Robotics Club IIT Kanpur

TAKNEEK LECTURE SE

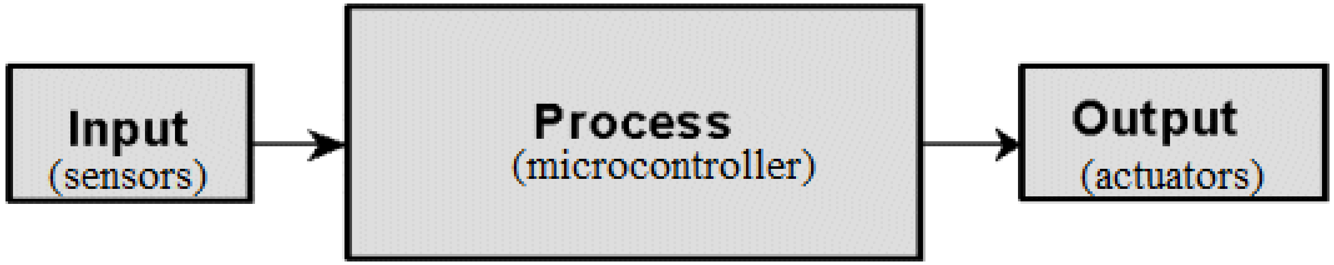
The 3 Schools of Robotics:

- **Mechanical Design**
- Types of motors
- Material selection
- Driving, gripping, clamping, climbing, lifting…

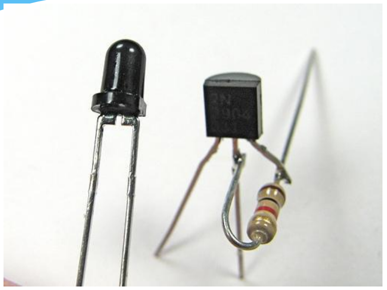
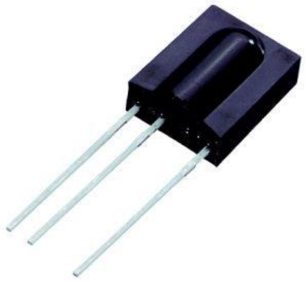
- **Electronics**
- Microcontrollers
- Motor Drivers
- Sensors (Infrared, Ultrasonic, etc)

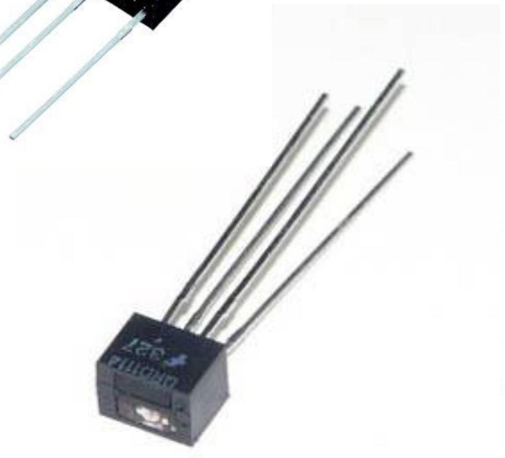
- **Programming**
- Microcontroller Programming
- Communicating with a computer.
- Communicating between multiple robots.

A simple autonomous robot



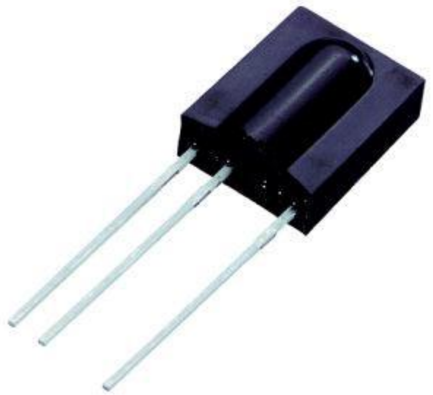
SENSORS:



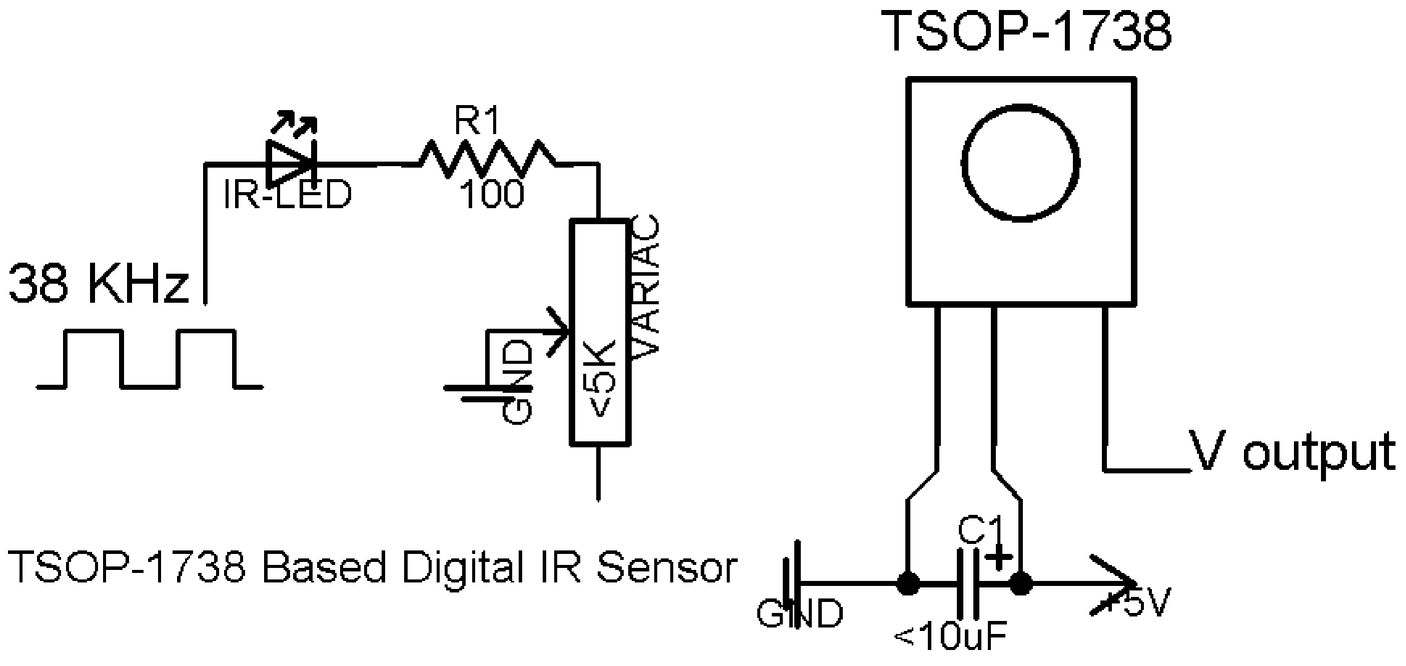
A **sensor** is a device that measures a physical quantity and converts it into a signal which can be read by microcontroller. (heat, motion, gyro, sonar, IR, etc…)

