο τελεστές εξάγονται από τον τομέα των καταχωρητών , εκτελείται η πράξη και το αποτέλεσμα αποθηκεύεται πίσω στο αρχείο των καταχωρητών και αυτό σε ένα κύκλο ρολογιού.

Έξι από τους 32 καταχωρητές (οι καταχωρητές R26, R27, R28, R29, R30, R31) μπορούν να χρησιμοποιηθούν σε ζευγάρια ως δείκτες έμμεσης διευθυνσιοδότησης των 16 bits, διευκολύνοντας τους υπολογισμούς διευθύνσεων. Ένας από αυτούς τους καταχωρητές μπορεί, επίσης, να χρησιμοποιηθεί και ως δείκτης διεύθυνσης για πρόσβαση σε πίνακες δεδομένων που είναι αποθηκευμένοι στη μνήμη Flash του προγράμματος. Οι καταχωρητές R26 και R27 αντιστοιχούν στον καταχωρητή X, οι καταχωρητές R28 και R29 στον καταχωρητή Y και οι καταχωρητές R30 και R31 στον καταχωρητή Z.

Όσον αφορά τη Λογική και Αριθμητική μονάδα (ALU), αυτή επιτρέπει αριθμητικές και λογικές πράξεις μεταξύ των καταχωρητών και μεταξύ μίας σταθεράς και ενός καταχωρητή. Μετά από μία αριθμητική πράξη, ο καταχωρητής κατάστασης (Status Register) ενημερώνεται, ώστε οι πληροφορίες να αντιστοιχούν στο αποτέλεσμα της πράξης.

Η ροή του προγράμματος παρέχεται υπό όρους και άνευ όρων από ένα διακόπτη (jump) που καλώντας τις εντολές, μπορεί να προσπελάσει άμεσα το σύνολο του χώρου διευθύνσεων. Οι περισσότερες εντολές AVR έχουν μια μορφοποίηση 16 bit. Κάθε διεύθυνση μνήμης του προγράμματος περιέχει μια εντολή 16- ή 32-bit.

Ο χώρος του προγράμματος μνήμης Flash χωρίζεται σε δύο τμήματα, το τμήμα του προγράμματος εκκίνησης και το τμήμα προγράμματος εφαρμογής. Και τα δύο τμήματα έχουν αποκλειστικό bit κλειδώματος για write και read / write προστασία. Η εντολή SPM που γράφει στο τμήμα μνήμης εφαρμογών Flash πρέπει να παρα-μένει στην ενότητα Πρόγραμμα εκκίνησης.

Κατά τη διάρκεια διακοπών και κλήσεων υπορουτίνας, η διεύθυνση του αποστολέα (Program Counter (PC)) αποθηκεύεται στη στοίβα. Η στοίβα βρίσκεται αποκλειστικά στην SRAM δεδομένων και, κατά συνέπεια, το μέγεθος της, περιορίζεται μόνο από το συνολικό μέγεθος της SRAM και τη χρήση αυτής.

Όλα τα προγράμματα του χρήστη πρέπει να αρχικοποιήσουν τον SP στη ρουτίνα Reset (πριν την εκτέλεση οποιασδήποτε υπορουτίνας ή διακοπής (interrupt)). Το Stack Pointer (SP) είναι προσβάσιμο για read / write κατά τη διάρκεια I/O. Τα δεδομένα της SRAM είναι εύκολα προσβάσιμα μέσω πέντε διαφορετικών μεθόδων που υποστηρίζει η αρχιτεκτονική AVR.

Ο χώρος μνήμης στην αρχιτεκτονική AVR είναι γραμμικά προσπελάσιμος και οργανωμένος σε «χάρτες». Μια ευέλικτη μονάδα διακοπής έχει τους δικούς της καταχωρητές ελέγχου στο σύστημα I/O με ένα πρόσθετο γενικό interrupt ενεργοποιημένο στο Stack Pointer (SP).

Όλα τα interrupt έχουν ξεχωριστό Interrupt Vector[5] στον πίνακα Interrupt Vector (interrupt Vector Table)[6] . Τα interrupt έχουν προτεραιότητα σύμφωνα με τη θέση στην οποία βρίσκονται. Όσο πιο «χαμηλά» βρίσκονται στη μνήμη τόσο υψηλότερη προτεραιότητα έχουν. Ο χώρος της μνήμης που δεσμεύεται για το I/O περιέχει 64 διευθύνσεις για λειτουργίες της CPU όπως καταχωρητές ελέγχου, SPI και άλλες I/O λειτουργίες.

Η μνήμη I/O μπορεί να προσπελαστεί απευθείας ή σαν χώρος δεδομένων που «ακολουθεί» το μητρώο αρχείου 0x20 – 0x5F. Επιπλέον ο ATmega328P έχει εκεταμένο I/O χώρο από 0x60 – 0xFF στη μνήμη SRAM όπου μόνο εντολές ST/STS/STD και LD/LDS/LDD μπορούν να εκτελεστούν.

[5] Το interrupt vector είναι η τοποθεσία στη μνήμη που βρίσκεται ο διαχειριστής διακοπών (interrupthandler) , ο οποίος καθορίζει τη σειρά προτεραιότητας των interrupt και τα κρατάει σε μια ουρά ώσπου να εκτελεστούν αν υπάρχουν περισσότερα από ένα.
http://whatis.techtarget.com/definition/interrupt-vector

[6]**interrupt vector table** (IVT), είναι ένας γενικός όρος για τη δομή δεδομένων που αντιστοιχεί μια λίστα από interrupt handlers με μια λίστα interrupt requests σε ένα πίνακα interrupt vectors
https://en.wikipedia.org/wiki/Interrupt_vector_table

4.3 Περιφερειακές συσκευές του Μικροελεγκτή

Οι περιφερειακές συσκευές είναι αυτές που διαχωρίζουν έναν μικροελεγκτή από έναν επεξεργαστή. Βρίσκονται όλες μέσα στην ίδια συσκευασία και η ύπαρξη ή όχι μιας μονάδας είναι από τα κύρια κριτήρια επιλογής του κατάλληλου μικροελεγκτή. Οι περιφερειακές μονάδες διασυνδέονται με την κεντρική μονάδα επεξεργασίας και την μνήμη δεδομένων με τρόπο άμεσο ώστε η ανταπόκριση να είναι άμεση.

Στους μικροελεγκτές αυτής της σειράς βρίσκουμε τα παρακάτω περιφερειακά :

Πίνακας 1 - Συνήθη περιφερειακά στους μικροελεγκτές AVR

Διεπαφές επικοινωνίας	USB, UART, I2C, CAN, LIN
Διεπαφές χρήστη	LCD Segment display
Μετατροπείς	ADC, DAC, PWM
Αισθητήρες	Θερμοκρασίας
Ταλαντωτές	32kHz, 128kHz, 8MHz
Μετρητές	8bit, 16bit

- **Serial Communications:** Ο AVR διαθέτει τρείς σειριακές επικοινωνίες, την USART, η οποία είναι χρήσιμη στην επικοινωνία με τον ηλεκτρονικό μας υπολογιστή, με radio modems, GPS. Τα αρχικά του USART σημαίνουν Universal Synchronous and Asynchronous Receiver and Transmitter, και δεν κάνει χρήση του ρολογιού σε αντίθεση με το UART, Universal Asynchronous Receive and Transmit, το οποίο χρησιμοποιεί το ρολόι. Την SPI, Serial Peripheral Interface, η οποία είναι καλή για γρήγορη επικοινωνία σε μικρές αποστάσεις, όπως μνήμες, ADCs και DACs. Τέλος την I2C η οποία είναι σαν ένα μικρό δίκτυο, και μας επιτρέπει να συνδέσουμε πάνω από 127 διαφορετικούς αισθητήρες στο ίδιο ζεύγος καλωδίων. Συσκευές που διαχειρίζονται μικρή

ποσότητα δεδομένων συνήθως χρησιμοποιούν την I2C. Επειδή κάθε μία από τις περιφερειακές συσκευές είναι ξεχωριστή μέσα στον AVR, μπορούμε να χρησιμοποιήσουμε κάθε μία από αυτές ταυτόχρονα.

- **Analog to digital converter:** Ένας αριθμός από αισθητήρια που χρησιμοποιούνται, δεν «μιλούν» την ψηφιακή μητρική γλώσσα του μικροελεγκτή. Αντίθετα θα λέγαμε ότι ο τρόπος επικοινωνίας τους γίνεται με συνεχόμενα αναλογικά σήματα τάσης. Για να διαβάσουμε και να διαχειριστούμε αυτές τις τιμές όπως θα κάναμε με οποιαδήποτε άλλα ψηφιακά δεδομένα, θα χρειαστεί να τις περάσουμε από ένα analog to digital converter, ADC. Υπάρχουν δύο τεχνικές που μπορούν να μας αποφέρουν μεγαλύτερη ακρίβεια και μείωση του θορύβου στον ADC, αυτές είναι η oversampling και exponential smoothing.

- **Interrupts:** Είναι μία συνάρτηση που μπορούμε να γράψουμε στο πρόγραμμα μας, η οποία θα εκτελείτε αυτόματα κάθε φορά που συναντάται η συνθήκη ενός interrupt. Λέγεται ρουτίνα διακοπής (interrupt) επειδή ο επεξεργαστής σταματάει οτιδήποτε έκανε στην κύρια ροή του προγράμματος και τρέχει την κατάλληλη συνάρτηση. Μόλις τελειώσει με την ρουτίνα διακοπής, δηλαδή μετά την εκτέλεση της συνάρτησης του interrupt, ο επεξεργαστής ξεκινά και πάλι από το σημείο που είχε σταματήσει.

Υπάρχουν πολλοί τρόποι να προκαλέσουμε ένα interrupt σε ένα AVR μικροελεγκτή. Πατώντας το κουμπί reset, αλλάζοντας μία τιμή εισόδου, σε παλμό του ρολογιού, συγκεκριμένη τιμή στον counter, δεδομένα από και προς την σειριακή θύρα, στο πέρας της μετατροπής analog-to-digital και πολλά άλλα.

- **Clock:** Σε πολλά ψηφιακά κυκλώματα η σειρά με την οποία πραγματοποιούνται τα συμβάντα είναι κρίσιμη. Μερικές φορές κάποιο συμβάν πρέπει να προηγείται κάποιου άλλου, ενώ μερικές φορές δύο συμβάντα πρέπει να πραγματοποιούνται ταυτόχρονα. Για να μπορούν οι σχεδιαστές να πετύχουν τις απαιτούμενες χρονικές σχέσεις, πολλά ψηφιακά κυκλώματα χρησιμοποιούν ρολόγια για την επίτευξη του χρονισμού.

Υπάρχουν τέσσερις τύποι ρολογιών που χρησιμοποιούνται στους μικροελεγκτές, κρύσταλλα (crystals), κεραμικά αντηχεία (ceramic resonators), RC (resistor, capacitor) ταλαντωτές και ταλαντωτές πυριτίου (silicon oscillators). Θα χρησιμοποιήσουμε τον RC ταλαντωτή ως βασικό ρολόι. Τρέχει στα 8 Mhz περίπου. Στη συνέχεια το ρολόι της CPU διαιρείται από το βασικό ρολόι, και τρέχει περίπου στο 1Mhz από προεπιλογή, αλλά μερικές φορές όταν χρειαζόμαστε επιπλέον ταχύτητα, το ανεβάζουμε στα 8Mhz . Μετά από το ρολόι της CPU έρχονται όλα τα άλλα περιφερειακά ρολό-για, τα περισσότερα εκ των οποίων έχουν δικό τους prescaler σε σχέση με την CPU. Υπάρχουν ρολόγια για το υποσύ-στημα εισόδου εξόδου, για τον ADC (μετατροπέα από αναλογικό σε ψηφιακό), RAM, FLASH και EEPROM.

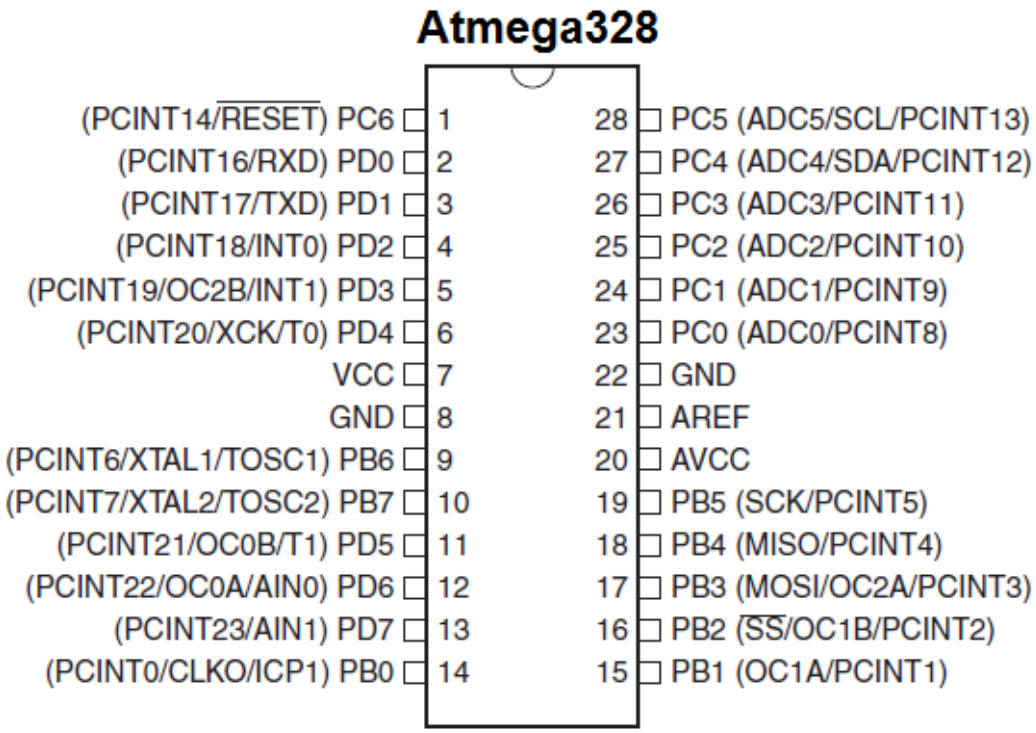
- **Timers/Counters:** Οι μικροεπεξεργαστές AVR έχουν ενσωματωμένους hardware μετρητές (counters). Οι μετρητές αυτοί κάνουν το προφανές, μετρούν πόσες φορές ένα pin ή μία εσωτερική πηγή άλλαξε την τάση της. Η πιο απλή εφαρμογή ενός counter, είναι να συνδέσουμε τον εσωτερικό counter με ένα κουμπί. Τώρα μπορούμε να δούμε πόσες φορές πατήθηκε αυτό το κουμπί απλά διαβάζοντας τον καταχωρητή (register) του μετρητή. Οι μετρητές μπορούν επίσης να συνδυαστούν και με ρολόγια, γι' αυτό και συχνά αναφέρονται ως timers/counters. Με ένα ρολόι και ένα timer, μπορείς να μετρήσεις είτε πόσο χρόνο διαρκεί κάποιο γεγονός, ή τη συχνότητα του γεγονότος αυτού.

4.4 Θύρες εισόδου/εξόδου (I/O PORTS)

Ο Atmega328 έχει 28 ακροδέκτες.

Από αυτούς οι 14 ακροδέκτες είναι για ψηφιακή είσοδο/έξοδο, από τους οποίους οι 6 μπορούν να χρησιμοποιηθούν σαν PWM[7] και 6 ακροδέκτες για αναλογικές εισόδους. Συνολικά έχουμε 20 ακροδέκτες εισόδου/εξόδου.

[7]Διαμόρφωση εύρους παλμών (**Pulse-widthmodulation - PWM**), ή διαμόρφωση παλμών διάρκειας (**pulse-durationmodulation- PDM**), είναι μια τεχνική διαμόρφωσης που χρησιμοποιείται για την κωδικοποίηση ενός μηνύματος σε παλμικό σήμα. Αν και αυτή η τεχνική διαμόρφωσης μπορεί να χρησιμοποιηθεί για την κωδικοποίηση των πληροφοριών για μετάδοση, η κύρια χρήση του είναι να επιτρέπει τον έλεγχο της ισχύος που παρέχεται στις ηλεκτρικές συσκευές, ιδίως για τα αδρανειακά φορτία, όπως κινητήρες.



Σχήμα 10 – Διάγραμμα ακροδεκτών

Το διάγραμμα των ακροδεκτών φαίνεται παρακάτω:

Ο παρακάτω πίνακας δίνει την περιγραφή για κάθε ακροδέκτη, καθώς επίσης και τη λειτουργία του.

Πίνακας 2 - Περιγραφή ακροδεκτών ATmega 328

Pin Number	**Description/ Περιγραφή**	**Function**	**Λειτουργία**
1	PC6	Reset	Reset
2	PD0	Digital Pin (RX)	Ψηφιακό pin (Λήψη)
3	PD1	Digital Pin (TX)	Ψηφιακό pin (αποστολή)
4	PD2	Digital Pin	Ψηφιακό pin

(https://en.wikipedia.org/wiki/Pulse-width_modulation) Με απλά λόγια είναι σαν ψηφιακό ρελέ.

5	PD3	Digital Pin (PWM)	Ψηφιακό pin(PWM)
6	PD4	Digital Pin	Ψηφιακό pin
7	Vcc	Positive Voltage (Power)	Θετική τάση τροφοδοσίας
8	GND	Ground	Γείωση
9	XTAL 1	Crystal Oscillator	Ταλαντωτής κρυστάλλου
10	XTAL 2	Crystal Oscillator	Ταλαντωτής κρυστάλλου
11	PD5	Digital Pin (PWM)	Ψηφιακό pin (PWM)
12	PD6	Digital Pin (PWM)	Ψηφιακό pin (PWM)
13	PD7	Digital Pin	Ψηφιακό pin
14	PB0	Digital Pin	Ψηφιακό pin
15	PB1	Digital Pin (PWM)	Ψηφιακό pin (PWM)
16	PB2	Digital Pin (PWM)	Ψηφιακό pin (PWM)
17	PB3	Digital Pin (PWM)	Ψηφιακό pin (PWM)
18	PB4	Digital Pin	Ψηφιακό pin
19	PB5	Digital Pin	Ψηφιακό pin
20	AVCC	Positive voltage for ADC (power)	Θετική τάση για τον A/D μετατροπέα
21	AREF	Reference Voltage	Τάση αναφοράς
22	GND	Ground	Γείωση
23	PC0	Analog Input	Αναλογική είσοδος
24	PC1	Analog Input	Αναλογική είσοδος
25	PC2	Analog Input	Αναλογική είσοδος
26	PC3	Analog Input	Αναλογική είσοδος

27	PC4	Analog Input	Αναλογική είσοδος
28	PC5	Analog Input	Αναλογική είσοδος

Όπως φαίνεται παραπάνω 20 από τους ακροδέκτες λειτουργούν σαν θύρες εισόδου/εξόδου. Αυτό σημαίνει ότι μπορούν να λειτουργήσουν σαν είσοδοι ή σαν έξοδοι στο κύκλωμα. Η χρήση τους καθορίζεται από το λογισμικό, 14 από αυτές είναι ψηφιακές από τις οποίες οι 6 μπορούν να λειτουργήσουν σαν έξοδοι ψηφιακού ρελέ (PWMoutput). Οι υπολειπόμενες 6 είναι για αναλογική είσοδο/ έξοδο.

Δύο από τους ακροδέκτες είναι για τη σύνδεση του κρυσταλλικού ταλαντωτή. Αυτός υπάρχει για να δίνει ένα παλμό ρολογιού στο Atmegachip. Ο παλμός αυτός χρειάζεται για το συγχρονισμό έτσι ώστε να μπορεί να επιτευχθεί η επικοινωνία μεταξύ του chip και του κυκλώματος στο οποίο συνδέεται.

Το chip επίσης χρειάζεται τροφοδοσία, έτσι 2 από τους ακροδέκτες, οι V_{cc} και GND, του παρέχουν την τροφοδοσία για να λειτουργήσει. Το Atmega328P είναι ένα χαμηλής ισχύος chip, έτσι χρειάζεται τάση μόνο μεταξύ 1,8 V και 5,5 V.

Το chip ATmega328P έχει στο εσωτερικό του έναν μετατροπέα από αναλογικό σε ψηφιακό (ADC). Αυτό υπάρχει γιατί αλλιώς το ATmega328P δεν θα είναι σε θέση να ερμηνεύσει αναλογικά σήματα.

Επειδή υπάρχει ο ADC, το τσιπ μπορεί να ερμηνεύσει αναλογική είσοδο, η οποία είναι ο λόγος για το ότι το τσιπ έχει 6 ακίδες για αναλογική είσοδο.

Το ADC έχει 3 ακίδες που χρειάζονται για τη λειτουργία του - AV_{CC}, AREF, και GND. Η AV_{CC} είναι η παροχή ηλεκτρικού ρεύματος, θετική τάση, ότι και το ADC. Το ADC χρειάζεται δικιά του τροφοδοσία για να λειτουργήσει.

- GND είναι η γείωση παροχής ηλεκτρικού ρεύματος.
- AREF είναι η τάση αναφοράς που χρησιμοποιεί ο ADC για να μετατρέψει ένα αναλογικό σήμα σε ψηφιακό αντίστοιχης αξίας. Αναλογικές τάσεις υψηλότερες από την τάση αναφοράς, θα πάρουν ψηφιακή τιμή 1, αλλιώς θα πάρουν ψηφιακή τιμή 0.

Καθώς ο ADC για το ATmega328P είναι ένας 10-bit μετατροπέας, που σημαίνει ότι παράγει μια 10-bit ψηφιακή τιμή δηλαδή, μετατρέπει ένα αναλογικό σήμα στην ψηφιακή τιμή του, με την τιμή AREF να είναι η τιμή αναφοράς για την μετατροπή του αναλογικού σήματος σε ψηφιακό. Έτσι, η εικόνα ενός αναλογικού σήματος φαίνεται από αυτή την ψηφιακή τιμή και συνεπώς, είναι η ψηφιακή τιμή του. Ο τελευταίος ακροδέκτης είναι το pin RESET. Αυτό επιτρέπει στο πρόγραμμα να κάνει επανεκκίνηση.

Σημείωση : Αναλυτικά για τον προγραμματισμό I/O στο Κεφάλαιο 7.1

4.5 Χρονιστές/Μετρητές

4.5.1 Ρολόι Συστήματος (clk)

Ο μικροελεγκτής για να λειτουργήσει πρέπει να παλμοδοτείται από ένα κύκλωμα χρονισμού. Ο ATmega328p μπορεί να παλμοδοτηθεί από τις πηγές που φαίνονται στον παρακάτω πίνακα και επιλέγονται από τα αντίστοιχα flash fuse bits CKSEL3...0

Πίνακας 3 - Επιλογή πηγής Ρολογιού

Επιλογή πηγής ρολογιού	*CKSEL3...0*
Χαμηλής ισχύος Εξωτερικός κρύσταλλος	1111 - 1000
Εξωτερικός κρύσταλλος	0111 - 0110
Εξωτερικός κρύσταλλος χαμηλής συχνότητας	0101 – 0100
Εσωτερικός ταλαντωτής 128kHz	0011
Ισοσταθμισμένος εσωτερικός ταλαντωτής	0010
Εξωτερικό ρολόι	0000
Δεσμευμένο	0001

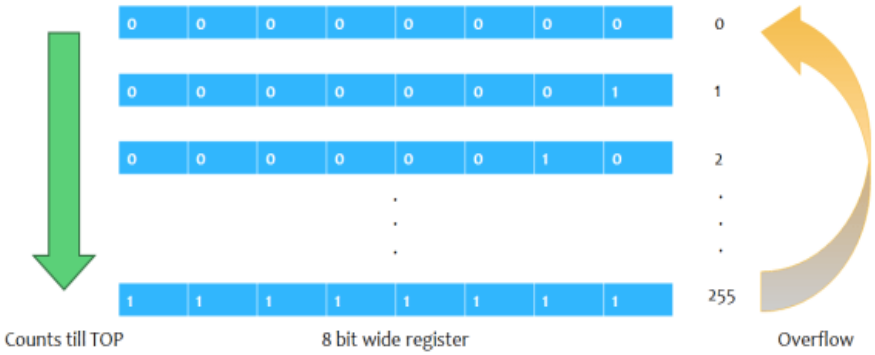
4.5.2 Χρονιστές

Οι timer/counters είναι από τις πιο κοινές διατάξεις που χρησιμοποιούνται σε έναν μικροελεγκτή. Είναι πολύ ευέλικτες περιφερειακές μονάδες και χρησι-μοποιούνται για να μετράνε περίοδους, να βρίσκουν το εύρος ενός παλμού, να μετράνε ταχύτητα, να βρίσκουν την συχνότητα ενός σήματος ή να παρέχουν σήματα χρονισμού. Οι timer/counters είναι στην ουσία δυαδικοί μετρητές. Όταν χρησι-μοποιούνται στην λειτουργία του χρονιστή , οι δυαδικοί μετρητές μετράνε τις περιόδους που εφαρμό-ζονται στις εισόδους τους, ενώ στην λειτουργία του μετρητή, μετράνε τους παλμούς που δέχονται στις εισόδους τους. Στην ουσία ένας χρονιστής είναι ένας κατα-χωρητής! Αλλά δεν είναι ένας κανονικός καταχωρητής, η τιμή του αυξάνεται/ μειώνεται αυτόματα.

Οι μικροελεγκτές AVR διαθέτουν μετρητές εύρους 8-bit και 16-bit. Ο ATmega328P διαθέτει 3 χρονιστές / μετρητές (timers/counters). Οι δύο είναι 8-bit και ονομάζεται T/C0 και T/C2 ενώ ο τρίτος είναι 16-bits και ονομάζεται T/C1.

4.5.3 T/C0 και T/C2

Οι timer0 και timer2 είναι τυπικοί 8-bit χρονιστές , χρησιμοποιούνται συχνά για να δίνουν σήμα χρονισμού σε μια διακοπή. Ξεκινάνε να μετράνε από 0 και μόλις φτάσουν την τιμή 255 υπερχειλίζουν και ξεκινάνε από την αρχή. Την στιγμή της υπερχείλισης προκαλείται μια διακοπή στο πρόγραμμα. Το πότε φτάνει ένας μετρητής στην υπερχείλιση του ,πρέπει να το γνωρίζει το πρόγραμμα ,έτσι ώστε να



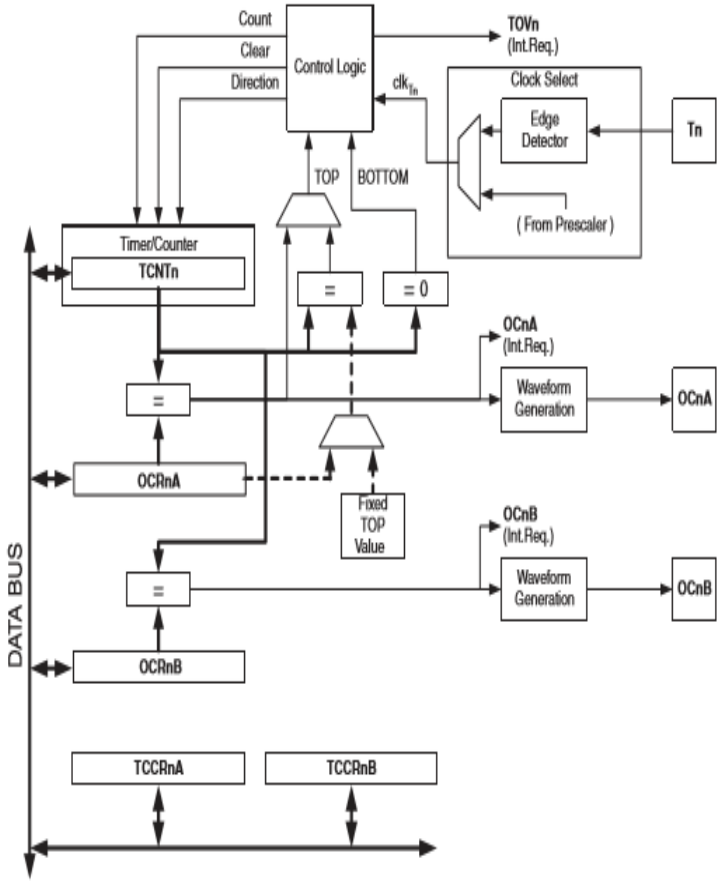
Σχήμα 11 - Λειτουργία υπερχείλισης

υπάρχει ακρίβεια στα αποτελέσματα που διέπονται από τους χρονιστές(βλ. σχήμα).

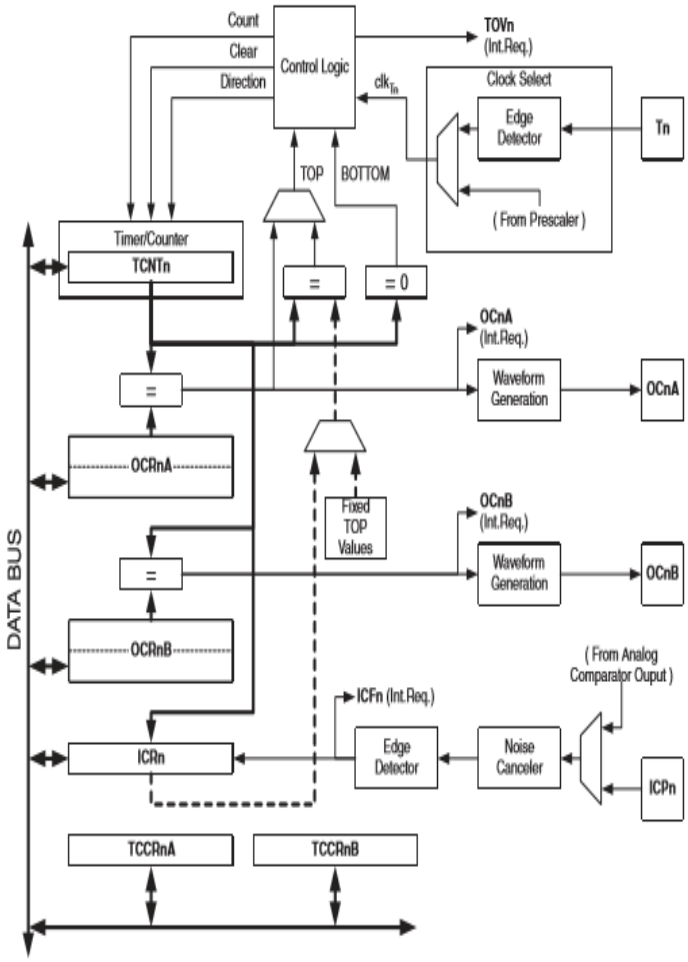
4.5.4 T/C1

Ο timer1 είναι πιο πολύπλοκος και χρησιμοποιείται σε περισσότερες εφαρμογές από τον timer0. Εκτός από τις λειτουργίες του timer0, ο timer1 διαθέτει τον καταχωρητή capture με εύρος 16 bit καθώς επίσης περιέχει και δύο καταχωρητές σύγκρισης εύρους 16 bit. Ο καταχωρητής capture μπορεί να χρησιμοποιηθεί για μέτρηση εύρους παλμού ή να μετράει χρόνους. Οι καταχωρητές σύγκρισης χρησιμοποιούνται για να παράγουν συχνότητες ή παλμούς από τον timer/counter σε ένα pin εξόδου του μικροελεγκτή.

