INTERFACING

TIMERS 2 [PWM & DC MOTOR & SERVO MOTOR]

AMIT

Pulse Width Modulation:

Definition:

- Pulse-width modulation (PWM) is a modulation process or technique used in most communication systems for encoding the amplitude of a signal right into a pulse width or duration of another signal, usually a carrier signal, for transmission.

- Although PWM is also used in communications, its main purpose is to control the power that is supplied to various types of electrical devices, most especially to inertial loads such as AC/DC motors.

- Pulse-width modulation (PWM), or pulse-duration modulation (PDM), is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts.

AMIT

Pulse Width Modulation:

Definition:

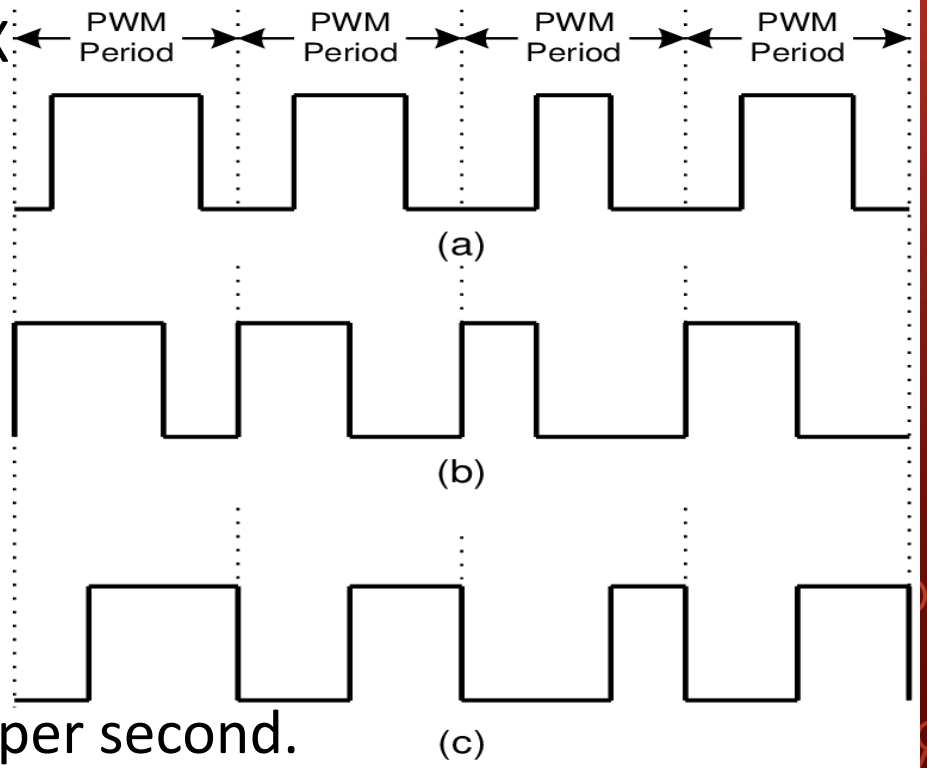
- As we mentioned before, PWM is a method to reduce the average of power.
- It can be generated by on-off sequence, I mean making a pin to be high then to be low in a speed sequence.
- PWM is like the potentiometer which also reduces the power according to its position which is human dependency controllable and most of power is consumed into it and it depends on voltage divider Law, but PWM is automatically control “by microcontroller” and the power is totally consumed inside the load, and it depends on root mean square Law.

AMIT

Pulse Width Modulation:

Main Parameters:

- Amplitude:
  - The voltage difference between the MAX voltage of cycle “on state” and the MIN voltage of cycle “off state”.



- Period:
  - It is the time of full cycle “High level time + Low level time”.

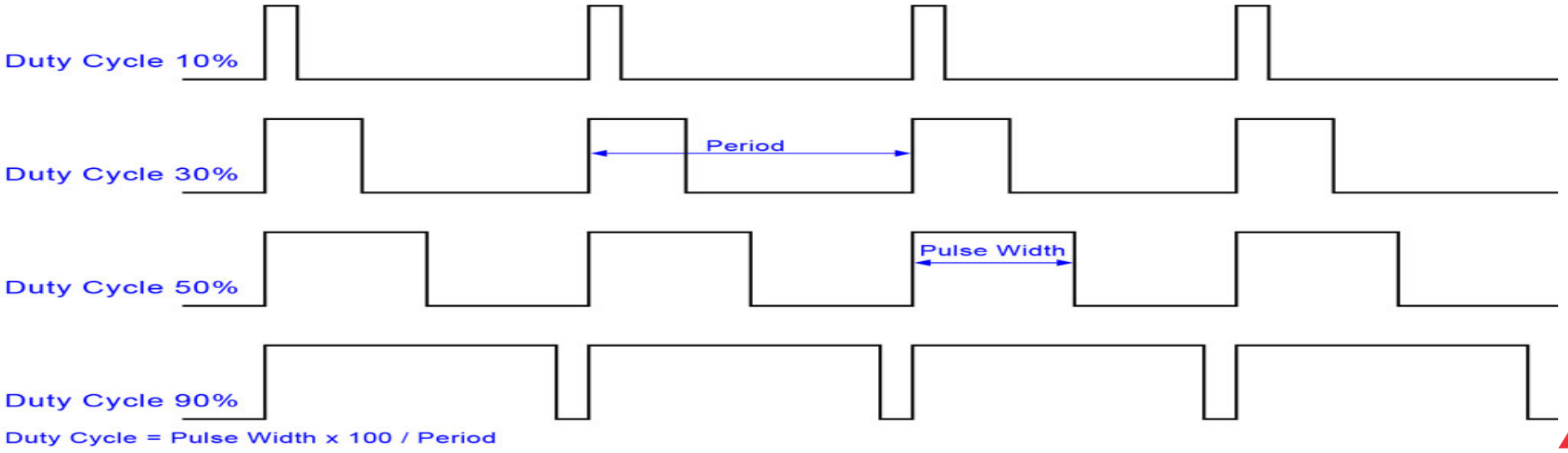
- Frequency:
  - It is the total number of cycles “period” per second.
  - $freq = \frac{1}{period\ of\ full\ cycle}$

AMIT

Pulse Width Modulation:

Main Parameters:

- Duty Cycle:
  - It is the ratio between the High-level time “on period” and the total time of the cycle “off period”.
  - $Duty\ Cycle = \frac{High\ level\ period}{time\ of\ full\ cycle} = \frac{Pulse\ Width}{Period}$

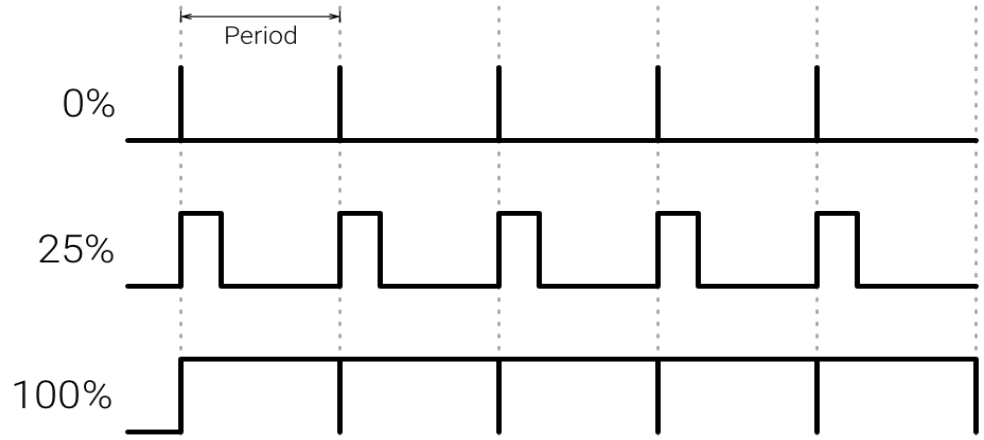


AMIT

Pulse Width Modulation:

Main Parameters:

- Duty Cycle:
  - It is the ratio between the High-level time “on period” and the total time of the cycle “off period”.
  - $Duty\ Cycle\ = \frac{High\ level\ period}{time\ of\ full\ cycle}$



- Root Mean Square:
  - It is the effective volt that affects on the circuit, I mean that the equivalent voltage that has the same effective of PWM voltage.
  - $RMS = Amplitude\ \times \sqrt{Duty\ Cycle}$

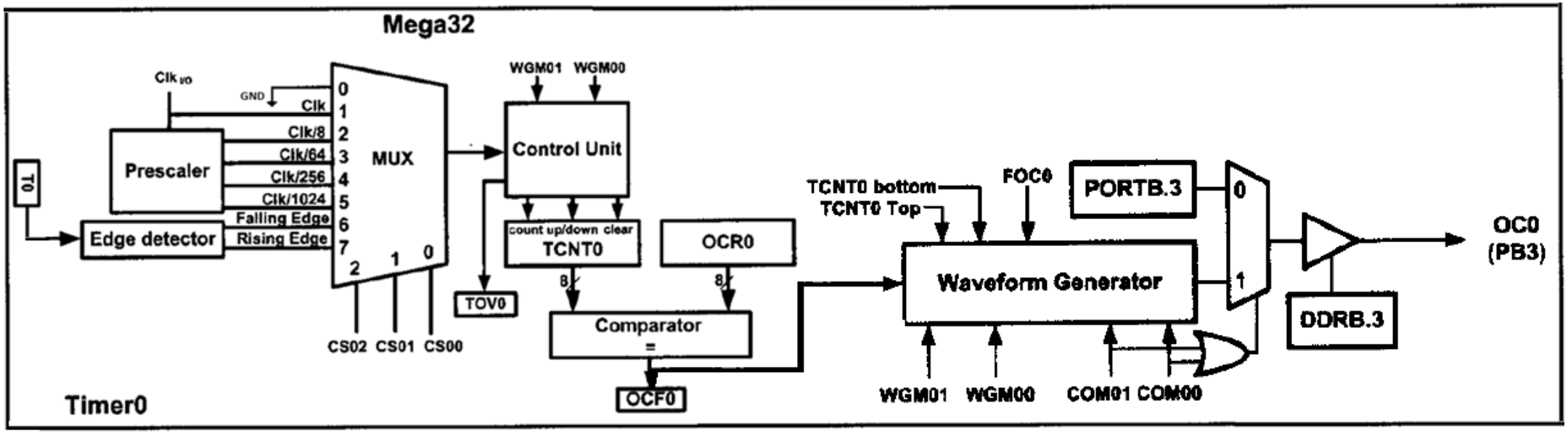
AMIT

Pulse Width Modulation:

Hardware Implementation:

- The hardware of PWM is integrated with the hardware of the Timers, so, the Timers can be configured to generate a PWM.

- Let's discover how to generate a PWM using Timer Hardware.



Pulse Width Modulation:

Hardware Implementation:

- As we mentioned before that there is a I/O pin that the wave generator can control its level value according to the value of “COMxx” bits into overflow and compare match mode.
- Now, the mode of Timer will be changed to PWM generation, so the control unit behavior will act to generate a PWM away from the behavior of overflow or compare match.
- The main function of TCNTx is to overflow if it reaches its top.
- The main function of OCRx is to Compare with TCNTx, then a signal will be generated if the value into TCNTx equals the value of OCRx.

AMIT

Pulse Width Modulation:

Hardware Implementation:

- Because any PWM needs to be high for a specific time, then to be low for a specific time, we will exploit TCNTx and OCRx to change the value level on OCx pin using wave generator.

- TCNTx will be increased every clock cycle as usual, till it reaches its overflow point, and OCRx will be compared with TCNTx value every clock cycle, till they become having the same value, a signal will be generated, So on OCRx signal the OCx pin value will be changed and on the top of TCNTx the OCx pin value will be changed again.

- The previous explanation is exactly FAST PWM, it is called by fast because the TCNTx is overflowed after reaching its TOP.

AMIT

Pulse Width Modulation:

FAST PWM:

- Because any PWM needs to be high for a specific time, then to be low for a specific time, we will exploit TCNTx and OCRx to change the value level on OCx pin using wave generator.

- TCNTx will be increased every clock cycle as usual, till it reaches its overflow point, and OCRx will be compared with TCNTx value every clock cycle, till they become having the same value, a signal will be generated, So on OCRx signal the OCx pin value will be changed and on the top of TCNTx the OCx pin value will be changed again.

- The previous explanation is exactly FAST PWM, it is called by fast because the TCNTx is overflowed after reaching its TOP.