TSOP 1738:

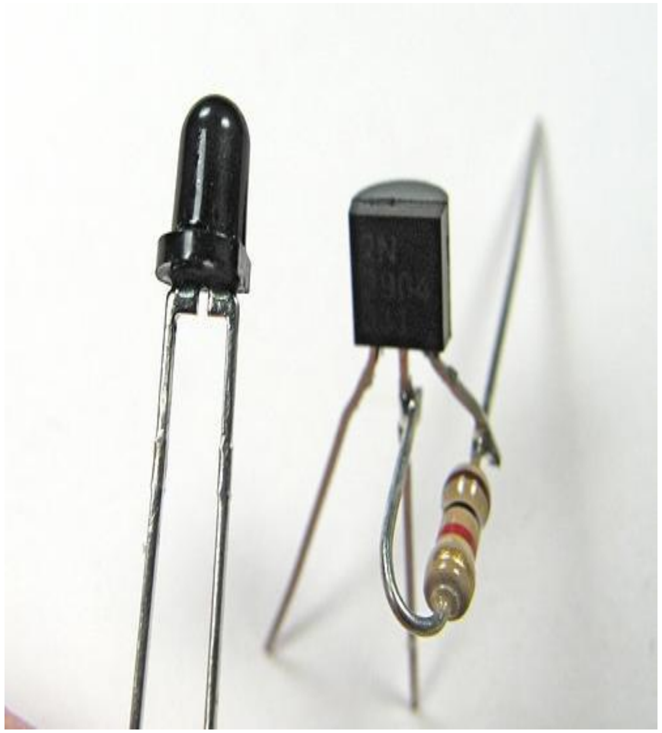


- It is a **DIGITAL IR SENSOR**, giving outputs of 0V and 5V.
- It does not respond to any stray IR, it only responds to IR falling on it at a pulse rate of 38 KHz (given by TIMER of microcontroller)
- Can accurately distinguish between 2 colors.

Circuit Diagram:

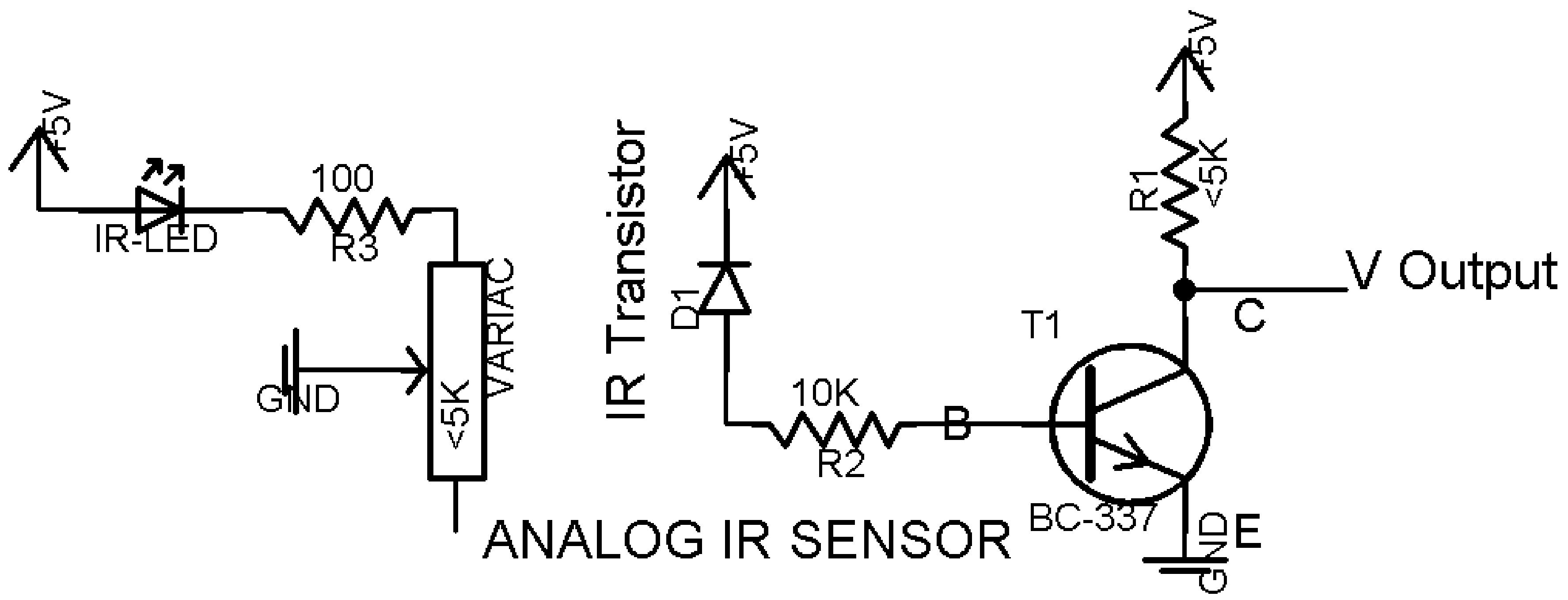


IR Analog Sensor:



- It gives output in the range of 0V to 5V, depending on the amount of IR light falling on the photodiode. (read by ADC of microcontroller)
- Hence the name analog…

