ΔΗΜΟΚΡΙΤΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΡΑΚΗΣ
DEMOCRITUS UNIVERSITY OF THRACE

ΜΑΘΗΜΑ
ΟΡΓΑΝΩΣΗ ΥΠΟΛΟΓΙΣΤΩΝ (206ΕΥΥΚ)
ΠΠΣ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΕΑΡΙΝΟ 2023-2024

Διάλεξη Νο Χ:

Εισαγωγή στους μικροελεγκτές RISC AVR - Arduino (Μέρος Ι)

Δ. Καραμπατζάκης, Επίκουρος Καθηγητής

email. dkara@cs.duth.gr

iPES Lab

Πληροφορίες για την επικοινωνία με τον διδάσκοντα

Διδάσκων:

Δημήτρης Καραμπατζάκης, Επίκουρος Καθηγητής

Αναλογικά και Ψηφιακά Ηλεκτρονικά Συστήματα

Μέλος Εργαστηρίου Βιομηχανικών και Εκπαιδευτικών Ενσωματωμένων Συστημάτων

Επικοινωνία:

email. dkara@cs.duth.gr

web. http://www.internetofthings.gr/

Ώρες Γραφείου:

μετά από συνεννόηση με email στο γραφείο ΦΕ 315 (πάνω από αιθ. Α1)

Κύριο Σύγγραμμα Μαθήματος (ΕΥΔΟΞΟΣ)

Οργάνωση και Σχεδίαση Υπολογιστών

Συγγραφέας: Πογαρίδης Δημήτριος

Έτος Έκδοσης: 2019

Κωδικός στον Εύδοξο: **86192986**

Λογισμικό - Αναπτυξιακό

- Α’ μέρος μαθήματος (CISC):
  - Assembly για τον Motorola68000
  - Λογισμικό easy68k http://www.easy68k.com/

- Β’ μέρος μαθήματος (RISC):
  - Υλοποίηση σχεδιάσεων σε αναπτυξιακό Arduino (προαιρετική αγορά του σύμφωνα με τις οδηγίες)
  - Λογισμικό Arduino IDE: https://www.arduino.cc/en/Main/Software
  - Η γλώσσα προγραμματισμού (C++) και οι εντολές που υποστηρίζει είναι διαθέσιμες στο: https://www.arduino.cc/reference/en/

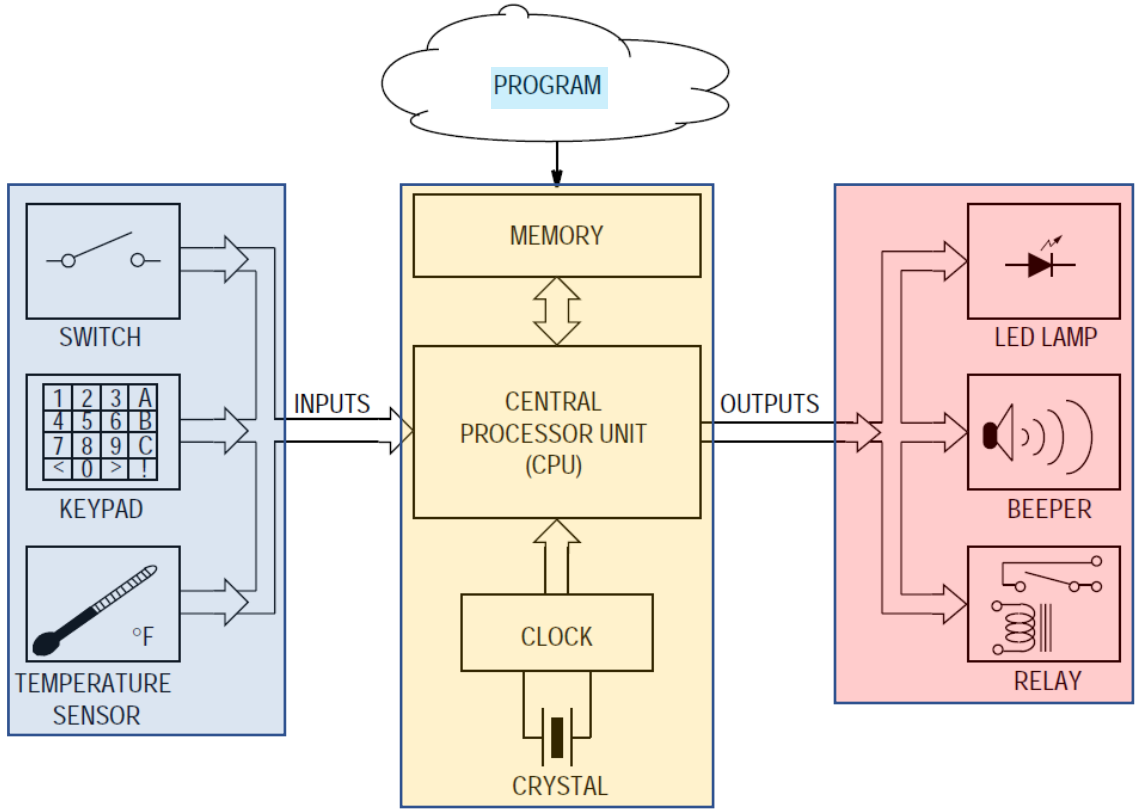
Τι είναι ένα Ενσωματωμένο Σύστημα;

Τα **Ενσωματωμένα Συστήματα** είναι υπολογιστικά συστήματα (συσκευές) που περιέχουν ένα προγραμματιζόμενο επεξεργαστή που λειτουργεί ως ελεγκτής (microcontroller) και επιτελούν ειδικές εργασίες (λειτουργίες).

Το ενσωματωμένο σύστημα δε διαθέτει επεξεργαστή γενικού σκοπού, έχει όμως μνήμη και δυνατότητες διασύνδεσης με συστήματα διεπαφής εισόδου/εξόδου.

Επειδή τα συστήματα αυτά είναι ειδικού σκοπού στην αγορά υπάρχει πληθώρα ολοκληρωμένων και συστημάτων για κάθε εφαρμογή.

Τα ενσωματωμένα συστήματα έχουν τη δική τους Αρχιτεκτονική Υπολογιστών (συνήθως RISC) και ανάλογα με την τεχνολογία υλοποίησης εμφανίζουν παραλλαγές στην οργάνωσή τους.

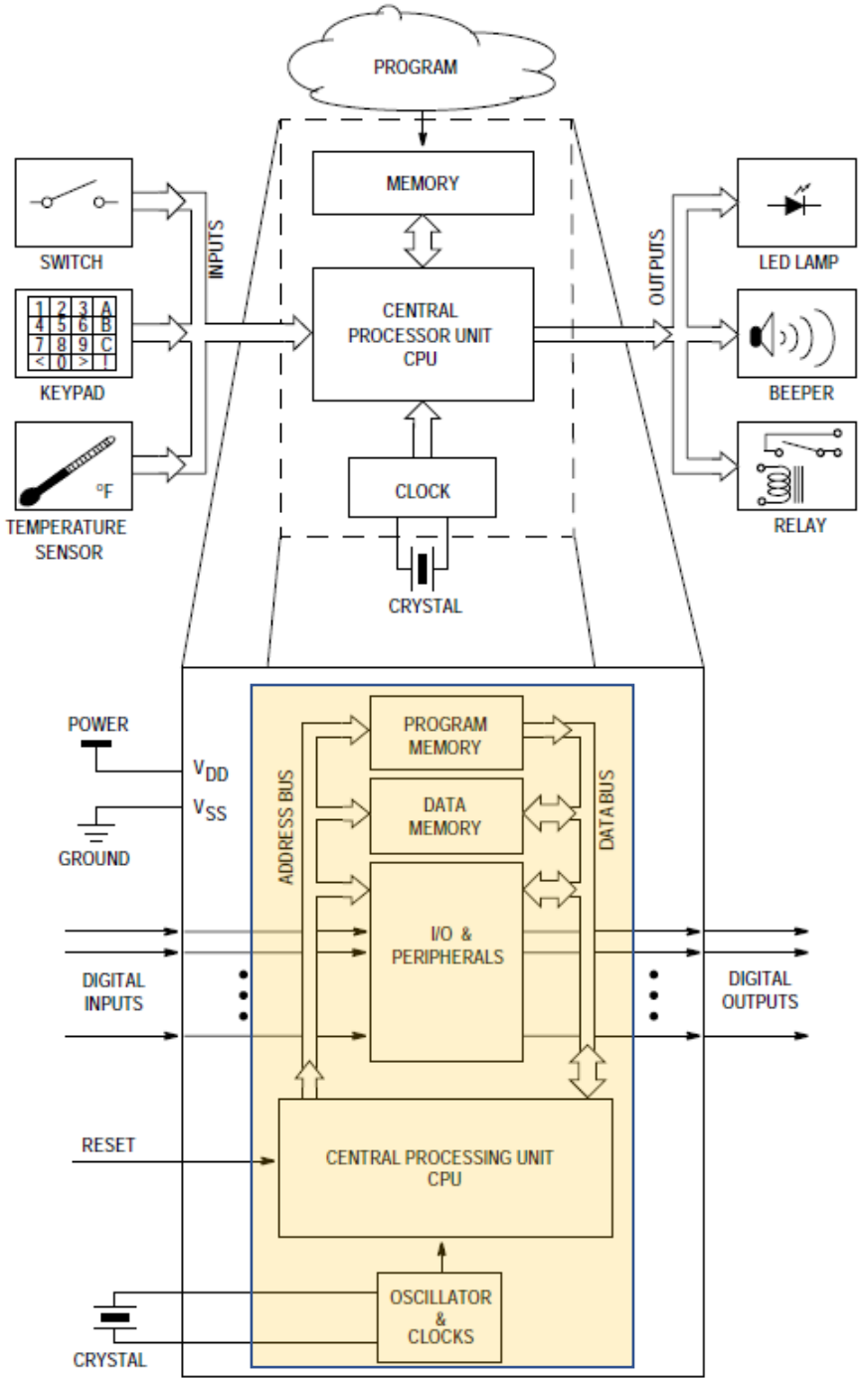


Βασικά περί ΕΣ…

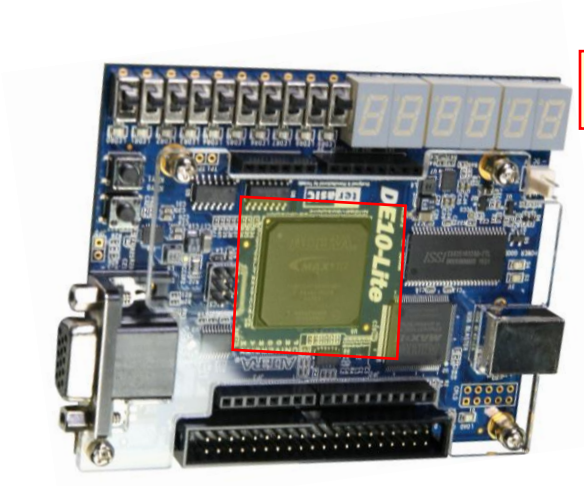
Τι είναι μια περιφερειακή συσκευή Εισόδου/Εξόδου?
Είναι μια συσκευή που είναι προσαρμοσμένη σε έναν μικροελεγκτή (ή επεξεργαστή), που δεν είναι μέρος αυτού, και η λειτουργία της είναι εξαρτημένη από τον μικροελεγκτή.

Πως ελέγχουμε τις περιφερειακές συσκευές σε ένα ενσ. σύστημα?

- Αν ελέγχουμε τις περιφερειακές συσκευές με **διακριτά ηλεκτρονικά (ψηφιακές πύλες, τρανζίστορ)** ή **ολοκληρωμένα ειδικών εφαρμογών (ASIC)** ή **προγραμματιζόμενες διατάξεις υλικού (FPGA)** **ο έλεγχος γίνεται με υλικό (hardware)**.
- Αν ελέγχουμε τις περιφερειακές συσκευές **με μικροελεγκτή** ο έλεγχος **γίνεται με λογισμικό (software).**
- Αν ελέγχουμε τις περιφερειακές συσκευές με ένα **System on a Chip (SoC)** τότε ο έλεγχος μπορεί γίνεται **με λογισμικό ή / και υλικό (software/hardware)**.

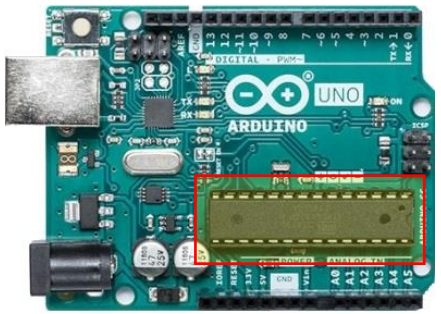


Intel FPGA University Program



FPGA

Arduino Uno-Zero

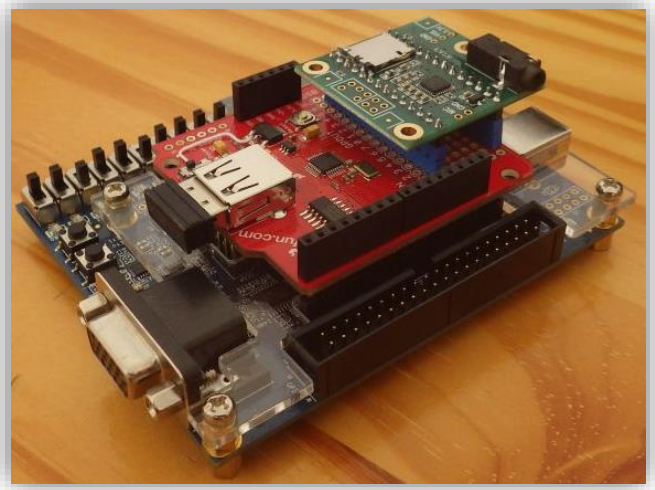


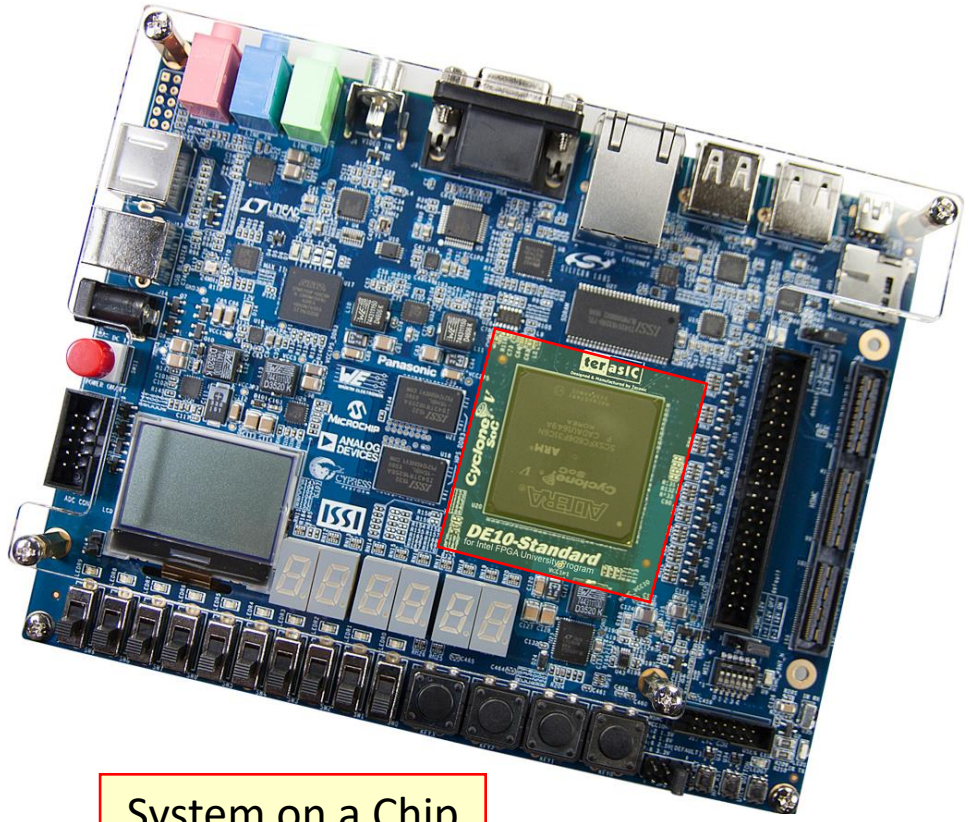
µcontroller

DE10-Lite

MAX10 FPGA

Arduino connector





System on a Chip

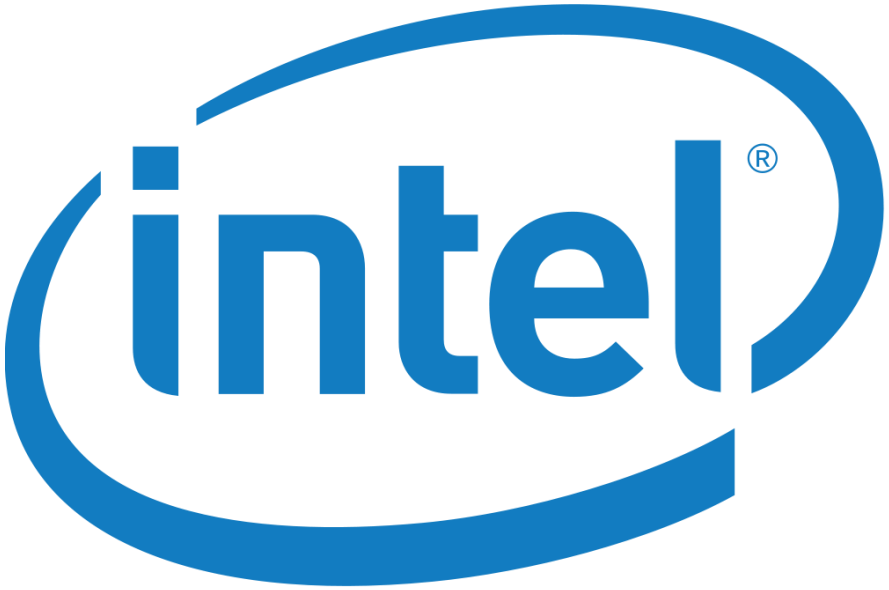
DE10-Standard

Cyclone V FPGA SoC

Dual-core Cortex-A9 HPS ARM

RISC vs. CISC

vs.



RISC-V: The Free and Open RISC Instruction Set Architecture

RISC vs. CISC

RISC	CISC
Έμφαση στο Λογισμικό (software)	Έμφαση στο Υλικό (hardware)
Εντολές ενός κύκλου και σταθερού μήκους	Σύνθετο Σετ εντολών πολλών κύκλων μηχανής
Register-to-register: Τα “Load” και “Store” είναι ανεξάρτητες εντολές	Memory-to-memory: Τα “Load” και “Store” εμπλέκονται στις εντολές
Μεγάλο μέγεθος κώδικα, χαμηλούς κύκλους ανά δευτερόλεπτο	Μικρό μέγεθος κώδικα, υψηλούς κύκλους ανά δευτερόλεπτο
Τα Transistors χρησιμοποιούνται για την υλοποίηση καταχωρητών μνήμης	Τα Transistors χρησιμοποιούνται για την υλοποίηση σύνθετων εντολών

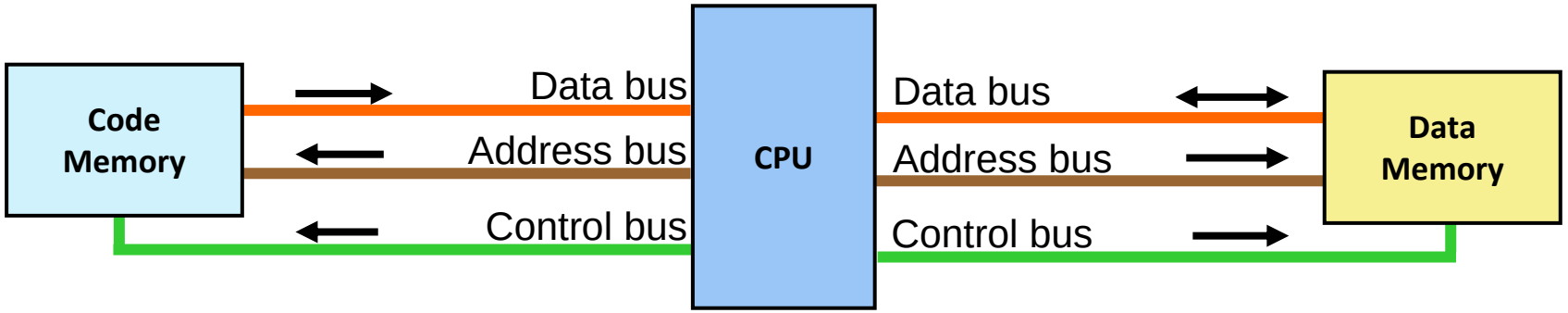
- Η μάχη RISC vs. CISC γίνεται δημοφιλές θέμα στο διαδίκτυο κάθε φορά που η Intel (CISC) ή η Apple/ARM (RISC) εισαγάγει έναν νέο επεξεργαστή στην αγορά.
- Τα περισσότερα PC είναι CISC. Για παράδειγμα Intel και AMD CPU’s.
- Αρκετοί ισχυρίζονται ότι η αρχιτεκτονική RISC είναι η αρχιτεκτονική του μέλλοντος (η RISCV είναι η τελευταία ελπιδοφόρα έκδοση).
- Αλλά αν και οι επεξεργαστές RISC είναι στην αγορά από το 1980, δεν κατάφεραν να βγάλουν από την αγορά τους CISC, και η αλήθεια είναι ότι και οι δύο έχουν πλέον «κλεμμένα» χαρακτηριστικά ο ένας από τον άλλο.

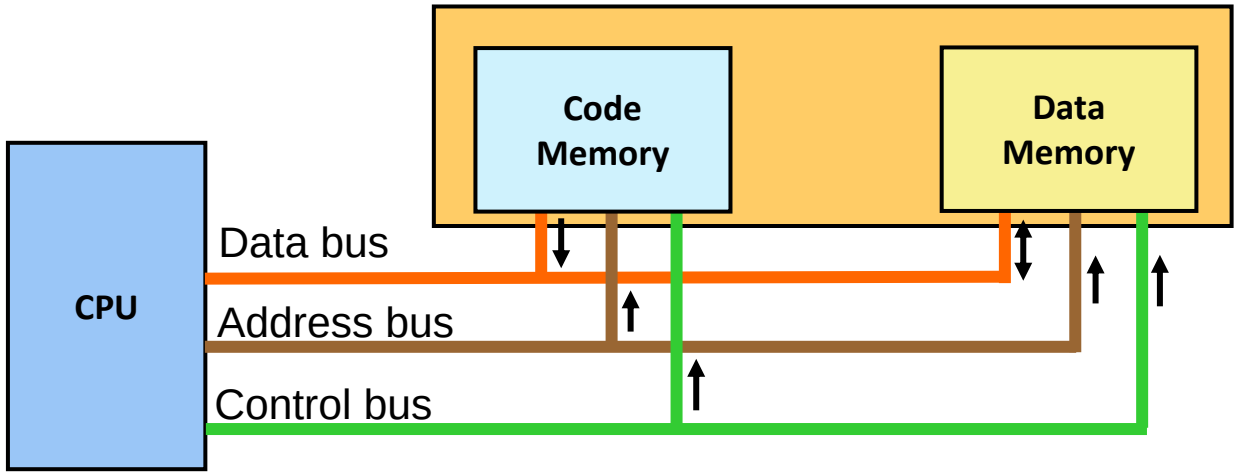
Harvard vs. Von Neumann (Princeton) Architectures

Harvard architecture

(RISC)

AVR-ARM





Von Neumann (Princeton) architecture

(CISC)

M68000-Intel-AMD

AVR RISC

Η **Αρχιτεκτονική Συνόλου Εντολών** στους υπολογιστές **RISC (Reduced Instruction Set Computer)** και τους μεταγλωττιστές τους έχει αναπτυχθεί με τέτοιο τρόπο ώστε να βελτιστοποιεί και τα δύο. Σύμφωνα με αυτό τον στόχο ένας σχετικά υψηλής απόδοσης επεξεργαστής μπορεί να υπάρξει με τη «μείωση» του φόρτου κάθε μιας από τις εντολές που οδηγεί σε πιο απλό υλικό (Hardware) που είναι μικρότερό, ταχύτερο και φθηνότερο.

Είναι συνηθισμένο οι εντολές να είναι σταθερού μήκους 16-bit. Οι εντολές έχουν από κανέναν μέχρι δύο 2 τελεστές. Πολύ από τους σύγχρονους RISC επεξεργαστές έχουν μέχρι και τρεις (3) τελεστές. Το Αρχείο Καταχωρητών (Register File) του AVR CPU περιέχει 32 καταχωρητές Γενικού Σκοπού των 8-bit που συνήθως είναι ορθογώνιοι/πανομοιότυποι (Orthogonal or identical) και οι εντολές μπορούν να χρησιμοποιούν οποιονδήποτε καταχωρητή με αποτέλεσμα να απλοποιείται ο σχεδιασμός του μεταγλωττιστή.

AVR RISC

Οι RISC επεξεργαστές υλοποιούν στην αρχιτεκτονική τους την πρόσβαση στη μνήμη με το μοντέλο Φόρτωση-Αποθήκευση (**Load-Store Memory Access**).

Αρχικά και για να μπορέσεις να προχωρήσεις πρέπει να φορτώσεις τα δεδομένα από τη μνήμη σε έναν από τους καταχωρητές και μετά να χρησιμοποιήσεις εντολές καταχωρητή-καταχωρητή (**Register-Register**) για να χειριστείς / επεξεργαστείς τα δεδομένα.

Τέλος, το αποτέλεσμα αποθηκεύεται πίσω στη μνήμη. Για παράδειγμα η εντολή ADD σε μια RISC CPU θα πρέπει τους δύο τελεστές να τους έχεις σε καταχωρητές, ενώ σε μια CISC θα μπορούσε ένας από δύο τελεστές να είναι στη μνήμη.

