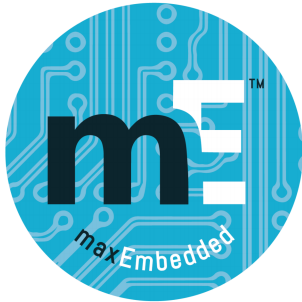


[maxEmbedded Index](#)[Categories »](#)[Tools](#)

Search This Site...



a guide

Embedded

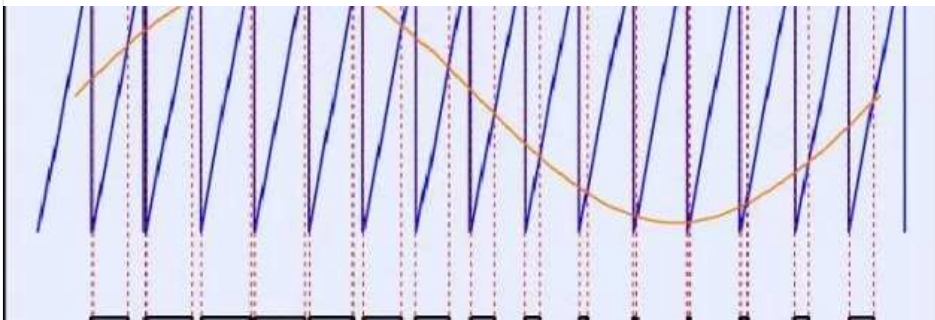
embedded

computer vision

[Home](#)[mE Index](#)[Getting Started](#)[Atmel AVR »](#)[Electronics](#)[More »](#)[About »](#)[Home](#)[Atmel AVR](#)

AVR Timers – PWM Mode – Part I

Posted by [Mayank](#) on Aug 7, 2011 in [Atmel AVR](#), [Microcontrollers](#) | [51 comments](#)



AVR Timers – PWM Mode – Part I

Pulse Width Modulation (PWM) is a very common technique in telecommunication and power control. Learn how easily you can do so using AVR! This post discusses all the necessary theoretical concepts related to PWM. Here it goes...



Welcome back! Till now, in the AVR Timers, we have discussed regarding the timer concepts, prescalers, interrupts, ctc mode, etc. Refer to the following tutorials for them.

- [Introduction to AVR Timers](#)
- [AVR TIMER0](#)
- [AVR TIMER1](#)
- [AVR TIMER2](#)

Search maxEmbedded

Search for:

Search

Popular

Recent

Random

MON
20[The ADC of the AVR](#)

Posted by Mayank in Atmel AVR,
Microcontrollers

FRI
24[AVR Timers – TIMER0](#)

Posted by Mayank in Atmel AVR,
Microcontrollers

TUE
06[RF Module Interfacing without Microcontrollers](#)

Posted by Mayank in Electronics

THU
16[LCD Interfacing with AVR](#)

Posted by Mayank in Atmel AVR,
Microcontrollers

THU
20[Making an RF Car](#)