AMIT

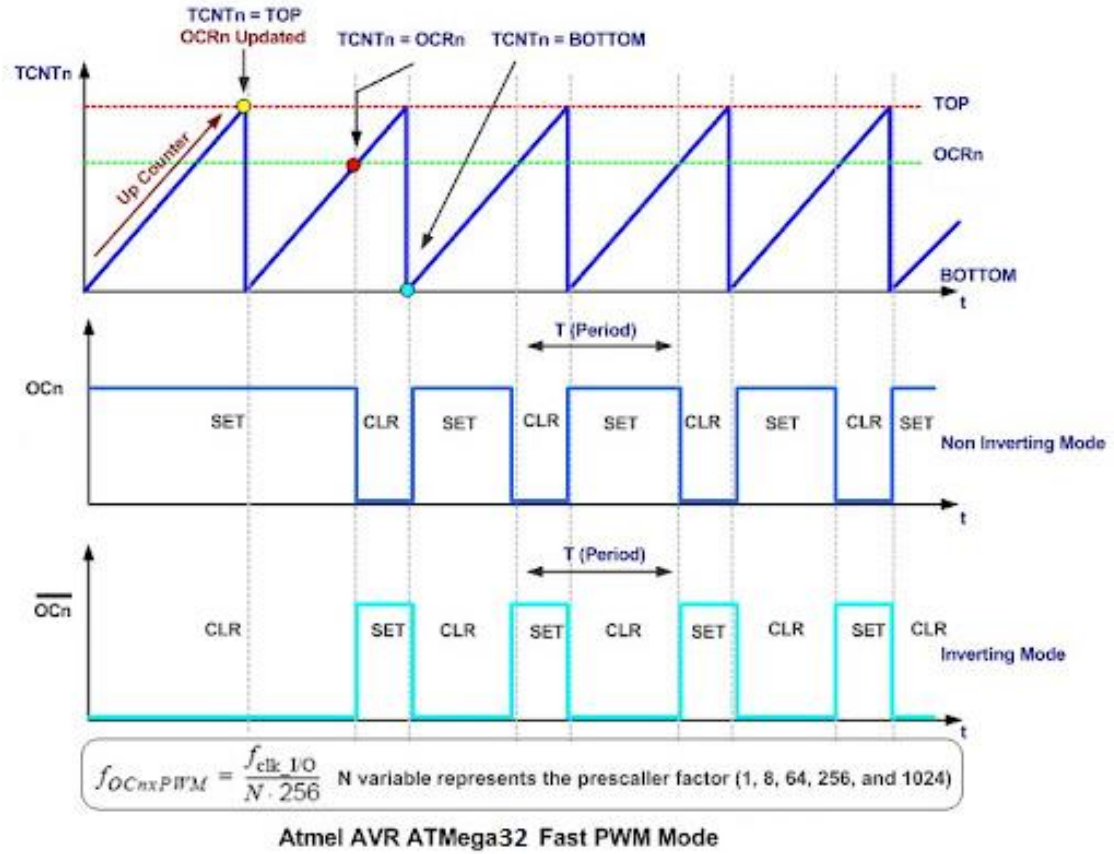
Pulse Width Modulation:

FAST PWM:

- As the shown figure, Fast PWM mode into the Timers, the wave generator will change the value of OCx pin every comparing and the Top of TCNTx.

- As we mentioned, OCx pin will be changed on OCRx value and will be changed again on Top on TCNTx, so we have two modes:
  - Inverted Mode.
  - Non-inverted Mode.

    Let us declare the difference.



Atmel AVR ATMega32 Fast PWM Mode

AMIT

Pulse Width Modulation:

FAST PWM:

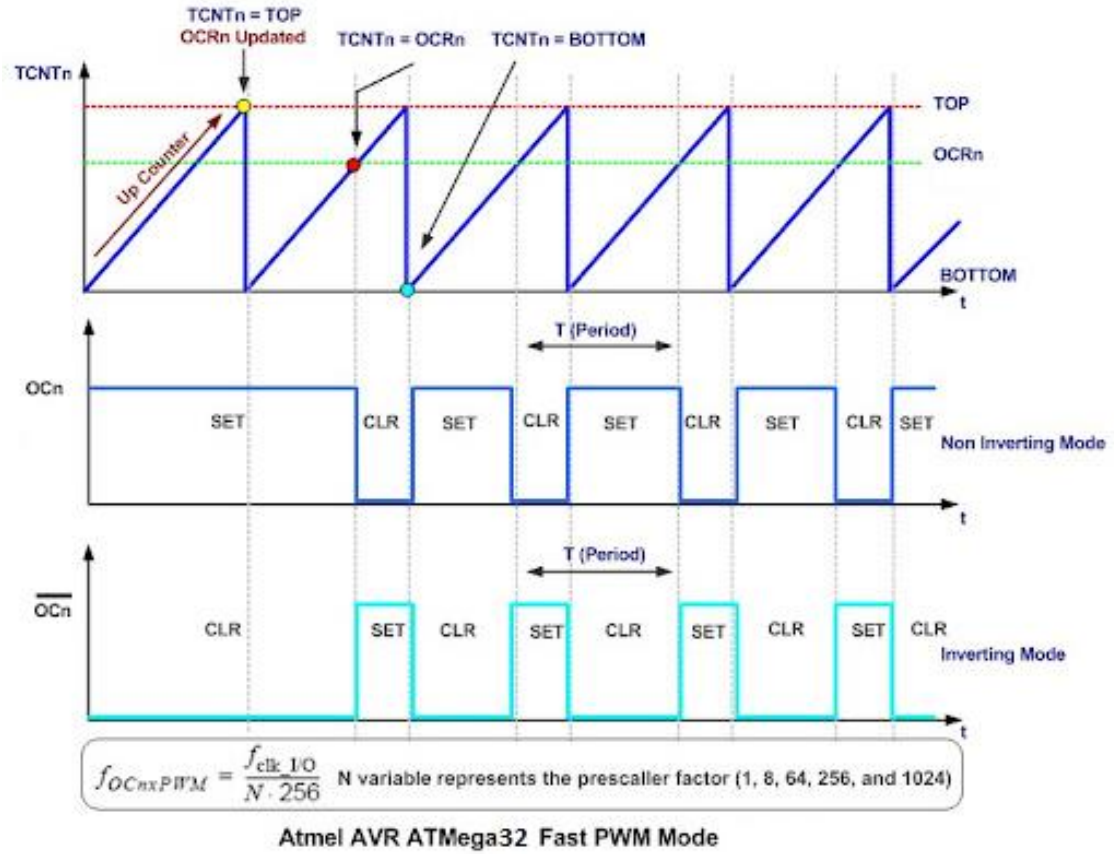
- Non-inverted Mode:

  - In this mode, the action on OCx pin will be clearing the pin on comparing and setting the pin on Top of TCNTx.

  - The Law of Duty Cycle will be:

$$Duty\ Cycle = \frac{OCRx\ value + 1}{Top\ of\ TCNTx + 1}$$

  - It is called non-inverted because the relation between OCRx value and the Duty Cycle is non-inverted.



Atmel AVR ATMega32 Fast PWM Mode

AMIT

Pulse Width Modulation:

FAST PWM:

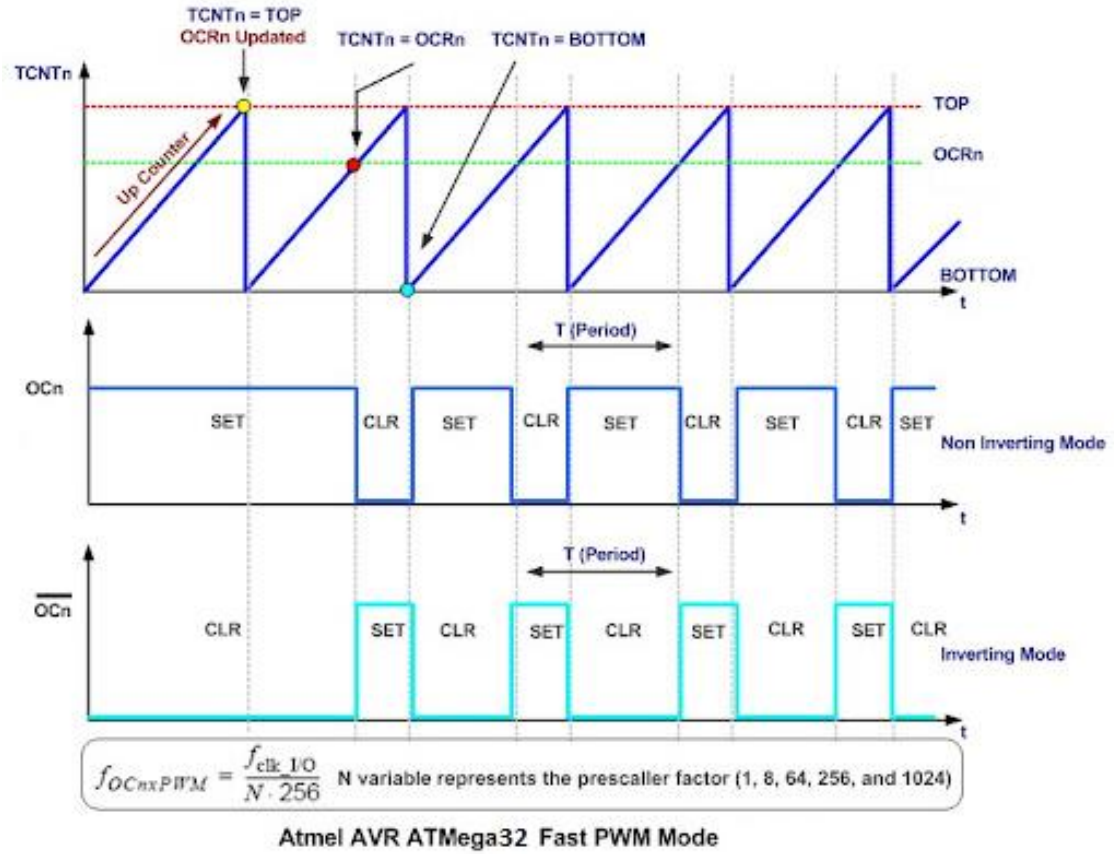
- Inverted Mode:

  - In this mode, the action on OCx pin will be setting the pin on comparing and clearing the pin on Top of TCNTx.

  - The Law of Duty Cycle will be:

$$Duty\ Cycle = \frac{Top\ of\ TCNTx - OCRx\ value}{Top\ of\ TCNTx+1}$$

  - It is called inverted because the relation between OCRx value and the Duty Cycle is inverted.



Atmel AVR ATMega32 Fast PWM Mode

AMIT

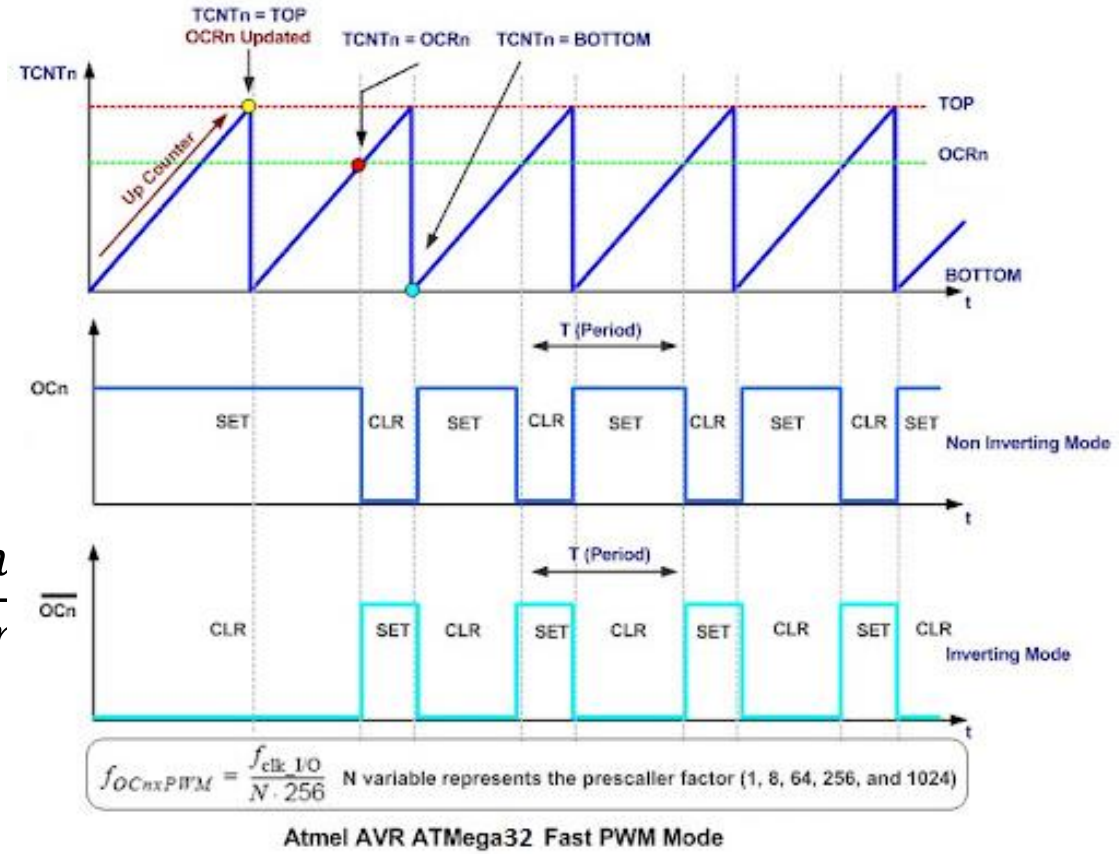
Pulse Width Modulation:

FAST PWM:

- The frequency of this PWM can be calculated by:

$$PWM\ frequency = \frac{frequency\ of\ system}{counts \times prescaler}$$

$$PWM\ frequency = \frac{frequency\ of\ system}{(TOP + 1) \times prescaler}$$



Atmel AVR ATMega32 Fast PWM Mode

AMIT

Pulse Width Modulation:

Phase-Correct PWM:

- As we mentioned before in Fast PWM, it is called by fast PWM because the TCNTx is overflowed after reaching it Top, but the phase-correct PWM TCNTx counts down after reaching its Top, so the total counts of this cycle will be double of Top value.

- Because TCNTx counts-up then counts down, it will be compared with OCRx twice “at counting-up and at counting-down”, the action on OCx pin will be taken on OCRx value only.

- OCx pin will be triggered at OCRx value while counting-up, and an inverse triggering on OCx will be taken while counting down.