AVR RISC

Τροποποιημένο Μοντέλο Μνήμης Αρχιτεκτονικής Harvard

(Modified Harvard Memory Model)

Σύμφωνα με το μοντέλο μνήμης Harvard διαχωρίζεται το πρόγραμμα και τα δεδομένα σε ξεχωριστά φυσικά συστήματα μνήμης (Flash και SRAM) τα οποία εμφανίζονται προγραμματιστικά και σε διαφορετικούς χώρους διευθύνσεων (Address Spaces). Στο modified Harvard μοντέλο υπάρχει η δυνατότητα να διαβάζεις/γράφεις δεδομένα από/προς τη μνήμη του προγράμματος (Flash) χρησιμοποιώντας ειδικές εντολές.

Για παράδειγμα σε μια εντολή Immediate π.χ. ldi r16, 0x23 τα άμεσα δεδομένα και ο τελεστής κωδικοποιούνται στην ίδια εντολή και αποθηκεύονται στη μνήμη προγράμματος (Flash), με αποτέλεσμα τα δεδομένα να φιλοξενούνται στη μνήμη του προγράμματος. Στο μοντέλο Von Neumann (Princeton) έχει μόνο ένα χώρο μνήμης σε ένα κοινό σύστημα μνήμης όπου φιλοξενείται το πρόγραμμα και τα δεδομένα.

Αρχιτεκτονική και Οργάνωση Υπολογιστών

Η **Αρχιτεκτονική Υπολογιστών** αφορά παραμέτρους του συστήματος που είναι προσβάσιμες στους προγραμματιστές. Αφορά σχεδιαστικές επιλογές για τις δυνατότητες του συστήματος (Σύνολο Εντολών, Μνήμη, I/O, Μεθόδους Διευθυνσιοδότησης, Αναπαράσταση δεδομένων, κλπ.). Επίσης, αφορά και σχεδιαστικά θέματα όπως αν η αρχιτεκτονική υποστηρίζει την εντολή του πολλαπλασιασμού?

Η **Οργάνωση Υπολογιστών (ή Μικροαρχιτεκτονική)** αφορά την υλοποίηση των παραμέτρους της αρχιτεκτονικής του συστήματος σε πραγματικό υλικό Hardware. Η επιλογή της υλοποίησης σε υλικό δεν πρέπει να απασχολεί τον προγραμματιστή και για το λόγο αυτό οι λειτουργίες πρέπει να δίνονται με διαφανή (transparent) τρόπο προς τα πάνω. Στο επίπεδο αυτό μας ενδιαφέρει με τι τεχνική και τεχνολογία θα υλοποιηθεί η κάθε μονάδα του συστήματος για να υλοποιηθεί η αρχιτεκτονική.

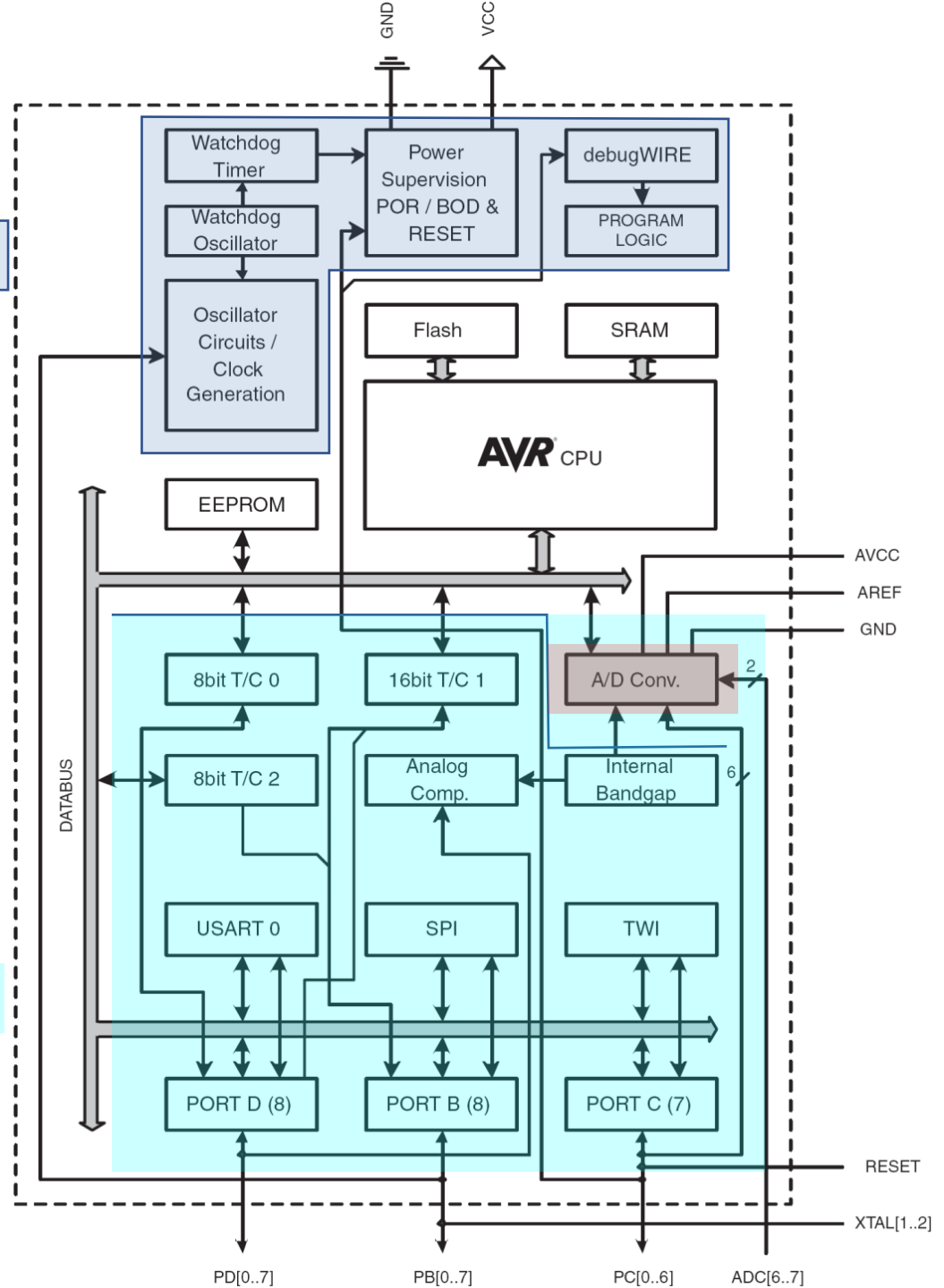
Πολλοί κατασκευαστές επεξεργαστών προσφέρουν μοντέλα υπολογιστικών συστημάτων που ακολουθούν την ίδια **Αρχιτεκτονική** αλλά υλοποιούνται με διαφορές στην **Οργάνωση** τους. Αυτό δημιουργεί και την αποκαλούμενη συμβατότητα προς τα πίσω των συστημάτων (backward compatibility), την όποια έχουν οι αρχιτεκτονικές Intel x86. Μια **Αρχιτεκτονική** μπορεί να επιβιώσει πολλά χρόνια αλλά η **Οργάνωση** αλλάζει κάθε φορά που βελτιώνονται οι τεχνικές σχεδίασης και η τεχνολογία υλοποίησης.

ATmega328P - Αρχιτεκτονική

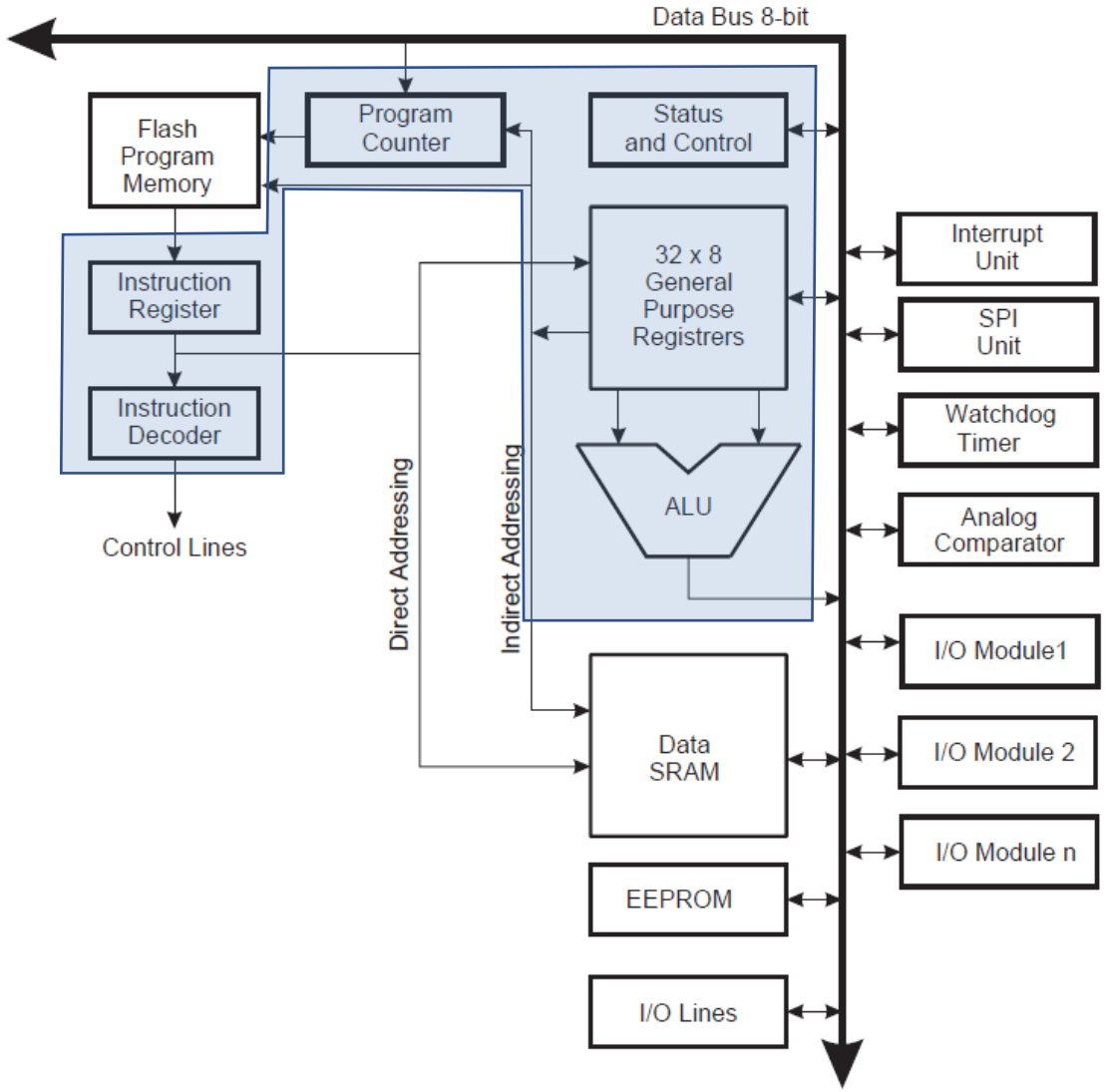
Το σύστημα τροφοδοσίας (Power) και Χρονισμού (Clock).

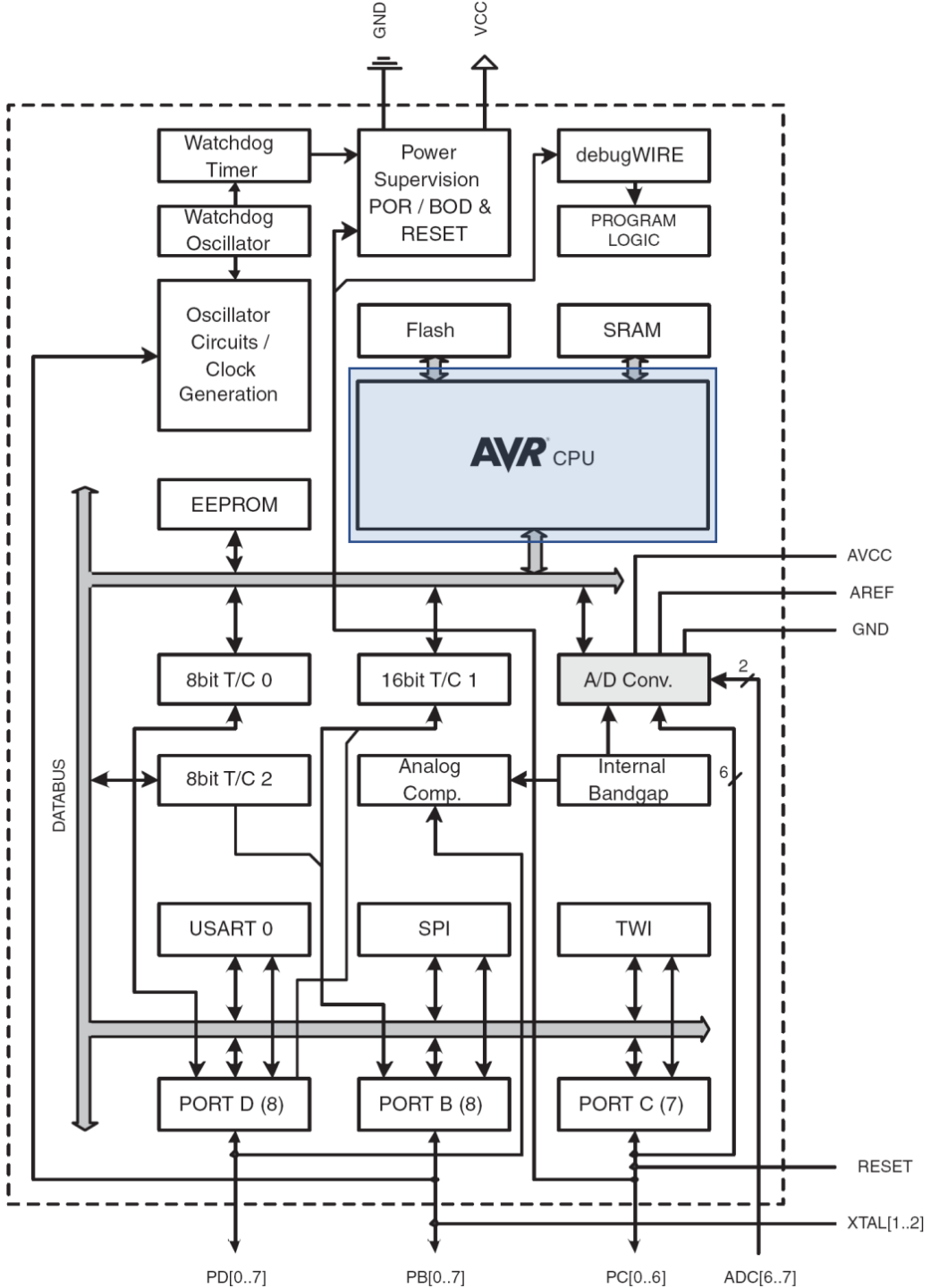
Μετατροπέας A/D (Analog to Digital Converter)

Μονάδες I/O και ειδικές λειτουργίες



ATmega328P – AVR RISC CPU

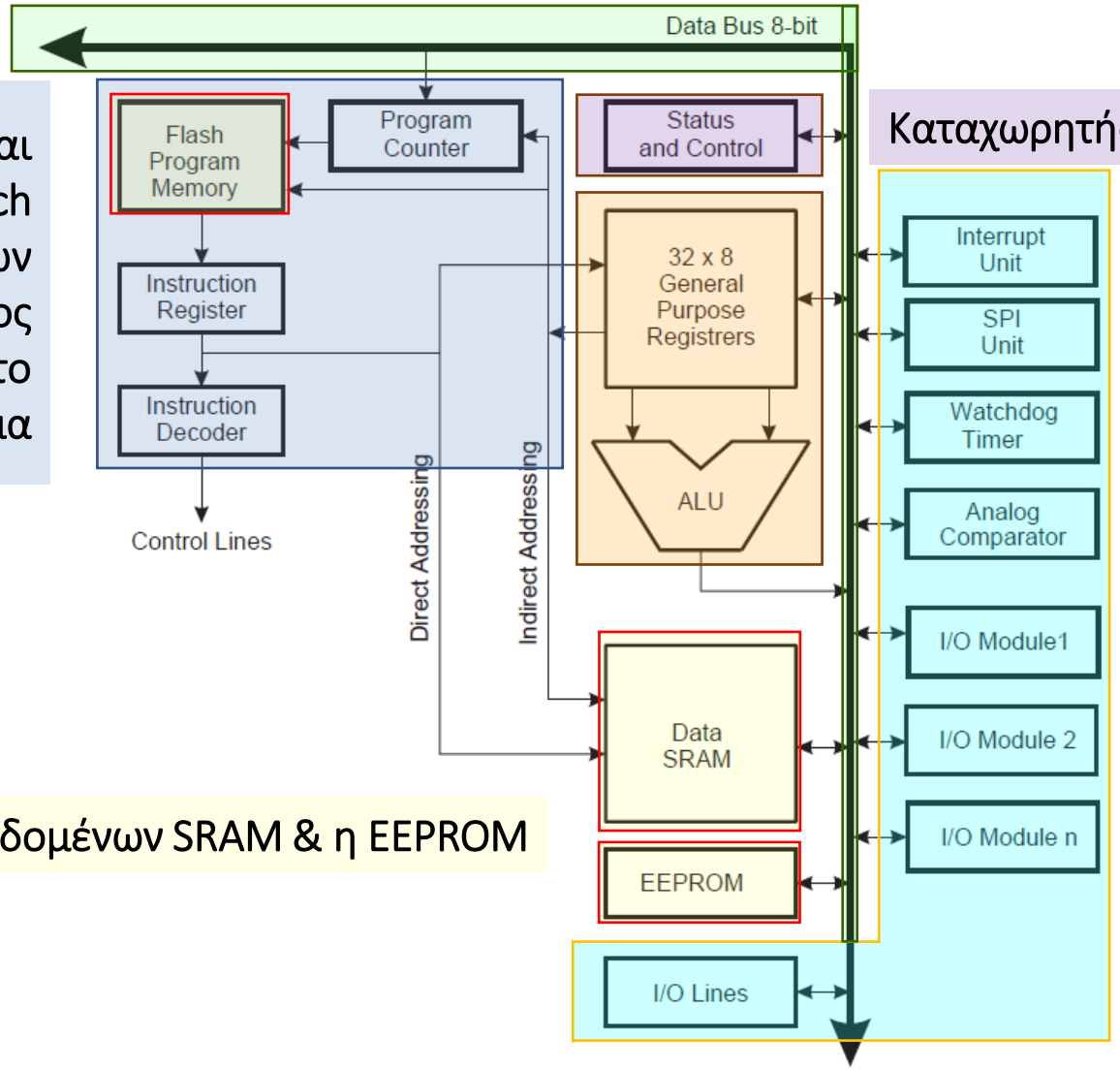




AVR CPU

Η μονάδα φόρτωσης και αποκωδικοποίησης (Fetch & Decode Unit) των εντολών προγράμματος και η Flash που έχει το πρόγραμμα

Διάδρομός δεδομένων – Data bus 8-bit



Καταχωρητής Κατάστασης και Μονάδα Ελέγχου

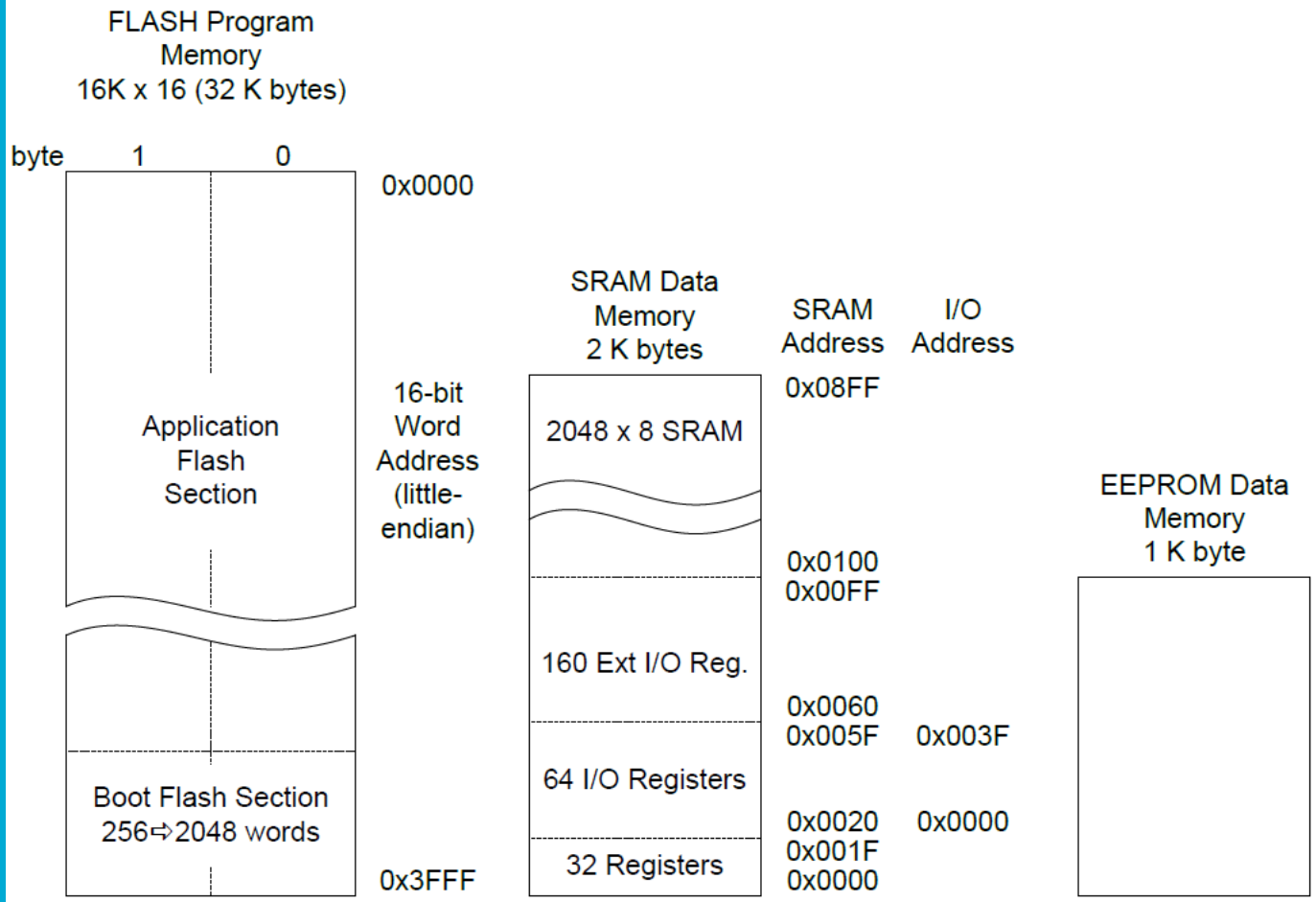
Η εκτελεστική μονάδα (Execution Unit) με τους Γενικού Σκοπού Καταχωρητές και την ALU.