Η αρχιτεκτονική των χρονιστών φαίνεται στα παρακάτω σχήματα:



Σχήμα 12 - 8-bit Timer/Counter Block Diagram



Σχήμα 13 - 16-bit Timer/Counter Block Diagram

Σημείωση : Αναλυτικότερα για τον προγραμματισμο με χρονιστές στο Κεφάλαιο 7.2.

4.6 Διακοπές (interrupt)

4.6.1 Τι είναι μία διακοπή (interrupt);[8]

Σε πολύ βασικό επίπεδο, μια διακοπή είναι ένα σήμα που *διακόπτει* (σταματά) την τρέχουσα ενέργεια του επεξεργαστή. Αυτό μπορεί να προκληθεί από ένα εξωτερικό γεγονός (αλλαγή στην κατάσταση ενός ακροδέκτη) ή από ένα εσωτερικό γεγονός (ένα σχήμα από χρονιστή ή το λογισμικό). Μόλις ενεργοποιηθεί η διακοπή, παύει την

[8] https://arduinodiy.wordpress.com/2012/02/27/arduino-hardware-interrupt/

τρέχουσα λειτουργία και επιβάλει στο πρόγραμμα να εκτελέσει μια διαφορετική λειτουργία. Αυτή η λειτουργία ονομάζεται διαχειριστής διακοπών (interrupt handler) ή ρουτίνα διακοπών (Interrupt Service Routine – ISR). Όταν αυτή η λειτουργία τελειώσει, το πρόγραμμα συνεχίζει από εκεί που σταμάτησε με τη διακοπή.

4.6.2 Είδη διακοπών (Interrupts)

Ο επεξεργαστής έχει δύο διαφορετικούς τύπους interrupt: «εξωτερικά» και «αλλαγή κατάστασης ακροδεκτών».

4.6.2.1 Εξωτερικά

Στον Atmega328P υπάρχουν δύο ακροδέκτες που υλοποιούν τα εξωτερικά interrupt: οι INT0 και INT1. Αυτά τα interrupt μπορούν να ρυθμιστούν ώστε να προκαλέσουν άνοδο ή πτώση του σήματος ή να το κατεβάσουν σε πολύ χαμηλό επίπεδο. Οι λειτουργίες αυτές υλοποιούνται από το υλικό και η διακοπή είναι πολύ γρήγορη.

4.6.2.2 Αλλαγή κατάστασης ακροδεκτών

Από την άλλη μεριά, η αλλαγή της κατάστασης μπορεί να υλοποιηθεί από πολύ περισσότερους ακροδέκτες. Στον Atmega328P μπορεί να υλοποιηθεί από οποιονδήποτε από τους 20 ακροδέκτες εισόδου/εξόδου.

Αυτοί πυροδοτούνται εξίσου στην αύξηση ή στη μείωση (στο ανέβασμα ή στο κατέβασμα) του σήματος, οπότε εξαρτάται από τον κώδικα του προγράμματος διακοπής να ρυθμίσει τους σωστούς ακροδέκτες να δεχτούν τα interrupt, για να αποφασίσουν τι έγινε (ποιος ακροδέκτης; ….αν το σήμα άλλαξε;)και να το χειριστούν σωστά. Επιπλέον η λειτουργία αυτή (αλλαγή κατάστασης ακροδεκτών) ομαδοποιείται σε τρεις θύρες (ομάδες) στον μικροελεγκτή (MCU), έτσι υπάρχουν μόνο 3 υπορουτίνες για τον έλεγχο ολοκλήρου του αριθμού των ακροδεκτών.

Ο ATmega328P έχει 26 πηγές διακοπών και ισάριθμες θέσεις στην περιοχή της Μνήμης Προγράμματος στις οποίες καθορίζεται η διεύθυνση της αντίστοιχης υπορουτίνας χειρισμού διακοπής(InterruptServiceRoutine – ISR). Οι θέσεις αυτές είναι σταθερές και αποτελούν τον πίνακα διακοπών(Interrupt Vector) ο οποίος απεικονίζεται στον παρακάτω Πίνακα.

Πίνακας 4 - Πίνακας Διακοπών (Interrupt)

VectorNo	Program Address (hex)[1]	Source	Interrupt Definition
1	0x0000[2]	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog SystemReset
2	0x0002	INT0	External Interrupt Request0
3	0x0004	INT1	External Interrupt Request1
4	0x0006	PCINT0	Pin Change Interrupt Request0
5	0x0008	PCINT1	Pin Change Interrupt Request1
6	0x000A	PCINT2	Pin Change Interrupt Request2
7	0x000C	WDT	Watchdog Time-outInterrupt
8	0x000E	TIMER2COMPA	Timer/Counter2 Compare MatchA
9	0x0010	TIMER2COMPB	Timer/Counter2 Compare MatchB
10	0x0012	TIMER2OVF	Timer/Counter2Overflow
11	0x0014	TIMER1CAPT	Timer/Counter1 CaptureEvent
12	0x0016	TIMER1COMPA	Timer/Counter1 Compare MatchA
13	0x0018	TIMER1COMPB	Timer/Coutner1 Compare MatchB
14	0x001A	TIMER1OVF	Timer/Counter1Overflow
15	0x001C	TIMER0COMPA	Timer/Counter0 Compare MatchA
16	0x001E	TIMER0COMPB	Timer/Counter0 Compare MatchB
17	0x0020	TIMER0OVF	Timer/Counter0Overflow
18	0x0022	SPI,STC	SPI Serial Transfer Complete

19	0x0024	USART,RX	USART RxComplete
20	0x0026	USART,UDRE	USART, Data RegisterEmpty
21	0x0028	USART,TX	USART, TxComplete
22	0x002A	ADC	ADC ConversionComplete
23	0x002C	EEREADY	EEPROMReady
24	0x002E	ANALOGCOMP	AnalogComparator
25	0x0030	TWI	2-wire SerialInterface
26	0x0032	SPMREADY	Store Program MemoryReady

Σημειώσεις

1. Όταν το bit IVSEL[9] στο MCUCR[10] έχει οριστεί, το αίτημα διακοπής, θα μετακινηθεί από την έναρξη του τμήματος Boot Flash. Η διεύθυνση του τότε κάθε αιτήματος διακοπής θα είναι η διεύθυνση στον πίνακα αυτό που προστίθεται στην αρχική διεύθυνση του τμήματος Boot Flash.

2 .Όταν ο διακόπτης BOOTRST είναι ενεργός (προγραμματισμένος), η συσκευή θα μεταβεί στη διεύθυνση Boot Loader σε επαναφορά (reset).

Αναλυτικότερα για τον προγραμματισμο με διακοπές στο Κεφάλαιο 7.3.

[9] InterruptVSelect (IVSEL) – όταν τεθεί στην τιμή 0 όλα τα αιτήματα διακοπής τοποθετούνται στην αρχή της flashmemory

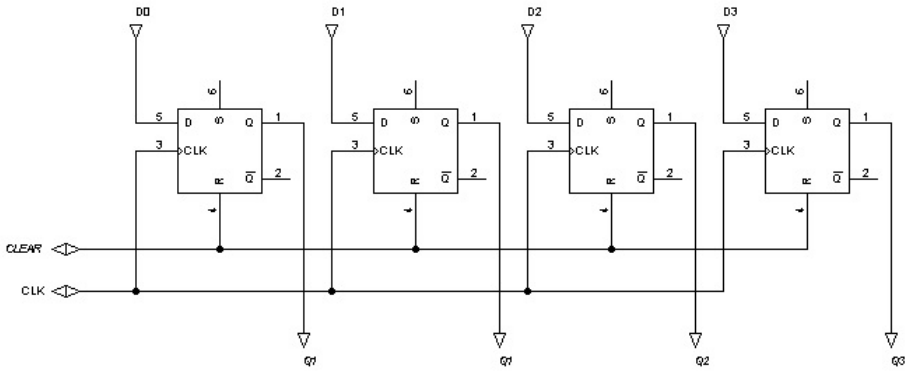
[10] MCUControlRegister (MCUCR) - περιέχει bit διαμόρφωσης το οποίο μας λέει τι είδους σήματα θα ενεργοποιούν τα εξωτερικά interruptINT0 και INT1.

4.7 Καταχωρητές

4.7.1 Τι είναι ένας καταχωρητής[11]

Σε έναν υπολογιστή, ο καταχωρητής είναι ένας υψηλής ταχύτητας ειδικός χώρος αποθήκευσης εντός του επεξεργαστή. Ένας καταχωρητής μπορεί να κρατάει εντολές, διευθύνσεις και οτιδήποτε άλλο είδος δεδομένων.

Από κατασκευαστική άποψη[12] ένας καταχωρητής μπορεί να θεωρηθεί σαν μια ομάδα



Σχήμα 14 - Συνδεσμολογία Flip – Flop σε καταχωρητή

από D Flip Flops που μοιράζονται τον ίδιο παλμό χρονισμού.

Το κάθε D flip flop (βλ. παράρτημα Α) κρατάει ένα bit μνήμης.

4.7.2 Οι καταχωρητές του ATmega168/328

Ο μικροελεγκτής όπως και όλοι οι μικροεπεξεργαστές/μικροελεγκτές έχει δύο ειδών καταχωρητών:

- Γενικού σκοπού.
- Ειδικού σκοπού.

[11] http://whatis.techtarget.com/definition/register
[12] http://sandbox.mc.edu/~bennet/cs314/slides/ch5me-4.pdf

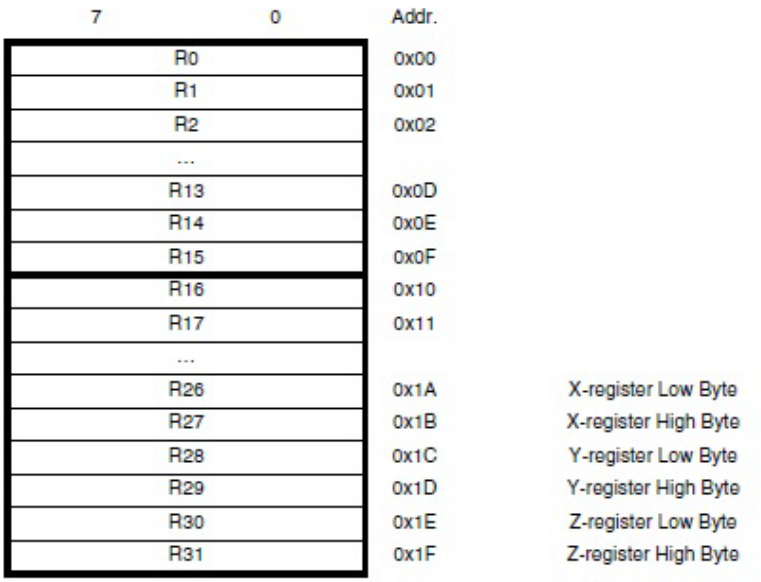
4.7.2.1 Καταχωρητές γενικού σκοπού

Οι καταχωρητές γενικού σκοπού είναι καταχωρητές τους οποίου μπορεί να χρησιμοποιήσει ο προγραμματιστής για να αποθηκεύσει προσωρινά δεδομένα κατά την εκτέλεση ενός προγράμματος. Ο Atmega328p έχει 32 καταχωρητές γενικού σκοπού.

Το Αρχείο καταχωρητών (Register File) έχει βελτιστοποιηθεί για τους μικροελεγκτές AVR με ένα σύνολο εντολών της ενισχυμένης αρχιτεκτονικής RISC. Οι εντολές αυτές έχουν ένα ή δύο ορίσματα (operands), τα οποία είναι πάντοτε οι καταχωρητές γενικού σκοπού R0 έως R31. Προκειμένου να επιτευχθεί η απαιτούμενη απόδοση και ευελιξία στα ακόλουθα συστήματα εισόδου/εξόδου που υποστηρίζονται από το RegisterFile, έχουμε:

- ένα όρισμα 8bit εξόδου και ένα 8bit εισόδου αποτελέσματος
- δύο ορίσματα 8bit εξόδου και ένα 8bit εισόδου αποτελέσματος
- δύο ορίσματα 8bit εξόδου και ένα 16bit εισόδου αποτελέσματος
- ένα όρισμα 16bit εξόδου και ένα 16bit εισόδου αποτελέσματος

Ο μικροελεγκτής AVR αποτελείται από 32 καταχωρητές γενικής χρήσης, οι οποίοι παρουσιάζονται στο παραπάνω σχήμα.

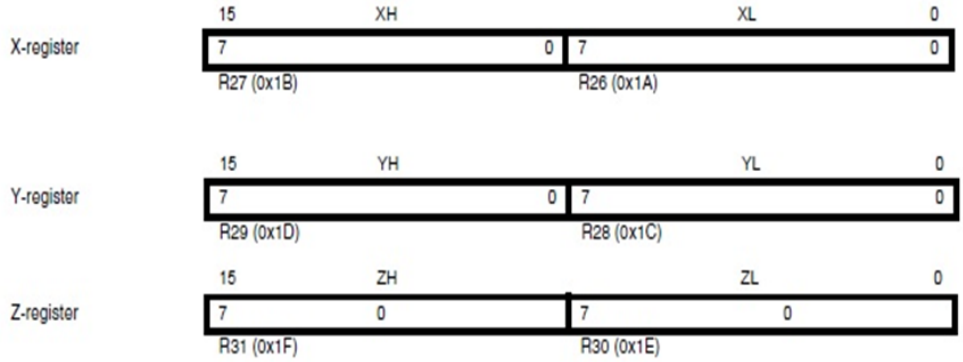


Σχήμα 15 - Καταχωρητές γενικής χρήσης

Κάθε καταχωρητής έχει μία διεύθυνση μνήμης δεδομένων, η χαρτογράφηση των οποίων δείχνει απευθείας στις 32 πρώτες θέσεις του χώρου δεδομένων του χρήστη. Η οργάνωση της μνήμης παρέχει μεγάλη ευελιξία όσον αφορά την πρόσβαση των καταχωρητών, όπως τα X-, Y- και Z- pointer registers, τα οποία μπορούν να ρυθμιστούν σε κάθε καταχωρητή μέσα στο αρχείο.

4.7.2.1.1 Καταχωρητές X,Y,Z

Οι καταχωρητές R26, R27, R28, R29, R30 και R31 έχουν μερικές επιπλέον λειτουργίες. Οι συγκεκριμένοι καταχωρητές είναι δείκτες διεύθυνσης εύρους 16-bit για την έμμεση διευθυνσιοδότηση του χώρου δεδομένων. Οι καταχωρητές αυτοί αντιστοιχίζονται (ανά δύο) σε 3 άλλους καταχωρητές στο X, Y και Z.



Σχήμα 16 – Οι καταχωρητές X, Y, Z

Ο καταχωρητής X ορίζεται από τους καταχωρητές R27, R26, αντίστοιχα ο καταχωρητής Y ορίζεται από τους καταχωρητές R29, R28 και ο καταχωρητής Z ορίζεται από τους καταχωρητές R31, R30. Οι καταχωρητές X, Y και Z έχουν εύρος 16bit και χρησιμοποιούνται ως δείκτες (δείκτης X, δείκτης Y, δείκτης Z).

4.7.2.2 Καταχωρητές ειδικού σκοπού

Οι καταχωρητές ειδικού σκοπού σχεδιάστηκαν για να εκτελούν κάποια συγκεκριμένη εργασία. Κάθε κατασκευαστής μικροεπεξεργαστή / μικροελεγκτή τους ονομάζει

διαφορετικά. Στο Παράρτημα Β παρουσιάζεται ο πίνακας με τους καταχωρητές ειδικού σκοπού για τον ATmega328.

Στην συνέχεια αναλύονται συνοπτικά τρείς από τους πιο συνηθισμένους καταχωρητές ειδικού σκοπού.

4.7.2.2.1 Program Counter[13]

Ο Program Counter (PC) είναι ένας καταχωρητής που υπάρχει σε όλους τους μικροεπεξεργαστές (CPU). Όλοι οι μικροελεγκτές περιέχουν ένα μικροεπεξεργαστή και αυτός έχει ένα program counter. Ο σκοπός αυτού είναι να κρατάει / αποθηκεύει τη ***διεύθυνση (address)*** της ***επόμενης εντολής*** που πρόκειται να εκτελεστεί από τον μικροεπεξεργαστή του μικροελεγκτή.

Το μέγεθος (width) του PC ενός μικροελεγκτή μετριέται σε bits και είναι ευθέως ανάλογο από το μέγεθος της μνήμης του μικροελεγκτή.

Καθώς μια εντολή φορτώνεται (fetched)[14], ο PC αυξάνει την τιμή του κατά 1. Μετά την εκτέλεση ο PC δείχνει την επομένη εντολή της ακολουθίας. Όταν γίνεται reset ή restart φυσιολογικά ο PC γυρίζει στην τιμή 0[15].

Δε συμπεριφέρεται σαν ένας συνηθισμένος καταχωρητής, αλλά χρησιμοποιούνται ειδικές εντολές (π.χ. JUMP, CALL, RTS) για να αλλάξουν οι τιμές του.[16]

4.7.2.2.2 StackPointer[17]

- Τι είναι το «stack» (στοίβα)

Το ***stack*** είναι ένα κομμάτι/χώρος από τη μνήμη δεδομένων που ορίζεται από τον προγραμματιστή. Αυτό το κομμάτι μπορεί να χρησιμοποιηθεί και από τον

[13] http://www.avr-tutorials.com/general/avr-program-counter
[14] Το Fetchείναι το πρώτο από τα δύο στάδια που εμπλέκονται στην επεξεργασία. Ο επεξεργαστής λειτουργεί με την εκτέλεση οδηγιών σε αυτό που λέγεται "fetch/executecycle." Ο επεξεργαστής «fetches» (διαβάζει από τη μνήμη) μια εντολή και μετά, ανάλογα την εντολή, την εκτελεί. Μετά διαβάζει την επόμενη εντολή κ.λ.π.
(http://searchsqlserver.techtarget.com/definition/fetch)
[15] http://whatis.techtarget.com/definition/program-counter
[16] http://www.dictionary.com/browse/program-counter
[17] http://www.avr-tutorials.com/general/avr-microcontroller-stack-operation-and-stack-pointer

μικροελεγκτή και από τον προγραμματιστή για προσωρινή αποθήκευση δεδομένων. Λειτουργεί με τη λογική του ***LIFO***[18]μηχανισμού.

- Τι είναι ο ***Stack Pointer***?

Ο stack pointer βασικά είναι ένας καταχωρητής ή ομάδα καταχωρητών που κρατά είτε «την διεύθυνση μνήμης της τελευταίας θέσης στο stack στην οποία τα δεδομένα έχουν αποθηκευτεί» είτε «τη διεύθυνση μνήμης από την επομένη διαθέσιμη θέση στο stack για να αποθηκευτούν δεδομένα». Αυτό εξαρτάται από τον σχεδιασμό του μικροελεγκτή.

Στους AVR μικροελεγκτές, όπως τους ATMega8515, ATMega16, ATTiny13 κλπ, το stackpointer κρατά την διεύθυνση από την επομένη διαθέσιμη θέση στο stack για να αποθηκευτούν δεδομένα.

- Ο AVR Stack Pointer

Στον 8-bit AVR μικροελεγκτή ο stack pointer μπορεί να αποτελείται ή από ένα μόνο καταχωρητή εισόδου/εξόδου τον ***SPL***[19] ή από δύο (2) καταχωρητές εισόδου/εξόδου τους ***SPL*** και ***SPH***[20]. Το μέγεθος του Stack Pointer εξαρτάται από τη μνήμη δεδομένων

Bit	15	14	13	12	11	10	9	8	
	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	SPH
	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	
	0	0	0	0	0	0	0	0	

Σχήμα 17 - Stack Pointer

που περιέχει ο μικροελεγκτής. Αν ολόκληρη η μνήμη μπορεί να προσπελαστεί χρησιμοποιώντας 8-bit τότε ο stackpointer έχει εύρος 8-bit δηλαδή μόνο ο SPL, σε άλλη περίπτωση αποτελείται από τους SPL και SPH.

[18]LIFOακρωνύμιο των λέξεων LastInFirstOut (τελευταίο μπαίνει, πρώτο βγαίνει)
[19] SPL – Stack Pointer Low
[20] SPH – Stack Pointer High

4.7.2.2.3 Καταχωρητής κατάστασης – StatusREGister (SREG)

Ο καταχωρητής κατάστασης περιέχει πληροφορίες που αφορούν τα αποτελέσματα των πιο πρόσφατων αριθμητικών πράξεων. Οι πληροφορίες αυτές μπορούν να χρησιμοποιηθούν για την αλλαγή της ροής του προγράμματος, προκειμένου να εκτελέσουν εργασίες υπό όρους. Επιπλέον, ο συγκεκριμένος καταχωρητής κατάστασης ενημερώνεται μετά από όλες τις πράξεις της ALU. Ακόμη, ο καταχωρητής κατάστασης περιλαμβάνει 8 bits, το καθένα από τα οποία αντιστοιχεί σε μία σημαία (flags). Οι σημαίες μας ενημερώνουν για την κατάσταση στην οποία βρίσκεται ο επεξεργαστής.

Bit	7	6	5	4	3	2	1	0
	I	T	H	S	V	N	Z	C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

I - Global Interrupt Enable
C - carry flag
N - Negative Flag
S - Sign Bit, S = N EXOR V
T - Bit Copy Storage
Z - Zero Flag
V - Two's Complement Overflow Flag
H - Half Carry Flag

Σχήμα 18 - Status Register

SREG – StatusRegister:

Παρακάτω ακολουθεί αναλυτική επεξήγηση για το κάθε bit

- **Bit 7 – I: Global Interrupt Enable –** Καθολική Ενεργοποίηση Διακοπών: Το συγκεκριμένο bit καθορίζει την ενεργοποίηση των διακοπών του συστήματος. Το bit ενεργοποίησης της καθολικής διακοπής πρέπει να οριστεί για τους καταχωρητές για να ενεργοποιηθεί. Η ενεργοποίηση της κάθε μεμονωμένης διακοπής πραγματοποιείται σε ξεχωριστά μητρώα ελέγχου (registers). Αν ο καταχωρητής του καθολικού διακόπτη είναι «καθαρός», κανένα από τα interrupt δεν είναι ενεργό, ανεξάρτητα από τους επιμέρους διακόπτες τους.

Το I-bit «καθαρίζει» μηχανικά μετά από ένα συμβάν (interrupt has occurred), και ορίζεται από τις RETI οδηγίες για να ενεργοποιηθούν μεταγενέστερες διακοπές. Το I-bit μπορεί επίσης να οριστεί και να καθαριστεί από την εφαρμογή με τις SEI και CLI εντολές όπως περιγράφονται στον οδηγό αναφοράς.

- **Bit 6 – T (Target): Bit Copy Storage –** Σημαία αντιγραφής – αποθήκευσης: Οι εντολές [BLD (BitLoaD – ανάγνωση ενός bit) και BST (BitSTore – αποθήκευση ενός bit)] του BitCopy χρησιμοποιούν το T-bit ως πηγή ή ως προορισμό για τη λειτουργία του συγκεκριμένου bit. Ένα bit από έναν καταχωρητή του RegisterFile μπορεί να αντιγραφεί σε T-bit με την εντολή BST και ένα bit σε T μπορεί να αντιγραφεί σε ένα bit ενός καταχωρητή του RegisterFile με την εντολή BLD.

- **Bit 5 – H: Half Carry Flag –** Σημαία δεκαδικού κρατουμένου: Το συγκεκριμένο bit ανανεώνεται, όταν υπάρχει δεκαδικό κρατούμενο μετά την εκτέλεση μερικών αριθμητικών πράξεων. Αυτό το bit είναι χρήσιμο στην αριθμητική BCD. Επίσης, προκύπτει από τα 4 λιγότερο σημαντικά ψηφία.

- **Bit 4 – S: SignBit, S = N⊕V –** Σημαία προσήμου: Το συγκεκριμένο bit αντιστοιχεί στη σημαία που προκύπτει από τη λογική πράξη Ή (OR) ανάμεσα στη σημαία αρνητικού προσήμου N και τη σημαία υπερχείλισης στην αριθμητική συμπληρώματος ως προς δύο.

- **Bit 3 – V: Two's Complement Overflow Flag –** Σημαία υπερχείλισης: Είναι η σημαία υπερχείλισης στην αριθμητική συμπληρώματος ως προς δύο.

- **Bit 2 – N: Negative Flag:** Σημαία που δηλώνει ότι ο αριθμός ο οποίος προκύπτει από μία αριθμητική ή από μία λογική πράξη είναι αρνητικός.

- **Bit 1 – Z: Zero Flag –** Σημαία μηδενισμού: Η συγκεκριμένη σημαία δηλώνει ότι το αποτέλεσμα μιας αριθμητικής ή λογικής πράξης είναι 0.

- **Bit 0 – C: Carry Flag –** Σημαία κρατουμένου: Η σημαία κρατουμένου δηλώνει ότι μετά από μία αριθμητική ή λογική πράξη έχει προκύψει κρατούμενο.

4.8 Μνήμες

Η μνήμη είναι σαν τον ανθρώπινο εγκέφαλο. Χρησιμοποιείται για να αποθηκεύει δεδομένα και εντολές. Η μνήμη του υπολογιστή είναι το αποθηκευτικό μέσο στο οποίο τα δεδομένα προς επεξεργασία και οι εντολές του προγράμματος αποθηκεύονται.[21]

Κατασκευαστικά υλοποιείται κατά κύριο λόγο από DFlipFlop σε διάταξη «latch»[22]

Ο Amtel 8-bit RISC μικροελεγκτής έχει ενσωματωμένες τις παρακάτω μνήμες:

- 32 Kb flash memory
- 1 Kb EEPROM
- 2 Kb SRAM

Οι οποίες αναλύονται παρακάτω.

4.8.1 Μνήμη προγράμματος Flash

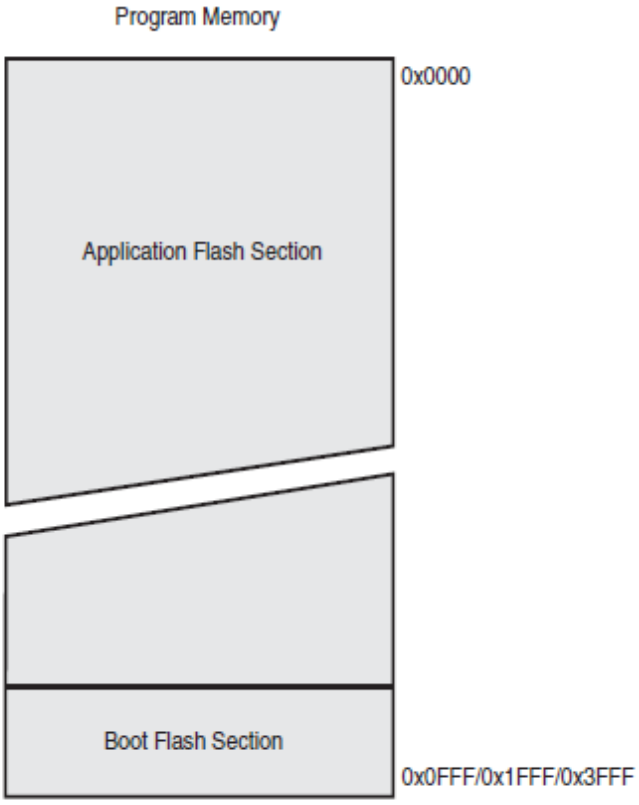
Ο μικροελεγκτής ATmega328P διαθέτει ενσωματωμένη μνήμη Flash (32Kbytes) για την αποθήκευση των προγραμμάτων και έχει την δυνατότητα να επανα-προγραμματίζεται μέσα στο σύστημα. Δεδομένου ότι όλες οι εντολές του AVR είναι εύρους 16 ή 32 bits, η Flash memory είναι οργανωμένη ως 16K x 16. Για την ασφάλεια του λογισμικού, ο χώρος μνήμης προγράμματος χωρίζεται σε δύο τμήματα: στο

[21] http://www.tutorialspoint.com/computer_fundamentals/computer_memory.htm
[22] RogerL. Tokheim, *Ψηφιακά Ηλεκτρονικά,* μτφ. Αριστοτέλης Μπιζόπουλος, 3η έκδοση, εκδ ΤΖΙΟΛΑ, Θεσσαλονίκη 1991, σελ 140-141

τμήμα εκκίνησης προγράμματος (Boot Program section) και στο τμήμα εφαρμογής προγράμματος (Application Program section). Επίσης, η μνήμη είναι αμετάβλητου τύπου, διατηρεί τα περιεχόμενά της αναλλοίωτα, ακόμα και όταν δεν υπάρχει τάση τροφοδοσίας, και το περιεχόμενό της μπορεί να γραφεί ξανά μέσω του προγραμματιστή για περίπου 10.000 φορές. Ο μετρητής προγράμματος (PC) του μικροελεγκτή έχει μέγεθος 14 bit, έτσι ώστε να καταφέρει να προσπελάσει όλες τις θέσεις μνήμης.