Posted by Yash in Electronics, Robotics

Browse maxE by Categories

Browse maxE by Categories

Select Category

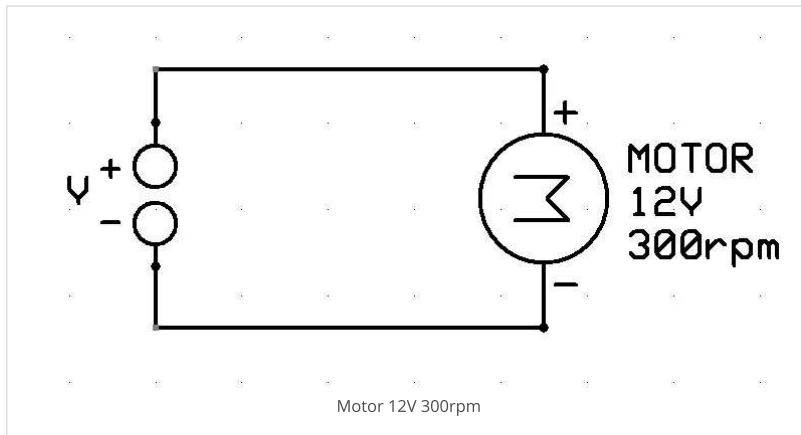


Subscribe to maxEmbedded

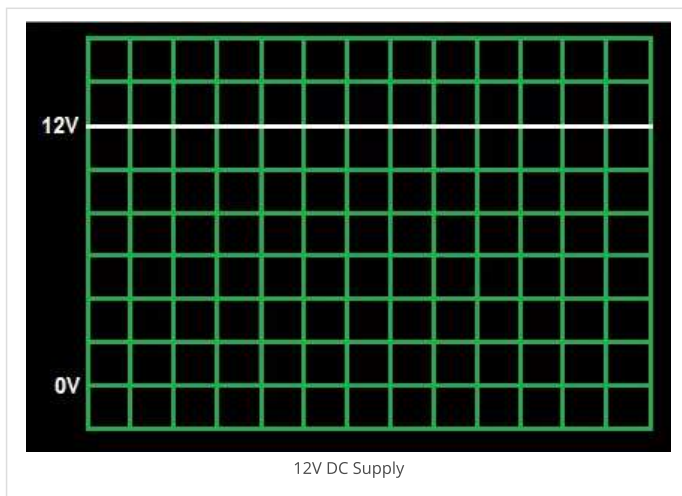
In this tutorial, I will introduce you to another mode of AVR Timers – PWM Mode.

Introduction

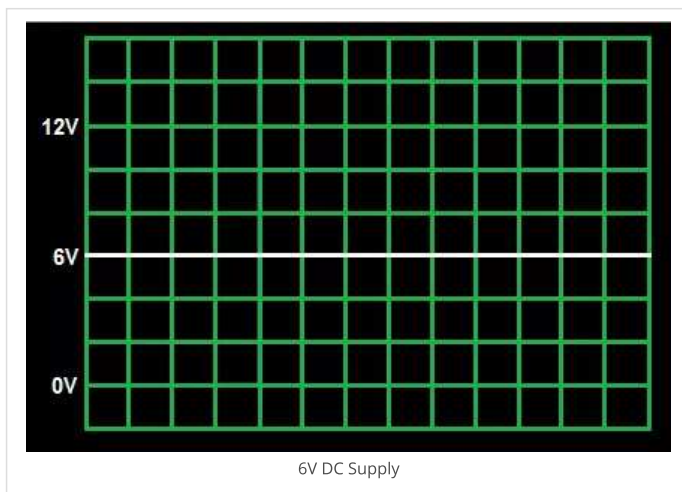
Let us assume that a DC motor is connected to the power supply as follows.



The motor is rated 12V/300rpm. This means that (assuming ideal conditions) the motor will run at 300 rpm only when 12V DC is supplied to it. If we apply 6V, the motor will run at only 150 rpm. For more details regarding controlling DC motor using AVR, view [this](#). Now let us provide the following signal (12V DC) to the motor.



The motor will rotate at 300 rpm. Now let us change the voltage level as follows (6V DC).



Email Address

Subscribe

[Like maxE on Facebook](#)