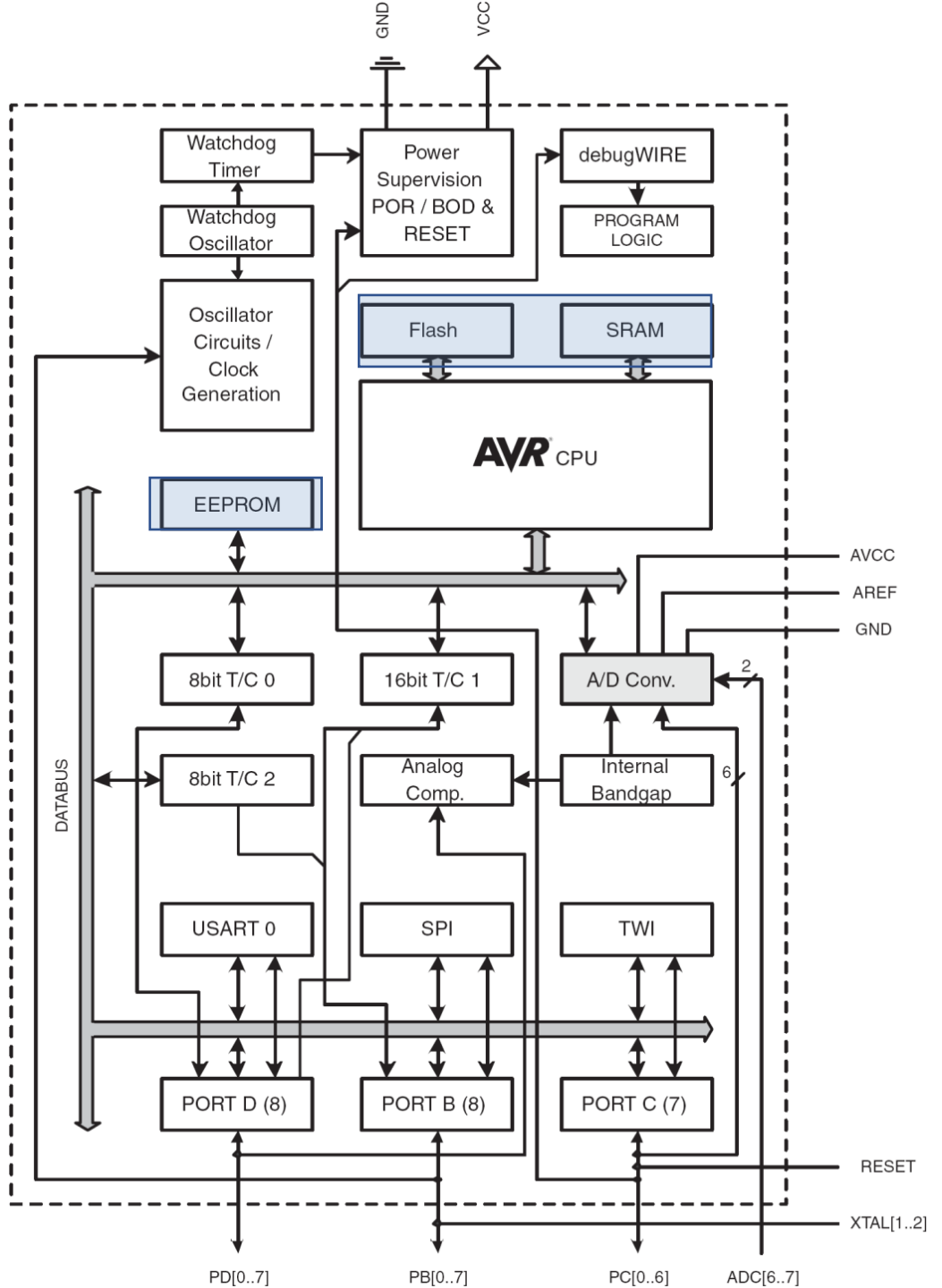
Η μνήμη δεδομένων SRAM & η EEPROM

Μονάδες I/O και ειδικές λειτουργίες

ATmega328P - Αρχιτεκτονική

Το σύστημα μνήμης και οι Χάρτες Μνήμης (Address Spaces)



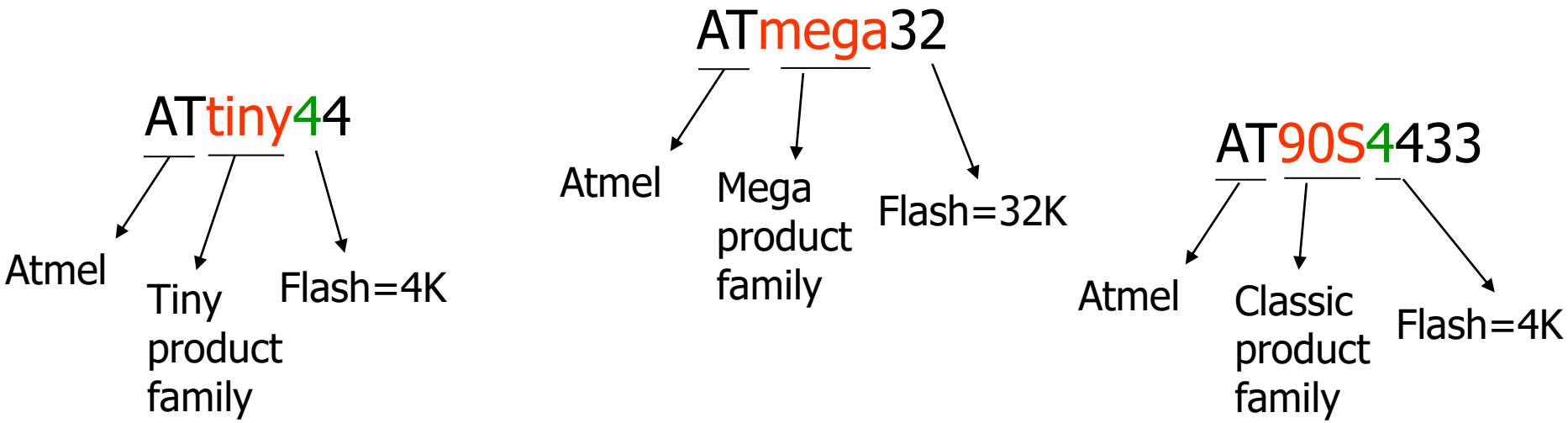


Η Μνήμη του ATmega328P (Arduino UNO)

Ο μικροεπεξεργαστής ATmega328 έχει τρεις (3) ομάδες μνήμης. Διαθέτει **Flash Memory**, στην οποία αποθηκεύεται το πρόγραμμα (τα Arduino sketch), την **SRAM (Static Random Access Memory)**, στην οποία τρέχει το πρόγραμμα και χρησιμοποιεί τις μεταβλητές, και **EEPROM**, η οποία χρησιμοποιείται από τους προγραμματιστές για την αποθήκευση μακροχρόνιων πληροφοριών.

- **2KB μνήμης SRAM:** Η ωφέλιμη μνήμη που μπορούν να χρησιμοποιήσουν τα προγράμματα για να αποθηκεύουν μεταβλητές, πίνακες κλπ. Η μνήμη χάνει τα δεδομένα της όταν η παροχή ρεύματος στο Arduino σταματήσει ή πατηθεί το κουμπί επανεκκίνησης.
- **1KB μνήμης EEPROM:** Μπορεί να χρησιμοποιηθεί για εγγραφή ή ανάγνωση δεδομένων από τα προγράμματα. Σε αντίθεση με την SRAM, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.
- **32KB μνήμης Flash:** 2 KB χρησιμοποιούνται από το firmware του Arduino που έχει εγκαταστήσει ήδη ο κατασκευαστής του. Το firmware (bootloader) είναι αναγκαίο για την εγκατάσταση προγραμμάτων στο μικροελεγκτή μέσω της θύρας USB. Τα υπόλοιπα 30KB της μνήμης Flash χρησιμοποιούνται για την αποθήκευση αυτών ακριβώς των προγραμμάτων, αφού πρώτα μεταγλωττιστούν στον υπολογιστή. Η μνήμη Flash, δε χάνει τα περιεχόμενά της με απώλεια τροφοδοσίας ή επανεκκίνησης.

Η Μνήμη του ATmega328P (Arduino UNO)



Type	Flash		RAM		EEPROM	
	F_END	Size, kB	RAMEND	Size, kB	E_END	Size, kB
ATmega8	$0FFF	8	$045F	1	$1FF	0.5
ATmega32	**$3FFF**	**32**	**$08FF**	**2**	**$3FF**	**1**
ATmega64	$7FFF	64	$10FF	4	$7FF	2
ATmega128	$FFFF	128	$10FF	4	$FFF	4

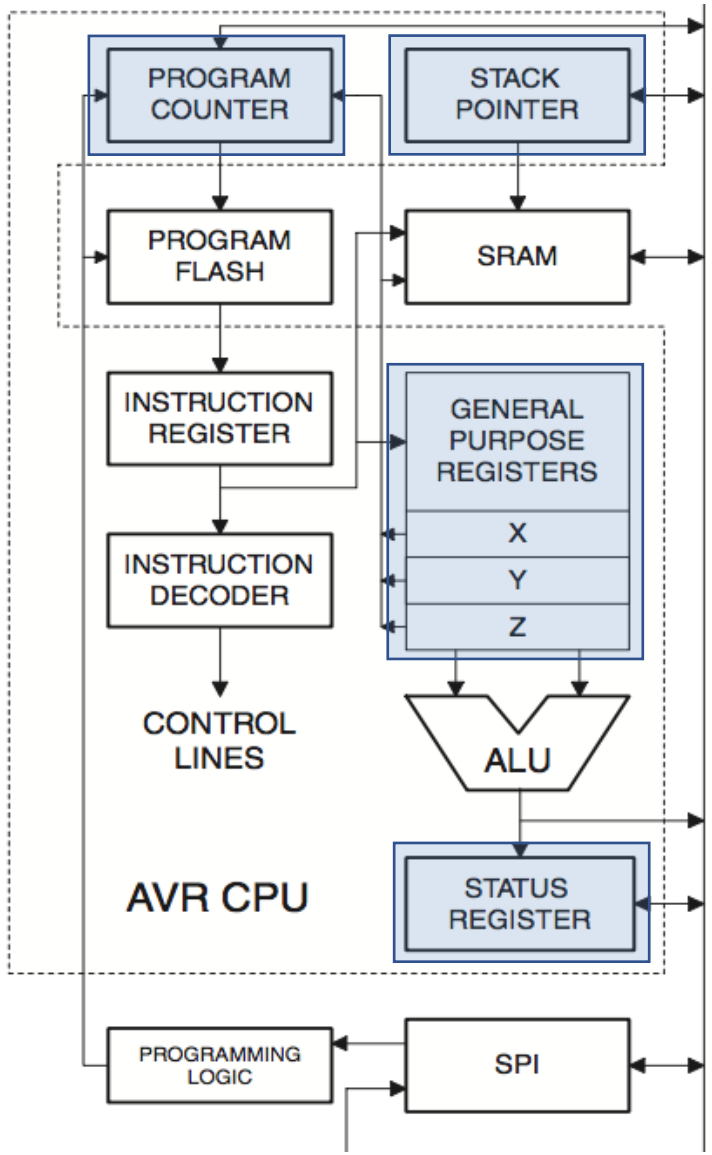
AVR ISA

Η **Αρχιτεκτονική του Συνόλου Εντολών** ενός μικροεπεξεργαστή (Instruction Set Architecture - ISA) εμπεριέχει όλους τους καταχωρητές που είναι προσβάσιμοι από τον προγραμματιστή.

Με άλλα λόγια, τους καταχωρητές που μπορούμε να μεταβάλουμε το περιεχόμενο τους με χρήση των διαθέσιμων εντολών της αρχιτεκτονικής του επεξεργαστή μας.

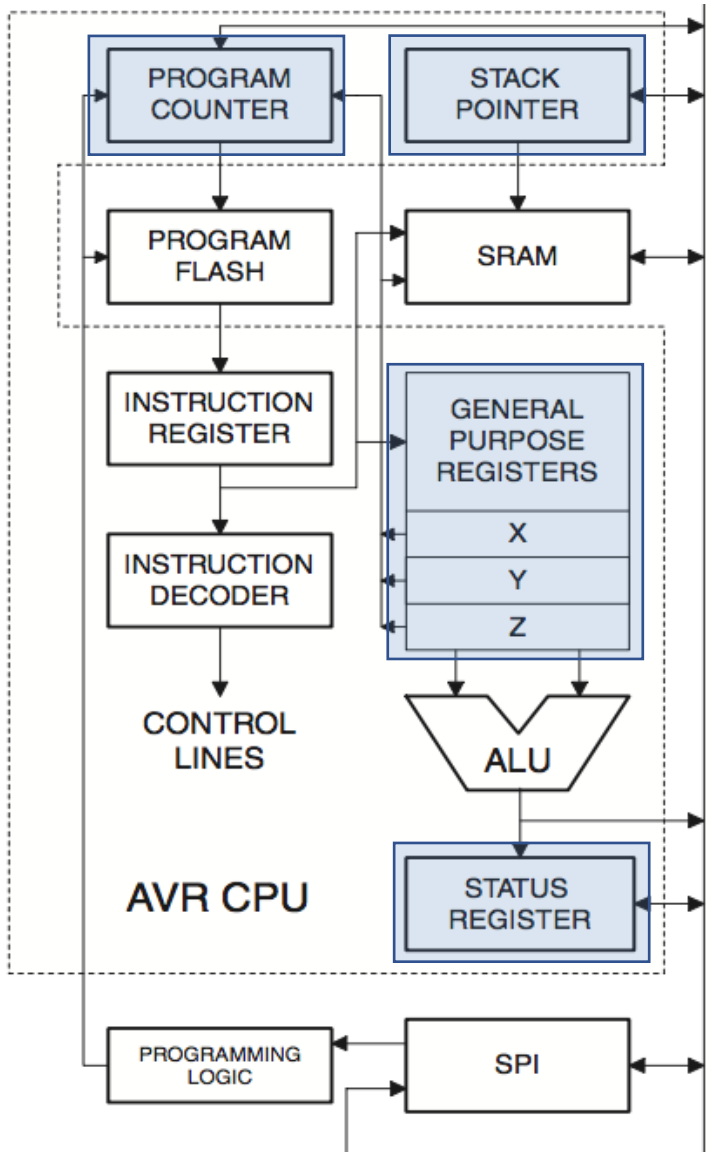
Σύμφωνα με το διάγραμμα του εσωτερικού του επεξεργαστή AVR αυτός διαθέτει τους παρακάτω **ISA καταχωρητές**:

- 32 x 8-bit Καταχωρητές Γενικού Σκοπού (General Purpose Registers)
- τον Καταχωρητή Κατάστασης (Status Register - SREG),
- τον Καταχωρητή Στοίβας (Stack Pointer - SP), και
- τον Μετρητή Προγράμματος (Program Counter - PC).



AVR ISA

Όνομα	Ετικέτα	Πλήθος	Μέγεθος	Λειτουργία
General Purpose Registers	R0-R31	32	8 bit	Αποθήκευση δεδομένων
Address Pointer	X,Y,Z	3	16 bit	Αποθήκευση Διευθύνσεων (Δείκτες)
Stack Pointer	SP	1	16 bit	Δείκτης Στοίβας
Program Counter	PC	1	14 bit	Μετρητής προγράμματος
Status Register	SREG	1	8 bit	Πληροφορίες για την κατάσταση του προγράμματος/CPU



Καταχωρητές Γενικού Σκοπού και Χάρτης Μνήμης Δεδομένων

Οι 32 καταχωρητές Γενικού Σκοπού (GP) (εικόνα 7-2) βρίσκονται στους χώρους μνήμης από **0x00-0x1F**.

Ο πίνακας με όλους τους καταχωρητές I/O ξεκινά από τη διεύθυνση **0x20 μέχρι και την 0xFF**.

Οι πρώτες 64 θέσεις στον πίνακα είναι επίσης διαθέσιμες για I/O διευθύνσεις **0x20-0x5F**.

Στην εικόνα 8-3 από τον οδηγό παρουσιάζεται ο χάρτης μνήμης για τα δεδομένα με τους:

- **32 GP καταχωρητές,**
- **64 I/O καταχωρητές,**
- **160 Extended I/O καταχωρητές και**
- **την on chip SRAM** (ανάλογα με το μέγεθος που διαθέτει).

Figure 7-2. **AVR** CPU General Purpose Working Registers

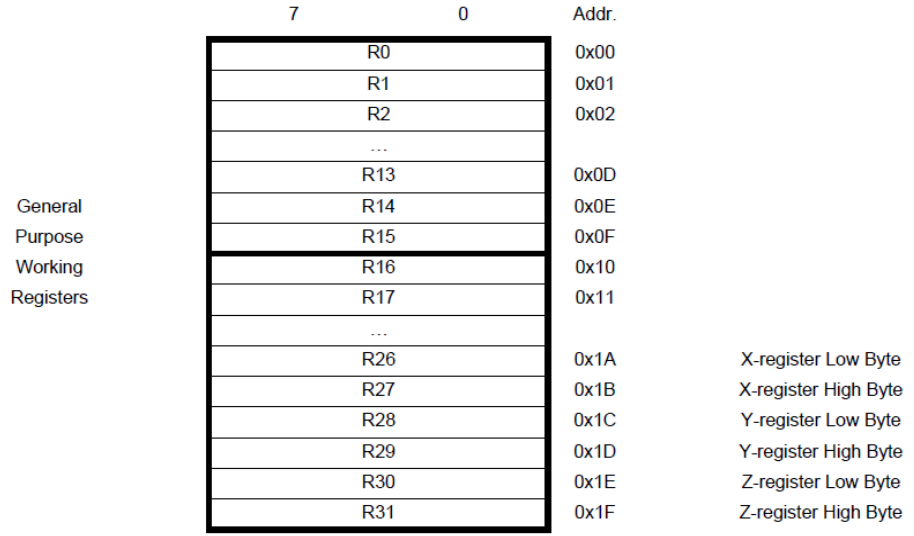
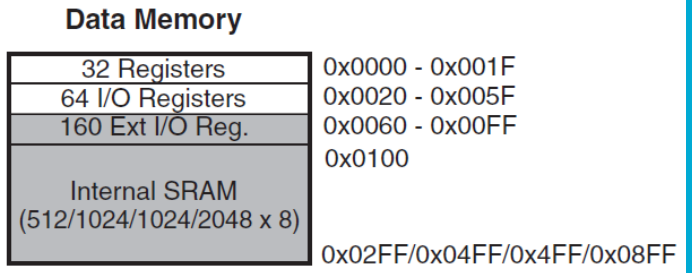


Figure 8-3. Data Memory Map



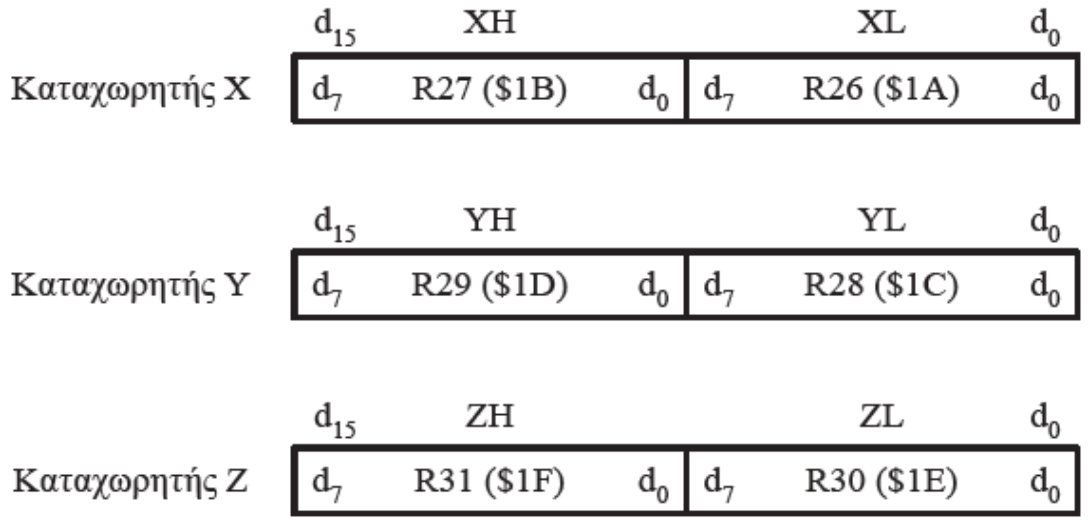
Καταχωρητές X, Y και Z

Το αρχείο καταχωρητών ταχείας πρόσβασης περιέχει 32 καταχωρητές 8-bit γενικού σκοπού, με δυνατότητα πρόσβασης σε όλους τους καταχωρητές μέσα σε έναν μόνο κύκλο ρολογιού. Αυτό σημαίνει ότι μια λειτουργία της αριθμητικής και λογικής μονάδας εκτελείται σε έναν κύκλο ρολογιού. Οι δύο τελεστέοι (operands) βγαίνουν από το αρχείο καταχωρητών, η πράξη εκτελείται και το αποτέλεσμα αποθηκεύεται πίσω στο αρχείο καταχωρητών σε έναν μόνο κύκλο ρολογιού.

Οι καταχωρητές R26 έως R31, χρησιμοποιούνται ανά ζεύγη και ως **καταχωρητές δείκτες 16 ψηφίων** για την έμμεση διευθυνσιοδότηση του χώρου μνήμης και ορίζονται με τον τρόπο που φαίνεται στο σχήμα.

Ο ένας από τους τρεις δείκτες διευθύνσεων μπορεί επίσης να χρησιμοποιείται και ως δείκτης στην αναζήτηση πινάκων που βρίσκονται στην ταχεία μνήμη δεδομένων (flash Data memory). **Αυτοί οι επιπρόσθετοι καταχωρητές είναι οι καταχωρητές 16 ψηφίων X, Y και Z.**

Στις διάφορες μεθόδους διευθυνσιοδότησης αυτοί οι καταχωρητές διεύθυνσης έχουν λειτουργίες όπως σταθερή μετατόπιση, αυτόματη αύξηση και αυτόματη μείωση.



Καταχωρητές Γενικού Σκοπού και Χάρτης Μνήμης Δεδομένων

Οι καταχωρητές GP, I/O, και η μνήμη SRAM αν και είναι στον ίδιο χώρο μνήμης είναι εντελώς διαφορετικοί στην λειτουργία. Μπορούν να προσπελαστούν με εντολές προσπέλασης μνήμης SRAM αλλά έχουν διαφορετικές λειτουργίες. **Οι καταχωρητές GP** μοιάζουν με τη μνήμη SRAM ως προς την ιδιότητα της αποθήκευσης δεδομένων αλλά δεν πρέπει να ξεχνάμε ότι βρίσκονται μέσα στον πυρήνα της CPU. Ο πυρήνας έχει άμεση προσπέλαση σε αυτούς με μεγάλη ταχύτητα και κάποιες πράξεις (όπως αριθμητικές, σύγκρισης, κλπ.) μπορούν μόνο να εκτελεστούν με χρήση των καταχωρητών. Άλλες εντολές χρησιμοποιούνται για τη μετακίνηση δεδομένων μεταξύ των GP και της SRAM.

Οι I\O καταχωρητές (με κάποιες εξαιρέσεις) είναι για τον χειρισμό των περιφερειακών (timers, USARTs, TWI, SPI, κλπ.).

Τέλος, **οι διευθύνσεις I/O με την σήμανση "reserved"** προειδοποιούν ότι εκεί δεν υπάρχει κάποια έγκυρη πληροφορία. Η ανάγνωση από μια δεσμευμένη διεύθυνση θα επιστρέψει μη καθορισμένα δεδομένα και η εγγραφή σε μια δεσμευμένη διεύθυνση δε θα έχει αποτέλεσμα.

Η Atmel το αναφέρει ως «δεσμευμένο» κυρίως για να υπογραμμίσει αυτό το γεγονός και ως προειδοποίηση για την αποφυγή τέτοιων ενεργειών. Η προειδοποίηση είναι κατάλληλη επειδή κάποια μελλοντική αναθεώρηση της συσκευής ενδέχεται να προσθέσει λειτουργικότητα και να τοποθετήσει έναν νέο καταχωρητή εισόδου / εξόδου σε ό, τι αφορά παλαιότερες συσκευές μια «δεσμευμένη» διεύθυνση.

Ο ATMega328P για τα δεδομένα χρησιμοποιεί **little-endian**, επομένως το λιγότερο σημαντικό byte (LSB) αποθηκεύεται στη χαμηλή διεύθυνση μνήμης ενώ το περισσότερο σημαντικό byte (MSB) αποθηκεύεται στην υψηλότερη θέση μνήμης.

Μέθοδοι Διευθυνσιοδότησης – AVR Addressing Modes

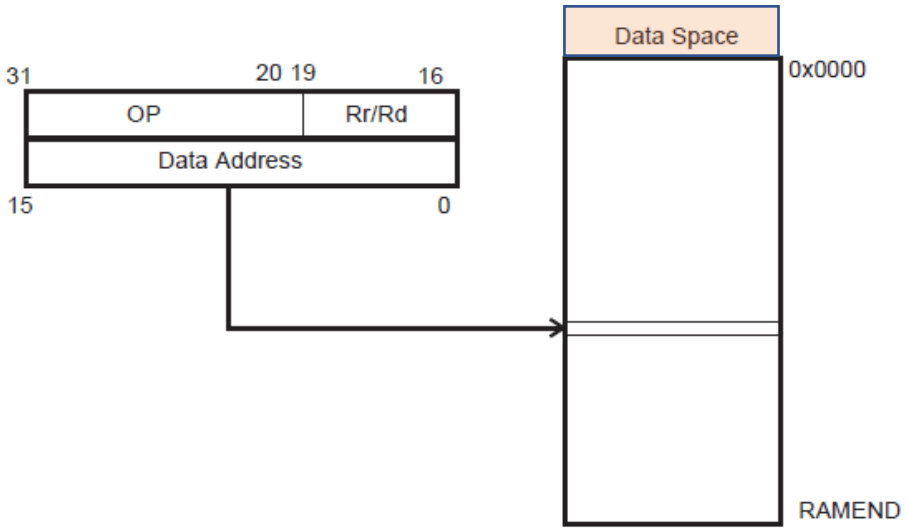
Ο AVR μπορεί να προσπελάσει τα δεδομένα με διάφορες μεθόδους. Τα δεδομένα μπορεί να βρίσκονται σε Καταχωρητές, ή τη Μνήμη ή να παρέχονται ως άμεσα δεδομένα. Αυτές οι διαφορετικές μέθοδοι προσπέλασης ονομάζονται Μέθοδοι Διευθυνσιοδότησης. Οι μέθοδοι διευθυνσιοδότησης είναι μέρος της Αρχιτεκτονικής του επεξεργαστή ή του Μοντέλου Προγραμματισμού του επεξεργαστή, σχεδιάζονται κατά την υλοποίηση της Αρχιτεκτονικής, υλοποιούνται σε υλικό (hardware) και χρησιμοποιούνται από τον προγραμματιστή. **Υπάρχουν πέντε (5) βασικοί μέθοδοι διευθυνσιοδότησης για τον AVR:**

- Register Direct (Άμεση Καταχωρητή)
  - Single Register (Μονού Καταχωρητή)
  - Two Registers (Δύο Καταχωρητών)
- I/O Direct (Άμεση Εισόδου/Εξόδου)
- Immediate (Απευθείας δεδομένων)
- Data Direct (Άμεση Μνήμης Δεδομένων)
- Data Indirect (Έμμεση Μνήμης Δεδομένων)
  - Data Indirect with Post-Increment (Μεταυξητική)
  - Data Indirect with Pre-Decrement (Προμειωτική)
  - Data Indirect with Displacement (με μετατόπιση)

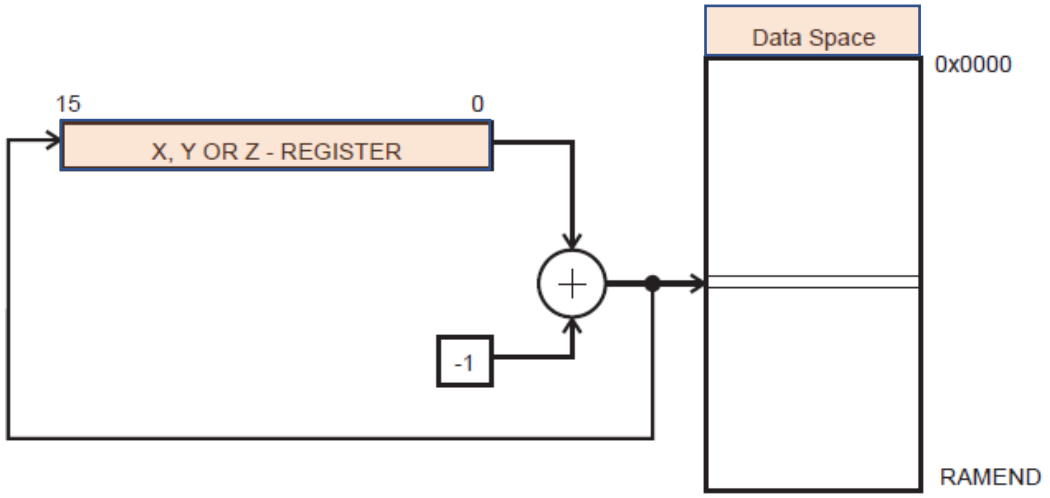
… και φυσικά υπάρχουν και μέθοδοι για την προσπέλαση της μνήμης προγράμματος.