4.8.2 Μνήμη δεδομένων (StaticRam ή SRAM)

Η στατική RAM (Static RAM - SRAM) είναι ένας τύπος μνήμης RAM που έχει την ικανότητα να διατηρεί αναλλοίωτα τα περιεχόμενά της για όσο χρονικό διάστημα

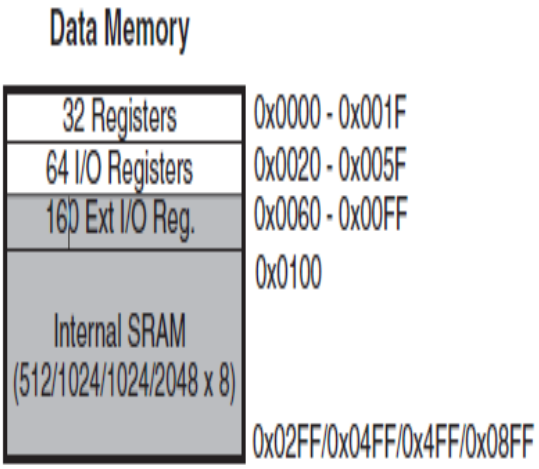


Σχήμα 19 - Χαρτογράφηση της Flash memory

τροφοδοτείται με ρεύμα, χωρίς να απαιτείται κάποια επιπλέον εξωτερική

επέμβαση. Στις SRAM's χρησιμοιούνται ειδικοί διακόπτες (switches), που μπορεί να είναι ανοικτοί ή κλειστοί (on/off). Ο τρόπος κατασκευής της SRAM μοιάζει περισσότερο με την τεχνολογία που εφαρμόζεται στους επεξεργαστές: πολλά ολοκληρωμένα κυκλώματα, που έχουν τοποθετηθεί πάνω σε μια μικρή πλακέτα πυριτίου.

Στο μικροελεγκτή ATmega328P οι 2303 θέσεις μνήμης δεδομένων αφορούν: το αρχείο καταχωρητών (RegisterFile), τη μνήμη εισόδου – εξόδου (I/O), την επεκτεταμένη μνήμη εισόδου – εξόδου (Extended I/O Memory) και την εσωτερική μνήμη δεδομένων SRAM. Οι πρώτες 32 θέσεις αντιστοιχούν στους καταχωρητές (στο



Σχήμα 20 – Χαρτογράφηση της μνήμης SRAM

αρχείο των καταχωρητών, RegisterFile), οι επόμενες 64 θέσεις είναι για τη μνήμη καταχωρητών εισόδου – εξόδου (I/OMemory), οι 160 που ακολουθούν αφορούν την επεκτεταμένη μνήμη καταχωρητών εισόδου – εξόδου (Extended I/O memory) και οι επόμενες 2048 θέσεις καλύπτουν την εσωτερική SRAM μνήμη.

4.8.3 Μνήμη EEPROM

Ο μικροελεγκτής ATmega328P περιέχει μία EEPROM μνήμη των 1Kbytes. Η μνήμη είναι οργανωμένη ως ξεχωριστός χώρος δεδομένων στο μικροελεγκτή, όπου απλά

bytes μπορούν να διαβαστούν και να γραφούν. Η EEPROM έχει δυνατότητα για τουλάχιστον 100.000 κύκλους εγγραφής/διαγραφής.

Για την προσπέλαση της CPU στη μνήμη αυτή χρησιμοποιούνται συγκεκριμένοι καταχωρητές που είναι ο καταχωρητής δεδομένων της Eeprom(EEDR), ο καταχωρητής διεύθυνσης(EEAR) και ο καταχωρητής ελέγχου της Eeprom(EECR). Οι μνήμες EEPROM αποτελούν εξέλιξη των προγενέστερων μνημών ROM. Οι συγκεκριμένες μνήμες, σε αντίθεση με τις αντίστοιχες RAM, διατηρούν τα περιεχόμενά τους και μετά τη διακοπή της τροφοδοσίας τους με ηλεκτρική ισχύ. Μπορούν, όμως, να διαγραφούν και να επαναπρογραμματιστούν με νέες πληροφορίες με ηλεκτρικό τρόπο, ακόμη και πάνω στο κύκλωμα στο οποίο είναι τοποθετημένες.

5 Σύνολο εντολών

5.1 Σετ Εντολών[23]

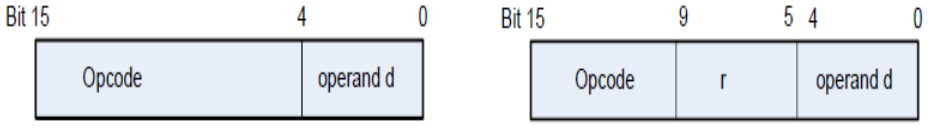
Ένα σύνολο εντολών είναι μια ομάδα εντολών για τη CPU σε γλώσσα μηχανής. Ο όρος μπορεί να αναφέρεται σε όλες τις πιθανές οδηγίες για έναν επεξεργαστή ή σε ένα υποσύνολο οδηγιών για να βελτιώσει την απόδοσή του σε ορισμένες περιπτώσεις. Όλοι οι επεξεργαστές έχουν σύνολα οδηγιών που καθοδηγούν τον επεξεργαστή να ενεργοποιήσει τα σχετικά τρανζίστορ. Μερικές οδηγίες είναι απλές εντολές ανάγνωσης, εγγραφής και εντολές κίνησης που κατευθύνουν τα δεδομένα σε διαφορετικό υλικό.

Στους CISC (Complex Instruction Set Computer) επεξεργαστές υπάρχει επίσης ένα επίπεδο μικροκώδικα, το οποίο περιλαμβάνει προγραμματιζόμενες εντολές που αποθηκεύονται στη flash μνήμη που μπορεί να ενημερωθεί.

Η RISC (μειωμένο σύνολο Οδηγίες υπολογιστή) αρχιτεκτονικής, από την άλλη πλευρά δεν απαιτεί μικροκώδικα, αλλά έχει ένα μεγαλύτερο σύνολο εντολών βάσης.

Για το πλήρες σετ εντολών βλέπε το παράρτημα Γ.

5.2 Δομή και κατηγορίες εντολών



Σχήμα 21– Τμήματα εντολής

Οι εντολές περιλαμβάνουν 2 τμήματα. Το πρώτο (operation code ή opcode) αποτελεί το λειτουργικό κώδικα που πληροφορεί τον επεξεργαστή για τις ενέργειες που πρέπει

[23] http://whatis.techtarget.com/definition/instruction-set

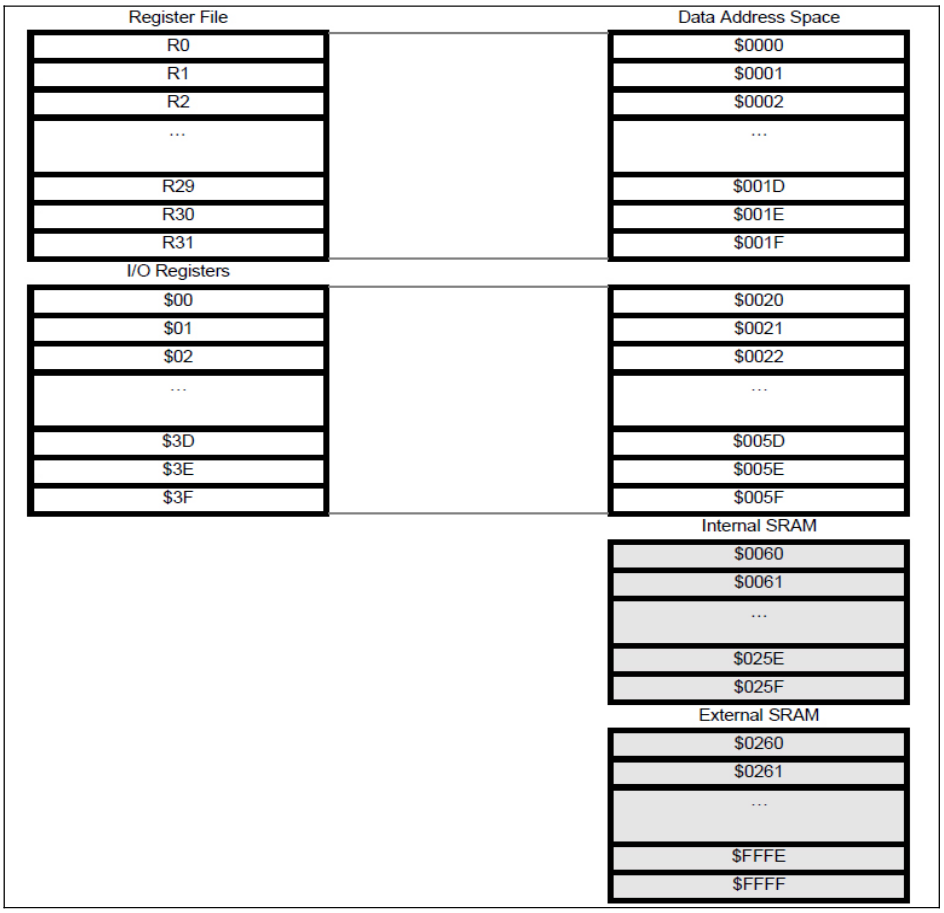
να εκτελεστούν. Το δεύτερο τμήμα προσδιορίζει τους τελεστές (r, operand) για τους οποίους θα λειτουργήσει ο κώδικας.

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

- Μεταφοράς δεδομένων
- Αριθμητικών και λογικών πράξεων
- Σε επίπεδο bit και ελέγχου bit
- Ελέγχου ροής του προγράμματος και διακλάδωσης

5.2.1 Εντολές μεταφοράς δεδομένων

Πρόκειται για εντολές μεταφοράς δεδομένων μεταξύ καταχωρητών ή μεταξύ



Σχήμα 22– Χαρτογράφηση μνήμης δεδομένων

καταχωρητών και μνήμης ή μεταξύ καταχωρητή και σταθεράς. Οι σημαίες δεν επηρεάζονται. Ο καταχωρητής είναι ένας από τους 32 καταχωρητές εργασίας R0-R31.

Για να γίνουν κατανοητοί οι διαθέσιμοι τρόποι διευθυνσιοδότησης πρώτα από όλα πρέπει να έχουμε στο μυαλό μας την χαρτογράφηση της μνήμης δεδομένων όπως αυτή φαίνεται στο παραπάνω σχήμα.

Παρατηρήση:

Οι καταχωρητές γενικού σκοπού αντιστοιχίζονται και σε συγκεκριμένη διεύθυνση, άρα πρόσβαση σε αυτούς μπορούμε να έχουμε τόσο αναφερόμενοι στο όνομά τους (R0….R31) όσο και μέσω της διεύθυνσής τους ($0000…… $001F). Το κομμάτι μνήμης από $0020 μέχρι $005F αντιστοιχίζονται σε καταχωρητές I/O, κατά συνέπεια πρέπει να είμαστε προσεκτικοί όταν κάνουμε πρόσβαση στο κομμάτι αυτό. Τέλος στο κομμάτι μνήμης από $0260 μέχρι $FFFF μπορούμε να έχουμε πρόσβαση μόνο αν υπάρχει εξωτερική μνήμη SRAM (στα πλαίσια του εργαστηρίου δεν χρειάζεται και κατά συνέπεια δεν υπάρχει).

Αφού έχουμε καλή αντίληψη για την διαθέσιμη μνήμη θα παρουσιάσουμε τους υποστηριζόμενους τρόπους διευθυνσιοδότησης, τις αντίστοιχες εντολές που παρέχονται για κάθε τρόπο, αντίστοιχα παραδείγματα και σημαντικές λεπτομέρειες.

5.2.1.1 Άμεσος Τρόπος Διευθυνσιοδότησης (Immediate)

Ο τρόπος αυτός μας δίνει τη δυνατότητα να μεταφέρουμε σε έναν καταχώρηση μια συγκεκριμένη αριθμητική ποσότητα. Η εντολή που υποστηρίζει τον τρόπο αυτό είναι η LDI. Η κατεύθυνση μεταφοράς, όπως και σε όλες τις εντολές, είναι από το δεύτερο όρισμα προς το πρώτο.

Πίνακας 5- Εντολή LDI

LDI Rd,K **Rd←K**	Περιγραφή : Άμεση αποθήκευση στον καταχωρητή προορισμού Rd της αριθμητικής ποσότητας k.
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Η εντολή **LDI R20, $16** θα έχει σαν αποτέλεσμα την μεταφορά του αριθμού 16H στα περιεχόμενα του καταχωρητή R20.
	Περίοδοι Ρολογιού : 1

5.2.1.2 Μεταφορά από καταχωρητή σε καταχωρητή

Η εντολή **MOV** μας δίνει τη δυνατότητα αντιγραφής δεδομένων μεταξύ δύο καταχωρητών.

Πίνακας 6 – Εντολή MOV

MOV Rd, Rr **Rd←Rr**	Περιγραφή : Αντιγραφή των δεδομένων του καταχωρητή Rr στον καταχωρητή Rd.
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Η εντολή **MOV R20, R16** θα έχει σαν αποτέλεσμα να αντιγραφούν τα δεδομένα του καταχωρητή **R16** στον **R20**. Με άλλα λόγια, μετά την εκτέλεση της εντολής και οι δύο καταχωρητές θα περιέχουν ότι περιείχε ο R16 πριν την εκτέλεση.
	Περίοδοι Ρολογιού : 1

5.2.1.3 Απ' ευθείας διευθυνσιοδότηση(Direct)

Με τις αντίστοιχες εντολές ο AVR μας επιτρέπει να διαβάζουμε και να γράφουμε απ' ευθείας (Direct) σε μια θέση μνήμης SRAM. Οι εντολές αυτές είναι οι **LDS** (**L**oad **D**irect from **S**RAM) και **STS** (**S**tore **D**irect to **S**RAM).

Πίνακας 7 – Εντολή LDS

LDS Rd, k **Rd←(k)**	Περιγραφή : Μεταφορά των δεδομένων από τη διεύθυνση μνήμης SRAM k στον καταχωρητή Rd.
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Η εντολή **LDS R20, $70** θα έχει σαν αποτέλεσμα να διαβαστούν τα περιεχόμενα της διεύθυνσης **$70** και να αντιγραφούν στα περιεχόμενα **R20**.
	Περίοδοι Ρολογιού : 3

Πίνακας 8 – Εντολή STS

STS k, Rd **(k)←Rd**	Περιγραφή : Μεταφορά των δεδομένων από τον καταχωρητή Rd στη διεύθυνση μνήμης SRAM k.
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Η εντολή **STS $70, R20** θα έχει σαν αποτέλεσμα τα περιεχόμενα του καταχωρητή **R20** να γραφούν στη διεύθυνση μνήμης **$70**.
	Περίοδοι Ρολογιού : 3

5.2.1.4 Έμμεση διευθυνσιοδότηση (Indirect)

Όπως έχει αναφερθεί και στο Κεφάλαιο Καταχωρητές, υπάρχουν τρεις λογικοί καταχωρητές X, Y, Z εύρους 16bit . Για να μπορέσουμε να εκφράσουμε 16bit αριθμούς χρησιμοποιούμε συγκεκριμένα ζευγάρια των 8bit καταχωρητών γενικού σκοπού. Συγκεκριμένα ο X ορίζεται από το ζευγάρι R27, R26, ο Y ορίζεται από το ζευγάρι R29, R28 και ο Z ορίζεται από το ζευγάρι R31, R30.

Ο όρος έμμεση διευθυνσιοδότηση αφορά ακριβώς τη χρήση αυτών των λογικών καταχωρητών για να επιτύχουμε πρόσβαση στο address space. Οι εντολές που χρησιμοποιούν το συγκεκριμένο τρόπο διευθυνσιοδότησης είναι οι **LD, LDD, ST, STD.**

Πίνακας 9 – Εντολές LD, LDD

1) LD Rd, X **Rd←(X)** **2) LD Rd, X+** **Rd←(X), X←X+1** **3) LD Rd, -X** **X←X-1, Rd ← (X)**	Περιγραφή : Στην περίπτωση 1 γίνεται μεταφορά δεδομένων από τη διεύθυνση μνήμης SRAM X στον καταχωρητή Rd. Στην περίπτωση 2 ***μετά*** την μεταφορά γίνεται αύξηση των δεδομένων του X κατά 1. Στην περίπτωση 3 ***πριν*** τη μεταφορά γίνεται μείωση των δεδομένων του X κατά 1.
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Αναλυτικά παρακάτω
	Περίοδοι Ρολογιού : 2
LDD Rd, Y+q **Rd← (Y+q)**	Περιγραφή : Μεταφορά των δεδομένων από τη διεύθυνση μνήμης SRAM (Y+q) στον καταχωρητή Rd
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Αναλυτικά παρακάτω
	Περίοδοι Ρολογιού : 2

Πίνακας 10 - Εντολές ST, STD

1) ST X, Rr **(X)← Rr** **2) ST X+, Rr** **(X)← Rr, X← X+1** **3) ST –X, Rr** **X← X-1, (X) ← Rr**	Περιγραφή : Στην περίπτωση 1 γίνεται μεταφορά των δεδομένων από τον καταχωρητή Rr στην διεύθυνση μνήμης SRAM X. Στην περίπτωση 2 ***μετά*** την μεταφορά γίνεται αύξηση των δεδομένων του X κατά 1. Στην περίπτωση 3 ***πριν*** τη μεταφορά γίνεται μείωση των δεδομένων του X κατά 1.
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Αναλυτικά παρακάτω
	Περίοδοι Ρολογιού : 2
STD Z+q, Rr **(Z+q)← Rr**	Περιγραφή : Μεταφορά των δεδομένων από τον καταχωρητή Rr στη διεύθυνση μνήμης SRAM (Z+q)
	Ενημέρωση σημαιών : Καμία
	Παράδειγμα : Αναλυτικά παρακάτω
	Περίοδοι Ρολογιού : 2

ΠΑΡΑΔΕΙΓΜΑ 1:

Το παρακάτω πρόγραμμα εκτελεί τις ακόλουθες λειτουργίες:

1. Δίνει στους καταχωρητές R16, R17, R18, R19 το ονόματα A,B,C,D αντίστοιχα.
2. Φορτώνει στους παραπάνω καταχωρητές τις απόλυτες τιμές $3E, $0A, $1F, $55.
3. Μεταφέρει τα δεδομένα των A,B,C,D στους R10, R11, R12, R13 αντίστοιχα.
4. Μεταφέρει τα δεδομένα των διευθύνσεων $15 και $16 στους καταχωρητές R14, R15.

5. Χρησιμοποιώντας τον απ' ευθείας τρόπο διευθυνσιοδότησης μεταφέρει τα περιεχόμενα των καταχωρητών R14 και R15 στους καταχωρητές R1 και R2.
6. Μεταφέρει τα περιεχόμενα του R19 στη διεύθυνση $0278

Λύση

```
;Ερώτημα 1
.def A=R16
.def B=R17
.def C=R18
.def D=R19

;Ερώτημα 2
ldi A, $3E
ldi B, $0A
ldi C, $1F
ldi D, $55

;Ερώτημα 3
mov R10, A
mov R11, B
mov R12, C
mov R13, D

;Ερώτημα 4
lds R14, $15
lds R15, $16

;Ερώτημα 5
sts $01, R14
sts $02, R15

;Ερώτημα 6
ldi R27, $02
ldi R26, $78
st X, R19
```

Σημαντικές Παρατηρήσεις:

- Η εντολή άμεσης φόρτωσης **LDI** (**L**oa**D** **I**mmediate) δεν μπορεί να χρησιμοποιηθεί σε συνδυασμό με έναν από τους πρώτους 16 καταχωρητές (R0-R15), μπορεί να χρησιμοποιηθεί μόνο με έναν από τους R16-R32. Έτσι λοιπόν η εντολή **LDI R15 , $16** είναι λάθος.
- Αν και οι καταχωρητές δεν είναι SRAM μνήμες σε φυσικό επίπεδο το γεγονός ότι μοιράζονται το ίδιο address space μας δίνει τη δυνατότητα να έχουμε πρόσβαση στους καταχωρητές μέσω των εντολών απ' ευθείας και έμμεσης διευθυνσιοδότησης. π.χ. η εντολή **STS $03, R24** θα έχει σαν απότέλεσμα τα περιεχόμενα του **R24** να μεταφέρονται στον **R3**.
- Οι εντολές φόρτωσης και ανάγνωσης δεδομένων από την μνήμη με μετατόπιση (LDD,STD) δεν μπορούν να συνδυαστούν με τον καταχωρητή X.

5.2.2 Εντολές αριθμητικών και λογικών πράξεων

5.2.2.1 Εντολές αριθμητικών πράξεων

Οι αριθμητικές πράξεις όπως πρόσθεση και αφαίρεση είναι θεμελιώδης σε πολλά υπολογιστικά συστήματα. Οι περισσότερες γλώσσες προγραμματισμού υποστηρίζουν διάφορους αριθμητικούς τύπους δεδομένων τα οποία χρησιμοποιούνται για την πραγματοποίηση αριθμητικών υπολογισμών.

Για την εκτέλεση αριθμητικών οι μικροελεγκτές AVR διαθέτουν ένα ευρύ φάσμα εντολών. Οι εντολές αυτές έχουν ένα ή δύο ορίσματα (operands) τα οποία είναι πάντοτε οι καταχωρητές γενικού σκοπού R0 έως R31. Δεν υπάρχει δυνατότητα να γίνει απευθείας πράξη μεταξύ του περιεχομένου δύο **θέσεων μνήμης**. Επομένως για να γίνει μία αριθμητική ή λογική πράξη πρέπει πρώτα να μεταφερθούν τα ορίσματα από τις θέσεις μνήμης που είναι αποθηκευμένα να γίνει η πράξη και το αποτέλεσμα να γραφεί σε μία άλλη θέση μνήμης.

Οι εντολές αριθμητικών πράξεων διακρίνονται σε εντολές πρόσθεσης και αφαίρεσης με ή χωρίς κρατούμενο μεταξύ δύο καταχωρητών ή ενός καταχωρητή και μιας σταθεράς ή ζεύγους καταχωρητών και μιας σταθεράς.

5.2.2.1.1 Εντολές πρόσθεσης

Πίνακας 11 – Εντολή ADD

ADD Rd, Rr **Rd← Rd + Rr**	Περιγραφή : Πρόσθεση δύο καταχωρητών χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Παράδειγμα : ADD R4, R5
	Περίοδοι Ρολογιού : 1

Πίνακας 12 – Εντολή ADC

ADC Rd, Rr **Rd← Rd + Rr + C**	Περιγραφή : Πρόσθεση δύο καταχωρητών με χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Παράδειγμα : ADC R5, R6
	Περίοδοι Ρολογιού : 1

Πίνακας 13 – Εντολή ADIW

ADIW Rd, K **Rd+1:Rd←** **Rd+1:Rd + K**	Περιγραφή : Πρόσθεση μιας τιμής K (0-63) σε ένα ζεύγος καταχωρητών και αποθήκευση στο ζεύγος καταχωρητών.
	Ενημέρωση σημαιών : Z, C, N, S, V
	Παράδειγμα : ADDIW R24:R23,5
	Περίοδοι Ρολογιού : 2

5.2.2.1.2 Εντολές αφαίρεσης

Πίνακας 14 – Εντολή SUB

SUB Rd, Rr **Rd← Rd - Rr**	Περιγραφή : Αφαίρεση δύο καταχωρητών χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Παράδειγμα : SUB R4, R5
	Περίοδοι Ρολογιού : 1

Πίνακας 15 – Εντολή SUBI

SUBI Rd, K **Rd← Rd - K**	Περιγραφή : Αφαίρεση μιας τιμής από έναν καταχωρητή χωρίς χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Παράδειγμα : SUBI R4, 15
	Περίοδοι Ρολογιού : 1

Πίνακας 16 – Εντολή SBC

SBC Rd, Rr **Rd←Rd - Rr - C**	Περιγραφή : Αφαίρεση δύο καταχωρητών με χρήση κρατουμένου και αποθήκευση στον καταχωρητή προορισμού Rd.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Παράδειγμα : SBC R4, R5
	Περίοδοι Ρολογιού : 1

Πίνακας 17 – Εντολή SBCI

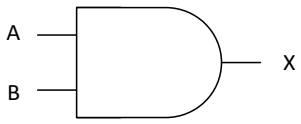
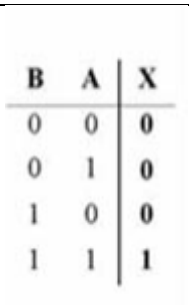
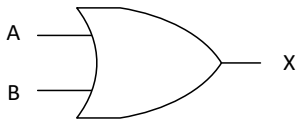
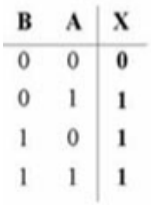
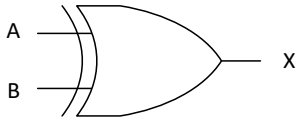
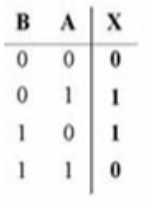
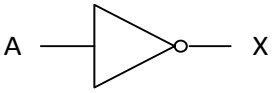
SBCI Rd, K **Rd←Rd - K - C**	Περιγραφή : Αφαίρεση μιας τιμής από έναν καταχωρητή με χρήση κρατουμένου και αποθήκευση στον καταχωρητή.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Παράδειγμα : SBCI R9, 2
	Περίοδοι Ρολογιού : 1

5.2.2.2 Εντολές Λογικών Πράξεων

Μαζί με τις αριθμητικές πράξεις οι αντίστοιχες λογικές πράξεις είναι οι κυριότερες λειτουργίες που μπορούν να εκτελεστούν από έναν μικροελεγκτή και συγκεκριμένα από την ALU (Arithmetic and Logic Unit) μονάδα του.

Κατά συνέπεια οι αντίστοιχες εντολές που υποστηρίζονται αποτελούν ένα ιδιαίτερα σημαντικό υποσύνολο του ρεπερτορίου εντολών κάθε μικροελεγκτή ή μικρο-επεξεργαστή.

Πίνακας 18 – Άλγεβρα Boole, Λογικές Πύλες

Ονομασία	Σύμβολο Πύλης	Σχέση Άλγεβρας Boole	Πίνακας Αλήθειας
AND		$X = A \cdot B$	
OR		$X = A + B$	
XOR		$X = A \oplus B$	
NOT		$X = \overline{A}$	

Στο παραπάνω πίνακα απεικονίζονται οι βασικές πύλες (ή αλλιώς πράξεις άλγεβρας BOOLE) που υποστηρίζονται με συγκεκριμένες εντολές.

Επίσης θυμίζουμε ότι οι πύλες AND, OR, EXOR σε συνδυασμό με την NOT υλοποιούν τις παράγωγες πύλες NAND, NOR, XNOR αντίστοιχα οι οποίες, φυσικά, περιγράφονται με αντίστροφους πίνακες αληθείας σε σχέση με τις βασικές πύλες από τις οποίες προέρχονται.

Οι παράγωγες πύλες δεν υποστηρίζονται από συγκεκριμένες εντολές του ρεπερτορίου εντολών του AVR και κατά συνέπεια όταν πρέπει να παραχθεί το αντίστοιχο αποτέλεσμα στο κώδικάς μας θα πρέπει να χρησιμοποιήσουμε συνδυαστικά τις υπόλοιπες εντολές.

Συγκεκριμένα οι εντολές που υποστηρίζονται είναι οι εξής:

5.2.2.2.1 Λογική Πράξη AND

Πίνακας 19 – Εντολή AND

AND Rd, Rr **Rd← Rd · Rr**	Περιγραφή : Εκτέλεση της λογικής πράξης AND μεταξύ των περιεχομένων των καταχωρητών Rd και Rr και αποθήκευση του αποτελέσματος στο Rd.
	Ενημέρωση σημαιών : Z, N, V
	Περίοδοι Ρολογιού : 1

Πίνακας 20 – Εντολή ANDI

ANDI Rd, k **Rd← Rd · Rr**	Περιγραφή : Εκτέλεση της λογικής πράξης AND μεταξύ του περιεχομένου του καταχωρητή Rd και της ποσότητας k και αποθήκευση του αποτελέσματος στο Rd.
	Ενημέρωση σημαιών : Z, N, V
	Περίοδοι Ρολογιού : 1

Σημαντική Παρατήρηση*: Στην περίπτωση της AND σαν Rd,Rr μπορούν να χρησιμοποιηθούν οποιοιδήποτε καταχωρητές εκ των 32 γενικού σκοπού (R1-R31).Στην περίπτωση της ANDI σαν Rd μπορεί να χρησιμοποιηθεί οποιοσδήποτε εκ των 16 τελευταίων καταχωρητών γενικού σκοπού (R16-R31).*

5.2.2.2.2 Λογική Πράξη OR

Πίνακας 21 - Εντολή OR

OR Rd, Rr **Rd← Rd + Rr**	Περιγραφή : Εκτέλεση της λογικής πράξης OR μεταξύ των περιεχομένων των καταχωρητών Rd και Rr και αποθήκευση του αποτελέσματος στο Rd.
	Ενημέρωση σημαιών : Z, N, V
	Περίοδοι Ρολογιού : 1

Πίνακας 22 - Εντολή ORI

ORI Rd, k **Rd← Rd + Rr**	Περιγραφή : Εκτέλεση της λογικής πράξης OR μεταξύ του περιεχομένου του καταχωρητή Rd και της ποσότητας k και αποθήκευση του αποτελέσματος στο Rd.
	Ενημέρωση σημαιών : Z, N, V
	Περίοδοι Ρολογιού : 1

Σημαντική Παρατήρηση:

Στην περίπτωση της OR σαν Rd, Rr μπορούν να χρησιμοποιηθούν οποιοιδήποτε κατάχωρητές εκ των 32 γενικού σκοπού (R1-R31). Στην περίπτωση της ORI σαν Rd μπορεί να χρησιμοποιηθεί οποιοσδήποτε εκ των 16 τελευταίων καταχωρητών γενικού σκοπού (R16-R31).