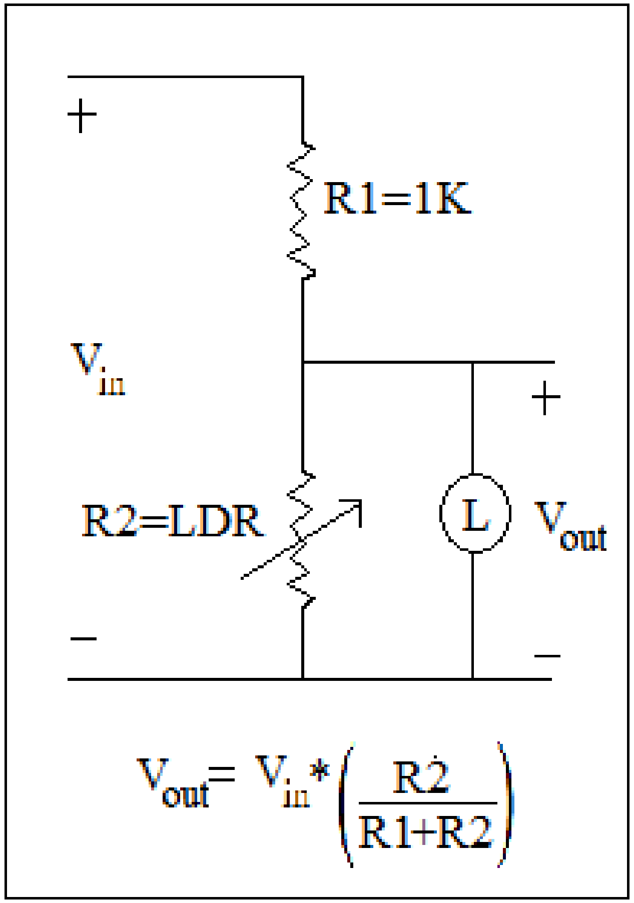
Circuit Diagram:



QRD1114:

* It is an easy-to-use, simple analog sensor (Output range: 0V to 5V)
* Has a very small range: 0.5mm to 1cm
* Used to distinguish between bright and dark colours

LDR (Light Dependent Resistor)



$$V_{out} = V_{in} * \left(\frac{R2}{R1+R2} \right)$$

•made of calcium sulfate…

•When no light falls on it, the LDR behaves as an open circuit (very high resistance: 10^6 ohms)

•As the intensity of light increases, the resistance drops (to about 10 to 100 ohms)

MICRO-CONTROLLER:

A microcontroller is the brain of the robot. They are:

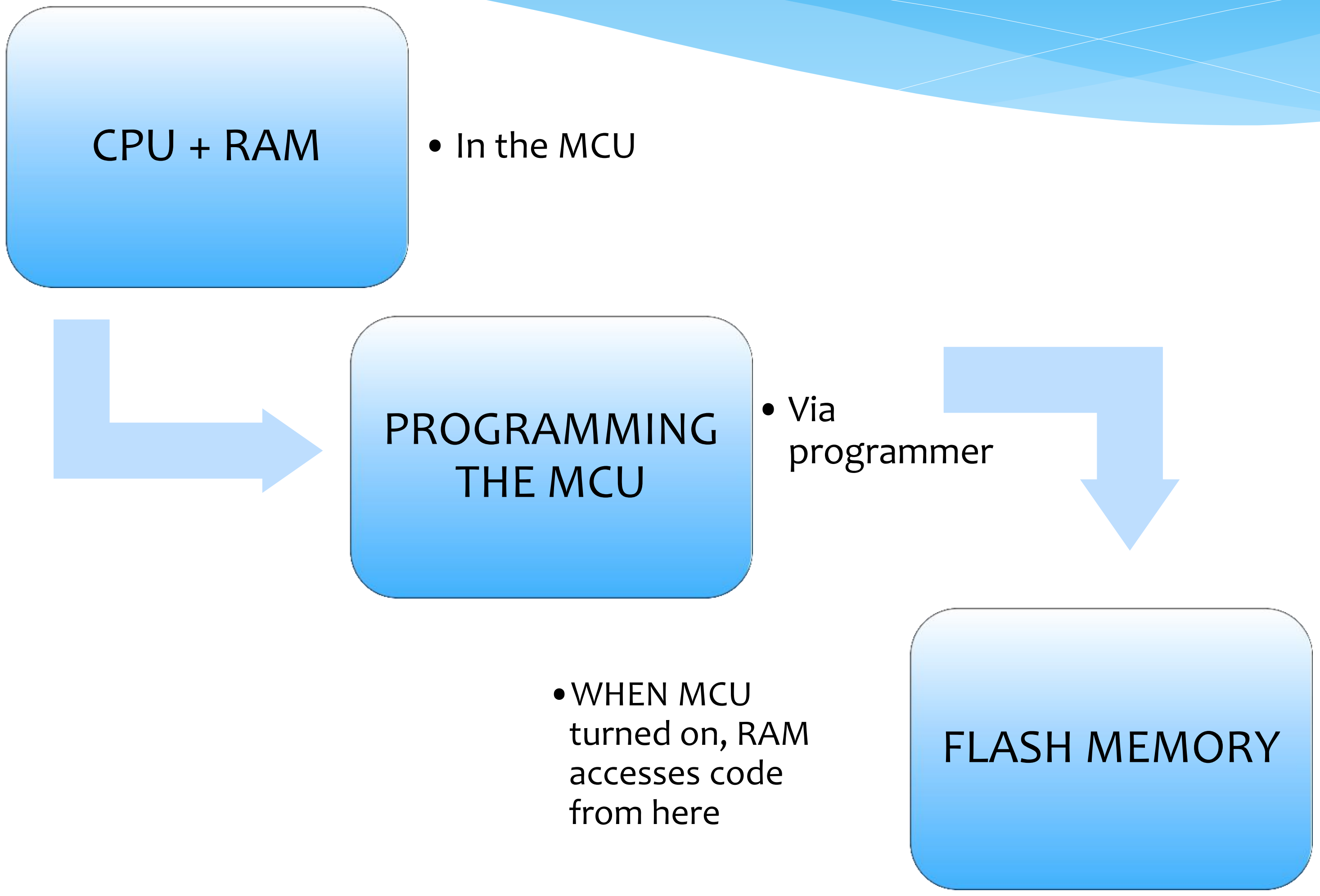
1. Integrated Circuits (ICs)
2. Programmable – You can define what they do.
   - This is what makes them different from other simpler ICs, such as AND gates, inverters, etc
   - Logic tables relating Input and Output can be defined by you
3. Many useful features built in to the uC itself

Kinds of Microcontrollers:

- AVR
- PIC
- Intel 8051
- Rabbit
- Zilog
- Many, Many more

We will be using AVR microcontroller, of Atmel's ATMEGA family… Mostly, we will use Atmega 16…

How a UC works?