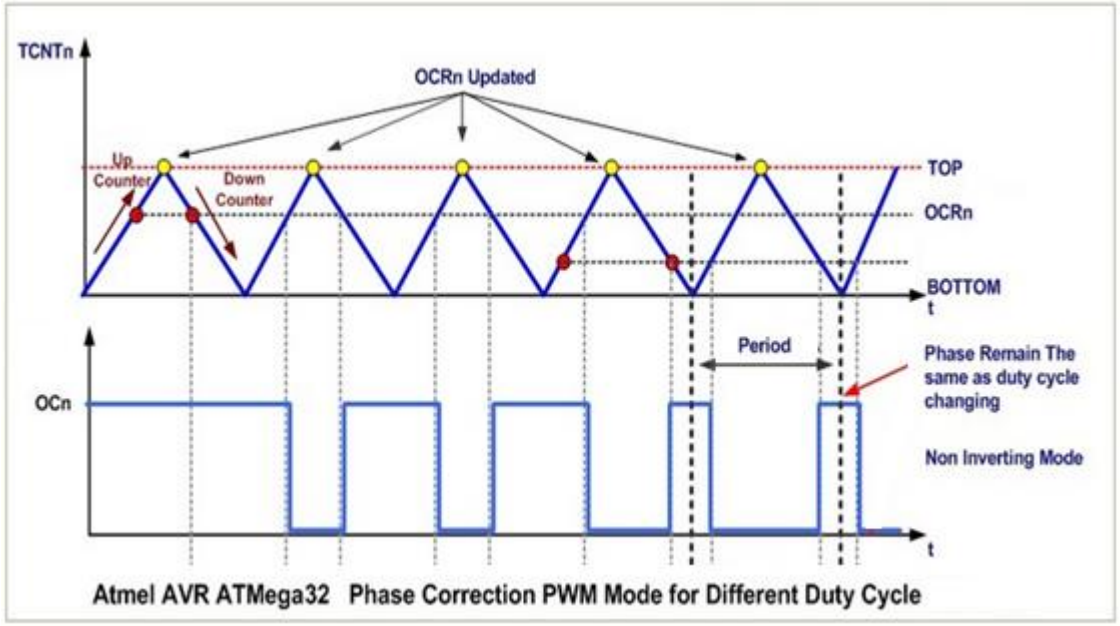
AMIT

Pulse Width Modulation:

Phase-Correct PWM:

- As the shown figure, phase-correct PWM mode into the Timers, the wave generator will change the value of OCx pin every comparing while counting-up and down.

- As we mentioned, OCx pin will be changed on OCRx value twice while counting-up and down, so we have two modes:
  - Inverted Mode.
  - Non-inverted Mode.



Pulse Width Modulation:

Phase-Correct PWM:

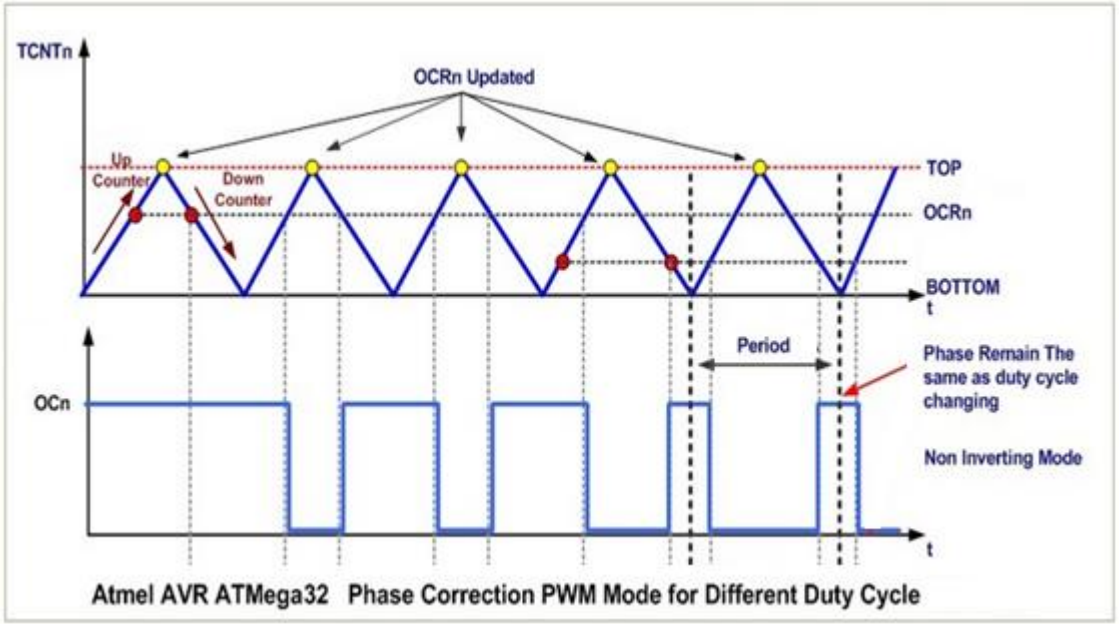
- Non-inverted Mode:

  - In this mode, the action on OCx pin will be clearing the pin while counting-up and setting while counting down.

  - The Law of Duty Cycle will be:

$$Duty\ Cycle = \frac{2 \times OCRx\ value}{2 \times Top\ of\ TCNTx}$$

  - It is called non-inverted because the relation between OCRx value and the Duty Cycle is non-inverted.



Atmel AVR ATMega32 Phase Correction PWM Mode for Different Duty Cycle

AMIT

Pulse Width Modulation:

Phase-Correct PWM:

- Inverted Mode:

  - In this mode, the action on OCx pin will be setting the pin while counting-up and clearing while counting down.

  - The Law of Duty Cycle will be:

$$Duty\ Cycle = \frac{2 \times Top\ of\ TCNTx - 2 \times OCRx\ value}{2 \times Top\ of\ TCNTx}$$

  - It is called inverted because the relation between OCRx value and the Duty Cycle is inverted.

AMIT

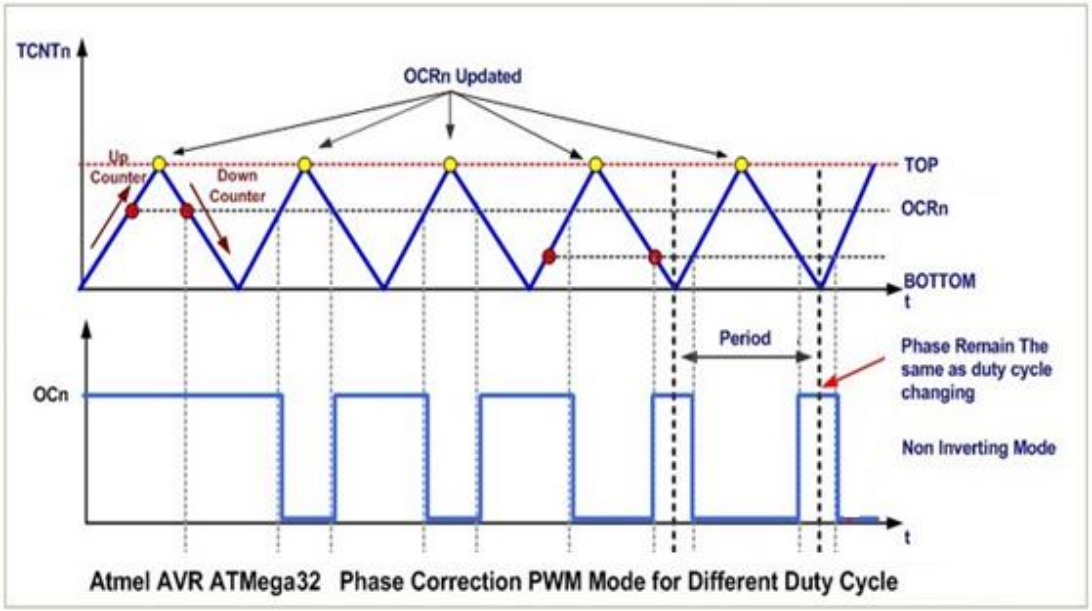
Pulse Width Modulation:

Phase-Correct PWM:

- The frequency of this PWM can be calculated by:

$$PWM\ frequency = \frac{frequency\ of\ system}{counts \times prescaler}$$

$$PWM\ frequency = \frac{frequency\ of\ system}{(2 \times TOP) \times prescaler}$$



Atmel AVR ATMega32 Phase Correction PWM Mode for Different Duty Cycle

Pulse Width Modulation:

How to Configure Timer to generate PWM:

- Configuration of Timer0:

  - from TCCR0, there are two bits "**WGM00, WGM01**", they can be set as:

    - Fast PWM "**11**".
    - Phase-Correct PWM "**01**".

  - Configuration of Mode "Inverted or Non-Inverted" by setting "**COM00, COM01**" bits.

Table 38. Waveform Generation Mode Bit Description[1]

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
3	1	1	Fast PWM	0xFF	TOP	MAX

AMIT

Pulse Width Modulation:

How to Configure Timer to generate PWM:

- Configuration of Timer0:

  - from TCCR0, there are two bits "**WGM00, WGM01**", they can be set as:

    - Fast PWM "**11**".
    - Phase-Correct PWM "**01**".

  - Configuration of Mode "Inverted or Non-Inverted" by setting "**COM00, COM01**" bits.

Table 38. Waveform Generation Mode Bit Description[1]

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
3	1	1	Fast PWM	0xFF	TOP	MAX

AMIT

Pulse Width Modulation:

How to Configure Timer to generate PWM:

- Configuration of Timer0:

  - from TCCR0, there are two bits "**WGM00, WGM01**", they can be set as:
    - Fast PWM "**11**".
    - Phase-Correct PWM "**01**".

Table 38. Waveform Generation Mode Bit Description[1]

Mode	WGM01 (CTC0)	WGM00 (PWM0)	Timer/Counter Mode of Operation	TOP	Update of OCR0	TOV0 Flag Set-on
1	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
3	1	1	Fast PWM	0xFF	TOP	MAX

- Configuration of Mode "Inverted or Non-Inverted" by setting "**COM00, COM01**" bits.

AMIT

Pulse Width Modulation:

How to Configure Timer to generate PWM:

- Configuration of Timer0:
  - from TCCR0, there are two bits "**COM00, COM01**" bits which configure the Mode "Inverted or Non-Inverted" PWM:

  - Phase-Correct PWM:
    - "**10**": Non-Inverted.
    - "**11**": Inverted.

Table 41. Compare Output Mode, Phase Correct PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match when up-counting. Set OC0 on compare match when downcounting.
1	1	Set OC0 on compare match when up-counting. Clear OC0 on compare match when downcounting.

  - Fast PWM:
    - "**10**": Non-Inverted.
    - "**11**": Inverted.

Table 40. Compare Output Mode, Fast PWM Mode(1)

COM01	COM00	Description
0	0	Normal port operation, OC0 disconnected.
0	1	Reserved
1	0	Clear OC0 on compare match, set OC0 at TOP
1	1	Set OC0 on compare match, clear OC0 at TOP

AMIT

Pulse Width Modulation:

How to Configure Timer to generate PWM:

- Configuration of Timer0:
  - from TCCR0, there are three bits "**CS00, CS01, CS02**" bits which configure the prescaler of division factor:

    - they can be configured as they were configured before when the Timer0 was configured as overflow or comparing match modes.

Table 42. Clock Select Bit Description

CS02	CS01	CS00	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/(No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T0 pin. Clock on falling edge.
1	1	1	External clock source on T0 pin. Clock on rising edge.

AMIT

Pulse Width Modulation:

How to Configure Timer to generate PWM:

- Configuration of Timer0:

  - from OCR0, it is used to set your specific duty cycle if the PWM is inverted or non-inverted.

    - Non-inverted:

      $$\text{OCR0} = Duty\ cycle \times Counts$$

      $$\text{OCR0} = (Duty\ cycle \times (TOP + 1)) - 1$$, if it is fast PWM.

      $$\text{OCR0} = Duty\ cycle \times TOP$$, if it is phase-correct PWM.

    - Inverted:

      $$\text{OCR0} = Duty\ cycle \times Counts,$$

      $$\text{OCR0} = TOP - Duty\ cycle(TOP + 1)$$, if it is fast PWM.

      $$\text{OCR0} = TOP \times (1 - Duty\ cycle)$$, if it is phase-correct PWM.

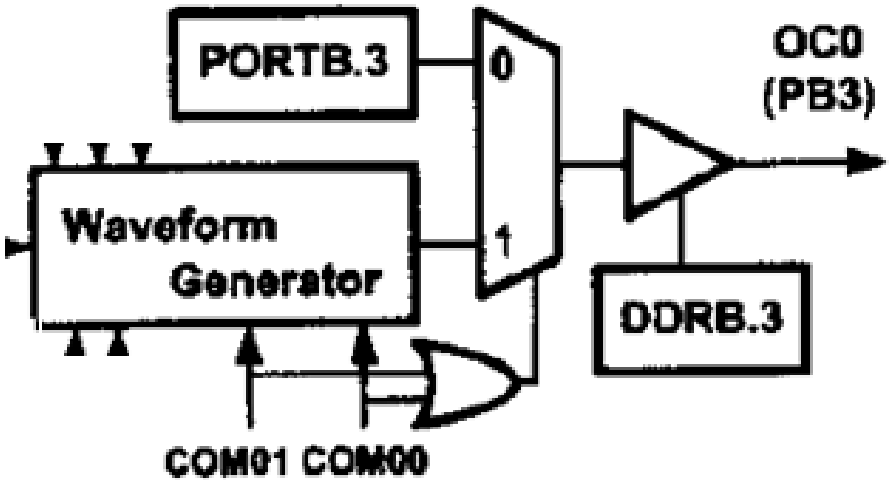
Pulse Width Modulation:

How to Configure Timer to generate PWM:

- Configuration of Timer0:

  - According to this schematic of hardware circuits, OC0 pin must be configured as an output pin to manage the wave generator to control the level of OC0 pin.

  - OC0 pin and PB3 are shared pin, so there is a multiplexer to select the controller of PB3 or OC0.