AVR Addressing Modes - Παραδείγματα

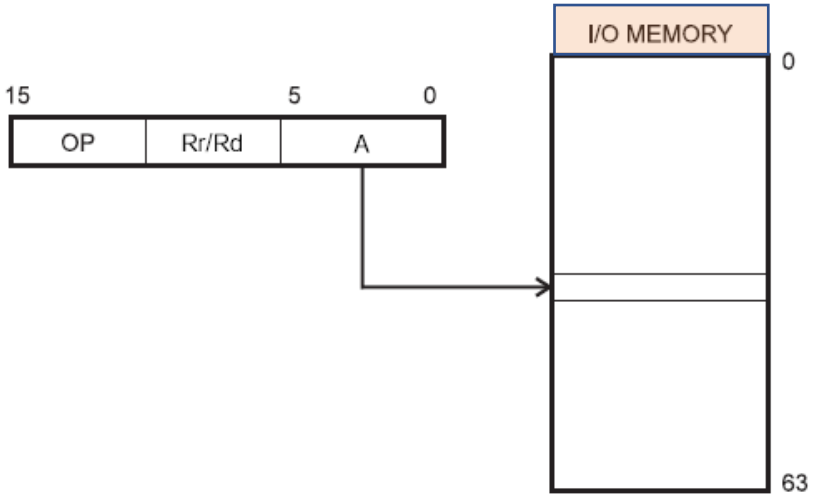
- Data Direct (Άμεση Μνήμης Δεδομένων)



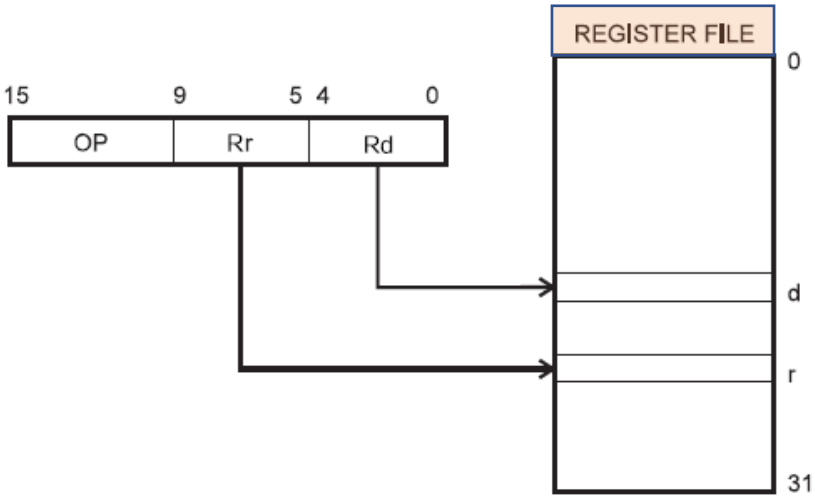
- Data Indirect with Pre-Decrement (Προμειωτική)



- I/O Direct (Άμεση Εισόδου/Εξόδου)



- Two Register Direct (Άμεση Δύο Καταχωρητών)



AVR Εντολές

Ο AVR του ATMega328P διαθέτει 131 εντολές με τις περισσότερες να είναι του ενός κύκλου εκτέλεσης.
Οι εντολές μπορούν σύμφωνα με τη λειτουργία τους να κατηγοριοποιηθούν στις παρακάτω κατηγορίες:

1. Μεταφοράς Δεδομένων (Data Transfer).
2. Αριθμητικές και Λογικές (Arithmetic and Logical).
3. Λογικού ψηφίου Bit και Bit – Test.
4. Ελέγχου – Διακλάδωσης (Control Transfer – Branch).
5. Διαχείρισης Μνήμης (MCU Control).

AVR Εντολές

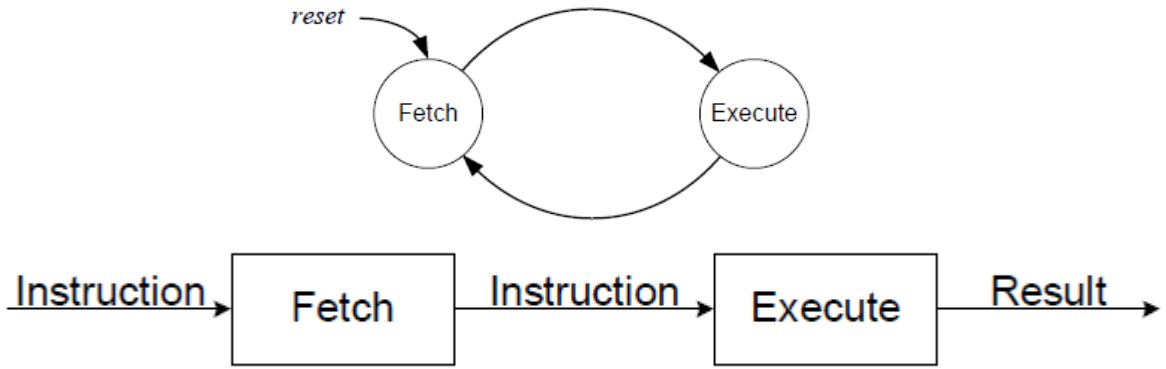
- **Οι εντολές Μεταφοράς Δεδομένων** χρησιμοποιούνται για φορτώσουμε (**Load**) και αποθηκεύσουμε (**Store**) δεδομένα στους **Καταχωρητές Γενικού Σκοπού (Αρχείο Καταχωρητών – Register File)**. Εξαίρεση είναι οι εντολές Push και Pop που αλλάζουν τον **Stack Pointer (SP)**. Από τον ορισμό τους οι εντολές αυτές δεν επηρεάζουν τον Καταχωρητή Κατάστασης (**SREG -Status Register**).
- **Οι αριθμητικές, λογικές και οι Bit / Bit-test εντολές** χρησιμοποιούν την **ALU** για να επεξεργαστούν τα δεδομένα που είναι αποθηκευμένα στους καταχωρητές.
- Τα bit κατάστασης (**Flags**) του **SREG** παρέχουν σημαντικές πληροφορίες σχετικές με το αποτέλεσμα των εντολών (από όποιες επηρεάζονται). Για παράδειγμα το overflow flag (V) bit του SREG θα δείξει αν μια πρόσθεση δύο προσημασμένων αριθμών έχει υπερχείλιση ή όχι.
- Ο επεξεργαστής AVR φορτώνει και εκτελεί τις εντολές προγράμματος ενώ αυτόματα αυξάνει τον Μετρητή Προγράμματος (**Program Counter**) να δείχνει στην επόμενη εντολή προς εκτέλεση. Οι εντολές Ελέγχου - διακλάδωσης επιτρέπουν στον προγραμματιστή να αλλάξει τη ροή του προγράμματος με ή χωρίς συνθήκη. Σε συνέχεια του προηγούμενου παραδείγματος, η ροή του προγράμματος μπορεί να αλλάξει αν έχουμε υπερχείλιση στην πρόσθεση (V=1) και πρέπει να χειριστεί το αποτέλεσμα μια ρουτίνας διόρθωσης.

Διοχέτευση 2-σταδίων (2-stage Pipeline)

Ο επεξεργαστής AVR RISC διαθέτει σύστημα διοχέτευσης εντολών **2-σταδίων (Two-stage Instruction Pipeline)** που αποτελείται από τη **Φόρτωση και την Εκτέλεση (fetch and execute)** με αποτέλεσμα οι περισσότερες από τις εντολές να εκτελούνται σε έναν κύκλο ρολογιού.

Συνεπώς, και η απόδοση του 20 MHz επεξεργαστή θα προσεγγίζει τα 20 MIPS (Millions of Instructions Per Second). Αν τον συγκρίνουμε με τον επεξεργαστή 8051 που είναι CISC και χρειάζεται το λιγότερο 12 clock cycles για να εκτελέσει μια εντολή (12 MHz clock = 1 MIPS).

Η **Διοχέτευση (Pipelining)** είναι μια τεχνική που σπάει τις διεργασίες, όπως είναι οι επεξεργασία των εντολών ή οι συναλλαγές των διαδρόμων, σε μικρότερα διακριτά στάδια ή διάρκειες έτσι ώστε μια διεργασία που ακολουθεί να μπορεί να ξεκινήσει πριν να έχει ολοκληρωθεί η προηγούμενη.



Διοχέτευση 2-σταδίων (2-stage Pipeline)

Για τις περισσότερες εντολές, ειδικά σε αυτές που βασίζονται σε τροποποιημένο μοντέλο μνήμης Harvard, δεν είναι δυνατή η πρόσβαση στη μνήμη του προγράμματος κατά τη διάρκεια του κύκλου εκτέλεσης. Αυτός ο χρόνος διακοπής μνήμης θα μπορούσε να χρησιμοποιηθεί για τη λήψη της επόμενης εντολής που θα εκτελεστεί, παράλληλα με τον κύκλο εκτέλεσης της τρέχουσας εντολής.

Εδώ λοιπόν είναι μια ευκαιρία για διοχέτευση (Pipelining)!

Η διοχέτευση έχει δύο ανεξάρτητα στάδια.

Το πρώτο στάδιο παίρνει μια εντολή και την τοποθετεί στο Instruction Register (IR), ενώ το δεύτερο στάδιο εκτελεί την εντολή. Η διοχέτευση δύο σταδίων ονομάζεται επίσης προ-φόρτωση εντολών (instruction pre-fetch) και συναντάται σε μερικούς από τους πρώτους μικροεπεξεργαστές, συμπεριλαμβανομένου του Intel 8086.

Ένα στάδιο διοχέτευσης ξεκινά και τελειώνει σε έναν καταχωρητή και συγχρονίζεται από ένα κοινό ρολόι. Μεταξύ των καταχωρητών υπάρχει συνδυαστική λογική. Παρόλο που είναι διαισθητικό, η μνήμη προγράμματος Flash μπορεί να θεωρηθεί ως συνδυαστική λογική με μια διεύθυνση που δημιουργεί μια λέξη δεδομένων.

Διοχέτευση 2-σταδίων (2-stage Pipeline)

Όσον αφορά την AVR αρχιτεκτονική οι δύο καταχωρητές ενδιαφέροντος είναι ο **Μετρητής Προγράμματος (PC) και ο Καταχωρητής Εντολής (IR)**.

Χωρίς διοχέτευση, αυτοί οι δύο καταχωρητές (PC, IR) θα απαιτούσαν δύο κύκλους ρολογιού για την ολοκλήρωση ενός βασικού κύκλου λειτουργίας του υπολογιστή.

Συγκεκριμένα, λαμβάνεται μια εντολή (1) και στη συνέχεια (2) εκτελείται.

Για την αρχιτεκτονική RISC, οι περισσότερες εντολές εκτελούνται σε έναν κύκλο (elemental instructions). Σε αυτόν τον τέλειο κόσμο όπου όλες οι εντολές χρειάζονται έναν κύκλο για ανάκτηση και έναν κύκλο για εκτέλεση, μετά από μια αρχική καθυστέρηση ενός κύκλου για να γεμίσει η διοχέτευση (latency), κάθε εντολή θα διαρκέσει μόνο έναν κύκλο για να ολοκληρωθεί.

	Time 1	2	3	4
Fetch	Instr. 1	Instr. 2	Instr. 3	Instr. 4
Execute		Instr. 1	Instr. 2	Instr. 3

Παράδειγμα Εντολής (LDI)

Status Register (SREG)

SREG:	Status Register
C:	Carry Flag
Z:	Zero Flag
N:	Negative Flag
V:	Two's complement overflow indicator
S:	N ⊕ V, For signed tests
H:	Half Carry Flag
T:	Transfer bit used by BLD and BST instructions
I:	Global Interrupt Enable/Disable Flag

Registers and Operands

Rd:	Destination (and source) register in the Register File
Rr:	Source register in the Register File
R:	Result after instruction is executed
K:	Constant data
k:	Constant address
b:	Bit in the Register File or I/O Register (3-bit)
s:	Bit in the Status Register (3-bit)
X,Y,Z:	Indirect Address Register (X=R27:R26, Y=R29:R28 and Z=R31:R30)
A:	I/O location address
q:	Displacement for direct addressing (6-bit)

LDI – Load Immediate

Description:

Loads an 8 bit constant directly to register 16 to 31.

Operation:

(i) Rd ← K

	Syntax:	Operands:
(i)	LDI Rd,K	16 ≤ d ≤ 31, 0 ≤ K ≤ 255

Program Counter:
PC ← PC + 1

16-bit Opcode:

1110	KKKK	dddd	KKKK

Status Register (SREG) and Boolean Formula:

I	T	H	S	V	N	Z	C
–	–	–	–	–	–	–	–

Example:

```
clr   r31       ; Clear Z high byte
ldi   r30,$F0   ; Set Z low byte to $F0
lpm             ; Load constant from Program
                ; memory pointed to by Z
```

Words: 1 (2 bytes)
Cycles: 1

AVR Instruction Set (οι πιο συχνές)…

AVR® Assembly Language Summary[1] (J. Vasconcelos, 2008)

Category	Instruction	Example	Meaning	Comments
Arithmetic	Add	**add** R5, R6	R5 = R5 + R6	
	Subtract	**sub** R5, R6	R5 = R5 – R6	
	Negate (2's complement)	**neg** R5	R5 = 0 – R5	R5 = 1 + (~R5)
	Clear register	**clr** R5	R5 = 0	
	Set register	**ser** R5	R5 = $FF	
	Increment register	**inc** R5	R5 = R5 + 1	
	Decrement register	**dec** R5	R5 = R5 – 1	
	Test zero or negative	**tst** R5	If Rd <= 0, SREG[N,Z] = 1	
	Multiply	**mul** R5, R6	R1,R0 = R5 x R6	Result is 16-bit long
Bitwise	Set individual bits	**sbr** R16, 65	R16[6] =1, R16[0] =1	R16= R16 OR $41
	Clear individual bits	**cbr** R16, 65	R16[6] =0, R16[0] =0	R16 = R16 AND ~R5
	Shift bits to the left	**lsl** R5	R5 = R5 x 2	R5[7]=R5[6] … R5[0]=0
	Shift bits to the right	**lsr** R5	R5 = R5 / 2	R5[7]=0 … R5[0]=R5[1]
Logical	And	**and** R5, R6	R5 = R5 AND R6	
	Or	**or** R5, R6	R5 = R5 OR R6	
	Xor (exclusive or)	**eor** R6, R5	R5 = R5 XOR R6	
	Flip bits (1's complement)	**com** R5	R5 = $FF – R5	R5 = ~R5
	Compare two registers	**cp** R5, R6	If R5=R6, SREG[Z] =1 else SREG[Z]=0	Performs (R5-R6) to update SREGS

AVR® Assembly Language Summary[1] (J. Vasconcelos, 2008)

Category	Instruction	Example	Meaning	Comments
Data transfer	Load immediate	**ldi** R16, 0xF0	R16 = $F0	
	Copy register	**mov** R19, R20	R19 = R20	
	Send data to I/O port	**out** PortB, R16	I/O(PortB) = R16	
	Get data from I/O port	**in** R17, PinB	R17 = I/O(PinB)	
	Load register from memory	**ld** R5, X	R5 = mem[X]	X is 16-bit register R5 is a 8-bit register
		ld R5, X+	R5 = mem[X], X= X+1	
	Store register in memory	**st** X, R5	mem[X] = R5	X is 16-bit register R5 is a 8-bit register
		st X+, R5	mem[X] = R5, X+1	
	Save register in stack	**push** R5	STACK[top] = R5	SP = SP –1
	Load register from stack	**pop** R5	R5 = STACK[top]	SP = SP +1
Conditional branch	Skip if bit set	**sbrs** R17, 6	Skip next instruction if R17[6] = 1	
	Skip if bit cleared	**sbrc** R17,6	Skip next instruction if R17[6] = 0	
	Skip if registers are equal	**cpse** R16,R17	Skip next instruction if R16=R17	
	Branch on equal	**breq** LABEL	if regA=RegB, goto LABEL	Registers have to be compared right before the branch: cp regA, regB
	Branch on not equal	**brne** LABEL	if regA<>RegB, goto LABEL	
	Branch on greater than **(signed)**	**brge** LABEL	if regA>RegB, goto LABEL	
	Branch on lesser than **(signed)**	**brlt** LABEL	if regA<RegB, goto LABEL	
Non-conditional jump	Jump	**rjmp** LABEL	Goto LABEL	Jump relative to PC
	Absolute jump	**jmp** LABEL	Goto LABEL	Jump to specific instruction location.
	Procedure call	**rcall** PROC	Goto PROC	Ret.Addr. saved in STACK
	Return from subroutine	**ret**	Goto return address	Ret.Addr.retrieved STACK
Misc.	Reset watchdog timer	**wdr**	Reset watchdog timer	
	No operation	**nop**		

- Instructions accepting immediate values (LDI, ADIW, SUBI, ORI, etc.) work only on the upper half of the register file (R16–R31).
- Skip instructions (SBRC, SBRS, SBIC, and SBIS) jump over the next instruction if any bit within the register is set (1) or clear (0).
- Any bit within a general-purpose register can be set or clear (CBR, CBI, SBR, SBI). SER and CLR wipe out the contents of the entire register at once.
- There are two conditional branch instructions for each flag in the status register.
- Every flag in its status register can be set (1) or cleared (0) with SEx/CLx instructions.

ATMega32 – Input / Output

Ένα από τα βασικά χαρακτηριστικά ενός υπολογιστικού συστήματος είναι η δυνατότητα του να ανταλλάσσει δεδομένα με **εξωτερικές συσκευές εισόδου/εξόδου (I/O devices)** και να επιτρέπει στον χρήστη μέσω του υπολογιστή να αλληλοεπιδρά με αυτά:

Συσκευές Εισόδου:
Διακόπτες, Πληκτρολόγια, ποντίκια , σαρωτές, κάμερες, κλπ.

Συσκευές Εξόδου:
LED/LCD οθόνες, ηχεία, εκτυπωτές, κλπ.

Συνηθισμένες διαπαφές I/O (I/O interfaces):
Βασικές I/O: Απλές δύο καταστάσεων συσκευές όπως LED και διακόπτες.
Παράλληλο I/O (Parallel I/O): Ανταλλαγή δεδομένων ενός byte κάθε χρονική στιγμή.
Σειριακό I/O (Serial I/O): Ανταλλαγή δεδομένων ενός bit κάθε χρονική στιγμή.

Microprocessor I/O Interfacing

Οι περισσότερες κλήσεις I/O πραγματοποιούνται από εφαρμογές ή το λειτουργικό σύστημα, και αφορούν μετακίνηση δεδομένων μεταξύ περιφερειακών συσκευών και κύριας μνήμης.

Στους υπολογιστές υπάρχουν τρεις (3) τρόποι επικοινωνίας με τις συσκευές:

- Με διευθυνσιοδότηση συσκευών Εισόδου / Εξόδου (I/O Addressing).
- Με Απευθείας Πρόσβαση στη Μνήμη (Direct Access Memory).
- Με Διακοπές (Interrupts).

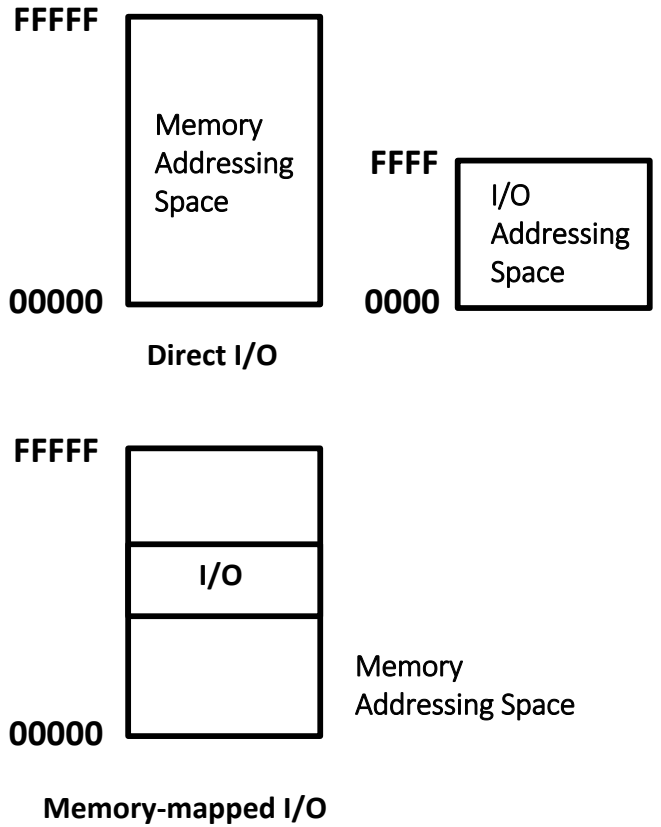
Microprocessor I/O Interfacing

Βασισμένο στις θύρες (Port-Based: Standard I/O-Direct I/O)

- Οι διευθύνσεις I/O είναι ξεχωριστές από τις διευθύνσεις μνήμης.
- Το πρόγραμμα διαβάζει και γράφει τη θύρα (port) όπως σε έναν καταχωρητή.
- π.χ., PA0 = 0xFF
  - ***Πλεονεκτήματα***: Δε χρησιμοποιεί χώρο διευθύνσεων από τη μνήμη.
  - ***Μειονεκτήματα***: Χρησιμοποιεί μόνο IN ή OUT εντολές για τη μεταφορά δεδομένων.

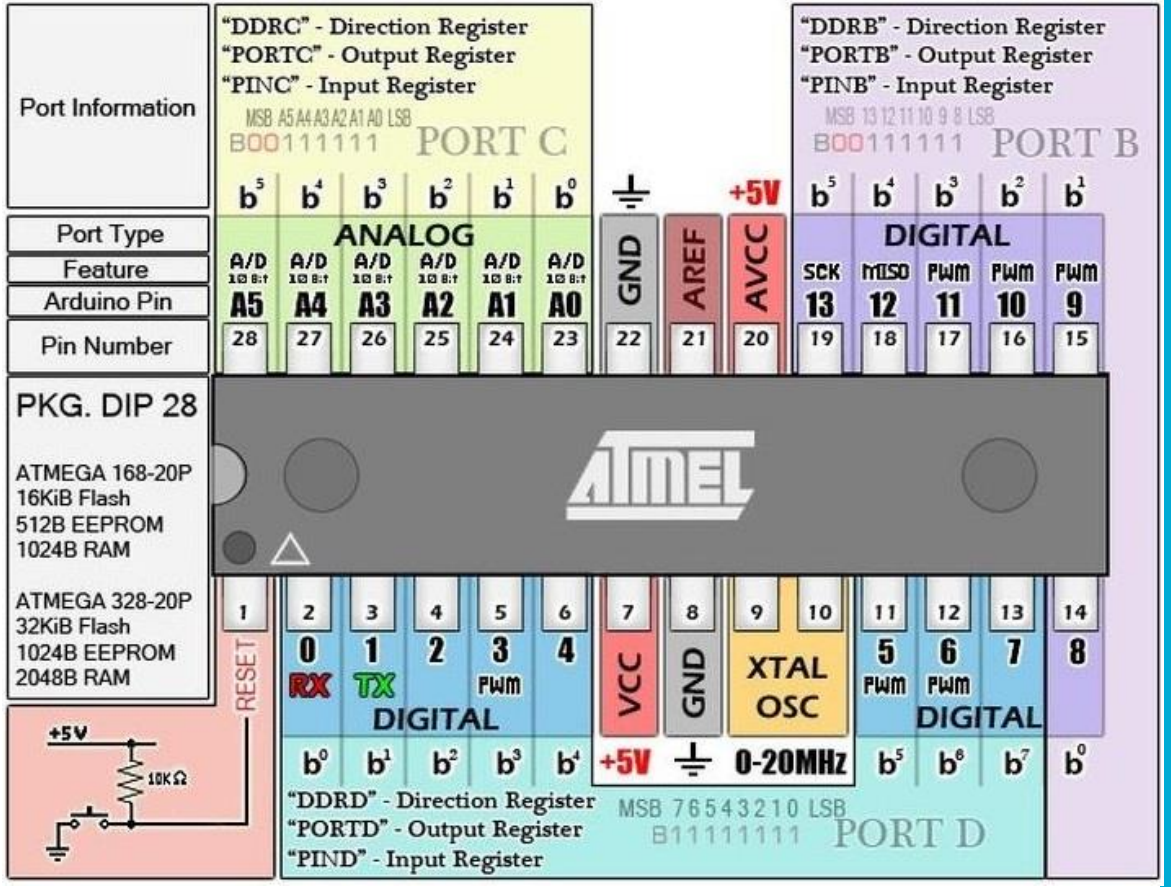
Με Αντιστοίχιση Μνήμης (Memory-Mapped I/O)

- Οι θύρες I/O χειρίζονται ως θέσεις μνήμης.
  - ***Πλεονεκτήματα***: Πρόσβαση στις θύρες I/O όπως σε θέσεις μνήμης.
  - ***Μειονεκτήματα***: Χρησιμοποιεί χώρο της μνήμης.