5.2.2.2.3 Λογική Πράξη XOR

Πίνακας 23 – Εντολή EOR

EOR Rd, Rr **Rd←Rd⊕Rr**	Περιγραφή : Εκτέλεση της λογικής πράξης XOR μεταξύ των περιεχομένων των καταχωρητών Rd, Rr και αποθήκευση του αποτελέσματος στο Rd.
	Ενημέρωση σημαιών : Z, N, V
	Περίοδοι Ρολογιού : 1

Σημαντική Παρατήρηση: Σαν Rd, Rr μπορούν να χρησιμοποιηθούν οποιοιδήποτε καταχωρητές εκ των 32 γενικού σκοπού (R1-R31).

5.2.2.2.4 Λογική Πράξη NOT

Πίνακας 24 - Εντολή COM

COM Rd **Rd← Rd'**	Περιγραφή : Υπολογίζει τον αρνητικό αριθμό **Rd** σε μορφή συμπληρώματος του 1. Αυτό γίνεται αντιστρέφοντας όλα τα bit της ποσότητας που είναι αποθηκευμένη στο **Rd**. Γίνεται λοιπόν κατανοητό ότι η ενέργεια αυτή είναι ισοδύναμη με την εφαρμογή της πράξης **NOT** στην ποσότητα του **Rd.**
	Ενημέρωση σημαιών : Z, C, N, V
	Περίοδοι Ρολογιού : 1

Σημαντική Παρατήρηση: Σαν Rd μπορεί να χρησιμοποιηθεί οποιοσδήποτε καταχωρητής εκ των 32 γενικού σκοπού (R1-R31).

Μια ιδιαίτερα σημαντική χρήση των εντολών αυτών είναι ότι βασιζόμενοι στους πίνακες αληθείας των λογικών πυλών οι αντίστοιχες εντολές μπορούν να χρησιμοποιηθούν για να θέσουν ή να καθαρίσουν συγκεκριμένα bit σε μια 8 bit ποσότητα. Η συγκεκριμένη λειτουργία συναντάται και ως «εφαρμογή μάσκας» σε μια αριθμητική ποσότητα. Κάποιοι σχετικοί κανόνες που μπορούν εύκολα να εξαχθούν από τους πίνακες αληθείας είναι:

- ✓ Όταν ένα από τα δύο ορίσματα σε μία πράξη AND είναι 0 τότε το αποτέλεσμα είναι πάντα 0 ανεξαρτήτως του δεύτερου ορίσματος.
- ✓ Όταν ένα από τα δύο ορίσματα σε μία πράξη AND είναι 1 τότε το αποτέλεσμα είναι πάντα η τιμή του δεύτερου ορίσματος.
- ✓ Όταν ένα από τα δύο ορίσματα σε μία πράξη OR είναι 1 τότε το αποτέλεσμα είναι πάντα 1 ανεξαρτήτως του δεύτερου ορίσματος.
- ✓ Όταν ένα από τα δύο ορίσματα σε μία πράξη OR είναι 0 τότε το αποτέλεσμα είναι πάντα η τιμή του δεύτερου ορίσματος.
- ✓ Το αποτέλεσμα μιας πράξης XOR μεταξύ της ίδια ποσότητας είναι πάντα 0.

Επιπλέον πολύ σημαντικές και ευρέως χρησιμοποιούμενες είναι οι εντολές αύξησης και μείωσης των περιεχομένων των καταχωρητών. Αυτές είναι :

Πίνακας 25 – Εντολή INC

INC Rd **Rd← Rd + 1**	Περιγραφή : Αυξάνει τα περιεχόμενα του καταχωρητή Rd κατά 1.
	Ενημέρωση σημαιών : Z, N, V
	Περίοδοι Ρολογιού : 1

Πίνακας 26 – Εντολή DEC

DEC Rd **Rd← Rd - 1**	Περιγραφή : Μειώνει τα περιεχόμενα του καταχωρητή Rd κατά 1
	Ενημέρωση σημαιών : Z, N, V
	Περίοδοι Ρολογιού : 1

5.2.3 Εντολές σε επίπεδο bit και ελέγχου bit

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

Πίνακας 27 – Εντολή SBI

SBI P, b **I/O(P, b)← 1**	Περιγραφή : Ορίζουμε το bit στον καταχωρητή I/O (θέτει την τιμή 1).
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 2

Πίνακας 28 – Εντολή CBI

CBI P, b **I/O(P, b)← 0**	Περιγραφή : Μηδενίζει το bit στον καταχωρητή I/O.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 2

Πίνακας 29 – Εντολή BSET

BSET s **SREG(s)← 1**	Περιγραφή : Ορίζει τα flags του καταχωρητή κατάστασης SREG.
	Ενημέρωση σημαιών : SREG(s)
	Περίοδοι Ρολογιού : 1

Πίνακας 30 – Εντολή BCLR

BCLR s **SREG(s)← 0**	Περιγραφή : Μηζενίζει τα flags του καταχωρητή κατάστασης SREG.
	Ενημέρωση σημαιών : SREG(s)
	Περίοδοι Ρολογιού : 1

Πίνακας 31 – Εντολή SEI

SEI **SREG(s)← 1**	Περιγραφή : Ενεργοποιεί το διακόπτη της καθολικής διακοπής (GlobalInterrupt).
	Ενημέρωση σημαιών : I
	Περίοδοι Ρολογιού : 1

Πίνακας 32 – Εντολή CLF

CLF **SREG(s)←1**	Περιγραφή : Απενεργοποιεί το διακόπτη της καθολικής διακοπής (GlobalInterrupt).
	Ενημέρωση σημαιών : I
	Περίοδοι Ρολογιού : 1

5.2.4 Εντολές ελέγχου ροής του προγράμματος και διακλάδωσης

Οι εντολές διακλάδωσης χρησιμεύουν στη δημιουργία διακλαδώσεων μέσα σ' ένα πρόγραμμα, αν θέλουμε να παρακάμψουμε μια σειρά από εντολές όταν ικανοποιείται ή όταν δεν ικανοποιείται κάποια συνθήκη. Επίσης χρησιμεύουν στις διαδικασίες ανακυκλώσεων και επανεκτελέσεων τμημάτων του προγράμματος. Στις εντολές διακλάδωσης ανήκουν και οι εντολές κλήσης και επιστροφής από υπορουτίνα. Μερικές από τις εντολές αυτές δεν έχουν όρισμα. Όσες όμως έχουν, έχουν σαν όρισμα την ετικέτα της εντολής, στην οποία θα γίνει η διακλάδωση. Οι εντολές διακλάδωσης αλλάζουν την ροή του προγράμματος αντικαθιστώντας τα περιεχόμενα του PC (Program Counter) με την διεύθυνση της εντολής που θα γίνει η διακλάδωση. Οι εντολές διακλάδωσης μπορούν να χωριστούν σε δύο κατηγορίες:

- Εντολές διακλάδωσης χωρίς συνθήκη.
- Εντολές διακλάδωσης υπό συνθήκη.

5.2.4.1 Διακλάδωση χωρίς συνθήκη

Με τις εντολές αυτές μπορεί να γίνει «άνευ όρων» διακλάδωση, δηλαδή διακλάδωση χωρίς συνθήκη. Παρακάτω αναφέρονται κάποιες από τις εντολές μας παρέχουν τη δυνατότητα να εκτελέσουμε άλμα σε επιθυμητή διεύθυνση, να καλέσουμε μια ρουτίνα και να επιστρέψουμε από αυτήν.

Πίνακας 33 – Εντολή RJMP

RJMP k **PC ←PC + k + 1**	Περιγραφή : Relative Jump. Σχετικό «άλμα» σε μια διεύθυνση της μνήμης προγράμματος μεταξύ PC-2k +1 και PC+2k. Το k είναι μια ετικέτα (label) στην αριστερή στήλη του προγράμματος. Σαν ετικέτα μπορούμε να βάλουμε ότι θέλουμε αρκεί να μην είναι όνομα εντολής, να μην έχει κενά και να είναι σε λατινικούς χαρακτήρες.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 2

Πίνακας 34 – Εντολή JMP

JMP k **PC←k**	Περιγραφή : Direct Jump. Απλό «άλμα» σε μια θέση διεύθυνση της μνήμης προγράμματος.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 3

5.2.4.1.1 Εντολές Κλίσης Υπορουτινών

Με τις εντολές αυτές μπορεί να γίνει κλήση σε μία ρουτίνα.

Πίνακας 35 – Εντολή CALL

CALL k **PC←k**	Περιγραφή : Direct Subroutine Call. Κλήση υπορουτίνας εντός της μνήμης προγράμματος. Η διεύθυνση επιστροφής αποθηκεύεται στη στοίβα.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 4/5

Πίνακας 36 – Εντολή RET

RET **PC←Stack**	Περιγραφή : Subroutine Return. Επιστροφή από υπορουτίνα. Η διεύθυνση επιστροφής φορτώνεται από τη στοίβα.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 4/5

Πίνακας 37 – Εντολή RCALL

RCALL k **PC ← PC + k + 1**	Περιγραφή : Relative Subroutine Call. Σχετική κλήση ρουτίνας σε μια διεύθυνση μεταξύ PC+2k+1 και PC+2k. Η διεύθυνση επιστροφής αποθηκεύεται στη στοίβα.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 3/4

5.2.4.2 Διακλάδωση υπό συνθήκη

5.2.4.2.1 Διακλαδώσεις που γίνεται έλεγχος byte

Οι εντολές με τις οποίες γίνεται «υπό συνθήκη» διακλάδωση μπορούν να ελέγχουν είτε ένα bit, είτε ένα byte αν ικανοποιεί κάποια συνθήκη που όταν αληθεύει θα λάβει χώρα το άλμα. Στην ουσία εξετάζουν μία από τις επτά σημαίες του καταχωρητή κατάστασης (SREG) για ν' αποφασίσουν αν πρέπει να γίνει το άλμα ή όχι. Οι σημαίες αυτές επηρεάζονται από τις πράξεις αλλά και από κάποιες άλλες εντολές που εξετάζονται παρακάτω.
Δεν θα εξετάσουμε όλες τις εντολές διακλάδωσης «υπό συνθήκη», αλλά μόνο αυτές που χρησιμοποιούνται πιο συχνά:

Πίνακας 38 – Εντολή CP

CP Rd, Rr **Rd - Rr**	Περιγραφή : Compare. Σύγκριση με υπολογισμό της διαφοράς χωρίς χρήση κρατουμένου μεταξύ των δύο καταχωρητών.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Περίοδοι Ρολογιού : 1

Πίνακας 39 – Εντολή CPC

CPC Rd, Rr **Rd - Rr**	Περιγραφή : Compare with Carry. Σύγκριση με υπολογισμό της διαφοράς με χρήση κρατουμένου μεταξύ των δύο καταχωρητών.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Περίοδοι Ρολογιού : 1

Πίνακας 40 – Εντολή CPI

CPI Rd, k **Rd - k**	Περιγραφή : Compare Register with Immediate. Σύγκριση με υπολογισμό της διαφοράς με χρήση κρατουμένου μεταξύ του καταχωρητών Rd και μιας σταθεράς.
	Ενημέρωση σημαιών : Z, C, N, S, V, H
	Περίοδοι Ρολογιού : 1

Πίνακας 41 – Εντολή CPSE

CPSE Rd, Rr **Αν Rd = Rr τότε** **PC←PC+2 (ή 3)** **Αλλιώς PC←PC+1**	Περιγραφή : Compare, Skip if Equal. Σύγκριση μεταξύ των καταχωρητών Rd και Rr. Σε περίπτωση ισότητας παρακάμπτεται η επόμενη εντολή.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 1/2/3

Οι εντολές CP, CPI χρησιμοποιούνται απλά για να κάνουν μία σύγκριση. Για να εκμεταλλευτούμε το αποτέλεσμα αυτής της σύγκρισης χρησιμοποιούμε τις εντολές “BRANCH” : BRXX k : όπου XX μπορεί να είναι EQ, LO, NE, GE κλπ.

Αφού έχουν συγκριθεί δύο καταχωρητές (registers), ανάλογα με την κατάσταση που έχουν οι σημαίες (flag) στον SREG γίνεται διακλάδωση στο ανάλογο label (στην διεύθυνση PC+k+1 της μνήμης flash με τιμές του k από -64 έως +63 (-64<k<+63).

Πίνακας 42 – Εντολή BRNE

BRNE k **Αν Rd <> Rr (Z=0)** **τότε PC←PC+k+1** **Αλλιώς PC←PC+1**	Περιγραφή : Branch if not Equal. Σχετικό άλμα σε περίπτωση ανισότητας. Όταν η σημαία μηδενός Z ισούται με 0 (Z=0) τότε εκτελείται σχετική διακλάδωση.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : ½

Πίνακας 43 – Εντολή BREQ

BREQ k **Αν Rd = Rr τότε** **PC←PC+k+1** **Αλλιώς PC←PC+1**	Περιγραφή : Branch if Equal. Σχετικό άλμα σε περίπτωση ισότητας.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : ½

Πίνακας 44 – Εντολή BRGE

BRGE k **Αν Rd >= Rr** **(S=N⊕V=0) τότε** **PC←PC+k+1**	Περιγραφή : Branch if Greater or Equal, Signed. Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση προσημασμένου δώσει μεγαλύτερο ή ίσο (S=0) τότε εκτελείται σχετική διακλάδωση.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : ½

Πίνακας 45 – Εντολή BRLO

BRLO k **Αν Rd < Rr τότε** **PC←PC+k+1** **Αλλιώς PC←PC+1**	Περιγραφή : Branch if Lower. Σχετικό άλμα υπό συνθήκη. Όταν η σύγκριση προσημασμένου δώσει μικρότερο (S=0) τότε εκτελείται σχετική διακλάδωση.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : ½

5.2.4.2.2 Διακλαδώσεις που γίνεται έλεγχος bit

Στην ουσία όλες οι παραπάνω εντολές Branch εξετάζουν κάποιο bit στον SREG και ανάλογα μεταβαίνει ο PC. Όμως υπάρχει η δυνατότητα να εξεταστεί ένα bit και να πραγματοποιηθεί διακλάδωση χωρίς τη μεσολάβηση των εντολών CP, CPI κλπ. Γίνεται έλεγχος bit κατ' ευθείαν σε κάποιο από τα 8 bit ενός register ή μιας I/O πόρτας.
Χρησιμοποιείται η εντολή Skip if Bit in (είτε Register είτε I/O) is (είτε Set είτε Clear)

Πίνακας 46 – Εντολή SBRS

SBRS Rr, b **Rr, b**	Περιγραφή : Skip if Bit in Register is set. Αγνοείται η επόμενη εντολή αν το bit b με 0 <b< 7 σε έναν καταχωρητή Rr είναι σε λογικό 1.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 1/2/3

Πίνακας 47 – Εντολή SBRC

SBRC Rr, b **Rr, b**	Περιγραφή : Skip if Bit in Register Cleared. Αγνοείται η επόμενη εντολή αν το bit b με 0 <b< 7 σε έναν καταχωρητή Rr είναι σε λογικό 0.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 1/2/3

Πίνακας 48 – Εντολή SBIS

SBIS A, b **Αν I/O (A,b)=1** **τότε PC←PC+2 (ή 3) αλλιώς** **PC←PC+1**	Περιγραφή : Skip if Bit in I/O Register is Set. Αγνοείται η επόμενη εντολή αν το bit b με 0 <b< 7 σε μία I/O θέση μνήμης A είναι σε λογικό 1.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 1/2/3

Πίνακας 49 – Εντολή SBIC

SBIC A, b **Αν Rd<Rr τότε** **PC←PC+k+1** **αλλιώς PC←PC+1**	Περιγραφή : Skip if Bit in I/O Register Cleard. Αγνοείται η επόμενη εντολή αν το bit b με 0 <b< 7 σε μία I/O θέση μνήμης A είναι σε λογικό 0.
	Ενημέρωση σημαιών : καμία
	Περίοδοι Ρολογιού : 1/2/3

6 Περιβάλλον προγραμματισμού(AtmelStudio)

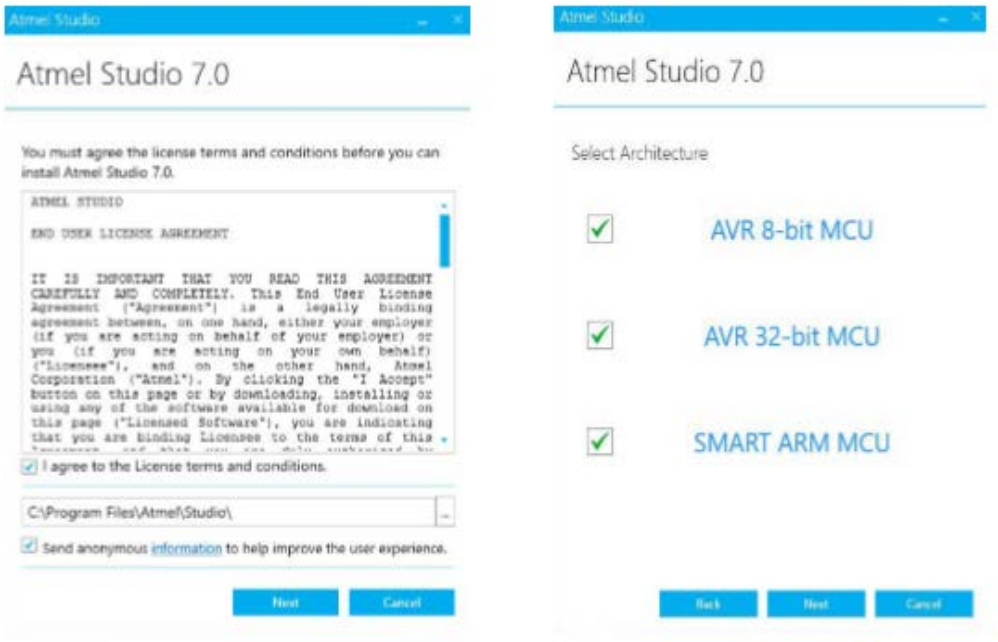
6.1 Εγκατάσταση του Λογισμικού Atmel Studio 7

Για την ανάπτυξη μιας εφαρμογής στον μικροελεγκτή είναι απαραίτητος ο σχεδιασμός και η υλοποίηση του πηγαίου κώδικα. Αυτό μπορεί να γίνει με την χρήση του λογισμικού Atmel Studio 7 αφού πρώτα το εγκαταστήσουμε στον Η/Υ μας. Το λογισμικό αυτό είναι διαθέσιμο από τον δικτυακό τόπο της εταιρίας ATMEL, http://www.atmel.com/products/avr/ (επιλογή Tools, Software Tools, Amtel Studio,

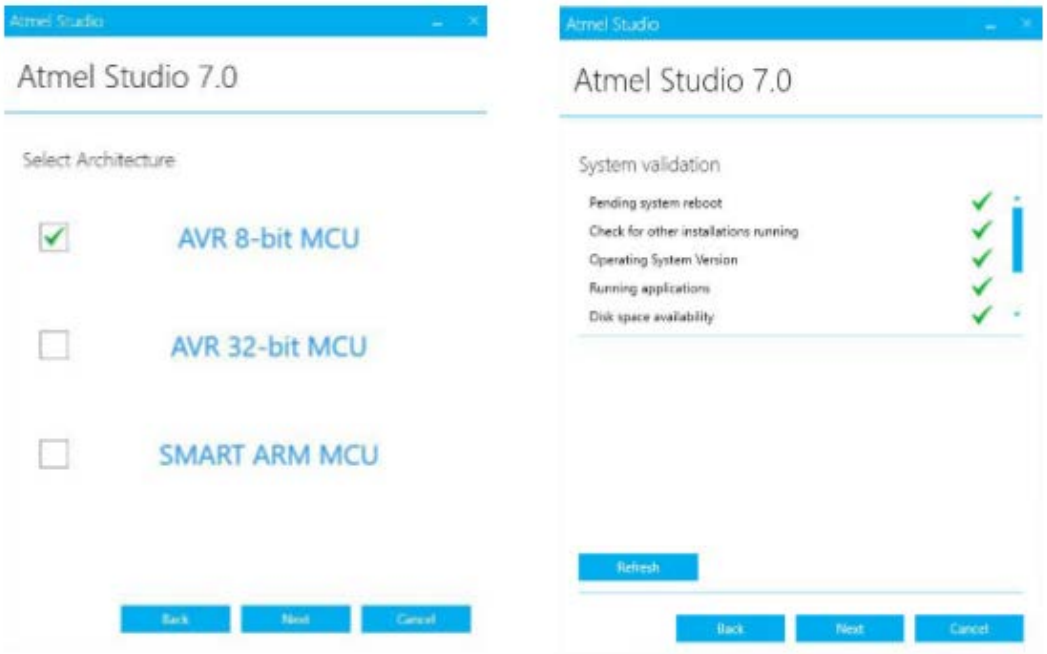
Εικόνα 6-1 – Αρχική οθόνη εγκατάστασης

Amtel Studio 7). Αφού κατεβάσουμε το λογισμικό Amtel Studio 7, τρέχουμε το αρχείο "as-installer-7.0.790-full.exe" εμφανίζεται η παραπάνω εικόνα (Εικόνα 6.1), και ξεκινάει η εγκατάσταση.

Παρακάτω ακολουθούν όλα τα βήματα (όλες οι φόρμες που παρουσιάζονται) κατά την διαδικασία εγκατάστασης του προγράμματος (Εικόνες 6.2 – 6.8). Ακολουθούμε τα βήματα ένα-ένα μέχρι να καταλήξουμε στην εγκατάσταση του λογισμικού.

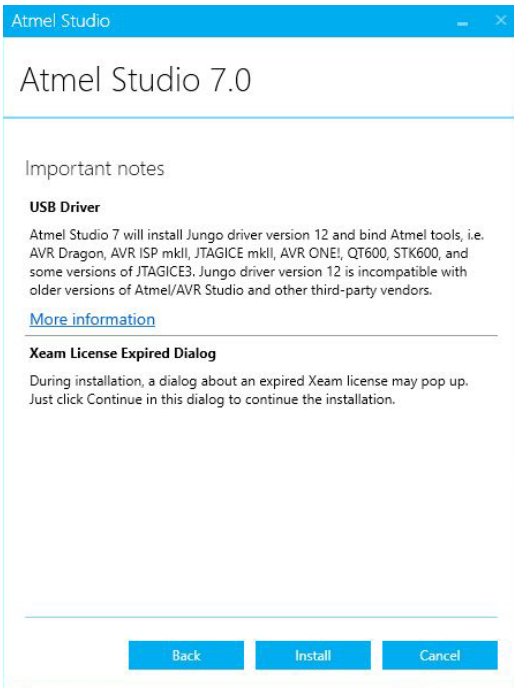


Εικόνα 6-2 - Λεπτομέρειες Εγκατάστασης

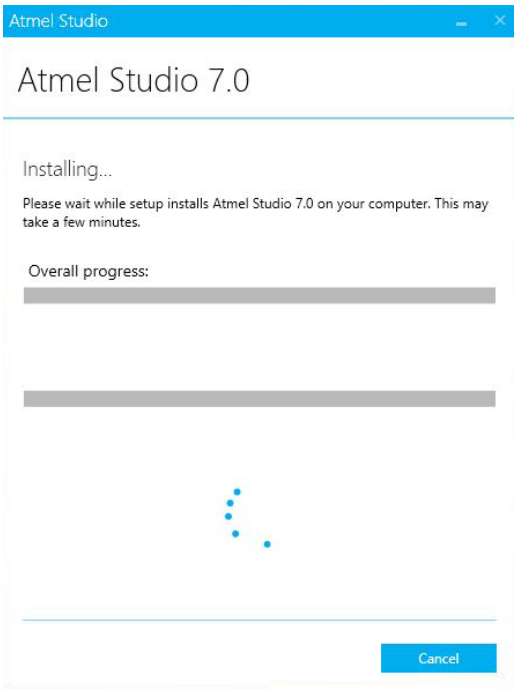


Εικόνα 6-3 – Λεπτομέρειες Εγκατάστασης

Επιλέγουμε ποιο μικροελεγκτή έχουμε (8-bit, 32-bit ή Smart ARM) αποτσεκάροντας αυτούς που δεν χρειαζόμαστε (εικόνα 6.3). Στο επόμενο βήμα μας εμφανίζει μια



Εικόνα 6-4 – Λεπτομέρεια Εγκατάστασης



Εικόνα 6-5 – Λεπτομέρεια Εγκατάστασης

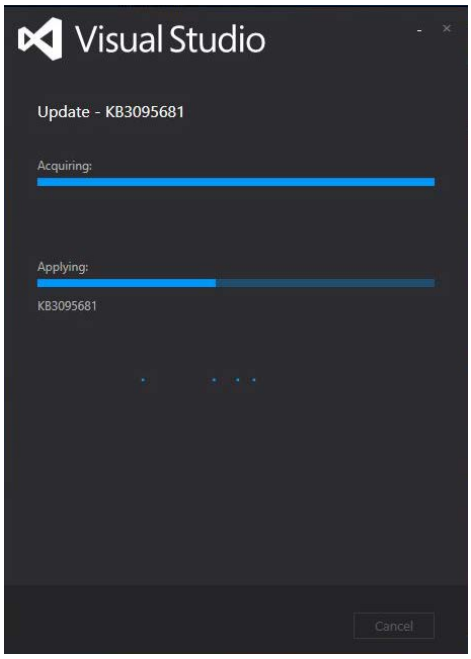
οθόνη που επιβεβαιώνει αυτά που ζητήθηκαν να γίνουν.

Αφού συμφωνούμε πατάμε το “Next”.

Στην εικόνα 6.4 επιλέγουμε να εγκαταστήσουμε το usb driver και πατάμε “install” και συνεχίζουμε την εγκατάσταση.

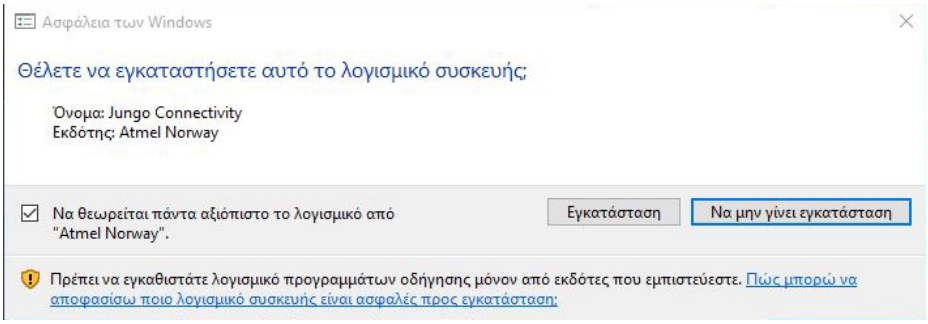
Συνεχίζοντας γίνεται η εγκατάσταση του “Visual Studio” (εικόνα 6.6)

Στην εικόνα 6.7 (ασφάλεια των windows) επιλέγουμε «εγκατάσταση» και συνεχίζεται η εγκατάσταση ως το τέλος (εικόνα 6.8). Στο σημείο αυτό θα πρέπει να αναφερθεί ότι η παρακάτω οθόνη (εικόνα



Εικόνα 6-6 - Εγκατάσταση Visual Studio

6.7) εμφανίζεται μόνο σε windows Vista και νεότερα και εφόσον ο χρήστης δεν έχει

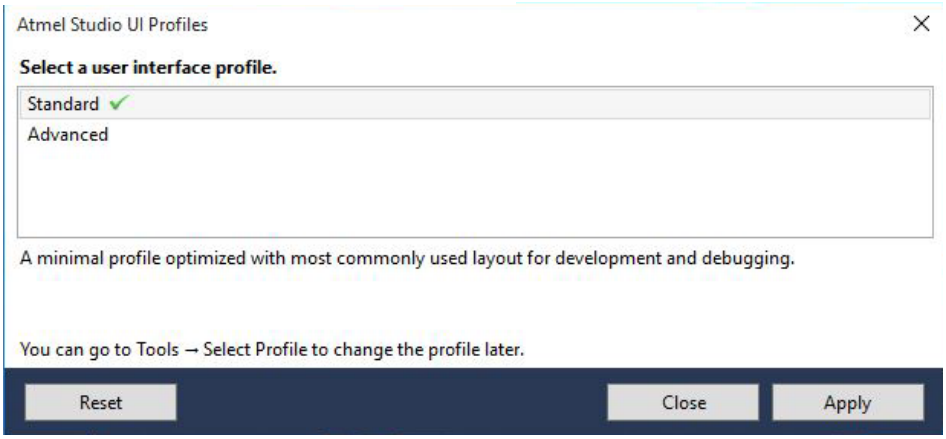


Εικόνα 6-7 – Εγκατάσταση Driver

απενεργοποιήσει το GUI (Graphic User Interface).

Εικόνα 6-8 – Τέλος Εγκατάστασης

Την πρώτη φορά που θα ανοίξουμε το πρόγραμμα μας ζητάει να προκαθορίσουμε το είδος προφίλ που θα χρησιμοποιήσουμε – αφήνουμε το προεπιλεγμένο (εικόνα 6.9), το οποίο και μπορεί να αλλάξει αργότερα.

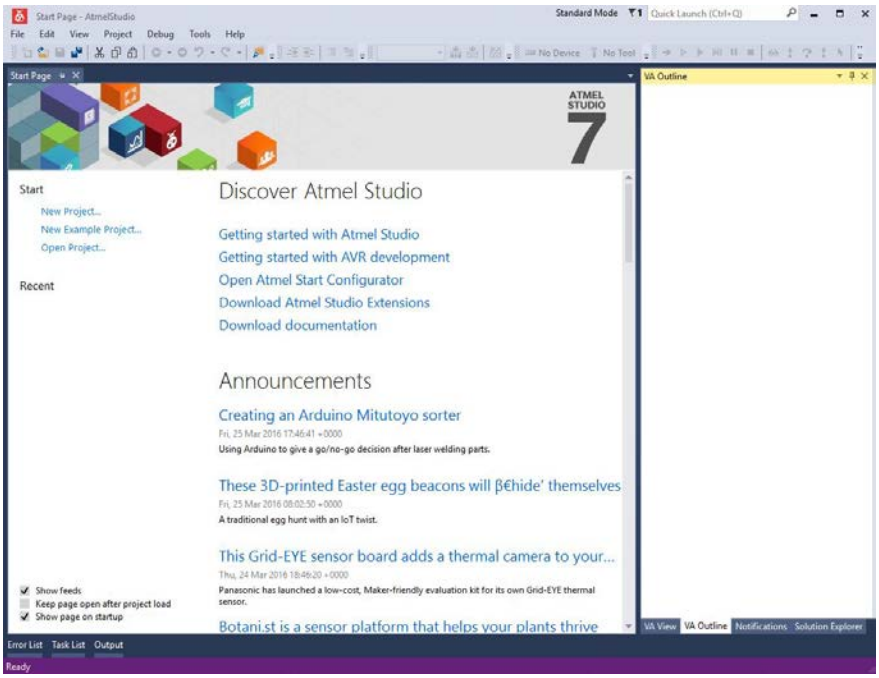


Εικόνα 6-9 – Επιλογή Profil

6.2 Περιβάλλον του Λογισμικού Atmel Studio 7

Παρακάτω αναφέρεται βήμα προς βήμα η διαδικασία εκείνη που πρέπει να ακολουθηθεί για την συγγραφή και εκτέλεση ενός προγράμματος για τον AVR.