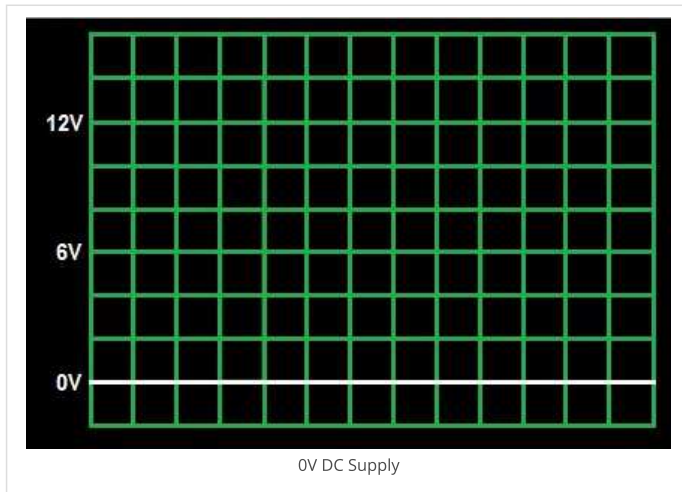


MaxEmbe...
4.4K likes

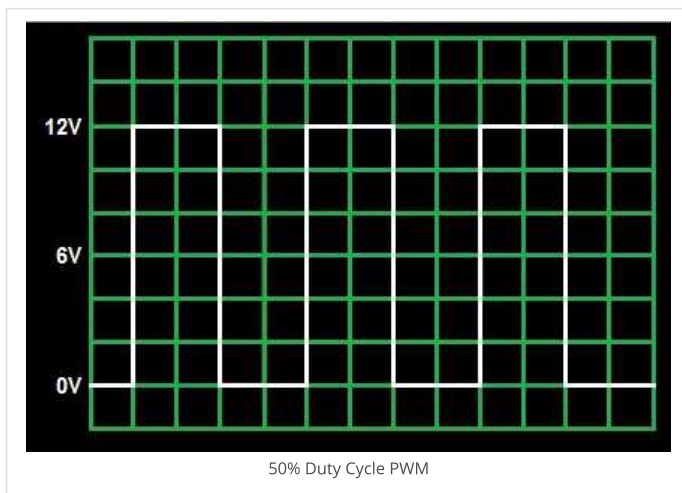


Sponsored Links

level once again as follows (0V DC).



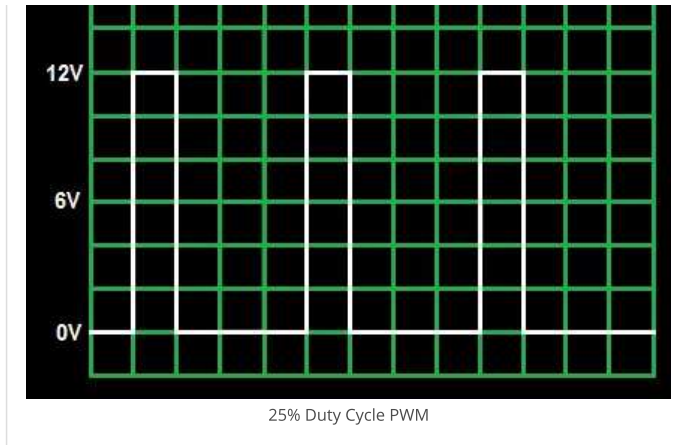
This time, unsurprisingly, the motor doesn't run at all. Okay, so let's make it more interesting. *What if* we provide the following supply to the motor.



Now how will the motor respond? Will it start for some time, then after some time it stops, then starts, then stops, and then starts again, and so on. Or will it get confused and simply blast? :D

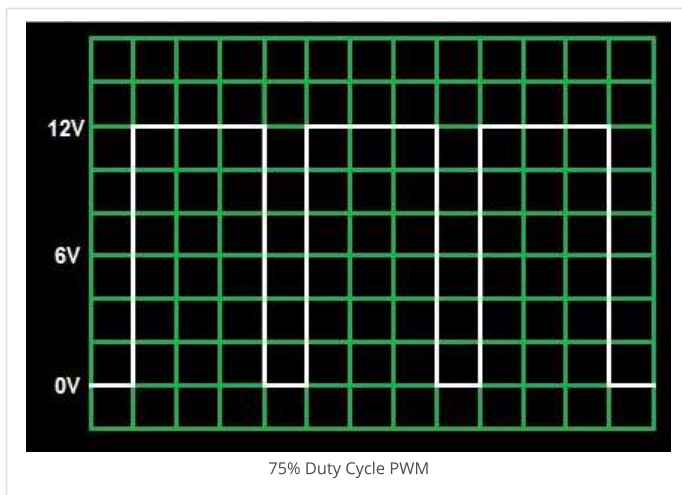
Well, each and every body in this world has some inertia. Say the motor above rotates whenever it is powered on. As soon as it is powered off, it will *tend* to stop. But it doesn't stop immediately, it takes some time. But before it stops completely, it is powered on again! Thus it starts to move. But even now, it takes some time to reach its full speed. But before it happens, it is powered off, and so on. Thus, the overall effect of this action is that the motor rotates continuously, but at a lower speed. In the above case, the motor will behave exactly as if a 6V DC is supplied to it, i.e. rotate at 150 rpm!

