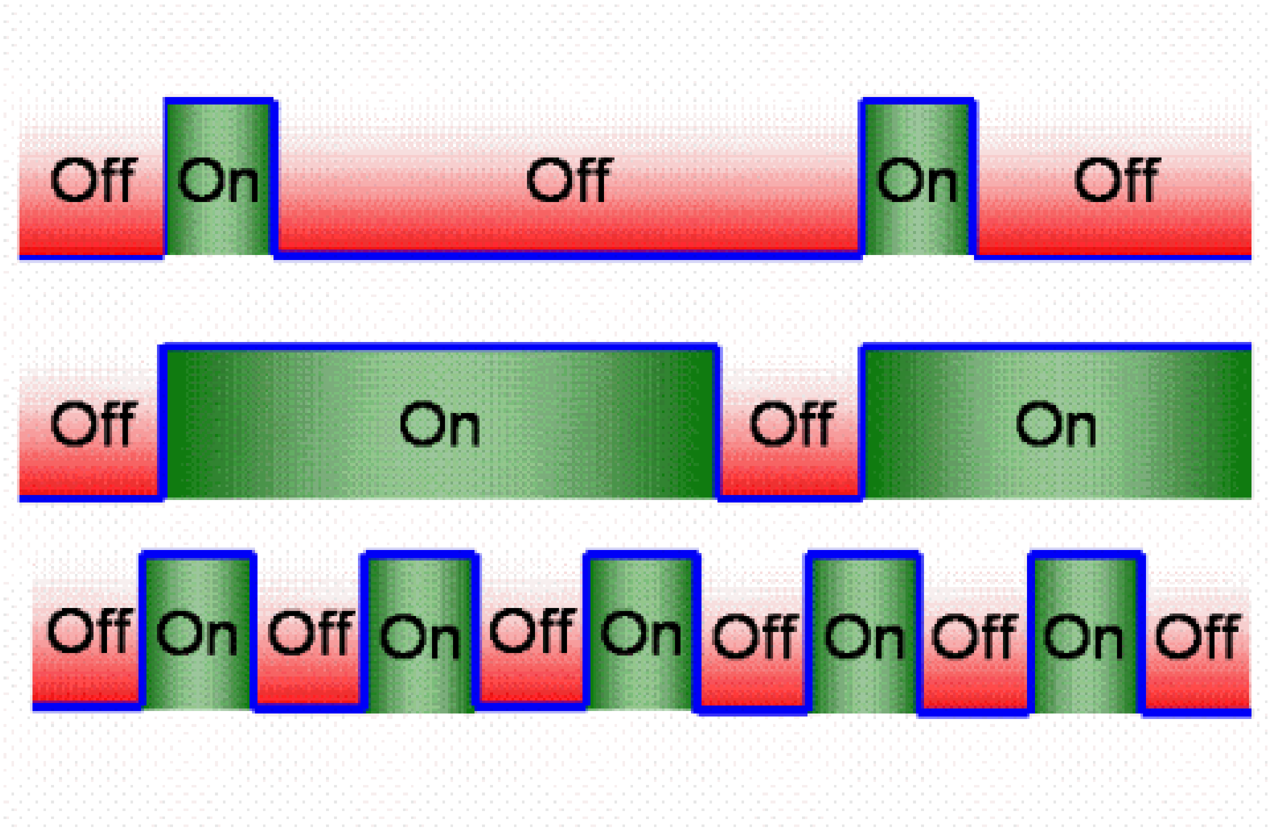
PWM IN AVR

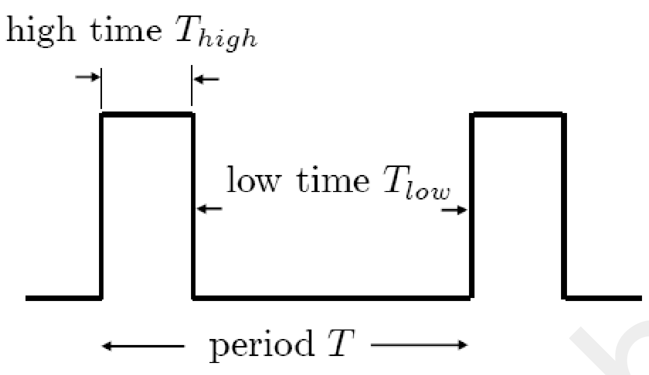


Developed by:

Krishna Nand Gupta
Prashant Agrawal
Mayur Agarwal

PWM (pulse width Modulation)

What is PWM?



