ROBOTICS LECTURE SERIES
INTRODUCTION TO MICROCONTROLLERS

By: Nikhil Soni

Why Microcontrollers?

- Sensors take input
- Actuators give output
- Then why do we need microcontrollers?
- Microcontrollers take input from the sensors, process the data and trigger output accordingly.

MICROCONTROLLERS

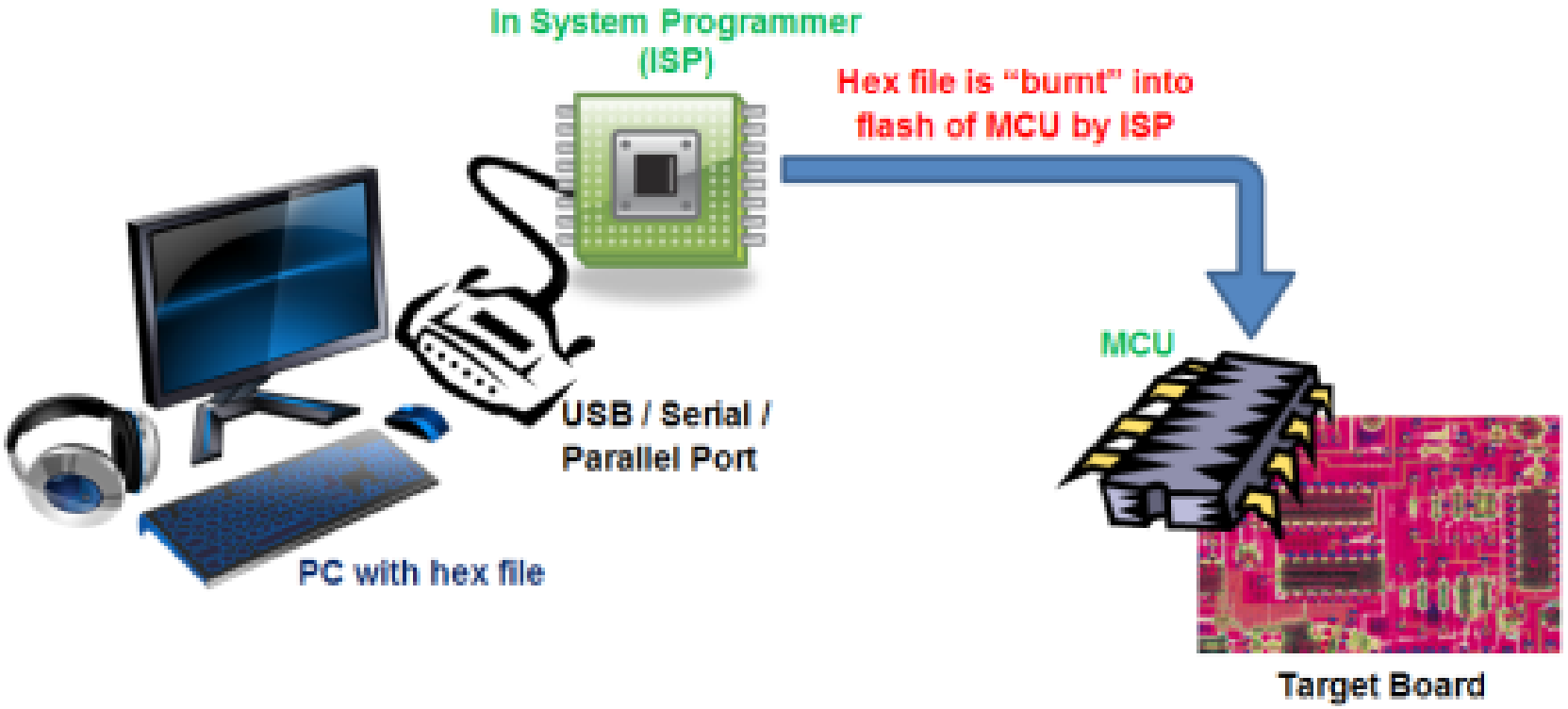
- A microcontroller can be considered as a very small and simple version of a computer on a single IC which is used for a specific purpose.
- It has a CPU, flash memory ,RAM, EEPROM and many on- chip peripherals .

I AM EVERY WHERE!!!!

PlaywithRobots

Basic Tools for AVR MCU

- PC
- In system programmer(ISP): The device which connects our MCU to PC
- A target board
- AVR MCU
- A C compiler which is free in AVR: It converts high level language into machine language(.hex file)
- A programmer software: It loads this .hex file in flash memory of MCU.


In System Programmer
(ISP)
Hex file is "burnt" into
flash of MCU by ISP
MCU
USB / Serial /
Parallel Port
PC with hex file
Target Board

MAIN TERMS USED IN MCU

- **Clock**: It is basically a signal that oscillates between high and low with a fixed time period
- **Clock source**: Generally crystal oscillators are used to provide a clean, voltage and temperature independent clock source.
- **Flash memory**: Program that is to be executed by microcontroller is stored here.

EG: Usually used for program code and data constants.

- **SRAM**: Static Random Access Memory, this is the volatile memory of microcontroller. That means when power is turned off, it loses all its data.

EG: variables assigned by us in code are stored here.

- **EEPROM**: It is Electrically Erasable Programmable ROM. It retains its data even after power off.