Okay, so now, let's modify the signal as follows.



Now what happens? Yes! You guessed it right! (I hope so ;)) Since the on-time is less than the off-time, the effective speed of the motor reduce. In this case, the speed becomes 75 rpm (since off-time = 3 times on-time, i.e. speed = $300/4 = 75$ rpm).

Now it's your turn to say what happens in this case:



This is what we call **Pulse Width Modulation**, commonly known as **PWM**.

PWM – Pulse Width Modulation

PWM stands for **Pulse Width Modulation**. It is basically a **modulation** technique, in which the width of the carrier pulse is varied in accordance with the analog message signal. As described above, it is commonly used to control the power fed to an electrical device, whether it is a motor, an LED, speakers, etc.

PWM Generation

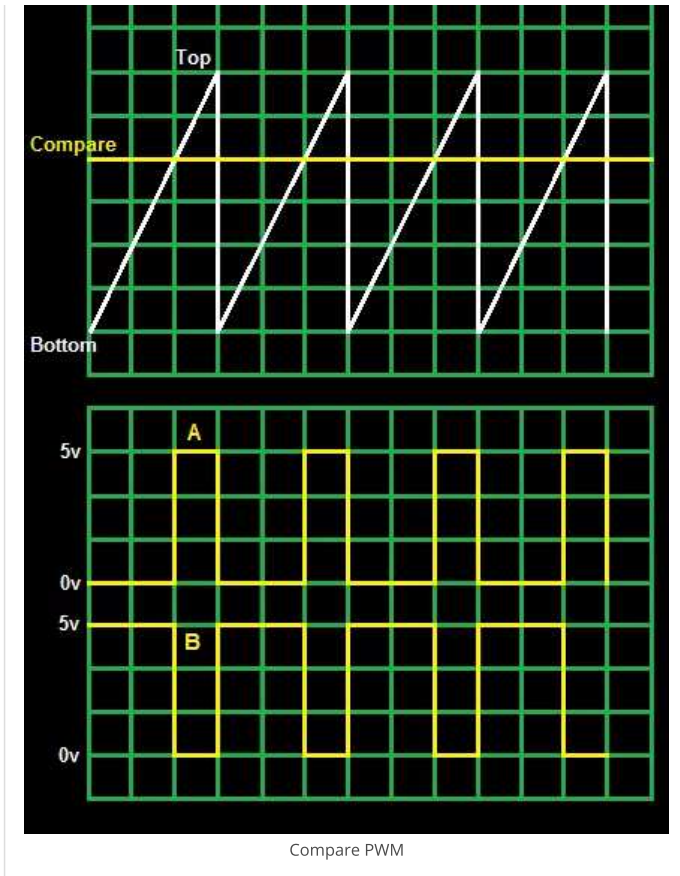
The simplest way to generate a PWM signal is by comparing the a predetermined waveform with a fixed voltage level as shown below.