ATMega32 I/O Port Registers

- Ο ATMega328 διαθέτει τρεις φυσικές θύρες I/O (Ports), τις **B, C και D**.
  - **Port B** είναι μια 8-bit αμφίδρομη (bi-directional) I/O θύρα.
  - **Port D** είναι μια 8-bit αμφίδρομη (bi-directional) I/O θύρα.
  - **Port C** είναι μια 7-bit αμφίδρομη (bi-directional) I/O θύρα.
- Σε κάθε ακροδέκτη υπάρχουν “Pull-up” resistors που μπορούν να ενεργοποιηθούν/ απενεργοποιηθούν προγραμματιστικά.



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	93
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	93
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	93
0x08 (0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	92
0x07 (0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	92
0x06 (0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	92
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	92
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	92
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	92

ATMega32 I/O Port Registers

Στον AVR υπάρχουν τρεις (3) καταχωρητές που είναι σχετικοί με τις θύρες I/O:

- **PORTx:** PORTx data register (read/write).
- **DDRx:** Data Direction bit (input/output) in DDRx register (read/write).
- **PINx:** PINx register (read only).

PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

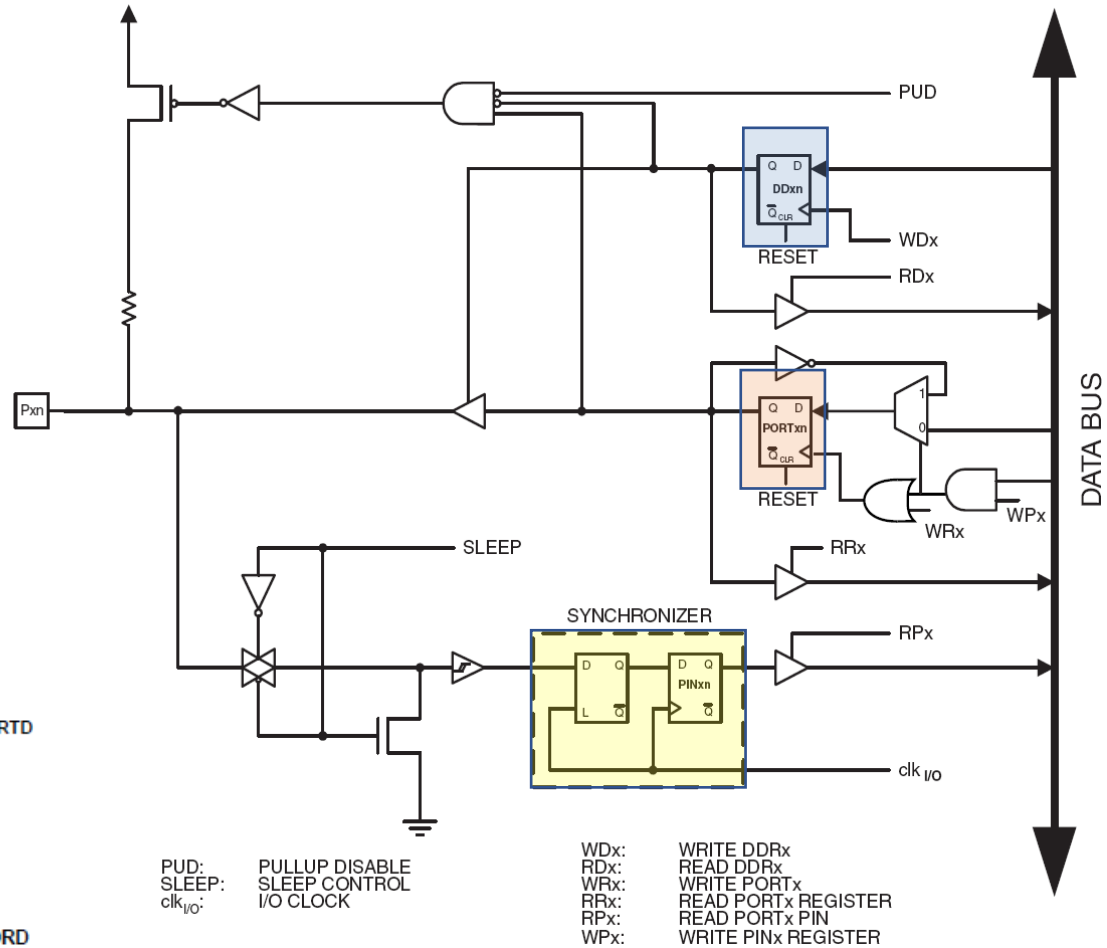
DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Figure 13-2. General Digital I/O[1]



DDRx Data Direction Register (read/write)

Ο **DDRx** χειρίζεται την κατεύθυνση των δεδομένων για κάθε ακίδα (pin) μιας θύρας **PORTx**.
Για παράδειγμά ο **DDRD** καταχωρητής χειρίζεται και τα 8 pin (0-7) της θύρας **PORTD**.

- *Αν ορίσουμε το Pin 4 σε 0 δηλ. DDRD4=0 τότε ορίζεται ως ακίδα Εισόδου (clear bit in DDRD4)*
- *Αν ορίσουμε το Pin 0 σε 1 δηλ. DDRD0=1 τότε ορίζεται ως ακίδα Εξόδου (set bit in DDRD0)*

Αν θέλουμε να αλλάξουμε τη φορά των δεδομένων σε μια ομάδα ακίδων μια θύρας **PORTx**:

1. Καθορίζουμε ποια bits πρέπει να γίνουν 0 και ποια 1 στον DDRx.
2. Αποθηκεύουμε τον δυαδικό ή δεκαεξαδικό αριθμό στον DDRx.

DDRD – The Port D Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x0A (0x2A)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	DDRD
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	

PORTx Data Register (read/write) / Pull-up Resistors

Στους μικροελεγκτές AVR συνήθως οι ακίδες εισόδου είναι σε **κατάσταση υψηλής εμπέδησης (Hi-Z)**. Αυτό τις κάνει ευαίσθητες στον θόρυβο και τη σύλληψη λανθασμένων τιμών σήματος. Γι'αυτό το ολοκληρωμένο μας διαθέτει αντιστάσεις αναρρίχησης (**Pull-up resistors**) για τη μείωση του θορύβου.

Με τον καταχωρητή **PORTx** ενεργοποιούμε τις Pull-up resistors όταν η ακίδα μας είναι **είσοδος** ως εξής:

- PORTxn= 1, Pull-up resistors ενεργοποιούνται (για το n-οστό pin).
- PORTxn= 0, Pull-up resistors απενεργοποιούνται (για το n-οστό pin).

Με τον καταχωρητή **PORTx** ενεργοποιούμε τις Pull-up resistors όταν η ακίδα μας είναι **έξοδος** ως εξής:

- PORTxn= 1, portx n-οστο pin οδηγείται σε υψηλό 1.
- PORTxn= 0, portx n-οστο pin οδηγείται σε χαμηλό 0.

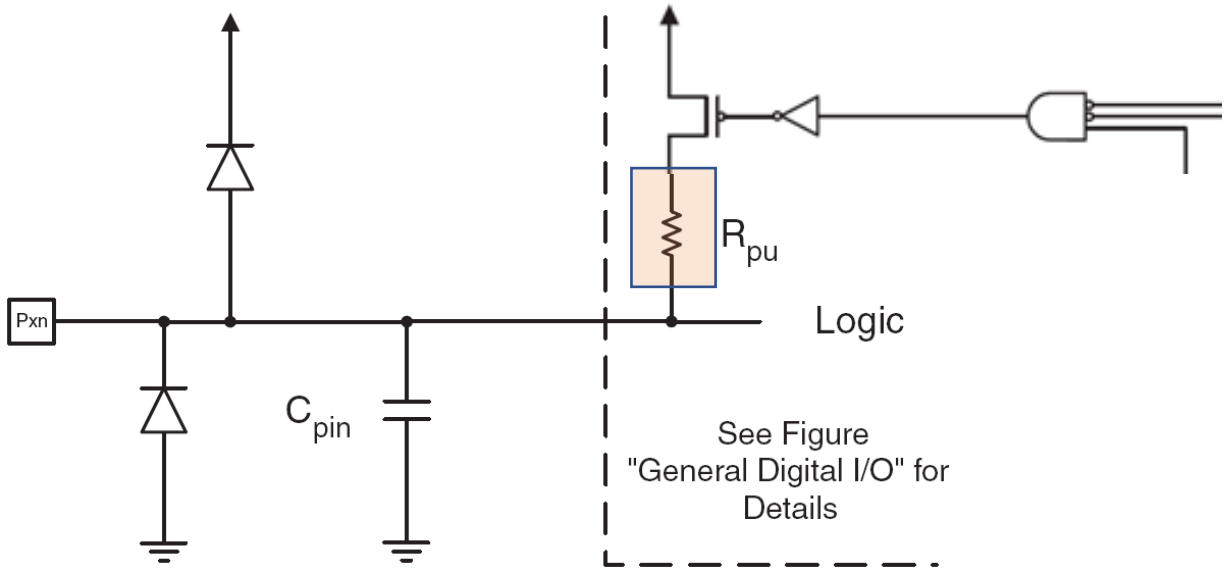
PORTD – The Port D Data Register

Bit	7	6	5	4	3	2	1	0	
0x0B (0x2B)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	PORTD
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	

Pull-up Resistors

Symbol	Parameter	Condition	Min.	Typ.	Max.	Units
R_{PU}	I/O Pin Pull-up Resistor		20		50	kΩ

Οι αντιστάσεις Pull-up χρησιμοποιούνται για να τραβήξουν τα λογικά σήματα προς τα πάνω (στη λογική 1). Έτσι, όταν κάποιο σήμα εισόδου πρέπει να ρυθμιστεί στη λογική 1, αλλά μπορεί να χρειαστεί να αλλάξει για κάποιον λόγο στο 0 σε κάποια άλλη στιγμή, μια αντίσταση Pull-up μπορεί να κρατήσει το σήμα στη λογική 1 έως ότου το σήμα οδηγηθεί σε κατάσταση 0.



Τυπική εφαρμογή για αντιστάσεις αναρρίχησης είναι η σύνδεση της τάσης λειτουργίας κυκλώματος (+5V συνήθως) με τον ακροδέκτη εισόδου 4,7 kΩ (ή κάποια άλλη κατάλληλη τιμή αντίστασης συνήθως μεταξύ 1 kΩ και 10 kΩ, στον ATmega328P έχει τιμή από 20 έως 50kΩ). Αυτή η αντίσταση διατηρεί το σήμα στη λογική 1. Όταν το σήμα πρέπει να γίνει 0, οδηγείται προς τα κάτω συνδέοντας τον ακροδέκτη εισόδου στη γείωση (συνήθως μέσω ενός κουμπιού, του διακόπτη DIP ή ενός τρανζίστορ).

Οι Pull-down αντιστάσεις λειτουργούν με τον αντίθετο τρόπο. Διατηρούν σήμα λογική 0 έως ότου κάτι συνδεθεί με την είσοδο σήματος στα +5V (στην περίπτωση που είναι η τάση λειτουργίας του λογικού κυκλώματος).

PINx Port Input Pin Register (read only)

Ο **PINx** καταχωρητής περιέχει την κατάσταση όλων των ακίδων μιας θύρας.
π.χ. ο **PINB** έχει όλα τα bit για τις ακίδες τις θύρας B.

Αν το pin είναι είσοδος, τότε η ακίδα θα «μιμείται» το λογικό επίπεδό που ορίστηκε να έχει.

Αν το pin είναι έξοδος, το bit θα περιέχει το δεδομένο που ήταν η προηγούμενη έξοδος αυτής της ακίδας (η τιμή μια ακίδας εξόδου αποθηκεύεται (latched) στο PINxn bit).

PIND – The Port D Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x09 (0x29)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	PIND
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

AVR Assembly Language

- **Από τη πλευρά του Υλικού (Hardware)**
  - Η Assembly διδάσκει πως ο υπολογιστής δουλεύει σε επίπεδο μηχανής (π.χ. καταχωρητές).
- **Από τη πλευρά του Λογισμικού (Software)**
  - Πολλές από τις δομές προγραμματισμού υψηλού επιπέδου βασίζονται στη γλώσσα Assembly και την αρχιτεκτονική υπολογιστών (Data types, addressing modes, stack, input/output).
- Η Assembly δε χρησιμοποιείται μόνο για να παρουσιάσουμε αλγορίθμους, αλλά για να παρουσιάσουμε τι πραγματικά συμβαίνει μέσα στην Κεντρική Μονάδα Επεξεργασίας.

AVR Assembler – Σύνταξη δηλώσεων

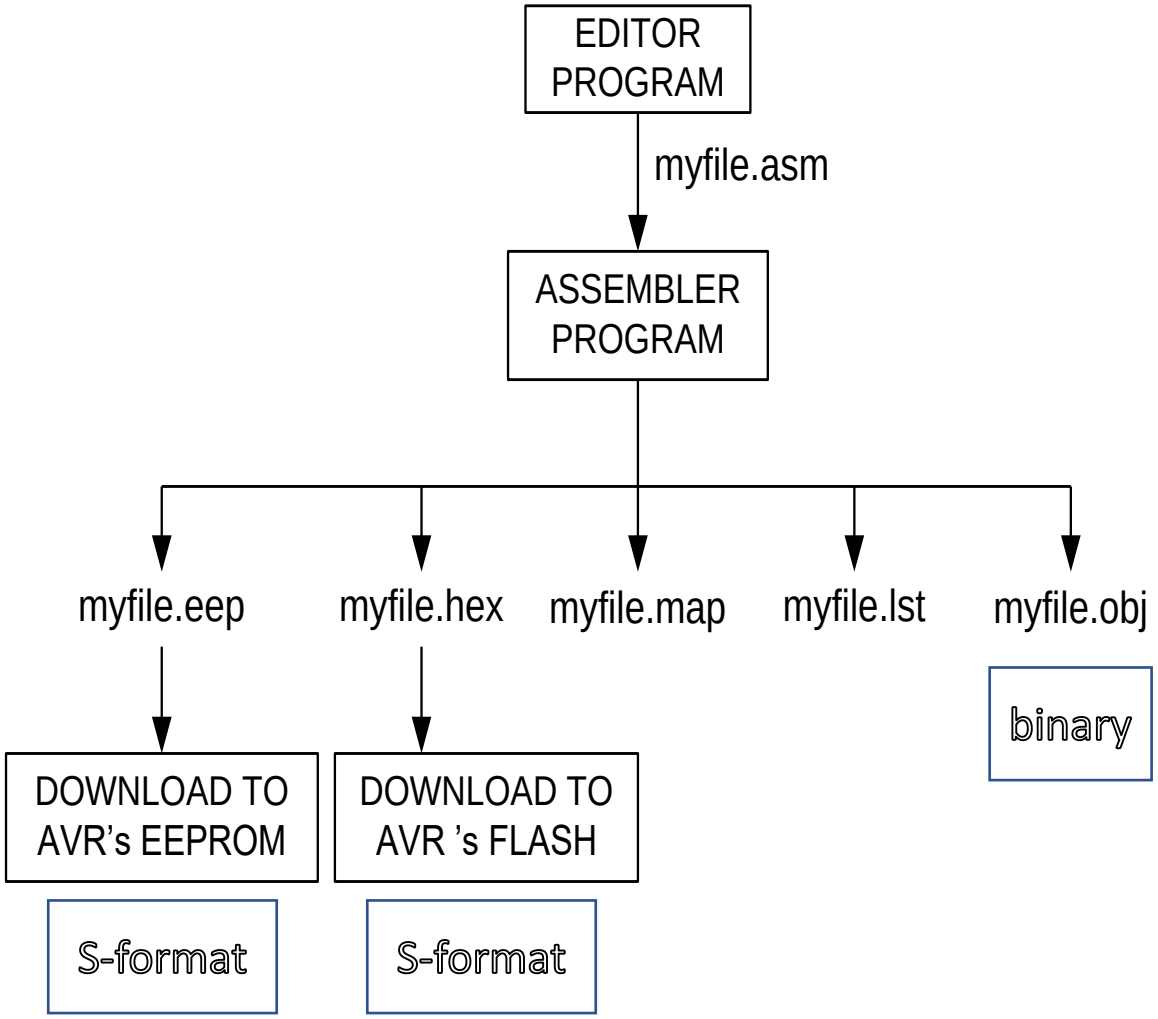
- Ο **Συμβολομεταφραστής (Assembler)** επεξεργάζεται πηγαία αρχεία που περιέχουν δηλώσεις **μνημονικών εντολών (mnemonics), ετικετών (labels) και ψευδοεντολών (directives)**.
- Οι δηλώσεις των εντολών και των ψευδοεντολών συνήθως έχουν και τελεστές.
- Κάθε γραμμή κώδικα περιορίζεται μέχρι τους 120 χαρακτήρες (characters).
- Κάθε γραμμή κώδικα μπορεί να ξεκινά με μια ετικέτα (**label**), η οποία είναι ένα αλφαριθμητικό κείμενο που τερματίζει με : (colon). Οι ετικέτες μπορούν να χρησιμοποιηθούν ως στόχοι για τις εντολές jump/branch και ως μεταβλητές στη μνήμη προγράμματος και τη RAM.

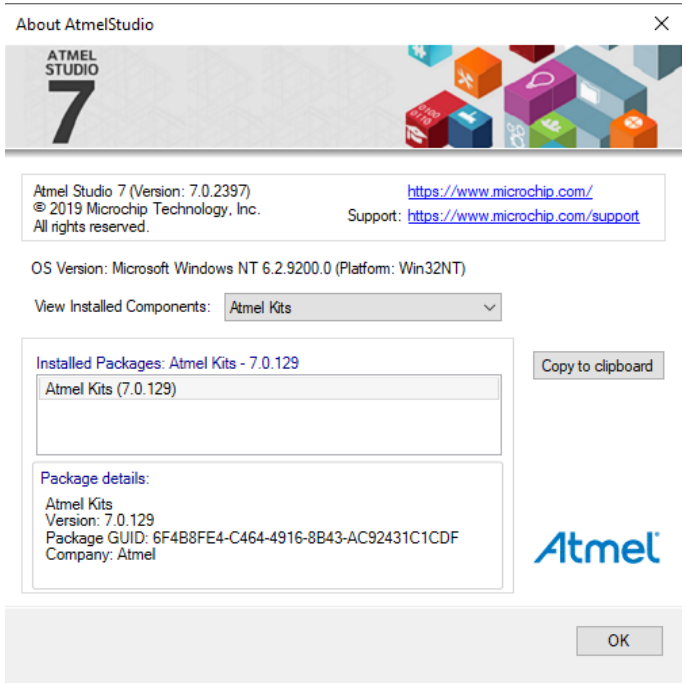
Ένα παράδειγμα της σύνταξης γραμμών κώδικα παρουσιάζεται παρακάτω:

```
<label>  <opcode>     <operand>   <;comments>
<label>  <opcode>     <operands>  <;comments>
<label>  <directive>  <operands>  <;comments>
Comments
Empty line
```

```
test:    rjmp    test        ; Infinite loop (Instruction)
main:    ldi     R16, 9      ; Store value 9 into register R16
label:   .EQU    var1=100    ; Set var1 to 100 (Directive)
         .EQU    var2=200    ; Set var2 to 200
; Pure comment line
```

AVR Assembler – Αρχεία Εξόδου



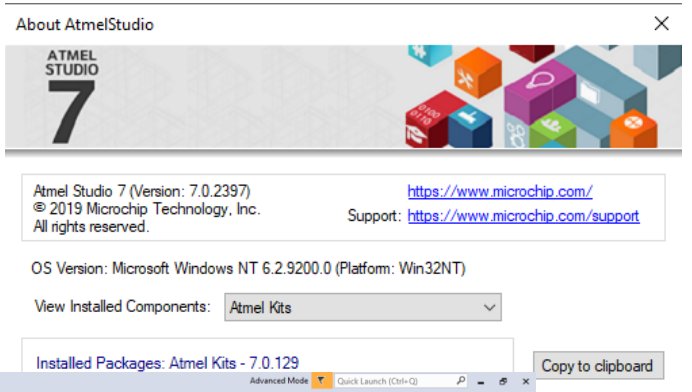


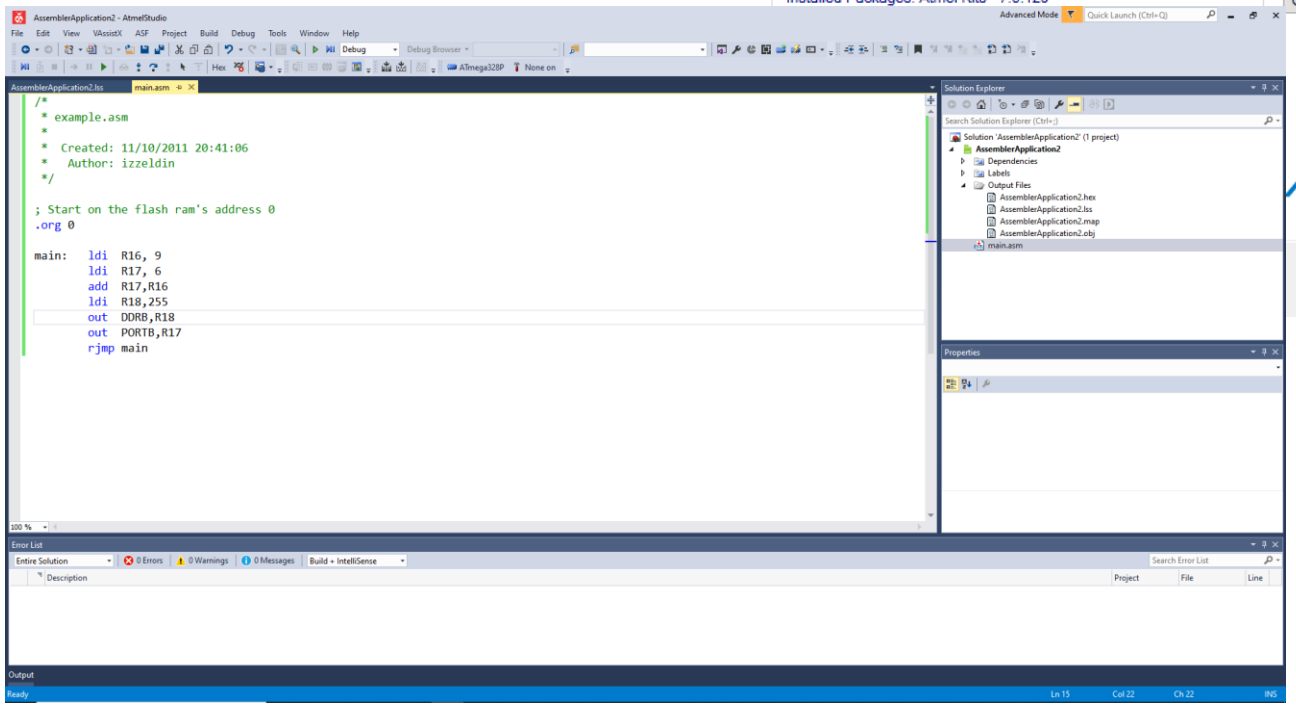
AVR Assembler – Πως λειτουργεί…

- Ο **assembler** είναι υπεύθυνος για τη μετάφραση (**translate**) του προγράμματος assembly σε κώδικα μηχανής (**machine code**).
- Η διαδικασία μετάφρασης αποτελείται από την ανάγνωση εντολής - εντολής και μετατροπής της στον αντίστοιχο κωδικό εντολής μηχανής.
- **LC (location counter)**:
  - Ο προσομοιωτής του Μετρητή Προγράμματος (PC) για το Assembler.
  - Όταν ένα πρόγραμμα συμβολομεταφραστεί, ο LC χρησιμοποιείται για να παρακολουθήσουμε τη θέση μνήμης προγράμματος για κάθε εντολή που πρόκειται να εκτελεστεί (όπως ακριβώς λειτουργεί ο PC στον επεξεργαστή).
  - Με τον τρόπο αυτό παράγεται ορθά ο κώδικας μηχανής από τον κώδικα assembly.
- Καθώς συναντώνται ετικέτες ή ονόματα μεταβλητών, οι διευθύνσεις πρέπει να συμπληρωθούν, οι μετατοπίσεις εντολών branch υπολογίζονται κλπ.
- Οι ετικέτες στην assembly μπορούν να χρησιμοποιηθούν πριν να δηλωθούν.