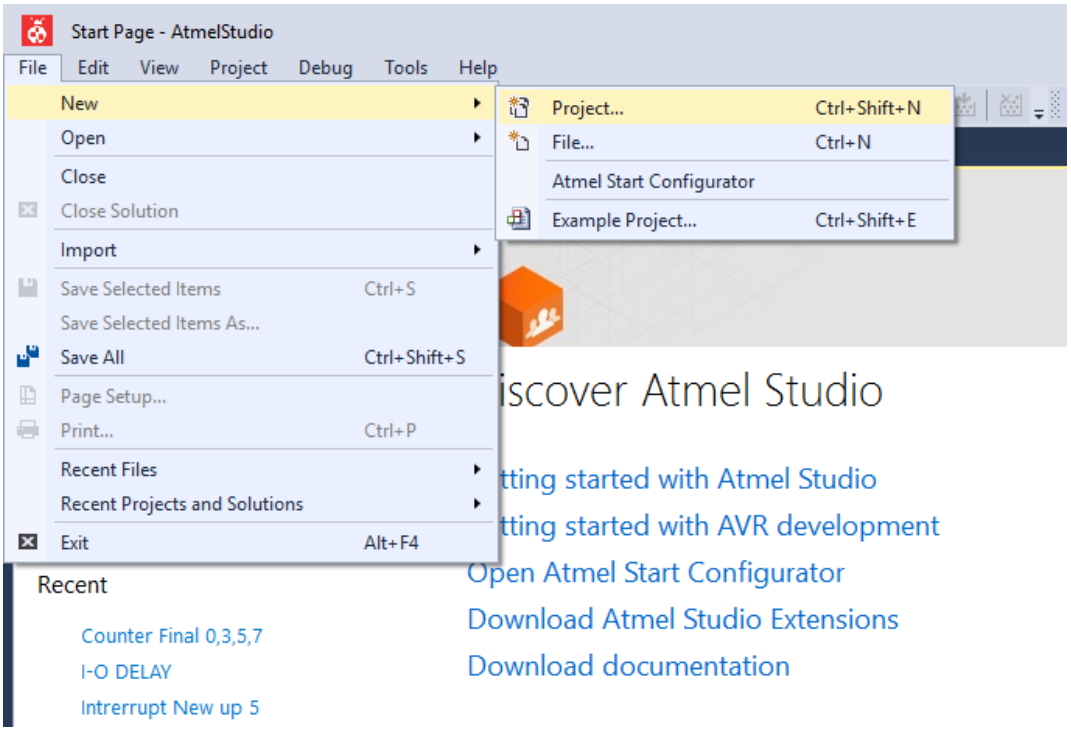
6.2.1 Δημιουργία νέου έργου (project)/Ορισμόςπαραμέτρων



Εικόνα 6-10 – New Project

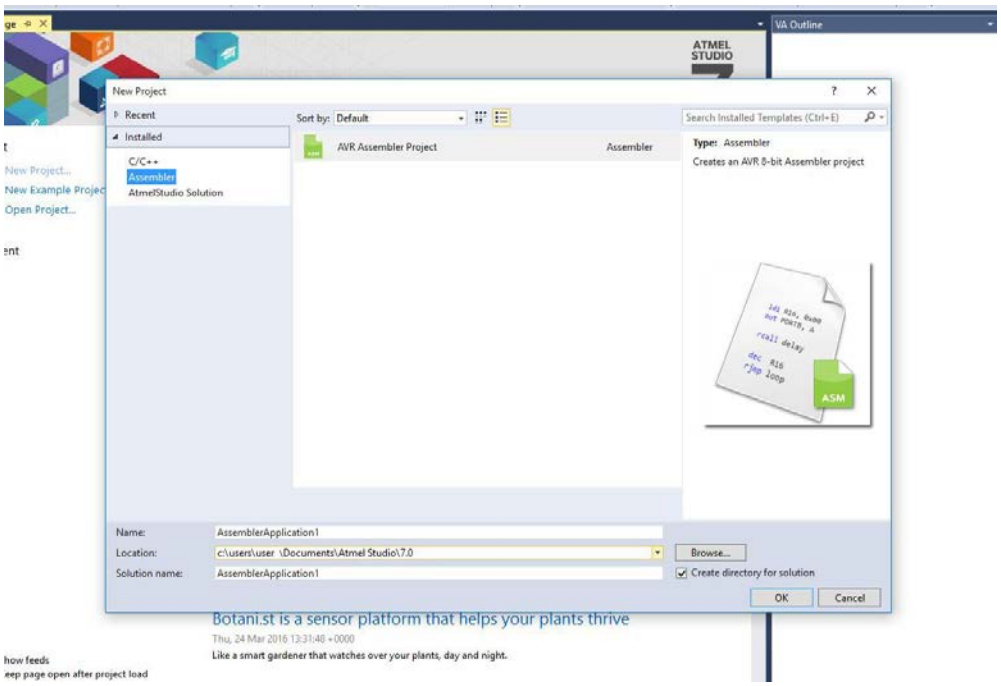
Η φιλοσοφία λειτουργίας του λογισμικού Atmel Studio είναι ότι κάθε πρόγραμμα αποθηκεύεται σε ένα φάκελο μαζί με όλα τα αρχεία που προκύπτουν κατά την μεταγλώττιση του. Στις εικόνες 6.11 και 6.12 βλέπουμε τη διαδικασία δημιουργίας Νέου Έργου (New Project). Μέσω της εικόνας 6.10, μπορούμε να επιλέξουμε κατευθείαν την επιλογή *“New Project”.* Στο παράθυρο αυτό μπορούμε δηλαδή να δημιουργήσουμε ένα νέο έργο ή να επιλέξουμε να δουλέψουμε ένα έργο που ήδη υπάρχει με την επιλογή “Open Project”. Στην περίπτωση μας, αφού θέλουμε να δημιουργήσουμε ένα νέο έργο, θα επιλέξουμε την επιλογή *“New Project”*.

Επιλέγουμε το "New Project" ώστε να εμφανιστεί η οθόνη επιλογών του project . Ή



Εικόνα 6-11 - Menu

εναλλακτικά μπορούμε να πάμε από το μενού *"File"* - *"New"* – *"Project"* (εικόνα 6.11).



Εικόνα 6-12 - Assembler

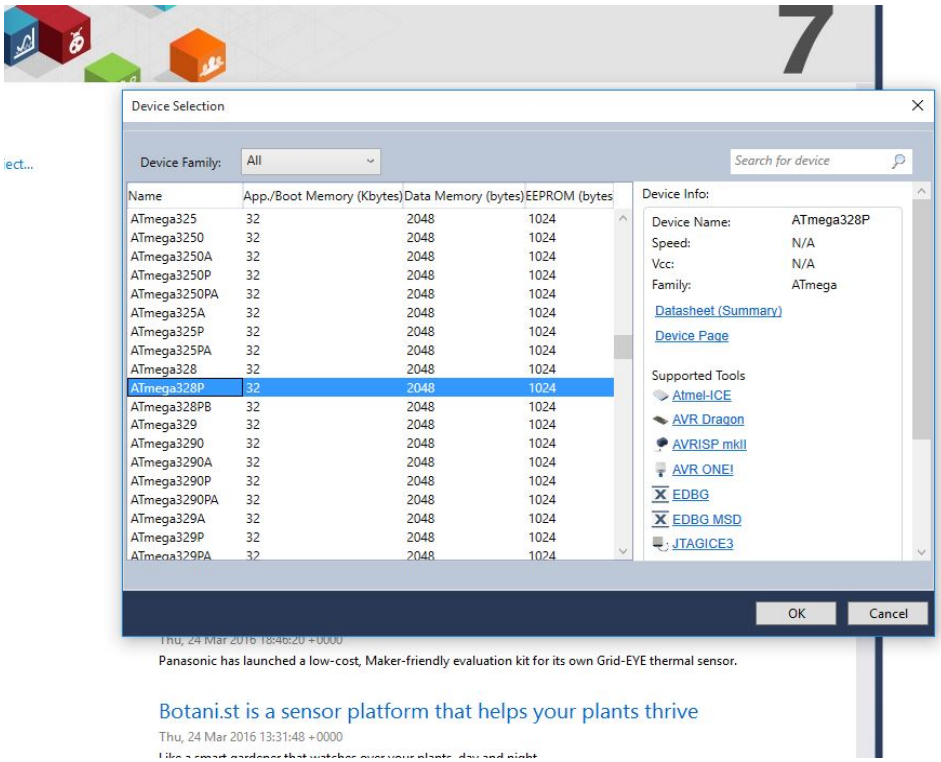
Αφού επιλέξουμε την δημιουργία Νέου Έργου (New Project), ύστερα θα πρέπει να ορίσουμε τις παραμέτρους του έργου (project) στο οποίο θα δουλέψουμε. Στην οθόνη (Εικόνα 6.12) που θα εμφανιστεί θα πρέπει να ορίσουμε:

➔ *Τον τύπο του έργου (Project Type)*

Σε αυτό το σημείο θα πρέπει να ορίσουμε το είδος του έργου (project type) που θέλουμε να δημιουργήσουμε. Στην περίπτωση μας θα επιλέγουμε πάντα τύπο έργου για συμβολομεταφραστή, δηλαδή *"**Assembler**"*. (Εικόνα 6.12)

- *Το όνομα του έργου (Project Name)*

Σε αυτό το σημείο θα πρέπει να ορίσουμε το όνομα του έργου (project name) που δημιουργήσαμε.

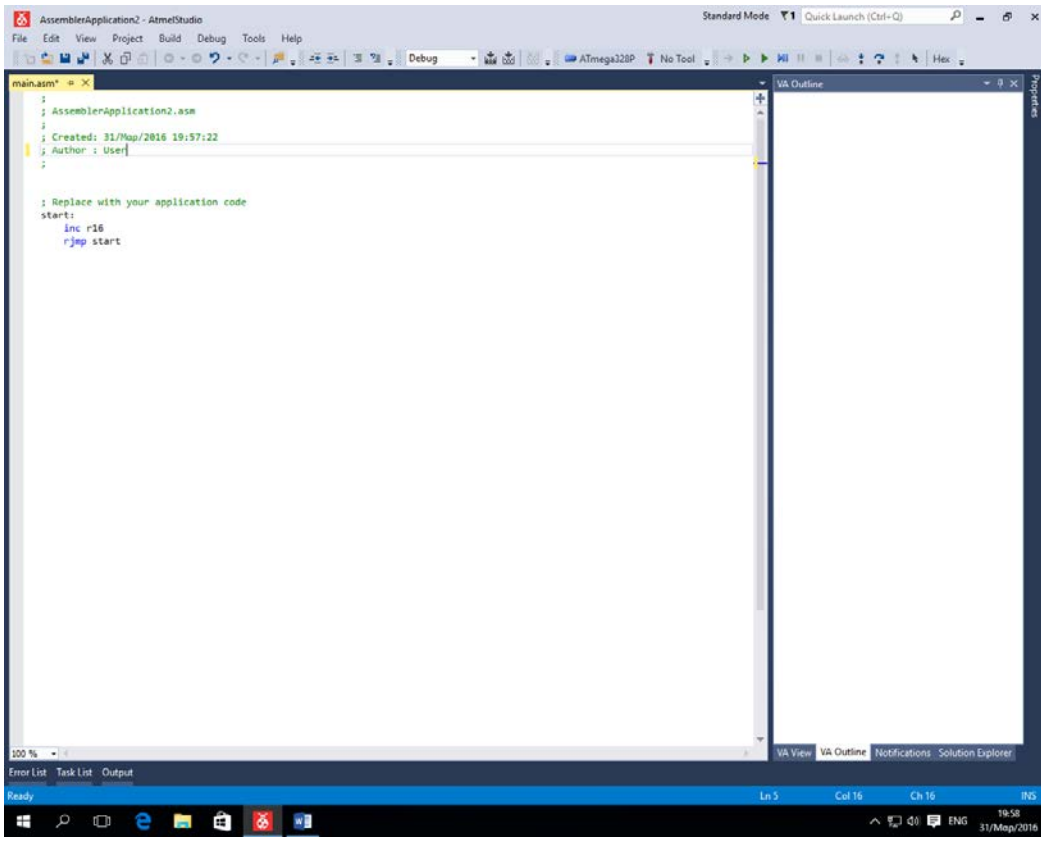


Εικόνα 6-13 – Device Selection

- Ορισμός άλλων παραμέτρων

Αφού συμπληρώσουμε σωστά όλα τα στοιχεία μπορούμε να πατήσουμε την επιλογή "Next" για να εμφανιστεί η οθόνη επιλογών μοντέλου μικροελεγκτή (Device

Selection) (Εικόνα 6.13). Το Atmel Studio 7 μπορεί να υποστηρίξει και να χρησιμοποιηθεί από ένα ευρύ φάσμα εργαλείων αποσφαλμάτωσης (debug).

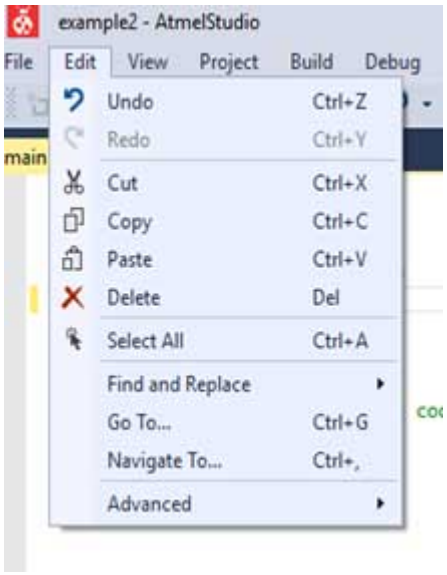


Εικόνα 6-14 – Οθόνη εργασίας

Αφού έχουμε ξανά ελέγξει όλες τις παραπάνω επιλογές, ύστερα πατάμε την επιλογή "Ok" και δημιουργείται το έργο (project). Εμφανίζεται η οθόνη εργασίας του Atmel Studio 7 όπου μπορούμε να γράψουμε το πρόγραμμα για το συμβολομεταφραστή (εικόνα 6.14).

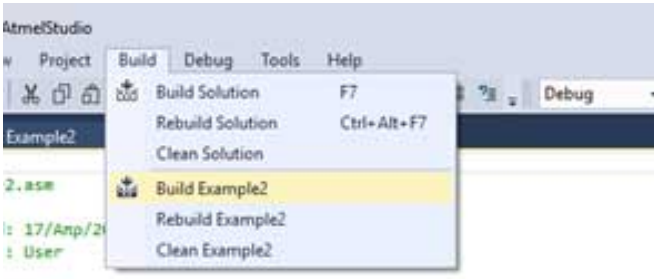
6.2.1.1 Εκτέλεση προγράμματος

Η γραμμή μενού περιλαμβάνει ορισμένες επιλογές. Η επιλογή File έχει αναφερθεί παραπάνω. Η επιλογή Edit περιλαμβάνει λειτουργίες επεξεργασίας του κειμένου (cut, copy, paste, delete, undo, redo κ.ά) όπως φαίνεται στην εικόνα 6.15.



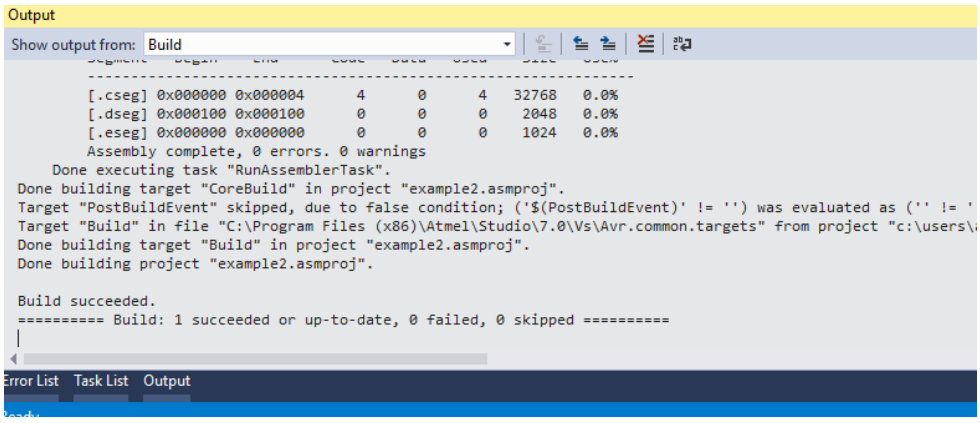
Εικόνα 6-15 – Μενού Edit

Αφού γράψουμε το πρόγραμμα στον editor του Atmel Studio, από την επιλογή *Build* (εικόνα 6.16) επιλέγουμε την εντολή Build Example2, για να δημιουργηθεί το εκτελέσιμο αρχείο του προγράμματός μας.



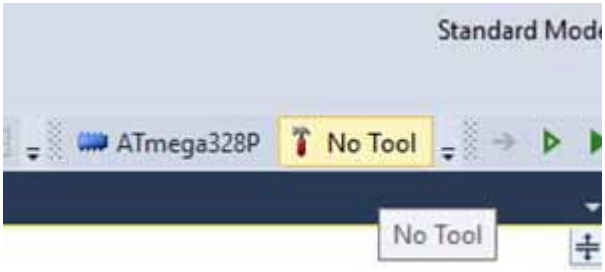
Εικόνα 6-16 – Μενού Build

Αν ο κώδικάς μας έχει συντακτικά σφάλματα, αυτά θα φανούν στο παράθυρο εξόδου του μεταγλωττιστή, όπως φαίνεται στην εικόνα 6.17.



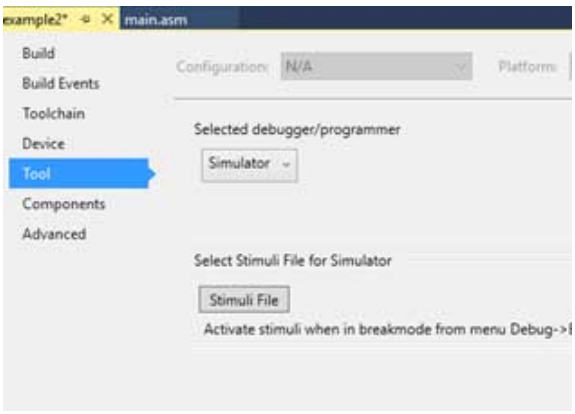
Εικόνα 6-17 – Output Window

Αφού έχει δημιουργηθεί το εκτελέσιμο αρχείο του προγράμματος, μπορούμε να τρέξουμε το πρόγραμμά μας. Πριν από αυτό, όμως, θα πρέπει να επιλέξουμε τον προσομοιωτή (simulator), ώστε να μπορέσουμε να ελέγξουμε τα αποτελέσματα του προγράμματος.



Εικόνα 6-18 - Tool

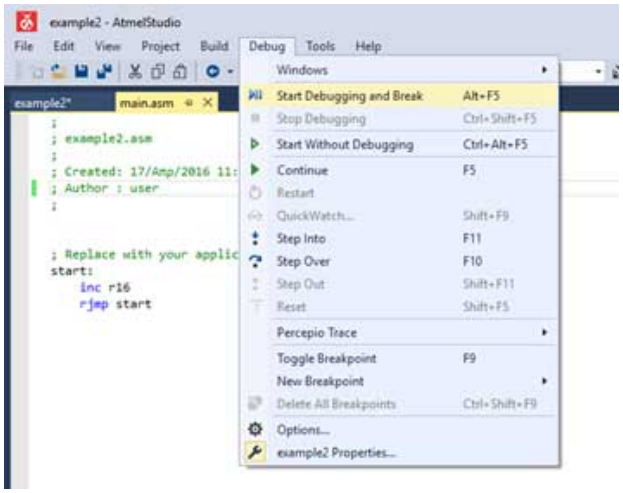
Ακριβώς πάνω από τον editor, πατάμε το No Tool (εικόνα 6.18) και μας εμφανίζει το παράθυρο για την επιλογή του προσομοιωτή (simulator) (εικόνα 6.19) ή τον προγραμματιστή που χρησιμοποιούμε(π.χ AVRISP mkII).



Εικόνα 6-19 - Επιλογή Simulator

Αφού επιλέξουμε το Simulator, επιστρέφουμε στο αρχείο main.asm. Πλέον, είμαστε έτοιμοι, για να τρέξουμε το πρόγραμμά μας.

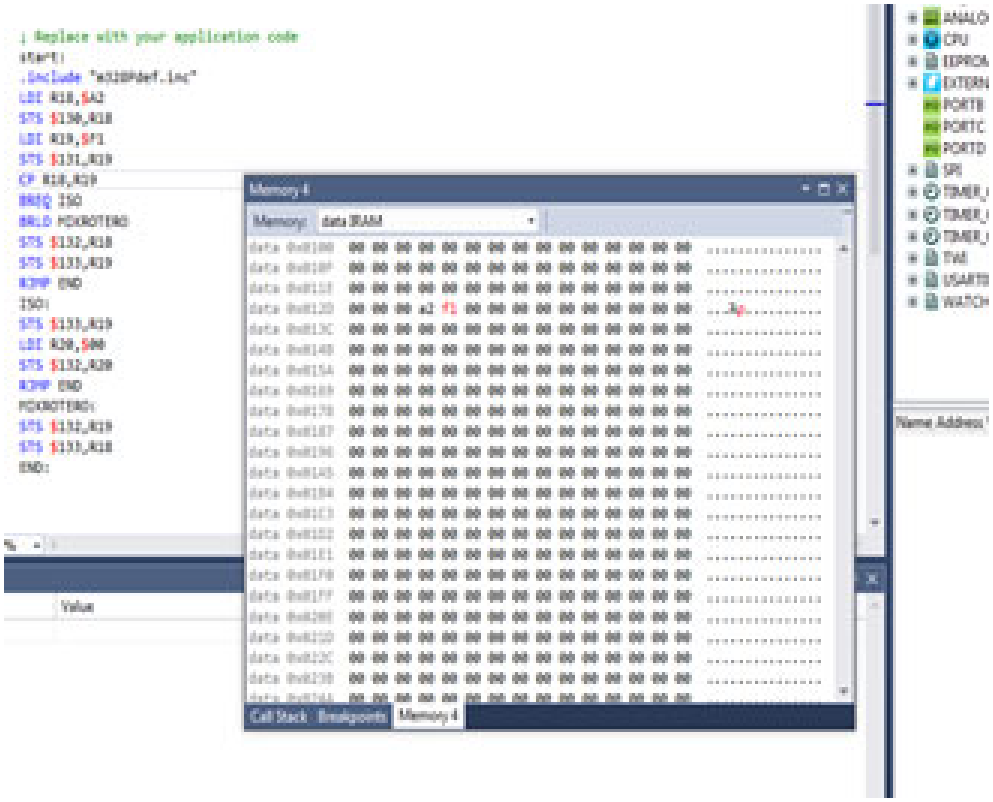
Από την επιλογή Debug, πατάμε Start Debugging and break ή Alt+F5 (εικόνα 6.20) και αρχίζει η εκτέλεση του προγράμματος. Στην ίδια εικόνα, μπορούμε να δούμε ότι για τη βηματική εκτέλεση του προγράμματος πατάμε F11, προκειμένου να σταματήσουμε το πρόγραμμα, πατάμε Stop Debugging, για τη συνέχιση του προγράμματος επιλέγουμε Continue



Εικόνα 6-20 - Menu Debug

ή F5 και για την επανεκκίνησή του, επιλέγουμε Restart.

Κατά τη διάρκεια της εκτέλεσης του προγράμματος, εμφανίζονται διάφορα



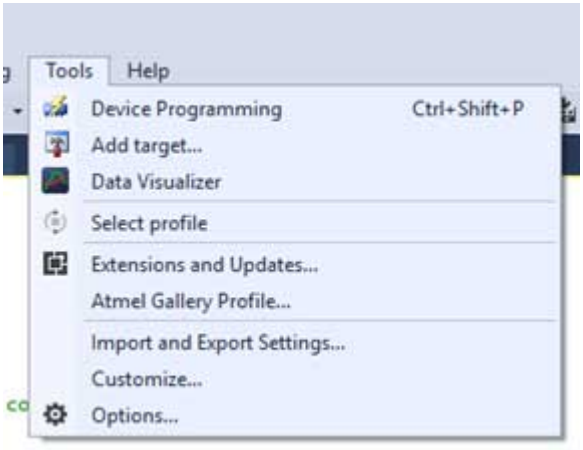
Εικόνα 6-21 - Debugging

παράθυρα (εικόνα 6.21). Το πρώτο παράθυρο (Processor) δείχνει την κατάσταση του επεξεργαστή , το δεύτερο (Memory 4) δείχνει τις τιμές της μνήμης του προγράμματος που εκτελούμε, στο τρίτο (I/O) δείχνει τις τιμές σε διάφορες παραμέτρους στου μικροελεγκτή. Στο παράθυρο Processor παρατηρούμε ότι υπάρχει ο μετρητής προγράμματος (Program Counter), ο Stack Pointer, οι καταχωρητές X, Y, και Z, οι σημαίες (Status Register), ο κύκλος ρολογιού (Cycle Counter), η συχνότητα (Frequency) και οι 32 καταχωρητές (R00 – R31). Στο παράθυρο της μνήμης, προκειμένου να δούμε τις τιμές που έχουν οι θέσεις μνήμης του προγράμματός μας στο πλαίσιο Memory, επιλέγουμε dataIRAM.

Επίσης, όπως μπορούμε να παρατηρήσουμε στην παραπάνω εικόνα, όταν έχουμε νέες τιμές στους καταχωρητές ή σε μία θέση μνήμης (το ίδιο ισχύει για τις σημαίες, για τον Program Counter αλλά και για όλα τα υπόλοιπα), οι τιμές αυτές

επισημαίνονται με κόκκινο χρώμα (στην εικόνα 6.21 έχει αλλάξει η τιμή του καταχωρητή R18 και, πλέον, έχει την τιμή $A2).

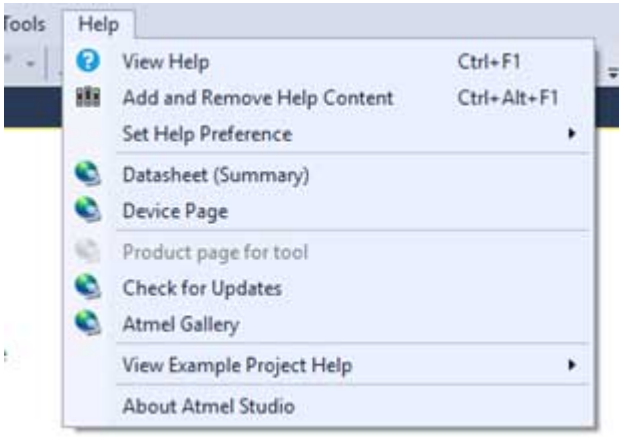
6.2.2 Γραμμή Menu



Εικόνα 6-22 – Menu Tools

Εκτός από τα παραπάνω μενού στη γραμμή των μενού υπάρχουν ακόμα οι επιλογές *Tools* και *Help*.

Στην επιλογή *Tools* υπάρχουν λειτουργίες για την ενεργοποίηση των εργαλείων εξομοίωσης του AVR και τον προγραμματισμό του μικροελεγκτή (εικόνα 6.22).



Εικόνα 6-23 – Menu Help

Επίσης, η επιλογή Help (εικόνα 6.23) περιέχει θέματα βοήθειας που αφορούν το μικροελεγκτή και το Atmel Studio, ενώ ελέγχει για τυχόν ενημερώσεις κ.ά.

7 Προγραμματισμός του ATmega328p

7.1 I/O (Port's, PushButtons,Leds)

Για όλες τις θύρες εισόδου – εξόδου του συστήματος υπάρχουν συνολικά τρεις διευθύνσεις στη μνήμη εισόδου – εξόδου, οι οποίες σχετίζονται με κάθε μία από αυτές.

Η μία από τις διευθύνσεις αυτές απαιτείται για τον καθορισμό της κατεύθυνσης των ακροδεκτών, δηλαδή καθορίζει ποιοι από τους ακροδέκτες θα λειτουργούν ως είσοδοι και ποιοι ως έξοδοι και συμβολίζονται ως DDRx. Ο καταχωρητής DDRx είναι ο λεγόμενος καταχωρητής κατεύθυνσης. Τοποθετώντας ένα bit του κατα-χωρητή αυτού σε λογικό 1, ο ακροδέκτης της θύρας PORTx στον οποίο αντιστοιχεί, θα λειτουργεί ως έξοδος. Μετά από αυτή την ενέργεια μπορούμε να οδηγούμε κατά βούληση την λογική στάθμη του αντίστοιχου ακροδέκτη εξόδου χρησι-μοποιώντας τις εντολές CBI ή SBI, ή ακόμη και τη εντολή OUT.

Η δεύτερη διεύθυνση αφορά στα δεδομένα που πρόκειται να εγγραφούν σε εκείνους (ή και όλους) τους ακροδέκτες που έχουν προγραμματιστεί ως έξοδοι και συμβολίζονται ως PORTx.

Τέλος, η τρίτη διεύθυνση αφορά στα δεδομένα που διαβάζονται από εκείνους (ή και όλους) τους ακροδέκτες που έχουν προγραμματιστεί ως είσοδοι και συμβολίζονται ως PINx.