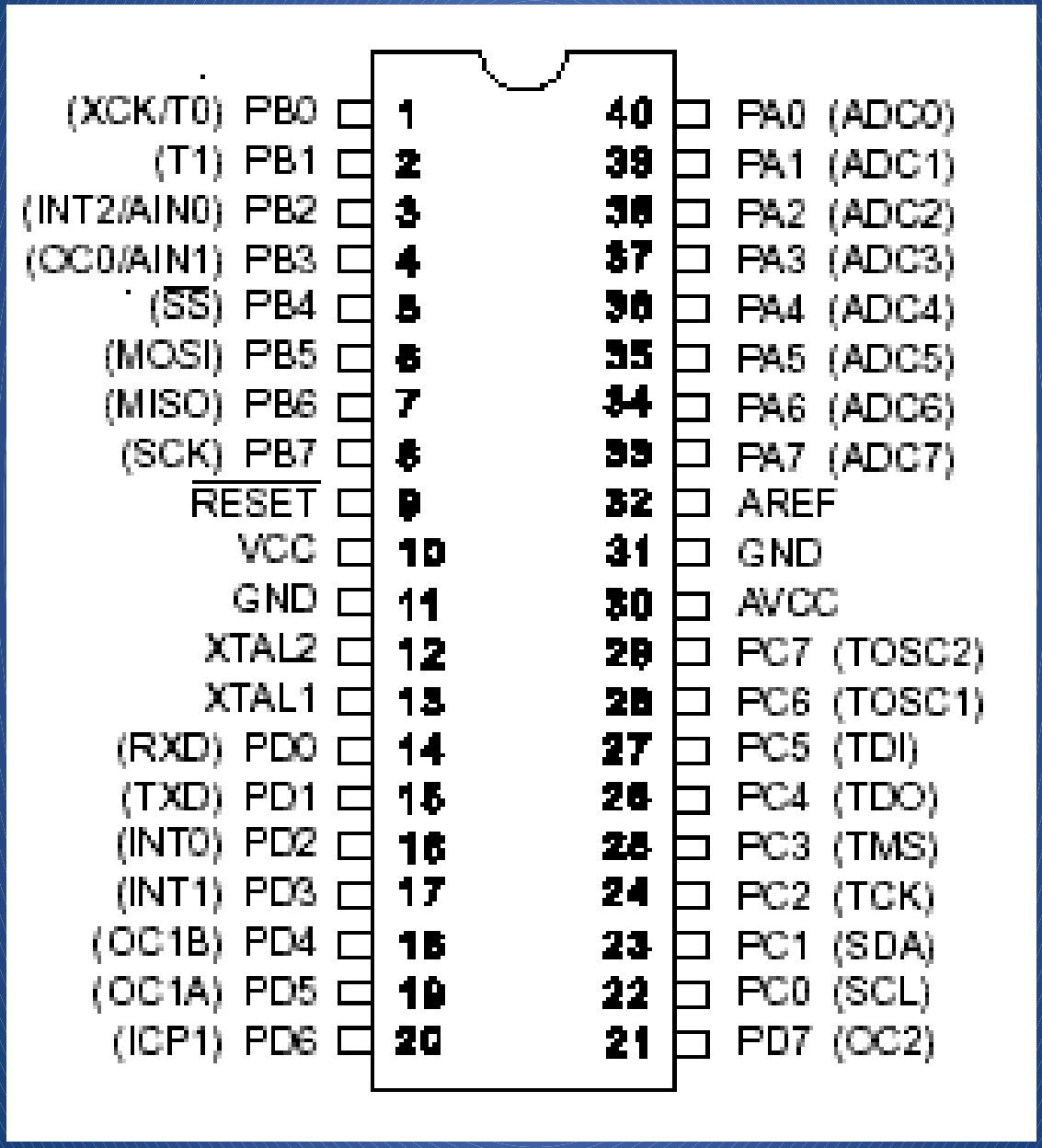
➢ If we want some variables to be stored permanently, then they will be stored here.

- **Ports** : A PORT is a point where data internal to the MCU comes out. EG: PORT A, B,C etc
- **VCC**: Power Supply
- **GND**: Ground

MEMORY

- These pins on MCU can be controlled by a program.
- MCU contains flash memory where it stores this program.
- The flash memory can be erased easily and a new program can be burned.

AT Mega 16


(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OC1B) PD4 18
(OC1A) PD5 19
(ICP1) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)

AVR(AT MEGA 16)

- A PORT is the point where data internal to the MCU comes out. Simply, a combination of pins in an MCU is a PORT.
- A port contains 8 GPIO(General Purpose Input Output pins).
- There are four ports namely A,B,C and D in ATMEGA16.
- Not all pins are GPIO pins, there are some pins (like Vcc, GND, XTAL, etc) which are for other functions and cannot be turned on/off for interfacing

GPIO has two modes:

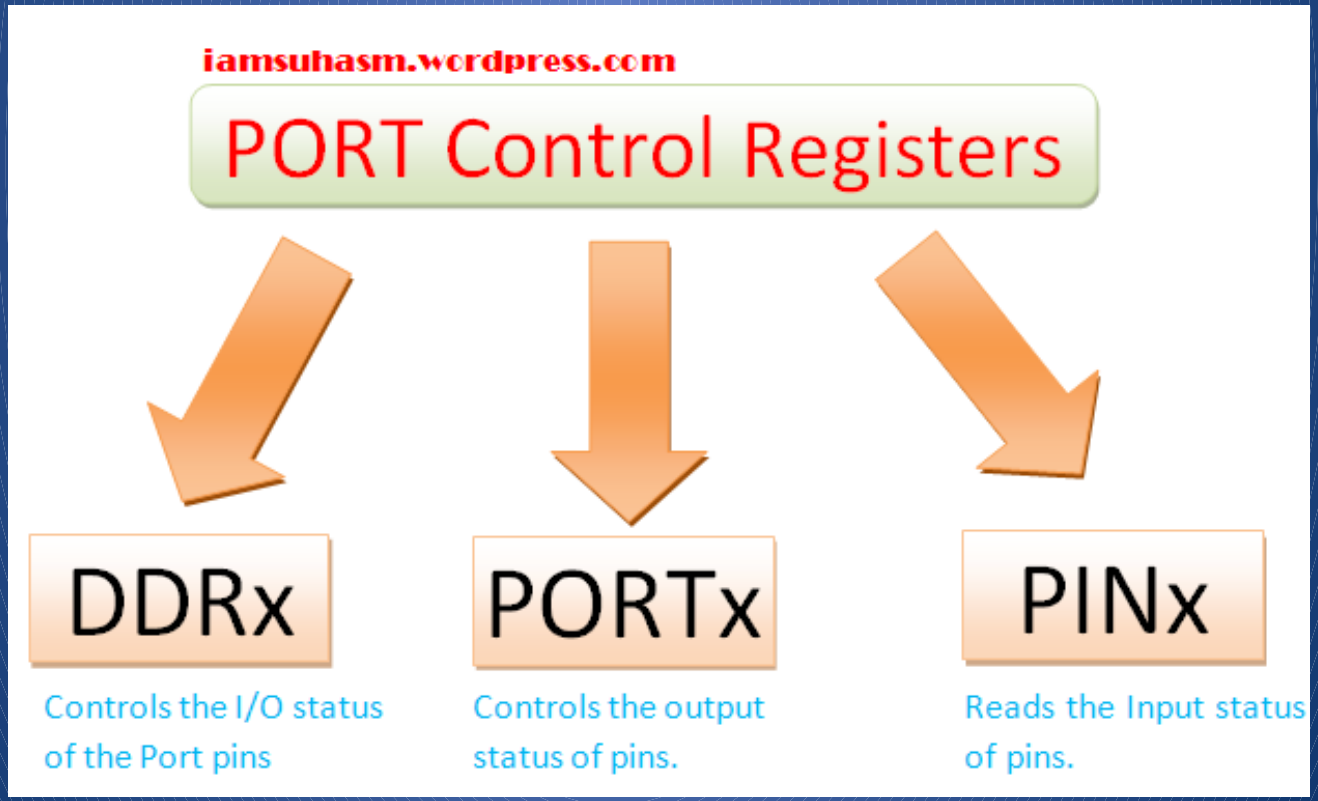
- **Input mode:** In this mode, the MCU can read the values at the pins . They take in the sensor data.
- **Output mode:**

HIGH- Vcc

LOW-Ground

. It is used to send commands to external hardware like servos, LEDs, etc

- Each port in AVR has three related registers.



Internal Peripherals

- In addition to this, each IO pin is defined for an additional purpose like ADC, USART, timers ,etc. These are called *peripherals.*
- The secondary features of these pins become active only if you enable certain bits of some registers