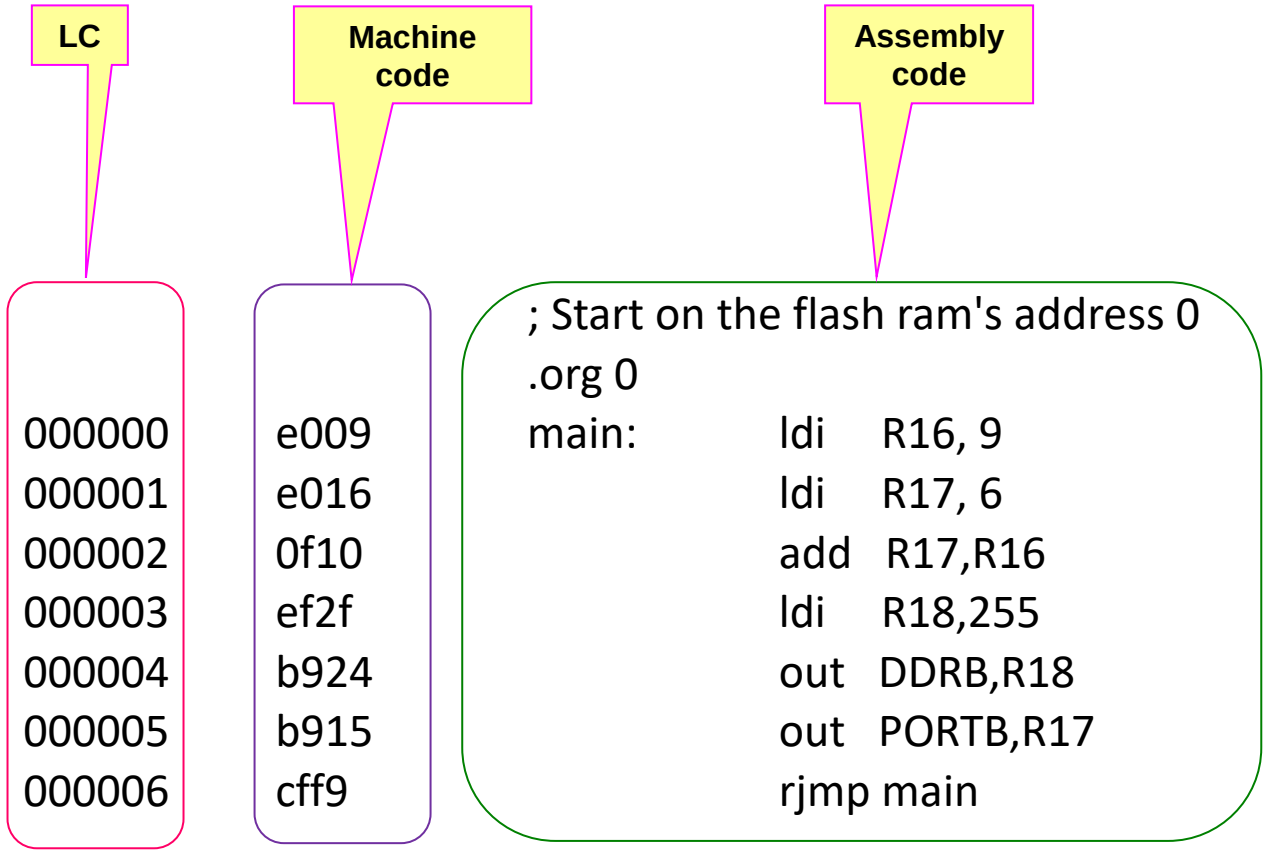
AVR Assembler – Παράδειγμα

```
; Start on the flash ram's address 0
.org 0

main:   ldi   R16, 9
        ldi   R17, 6
        add   R17,R16
        ldi   R18,255
        out   DDRB,R18
        out   PORTB,R17
        rjmp main
```





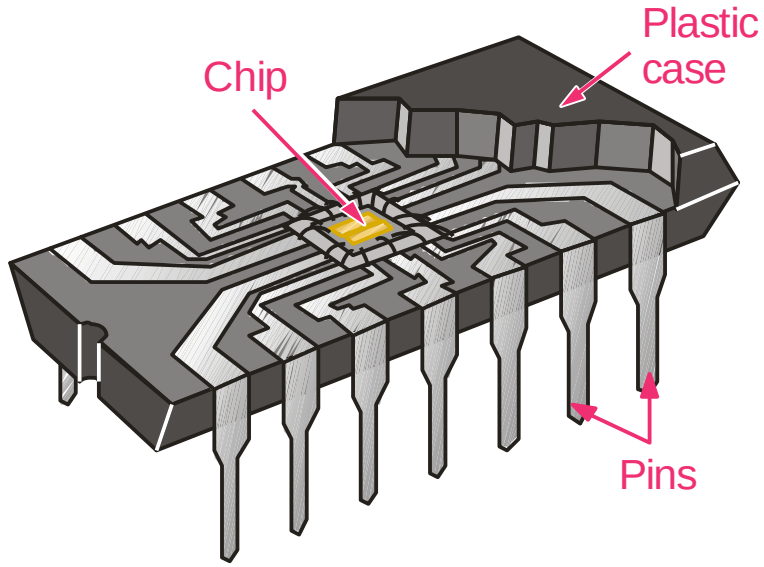
AVR Assembler – Παράδειγμα

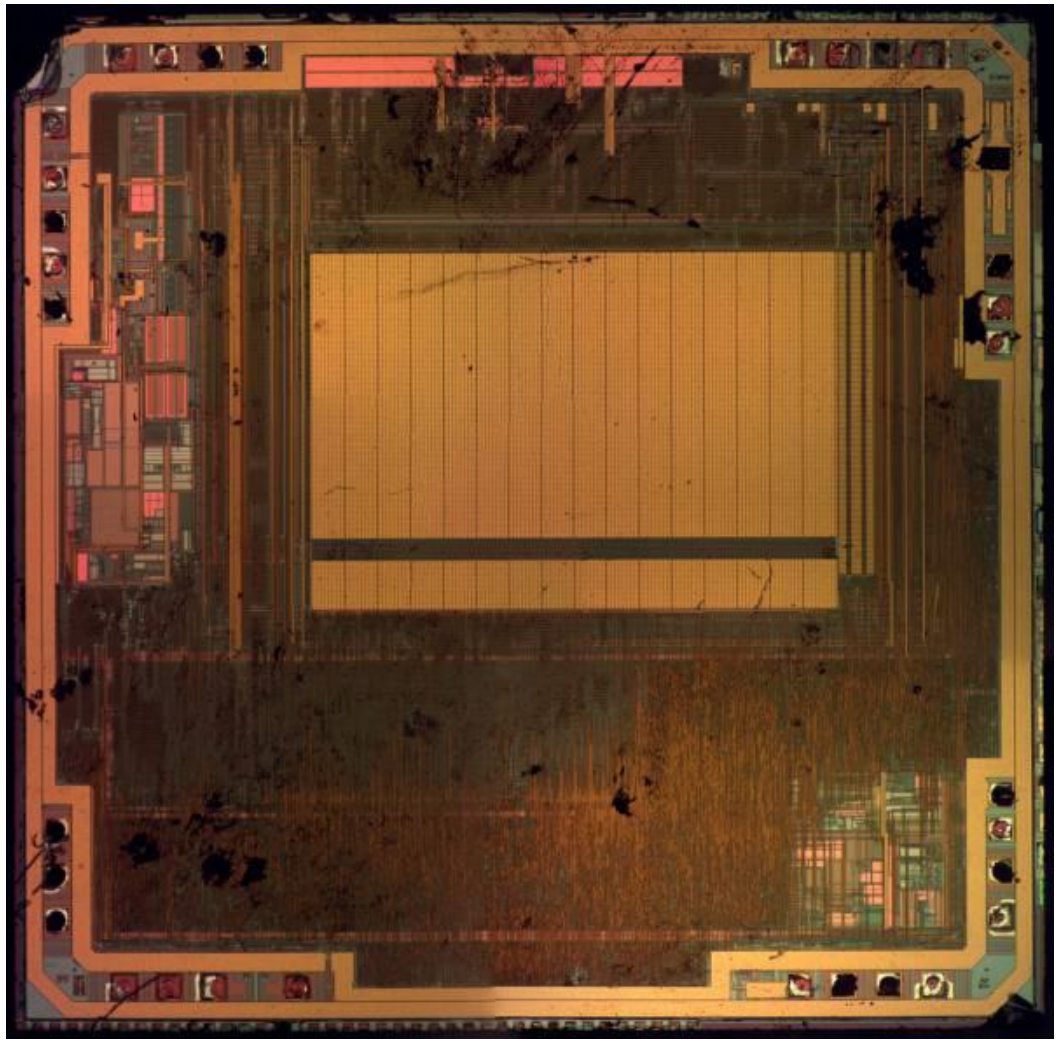


Assembly complete, 0 errors, 0 warnings

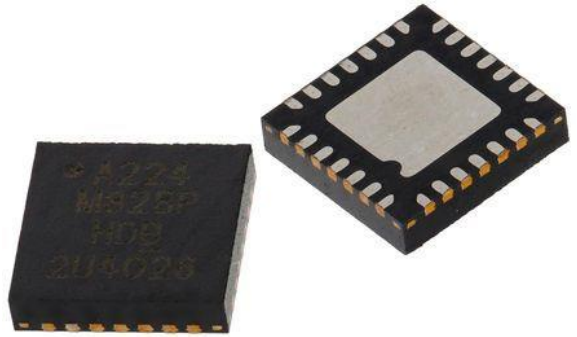
```
RESOURCE USE INFORMATION
------------------------

Notice:
The register and instruction counts are symbol table hit counts,
and hence implicitly used resources are not counted, eg, the
'lpm' instruction without operands implicitly uses r0 and z,
none of which are counted.

x,y,z are separate entities in the symbol table and are
counted separately from r26..r31 here.

.dseg memory usage only counts static data declared with .byte

"ATmega328P" register use summary:
x  :   0 y  :   0 z  :   0 r0 :   0 r1 :   0 r2 :   0 r3 :   0 r4 :   0
r5 :   0 r6 :   0 r7 :   0 r8 :   0 r9 :   0 r10:   0 r11:   0 r12:   0
r13:   0 r14:   0 r15:   0 r16:   2 r17:   3 r18:   2 r19:   0 r20:   0
r21:   0 r22:   0 r23:   0 r24:   0 r25:   0 r26:   0 r27:   0 r28:   0
r29:   0 r30:   0 r31:   0
Registers used: 3 out of 35 (8.6%)

"ATmega328P" instruction use summary:
.lds  :   0 .sts  :   0 adc   :   0 add   :   1 adiw  :   0 and   :   0
andi  :   0 asr   :   0 bclr  :   0 bld   :   0 brbc  :   0 brbs  :   0
brcc  :   0 brcs  :   0 break :   0 breq  :   0 brge  :   0 brhc  :   0
brhs  :   0 brid  :   0 brie  :   0 brlo  :   0 brlt  :   0 brmi  :   0
brne  :   0 brpl  :   0 brsh  :   0 brtc  :   0 brts  :   0 brvc  :   0
brvs  :   0 bset  :   0 bst   :   0 call  :   0 cbi   :   0 cbr   :   0
clc   :   0 clh   :   0 cli   :   0 cln   :   0 clr   :   0 cls   :   0
clt   :   0 clv   :   0 clz   :   0 com   :   0 cp    :   0 cpc   :   0
cpi   :   0 cpse  :   0 dec   :   0 eor   :   0 fmul  :   0 fmuls :   0
fmulsu:   0 icall :   0 ijmp  :   0 in    :   0 inc   :   0 jmp   :   0
ld    :   0 ldd   :   0 ldi   :   3 lds   :   0 lpm   :   0 lsl   :   0
lsr   :   0 mov   :   0 movw  :   0 mul   :   0 muls  :   0 mulsu :   0
neg   :   0 nop   :   0 or    :   0 ori   :   0 out   :   2 pop   :   0
push  :   0 rcall :   0 ret   :   0 reti  :   0 rjmp  :   1 rol   :   0
ror   :   0 sbc   :   0 sbci  :   0 sbi   :   0 sbic  :   0 sbis  :   0
sbiw  :   0 sbr   :   0 sbrc  :   0 sbrs  :   0 sec   :   0 seh   :   0
sei   :   0 sen   :   0 ser   :   0 ses   :   0 set   :   0 sev   :   0
sez   :   0 sleep :   0 spm   :   0 st    :   0 std   :   0 sts   :   0
sub   :   0 subi  :   0 swap  :   0 tst   :   0 wdr   :   0
Instructions used: 4 out of 113 (3.5%)

"ATmega328P" memory use summary [bytes]:
Segment   Begin    End      Code   Data   Used    Size   Use%
----------------------------------------------------------------
[.cseg] 0x000000 0x00000e     14      0     14   32768   0.0%
[.dseg] 0x000100 0x000100      0      0      0    2048   0.0%
[.eseg] 0x000000 0x000000      0      0      0    1024   0.0%

Assembly complete, 0 errors, 0 warnings
```

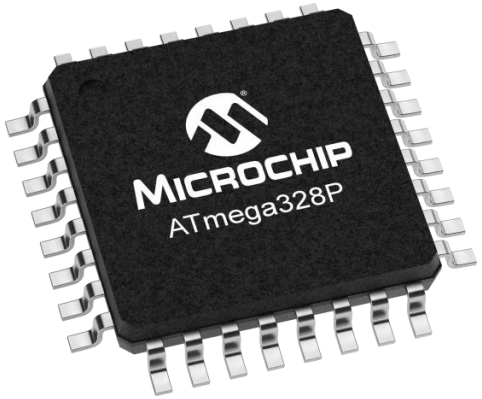
ATMega328P Chip Die (Ψηφίδα)



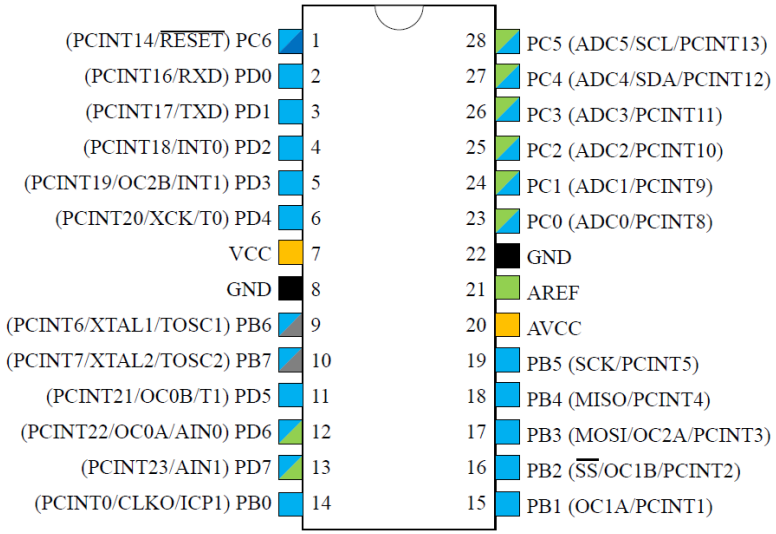


ATMega328P (Ολοκληρωμένο)

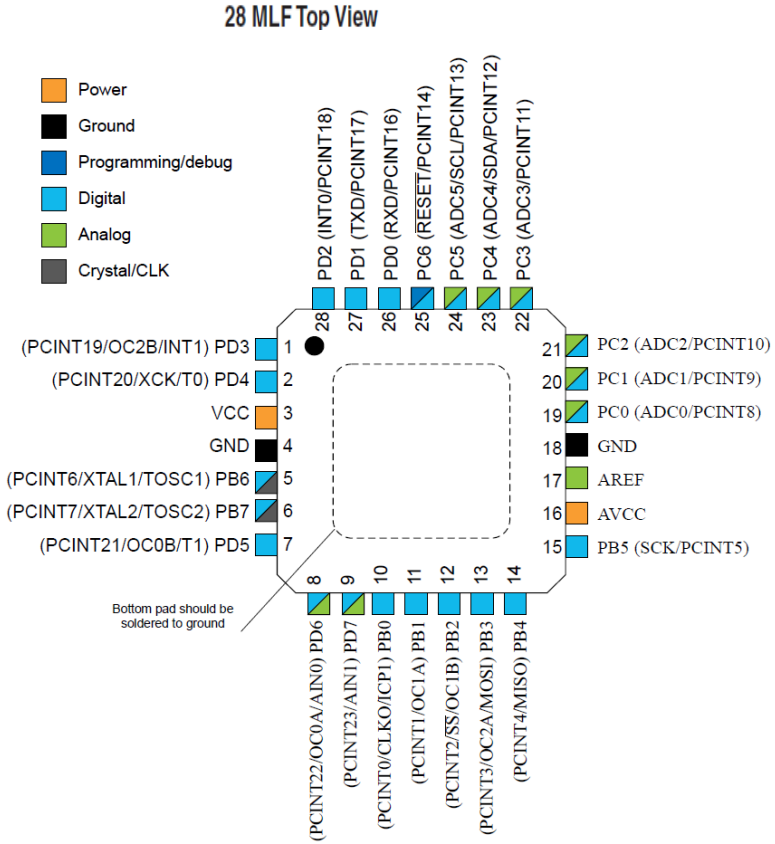




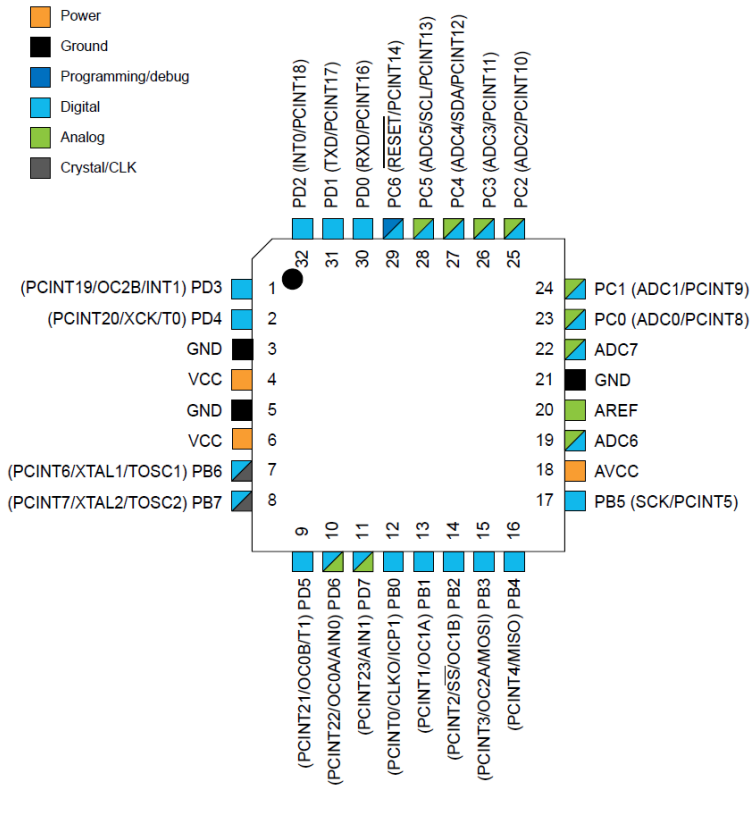
PDIP



28 MLF Top View



TQFP Top View



ATMega328P (Arduino)

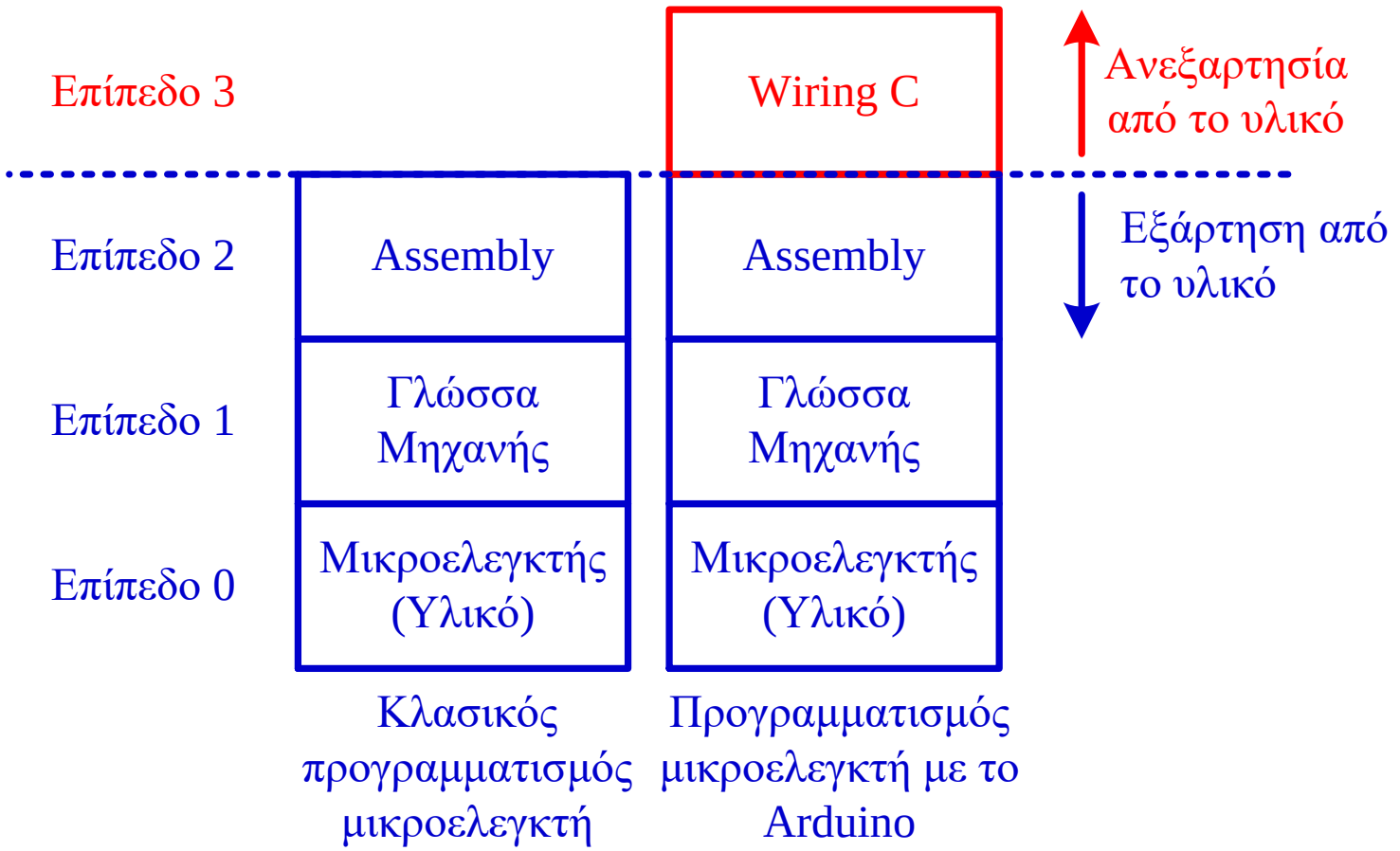
Ο Κινέζος!!!

Η ΑΝΑΠΤΥΞΙΑΚΗ ΠΛΑΤΦΟΡΜΑ ARDUINO

Η διαφορά αυτής της πλατφόρμας σε σχέση με τους μικροελεγκτές που έχετε μελετήσει μέχρι τώρα έγκειται στο επίπεδο αφαίρεσης στο οποίο μπορεί να γίνει ο προγραμματισμός.

Με άλλα λόγια, ο προγραμματισμός του μικροελεγκτή στην πλατφόρμα Arduino γίνεται με τη γλώσσα Wiring C που αποτελεί μια παραλλαγή της γνωστής C++.

Έτσι, ο προγραμματισμός μιας τέτοιας πλατφόρμας είναι φιλικότερος και επιτρέπει τη γρηγορότερη ανάπτυξη εφαρμογών χωρίς να προϋποθέτει τη λεπτομερή γνώση της αρχιτεκτονικής του μικροελεγκτή.

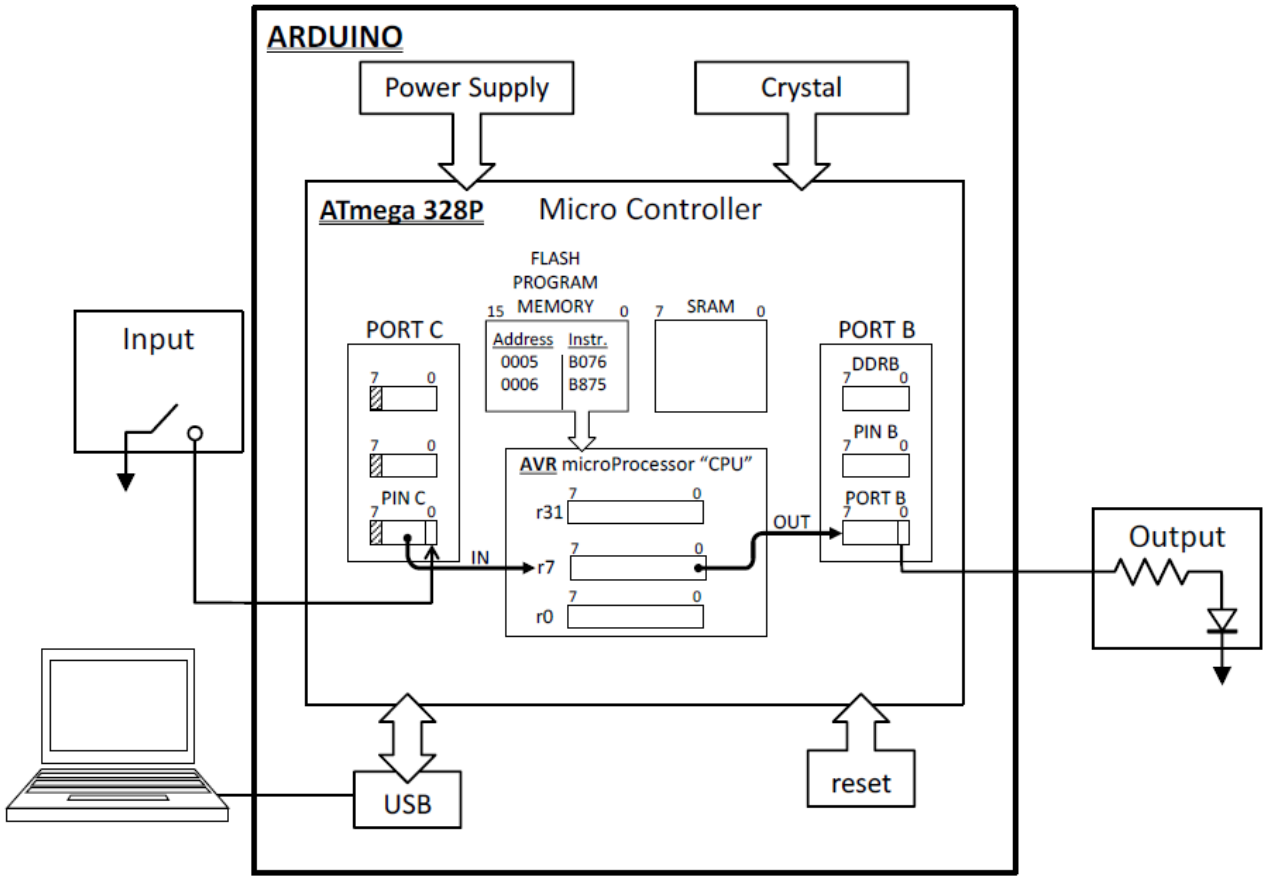


Η ΠΡΟΣΕΓΓΙΣΗ ARDUINO

Ο προγραμματισμός οποιουδήποτε μικροελεγκτή προϋποθέτει την ύπαρξη είτε στοιχειωδών κυκλωμάτων, είτε συγκεκριμένων διασυνδέσεων (μέσω κατάλληλων αντιστάσεων) με ένα υπολογιστικό σύστημα (π.χ. μέσω σειριακής ή παράλληλης θύρας), είτε ειδικού προγραμματιστή.