A PWM signal is a periodic rectangular pulse :

Frequency = (1/T)
Duty Cycle = (T_{high}/T)

What is need of PWM?

I answer this in respect to our ROBOT here you are having 300RPM motor and voltage rating is 12 Volt so if you want to run motor to 150RPM then what will you do?? Reduce voltage to 6 volt but that's not good idea RPM is not linear function of voltage and as voltage decrease from specified rating Torque will also get reduce. So here come PWM concept what you can do is you can switch on and switch off the motor repetitively such that effectively you get 50% **ON** time if you do it reasonably fast then motor seems running continuous.

How PWM is implemented in AVR?

What do you exactly need? A constant time period square wave whose duty cycle we can control in code, right?

Clock period of a microcontroller is:

Tclk =1/(clock frequency)

E.g. In ATmega 8 default clock frequency is near about 1 Mega Hz.

$$T_{system\text{-}clk}=1/(1M\ Hz)=1\ \mu s$$

To get desire frequency you need to change clocking of timer that is done by prescaler so

$$\mathbf{T_{clk}} = \text{prescaler} * T_{system\text{-}clk}$$

To implement PWM these variables are required:

- ❖ **TOP=** Number of clock cycles for one time period of PWM

  Nt=T/Tclk

- ❖ **OCR** = Number of clock cycles for On Time of PWM

  OCR (output compare register)

  OCR= Ton/Tclk

- ❖ **Timer_value**= It is value of timer, that counts from **TOP** to zero and Zero to **TOP** in each cycle.

  ****** To control DC motor need **dual slope phase correct PWM** this complicated term only mean that counter will count from **TOP** to zero and Zero to **TOP** in time period. So time period of

Time period of the square wave is decided by 2 variables

- T_{clk}
- TOP

$$\mathbf{T= T_{clk}*TOP}$$
or
$$\mathbf{T= T_{clk}*TOP}\ *2 \quad \text{(dual slope phase correct PWM)}$$

We adjust both variables to get desired time period by changing some control bits in control registers.

Here in this tutorial we will set **TOP=255**

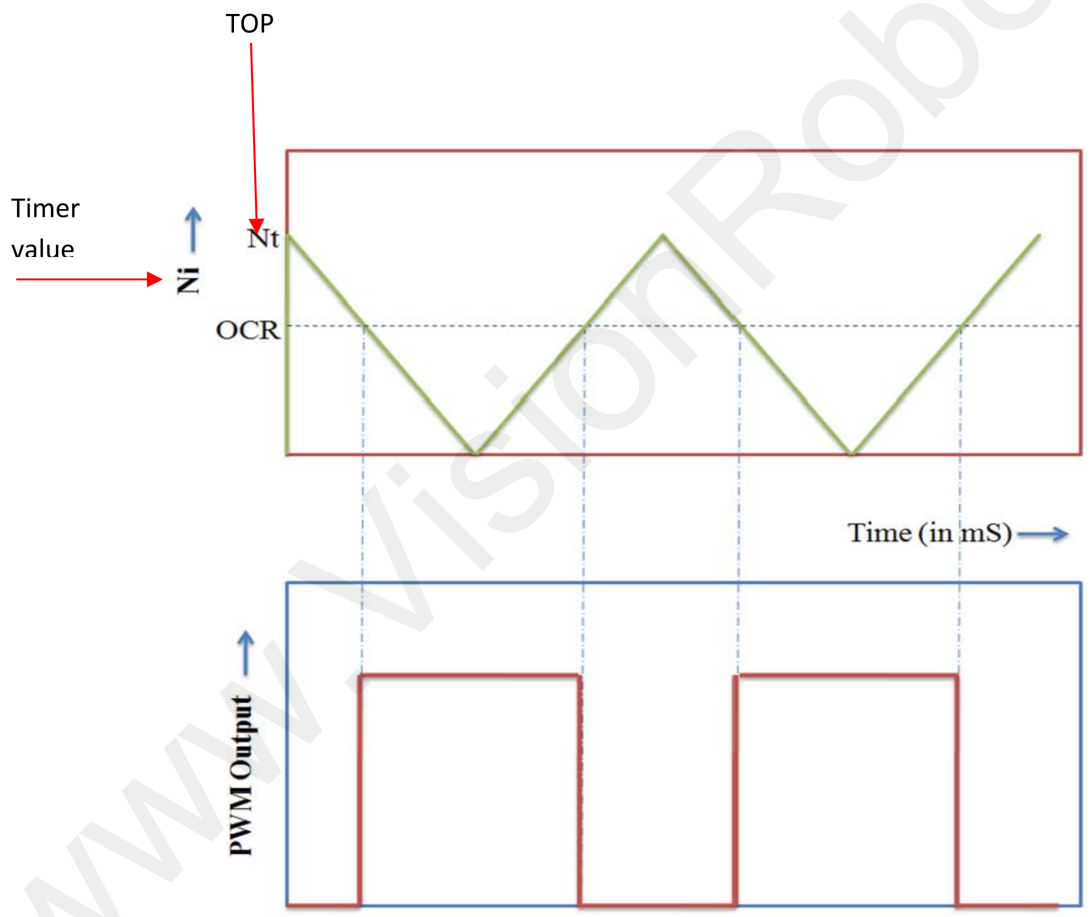
Duty cycle is decided by only one variable OCR