  - "**COM00, COM01**" are connected on OR gate, so, if these two bits values are zeros, the pin is connected to PORTB and disconnected from wave generator, but if any of them becomes one, the pin is disconnected from PORTB and connected to wave generator.

AMIT

Pulse Width Modulation:

Changing of PWM Frequency:

- On Timer0 or Timer2:

$$PWM\ frequency = \frac{frequency\ of\ system}{(255+1)\times prescaler}\ for\ fast\ PWM.$$

$$PWM\ frequency = \frac{frequency\ of\ system}{(2\times 255)\times prescaler}\ \text{for phase-correct PWM.}$$

- The frequency of PWM depends on:
  - System frequency: it depends hardware of microcontroller, so it will be fixed for the same system.
  - Prescaler: it depends hardware clock reduction circuit, it has five or seven options only, so it will be considered a fixed parameter.
  - Counts: here they are fixed, because TCNTx is increased until overflowing.

AMIT

Pulse Width Modulation:

Changing of PWM Frequency:

- On Timer0 or Timer2:

  According to the previous, the frequency of PWM in Timer0 or Timer2 will be constant as the following table:

Prescaler	Fast PWM at Timer0/2	Phase-Correct PWM at Timer0/2
1	62,500 Hz	31,372.55 Hz
8	7812.5 Hz	3,921.57 Hz
32	1,953.125 Hz (Timer2 only)	980.39 Hz (Timer2 only)
64	976.5625 Hz	490.20 Hz
128	488.28 Hz (Timer2 only)	245.10 Hz (Timer2 only)
256	244.14 Hz	122.55 Hz
1024	61.04 Hz	30.64 Hz

AMIT

Pulse Width Modulation:

Timer 1:

- Modes of Timer1:
  Timer1 is a 16-bit Timer, all registers into it are 16-bit size, it also has four bits for mode configuration.

  It has Two channels of PWM "**OC1A, OC1B**".

  It has an input-capture unit.

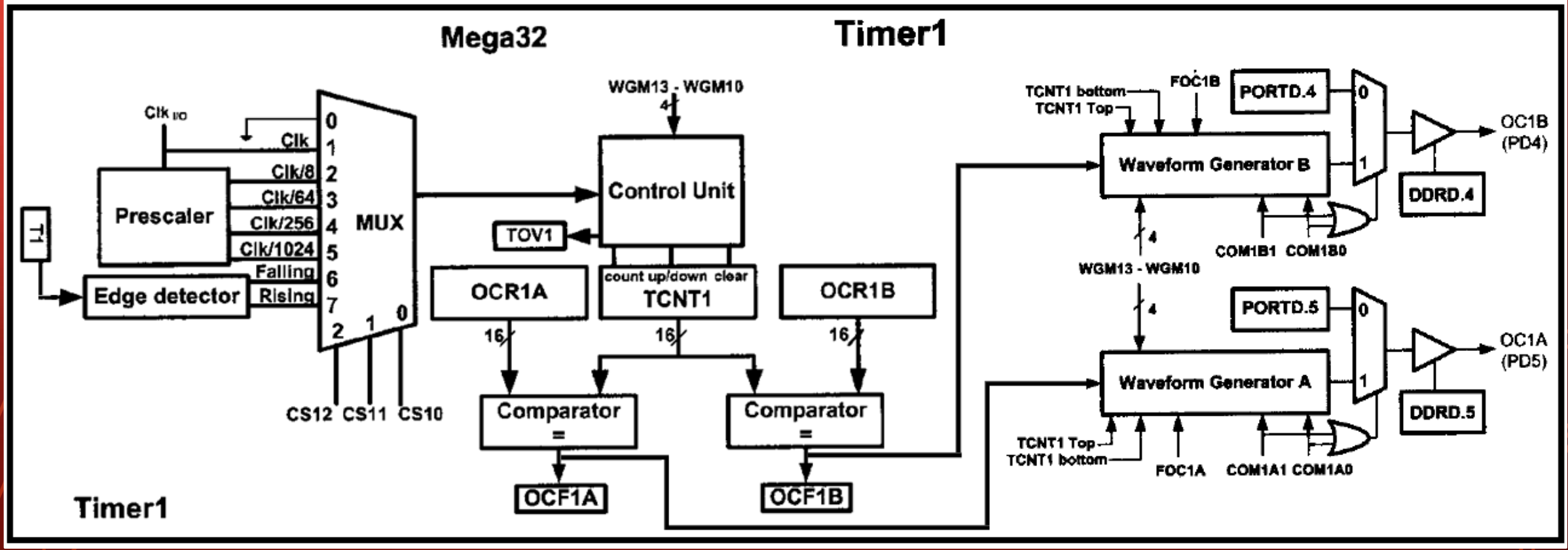
  It has about 16 mode as the following table at the next slide:

AMIT

Pulse Width Modulation:

Timer 1:

- Hardware of Timer1:



Pulse Width Modulation:

Timer 1:

Table 47. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	TOP	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	TOP	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	TOP	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	Reserved	–	–	–
14	1	1	1	0	Fast PWM	ICR1	TOP	TOP
15	1	1	1	1	Fast PWM	OCR1A	TOP	TOP

Pulse Width Modulation:

Timer 1:

- Normal: “**0000**”
  it is the overflow mode like Timer0/2, but it is overflowed after its Top “**65,535**”.

- PWM, Phase-Correct “**8-bit**”: “**0001**”
  it is phase-correct mode exactly like Timer0/2, with any difference between of them.

- PWM, Phase-Correct “**9-bit**”: “**0010**”
  it is phase-correct mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is “**511**”.

AMIT

Pulse Width Modulation:

Timer 1:

- PWM, Phase-Correct “**10-bit**”: “**0011**”
  it is phase-correct mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is “**1023**”.

- CTC, Comparing Match “**OCR1A**”: “**0100**”
  it is compare time mode exactly like Timer0/2, but the comparing occurs only when TCNT1 equals OCR1A,
  “**Compares only with Channel A only**”.

AMIT

Pulse Width Modulation:

Timer 1:

- PWM, Fast PWM “**8-bit**”: “**0101**”
  it is Fast PWM mode exactly like Timer0/2, with any difference between of them.

- PWM, Fast PWM “**9-bit**”: “**0110**”
  it is Fast PWM mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is “**511**”.

- PWM, Fast PWM “**10-bit**”: “**0111**”
  it is Fast PWM mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is “**1023**”.

AMIT

Pulse Width Modulation:

Timer 1:

- PWM, Phase-Correct “**ICR1**”: “**1000**”
  it is phase-correct mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is the value of input-capture register “**ICR1**”, the triggering event will be occurred on “OCR1” and “OCR1B” which the two channels will run at the same frequency if their values are less than the value of “ICR1”.

- PWM, Phase-Correct “**OCR1A**”: “**1001**”
  it is phase-correct mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is the value of output-comparing register A “**OCR1A**”, the triggering event will be occurred on “OCR1B”,if “OCR1B” value is less than the value of “OCR1A”.

AMIT

Pulse Width Modulation:

Timer 1:

- PWM, Phase-Correct “**ICR1**”: “**1010**”
- PWM, Phase-Correct “**OCR1A**”: “**1011**”
  they are identically similar to the two-previous modes, but the only difference between of them that into these mode the updating of OCR1A/ICR1 “**we will discuss it later**” occurs on the Top of TCNT1, but the previous mode the updating of OCR1x occurs on the Bottom of TCNT1.
- CTC, Comparing Match “**ICR1**”: “**1100**”
  it is compare time mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is the value of input-capture register “**ICR1**”, the comparing match will be occurred on “OCR1” and “OCR1B”, if their values are less than the value of “ICR1” on which TCNT1 will be zeroized.

AMIT

Pulse Width Modulation:

Timer 1:

- PWM, Fast PWM "**ICR1**": "**1110**"
  it is Fast PWM mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is the value of input-capture register "**ICR1**", the triggering event will be occurred on "OCR1" and "OCR1B" which the two channels will run at the same frequency if their values are less than the value of "ICR1".

- PWM, Fast PWM "**OCR1A**": "**1111**"
  it is Fast PWM mode exactly like Timer0/2, but the difference between of them is that the top of Timer1 is the value of output-comparing register A "**OCR1A**", the triggering event will be occurred on "OCR1B",if "OCR1B" value is less than the value of "OCR1A".

AMIT

Pulse Width Modulation:

Changing of PWM Frequency:

- On Timer1:

  there are about twelve modes to generate a PWM into Timer1:

  - PWM, fast or phase correct **“8 / 9 / 10 bit”**, these modes have a fixed frequencies like timer0/2 at every value of prescaler.

  - PWM, fast or phase correct “**ICR1 , OCR1A**”, these modes allow the user to assign any value of counts into “**ICR1 or OCR1A**”, so now we can generate any frequency by calculating the required value to generate this frequency.

AMIT

Pulse Width Modulation:

Changing of PWM Frequency:

- Frequency Law:

$$PWM\ frequency = \frac{frequency\ of\ system}{Counts \times prescaler}$$

- Frequency of Fast PWM Law:

$$PWM\ frequency = \frac{frequency\ of\ system}{Top \times prescaler}$$

- Frequency of phase-correct Law:

$$PWM\ frequency = \frac{frequency\ of\ system}{(2 \times Top) \times prescaler}$$

AMIT

Pulse Width Modulation:

Changing of PWM Frequency:

- The new top “counts” for Fast PWM mode:

$$\mathbf{OCR1A} \text{ or } \mathbf{ICR1} = \frac{frequency\ of\ system}{PWM\ frequency\ required \times prescaler}$$

- The new top “counts” for Phase-Correct PWM mode :

$$\mathbf{OCR1A} \text{ or } \mathbf{ICR1} = \frac{frequency\ of\ system}{2 \times PWM\ frequency\ required \times prescaler}$$

AMIT

Pulse Width Modulation:

Updating of OCRx / ICR1 Registers into PWM modes:

- The updating of OCRx/ICR1 register differs according to the Timer if it was overflow/compare-match or fast/phase-correct.

- The updating of OCRx/ICR1 at overflow/compare-match modes is immediately occurred.

- The updating of OCRx/ICR1 at fast/phase-correct PWM modes is commonly occurred at the Top value of TCNTx except the "**1000 , 1001**" modes at Timer1, the updating is occurs at the bottom of TCNT1.

but, how can the updating be delayed until the top/bottom of TCNTx?

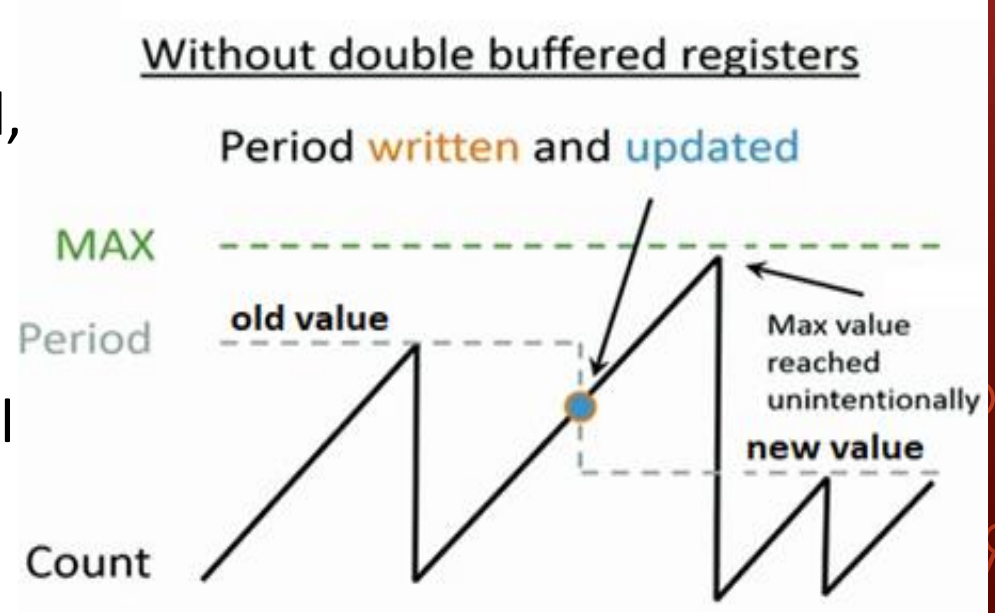
AMIT

Pulse Width Modulation:

Updating of OCRx / ICR1 Registers into PWM modes:

- The OCRx/ICR1 registers in general are double-buffered registers, the double-buffered register save the updated value into it until the TCNTx reaches its Top, then it will copied to OCRx/ICR1 by hardware.

- If these registers are not a doubled-buffered, the following scenario will be occurred, Like the following figure, the updating occurs when TCNTx becomes higher than the new value that be updated, so the comparing will not be occurred and the TCNTx will be overflowed.

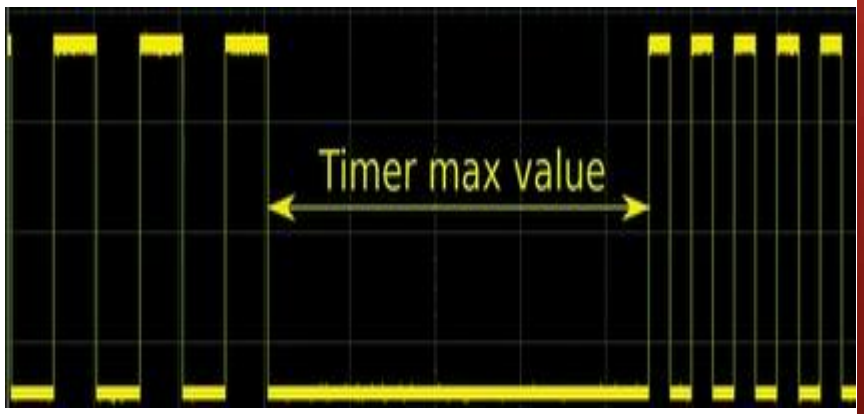


AMIT

Pulse Width Modulation:

Updating of OCRx / ICR1 Registers into PWM modes:

- According to the previous, the PWM will contain a stretching cycle into it like the following figure.