- In the MCU
- Via programmer
- WHEN MCU turned on, RAM accesses code from here

How to program a microcontroller:

- A microcontroller understands only binary language (0 and 1)… We write the code in C language in **CVAVR** (Compiler + IDE).
- We compile the code to generate the .hex file (which the machine understands)
- We transfer the .hex file to Atmega using a Programmer

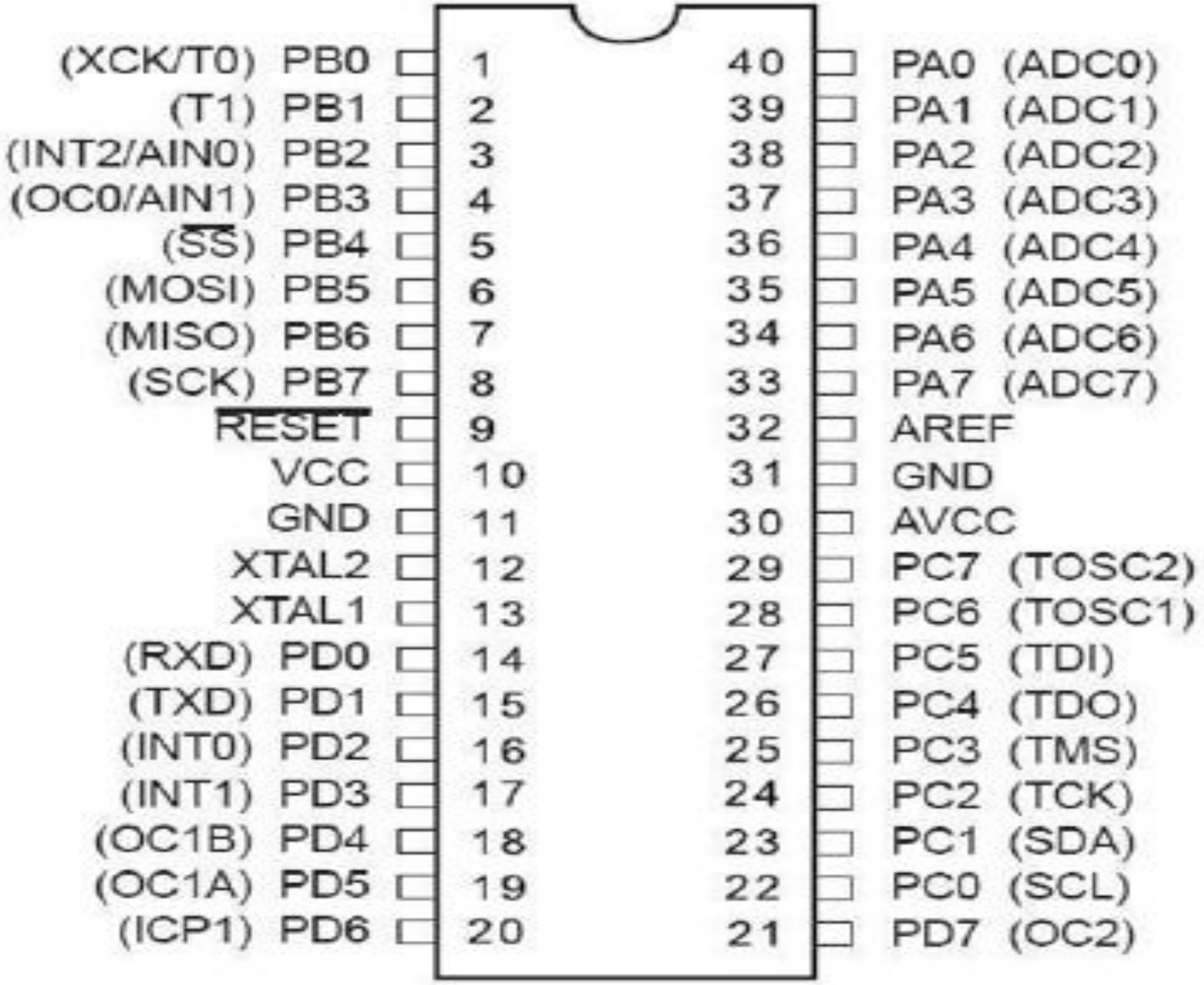
Hardware: Serial Programmer, STK 500, USB etc… (Contains chips to convert voltage levels and protocols)

Software: AVR Studio, AVRDUDE, etc…

ATMEGA 16

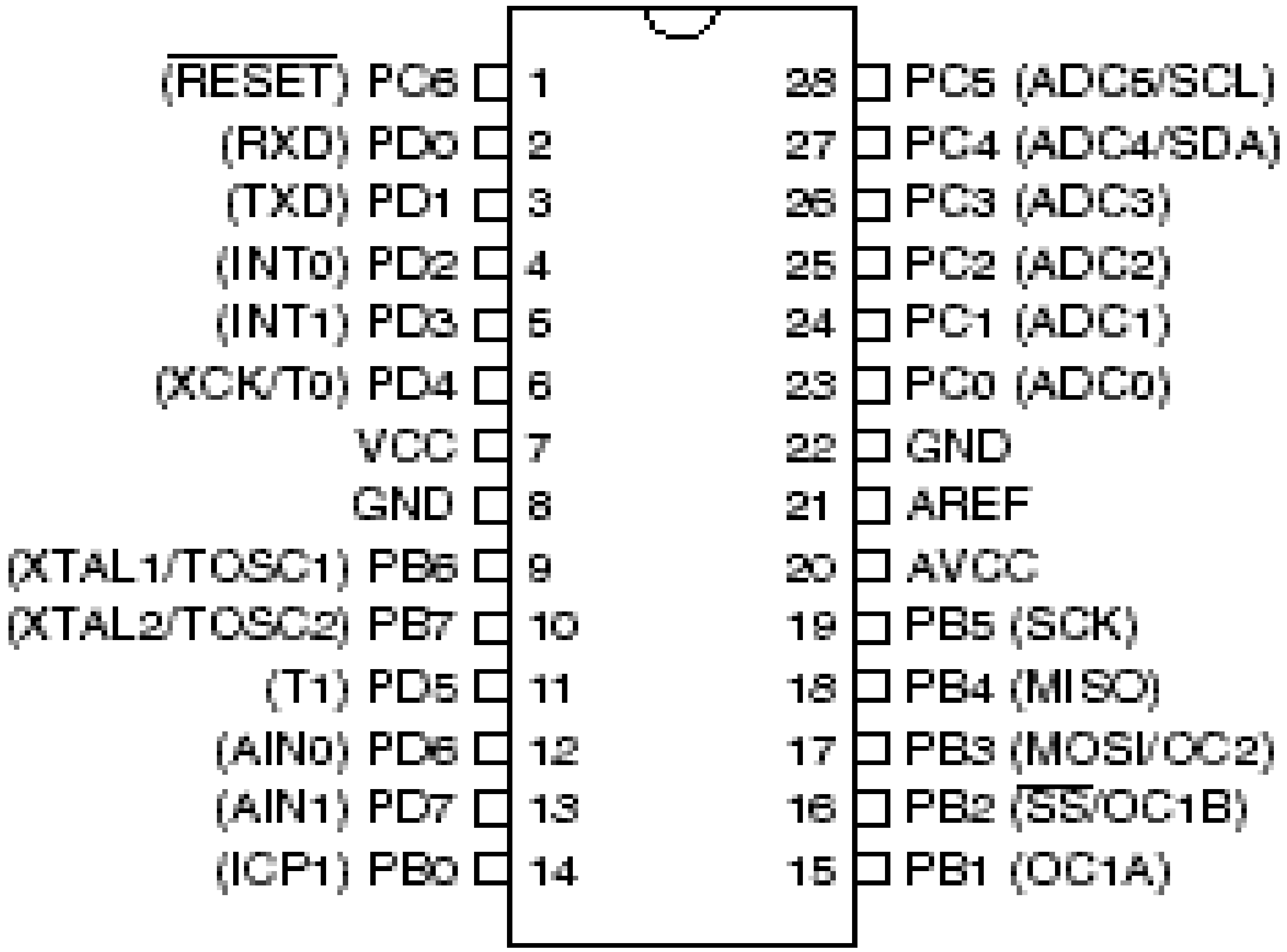
- 16 KB Flash Memory…
- 40 pin $\mathcal{U}$C…
- 32 I/O Pins
- These pins have other functions like UART, ADC, SPI, LCD, Interrupt, TIMERS, etc…
- Maximum Voltage 5V, current rating about 10 milli-amperes