Duty cycle = (OCR/TOP)

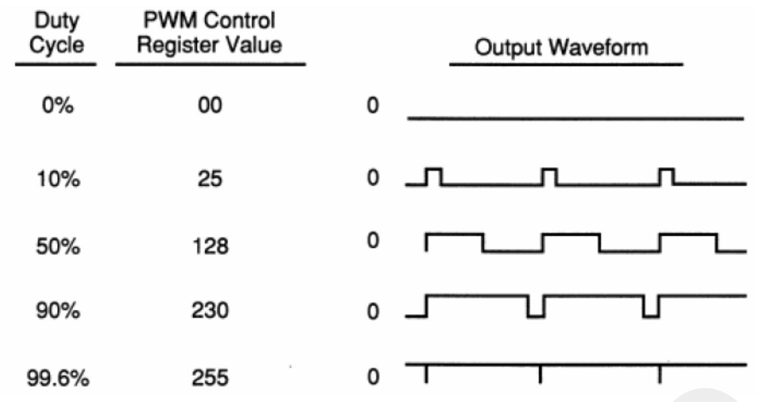
Duty cycle = (OCR/TOP)

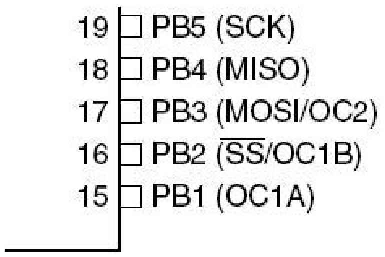
When if Timer_value ≥ OCR then PWM Output=Low

And if Timer_value < OCR then PWM Output=High



PWM module: specifying the duty cycle





You must be wondering where where will this squre wave will appear. In avr microcontroller there is certain PWM pins where these wave will apear. Like in ATmega8 PIN 15,16 and 17. If you want to control only 2 DC motor then 15 and 16 pins are sufficient.

Now suppose you want to generate a square wave that has 50% duty cycle at pin 15(OC1A) Then how your code will look like

```
Int main()
{
        Step 1  Set PB1 (OC1A) as output pin;
        Step 2  Enable PWM;
        Step 3  Select Phase correct PWM mode and TOP value;
        Step 4  Set OCR value TOP/2;
        Step 5  Set prescaler value and clock source ;
        Step 6  Start PWM;
While(1)
        {
                //do any job here
        }
        Return 0;
}
```

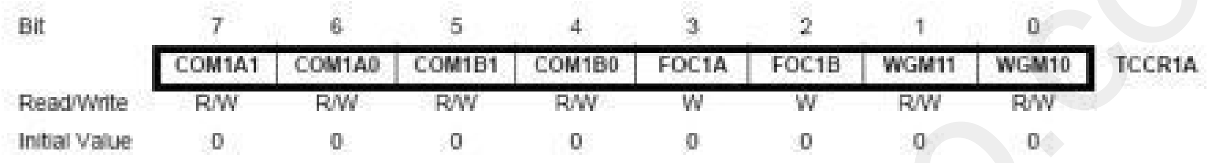
Now if you connect Oscilloscope at pin 15 you can see a square wave with 50% duty cycle ☺

You have learnt PWM just crazy syntax part of AVR PWM setting is left.