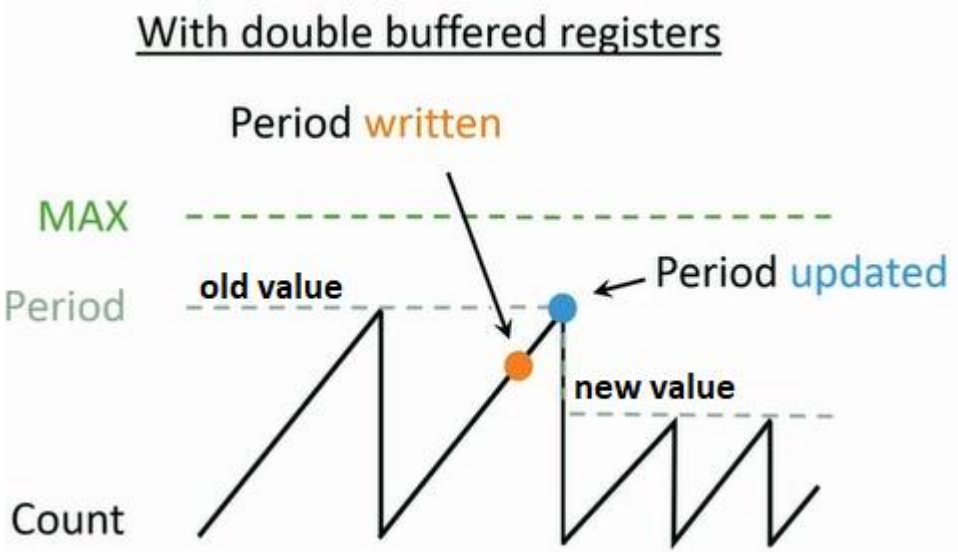
- So, This mechanism of double-buffered allows to avoid generating a sudden stretching within the sequence of PWM cycles,
  like the following figure.

AMIT

Pulse Width Modulation:

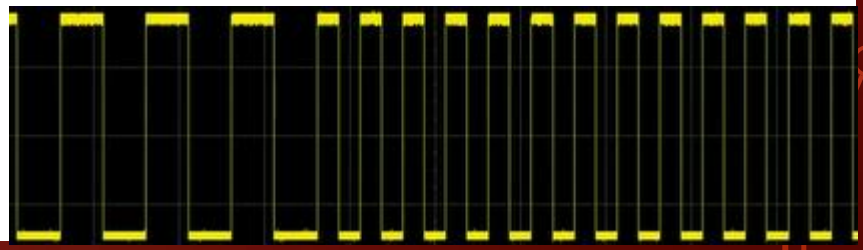
Updating of OCRx / ICR1 Registers into PWM modes:

- So, when the double-buffered mechanism is used, the TCNTx register will not be overflowed like the previous mechanism.



- It is the same if the updating occurs at the bottom of TCNTx, the main reason to use it is avoiding updating while running by a value when TCNTx may has a higher value than it because it will cause an overflow.

- The output PWM after using the double-buffered do not contain a stretched clock.



Pulse Width Modulation:

Accessing the 16-bit registers:

- The TCNT1, OCR1A/B, and ICR1 are 16-bit registers that can be accessed by the AVR CPU via the 8-bit data bus.
- The 16-bit register must be accessed byte-wise, using two read or write operations.
- Each 16-bit timer has a single 8-bit TEMP register for temporary storing of the high byte of the 16-bit access.
- The same temporary register is shared between all 16-bit registers within each 16-bit timer.

Pulse Width Modulation:

Accessing the 16-bit registers:

- Accessing the low byte triggers the 16-bit read or write operation:
  - When the low byte of a 16-bit register is written by the CPU, the high byte that is currently stored in TEMP
  - When the low byte being written, both are copied into the 16-bit register in the same clock cycle.
  - When the low byte of a 16-bit register is read by the CPU, the high byte of the 16-bit register is copied into the TEMP register in the same clock cycle as the low byte is read, and must be read subsequently.

PWM Driver:

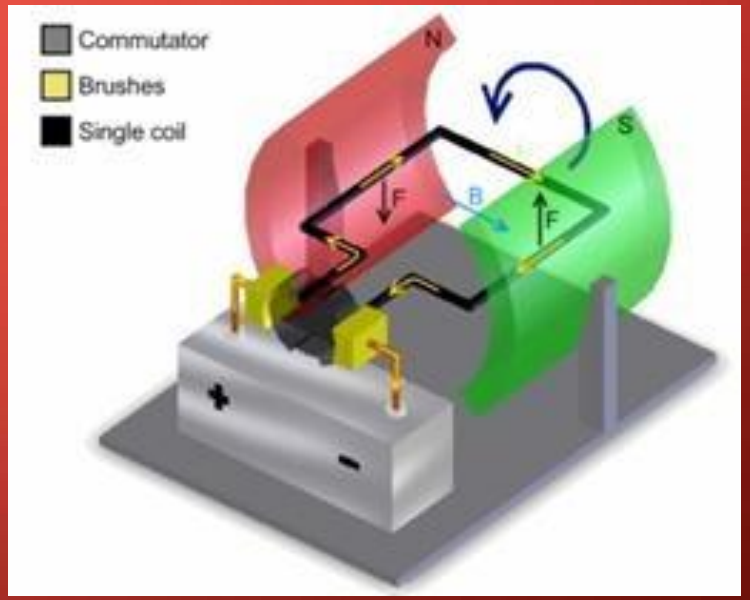
Time To Code

AMIT

DC MOTORS:

- **The main idea of DC Motors is :**
  - **There are three parameters:**
    - Electricity.
    - Magnetic field.
    - Motion.

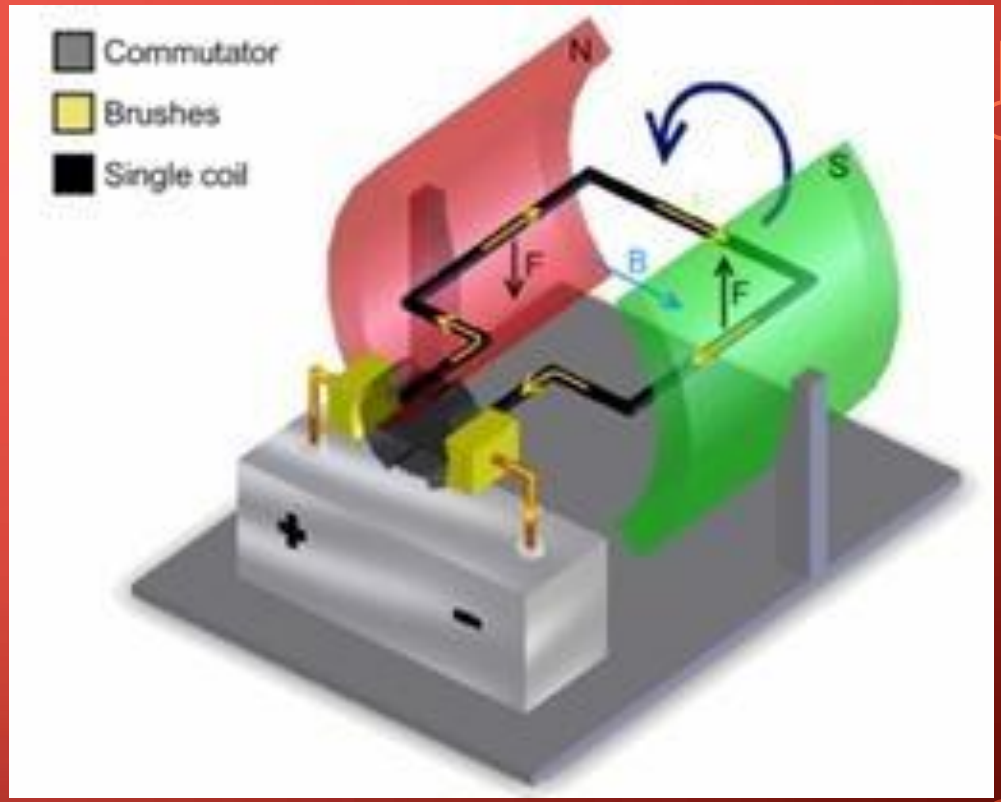
    If any two parameters exist, third parameter will be generated.
  - So, if there an electricity in a coil into a magnetic field, the motion will be generated.
  - Fleming's Rule is used in an electric motor to determine the direction of the motive force of a conductor through which a current is flowing and cutting magnetic field lines.



AMIT

DC MOTORS:

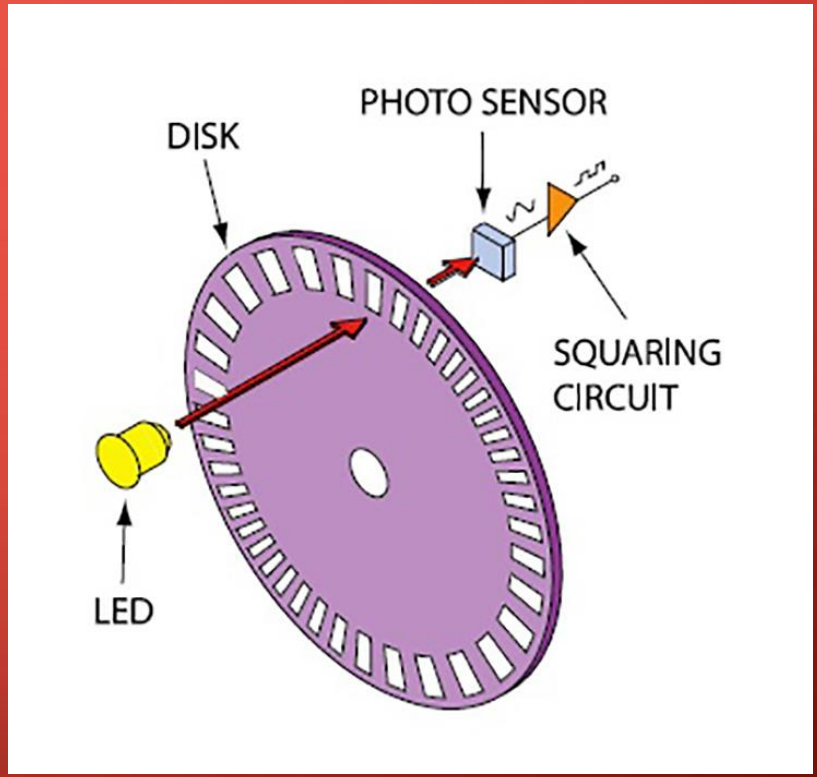
- Advantages of DC Motors:
  - DC Motors generate a continuous motion.
  - Simple to drive.
  - Has a small size.
  - Has an easy structure.
  - Low Cost.



AMIT

DC MOTORS:

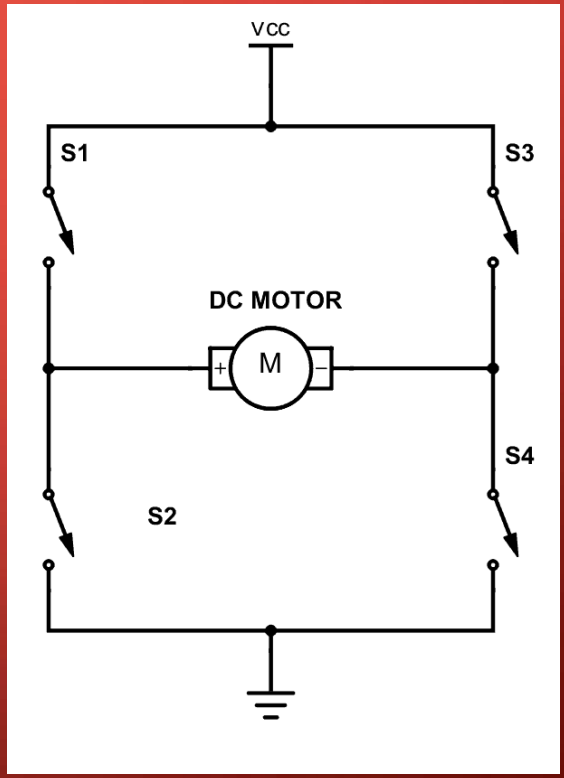
- Disadvantages of DC Motors:
  - The direction is Uncontrollable without an extra hardware circuit.
  - The rotation angle is Uncontrollable without an extra hardware circuit.
    - We can use Encoder circuit like exists into the figure besides.
  - Its speed highly varies if the load is changed or current or volt.