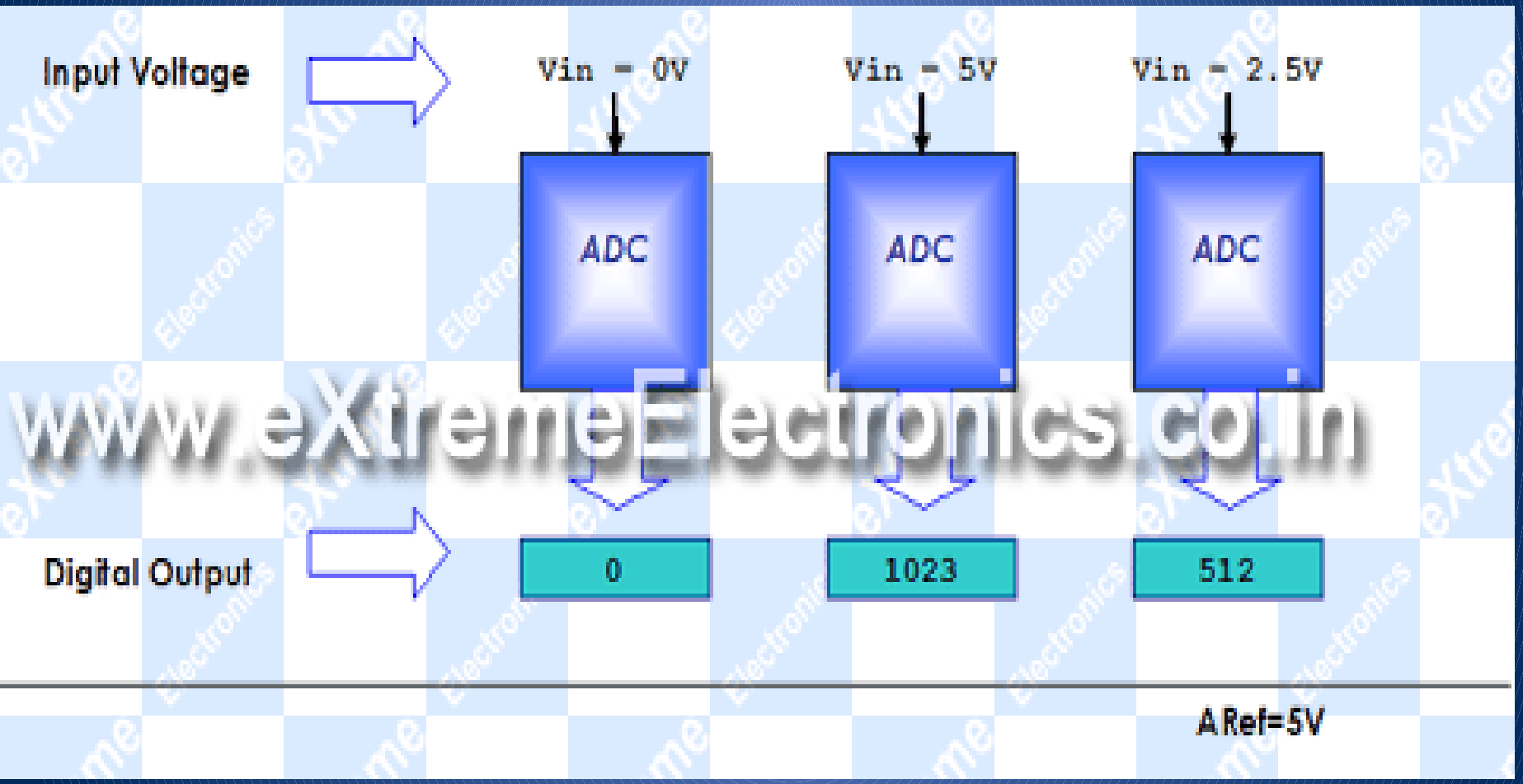
Internal Peripherals

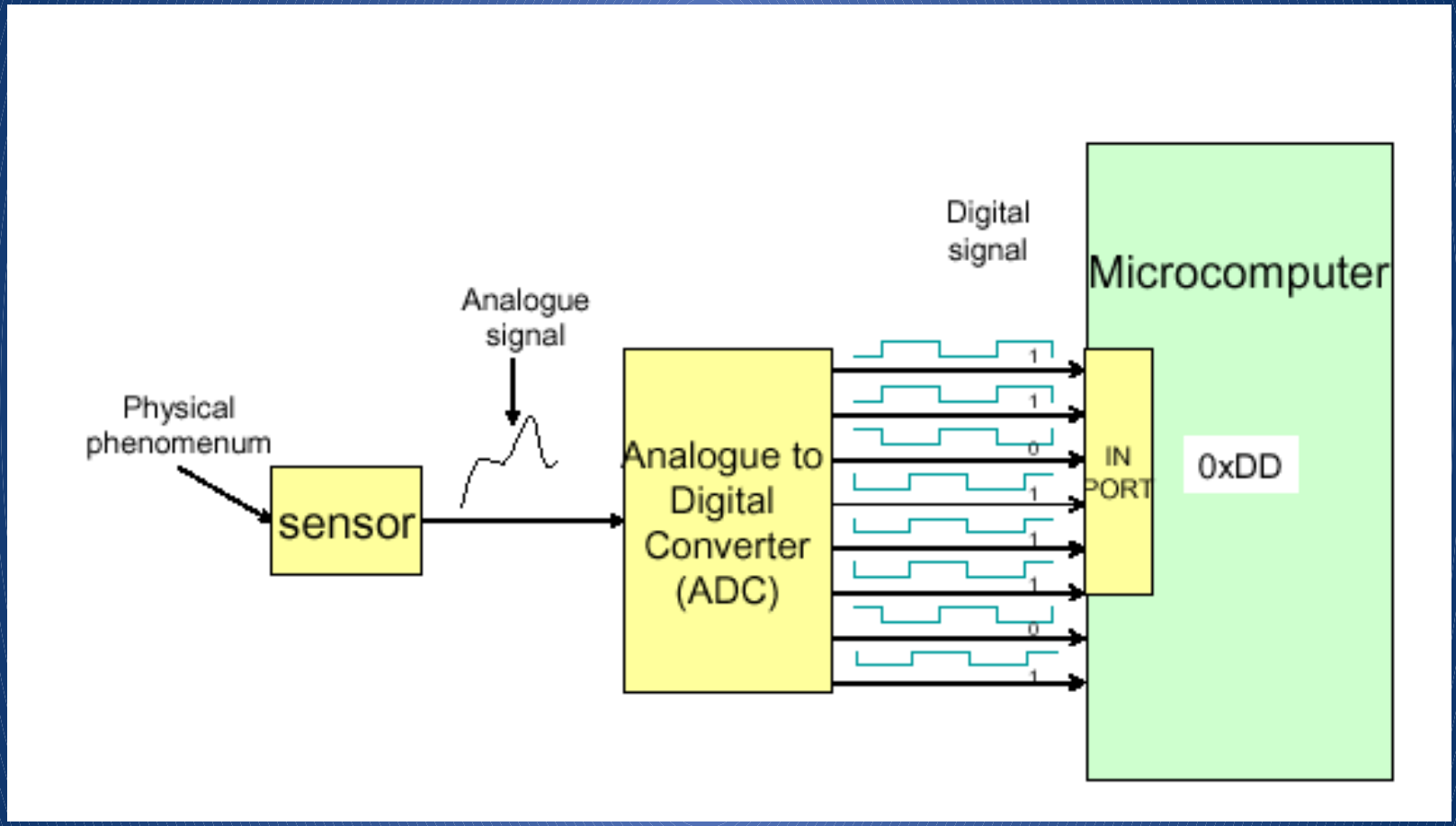
- ADC
- TIMER
- USART
- SPI

Every peripheral has specific registers to provide communication between CPU and that particular peripheral.

ADC

- Analog ports are necessary to connect sensors to our robot.
- ADC ports receive analog signals and convert them into digital number within certain numerical range depending on the *resolution*
- ADC channels are shared with PORTA of AVR.




Digital signal
Microcomputer
Analogue signal
Physical phenomenum
sensor
Analogue to Digital Converter (ADC)
1
1
0
1
1
1
0
1
IN PORT
0xDD

TIMERS

- A timer in general is a register which counts up/down.
- They have a resolution of 8-16 bits.
- A 8 bit timer can count from 0-255.
- They run parallel and independent of CPU and interact with CPU by issuing in

Interrupts

They are of two types.