Οι πιο απλές εντολές εισόδου-εξόδου είναι η 'in' και 'out'.

Η 'in' διαβάζει την τιμή μιας θύρας ή του καταχωρητή ενός εσωτερικού περιφερειακού (χρονιστή, UART) και τη σώζει σε έναν καταχωρητή.

Πίνακας 50 – Εντολή IN

Σύνταξη:	Καταχωρητές:	Διευθύνσεις:
IN Rd,A	$0 \leq d \leq 31$	$0 \leq A \leq 63$

Παράδειγμα:

```
IN  R25, PORTB      ;Διάβασε την Port B
CPI R25, 4          ;Σύγκρινε την τιμή με την σταθερά “4”
BREQ exit           ;Διακλάδωσε αν  R25 = 4
...
exit: nop           ;Προορισμός διακλάδωσης ( κενός  κύκλος)
```

Η **‘out’** γράφει την τιμή ενός κατ/τη σε μια θύρα ή στον κατ/τη ενός εσωτερικού περιφερειακού.

Πίνακας 51 – Εντολή OUT

Σύνταξη:	Καταχωρητές:	Διευθύνσεις:
OUT A,Rr	0 ≤ r ≤ 31	0 ≤ A ≤ 63

Παράδειγμα:

```
CLR r16             ;Καθάρισε τον r16
SER r17             ;Όλα “1” στον  r17
OUT PORTB,r16       ;Γράψε “0” στην Port B
NOP                 ;Περίμενε (κενος κύκλος)
OUT PORTB,r17       ;Γράψε “1” στην Port   B
```

Επίσης, υπάρχουν οι εντολές ‘sbi’ και ‘cbi’ για χειρισμό μεμονωμένων ψηφίων μιας θύρας και οι sbic ,sbis για τον έλεγχο των ψηφίων μιας θύρας I/O.

Μετατροπή θύρας σε είσοδο:

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

Εναλλακτικά η PORTD μετατρέπεται σε είσοδο:

```
LDI R12, 0x00          ;τα ψηφία της θύρας PORTD
OUT DDRD, R12          ;γίνονται  είσοδοι
ή
CLR R21                ;τα  ψηφία της θύρας PORTD γίνονται   είσοδοι
OUT DDRD, R21          ;μετατροπή θύρας σε   είσοδο
LDI R18, 0b11111111    ;τα ψηφία της θύρας PORTB
OUT DDRB, R18          ;  γίνονται  έξοδοι
```

Εναλλακτικά η PORTB μετατρέπεται σε έξοδο:

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

7.2 Χρονιστές – Μετρητές(Timer/Counter)

7.2.1 Βασικές έννοιες

Είναι γνωστός σε όλους ο ακόλουθος τύπος:

$$Time\ Period = \frac{1}{Frequency}$$

Ας υποθέσουμε, ότι πρέπει να αναβοσβήνει ένα LED κάθε 10 ms. Αυτό σημαίνει ότι η περίοδος είναι 1 / 10ms = 100 Hz. Η συχνότητα ρολογιού του επεξεργαστή(F_CPU)

είναι 4 MHz. Όπως έχει προαναφερθεί (κεφάλαιο 4.5) ο χρονιστής μετράει από το 0 έως το TOP. Για ένα χρονιστή 8-bit, μετρά από 0 έως 255, ενώ για ένα 16-bit μετράει από το 0 έως το 65535. Μετά από αυτό, θα υπερχειλίσει. Αυτή η τιμή αλλάζει σε κάθε παλμό ρολογιού. Ας πούμε ότι η τιμή του χρονομέτρου είναι μηδέν τώρα. Για να πάει από το 0 στο 1, χρειάζεται ένα παλμό ρολογιού. Για να πάει από 1 έως 2, χρειάζεται ένα άλλο παλμό ρολογιού. Για να πάει 2-3, χρειάζεται ακόμα ένα παλμό ρολογιού. Και ούτω καθεξής. Για F_CPU = 4 MHz, η χρονική περίοδος είναι ίση με :

T = 1 / 4M = 0,00025 ms. Έτσι, για κάθε μετάβαση (0 έως 1, 1 έως 2, κλπ), χρειάζεται μόνο 0,00025 ms!

Τώρα, όπως προαναφέρθηκε, χρειαζόμαστε μια καθυστέρηση 10 ms. Για να πάρουμε μια ιδέα για το πόσο χρόνο απαιτεί, μπορούμε να υπολογίσουμε τον αριθμό του χρονιστή (Timer Count) από τον ακόλουθο τύπο:

$$Timer\ Count = \frac{Required\ Delay}{Clock\ Time\ Period} - 1$$

Όπου θέτοντας **Required Delay**(Καθυστέρηση που απαιτείται) **= 10 ms** και **Clock Time Period** (Χρόνος Περίοδου Ρολογιού) **= 0,00025 ms** , ο αριθμός του χρονιστή (Timer Count)είναι ίσος με : **Timer Count = 39999** .

Τώρα, για να επιτευχθεί αυτό, δεν μπορούμε σίγουρα να χρησιμοποιήσουμε ένα χρονιστή 8-bit (δεδομένου ότι έχει ένα ανώτατο όριο των 255, μετά το οποίο υπερχειλίζει). Ως εκ τούτου, μπορούμε είτε να χρησιμοποιήσουμε ένα χρονιστή 16-bit (το οποίο είναι ικανό να μετρά έως 65.535) για να επιτευχθεί αυτή η καθυστέρηση, είτε να χρησιμοποιήσουμε προδιαιρέτη συχνότητας (**prescaler**). Στην συνέχεια θα δούμε τους προδιαιρέτες (prescaler) χρόνου που έχουν οι χρονιστές.

7.2.2 Προδιαιρέτες χρόνου(prescaler)

Οι μονάδες των χρονιστών μπορούν να χρησιμοποιούν μεγάλη ποικιλία εσωτερικών συχνοτήτων ,οι οποίες προκύπτουν από την διαίρεση του κεντρικού ρολογιού του μικροελεγκτή ή από μια εξωτερική πηγή. Ο καταχωρητής **Timer Counter Control Register** (TCCRx) εμπεριέχει τα bits ελέγχου του προδιαιρέτη CSx2 CSx1 και CSx0 τα οποία ορίζουν την συχνότητα λειτουργίας του χρονιστή καθώς και την πηγή του χρονισμού. Παρακάτω βλέπουμε τον καταχωρητή TCCRx καθώς και τον συνδυασμό των bit που καθορίζουν την πηγή χρονισμού.

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
					CS02	CS01	CS00

Πίνακας 52 – Προδιαιρέτης (Prescaler)

Περιγραφή	***CS02***	***CS01***	***CS0***
Timer0 απενεργοποιημένος	0	0	0
Συχνότητα συστήματος clk	0	0	1
Συχνότητα συστήματος / 8 (clk/8)	0	1	0
Συχνότητα συστήματος / 64 (clk/64)	0	1	1
Συχνότητα συστήματος / 256 (clk/256)	1	0	0
Συχνότητα συστήματος / 1024 (clk/1024)	1	0	1
Εξωτερικό pinT0 (στην κάθοδο του παλμού)	1	1	0
Εξωτερικό pinT0 (στην άνοδο του παλμού)	1	1	1

7.2.3 Επιλέγοντας προδιαιρέτη χρονου(prescaler)

Ας πάρουμε ένα παράδειγμα για να καταλάβουμε με ποιον τρόπο επιλέγουμε ένα προδιαιρέτη χρόνου.

Χρειαζόμαστε μια καθυστέρηση 184 ms και έχουμε F_CPU = 4 MHz. Θα πρέπει να επιλέξουμε από τις ακόλουθες τιμές prescaler : 8, 64, 256 και 1024. Ένας prescaler (προδιαιρέτης) 8 σημαίνει ότι η συχνότητα του ρολογιού θα είναι F_CPU / 8. Τώρα αντικαθιστώντας κάθε μία από αυτές τις τιμές στον παραπάνω τύπο, παίρνουμε τα εξής αποτελέσματα:

Απαιτούμενη Καθυστέρηση (Required Delay) = 184 ms

Συχνότητα ρολογιού (F_CPU) = 4Mhz

Πίνακας 53 – Τιμές Prescaler

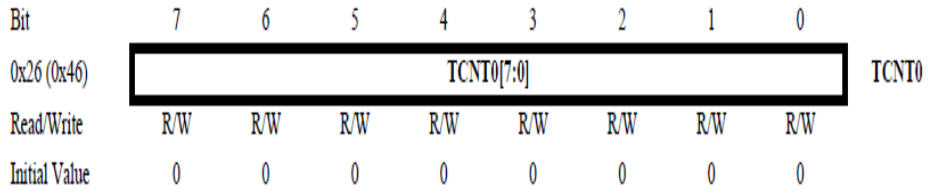
Prescaler	***Clock Frequency (kHz)***	***Timer Count***
8	500	91999
64	62,5	11499
256	15625	2874
1024	3906,25	717,75

Ο Prescaler 8 δεν μπορεί να χρησιμοποιηθεί ως η τιμή του μετρητή γιατί υπερβαίνει το όριο των 65535. Επίσης, δεδομένου ότι το χρονιστής παίρνει πάντα ακέραιες τιμές, δεν μπορούμε να επιλέξουμε τον Prescaler 1024 καθώς η τιμή του μετρητή περιέχει δεκαδικά ψηφία. Από τα παραπάνω, βλέπουμε ότι οι τιμές prescaler των 64 και 256 είναι οι ιδανικές για να χρησιμοποιηθούν. Αλλά ανάμεσα σε αυτές τις δύο, θα επιλέξουμε Prescaler 64 καθώς μας παρέχει μεγαλύτερη ανάλυση.

Έτσι, επιλέγουμε πάντα prescaler που δίνει την τιμή του μετρητή εντός του εφικτού ορίου (255 ή 65535, ανάλογα τον χρονιστή που χρησιμοποιούμε 8 bit ή 16bit) και την τιμή του μετρητή πάντα σε ακέραια μορφή.

7.2.4 Καταχωρητές που χρησιμοποιούνται

- **TCNT0** – Timer/Counter Register

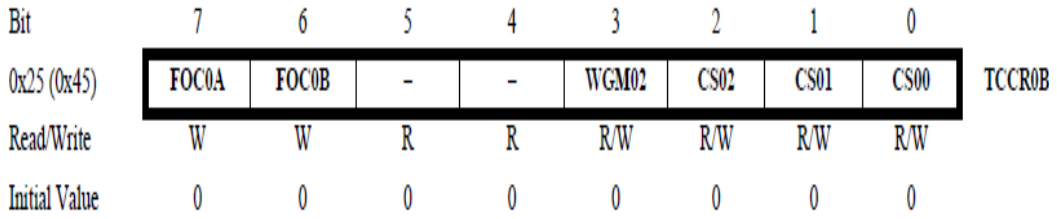


Σχήμα 23 – Timer / Counter Register

Η τιμή του μετρητή είναι αποθηκευμένη εδώ και αυξάνεται / μειώνεται αυτόματα. Αλλά ο συγκεκριμένος καταχωρητής δεν θα ενεργοποιηθεί αν δεν ενεργοποιηθεί ο χρονιστής! Έτσι, θα πρέπει να ρυθμιστεί ο χρονιστής σύμφωνα με τα παρακάτω:

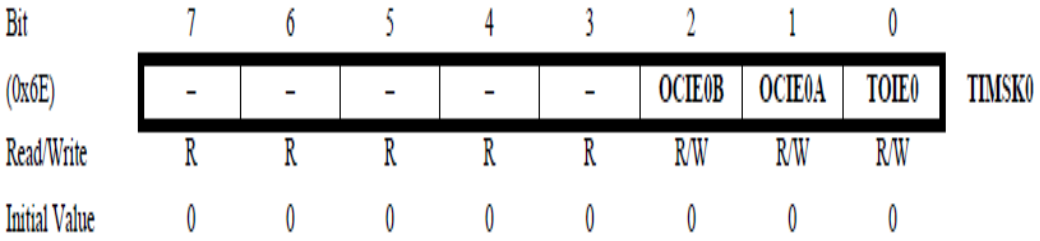
- **TCCR0B** – Timer/Counter Control Register B

Εδώ επιλέγουμε την τιμή που αντιστοιχεί στον προδιαιρέτη χρόνου που θα χρησιμοποιήσουμε. Μόλις ρυθμίσουμε prescaler η τιμή του μετρητή ξεκινάει και αυξάνετε.



Σχήμα 24 - Timer Counter Control Register B

- **TIMSK0** – Timer/Counter Interrupt Mask Register

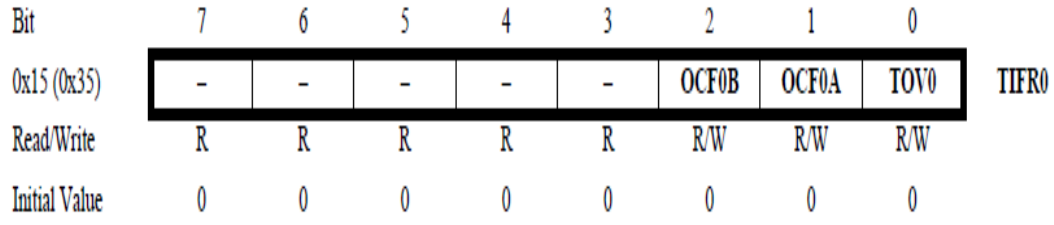


Σχήμα 25 - Timer Counter Interrupt Mask Register

Ο Timer/Counter Interrupt Mask επιτρέπει ή όχι την υπερχείλιση διακοπής στον Timer0. Και συγκεκριμένα στο bit0 ρυθμίζεται ο TOIE0:

- ✓ Αν bit0 = 0 δεν επιτρέπει την υπερχείλιση.
- ✓ Αν bit0 = 1 επιτρέπει την υπερχείλιση.

- **TIFR0** – Timer/Counter 0 Interrupt Flag Register



Σχήμα 26 - Timer / Counter 0 Interrupt Flag Register

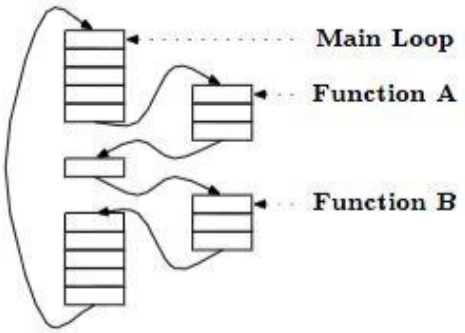
Ο **Timer/Counter Interrupt Flag Register** ελέγχει το flag TOV0 (bit0) που στη περίπτωση που είναι ενεργό (τιμή=1) έχουμε overflow.

7.3 Διακοπές(Interrupts)

7.3.1 Βασικά για τα interrupts

Ένας μικροελεγκτής κανονικά, εκτελεί τις εντολές σε μια ομαλή ακολουθία που υπαγορεύεται από το πρόγραμμα.

Ωστόσο, ένας μικροελεγκτής πρέπει επίσης να είναι έτοιμος να χειριστεί έκτακτα γεγονότα που μπορεί να συμβούν μέσα ή έξω από το μικροελεγκτή.



Σχήμα 27 – Basic Interrupt

Το σύστημα διακοπών (interrupt system) που υπάρχει σε ένα μικροελεγκτή του επιτρέπει να ανταποκριθεί σε αυτά τα εσωτερικά και εξωτερικά διαμορφούμενα γεγονότα. Εξ ορισμού δεν ξέρουμε πότε θα συμβούν αυτά τα γεγονότα.

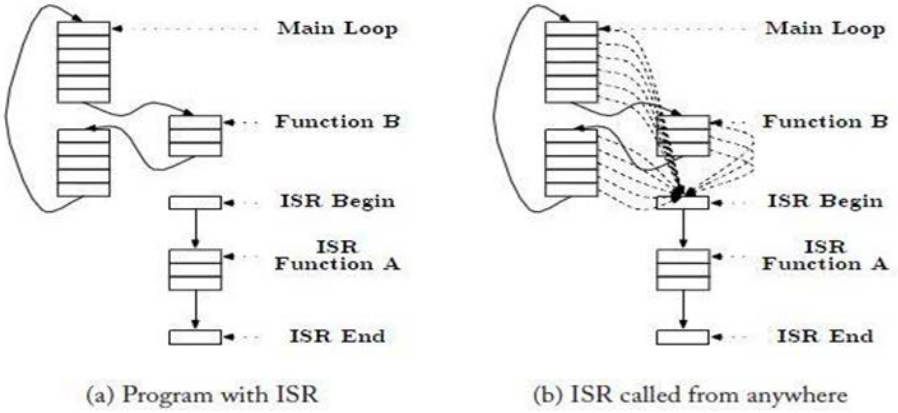
Όταν συμβεί μια περίπτωση διακοπής, μια περιφερειακή μονάδα (μέσω μιας σημαίας) ειδοποιεί το σύστημα ότι προέκυψε διακοπή.

Ο μικροελεγκτής θα ολοκληρώσει κανονικά την εντολή που εκτελεί εκείνη τη στιγμή.

Η διεύθυνση της επόμενης εντολής του κυρίου προγράμματος σώζεται στη στοίβα.

Η διεύθυνση της ρουτίνας εξυπηρέτησης διακοπής(Interrupt Service Routine, ISR) φορτώνεται στον μετρητή προγράμματος μέσω μιας εντολής icall, rcall κλπ., ώστε να εκτελεστεί η 1η εντολή της ISR.

Οι εντολές της ρουτίνας εξυπηρέτησης διακοπής εκτελούνται διαδοχικά μέχρι και την τελευταία εντολή τύπου reti, όπου η διεύθυνση επιστροφής φορτώνεται από τη στοίβα και η σημαία ολικών διακοπών τίθεται. Μόλις η ISR ολοκληρωθεί, επιστρέφει ο έλεγχος στο πρόγραμμα και ο μικροελεγκτής θα συνεχίσει την επεξεργασία από εκεί που σταμάτησε πριν συμβεί η διακοπή.



Σχήμα 28 – Λειτουργία Interrupt

7.3.2 Οι λόγοι που χρησιμοποιούνται τα Interrupts

- Να ανιχνεύσουνε αλλαγές στα pinchanges (πχ. Εκκίνηση κινητήρα, πάτημα κουμπιού κλπ)
- Watchdogtimer (πχ αν δεν συμβεί τίποτα μετά από 8 seconds, διέκοψε - ειδοποίησε με)
- Timer interrupts – χρησιμοποιείται για σύγκριση/ υπερχείλιση χρονιστών
- SPI μεταφορά δεδομένων
- I2C μεταφορά δεδομένων
- USART μεταφορά δεδομένων
- ADC μετατροπή conversions (αναλογικό σε ψηφιακό)
- Ετοιμότητα για χρήση EEPROM
- Ετοιμότητα Flash memory

7.3.3 Καταχωρητές που χρησιμοποιούνται

- **EICRA** – External Interrupt Control Register A

Ο Ελεγκτής εξωτερικής διακοπής (External Interrupt Control Register) περέχει τα bit ελέγχου για την ευαισθησία του interrupt (σχ. 29)

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Σχήμα 29 - External Interrupt Control Register A

- **Bit 7:4 - Δεσμευμένα**

  Αυτά τα bit δεν χρησιμοποιούνται στον ATmega328p, και για αυτό το λόγο είναι πάντα 0.

- **Bit 3, 2 – ISC11, ISC10 Bit1 και 0 ελέγχου ευαισθησίας interrupt 1**

  Το εξωτερικό interrupt 1 ενεργοποιήται από το pin INT 1 αν το bit I-flag του SREG και το αντίστοιχο pin του EIMSK έχουν ενεργοποηθεί. Η ευαισθησία του καθορίζεται από τις τιμές των ISC11 και ISC10 στον πίνακα 54.

Πίνακας 54 - ICS11, ISC10

ISC11	ISC10	Περιγραφή
0	**0**	Η χαμηλή τιμή του INT1 προκαλεί το interrupt
0	**1**	Οποιαδήποτε λογική μεταβολή προκαλεί το interrupt
1	**0**	Η πτώση προκαλεί το interrupt
1	**1**	Η άνοδος προκαλεί το interrupt

- **Bit 1, 0 – ISC01, ISC00 Bit1 και 0 ελέγχου ευαισθησίας interrupt 0**

  Το εξωτερικό interrupt 0 ενεργοποιήται από το pinINT 0 αν το bitI-flag του SREG και το αντίστοιχο pin του EIMSK έχουν ενεργοποηθεί. Η ευαισθησία του καθορίζεται από τις τιμές των ISC01 και ISC00 στον πίνακα 55.

Πίνακας 55 – ISC01, ISC00

ISC01	ISC00	Περιγραφή
0	**0**	Η χαμηλή τιμή του INT0 προκαλεί το interrupt
0	**1**	Οποιαδήποτε λογική μεταβολή προκαλεί το interrupt
1	**0**	Η πτώση προκαλεί το interrupt
1	**1**	Η άνοδος προκαλεί το interrupt

- **EIMSK** – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Σχήμα 30 - External Interrupt Mask Register

- **Bit 7:2 - Δεσμευμένα**

  Αυτά τα bit δεν χρησιμοποιούνται στον ATmega328p, και για αυτό το λόγο είναι πάντα 0.

- **Bit 1 - INT1**

  Αν ενεργοποιηθεί μαζί με το I-bit του SREG ενεργοποιεί το INT1.

- **Bit 1 - INT0**

  Αν ενεργοποιηθεί μαζί με το I-bit του SREG ενεργοποιεί το INT0.

8 Παραδείγματα Προγραμματισμού

Για να «τρέξουμε» τα παρακάτω παραδείγματα υλοποιήσαμε το κύκλωμα που παρουσιάζεται στο Παράρτημα Α (σχήμα 34), σαν εξωτερική διακοπή χρησιμοποιούμε ένα κύκλωμα με push-buttton και σαν led εξόδου ένα 7-segment display που περιγράφονται στο ίδιο παράρτημα.

8.1 Εύρεση μεγίστου-ελαχίστου

Ο αλγόριθμος εύρεσης μέγιστου και ελάχιστου μπορεί να αποτελέσει μέρος ενός μεγαλύτερου προγράμματος σε αναρίθμητές περιπτώσεις ή να και να χρησιμοποιηθεί ως αυτόνομη εφαρμογή.

Όπως γίνεται εύκολα κατανοητό στόχος του συγκεκριμένου αλγορίθμου είναι η εύρεση της μέγιστης ή/και ελάχιστης αριθμητικής τιμής σε μια συγκεκριμένη και σαφώς οριοθετημένη περιοχή μνήμης.

Ας θεωρήσουμε λοιπόν μια περιοχή μνήμης με αρχική θέση μνήμης την $70 και τελική την $7A και στόχο την εύρεση του μέγιστου αποθηκεμένου αριθμού και μεταφορά αυτού στην θέση μνήμης $80.

Συγκεκριμένα:

MAXIMUM→Περιέχει την μέγιστη τιμή που έχει βρεθεί ανά πάσα στιγμή και φυσικά θα περιέχει τη μέγιστη τιμή της περιοχής μνήμης μετά το τέλος του προγράμματος.

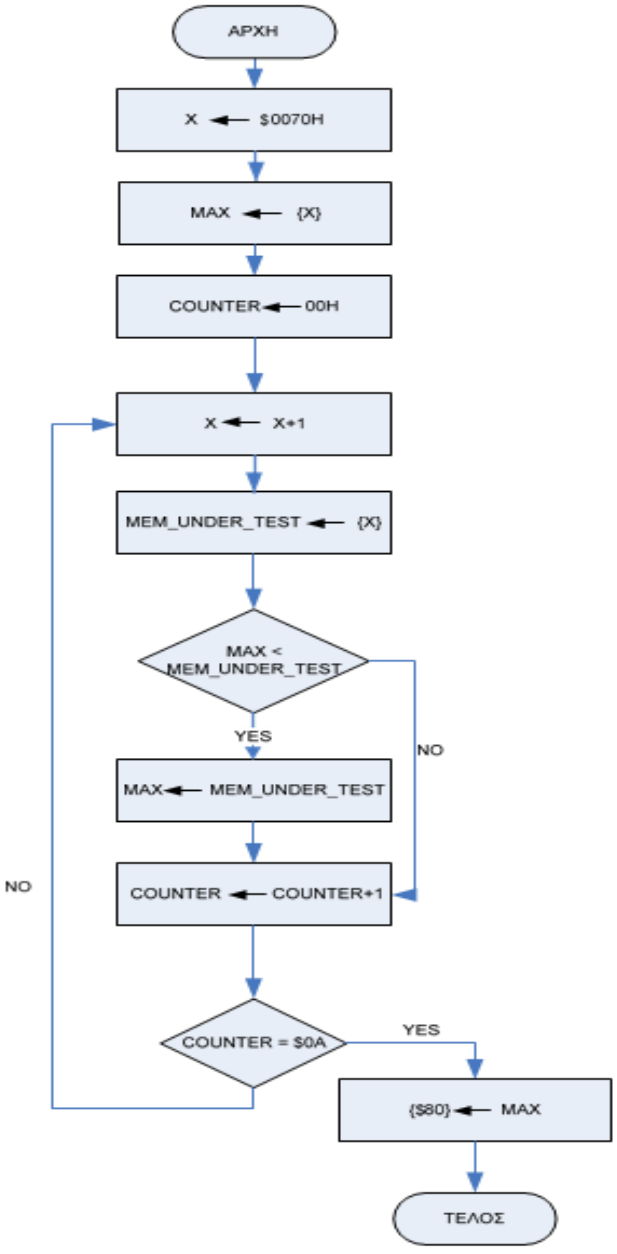
COUNTER →Βοηθητικός μετρητής ο οποίος περιέχει το πλήθος των θέσεων μνήμης που έχουν ελεγχθεί και κατά συνέπεια αποτελεί το κριτήριο για το πότε έχει ολοκληρωθεί η διαδικασία.

MEM_UNDER_TEST →Βοηθητικός καταχωρητής ο οποίος περιέχει το περιεχόμενο της μνήμης που ελέγχεται αν πάσα στιγμή.

Παρακάτω ακολουθεί η υλοποίηση του αντίστοιχου προγράμματος και το διάγραμμα ροής:

```
.include "m328Pdef.inc"

.def MAXIMUM=r20
.def COUNTER=r18
.def MEM_UNDER_TEST=r17
ldi XH, $00
ldi XL,$70
ld MAXIMUM,X
ldi COUNTER, 0
NEXT_CHECK:
ld MEM_UNDER_TEST, X+
cp MAXIMUM,MEM_UNDER_TEST
brcc NO_MAX_UPDATE
mov MAXIMUM,MEM_UNDER_TEST
NO_MAX_UPDATE:
inc COUNTER
cpi COUNTER,$0A
brne NEXT_CHECK
```



Σχήμα 31 – Διάγραμμα ροής παραδείγματος

8.2 Συγκρίσεις,Διακλαδώσεις & βρόχοι

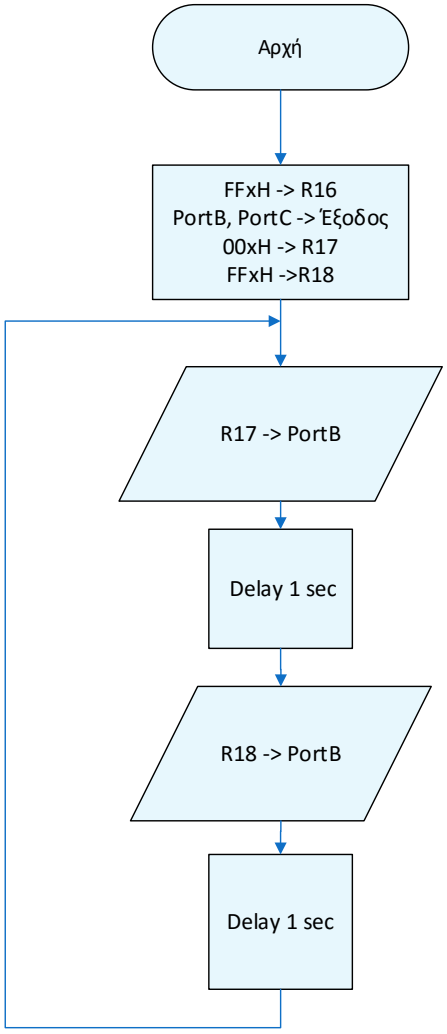
Το παρακάτω πρόγραμμα αρχικά αποθηκεύει δύο τυχαίους αριθμούς στις θέσεις μνήμης $130h και $131h. Στην συνέχεια τοποθετεί τον μεγαλύτερο στην θέση μνήμης $132h και τον μικρότερο στην $133h. Στην περίπτωση όμως που είναι ίσοι οι δύο αριθμοί, τοποθετεί το 00H στην θέση μνήμης $132H και στη θέση μνήμης $133h τον αριθμό.

```
.include "m328Pdef.inc"

ldi r18,$A2      ;μεταφορα των δεδομένων απο τον καταχωρητη
sts $130, r18    ;r18 στη διευθυνση μνήμης SRAM $130h
ldi r19,$F1      ;μεταφορά των δεδομένων απο τον καταχωρητή
sts $131, r19    ;R19 στη διευθυνση μνημης SRAM $131h
cp R18,R19
breq ISO
brlo MIKROTERO
sts $132, r18
sts $133, r19
rjmp END
ISO:
      sts $133, r19
      ldi r20,$00
      sts $132, r20
      rjmp END
MIKROTERO:
      sts $132, r19
      sts $133, r18
END:
```

8.3 Χειρισμός LED με χρήση θυρών – Υποπρόγραμμα καθυστέρησης

Το παρακάτω πρόγραμμα αναβοσβήνει στο display που βρίσκετε στους ακροδέκτες της θύρας B και C, τον αριθμό 0, με περιοδικότητα περίπου ένα (1) δευτερόλεπτο.