NOTE: for the sake of simplicity of this tutorial here I will consider only phase correct PWM to learn more about Timer and PWM go through datasheet of ATmega8

16-bit Timer/Counter Register Description

Timer/Counter 1 Control Register A – TCCR1A

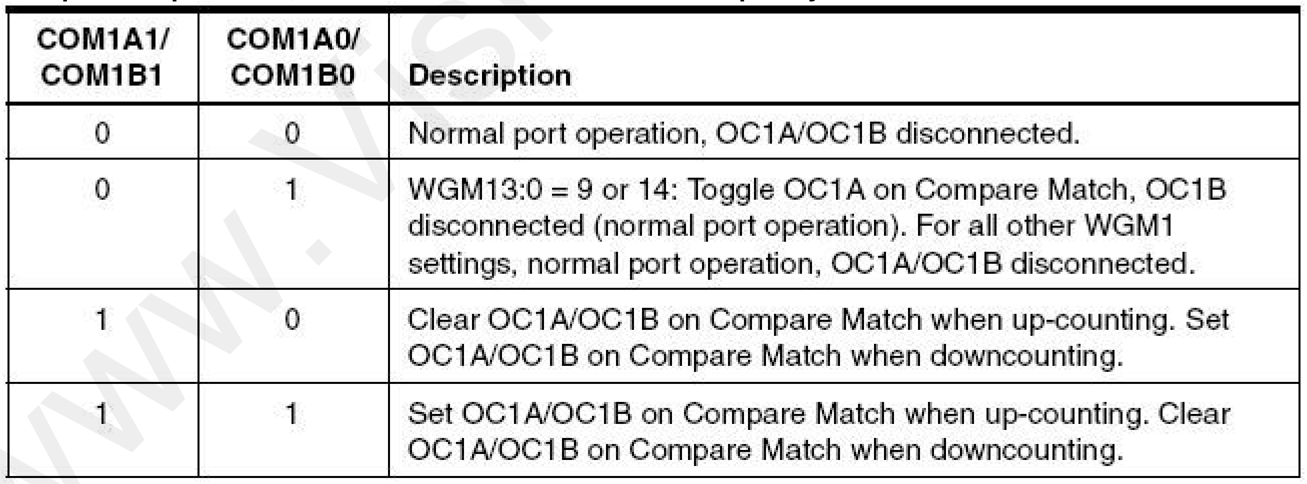


Bit	7	6	5	4	3	2	1	0	
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	W	W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

• Bit 7:6 – COM1A1:0: Compare Output Mode for channel A
• Bit 5:4 – COM1B1:0: Compare Output Mode for channel B

The COM1A1:0 and COM1B1:0 control the Output Compare Pins (OC1A and OC1B (respectively) behavior. If one or both of the COM1A1:0 bits are written to one, the OC1A output overrides the normal port functionality of the I/O pin it is connected. Similarly for COM1B1:0. However, note that the *Data Direction Register* (DDR) bit corresponding to the OC1A or OC1B pin must be written **1** in order to enable the output driver.

Compare Output Mode, Phase Correct and Phase and Frequency Correct PWM



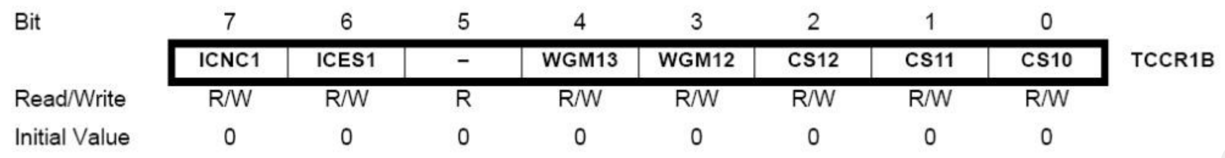
COM1A1/ COM1B1	COM1A0/ COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 9 or 14: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match when up-counting. Set OC1A/OC1B on Compare Match when downcounting.
1	1	Set OC1A/OC1B on Compare Match when up-counting. Clear OC1A/OC1B on Compare Match when downcounting.