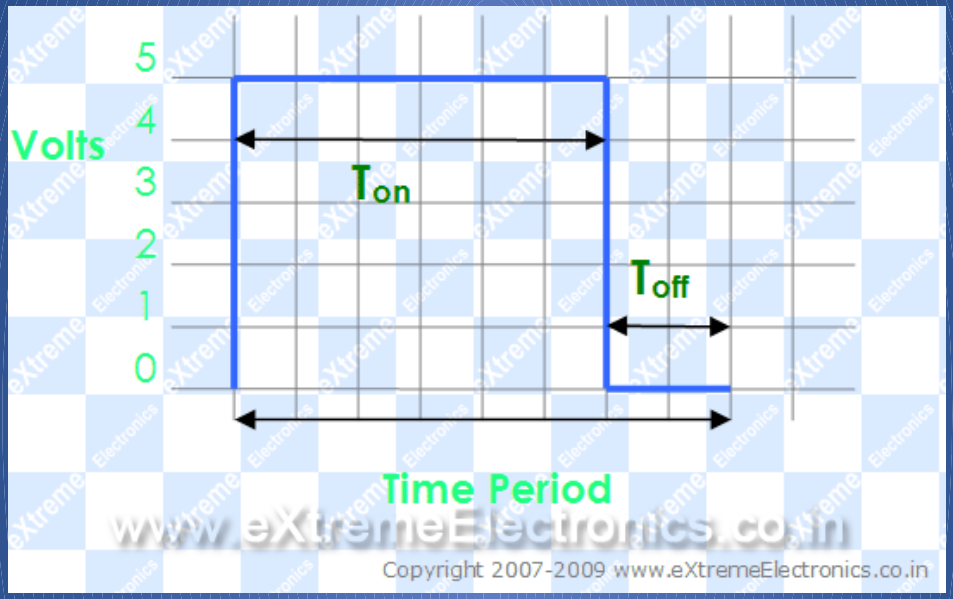
- **Overflow interrupt**: It is called when timer reaches its maximum value.
- **Compare Match interrupt**: It is called when timer reaches a certain value predefined by us.

PWM

- A PWM is used to generate analog outputs depending on value of *duty cycle* instead of regular digital outputs by MCU.
- **Duty Cycle**: It is the percentage of the total time the output is high.

  Duty cycle=($time_{on}$/time period)*100

A PWM waveform



- **Duty Cycle = 75% Analog Voltage Out = 75% of Vcc (5v) = 3.75 Volts**

PWM signal generation

- In AVR MCU, PWM signals are generated by TIMER units.
- There are two methods from which we can generate PWM signals from AVR TIMER0 (for Atmega16 and ATmega32 MCUs).
  1. Fast PWM
  2. Phase Correct PWM

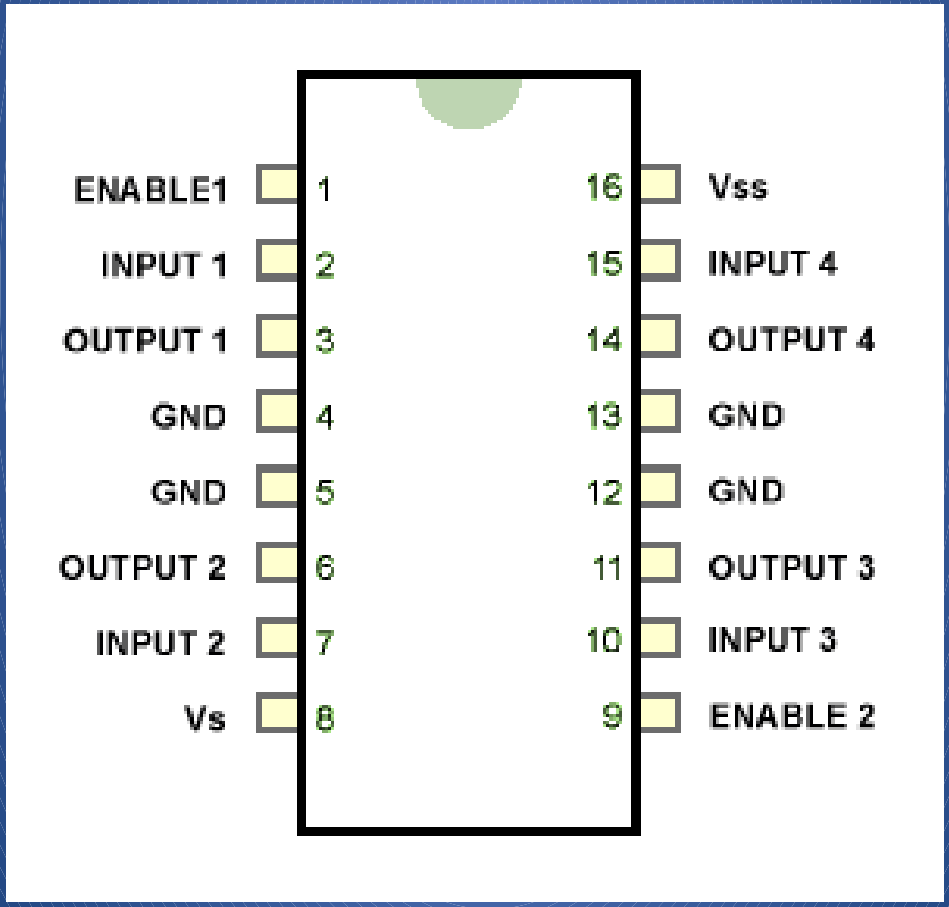
LCD

- We need to interface an LCD to our microcontroller so that we can display messages, outputs, etc.
- Sometimes using an LCD becomes almost inevitable for debugging and calibrating the sensors