58
Shares

52

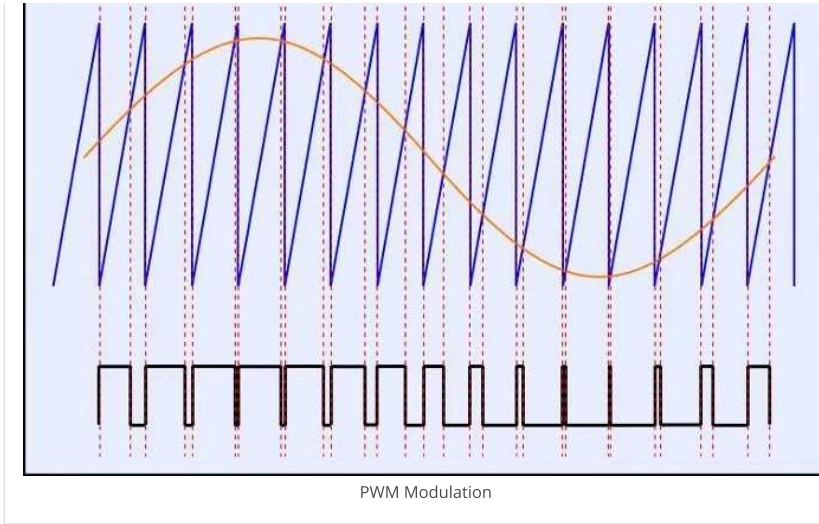
52



In the diagram shown above, we have a predetermined waveform, sawtooth waveform. We *compare* this waveform with a fixed DC level. It has three **compare output modes** of operation:

- **Inverted Mode** – In this mode, if the waveform value is greater than the compare level, then the output is set high, or else the output is low. This is represented in figure A above.
- **Non-Inverted Mode** – In this mode, the output is high whenever the compare level is greater than the waveform level and low otherwise. This is represented in figure B above.
- **Toggle Mode** – In this mode, the output toggles whenever there is a compare match. If the output is high, it becomes low, and vice-versa.

But it's always not necessary that we have a fixed compare level. Those who have had exposure in the field of analog/digital communication must have come across cases where a sawtooth carrier wave is compared with a sinusoidal message signal as shown below.



Here you can *clearly* see and understand the meaning of 'width' in Pulse *Width* Modulation! ;-)

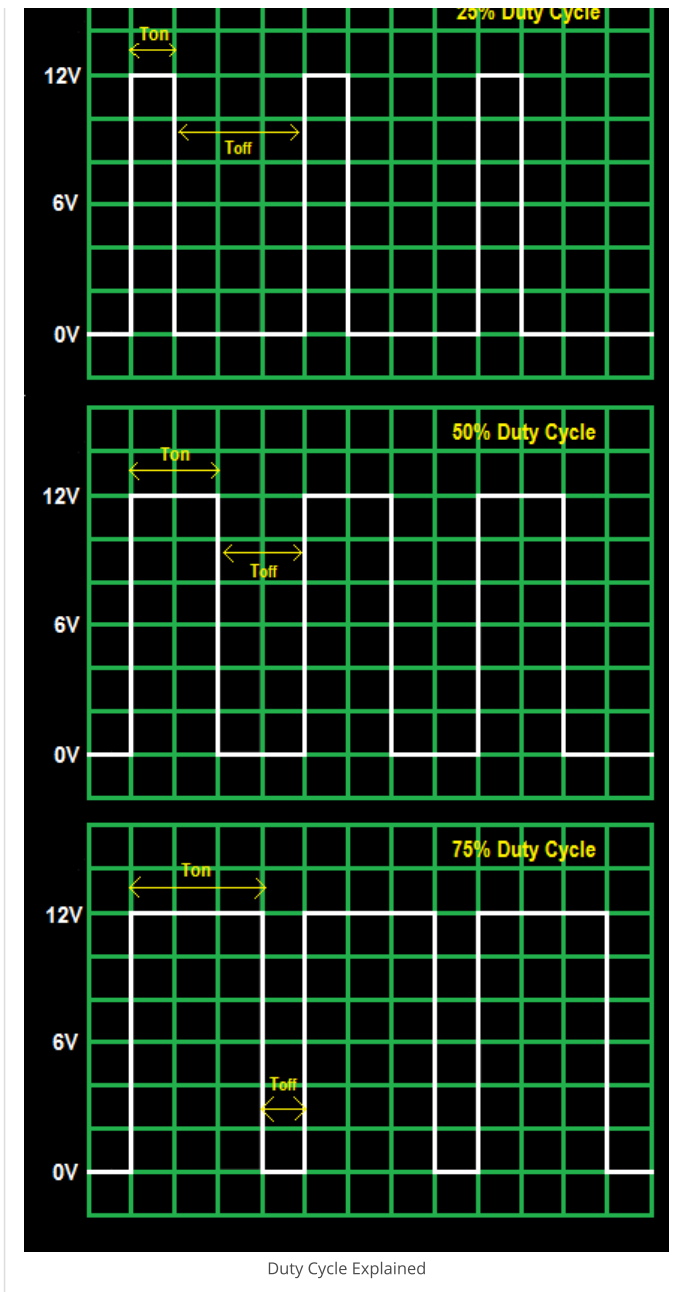
PWM can also be generated by making analog circuits like the one described [here](#).