Στην πλατφόρμα Arduino η συγκεκριμένη τεχνολογία απαλλάσσει το σχεδιαστή ή τον προγραμματιστή από όλες αυτές τις λεπτομέρειες και τα κυκλώματα.

Οτιδήποτε είναι απαραίτητο για τον προγραμματισμό του μικροελεγκτή, βρίσκεται πάνω στην κάρτα στην οποία είναι τοποθετημένος.

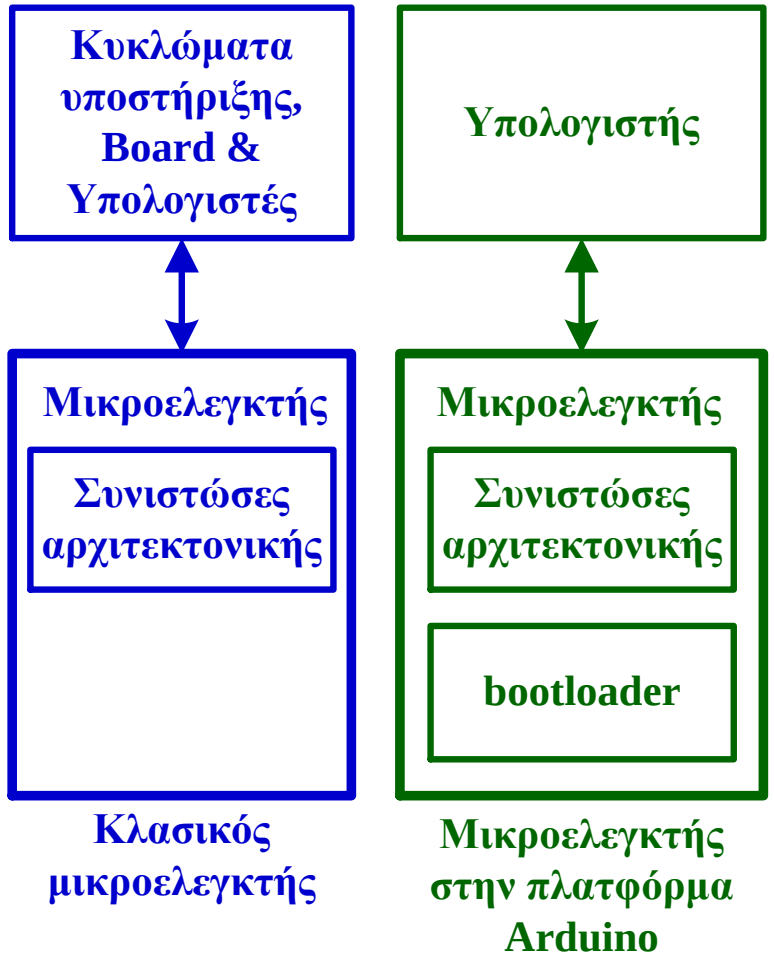


Η ΠΡΟΣΕΓΓΙΣΗ ARDUINO

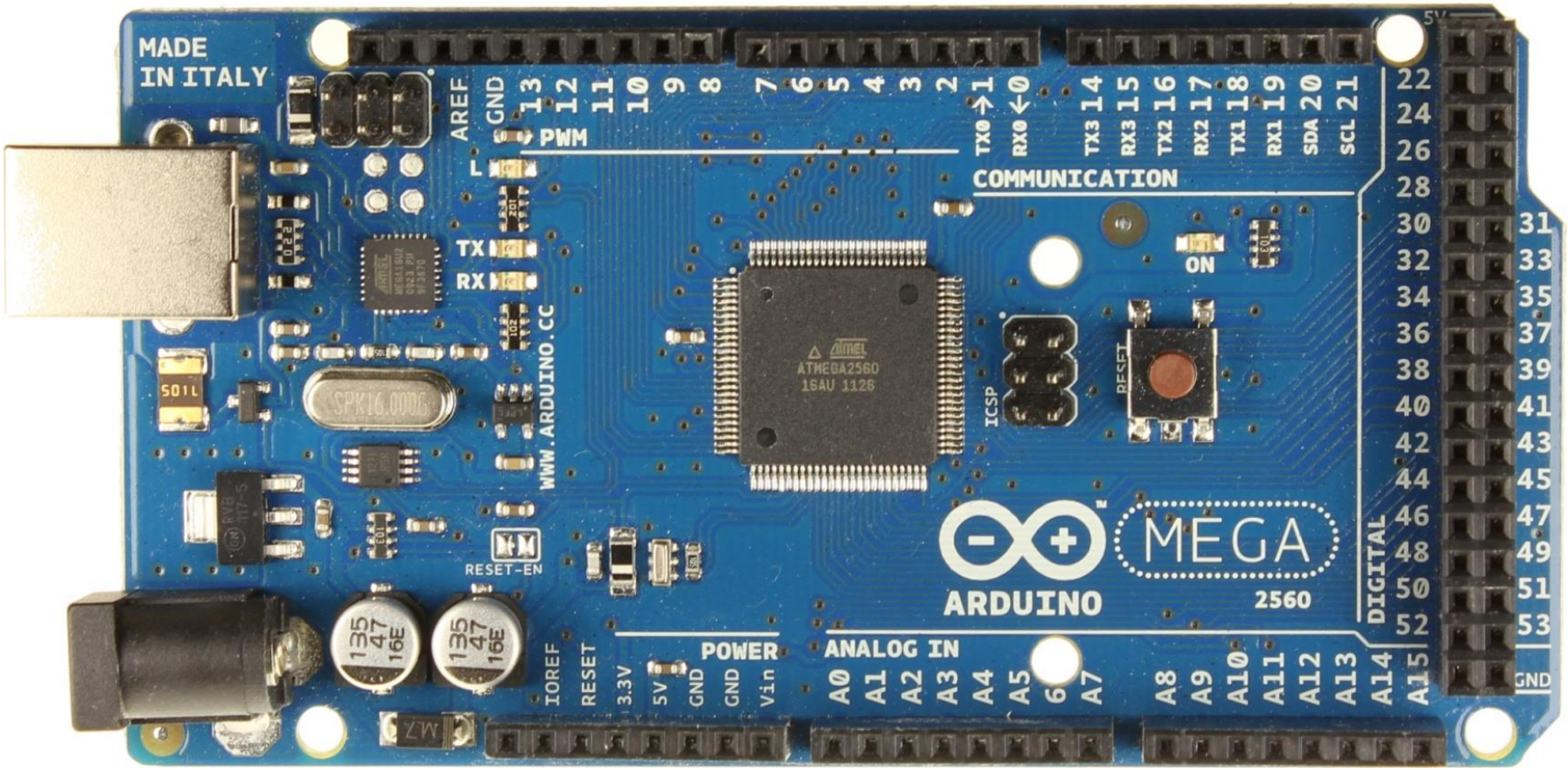
Το πρόγραμμα αναπτύσσεται στον υπολογιστή, ενώ η μεταφόρτωση του εκτελέσιμου κώδικα γίνεται μέσω της θύρας USB.

Επιπλέον, οι μικροελεγκτές που τοποθετούνται στα Arduino είναι εφοδιασμένοι με ένα ειδικό λογισμικό που ονομάζεται bootloader και δίνει τη δυνατότητα του επαναπρογραμματισμού.

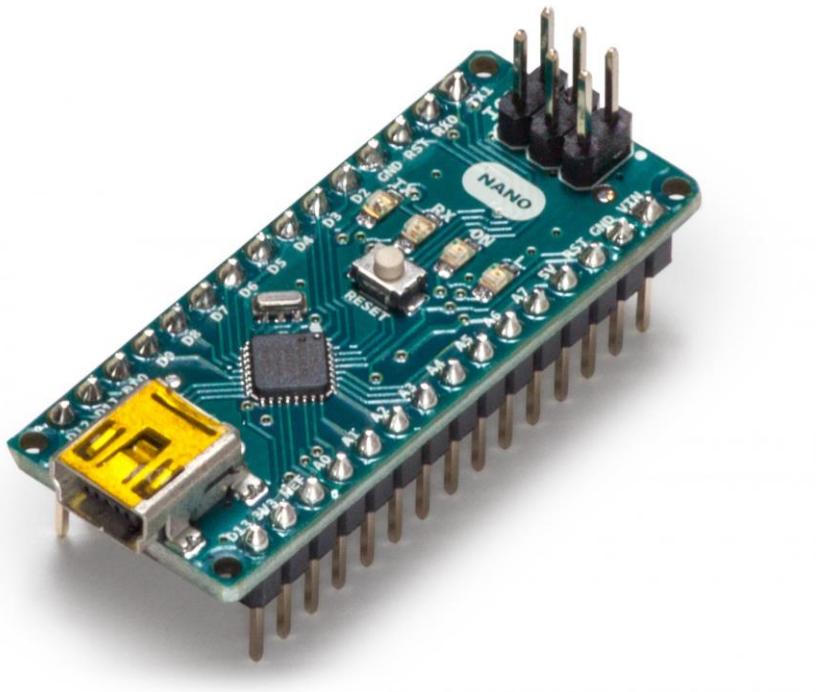
Οι πιο προχωρημένοι έχουν πάντα τη δυνατότητα να προγραμματίσουν τον μικροελεγκτή του Arduino, όπως και κάθε άλλο μικροελεγκτή, αφαιρώντας τον bootloader ή και χρησιμοποιώντας άλλες διασυνδέσεις και κυκλώματα.



ARDUINO Mega REV3 (ATmega2560)

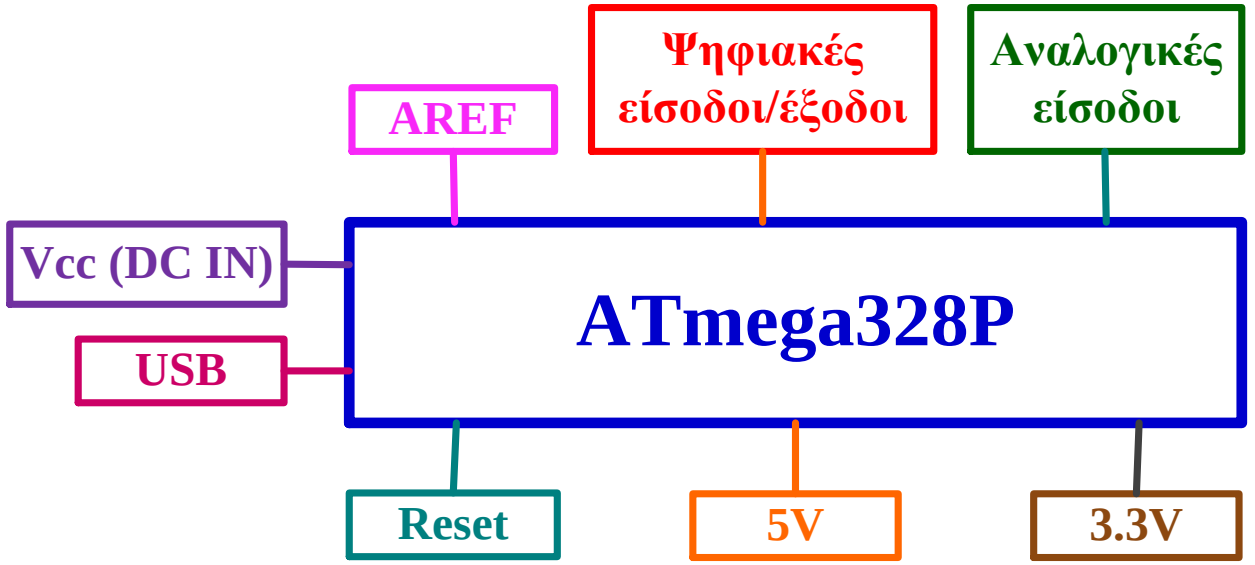


ARDUINO Nano (ATmega328)



ARDUINO UNO (ATmega328P)

Το ARDUINO UNO

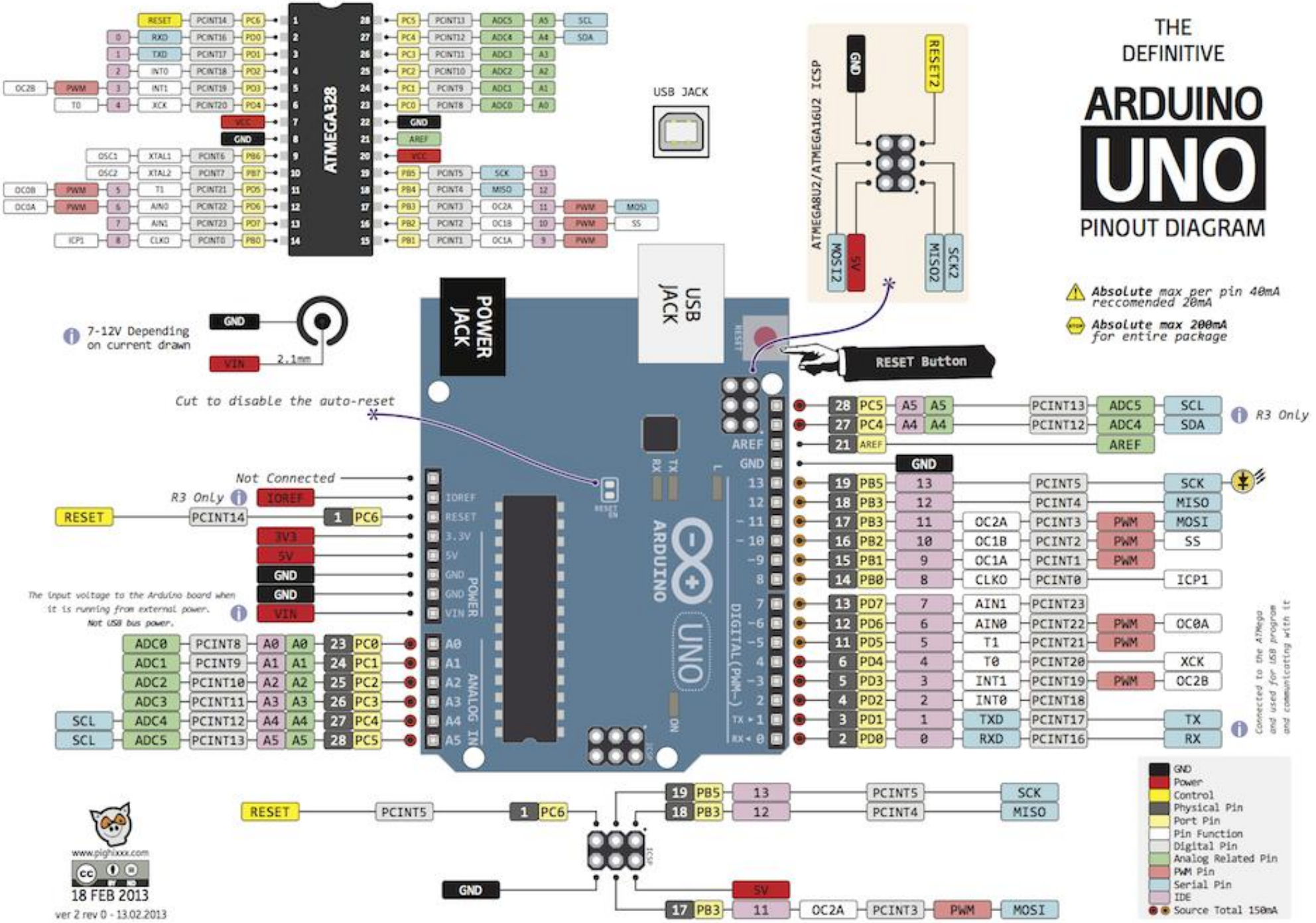


Το Arduino UNO που χρησιμοποιεί τον μικροελεγκτή ATmega328P, είναι η πιο δημοφιλής έκδοση Arduino. Σχεδόν όλες οι υπόλοιπες εκδόσεις βασίζονται στην ίδια φιλοσοφία. Το σχήμα παρουσιάζει σε διάγραμμα βαθμίδας τα βασικά χαρακτηριστικά και δυνατότητες που προσφέρονται μέσω της κάρτας.

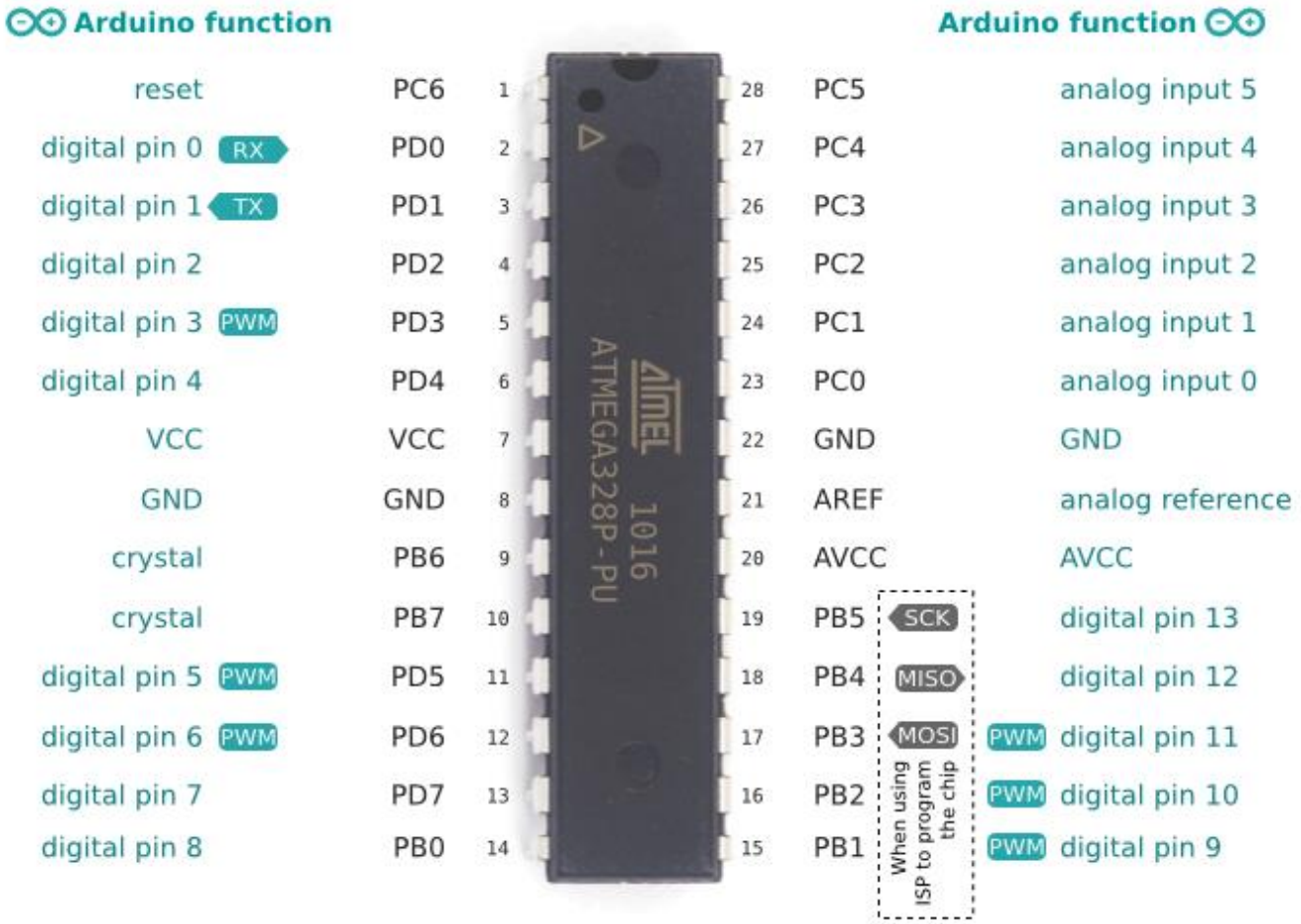
Βασικά Χαρακτηριστικά Arduino UNO R3

Πίνακας Β΄.3: Χαρακτηριστικά της αναπτυξιακής πλακέτας Arduino UNO R3

Μικροελεγκτής	ATMEGA328
Τάση λειτουργίας	5V DC
Τάση εισόδου	7-12V DC
Όρια τάσης εισόδου	6-20V DC
Ψηφιακοί ακροδέκτες I/O	14, (6 PWM έξοδοι)
Αναλογικοί ακροδέκτες εισόδου	6
Ισχύς συνεχόμενου ρεύματος ανά ακροδέκτη	40mA
Ισχύς συνεχόμενου ρεύματος για ακροδέκτη τάσης 3.3V	50mA
Μνήμη flash	32KB (ATMEGA328)
Μνήμη SRAM	2KB (ATMEGA328)
Μνήμη EEPROM	1KB (ATMEGA328)
Ταχύτητα ρολογιού	16MHz


THE DEFINITIVE
ARDUINO UNO
PINOUT DIAGRAM
ATMEGA328
USB JACK
ATMEGA8U2/ATMEGA16U2 ICSP
POWER JACK
USB JACK
RESET Button
Absolute max per pin 40mA
reccomended 20mA
Absolute max 200mA
for entire package
7-12V Depending on current drawn
2.1mm
Cut to disable the auto-reset
Not Connected
R3 Only
The input voltage to the Arduino board when it is running from external power.
Not USB bus power.
Connected to the ATMega and used for USB program and communicating with it
ARDUINO UNO
DIGITAL (PWM~)
ANALOG IN
POWER
www.pighixxx.com
18 FEB 2013
ver 2 rev 0 - 13.02.2013
GND
Power
Control
Physical Pin
Port Pin
Pin Function
Digital Pin
Analog Related Pin
PWM Pin
Serial Pin
IDE
Source Total 150mA

ATmega328P pin mapping



Το ARDUINO UNO

- **Vcc (DC IN)** Εξωτερική τροφοδοσία για αυτόνομη λειτουργία χωρίς υπολογιστή. Η τροφοδοσία αυτή μπορεί να προέρχεται από τροφοδοτικό, μπαταρία ή άλλο παρόμοιο στοιχείο.
- **USB** Σύνδεση με υπολογιστή μέσω USB για μεταφόρτωση κώδικα και τροφοδοσία. Η επικοινωνία USB υποστηρίζεται από ειδικό ολοκληρωμένο κύκλωμα (IC), το οποίο λειτουργεί συμπληρωματικά με τον μικροελεγκτή.
- **AREF** Τάση αναφοράς, που καθορίζει την ακρίβεια της μέτρησης όταν γίνεται ανάγνωση από τις αναλογικές εισόδους.
- **Reset** Υποστήριξη κουμπιού αλλά και εισόδου reset που ελέγχεται από το πρόγραμμα.
- **Digital I/O Ψηφιακές είσοδοι και έξοδοι**, που αναγνωρίζουν τις στάθμες 0 ή 5V. Επιπλέον, με την τεχνική διαμόρφωσης εύρους παλμού PWM (Pulse Width Modulation) μπορούν να παραχθούν και ψευδοαναλογικά σήματα.
- **Analog Inputs** Ανάγνωση αναλογικών σημάτων 0 έως 5V με δυνατότητα αποτύπωσης σε 1024 στάθμες.
- **3.3V** Τροφοδοσία εξωτερικών κυκλωμάτων στα 3.3V (π.χ. συνδεδεμένων αισθητήρων).
- **5V** Τροφοδοσία εξωτερικών κυκλωμάτων στα 5V (π.χ. συνδεδεμένων αισθητήρων).