Motor Drivers

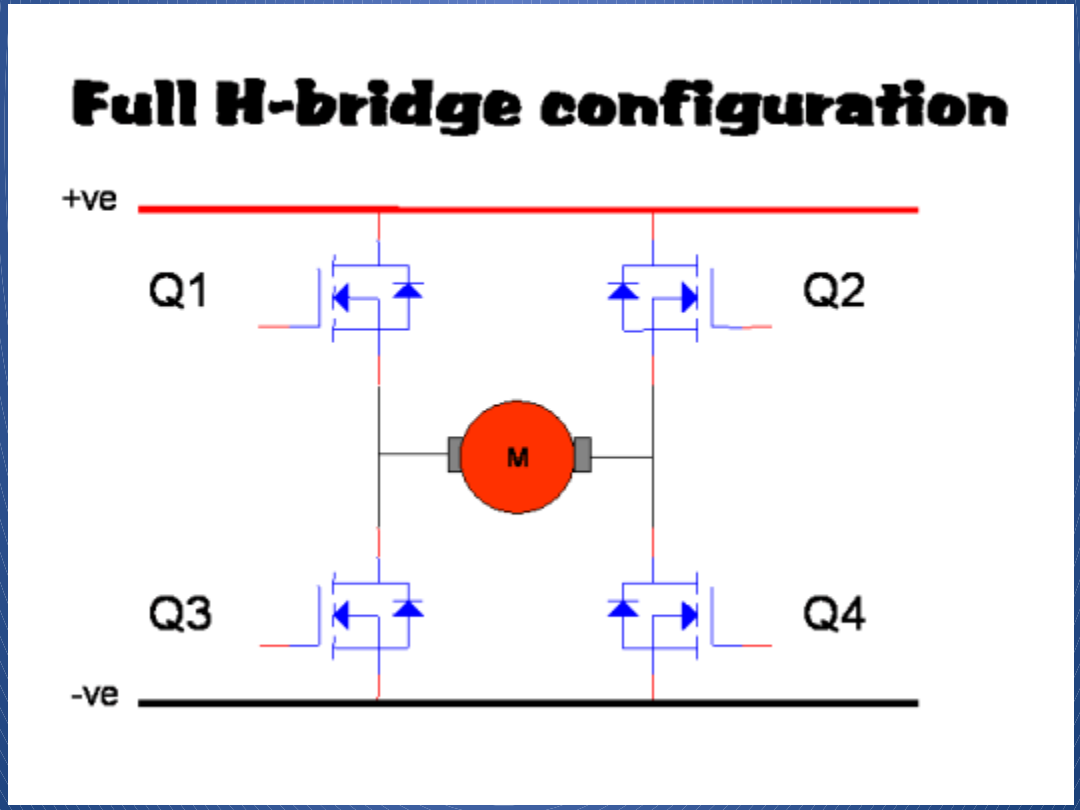
- The current from MCU output pins is not enough to run a motor.
- Therefore, we need an external driver circuit which can act as a bridge between IC and motors.
- We can make a circuit or there are IC available for same purpose. Eg. L293,L293D,etc