Duty Cycle

The Duty Cycle of a PWM Waveform is given by

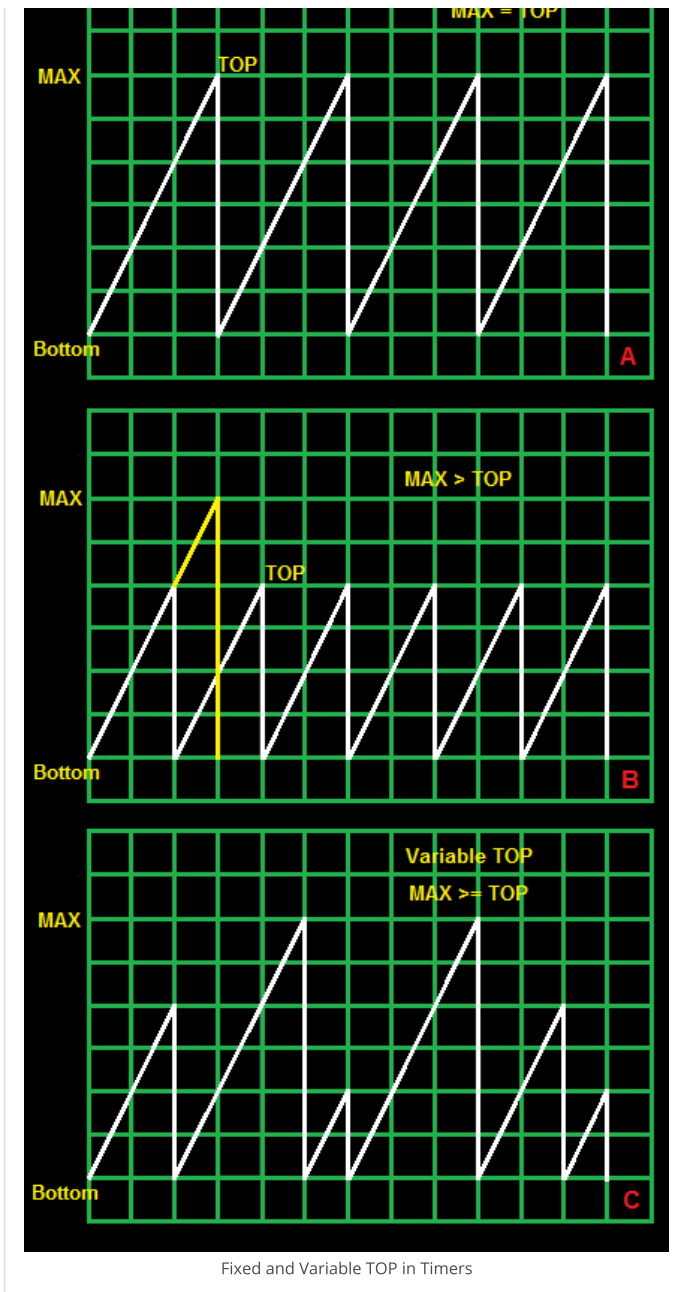
$$\text{Duty Cycle} = \frac{T_{on}}{T_{on} + T_{off}} \times 100 \%$$

This is clarified in the following diagram.



Timer Concepts – Revisited

In this section, we will revise some important and necessary concepts related to [timers](#). Consider the following timer diagram.



We are very well aware that the AVR provides us with an option of 8 and 16 bit timers. 8bit timers count from 0 to 255, then back to zero and so on. 16bit timers count from 0 to 65535, then back to zero. Thus for a 8bit timer, $MAX = 255$ and for a 16bit timer, $MAX = 65535$.