Σχήμα 32 - Flow Chart

Απάντηση:

```
.def dly1=r19
.def dly2=r20
.def dly3=r21

;Αρχικοποίηση του Stack Pointer
      ldi R16, LOW(RAMEND)
      out SPL,R16
      ldi R16, HIGH(RAMEND)
      out SPH,R16
      ldi R16, 0b11111111
      out DDRB, R16
; Ορίζουμε την Port B ως έξοδο
      out DDRC,R16
; Ορίζουμε την Port C ως έξοδο
      clr R17 ; Όλα "0" στον  r17
      ser R18 ; Όλα "1" στον r18

loop:
      out PORTB,R17;Γράφουμε στην
Port B τα περιεχόμενα του R17
      cbi PORTC, 2 ;σβήσιμο pin 2 portc
      rcall Delay  ;Καλούμε την υπορουτίνα  "Delay"
      out PORTB,R18;Γράφουμε στην Port B τα περιεχόμενα του R18
      sbi PORTC,2  ;αναμμα pin 2 portc
      rcall Delay  ;Καλούμε την υπορουτίνα  "delay"
      rjmp loop

Delay:            ;Για CLK(CPU) = 8 MHz – Delay 1 sec
    ldi dly1, 21
Delay1:
    ldi dly2, 42
Delay2:
    ldi dly3, 83
```

```
Delay3:
    dec dly3
    nop
    brne Delay3
     dec dly2
    brne Delay2
     dec dly1
    brne Delay1
     ret
```

8.4 Παραδείγμα με τους Χρονιστές

Το παρακάτω πρόγραμμα χρησιμοποιώντας τον TIMER0, ανάβει στο display που βρίσκετε στους ακροδέκτες της θύρας B και C, πρώτα τον αριθμό 0, μετά τον 3, μετά τον 5 και τέλος τον αριθμό 7.

```
.include "m328Pdef.inc"

.def leds = r17
.def tmp = r16
.def DelayTime = r19
.def dly1 = r21
.def dly2 = r22
.def dly3 = r24

;Αρχικοποίηση SP
ldi tmp, high(RAMEND)
out SPH, tmp
ldi tmp, low(RAMEND)
out SPL, tmp

; Ορισμός των port
ser tmp
out DDRC,   tmp       ; port c σαν έξοδο
out DDRB,   tmp       ; port b σαν έξοδο

; Αρχικοποίηση timer0
ldi DelayTime, 0x00   ;αρχικοποίηση του TCNT0 έτσι ώστε η
                      ;καταμέτρηση να ξεκινά
out TCNT0, DelayTime  ;από το DelayTime
ldi r20, 0x05
out TCCR0B, r20       ;ορισμός του prescaller σε 1024
ldi r23, 0x00
sts TIMSK0, r23       ;ενεργοποίηση του timer0
```

```
; γενική ενεργοποίηση των interrupt
SEI

jmp main

Clean_portC:
      cbi portc,3
      cbi portc,2
      ret
Clean_Overflow:
      Ldi tmp,1<<TOV0
      out TIFR0,tmp    ; Clear TOV0/ Clear pending interrupts
      ldi r23,  0x00
      sts TIMSK0, r23
      out TCNT0, DelayTime  ;αρχικοποίηση του Tcnt0 έτσι ώστε η
                            ;καταμέτρηση να ξεκινά στο 0
      reti
Digit0:                     ; εμφάνιση ψηφίου 0
      ldi leds,0b00111110
      sbi portc ,2
      out portb, leds
      reti
Digit3:                     ; εμφάνιση ψηφίου 3
      ldi leds, 0b00011110
      sbi portc ,3
      out portb, leds
      reti
Digit5:                     ; εμφάνιση ψηφίου 5
      ldi leds, 0b00011010
      sbi portc ,2
      sbi portc ,3
      out portb, leds
      reti
Digit7:                     ; εμφάνιση ψηφίου 7
      ldi leds, 0b00001110
      out portb, leds
      reti
Check_Overflow:
      sbis tifr0, tov0
      rjmp Check_Overflow
      reti

main:
      cbi portc,5           ;σβήσιμο led στο PortC 5 (PC5)
      call digit0
      call check_overflow
      call Clean_Overflow
      call Clean_portc
      call digit3
```

```
      call check_overflow
      call Clean_Overflow
      call Clean_portc
      call digit5
      call check_overflow
      call Clean_Overflow
      call Clean_portc
      call digit7
      call check_overflow
      call Clean_Overflow
      call Clean_portc
      clr leds
      out portb, leds
      sbi portb, 0          ;άναμμα κουκίδας
      cli                   ;γενική απενεργοποίηση των interrupt
      call check_overflow
      call Clean_Overflow
      sbi portc,5
      call check_overflow
      call Clean_Overflow
```

8.5 Παραδείγματα με Interrupts

Παράδειγμα 1ο

Στο πρόγραμμα αυτό κάθε φορά που προκαλείται εξωτερική διακοπή INT0 (ακροδέκτης PD2) αλλάζει η τιμή του display που βρίσκετε στους ακροδέκτες της θύρας B και C, σχηματίζοντας κάθε φορά διαφορετικό αριθμό. Συγκεκριμένα ξεκινάει με 0,μετα 1,2,3,4και τέλος ανάβει το 5.

```
.include"m328Pdef.inc"


.def temp = r16
.def leds = r17
.def digitCounter = r18

jmp reset                   ;Reset Handler
jmp interrupt0               ;IRQ0 Handler
reti

reset:
      ldi temp, high(RAMEND)  ; κύριο πρόγραμμα
      out SPH,temp            ; θέτουμε δείκτη στοίβας στην RAM
```

```
      ldi temp, low(RAMEND)
      out SPL, temp
      ser leds
      out DDRB, temp           ; portb ως έξοδο των leds
      out DDRC, temp           ; portc ως έξοδο των leds
      ldi leds, 0x00
      out portb, leds
      sbi portb, portb0         ; άναμμα κουκίδας στο display
                               ; (θέτω 0 στο portb0)
      ldi temp, 0x01
      out EIMSK, temp   ; ενεργοποίηση εξωτερικής διακοπής INT0
      ldi temp, 0x02    ; ορίζουμε η εξωτερική διακοπή INT0 να
      sts EICRA, temp   ; προκαλείται στην ακμή πτώσης
      sei               ; ενεργοποίηση συνολικά των διακοπών

loop:                    ; αναμονή εξωτερικής διακοπής INT0
  rjmp loop              ; (πάτημα διακόπτη 2)



interrupt0:              ; ρουτίνα εξυπηρέτησης διακοπής INT0
      cpi digitCounter, 0
      breq digit0
      cpi digitCounter, 1
      breq digit1
      cpi digitCounter, 2
      breq digit2
      cpi digitCounter, 3
      breq digit3
      cpi digitCounter, 4
      breq digit4
      cpi digitCounter, 5
      breq digit5
      clr leds
      out portb, leds
      call clean_portc
      sbi portb, portb0
      clr digitcounter
      reti
Digit0:                  ; εμφάνιση ψηφίου 0
      call clean_portc
      ldi leds,0b00111110
      sbi portc ,2
      out portb, leds
      inc digitCounter
      reti
Digit1:                    ; εμφάνιση ψηφίου 1
      call clean_portc
      ldi leds,0b00001100
```

```
      out portb, leds
      inc digitCounter
      reti
Digit2:                    ; εμφάνιση ψηφίου 2
      call clean_portc
      ldi leds,0b00110110
      sbi portc ,3
      out portb, leds
      inc digitCounter
      reti
Digit3:                    ; εμφάνιση ψηφίου 3
      call clean_portc
      ldi leds, 0b00011110
      sbi portc ,3
      out portb, leds
      inc digitCounter
      reti
Digit4:                    ; εμφάνιση ψηφίου 4
      call clean_portc
      ldi leds, 0b00001100
      sbi portc ,3
      sbi portc ,2
      out portb, leds
      inc digitCounter
      reti
Digit5:                    ; εμφάνιση ψηφίου 5
      call clean_portc
      ldi leds, 0b00011010
      sbi portc ,2
      sbi portc ,3
      out portb, leds
      inc digitCounter
      reti
Clean_portC:               ; καθαρισμός Port C
      cbi portc,3
      cbi portc,2
      ret
```

Παράδειγμα 2ο

Το πρόγραμμα αυτό είναι συνδυασμός των χρονιστών (timers) και των διακοπών (interrupts). Αρχικά το display που βρίσκεται στους ακροδέκτες της θύρας B και C, σχηματίζει τους αριθμούς 0, 1,2,3,4 και τέλος ανάβει το 5, λειτουργεί δηλαδή σαν «μετρητής» από το 0-5 παρεμβάλλοντας χρονική καθυστέρηση ανάμεσα στην εμφάνιση των αριθμών χρησιμοποιώντας τον TIMER0. Κάθε φορά που προκαλείται εξωτερική διακοπή INT0 (ακροδέκτης PD2) αντιστρέφετε η λειτουργία του «μετρητή» και ξεκινάει να μετραει από το 5 εως το 0. Η διαδικασία επαναλαμβάνεται οσο χρησιμοποιούμε την εξωτερική διακοπή.

```
.include"m328Pdef.inc"

.def leds = r17
.def tmp = r16
.def DelayTime = r19
.def counter = r20

jmp reset              ; Reset Handler
jmp interrupt0         ; IRQ0 Handler
reti                   ; Υπόλοιποι Handlers
reset:
; Αρχικοποίηση του SP
      ldi tmp, high(RAMEND)
      out SPH,tmp      ; θέτουμε δείκτη στοίβας στην RAM
      ldi tmp, low(RAMEND)
      out SPL, tmp
; Ορισμός των port
      ser tmp
      out DDRc,   tmp        ; port c σαν έξοδο
      out DDRB,   tmp        ; port b σαν έξοδο
; Αρχικοποίηση timer0
      ldi DelayTime, 0x00    ;αρχικοποίηση του TCNT0 έτσι ώστε η
                             ;καταμέτρηση να ξεκινά
      out TCNT0, DelayTime   ;από το DelayTime(0)
      ldi r20, 0x05
      out TCCR0B, r20        ;ορισμός του prescaller σε 1024
      ldi r23, 0x00
      sts TIMSK0, r23        ;ενεργοποίηση του timer0

;Ενεργοποίηση interrupt0
      ldi tmp, 0x01
      out EIMSK, tmp   ; ενεργοποίηση εξωτερικής διακοπής INT0
      ldi tmp, 0x02    ; ορίζουμε η εξωτερική διακοπή INT0 να
      sts EICRA, tmp   ; προκαλείται στην ακμή πτώσης
```

```
; γενική ενεργοποίηση των interrupt
      sei               ; ενεργοποίηση συνολικά των διακοπών
      sbi portc,0       ; ενεργοποίηση του bit 0 της port C σαν
                        ; flag για αύξηση ή μείωση του counter

check:                  ; έλεγχος flag
      sbis portc,0
      rjmp downcounter
      rjmp upcounter

loop:                   ; αναμονή εξωτερικής διακοπής INT0
      rjmp loop         ; (πάτημα διακόπτη 2)

interrupt0:             ; ρουτίνα εξυπηρέτησης διακοπής INT0
      sbic portc,0
      rjmp clearpin
      nop
      sbi portc,0
      rjmp ret1
clearpin:
      cbi portc,0
ret1:
      reti


Clean_Overflow:         ; Καθαρισμός TOV0 / Overflow
      ldi tmp,1<<TOV0
      out TIFR0,tmp     ; Καθαρισμός του TOV0
      ldi r23,  0x00
      sts TIMSK0, r23
      out TCNT0, DelayTime  ; αρχικοποίηση του Tcnt0 έτσι ώστε
                            ; η καταμέτρηση να ξεκινά στο 0
      ret
downcounter:      ; ορισμός του counter για αντίστροφη μέτρηση
      ldi counter, 5
      rjmp main
upcounter:        ; ορισμός του counter για κανονική μέτρηση
      ldi counter, 0
      rjmp main
setportc:         ; άναμμα led στο port C
      sbi portc,5
      call check_overflow
      ret
clearportc:       ; σβήσιμο led στο port C
      cbi portc,5
      ret
decrease:         ; έλεγχος του counter αν έχει τερματιστεί
                  ; και αν όχι μείωσή του
```

```
      cpi counter, 0
      breq downcounter
      dec counter
      rjmp main
increase:       ; έλεγχος του counter αν έχει τερματιστεί
                ; και αν όχι αύξησή του
      cpi counter, 5
      breq upcounter
      inc counter
      rjmp main
Digit3:
      call Clean_portC
      ldi leds, 0b00011110
      out portb, leds
      sbi portc ,3
      call check_overflow
      sbis portc,0
      rjmp decrease
      rjmp increase
main:
      cpi counter, 4
      breq digit4
      cpi counter, 3
      breq digit3
      cpi counter, 2
      breq digit2
      cpi counter, 1
      breq digit1
      cpi counter, 0
      breq digit0
      cpi counter, 5
      breq digit5
      ret
Digit0:
      call Clean_portC
      ldi leds,0b00111110
      out portb, leds
      sbi portc ,2
      call check_overflow
      call setportc
      sbis portc,0
      rjmp decrease
      rjmp increase

Digit1:
      call Clean_portC
      call clearportc
      ldi leds,0b00001100
      out portb, leds
```

```
      call check_overflow
      sbis portc,0
      rjmp decrease
      rjmp increase
Digit2:
      call Clean_portC
      ldi leds,0b00110110
      out portb, leds
      sbi portc ,3
      call check_overflow
      sbis portc,0
      rjmp decrease
      rjmp increase
Digit5:
      call Clean_portC
      call clearportc
      ldi leds, 0b00011010
      out portb, leds
      sbi portc ,2
      sbi portc ,3
      call check_overflow
      sbis portc,0
      rjmp decrease
      rjmp increase
Digit4:
      call Clean_portC
      ldi leds, 0b00001100
      out portb, leds
      sbi portc ,3
      sbi portc ,2
      call check_overflow
      sbis portc,0
      rjmp decrease
      rjmp increase
Check_Overflow:            ; έλεγχος αν έχει επέλθει overflow
      sbis tifr0, tov0
      rjmp Check_Overflow
      call clean_Overflow
      ret
Clean_portC:
      cbi portc,3
      cbi portc,2
      ret
```

9 Μετατρέποντας τον AVR σε Arduino

9.1 Εγκατάσταση του Bootloader

Για να μετατρέψουμε τον Atmega328p σε Arduino πρέπει να φορτώσουμε τον Bootloader (ο οποίος αποτελεί το "λειτουργικό" του Arduino) πάνω στον μικροελεγκτή. Ετσι λοιπόν ξεκινάμε με την εγκατάσταση του ArduinoIDE.

Για την εγκατάσταση του περιβάλλοντος προγραμματισμού κατεβάζουμε το πρόγραμμα από την επίσημη σελίδα του Arduino[24] τρέχουμε το εκτελέσιμο αρχείο και ακολουθούμε τις οδηγίες επί της οθόνης. Μόλις τελειώσει η εγκατάσταση πηγαίνουμε στην σελίδα https://www.arduino.cc/en/Tutorial/ArduinoToBreadboard και ακολουθούμε τα εξής βήματα:

1. Κατεβάζουμε το αρχείο breadboard-1-6-x.zip
2. Αποσυμπιέζουμε το παραπάνω αρχείο και μεταφέρουμε τα περιεχόμενά του μέσα στον φάκελο "hardware" που βρίσκεται μέσα στον φάκελο "Arduino"που δημιουργήθηκε μετά την εγκατάσταση.
3. Κάνουμε επανεκκίνηση το πρόγραμμα του ArduinoIDE
4. Ανοίγοντας το μενού Tools > Board menu θα πρέπει να δείτε και να επιλέξετε το υπομενού "ATmega328 on a breadboard (8 MHz internal clock)".

Στη συνέχεια για να φορτώσουμε τον bootloader,θα πρέπει το πρόγραμμα του ArduinoIDE να αναγνωρίσει τον AVR programmer που χρησιμοποιούμε στο κύκλωμα μας. Επειδή όμως ο Arduino IDE δεν είναι πλήρως ενημερωμένος για τον προγραμματιστή που χρησιμοποιούμε (AVR ISP XPII),πρέπει να φορτώσουμε εξωτερικούς driver ώστε να τον αναγνωρίσει[25]. Τους driver αυτούς θα τους βρείτε στην παρακάτω σελίδα : http://www.visualmicro.com/post/2014/01/17/AvrIsp-MkII-Usb-Driver-for-Arduino.aspx

Αφού τους εγκαταστήσουμε, στο μενού Tools -> Programmer θα πρέπει να δείτε και να επιλέξετε το υπομενού "AVR ISP mkII ".

[24] https://www.arduino.cc/en/Main/Software

[25] Ο Arduino IDE δεν έναι συμαβτός με τον AVR ISP Programmer MKII (orcompatible) σύμφωνα με https://forum.arduino.cc/index.php?topic=316037.0

Τώρα είσαστε έτοιμοι να φορτώσετε τον bootloader ακολουθώντας την διαδρομή στον Arduino IDE Tools -> Burn Bootloader.

9.2 Παράδειγμα σε γλώσσα Wiring

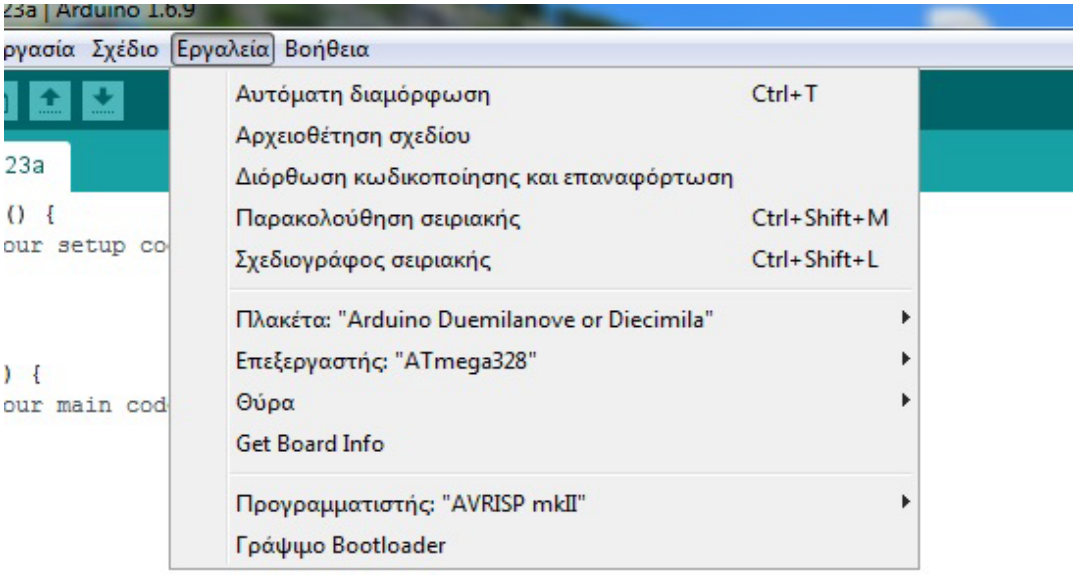
Για να επιβεβαιώσουμε ότι ο μικροελεγκτής μας μετατράπηκε σε Arduino θα τρέξουμε ένα απλό πρόγραμμα σε γλώσσα Wiring (την γλώσσα προγραμματισμού του Arduino), χρησιμοποιώντας το κύλώμα που παρουσιάζεται στο παράρτημα Α (σχήμα 37). Το πρόγραμμα αυτό αναβοσβήνει ένα led κάθε 1 δευτερόλεπτο που έχουμε συνδέσει στην PB5 (pin 19 του atmega328p) ακολουθώντας την αντιστοιχία των pin μεταξύ Arduino – Atmega328p όπως φαίνεται στον παρακάτω πίνακα.

Πίνακας 56 – Αντιστοίχιση Pin[26]

Arduino Label	**ATmega328p**		**Arduino Label**	**ATmega328p**	
	Port	**Pin**		**Port**	**Pin**
A0	PC0(ADC0)	23	**11**	PB3(MOSI)	17
A1	PC1(ADC1)	24	**10**	PB2(OC1B)	16
A2	PC2(ADC2)	25	**9**	PB1(OC1A)	15
A3	PC3(ADC3)	26	**8**	PB0	14
A4	PC4(ADC4)	27	**7**	PD7	13
A5	PC5(ADC5)	28	**6**	PD6	12
SCL	PC5(ADC5/SCL)	28	**5**	PD5	11
SDA	PC4(ADC4/SDA)	27	**4**	PD4	6
AREF	AREF	21	**3**	PD3(INT1)	5
GND	GND	22 & 8	**2**	PD2(INT0)	4
13	PB5(SCK)	19	**1**	PD1(TXD)	3
12	PB4(MISO)	18	**0**	PD0(RXD)	2

[26] http://www.microdigitaled.com/AVR/Hardware/Arduino/UsingArduinoBoardsInAtmelStudio.pdf

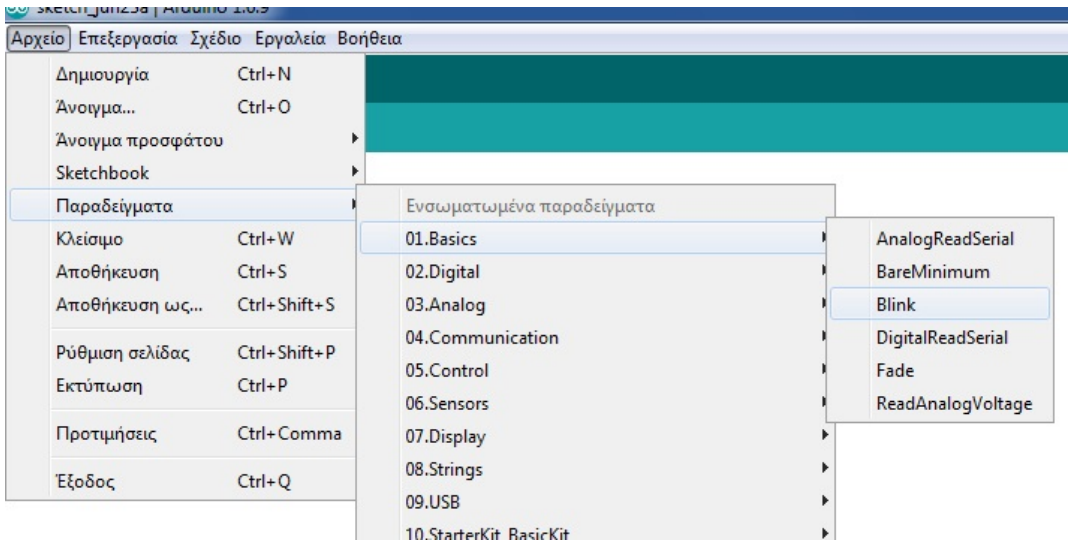
Τρέχουμε το πρόγραμμα χρησιμοποιώντας τον Arduino IDE ακολουθώντας την εξής



Εικόνα 9-1 – Menu Εργαλεία

διαδικασία :

Αρχικά θα ορίσουμε τις παραμέτρους του κυκλωματος που χρησιμοποιούμε, συγκεκριμένα πηγαίνουμε στο μενού *Εργαλεία (Tools)* και επιλέγουμε σαν πλακέτα και επεξεργαστή τις παραμέτρους που παρουσιάζονται στην εικόνα 9-1, και ως θύρα επιλέγουμε τη θύρα που δημιουργήθηκε κατά την εγκατάσταση της πλακέτας FTDI



Εικόνα 9-2 - Arduino Sketch

Basic που χρησιμοποιούμε.

Στη συνέχεια φορτώνουμε το παράδειγμα «Blink» όπως φαίνεται στην εικόνα 9-2.

Τέλος στην εικόνα 9-3 παρουσιάζεται ο κώδικας του sketch “Blink” ο οποίος για να

Εικόνα 9-3 – Sketch “Blink”

εκτελεστεί θα πρέπει να γίνει ανέβασμα (upload) στο κύκλωμά μας.

Συμπεράσματα

Η ολοκληρωμένη μελέτη που αναπτύχτηκε κατά την διάρκεια της πτυχιακής αυτής, απέδειξε πως οι μικροελεγκτές έχουν πολύ μεγάλες δυνατότητες. Οι μικροελεγκτές γενικότερα και ο ATmega328p ειδικότερα δεν αποτελούν απλά μια παραλλαγή του μικροεπεξεργαστή με κάποια συγκεκριμένα χαρακτηριστικά και ιδιότητες,αλλά είναι ισχυρά ολοκληρωμένα κυκλώματα που εξυπηρετούν εφαρμογές ειδικού σκοπού.

Χρησιμοποιώντας ως βάση τους μικροελεγκτές σε συνδυασμό με βασικές γνώσεις ηλεκτρονικής και προγραμματισμού, καθώς και τα απαραίτητα εξαρτήματα (ράστερ, αντιστάσεις, πυκνωτές, leds κ.ο.κ) μπορούν να παραχθούν οποιουδήποτε τύπου ηλεκτρικές εφαρμογές.

Ως τέτοιες εφαρμογές μπορουν να θεωρηθούν το κύκλωμα αναγνώρισης θερμοκρασίας και ενεργοποίηση αντίστοιχης συσκευής, το κύκλωμα συναγερμού (αυτοκινητού ή χώρου), ο έλεγχος των ψαλιδιών σε μηχανήματα κοπής, ο έλεγχος ηλεκτρομαγνητικής κλειδαριάς σε θωρακισμένη πόρτα κ.α.

Βιβλιογραφία

- Καλαμαράς Αλέξανδρος, *Ανάπτυξη εφαρμογών με τη σειρά μικροελεγκτών Mega της Atmel*, Πτυχιακή εργασία, Τμ. Ηλεκτρονικής, ΑΤΕΙ Θεσ/νικης, 2009
- Μπότσαρης Δημοσθένης, *Συσκευή USB για μέτρηση μαγνητικού πεδίου με αισθητήρα Hall,* Διπλωματική εργασία, Τμ. Ηλεκτολόγων Μηχανικών. ΑΠΘ, 2013
- Τσιλιγιάννης Γεώργιος, *Σχεδίαση και Κατασκευή Συστήματος ελέγχου φωτισμού με μικροελεγκτές AVR*, Μεταπτυχιακή διατριβή, Τμ Πληροφορικής, Πανεπιστήμιο Πειραιώς, 2010.
- Κωνσταντίνος Β.Ρακόπουλος, *Υλοποίηση αποκωδικοποιητή mp3 με τη χρήση του μικροελεγκτή AVR*, Διπλωματική εργασία, Τμ Ηλεκτρολόγων Μηχανικών, ΕΜΠ, 2005
- Roger L. Tokheim, *Ψηφιακά Ηλεκτρονικά,* μτφ. Αριστοτέλης Μπιζόπουλος, 3η έκδοση, εκδ ΤΖΙΟΛΑ, Θεσσαλονίκη 1991
- Πετράκης Ζαχαρίας, *Ανάπτυξη εφαρμογής τηλεχειριζόμενου οχήματος, με πλατφόρμα Arduino*, Πτυχιακή εργασία, τμ Μηχανικών Πληροφορικής, ΤΕΙ Κρήτης, 2014
- Βασιλική Παυλή, *Η Διδασκαλία εκπαιδευτικής ρομποτικής με την χρήση μικροελεκτών(π.χ. ARDUINO,PIC)*, Πτυχιακή Εργασία, Τμ Τεχνολογίας Πληροφορικής και Τηλεποικινωνιών, ΑΤΕΙ Λάρισας, 2013
- Μαρτίνης Στέλιος, *Οικιακός Αυτοματισμός με χρήση Μικροελεγκτή*, Διπλωματική εργασία, Τμ μηχανικών πληροφορικών & επικοινωνιακών Συστημάτων, Πανεπιστήμιο Αιγαίου, 2013
- Γεώργιος Αλεξίου, *Μικροεπεξεργαστές, τόμος Γ*, εκδ ΕΑΠ, Πάτρα, 2001
- Τσιρογιάννης Π. Νικόλαος, Μαργαρίτης Κ. Βασίλειος, *Υλοποίηση εργαστηριακών ασκήσεων του μαθήματος «Αρχιτεκτονικής Η/Υ» με το με*

AVR και χρήση του με στην κατασκευή χρονισμού προειδοποιητικού λαμπτήρα, Πτυχιακή Εργασία , Τμ Μηχανικών Πληροφορικής ΤΕ, ΤΕΙ Κεντρικής Μακεδονίας, 2014

- Δρ. Τοπάλης Ευάγγελος, *Μικροϋπολογιστές ΙΙ, Μικροεπεξεργατής AVR, Εργατηριακές ασκήσεις,* Τμ Ηλεκτρολόγων Μηχανικών ΤΕ, ΤΕΙ Δυτικής Μακεδονίας

Ιστογραφία

Τεχνικό εγχειρίδιο ATmega48A/PA/88A/PA/168A/PA/328/P Διαθέσιμο στη διεύθυνση http://www.atmel.com [πρόσβαση 23-11-2015]

Σύστημα Συλλογής δεδομένων σε υδάτινο Περιβάλλον Διαθέσιμο στη διεύθυνση http://1epal-serron.ser.sch.gr [πρόσβαση 12-12-2015]

Γενικά για τους μικροελεγκτές Διαθέσιμο στη σελίδα http://wizatronics.webnode.gr/news [πρόσβαση 22-12-2015]

Learning about electronics Διαθέσιμο στη διεύθυνση http://www.learningaboutelectronics.com [πρόσβαση 25-02-2016]

Using Arduino Board in Atmel Studio Διαθέσιμο στη διεύθυνση http://www.microdigitaled.com [πρόσβαση 15-05-2016]

How to configure the Atmel AVRISP MKii to work with Arduino IDE Διαθέσιμο στη διεύθυνση http://forum.arduino.cc [πρόσβαση 01-06-2016]

Αντιμετώπιση προβλημάτων με Arduino IDE topic 316037 Διαθέσιμο στη διεύθυνση http://forum.arduino.cc [πρόσβαση 23-05-2016]

Burning the Bootloader to an Arduino Uno using Atmel Studio and the AVRISP mkII Διαθέσιμο στη διεύθυνση https://startingelectronics.org/tutorials/arduino/ [πρόσβαση 06-06-2016]

Μάθετε το Arduino Μέρος 1ο, Διαθέσιμο στη διεύθυνση https://t-h.wikispaces.com [πρόσβαση 10-02-2016]

Instruction to AVR timers Διαθέσιμο στη διεύθυνση http://maxembedded.com [πρόσβαση 10-03-2016]

Παράρτημα Α

Flip Flop[27]

Τα λογικά κυκλώματα ταξινομούνται σε δύο κατηγορίες:

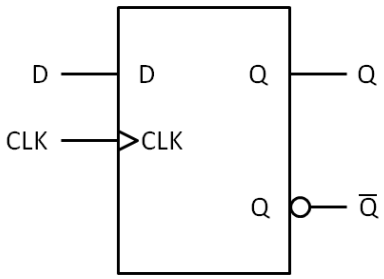
- Συνδυαστικά κυκλώματα (Πύλες AND, OR και NOT)
- Ακολουθιακά κυκλώματα

Στο συγκεκριμένο παράρτημα θα αναφερθούμε μόνο στα ακολουθιακά κυκλώματα.