H bridge using L293D



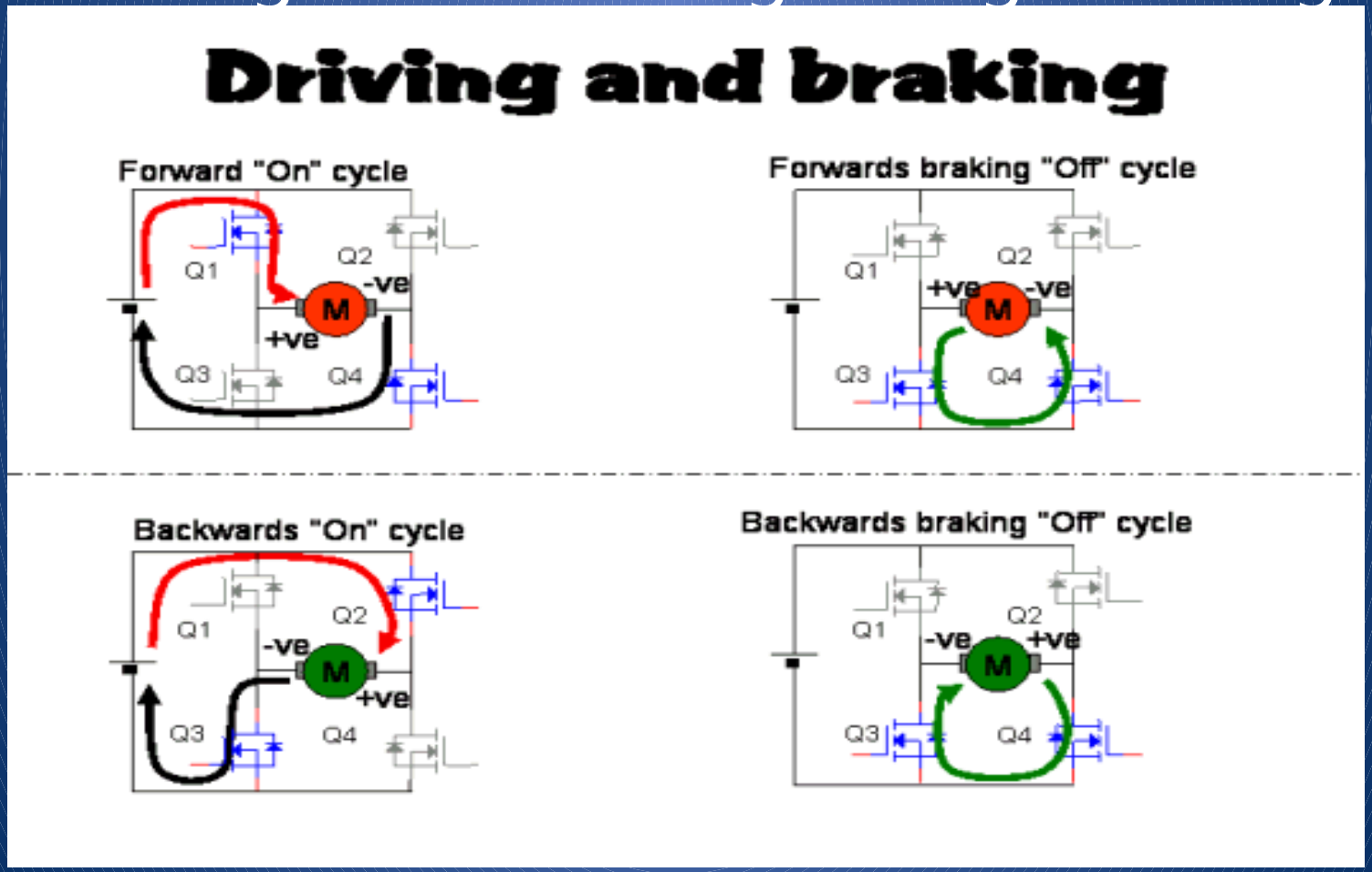
H Bridge

It’s a circuit to allow the rotation of motor in both direction



Working of DC motor

- Driving and Braking using H Bridge



Working of DC motor

- H-Bridge in short

S1	S2	S3	S4	Current Direction	Effect
1	0	0	1	1 to 2	Motor spins forward
0	1	1	0	2 to 1	Motor spins backward
1	1	0	0	-	Braking Occurs
0	0	0	0	-	Free running

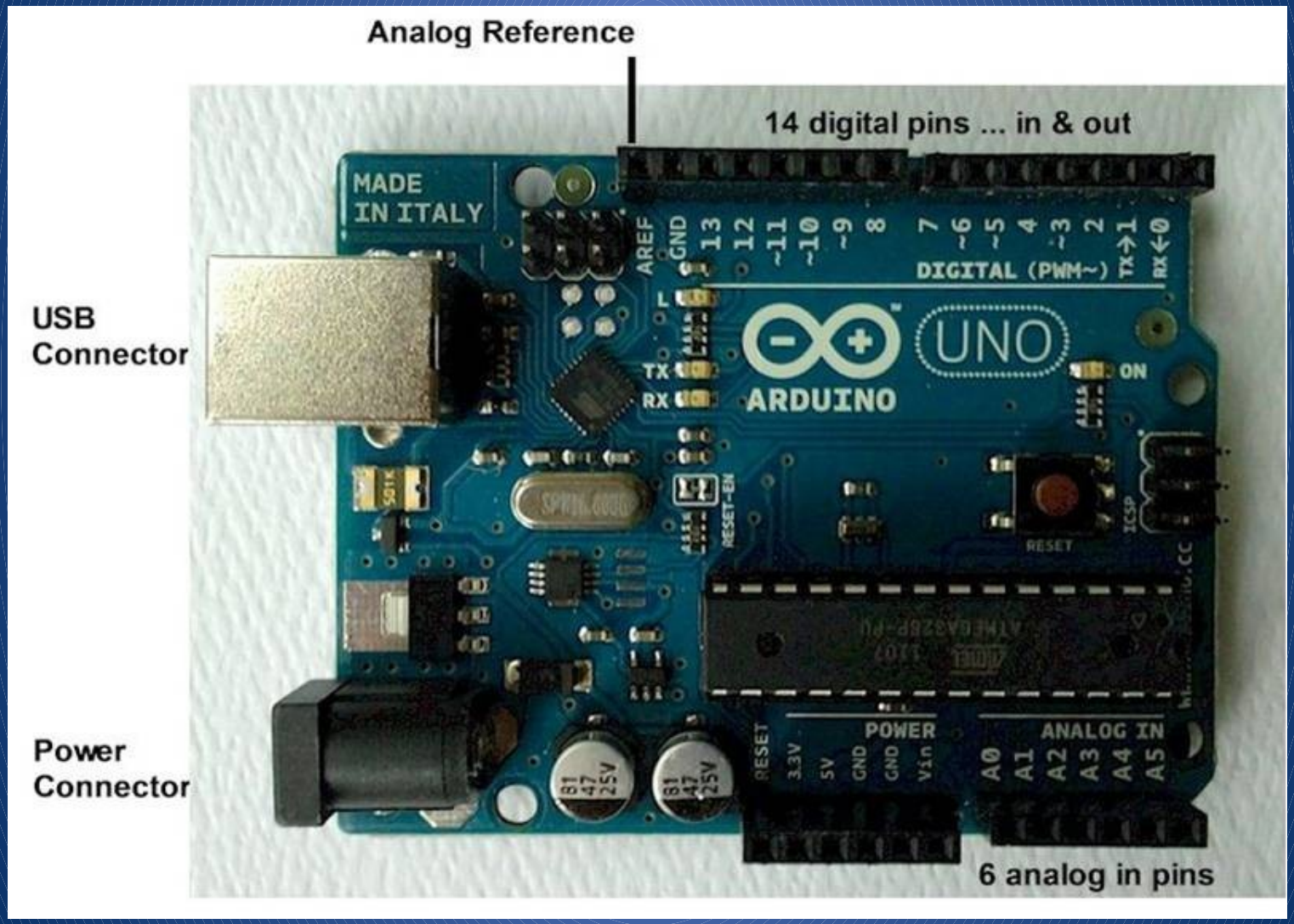
Development Boards

- Here we are going to discuss about 2 types of development boards in particular.
- 1. Arduino
- 2. AVR

ARDUINO

- Arduino is an AVR-based prototyping board with an emphasis on ease of use.
- It has separate pins for digital and analog purposes.
- There are 6 *analog in pins* and 14 *digital in/out pins* ground, Vcc …
- It consists of PWM output pins seperately.
- It also features a USB interface allowing serial communication through a USB device, eliminating the need for a separate AVR programmer.