The timer *always* counts from 0 to TOP, then overflows back to zero. In figure A shown above, $TOP = MAX$. Now, I guess you all are familiar with timers in **CTC Mode**, in which you can clear the timer whenever a compare match occurs. Due to this, the value of TOP can be reduced as shown in figure B. The yellow line shows how the timer would have gone in normal mode. Now, the **CTC Mode** can be extended to introduce variable TOP as shown in figure C (however there isn't any practical utility of this).

TOP never exceeds MAX. $TOP \leq MAX$.

Now that you are aware of the terminologies of TOP, BOTTOM and MAX, we can proceed to the different modes of operation.

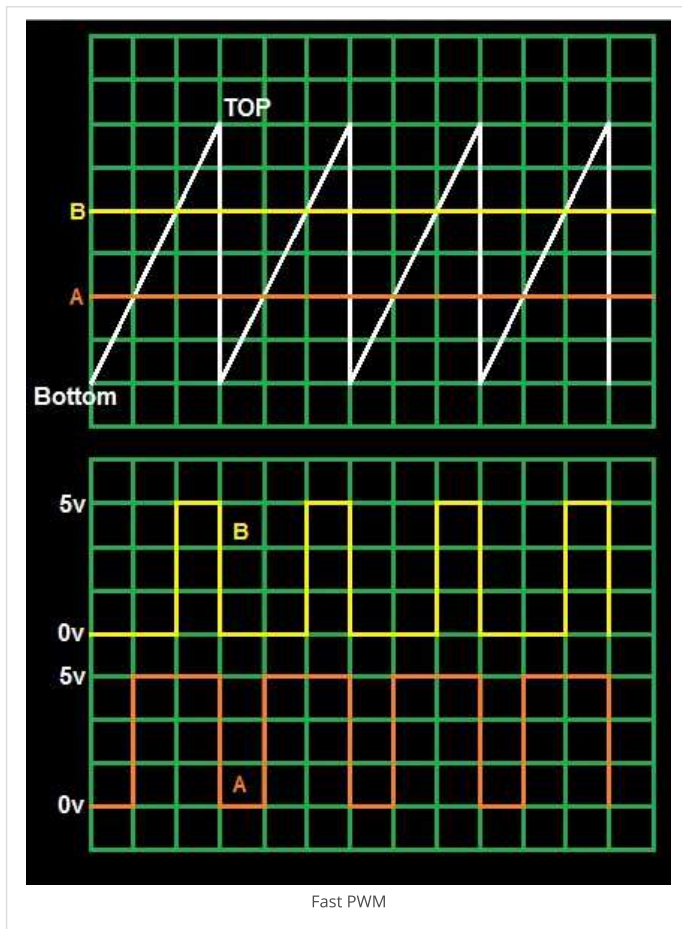
PWM Modes of Operation

In general, there are three modes of operation of PWM Timers:

- Phase Correct PWM
- Frequency and Phase Correct PWM

Fast PWM

Consider the following diagram.