Pin Configuration

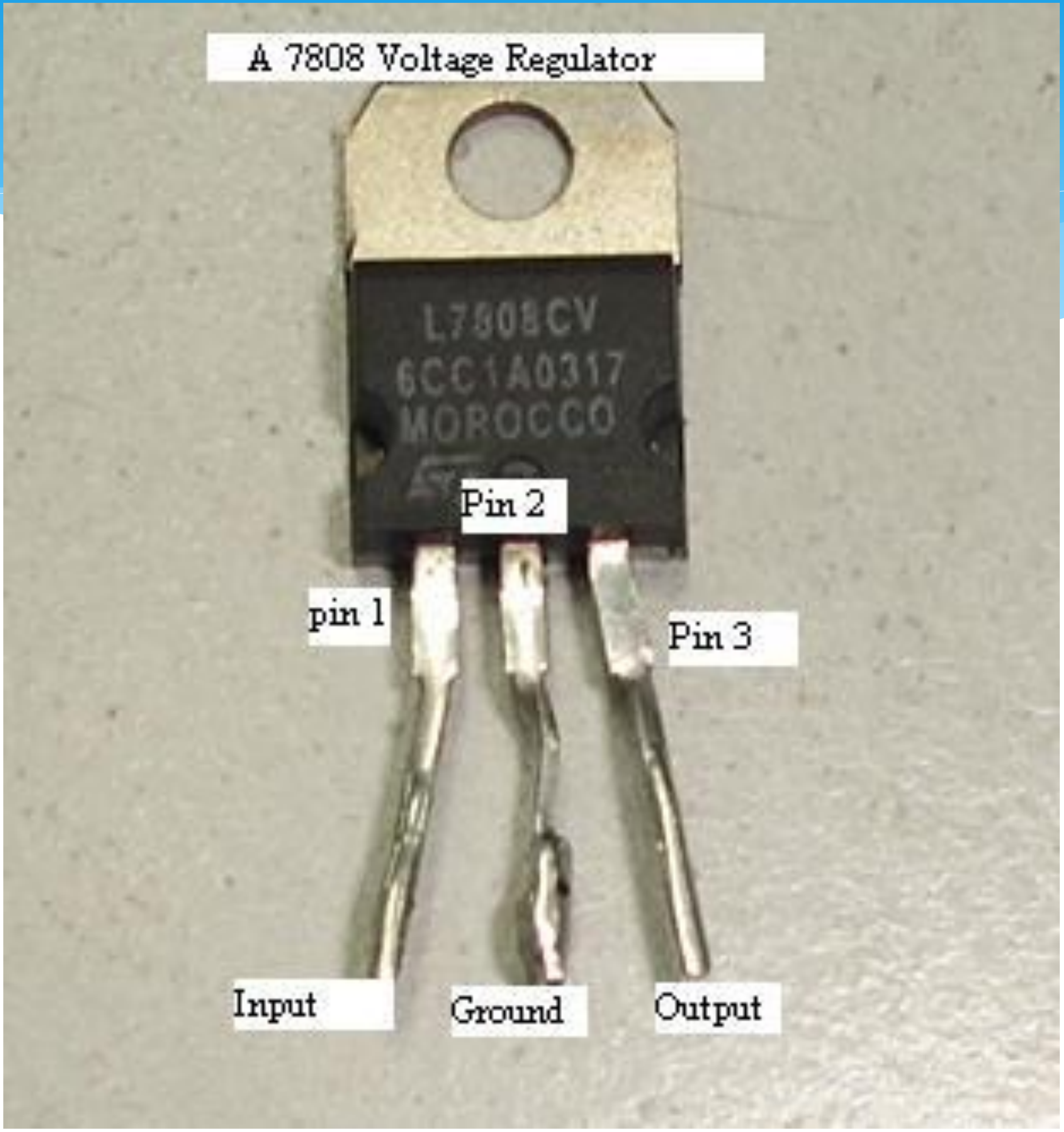


ATMEGA 8

- 8 KB Flash Memory…
- 28 pin UC…
- 32 I/O Pins
- These pins have other functions like UART, ADC, SPI, LCD, Interrupt, TIMERS, etc…
- But we will now focus on I/O and TIMER…
- Maximum Voltage 5V, current rating about 10 milli-amperes


(RESET) PC6 1
(RXD) PD0 2
(TXD) PD1 3
(INT0) PD2 4
(INT1) PD3 5
(XCK/T0) PD4 6
VCC 7
GND 8
(XTAL1/TOSC1) PB6 9
(XTAL2/TOSC2) PB7 10
(T1) PD5 11
(AIN0) PD6 12
(AIN1) PD7 13
(ICP1) PB0 14
28 PC5 (ADC5/SCL)
27 PC4 (ADC4/SDA)
26 PC3 (ADC3)
25 PC2 (ADC2)
24 PC1 (ADC1)
23 PC0 (ADC0)
22 GND
21 AREF
20 AVCC
19 PB5 (SCK)
18 PB4 (MISO)
17 PB3 (MOSI/OC2)
16 PB2 (SS/OC1B)
15 PB1 (OC1A)

* As Atmega requires 5V: To convert 12V (input) to 5V (output) we use the 7805 Voltage Regulator:
* Current Rating:0.5A



I/O Pins:

- Setting Data Direction (I/O?): DDRX Register
- PORTX Register
- PINX Register

6.1 DDRX (Data Direction Register)

First of all we need to set whether we want a pin to act as output or input. DDRX register sets this. Every bit corresponds to one pin of PORTX. Let's have a look on DDRA register.