• Bit 3 – FOC1A: Force Output Compare for channel A
• Bit 2 – FOC1B: Force Output Compare for channel B

These bits are not used in phase correct PWM set write there bit 0;

Functionality of WGM11 and WGM10 will be given later

Timer/Counter 1 Control Register B – TCCR1B



Bit	7	6	5	4	3	2	1	0	
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

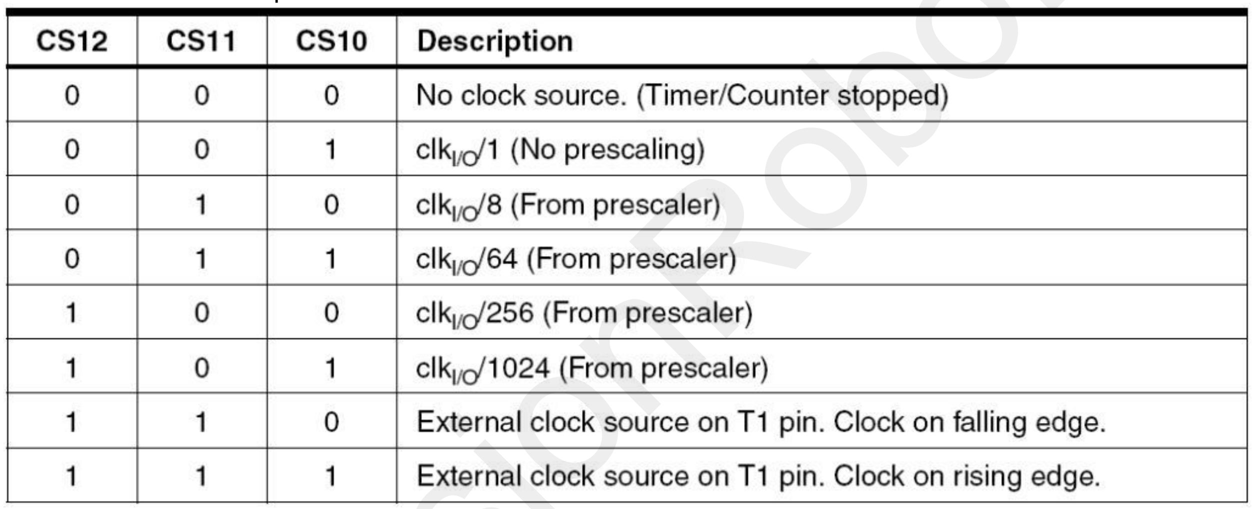
• Bit 7, BIT 6, BIT5

These bits is not used in PWM simply write these bit 0;

• Bit 2:0 – CS12:0: Clock Select

The three clock select bits select the clock source to be used by the Timer/Counter,

Clock Select Bit Description



CS12	CS11	CS10	Description
0	0	0	No clock source. (Timer/Counter stopped)
0	0	1	$clk_{I/O}$/1 (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 T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

Default No clock source is selected. as you select clock PWM will start

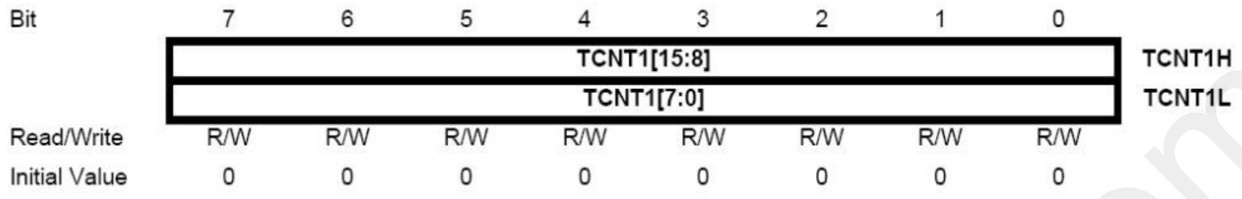
• Bit WGM13, WGM12 in TCCR1B and WGM11, WGM10 in TCCR1A

WGM stand for Wave generation mode. There are several PWM and time mode here we want to do setting for phase correct PWM for **TOP**=255 and for that you need to write