AMIT

CONTROLLING OF DC MOTORS:

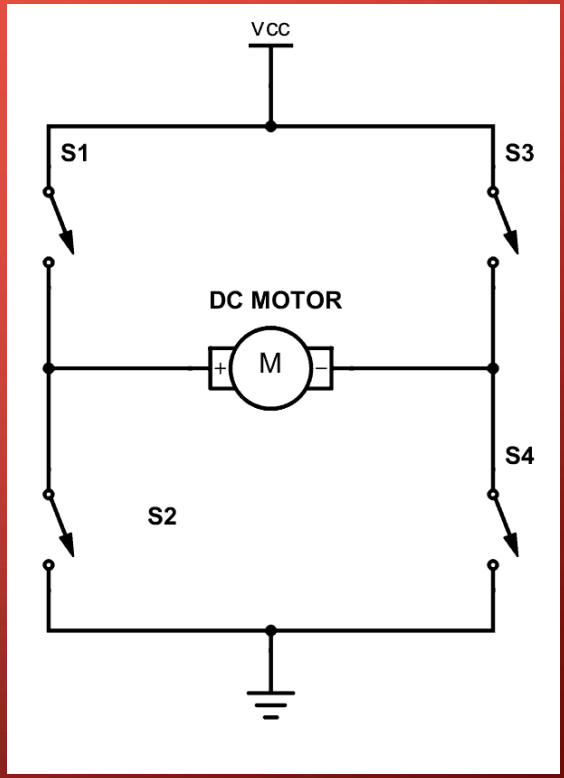
- The most common circuit is used to control the DC Motors is H-bridge.
- It is used to control the direction of DC Motor rotation.
- The figure besides, the simplest H-bridge can be implemented.
- It is implemented from four switches,
  - S1,S4 are closed, the motor will rotate at Clock-wise direction.
  - S2,S3 are closed, the motor will rotate at Counter-clock-wise direction.



AMIT

CONTROLLING OF DC MOTORS:

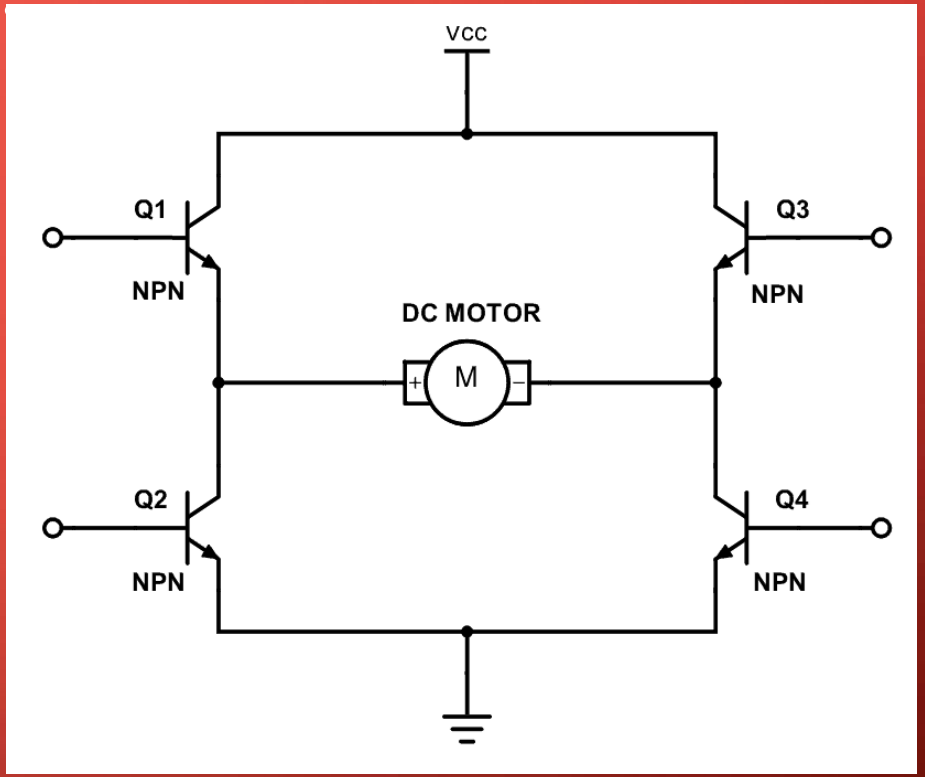
- Unfortunately, This circuit has two main defects:
  - It is a human dependency.
  - It may cause a short circuit if the S1,S2 or S3,S4 are closed at the same time.
- So, we will replace the switches by transistors.



AMIT

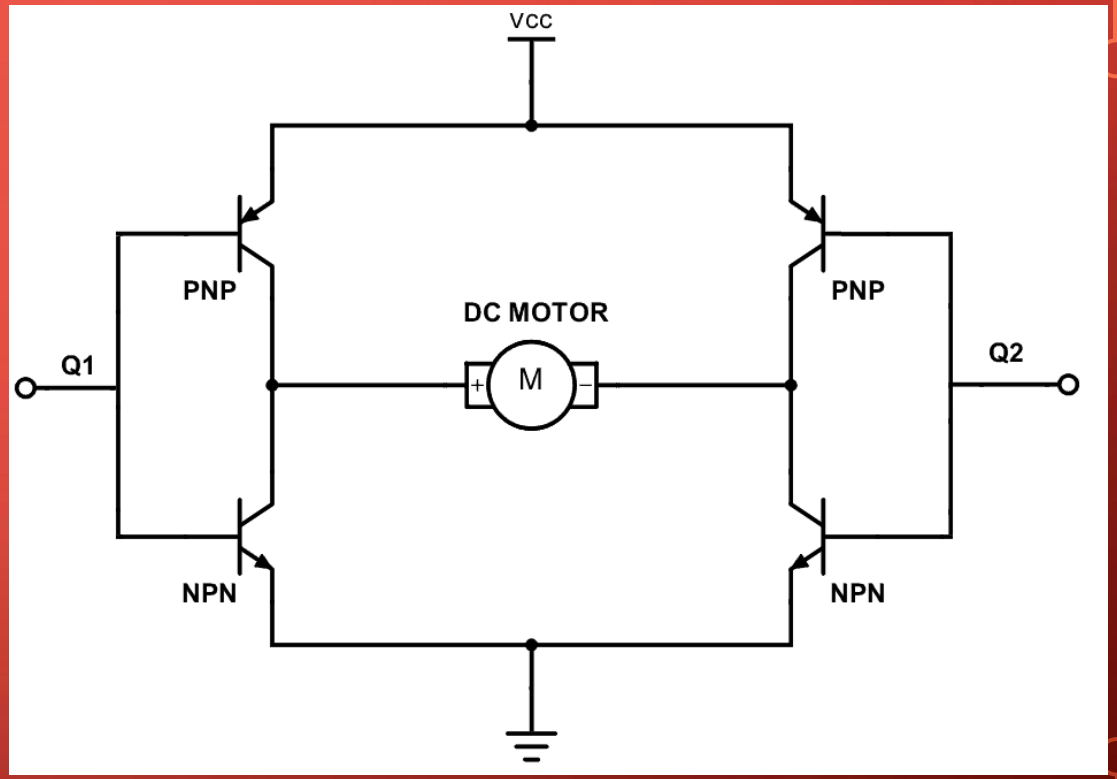
CONTROLLING OF DC MOTORS:

- We can Modify the last circuit by replacing switches by transistors.
- Now, it can be controlled by signals to change the polarity of motor and direction.
- Unfortunately, if Q3,Q4 or Q1,Q2 are closed, it will cause short circuit.



CONTROLLING OF DC MOTORS:

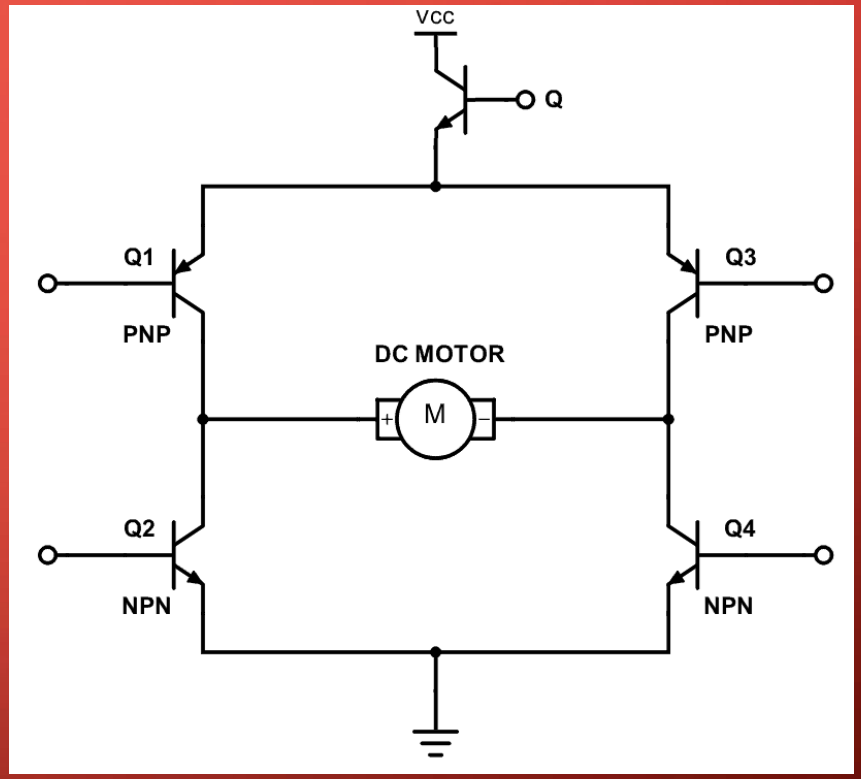
- To avoid short circuit case, we can replace the NPN transistor to PNP transistor.
- If Q1, Q2 has the same signal ,HIGH or LOW, the circuit will be opened, so there is no way to cause the short circuit.
- To change the direction, apply different signals on Q1,Q2.



AMIT

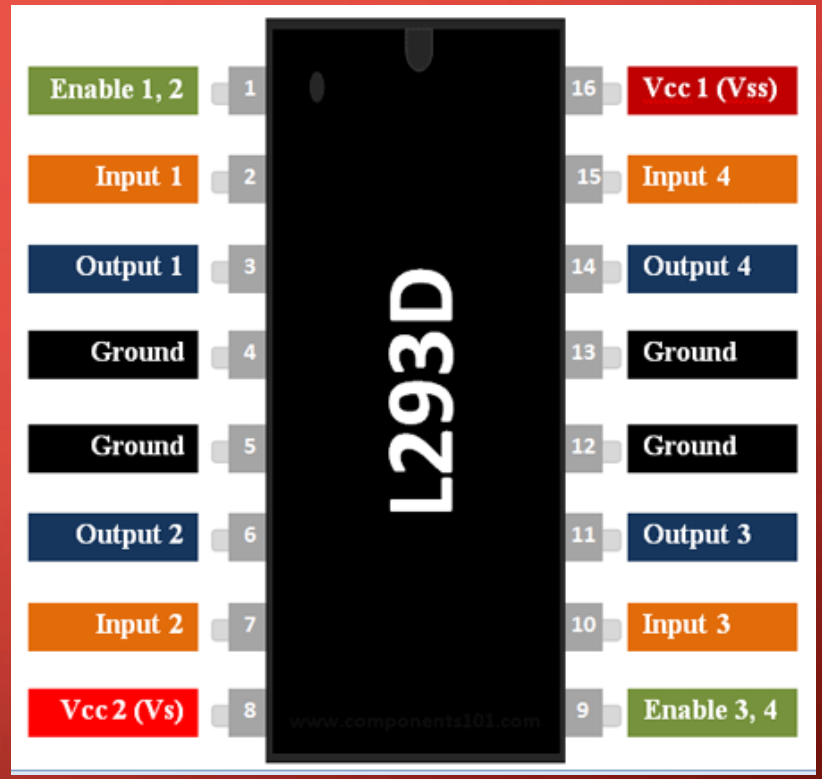
CONTROLLING OF DC MOTORS:

- This modification manage us to control the speed of the motor also, addition to the controlling on the direction.
- This is the same idea on which the L293D IC is built.



CONTROLLING OF DC MOTORS:

- L293D IC is The most common circuit is used to control the DC Motors.
- L293D is used to control two motors at the same time.
- It can control on the direction of speed and the direction of the DC Motor.



PIN CONFIGURATION OF L293D :

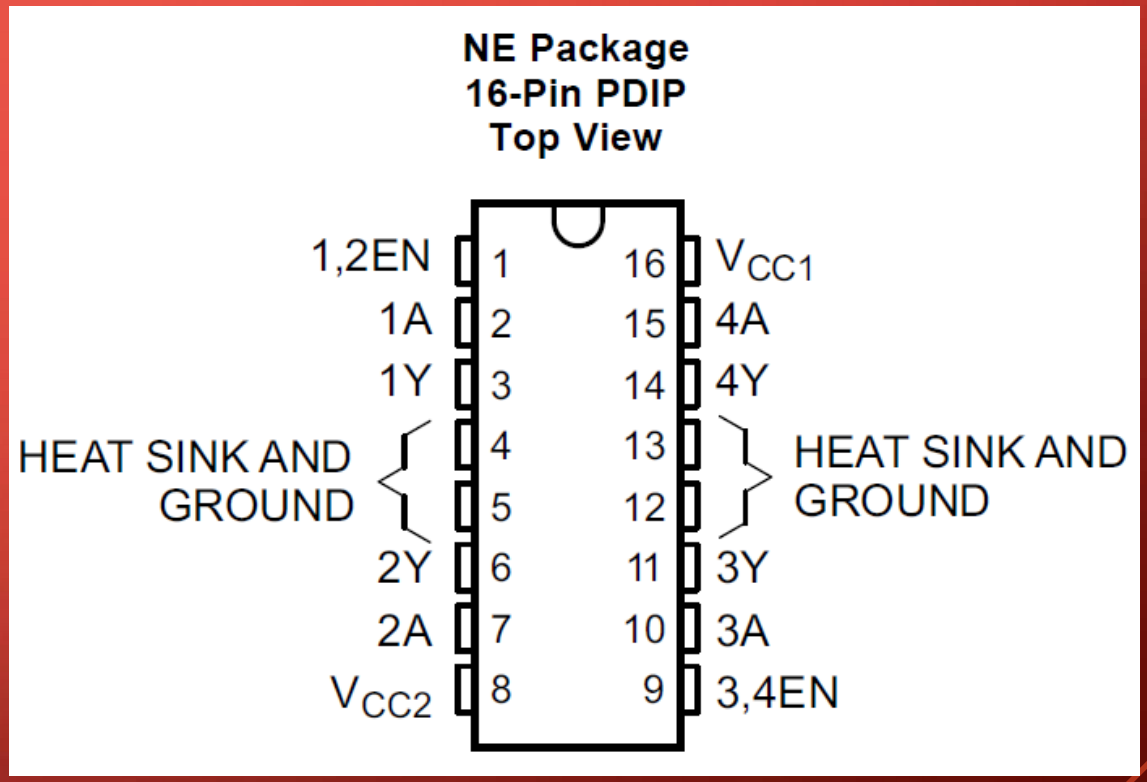
Pin Functions

PIN		TYPE	DESCRIPTION
NAME	NO.		
1,2EN	1	I	Enable driver channels 1 and 2 (active high input)
<1:4>A	2, 7, 10, 15	I	Driver inputs, noninverting
<1:4>Y	3, 6, 11, 14	O	Driver outputs
3,4EN	9	I	Enable driver channels 3 and 4 (active high input)
GROUND	4, 5, 12, 13	—	Device ground and heat sink pin. Connect to printed-circuit-board ground plane with multiple solid vias
V_{CC1}	16	—	5-V supply for internal logic translation
V_{CC2}	8	—	Power VCC for drivers 4.5 V to 36 V