Bit	7	6	5	4	3	2	1	0
PIN	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

Now to make a pin act as I/O we set its corresponding bit in its DDR register.

- To make Input set bit 0
- To make Output set bit 1

If I write **DDRA = 0xFF** (0x for Hexadecimal number system) that is setting all the bits of DDRA to be 1, will make all the pins of PORTA as Output.

Similarly by writing **DDRD = 0x00** that is setting all the bits of DDRD to be 0, will make all the pins of PORTD as Input.

Now let's take another example. Consider I want to set the pins of PORTB as shown in table,

PORT-B	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
Function	Output	Output	Input	Output	Input	Input	Input	Output
DDRB	1	1	0	1	0	0	0	1

For this configuration we have to set DDRB as **11010001** which in hexadecimal is **D1**. So we will write **DDRB=0xD1**

Summary

- DDRX -----> to set PORTX as input/output with a byte.
- DDRX.y ---> to set yth pin of PORTX as input/output with a bit (works only with CVAVR).

6.2 PORTX (PORTX Data Register)

This register sets the value to the corresponding PORT. Now a pin can be Output or Input. So let's discuss both the cases.

6.2.1 Output Pin

If a pin is set to be output, then by setting bit 1 we make output **High** that is +5V and by setting bit 0 we make output **Low** that is 0V.

Let's take an example. Consider I have set DDRA=0xFF, that is all the pins to be Output. Now I want to set Outputs as shown in table,

PORT-A	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
Value	High(+5V)	High(+5V)	Low(0V)	Low(0V)	Low(0V)	High(+5V)	High(+5V)	Low(0V)
PORTA	1	1	0	0	0	1	1	0

For this configuration we have to set **PORTA** as **11000110** which in hexadecimal is **C6**. So we will write **PORTA=0xC6;**

6.2.2 Input Pin

If a pin is set to be input, then by setting its corresponding bit in PORTX register will make it as follows,

- Set bit 0 ---> Tri-Stated
- Set bit 1 ---> Pull Up

Tristated means the input will *hang* (no specific value) if no input voltage is specified on that pin.

Pull Up means input will go to +5V if no input voltage is given on that pin. It is basically connecting PIN to +5V through a 10K Ohm resistance.

Summary

- PORTX ----> to set value of PORTX with a byte.
- PORTX.y --> to set value of yth pin of PORTX with a bit (works only with CVAVR).

6.3 PINX (Data Read Register)

This register is used to read the value of a PORT. If a pin is set as input then corresponding bit on PIN register is,

- 0 for Low Input that is $V < 2.5V$
- 1 for High Input that is $V > 2.5V$ (Ideally, but actually 0.8 V - 2.8 V is error zone !)

For an example consider I have connected a sensor on PC4 and configured it as an input pin through DDR register. Now I want to read the value of PC4 whether it is Low or High. So I will just check 4th bit of PINC register.

We can only read bits of the PINX register; can never write on that as it is meant for reading the value of PORT.

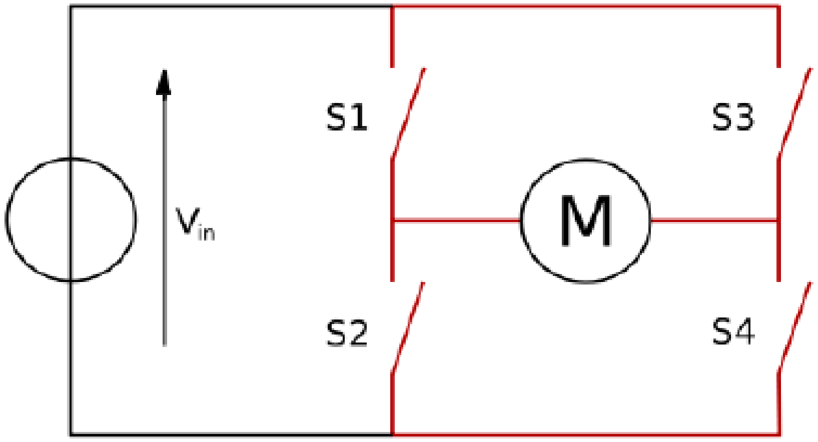
Summary

- PINX ----> Read complete value of PORTX as a byte.
- PINX.y --> Read yth pin of PORTX as a bit (works only with CVAVR).

How to Run Motors through UC?

- The Atmega16 has a current rating of 5-10 mA.
- The normal DC motor's current ratings start from 150 mA and above.
- So the motor cannot be directly attached to the Atmega, hence a **motor-driver** is used… (eg: L293, L298, etc…)

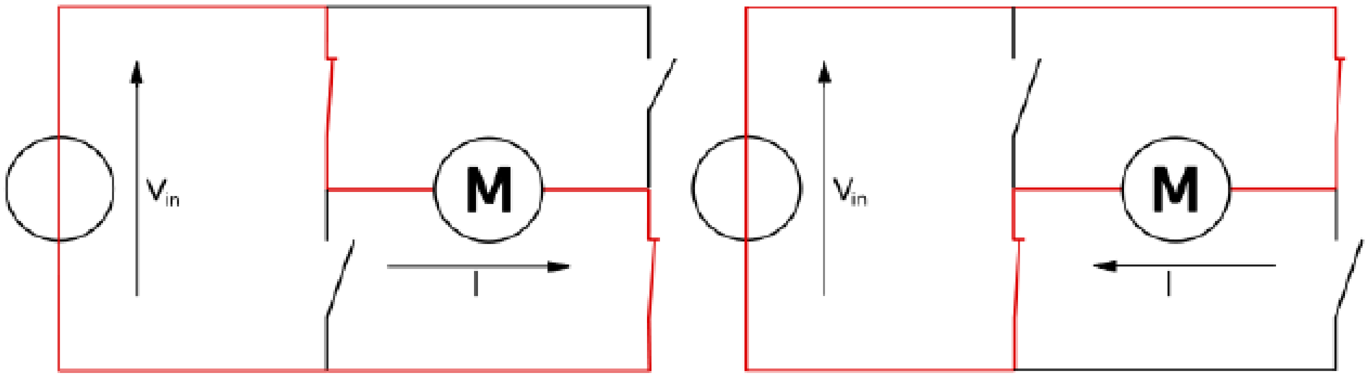
MOTOR DRIVERS: H -Bridge



Structure of an H-bridge (highlighted in red)