Το ARDUINO UNO

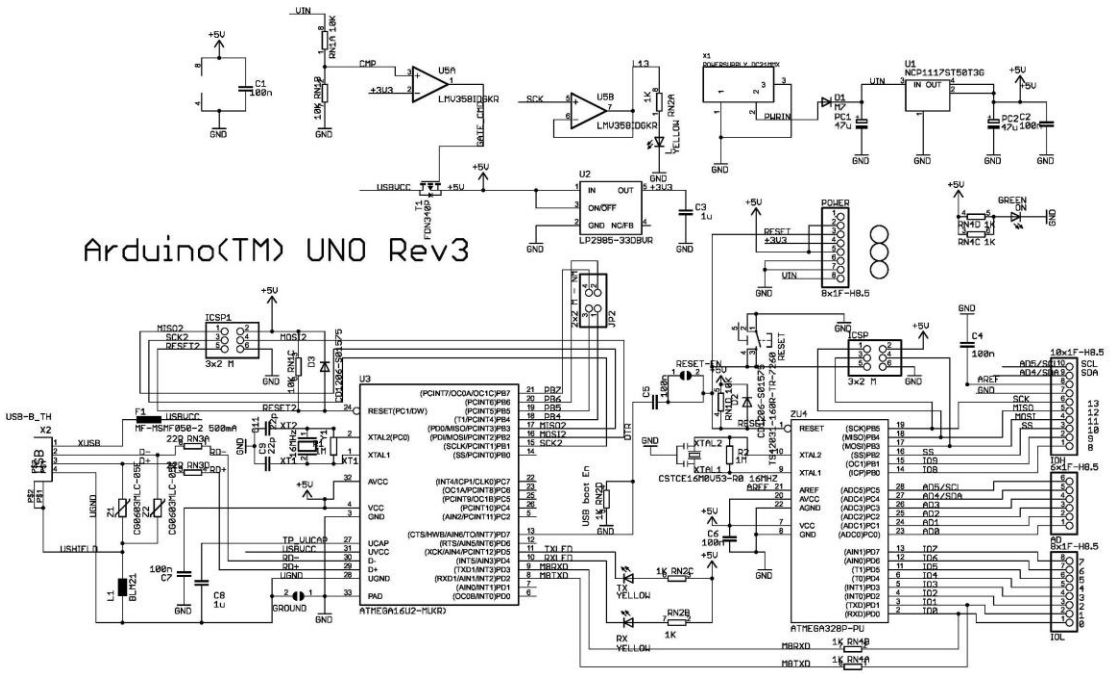
Όλες οι ακίδες αξιοποιούνται μέσω της κάρτας που φιλοξενεί τον μικροελεγκτή. Ο μικροελεγκτής μετά τον προγραμματισμό του μπορεί ακόμα και να αφαιρεθεί από την κάρτα και να τοποθετηθεί για αυτόνομη λειτουργία αφού πλαισιωθεί από τα κατάλληλα κυκλώματα.

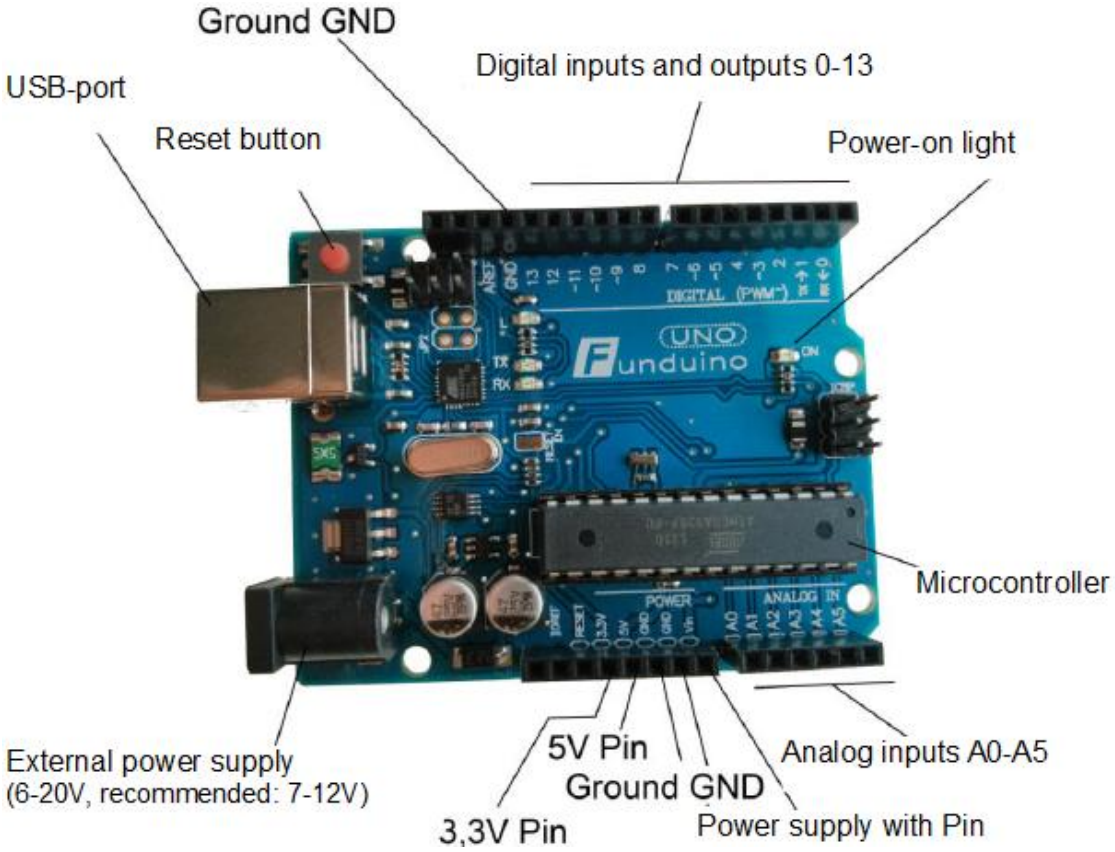
Η κάρτα δηλαδή μπορεί να χρησιμοποιείται μόνο για τον προγραμματισμό. Φυσικά υπάρχει και η δυνατότητα τοποθέτησης ολόκληρου του Arduino στο σημείο ενδιαφέροντος για την ολοκλήρωση της εφαρμογής.

Συμπερασματικά:

- ✓ Το Arduino είναι ένα «πακέτο» που προσφέρει άμεση αξιοποίηση και προγραμματισμό ενός μικροελεγκτή.
- ✓ Οι θύρες επικοινωνίας και οι ακίδες του μικροελεγκτή προσφέρονται μέσω της κάρτας για εύκολες συνδέσεις.
- ✓ Ο προγραμματισμός γίνεται μέσω της θύρας USB.
- ✓ Η ανάπτυξη του κώδικα γίνεται σε γλώσσα τύπου C/C++.

FUNDUINO


Arduino(TM) UNO Rev3

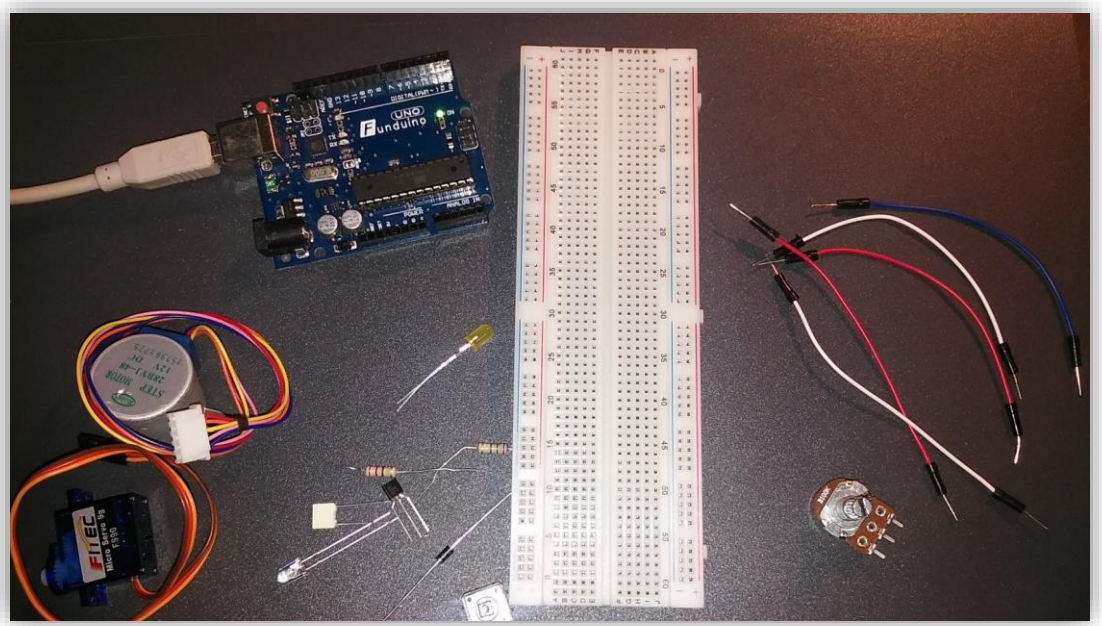

Ground GND
Digital inputs and outputs 0-13
USB-port
Reset button
Power-on light
Microcontroller
External power supply
(6-20V, recommended: 7-12V)
5V Pin
Ground GND
3,3V Pin
Analog inputs A0-A5
Power supply with Pin

IEES Lab box – τα βασικά για την πρώτη γνωριμία με το Arduino

Για να ξεκινήσουμε την χρήση του Arduino χρειαζόμαστε απλά εξαρτήματα και καλώδια:

- ✓ Το Arduino μας.
- ✓ Καλώδιο USB (με ανάλογο τύπο βύσματος).
- ✓ Αναπτυξιακή πλακέτα Breadboard.
- ✓ Φωτοδιόδους LED.
- ✓ Αντιστάτες (300Ω, 1ΚΩ, 10ΚΩ).
- ✓ Καλώδια για διασυνδέσεις.
- ✓ Ποτενσιόμετρο.
- ✓ Διακόπτες push button.

…και φυσικά το Arduino IDE!!!



Αντιστάτες – R

390KΩ +-10%

339Ω +-1%

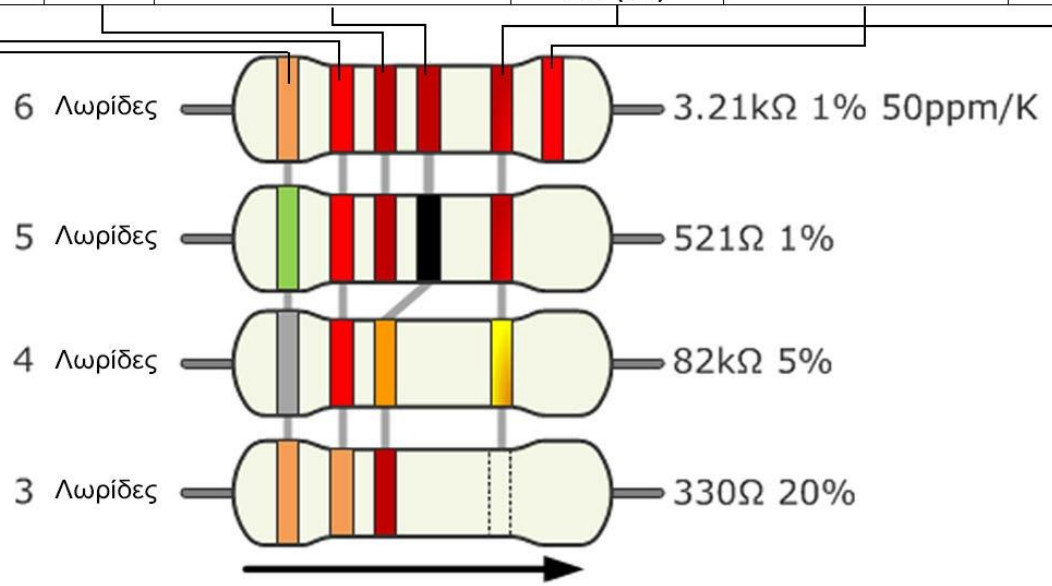
Οι αντιστάσεις ακριβείας, όπως είναι οι αντιστάσεις μεταλλικού στρώματος έχουν 5 χρωματικές λωρίδες αντί για 4. Η πέμπτη χρωματική λωρίδα θα βρίσκεται ποιο απομακρυσμένη από τις υπόλοιπες που δηλώνει την ανοχή της αντίστασης. Συνήθως στις αντιστάσεις 5 χρωμάτων η ανοχή είναι χρώματος καφέ ή κόκκινο.

Ο υπολογισμός της ονομαστικής τιμής σε αυτές τις περιπτώσεις γίνεται ως εξής:

Οι τρεις πρώτες χρωματικές ζώνες μας δίνουν την τιμή της αντίστασης και η τέταρτη ζώνη είναι ο πολλαπλασιαστής. Αν έχουμε μια αντίσταση 5 χρωματικών λωρίδων με χρώματα πορτοκαλί, πορτοκαλί, λευκό, μαύρο, καφέ, τότε αφού βρούμε το χρώμα της ανοχής που είναι το ποιο απομακρυσμένο, το φέρνουμε από την δεξιά μεριά οπότε διαβάζουμε την τιμή από αριστερά προς τα δεξιά ως εξής: 339Χ1=339Ω και ανοχή 1% σύμφωνα με τον πίνακα χρωμάτων.
Σε περίπτωση που η αντίστασή μας έχει 6 χρωματικές λωρίδες, τότε διαβάζουμε την αντίσταση όπως κάναμε στις αντιστάσεις 5 χρωματικών λωρίδων και η 6η χρωματική λωρίδα δηλώνει τον συντελεστή θερμοκρασίας, όπως φαίνεται και στον πίνακα χρωμάτων.

Αντιστάτες – R

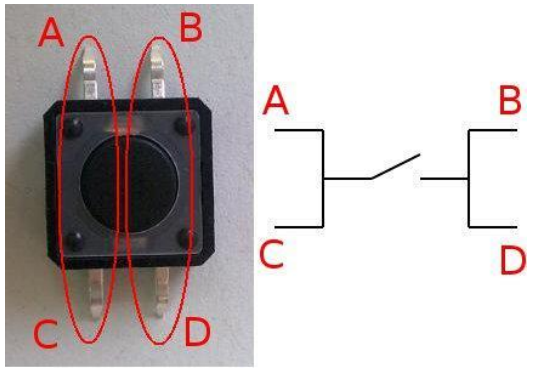
Χρώμα	Σημαντικά ψηφία			Πολλαπλασιαστής	Ανοχή (%)	Θερμοκρασία	Βαθμός αποτυχίας
Μαύρο	0	0	0	X1		250 (U)	
Καφέ	1	1	1	X10	1 (F)	100 (S)	1
Κόκκινο	2	2	2	X100	2 (G)	50 (R)	0.1
Πορτοκαλί	3	3	3	X1K		15 (P)	0.01
Κίτρινο	4	4	4	X10K		25 (Q)	0.001
Πράσινο	5	5	5	X100K	0.5 (D)	20 (Z)	
Μπλε	6	6	6	X1M	0.25 (D)	10 (Z)	
Βιολετί	7	7	7	X10M	0.1 (B)	5 (M)	
Γκρι	8	8	8	X100M	0.05 (A)	1 (K)	
Λευκό	9	9	9	X1G			
Χρυσό			**	X0.1	5 (J)		
Ασημί				X0.01	10 (K)		
Κενό					20 (M)		

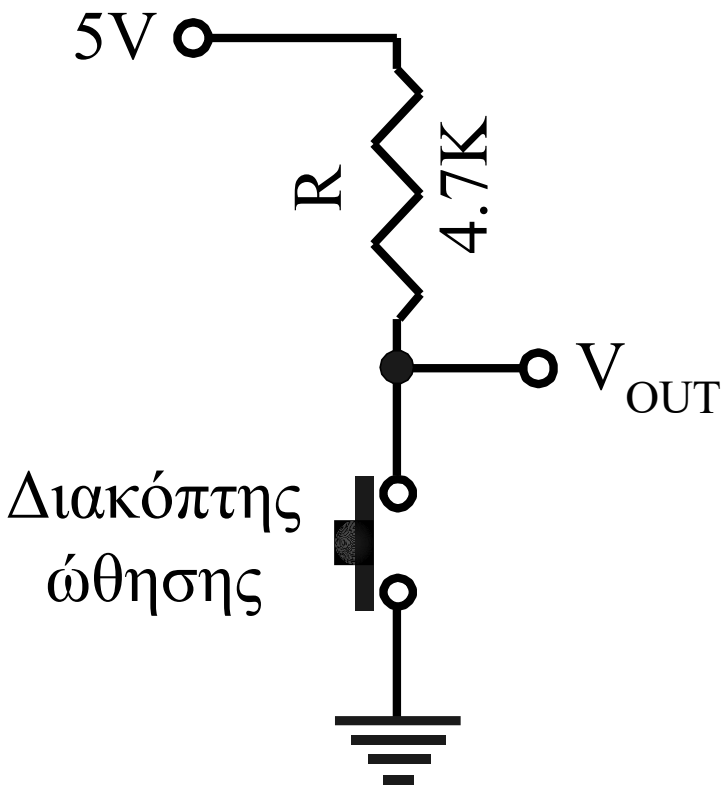


Αντιστάτες – R

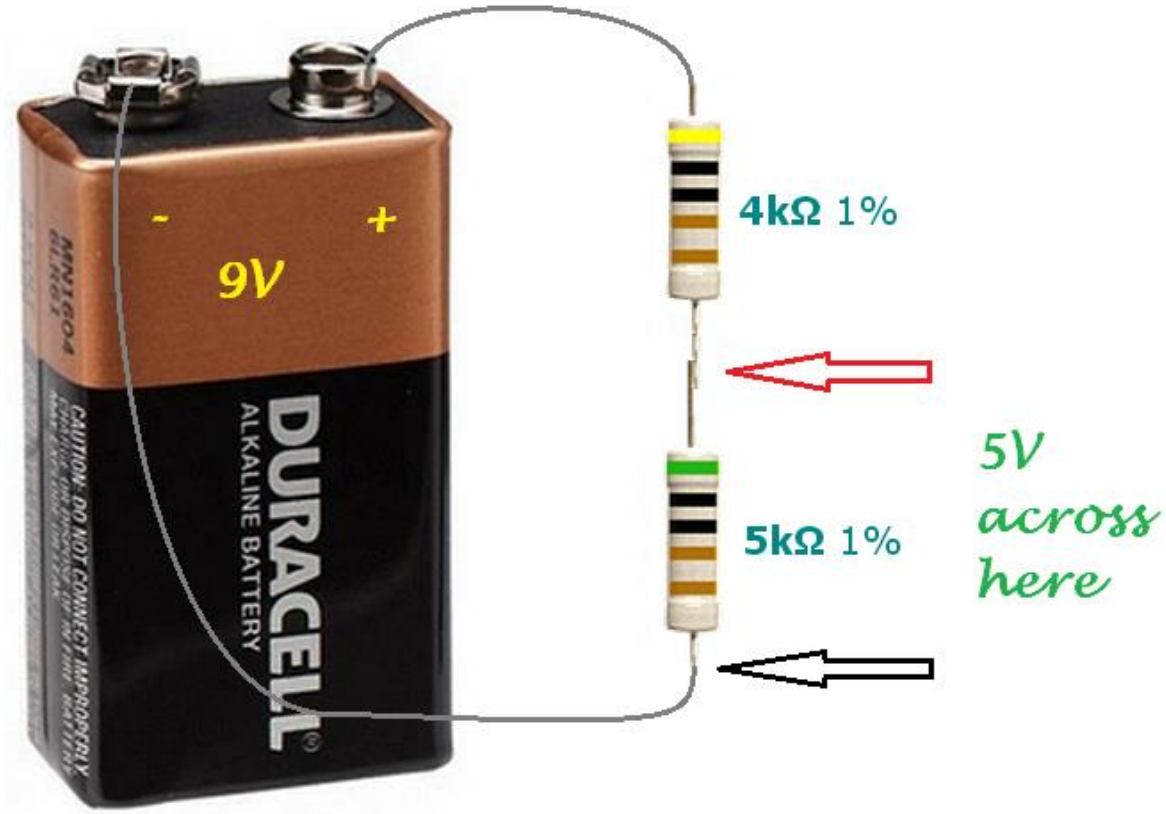
Standard Resistor Values (±5%)						
1.0	10	100	1.0K	10K	100K	1.0M
1.1	11	110	1.1K	11K	110K	1.1M
1.2	12	120	1.2K	12K	120K	1.2M
1.3	13	130	1.3K	13K	130K	1.3M
1.5	15	150	1.5K	15K	150K	1.5M
1.6	16	160	1.6K	16K	160K	1.6M
1.8	18	180	1.8K	18K	180K	1.8M
2.0	20	200	2.0K	20K	200K	2.0M
2.2	22	220	2.2K	22K	220K	2.2M
2.4	24	240	2.4K	24K	240K	2.4M
2.7	27	270	2.7K	27K	270K	2.7M
3.0	30	300	3.0K	30K	300K	3.0M
3.3	33	330	3.3K	33K	330K	3.3M
3.6	36	360	3.6K	36K	360K	3.6M
3.9	39	390	3.9K	39K	390K	3.9M
4.3	43	430	4.3K	43K	430K	4.3M
4.7	47	470	4.7K	47K	470K	4.7M
5.1	51	510	5.1K	51K	510K	5.1M
5.6	56	560	5.6K	56K	560K	5.6M
6.2	62	620	6.2K	62K	620K	6.2M
6.8	68	680	6.8K	68K	680K	6.8M
7.5	75	750	7.5K	75K	750K	7.5M
8.2	82	820	8.2K	82K	820K	8.2M
9.1	91	910	9.1K	91K	910K	9.1M

Διακόπτης ώθησης (push button)

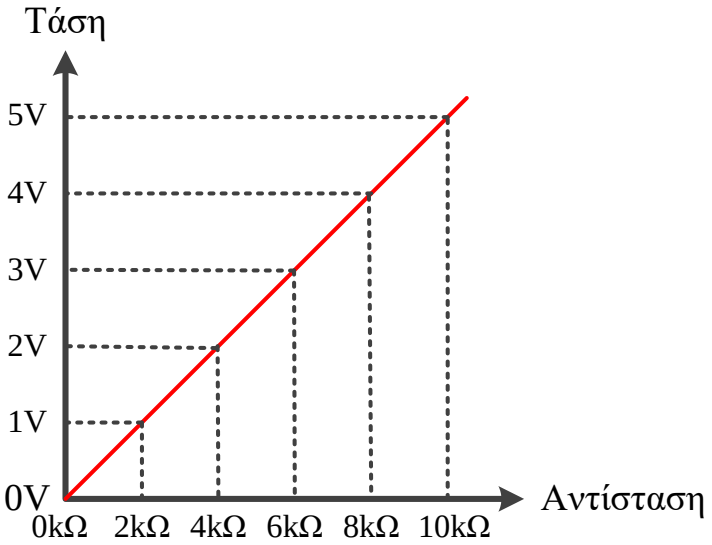


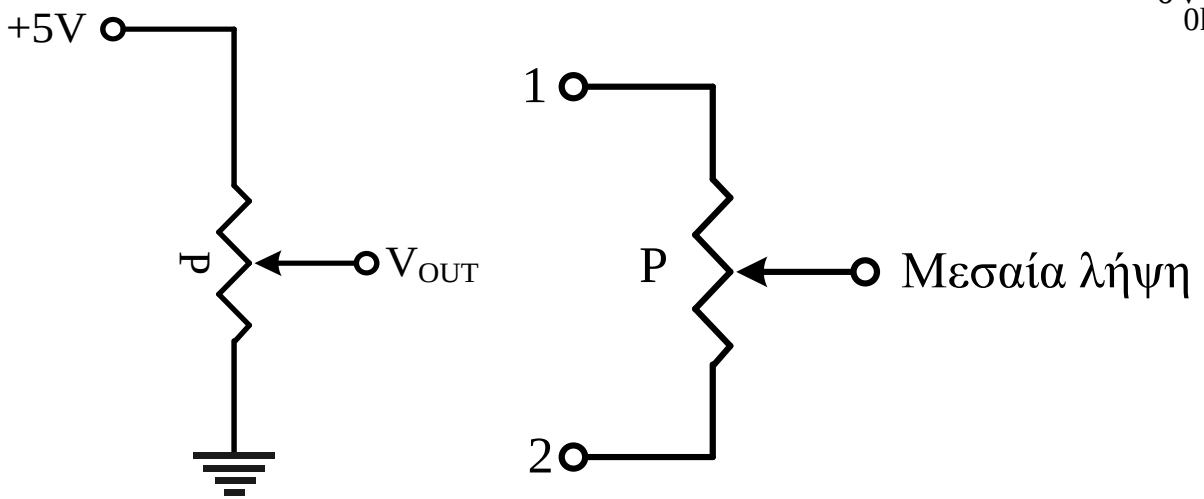


Μπαταρία

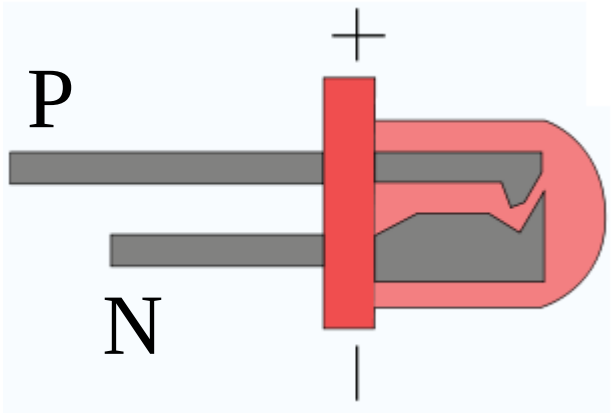


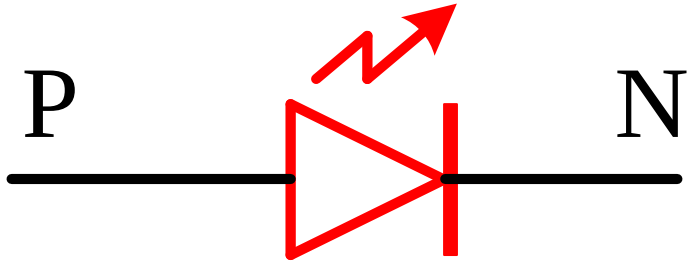
Ποτενσιόμετρο – Ροοστάτης





Φωτοδίοδος LED (Light Emitting Diode)





Η τάση σε διάφορους τύπους LED:

blue: 3.1V, white: 3.3V, green: 3.7V, yellow: 2.2V, red: 2.1V

Ενδεικτικές τιμές αντιστάσεων περιορισμού ρεύματος για διάφορα LED, όταν αυτά συνδέονται σε Pin του μικροελεγκτή που έχουν 5V:

LED:	white	red	yellow	green	blue	IR
resistor:	100 Ohm	200 Ohm	200 Ohm	100 Ohm	100 Ohm	100 Ohm

Τροφοδοσία breadboard/raster