AMIT

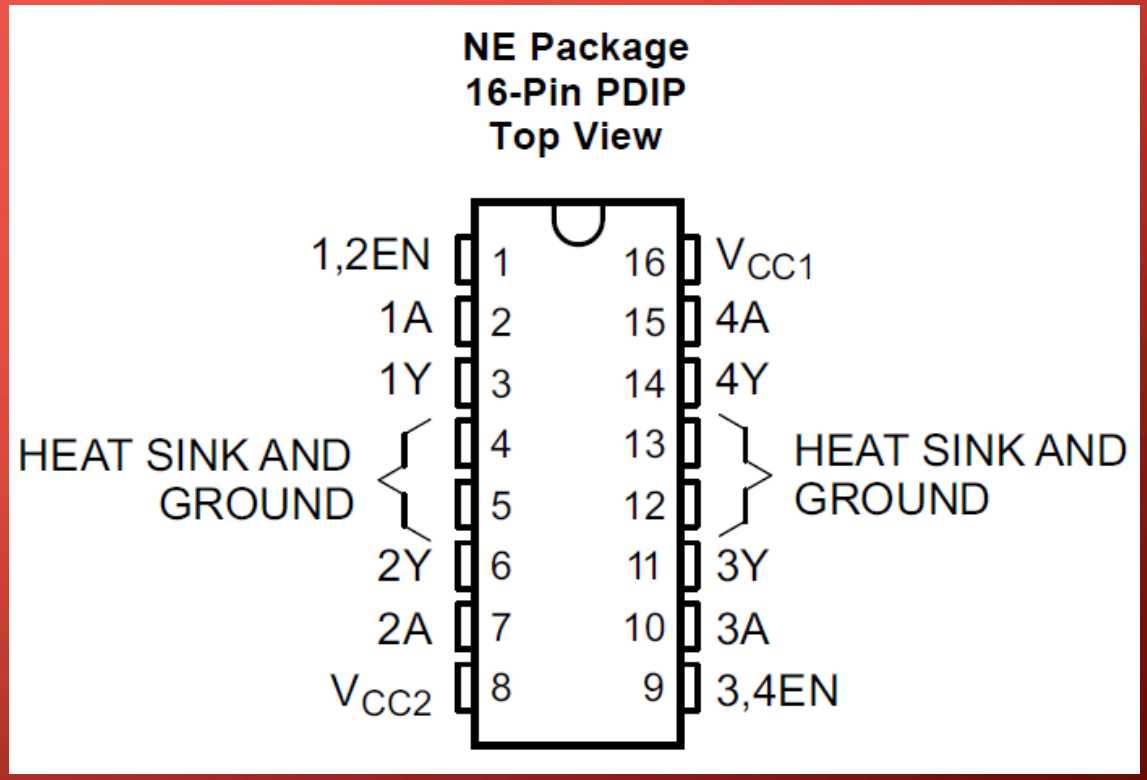
PIN CONFIGURATION OF L293D :

- Enable 1,2 & Enable 3,4 are used for enabling the activation of Directions pins, also they are used to control on the Motor speed if the signal on them became between 0→5 volts.
- 1A, 2A & A3, A4 are used for controlling the direction of motor, if 1A or A3 is High and 2A or A4 is Low, assume that the motor will rotate Clockwise, if the inverse, the motor will rotate Counterclockwise, if the same signals are applied on them, the motor will stop.



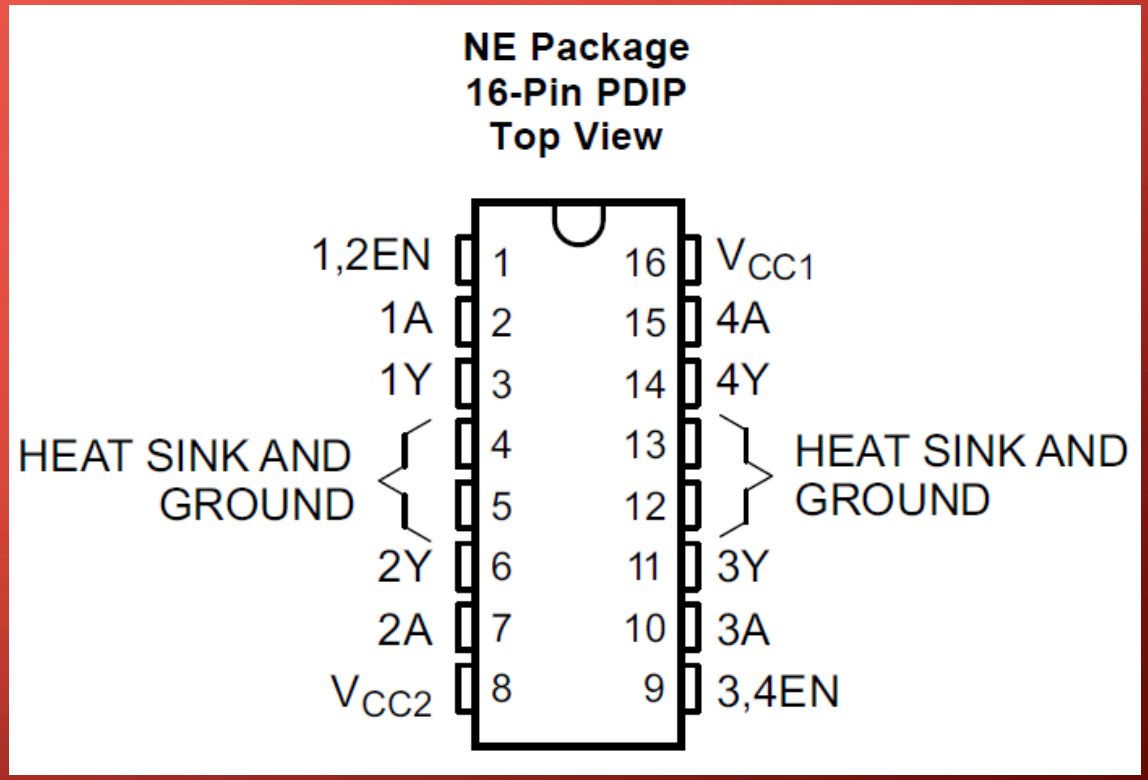
PIN CONFIGURATION OF L293D :

- Y1, Y2 & Y3, Y4 are used for connecting the polars of motors as output pins from the IC.
- Vcc2 is used to connect the volt of motors, this pin can be connected to up to 36 Volts.
- Vcc1 is used to feed the IC with power to manage to work, it is connected to 5 Volts.



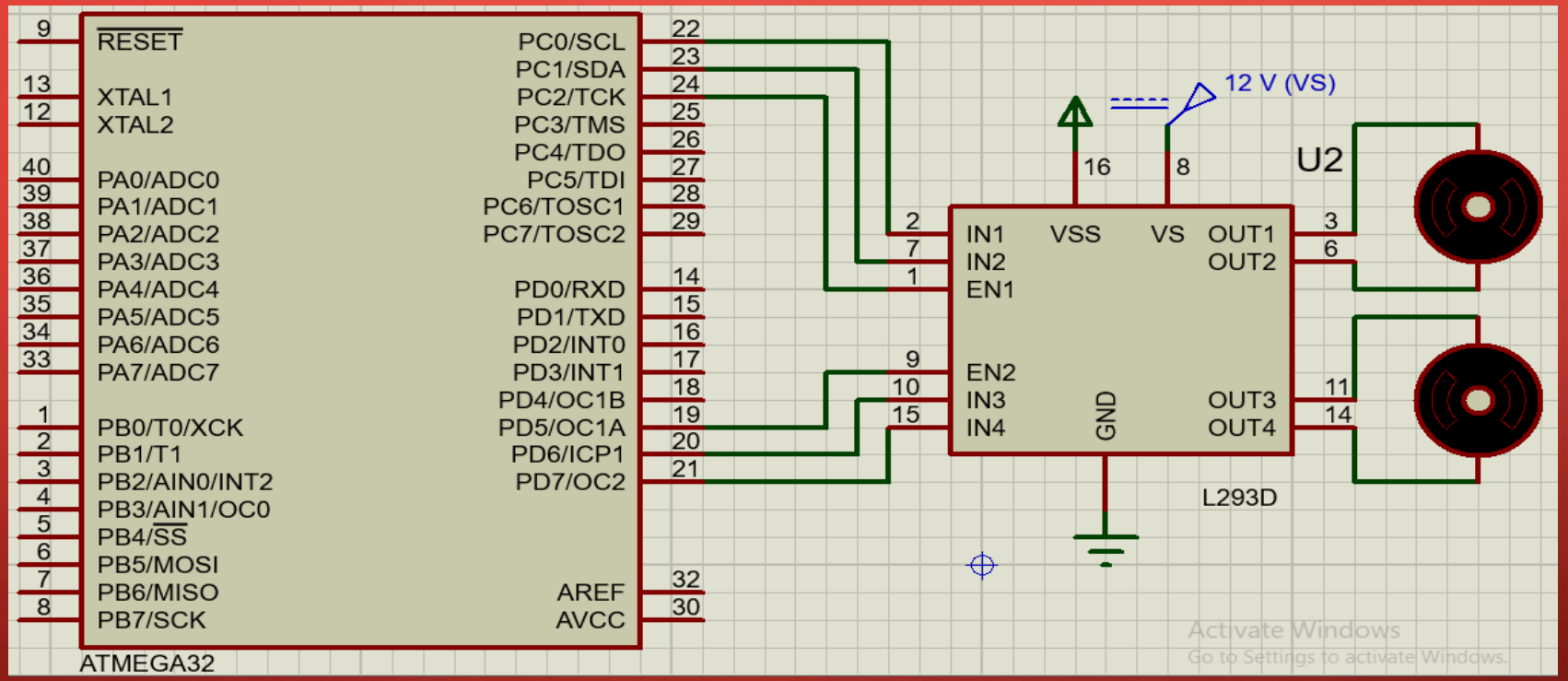
PIN CONFIGURATION OF L293D :

- Pins 3,4,13,14 are used for connecting to Ground.
- Also, these pins are used for heat transferring, I mean they are used to cool the IC circuit and throw the heat out of the IC to Air.



AMIT

PIN CONNECTION WITH ATMEGA32:



AMIT

SERVO MOTORS:

- **What is the Servo Motor?:**
  - A servo motor is a rotary actuator or a motor that allows for a precise control in terms of the angular position, acceleration, and velocity. Basically, it has certain capabilities that a regular motor does not have. Consequently, it makes use of a regular motor and pairs it with a sensor for position feedback.

AMIT

SERVO MOTORS:

- **Why the Servo Motor:**
  - The servo motor is most used for high technology devices in the industrial applications like automation technology. It is a self-contained electrical device, that rotates parts of machine with high efficiency and great precision. Moreover, the output shaft of this motor can be moved to a particular angle. Servo motors are mainly used in home electronics, toys, cars, airplanes and many more devices.
  - Thus, this blog discusses the definition, types, mechanism, principle, working, controlling, and lastly the applications of a servo machine.

AMIT

SERVO MOTORS:

- **Types of Servo Motors:**
  - Servo motors can be of different types based on their applications. The most important amongst them are AC servo motor, DC servo motor, brushless DC servo motor, positional rotation servo motor, continuous rotation servo motor, and linear servo motor.
  - A typical servo motor comprises of three wires namely- power, control, and ground. The shape and size of these motors depends on their applications.

AMIT

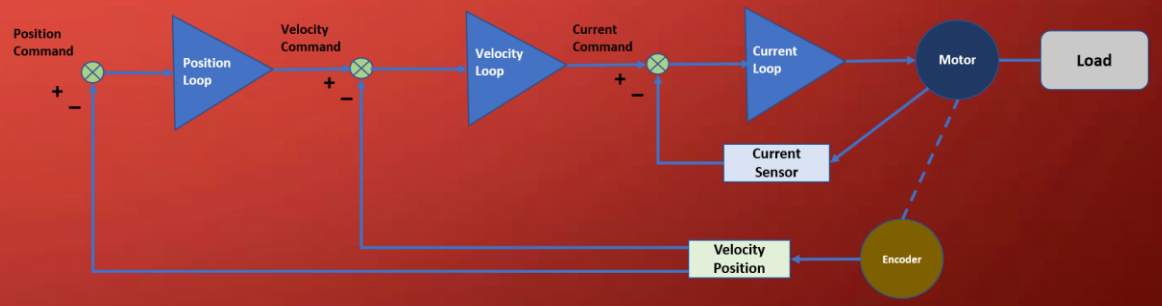
SERVO MOTORS:

- **Types of Servo Motors:**
  - Series motors.
  - Split series motor.
  - Shunt control motor.
  - Permanent magnet shunt motor.
  - Linear servo motor.
  - Continuous rotation servo motor.
  - Positional rotation servo motor.
  - Brushless DC servomotor.