S1	S2	S3	S4	Result
1	0	0	1	Motor rotates in one direction
0	1	1	0	Motor rotates in opposite direction
0	0	0	0	Motor free runs (coasts)
0	1	0	1	Motor brakes
1	0	1	0	Motor brakes

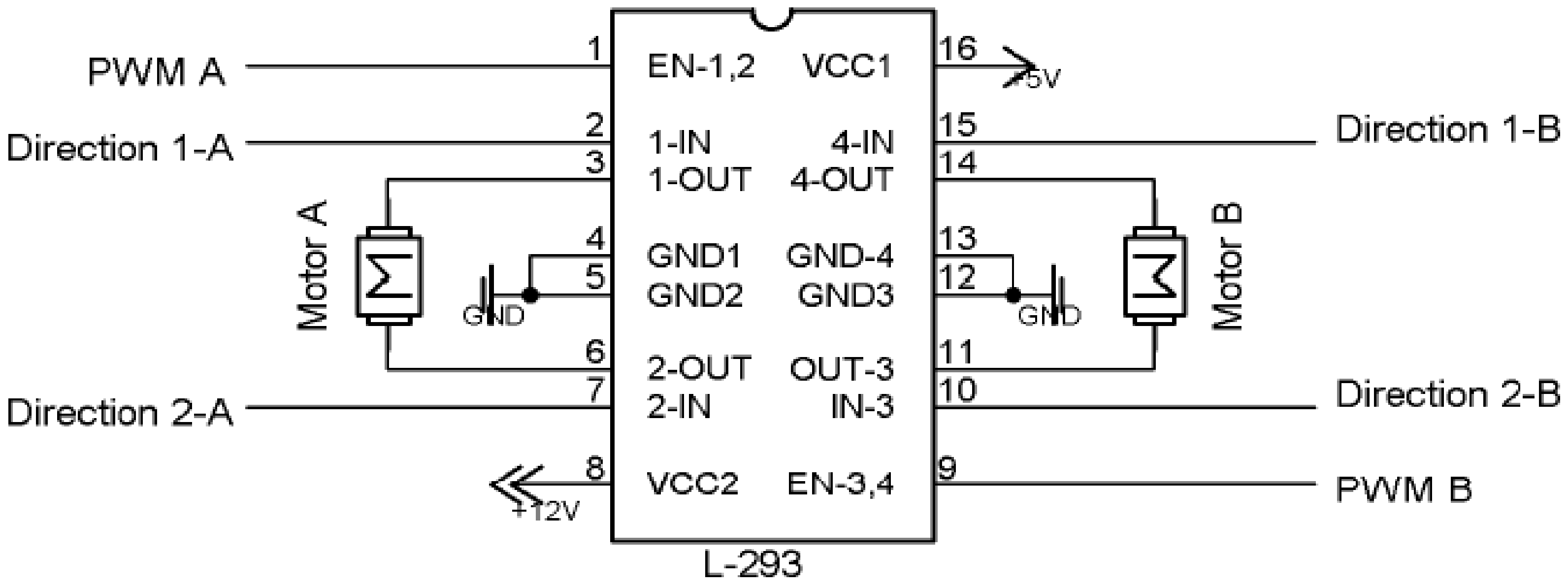
- To power the motor, you turn on two switches that are diagonally opposed.



The two basic states of an H-bridge.

L293 Motor Driver:

- Has 2 H-Bridge drivers, hence can drive 2 motors. (It does not amplify current, it only acts as a switch)



•PWM from Atmega's TIMER pin controls the speed of the motor…

TIMERS:

- Atmega has:
  - Timer 0, 8 bit (OC0 at PB3)
  - Timer 1, 16 bit (2 parts A and B each of 8 bit, OC1A at PD5, OC1B at PD4)
  - Timer 2, 8 bit (OC2 at PD7)

- There are 2 clocks:
  - System Clock (fs)
  - Timer Clock (ft = fs or fs/8 or fs/64 or ….)

There are total 4 settings for Timers,

1. **Clock Source**
2. **Clock value**
3. **Mode:** There are many modes of timers. We will be discussing following 2 modes,
   a. Fast PWM top = FFh
   b. CTC top=OCRx (x=0, 1A, 2)
4. **Output**

Basically each Timer has a counter unit with size 8 bit for **Timer 0, 2** and 16 bit for **Timer 1.**

Each counter has a register which increments by one on every rising edge of timer clock. After counting to its full capacity, 255 for 8 bit, it again starts from 0. By using this register we can have different modes.

- Timer/Counter (**TCNT0**) and Output Compare Register (**OCR0**) are 8-bit registers.

- **TOP:** The counter reaches the TOP when it becomes equal to the highest value in the count sequence. The TOP value can be assigned to be the fixed value 0xFF (MAX) or the value stored in the OCR0 Register. The assignment is dependent on the mode of operation.

9.2 Fast PWM Mode (For MOTORS)

PWM = Pulse Width Modulation.

This mode is used to generate pulse with

- Fixed Frequency (F)
- Variable Duty Cycle (D)