WGM13	WGM12	WGM11	WGM10
0	0	0	1

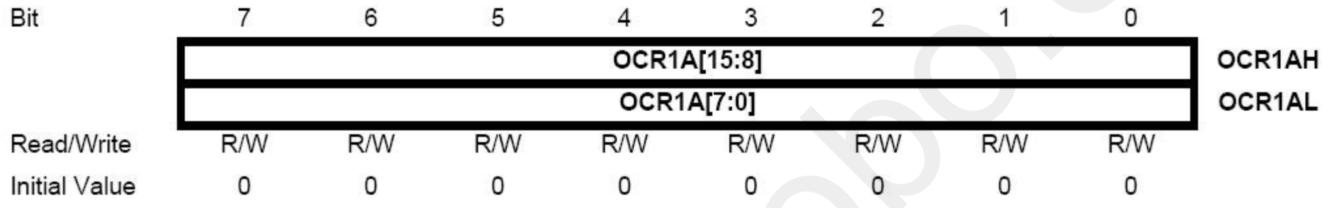
*****For rest of the mode check ATmega8 data sheet

Timer/Counter 1 – TCNT1H and TCNT1L



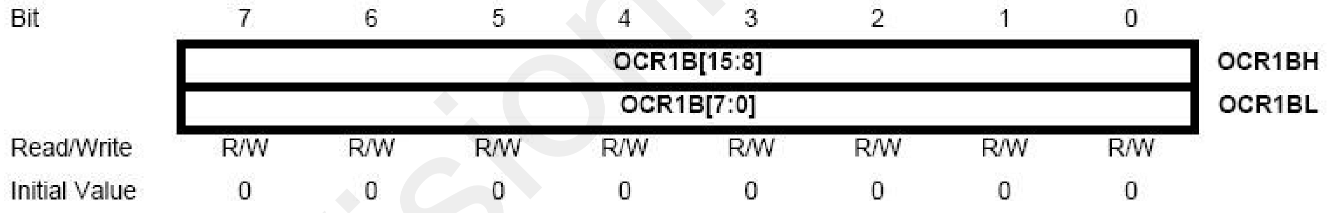
Bit	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Output Compare Register 1 A – OCR1AH and OCR1AL



Bit	7	6	5	4	3	2	1	0	
	OCR1A[15:8]								OCR1AH
	OCR1A[7:0]								OCR1AL
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	

Output Compare Register 1 B– OCR1BH and OCR1BL



Bit	7	6	5	4	3	2	1	0	
	OCR1B[15:8]								OCR1BH
	OCR1B[7:0]								OCR1BL
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	

Now we return to our code

```
#include <avr/io.h>
Int main()
{
        Step 1  Set PB1 (OC1A) as output pin;
        Step 2  Enable PWM;
        Step 3  Select Phase correct PWM mode and TOP value;
        Step 4  Set OCR value TOP/2;
        Step 5  Set prescaler value and clock source ;
        Step 6        Start PWM;
While(1)
        {
                //do any job here
        }
        Return 0;
}
```

Step 1 Set PB1 **(OC1A)** as output pin;

For detail check i/o tutorial

Code:

```
DDRB|=(1<<PORTB1);
```

Step 2 Enable PWM;

For this you need to write COM1A10=0 and COM1A11=1

Code:

```
TCCR1A=0xA1;
```

Step 3 Select **Phase correct PWM** mode and **TOP** value;

Here we will go for **TOP**=255

WGM13	WGM12	WGM11	WGM10
0	0	0	1

Step 4 Set OCR value TOP/2;

Code:

```
OCR1A=128;
```

Step 5 Set **prescaler** value and clock source ;

Here we will set prescaler 1024 for that we need to write CS12=1, CS11=0, CS10=1

****For detail see Clock Select Bit

Code:

```
TCCR1B=0x05;
```

Step 6 Starts PWM;

Step 5 will do this job

```
#include <avr/io.h>
int main()
{
	DDRB|=(1<<PORTB1);
	TCCR1A=0xA1;
	OCR1A=128;
	TCCR1B=0x05;

	While(1)
		{
			//any job
		}
	Return 0;
}
```

You can also check PWM with multimeter here you will se voltage changing in multimeter but if set square wave time period sufficient low . you will see average value in multimeter

HAPPY PWM CODING ☺