AMIT

HOW SERVO MOTOR WORKS:

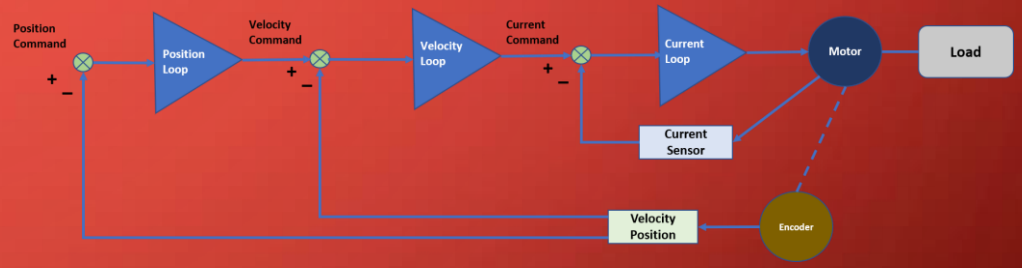
- **How Servo Motor works:**
  - A servo motor works as part of a closed loop system providing torque and velocity as commanded from a servo controller utilizing a feedback device to close the loop. The feedback device supplies information such as current, velocity, or position to the servo controller, which adjusts the motor action depending on the commanded parameters.



AMIT

HOW SERVO MOTOR WORKS:

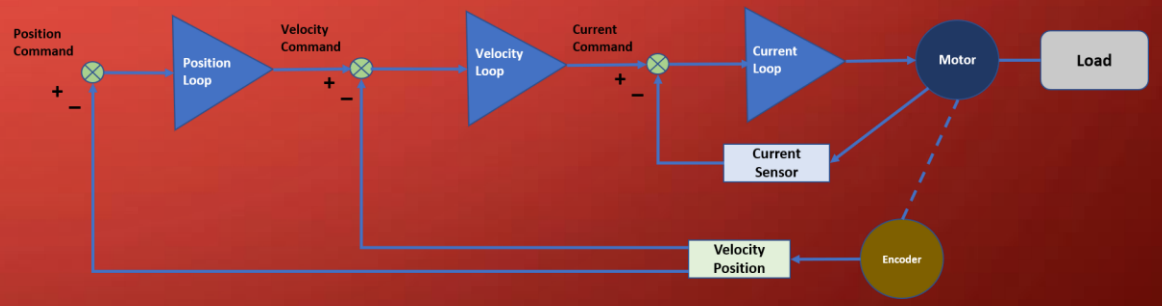
- **How Servo Motor works:**
  - A simple industrial servo motor consists of a permanent magnet DC motor with an integral tachometer that provides an output voltage proportional to speed. The drive electronics delivers the necessary voltage and current to the motor based on the voltage fed back from the tachometer. In this example, a commanded speed (represented as a command reference voltage) is set in the driver, then the circuitry in the driver compares the tachometer feedback voltage and determines if the desired speed has been accomplished - known as a closed velocity loop. The velocity loop is monitoring the commanded velocity and tachometer feedback, while the driver adjusts the power to the motor to maintain the desired commanded velocity.



AMIT

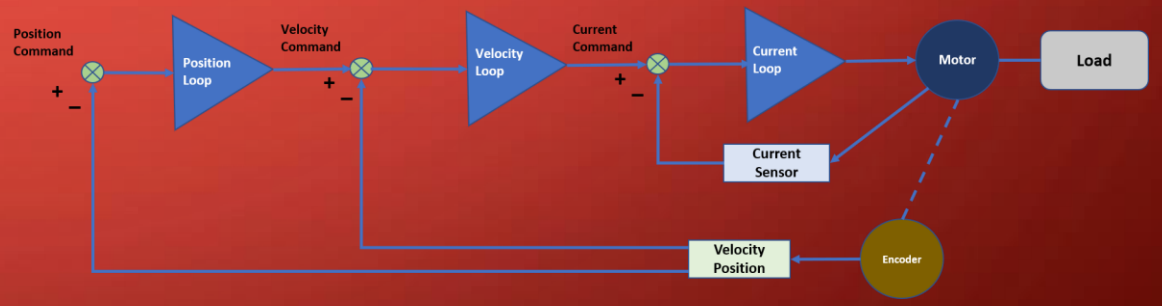
HOW SERVO MOTOR WORKS:

- **How Servo Motor works:**
  - A simple industrial servo motor consists of a permanent magnet DC motor with an integral tachometer that provides an output voltage proportional to speed. The drive electronics delivers the necessary voltage and current to the motor based on the voltage fed back from the tachometer. In this example, a commanded speed (represented as a command reference voltage) is set in the driver, then the circuitry in the driver compares the tachometer feedback voltage and determines if the desired speed has been accomplished - known as a closed velocity loop. The velocity loop is monitoring the commanded velocity and tachometer feedback, while the driver adjusts the power to the motor to maintain the desired commanded velocity.



HOW SERVO MOTOR WORKS:

- **<u>We can shorten the previous into four steps:</u>**
  - Pulse width to voltage converter send a voltage to the error Amp.
  - position sensor sense the motor shaft angle position and send it to the error amplifier.
  - error amp compares between these two signals and drive a voltage to the dc motor to correct its position.
  - Control pulse(PWM) sent to servo.



AMIT

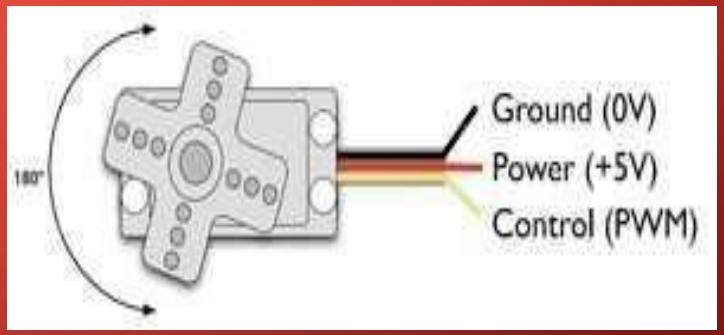
CONNECTION OF SERVO MOTOR:

Hardware Connection:

- Black Wire:
  - It is the Ground polar of the motor.
- Red Wire:
  - It is the Vcc polar of the motor.
- Orange Wire:
  - It is the Control Signal of the motor.

The most common servos works at 50 Hz frequency.

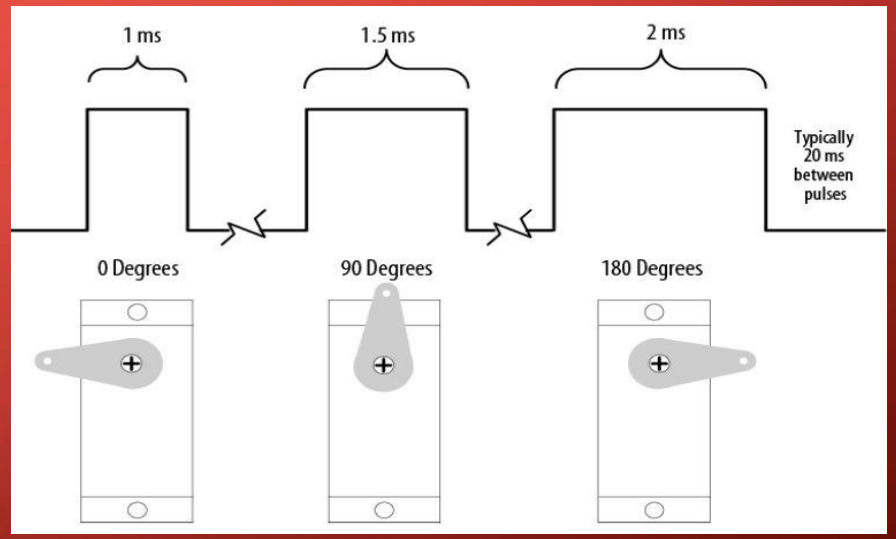
This means that the period is 20ms



AMIT

CONNECTION OF SERVO MOTOR:

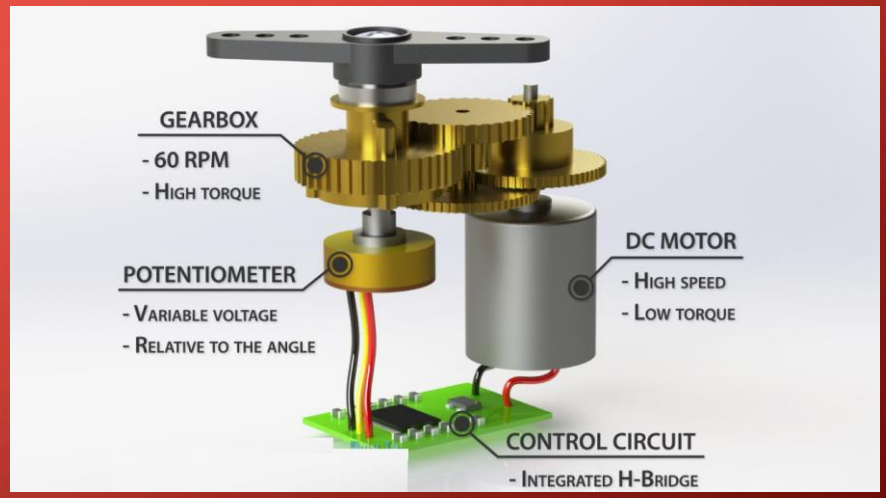
- **<u>The pulse width sent to servo ranges as follows:</u>**
  - Minimum: 1 millisecond ---> Corresponds to 0 rotation angle.
  - Maximum: 2 millisecond ---> Corresponds to 180 rotation angle.



AMIT

HOW TO CONTROL THE SERVO MOTOR:

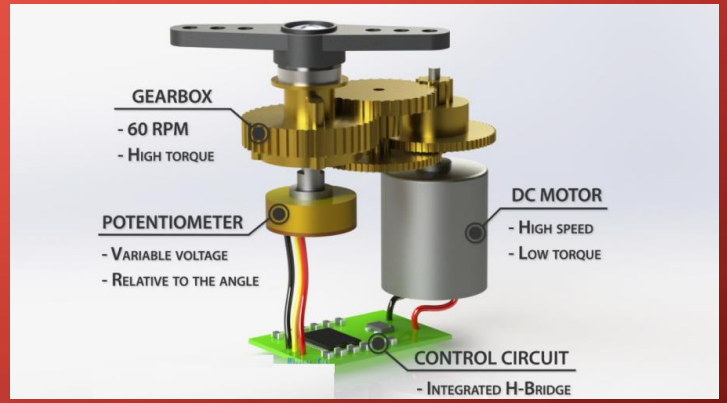
- **<u>There are three ways to control the servo motor:</u>**
  - **Software Delay:**
    - In this way you can generate a pulse on servo-signal pin to move it to 0-degree position by software delay as the following Sodo code:
      - Signal pin High;
      - delay 1 ms;
      - Signal pin Low;
      - delay 19 ms;



AMIT

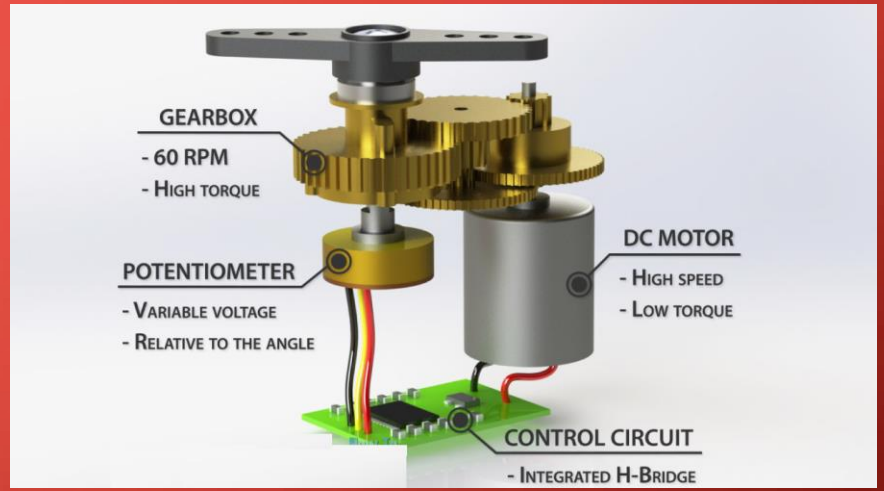
HOW TO CONTROL THE SERVO MOTOR:

- **<u>There are three ways to control the servo motor:</u>**
  - **Timer Interrupt:**
    - In this way you can generate a pulse on servo-signal pin to move it to 0-degree position by Timer interrupt as the following Sodo code:
      - Set timer interrupt after 1 ms;
      - Signal pin High at the 1st ISR;
      - At the next ISR Set Signal pin to LOW
      - Count another 18 ISR;
      - Repeat the cycle again.



AMIT

HOW TO CONTROL THE SERVO MOTOR:



- **<u>There are three ways to control the servo motor:</u>**
  - **PWM:**
    - In this way you can generate a pulse on servo-signal pin to move it to 0-degree position by PWM Mode on Timer 1 as the following Sodo code:
      - Set timer 1 at any PWM mode which its top is ICR or OCR1A;
      - Set your prescaler and calculate the time period to generate 50 Hz PWM;
      - Set your duty cycle 5% ;
      - Connect the Signal pin to the PWM channel you ran it.

AMIT

THANK YOU!