Ακολουθιακά κυκλώματα

Τα ακολουθιακά λογικά κυκλώματα περιλαμβάνουν χρονισμό και μονάδες μνήμης. Το βασικό στοιχείο δόμησής τους είναι το FlipFlop (FF). Τα FF σχεδιάζονται για να εκτελέσουν λειτουργίες μετρητών, καταχωρητών ολίσθησης και άλλων ποικίλων διατάξεων μνήμης. Υπάρχουν διάφοροι τύποι FF όπως οι R-S FlipFlop, D FlipFlop, J-K FlipFlop. Παρακάτω ακολουθεί μια βασική ανάλυση για το D FlipFlop

Το DFF (σχήμα 1) έχει μόνο μια είσοδο δεδομένων (D) και μια είσοδο ρολογιού (CLK). Οι έξοδοι σημειώνονται με Q και $\bar{Q}$. Πολλές φορές ονομάζεται και FF καθυστέρησης. Η λέξη «καθυστέρηση» περιγράφει τι συμβαίνει με τα δεδομένα ή τις πληροφορίες στην είσοδο D. Τα δεδομένα (ένα 0 ή 1) στην είσοδο D, καθυστερούνται κατά ένα παλμό ρολογιού, έως να φτάσουν στην έξοδο Q.

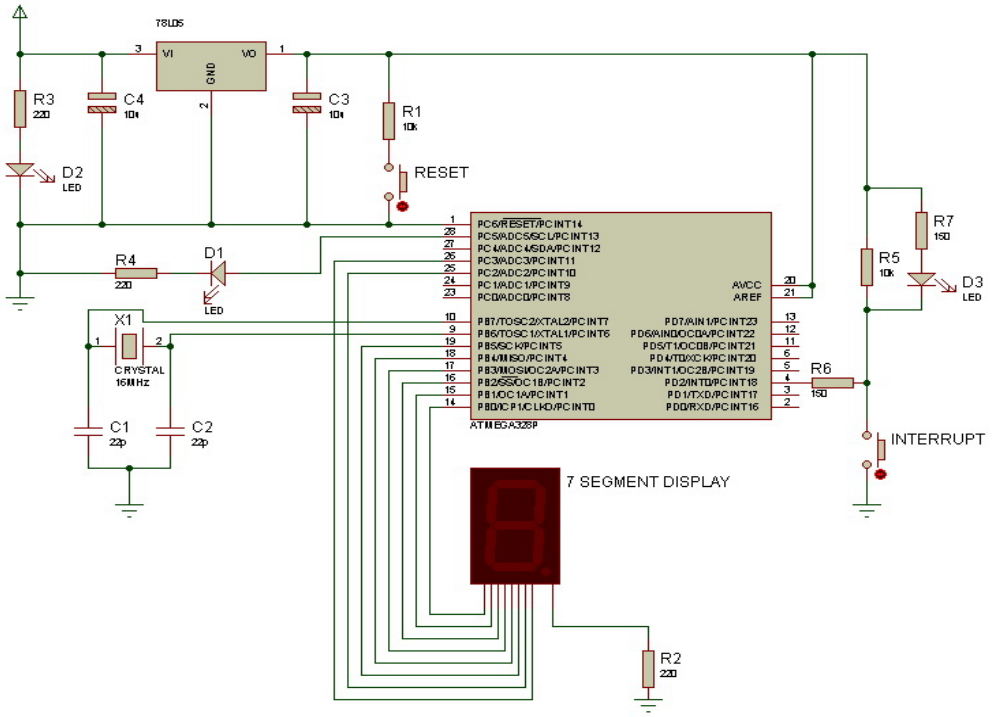


Σχήμα 33 - Λογικό σύμβολο D Flip Flop

Τα D Flip-Flop συνδέονται μαζί ώστε να εκτελούν τη λειτουργία καταχωρητών ολίσθησης και αποθήκευσης (shift και storage registers)

[27]RogerL. Tokheim, *Ψηφιακά Ηλεκτρονικά,* μτφ. Αριστοτέλης Μπιζόπουλος, 3η έκδοση, εκδ ΤΖΙΟΛΑ, Θεσσαλονίκη 1991, σελ 132- 137

Ηλεκτρονικό Κύκλωμα παραδειγμάτων

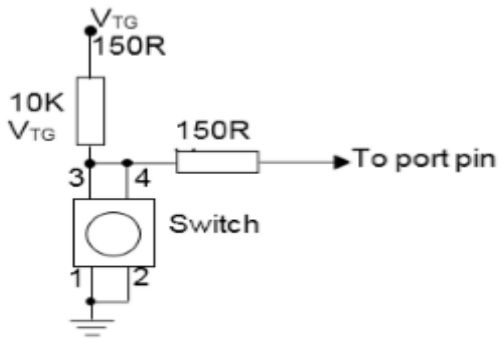


Σχήμα 34 – Ηλεκτρονικό διάγραμμα

Στο συγκεκριμένο κύκλωμα χρησιμοποιούμε ένα 7 Segment Display κοινής καθόδου

Κύκλωμα Interrupt

Χρησιμοποιούμε σαν interrupt το παρακάτω κύκλωμα

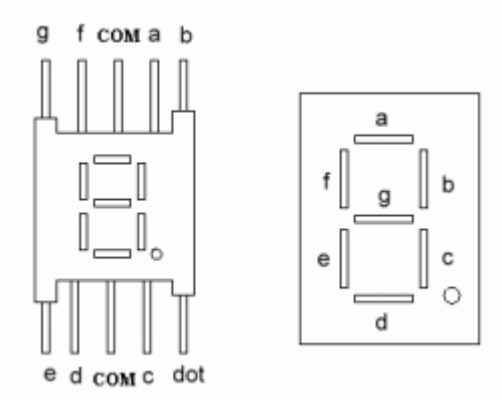


Όπου:

- V_{TG} είναι η τάση τροφοδοσίας
- PortPineίναιτοpintouINT0 (externalinterrupt 0)
- Switch είναι το pushbutton
- 150R είναι αντίσταση 150 Ohm
- 10K είναι αντίσταση 10KOhm

Σχήμα 35 – Κύκλωμα Interrupt

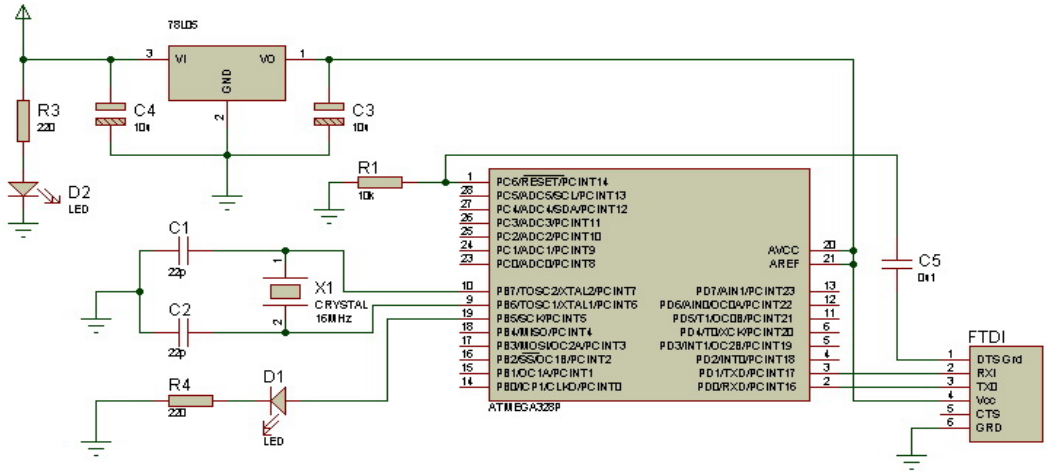
Seven Segment Display (7-Segment Display)



Σχήμα 36 - Seven Segment Display

Αποτελείται από επτά παραλληλόγραμμα Led, που το καθένα αντιστοιχίζεται με ένα από τα πρώτα 7 γράμματα της αγγλικής αλφαβήτου (βλ. διπλανή εικόνα) και έχουν μια κοινή κάθοδο ή άνοδο (pin com).

Κύκλωμα Arduino



Σχήμα 37 – Ηλεκτρονικό διάγραμμα με κύκλωμα FTDI

Το κύκλωμα FTDI που χρησιμοποιούμε είναι το FTDI Basic[28] της sparkfun και βασίζεται στο ολοκληρωμένο FT232 που μας μετατρέπει την USB σε σειριακή είσοδο για να συνδεθούμε με τον μικροελεγκτή που υποστηρίζει μόνο σειριακή επικοινωνία (ο τρόπος συνδεσμολογίας του αναφέρεται στο http://www.instructables.com/files/orig/FL3/V5P1/HFSH6WNJ/FL3V5P1HFSH6WNJ.pdf)

[28] Το σχηματικό διαγραμμα του κυκλώματος βρίσκεται στο https://www.sparkfun.com/datasheets/DevTools/Arduino/FTDI%20Basic-v21-5V.pdf

Παράρτημα Β

Καταχωρητές ειδικού σκοπού

Πίνακας 57 - Καταχωρητές ειδικού σκοπού

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
(0xC6)	UDR0	USART I/O DataRegister							
(0xC5)	UBRR0H					USART Baud Rate RegisterHigh			
(0xC4)	UBRR0L	USART Baud Rate RegisterLow							
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01/UDORD0	UCSZ00 /UCPHA0	UCPOL0
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	–
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
(0xBB)	TWDR	2-wire Serial Interface DataRegister							
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0
(0xB8)	TWBR	2-wire Serial Interface Bit RateRegister							
(0xB6)	ASSR	–	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB
(0xB5)	Reserved	–	–	–	–	–	–	–	–
(0xB4)	OCR2B	Timer/Counter2 Output Compare RegisterB							
(0xB3)	OCR2A	Timer/Counter2 Output Compare RegisterA							
(0xB2)	TCNT2	Timer/Counter2(8-bit)							
(0xB1)	TCCR2B	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20
(0x8B)	OCR1BH	Timer/Counter1 - Output Compare Register B HighByte							
(0x8A)	OCR1BL	Timer/Counter1 - Output Compare Register B LowByte							
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A HighByte							
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A LowByte							
(0x87)	ICR1H	Timer/Counter1 - Input Capture Register HighByte							
(0x86)	ICR1L	Timer/Counter1 - Input Capture Register LowByte							
(0x85)	TCNT1H	Timer/Counter1 - Counter Register HighByte							
(0x84)	TCNT1L	Timer/Counter1 - Counter Register LowByte							
(0x82)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–
(0x81)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10
(0x7F)	DIDR1	–	–	–	–	–	–	AIN1D	AIN0D
(0x7E)	DIDR0	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0
(0x7B)	ADCSRB	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
(0x79)	ADCH	ADC Data Register250							
(0x78)	ADCL	ADC Data Register250							
(0x70)	TIMSK2	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2
(0x6F)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1
(0x6E)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16
(0x6C)	PCMSK1	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0
(0x69)	EICRA	–	–	–	–	ISC11	ISC10	ISC01	ISC00
(0x68)	PCICR	–	–	–	–	–	PCIE2	PCIE1	PCIE0
(0x66)	OSCCAL	Oscillator	37						
(0x64)	PRR	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSART0	PRADC
(0x61)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0
0x3F(0x5F)	SREG	I	T	H	S	V	N	Z	C
0x3E(0x5E)	SPH	–	–	–	–	–	(SP10)[5.]	SP9	SP8
0x3D(0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
0x37(0x57)	SPMCSR	SPMIE	(RWWSB)[5.]	SIGRD	(RWWSRE)[5.]	BLBSET	PGWRT	PGERS	SPMEN
0x36(0x56)	Reserved	–	–	–	–	–	–	–	–
0x35(0x55)	MCUCR	–	BODS[6]	BODSE[6]	PUD	–	–	IVSEL	IVCE
0x34(0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF
0x33(0x53)	SMCR	–	–	–	–	SM2	SM1	SM0	SE
0x30(0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
0x2F(0x4F)	Reserved	–	–	–	–	–	–	–	–
0x2E(0x4E)	SPDR	SPI DataRegister	169						
0x2D(0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X

0x2C(0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
0x2B(0x4B)	GPIOR2	General Purpose	26						
0x2A(0x4A)	GPIOR1	General Purpose	26						
0x29(0x49)	Reserved	–	–	–	–	–	–	–	–
0x28(0x48)	OCR0B	Timer/Counter0							
0x27(0x47)	OCR0A	Timer/Counter0							
0x26(0x46)	TCNT0	Timer/Counter0(							
0x25(0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
0x24(0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
0x23(0x43)	GTCCR	TSM	–	–	–	–	–	PSRASY	PSRSYNC
0x22(0x42)	EEARH	(EEPROM Address	22						
0x21(0x41)	EEARL	EEPROM Address	22						
0x20(0x40)	EEDR	EEPROM	22						
0x1F(0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE
0x1E(0x3E)	GPIOR0	General Purpose	26						
0x1D(0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0
0x1C(0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0
0x1B(0x3B)	PCIFR	–	–	–	–	–	PCIF2	PCIF1	PCIF0
0x17(0x37)	TIFR2	–	–	–	–	–	OCF2B	OCF2A	TOV2
0x16(0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1
0x15(0x35)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0
0x0B(0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
0x0A(0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
0x09(0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
0x08(0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
0x07(0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
0x06(0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
0x05(0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0
0x04(0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
0x03(0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0

Σημείωση:

1. Για λόγους συμβατότητας με μελλοντικές συσκευές, τα δεσμευμένα bits θα πρέπει να γίνουν 0, αν υπάρχει πρόσβαση. Οι δεσμευμένες διευθύνσεις μνήμης για I/O δεν πρέπει να πειραχτούν ποτέ.
2. Οι καταχωρητές I/O μέσα στο εύρος διευθύνσεων από 0x00 – 0x1F είναι άμεσα προσβάσιμοι από τα Bits, χρησιμοποιώντας τις εντολές SBI και CBI. Σε αυτούς του καταχωρητές, η τιμή των μεμονωμένων bits μπορεί να ελεγχθεί χρησιμοποιώντας τις εντολές SBIS και SBIC.
3. Κάποια από τα status flags μηδενίζονται γράφοντας ένα λογικό 1 σε αυτά. Προσοχή, σε αντίθεση με τα περισσότερα AVRs, οι εντολές CBI και SBI λειτουργούν μόνο στο συγκεκριμένο bit, και επιπλέον μπορούν να χρησιμοποιηθούν στους καταχωρητές που περιέχουν τέτοια Status Flags. Οι εντολές CBI και SBI λειτουργούν με καταχωρητές από τη διεύθυνση 0x00 έως 0x1F.
4. Όταν χρησιμοποιούμε τις συγκεκριμένες εντολές IN και OUT η διεύθυνση 0x00 – 0x3F πρέπει να χρησιμοποιηθούν. Όταν διευθυνσιοδοτείς σε I/O καταχωρητές χρησιμοποιώντας τις εντολές LD και ST, πρέπει να προστεθεί και η διεύθυνση 0x20. Ο ATmega48A/PA/88A/PA/168A/PA/328/P είναι ένας περίπλοκος μικροελεγκτής με περισσότερες περιφερειακές μονάδες από αυτές που μπορεί να υποστηριχτούν μέσα στις 64 δεσμευμένες θέσεις του opcode για τις εντολές IN και OUT. Για χώρο I/O από τη διεύθυνση 0x60 – 0xFF στη SRAM μόνο οι εντολές ST/STS/STD και LD/LDS/LDD μπορούν να χρησιμοποιηθούν.
5. Ισχύει μόνο για ATmega88A/88PA/168A/168PA/328/328P

6. Οι εντολές BODS και BODSE είναι διαθέσιμες για εξαρτήματα pico Power στον ATmega48PA/88PA/328P γράφοντας ένα λογικό 1 σε αυτά. Προσοχή, σε αντίθεση με τα περισσότερα AVRs, οι εντολές CBI και SBI λειτουργούν μόνο στο συγκεκριμένο bit, και επιπλέον μπορούν να χρησιμοποιηθούν στους καταχωρητές που περιέχουν τέτοια Status Flags. Οι εντολές CBI και SBI λειτουργούν με καταχωρητές από τη διεύθυνση 0x00 έως 0x1F.
7. Όταν χρησιμοποιούμε τις συγκεκριμένες εντολές IN και OUT η διεύθυνση 0x00 – 0x3F πρέπει να χρησιμοποιηθούν. Όταν διευθυνσιοδοτείς σε I/O καταχωρητές χρησιμοποιώντας τις εντολές LD και ST, πρέπει να προστεθεί και η διεύθυνση 0x20. Ο ATmega48A/PA/88A/PA/168A/PA/328/P είναι ένας περίπλοκος μικροελεγκτής με περισσότερες περιφερειακές μονάδες από αυτές που μπορεί να υποστηριχτούν μέσα στις 64 δεσμευμένες θέσεις του opcode για τις εντολές IN και OUT. Για χώρο I/O από τη διεύθυνση 0x60 – 0xFF στη SRAM μόνο οι εντολές ST/STS/STD και LD/LDS/LDD μπορούν να χρησιμοποιηθούν
8. Ισχύει μόνο για ATmega88A/88PA/168A/168PA/328/328P
9. Οι εντολές BODSκαι BODSEείναι διαθέσιμες για εξαρτήματα pico Power στον ATmega48PA/88PA/328P

Παράρτημα Γ

Περίληψη Σετ Εντολών ATmega168/328

Πίνακας 58 - Σετ εντολών

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd,Rr	Add twoRegisters	Rd←Rd + Rr	Z,C,N,V,H	1
ADC	Rd,Rr	Add with Carry twoRegisters	Rd←Rd + Rr +C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate toWord	Rdh:Rd←Rdh:Rdl +K	Z,C,N,V,S	2
SUB	Rd,Rr	Subtract twoRegisters	Rd←Rd -Rr	Z,C,N,V,H	1
SUBI	Rd,K	Subtract Constant fromRegister	Rd←Rd - K	Z,C,N,V,H	1
SBC	Rd,Rr	Subtract with Carry twoRegisters	Rd←Rd - Rr -C	Z,C,N,V,H	1
SBCI	Rd,K	Subtract with Carry Constant fromReg.	Rd←Rd - K -C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate fromWord	Rdh:Rdl←Rdh:Rdl -K	Z,C,N,V,S	2
AND	Rd,Rr	Logical ANDRegisters	Rd←Rd •Rr	Z,N,V	1
ANDI	Rd,K	Logical AND Register andConstant	Rd←Rd •K	Z,N,V	1
OR	Rd,Rr	Logical ORRegisters	Rd ←Rd v Rr	Z,N,V	1
ORI	Rd,K	Logical OR Register andConstant	Rd←Rd v K	Z,N,V	1
EOR	Rd,Rr	Exclusive ORRegisters	Rd ←Rd ⊕Rr	Z,N,V	1
COM	Rd	One'sComplement	Rd←0xFF - Rd	Z,C,N,V	1
NEG	Rd	Two'sComplement	Rd←0x00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) inRegister	Rd ←Rd vK	Z,N,V	1
CBR	Rd,K	Clear Bit(s) inRegister	Rd←Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ←Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ←Rd −1	Z,N,V	1
TST	Rd	Test for Zero orMinus	Rd ←Rd •Rd	Z,N,V	1
CLR	Rd	ClearRegister	Rd ←Rd ⊕Rd	Z,N,V	1
SER	Rd	SetRegister	Rd←0xFF	None	1
MUL	Rd,Rr	MultiplyUnsigned	R1:R0←Rd xRr	Z,C	2
MULS	Rd,Rr	MultiplySigned	R1:R0 ←Rd xRr	Z,C	2
MULSU	Rd,Rr	Multiply Signed withUnsigned	R1:R0←Rd xRr	Z,C	2
FMUL	Rd,Rr	Fractional MultiplyUnsigned	R1:R0 ← (Rd x Rr) <<1	Z,C	2
FMULS	Rd,Rr	Fractional MultiplySigned	R1:R0 ← (Rd x Rr) <<1	Z,C	2
FMULSU	Rd,Rr	Fractional Multiply Signed withUnsigned	R1:R0 ← (Rd x Rr) <<1	Z,C	2
BRANCHINSTRUCTIONS					
RJMP	k	RelativeJump	PC←PC + k +1	None	2
IJMP		Indirect Jump to(Z)	PC←Z	None	2
JMP[1]	k	DirectJump	PC←k	None	3
RCALL	k	Relative SubroutineCall	PC ←PC + k +1	None	3
ICALL		Indirect Call to(Z)	PC ←Z	None	3
CALL[1]	k	Direct SubroutineCall	PC ←k	None	4
RET		SubroutineReturn	PC ←STACK	None	4
RETI		InterruptReturn	PC ←STACK	I	4
CPSE	Rd,Rr	Compare, Skip ifEqual	if (Rd = Rr) PC ←PC + 2 or3	None	1/2/3
CP	Rd,Rr	Compare	Rd −Rr	Z,N,V,C,H	1
CPC	Rd,Rr	Compare withCarry	Rd −Rr −C	Z,N,V,C,H	1
CPI	Rd,K	Compare Register withImmediate	Rd −K	Z,N,V,C,H	1
SBRC	Rr,b	Skip if Bit in RegisterCleared	if (Rr(b)=0) PC ←PC + 2 or3	None	1/2/3
SBRS	Rr,b	Skip if Bit in Register isSet	if (Rr(b)=1) PC ←PC + 2 or3	None	1/2/3
SBIC	P,b	Skip if Bit in I/O RegisterCleared	if (P(b)=0) PC←PC + 2 or3	None	1/2/3
SBIS	P,b	Skip if Bit in I/O Register isSet	if (P(b)=1) PC←PC + 2 or3	None	1/2/3
BRBS	s,k	Branch if Status FlagSet	if (SREG(s) = 1) then PC←PC+k +1	None	1/2
BRBC	s,k	Branch if Status FlagCleared	if (SREG(s) = 0) then PC←PC+k +1	None	1/2
BREQ	k	Branch ifEqual	if (Z = 1) then PC ←PC + k +1	None	1/2
BRNE	k	Branch if NotEqual	if (Z = 0) then PC ←PC + k +1	None	1/2
BRCS	k	Branch if CarrySet	if (C = 1) then PC ←PC + k +1	None	1/2
BRCC	k	Branch if CarryCleared	if (C = 0) then PC ←PC + k +1	None	1/2
BRSH	k	Branch if Same orHigher	if (C = 0) then PC ←PC + k +1	None	1/2
BRLO	k	Branch ifLower	if (C = 1) then PC ←PC + k +1	None	1/2
BRMI	k	Branch ifMinus	if (N = 1) then PC←PC + k +1	None	1/2
BRPL	k	Branch ifPlus	if (N = 0) then PC ←PC + k +1	None	1/2
BRGE	k	Branch if Greater or Equal,Signed	if (N ⊕V= 0) then PC←PC + k +1	None	1/2
BRLT	k	Branch if Less Than Zero,Signed	if (N ⊕V= 1) then PC←PC + k +1	None	1/2
BRHS	k	Branch if Half Carry FlagSet	if (H = 1) then PC ←PC + k +1	None	1/2
BRHC	k	Branch if Half Carry FlagCleared	if (H = 0) then PC ←PC + k +1	None	1/2
BRTS	k	Branch if T FlagSet	if (T = 1) then PC ←PC + k +1	None	1/2
BRTC	k	Branch if T FlagCleared	if (T = 0) then PC ←PC + k +1	None	1/2
BRVS	k	Branch if Overflow Flag isSet	if (V = 1) then PC←PC + k +1	None	1/2
BRVC	k	Branch if Overflow Flag isCleared	if (V = 0) then PC ←PC + k +1	None	1/2

BRIE	k	Branch if InterruptEnabled	if (I = 1) then PC ←PC + k +1	None	1/2
BRID	k	Branch if InterruptDisabled	if (I = 0) then PC ←PC + k +1	None	1/2
BIT AND BIT-TESTINSTRUCTIONS					
SBI	P,b	Set Bit in I/ORegister	I/O(P,b) ←1	None	2
CBI	P,b	Clear Bit in I/ORegister	I/O(P,b) ←0	None	2
LSL	Rd	Logical ShiftLeft	Rd(n+1) ←Rd(n), Rd(0) ←0	Z,C,N,V	1
LSR	Rd	Logical ShiftRight	Rd(n) ←Rd(n+1), Rd(7)←0	Z,C,N,V	1
ROL	Rd	Rotate Left ThroughCarry	Rd(0)←C,Rd(n+1)←Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right ThroughCarry	Rd(7)←C,Rd(n)←Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic ShiftRight	Rd(n) ←Rd(n+1),n=0...6	Z,C,N,V	1
SWAP	Rd	SwapNibbles	Rd(3...0)←Rd(7...4),Rd(7...4)←Rd(3..	None	1
BSET	s	FlagSet	SREG(s) ←1	SREG(s)	1
BCLR	s	FlagClear	SREG(s) ←0	SREG(s)	1
BST	Rr,b	Bit Store from Register toT	T ←Rr(b)	T	1
BLD	Rd,b	Bit load from T toRegister	Rd(b) ←T	None	1
SEC		SetCarry	C ←1	C	1
CLC		ClearCarry	C ←0	C	1
SEN		Set NegativeFlag	N←1	N	1
CLN		Clear NegativeFlag	N ←0	N	1
SEZ		Set ZeroFlag	Z←1	Z	1
CLZ		Clear ZeroFlag	Z←0	Z	1
SEI		Global InterruptEnable	I ←1	I	1
CLI		Global InterruptDisable	I ←0	I	1
SES		Set Signed TestFlag	S ←1	S	1
CLS		Clear Signed TestFlag	S ←0	S	1
SEV		Set Twos ComplementOverflow.	V←1	V	1
CLV		Clear Twos ComplementOverflow	V ←0	V	1
SET		Set T inSREG	T ←1	T	1
CLT		Clear T inSREG	T←0	T	1
SEH		Set Half Carry Flag inSREG	H ←1	H	1
CLH		Clear Half Carry Flag inSREG	H ←0	H	1
DATA TRANSFERINSTRUCTIONS					
MOV	Rd,Rr	Move BetweenRegisters	Rd←Rr	None	1
MOVW	Rd,Rr	Copy RegisterWord	Rd+1:Rd←Rr+1:Rr	None	1
LDI	Rd,K	LoadImmediate	Rd ←K	None	1
LD	Rd,X	LoadIndirect	Rd ← (X)	None	2
LD	Rd,X+	Load Indirect andPost-Inc.	Rd ←X), X ←X + 1	None	2
LD	Rd, -X	Load Indirect andPre-Dec.	X ←X - 1, Rd←X)	None	2
LD	Rd,Y	LoadIndirect	Rd ← (Y)	None	2
LD	Rd,Y+	Load Indirect andPost-Inc.	Rd ← (Y), Y ←Y + 1	None	2
LD	Rd, -Y	Load Indirect andPre-Dec.	Y ←Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect withDisplacement	Rd ← (Y + q)	None	2
LD	Rd,Z	LoadIndirect	Rd ←(Z)	None	2
LD	Rd,Z+	Load Indirect andPost-Inc.	Rd←(Z), Z ←Z+1	None	2
LD	Rd,-Z	Load Indirect andPre-Dec.	Z ←Z - 1, Rd←(Z)	None	2
LDD	Rd,Z+q	Load Indirect withDisplacement	Rd ← (Z + q)	None	2
LDS	Rd,k	Load Direct fromSRAM	Rd ← (k)	None	2
ST	X,Rr	StoreIndirect	(X) ←Rr	None	2
ST	X+,Rr	Store Indirect andPost-Inc.	(X) ←Rr, X ←X +1	None	2
ST	- X,Rr	Store Indirect andPre-Dec.	X←X - 1, (X) ←Rr	None	2
ST	Y,Rr	StoreIndirect	(Y) ←Rr	None	2
ST	Y+,Rr	Store Indirect andPost-Inc.	(Y) ←Rr, Y ←Y + 1	None	2
ST	- Y,Rr	Store Indirect andPre-Dec.	Y ←Y - 1, (Y) ←Rr	None	2
STD	Y+q,Rr	Store Indirect withDisplacement	(Y + q) ←Rr	None	2
ST	Z,Rr	StoreIndirect	(Z) ←Rr	None	2
ST	Z+,Rr	Store Indirect andPost-Inc.	(Z) ←Rr, Z ←Z +1	None	2
ST	-Z,Rr	Store Indirect andPre-Dec.	Z ←Z - 1, (Z)←Rr	None	2
STD	Z+q,Rr	Store Indirect withDisplacement	(Z + q)←Rr	None	2
STS	k,Rr	Store Direct toSRAM	(k) ←Rr	None	2
LPM		Load ProgramMemory	R0 ← (Z)	None	3
LPM	Rd,Z	Load ProgramMemory	Rd ←(Z)	None	3
LPM	Rd,Z+	Load Program Memory andPost-Inc	Rd ← (Z), Z ←Z+1	None	3
SPM		Store ProgramMemory	(Z) ←R1:R0	None	-
IN	Rd,P	InPort	Rd ←P	None	1
OUT	P,Rr	OutPort	P ←Rr	None	1
PUSH	Rr	Push Register onStack	STACK←Rr	None	2
POP	Rd	Pop Register fromStack	Rd←STACK	None	2
MCU CONTROLINSTRUCTIONS					
NOP		NoOperation		None	1
SLEEP		Sleep	(see specific descr. for Sleepfunction)	None	1
WDR		WatchdogReset	(see specific descr. forWDR/timer)	None	1
BREAK		Break	For On-chip DebugOnly	None	N/A