F = Ft / 256

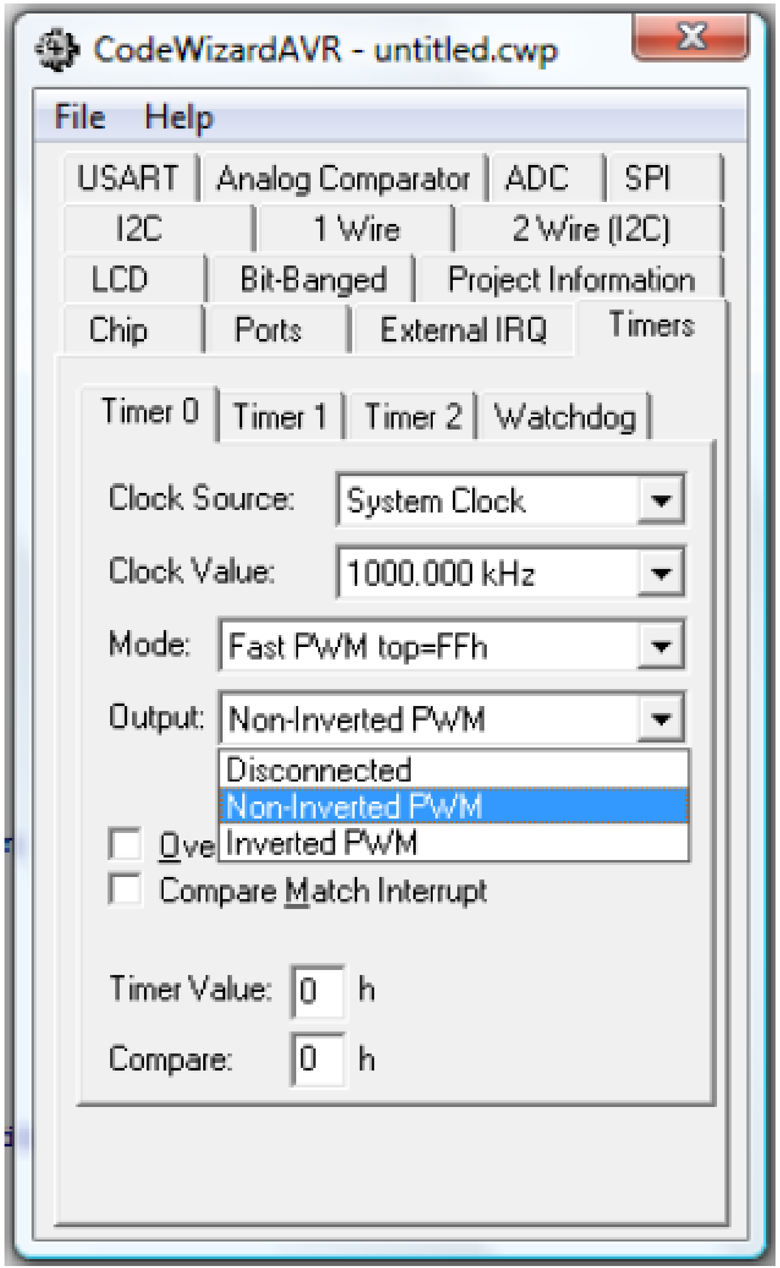
D = OCR0 / 255 (non inverted)

D = (255-OCR0) / 255 (inverted)

By changing OCR0 value we can change the duty cycle of the output pulse.

As OCR0 is an 8bit register it can vary from 0 o 255.

$0 \le OCR0 \le 255$



Clock Source: System Clock
Clock Value: 1000.000 kHz
Mode: Fast PWM top=FFh
Output: Non-Inverted PWM
Ove
Compare Match Interrupt
Timer Value: 0 h
Compare: 0 h

9.3 CTC Mode (For generating 38 KHz in TSOP Sensors)

CTC = Clear Timer on Compare Match.

This mode is to generate pulse with,

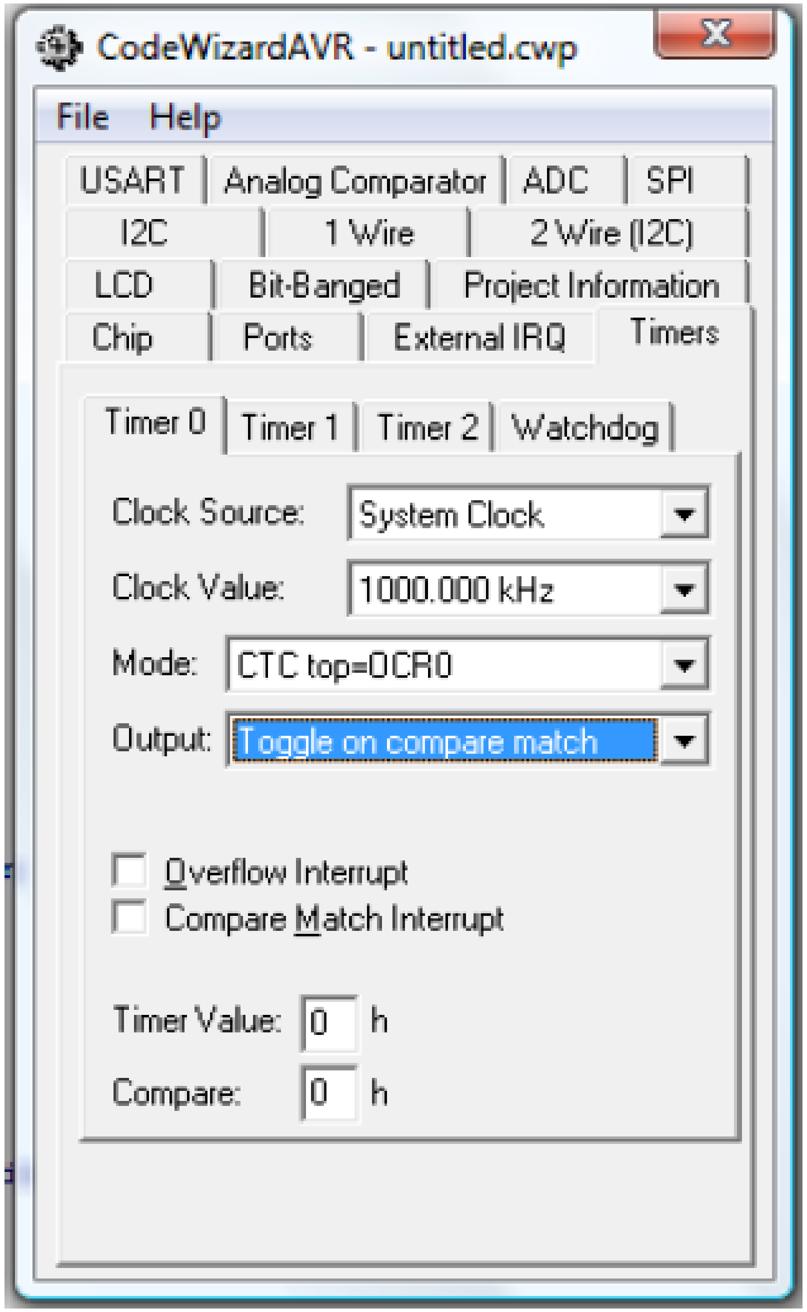
- Fixed Duty Cycle (D = 0.5)
- Variable Frequency (F)

$$F = \frac{ft}{2\,(OCR0+1)}$$

$$D = 0.5$$

By changing value of OCR0 we can change the value of output pulse frequency. As OCR0 is an 8bit register it can vary from 0 to 255.

$$0 \le OCR0 \le 255$$



Clock Source: System Clock
Clock Value: 1000.000 kHz
Mode: CTC top=OCR0
Output: Toggle on compare match
Overflow Interrupt [ ]
Compare Match Interrupt [ ]
Timer Value: 0 h
Compare: 0 h

NOW LET US HAVE A LOOK AT CVAVR AND AVR STUDIO AND HOW TO PRACTICALLY PROGRAM AN ATMEGA 16

thank you