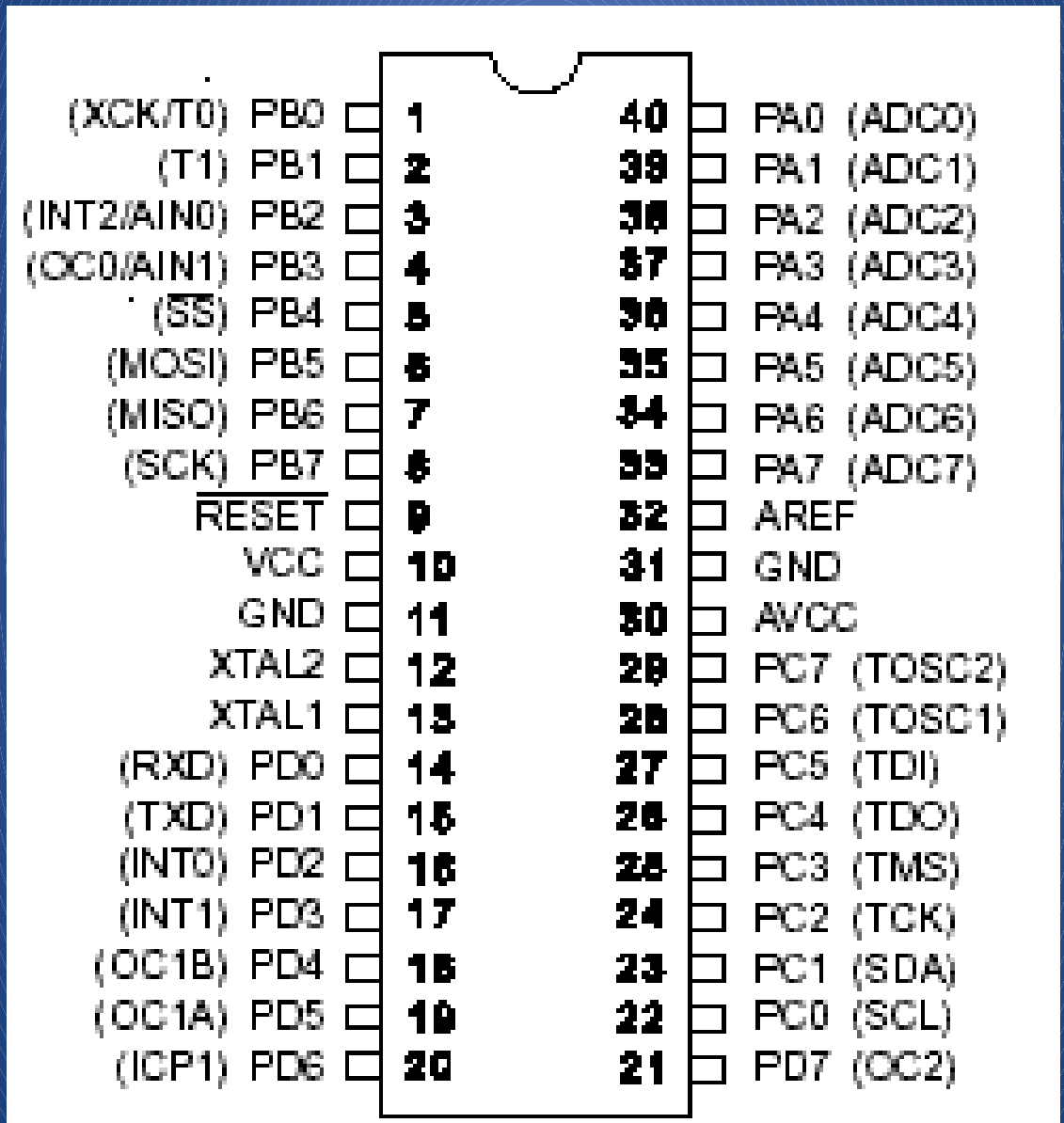
ARDUINO


Analog Reference
14 digital pins ... in & out
USB Connector
Power Connector
6 analog in pins
MADE IN ITALY
DIGITAL (PWM~)
UNO
ARDUINO
POWER
ANALOG IN
RESET

Why ARDUINO??

- The main reason is its programming environment.
- IDE includes many helpful libraries .
- While programming Arduino, you are least bothered with the internal hardware and registers of the microcontroller. All you do is call the functions written in the libraries (which are already provided)
- You can use Arduino if you want to prototype your project very fast, and are not much concerned about the programming part.

AT MEGA 16


(XCK/T0) PB0 1
(T1) PB1 2
(INT2/AIN0) PB2 3
(OC0/AIN1) PB3 4
(SS) PB4 5
(MOSI) PB5 6
(MISO) PB6 7
(SCK) PB7 8
RESET 9
VCC 10
GND 11
XTAL2 12
XTAL1 13
(RXD) PD0 14
(TXD) PD1 15
(INT0) PD2 16
(INT1) PD3 17
(OC1B) PD4 18
(OC1A) PD5 19
(ICP1) PD6 20
40 PA0 (ADC0)
39 PA1 (ADC1)
38 PA2 (ADC2)
37 PA3 (ADC3)
36 PA4 (ADC4)
35 PA5 (ADC5)
34 PA6 (ADC6)
33 PA7 (ADC7)
32 AREF
31 GND
30 AVCC
29 PC7 (TOSC2)
28 PC6 (TOSC1)
27 PC5 (TDI)
26 PC4 (TDO)
25 PC3 (TMS)
24 PC2 (TCK)
23 PC1 (SDA)
22 PC0 (SCL)
21 PD7 (OC2)

AVR

- In this, communication between PC and MCU is done using a programmer.
- It consists of a on-board driver.

QUERIES??