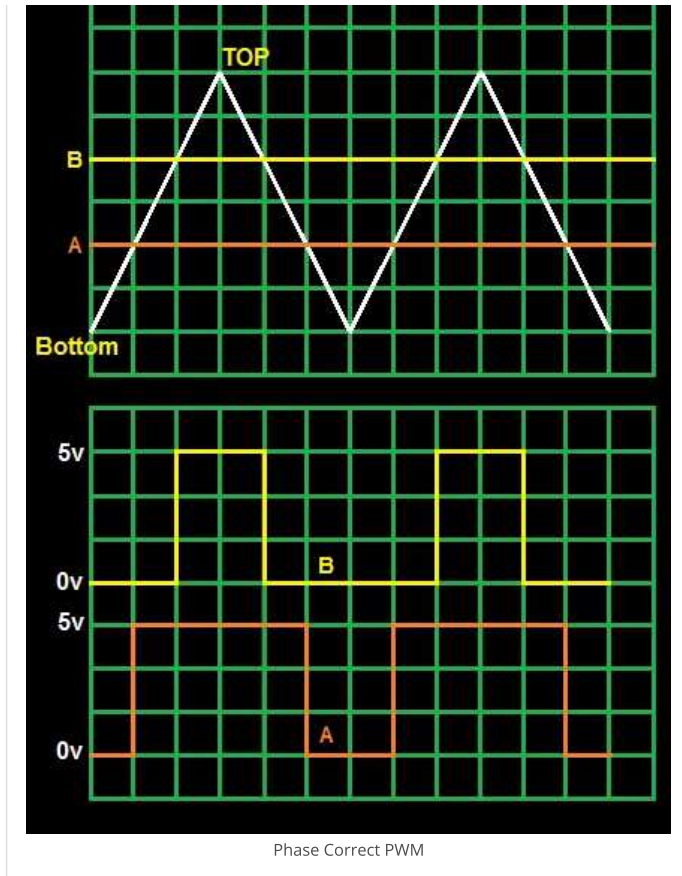


In simple terms, *this* is Fast PWM! We have a sawtooth waveform, and we compare it with a fixed voltage level (say A), and thus we get a PWM output as shown (in A). Now suppose we increase the compare voltage level (to, say B). In this case, as we can see, the pulse width has reduced, and hence the duty cycle. *But*, as you can see, both the pulses (A and B) end at the same time irrespective of their starting time.

In this mode, since sawtooth waveform is used, the timer counter TCNTn (n = 0,1,2) counts from BOTTOM to TOP and then it is simply allowed to overflow (or cleared at a compare match) to BOTTOM.

Phase Correct PWM

Now have a look at the following diagram.

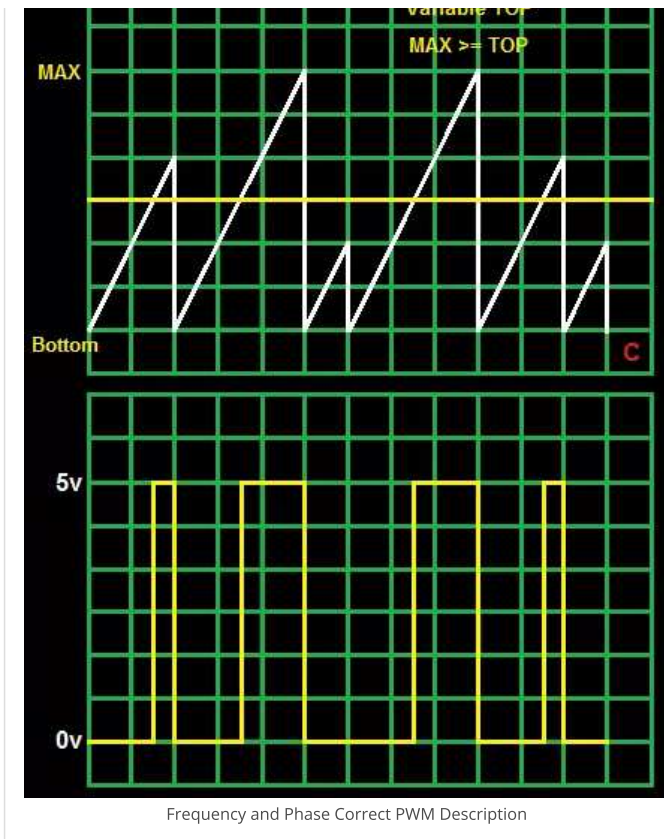


Here instead of a sawtooth waveform, we have used a triangular waveform. Even here, you can see how PWM is generated. We can see that upon increasing the compare voltage level, the duty cycle reduces. But unlike Fast PWM, the phase of the PWM is maintained. Thus it is called *Phase Correct PWM*.

By visual inspection, we can clearly see that the frequency of Fast PWM is twice that of Phase Correct PWM.

Frequency and Phase Correct PWM

Technically, Phase Correct PWM and Frequency and Phase Correct PWM are same *if* the TOP remains same. If we have a variable TOP, the frequency of the output wave will keep changing as shown below. The following illustration has a flaw (which one of my readers pointed out) that it basically represents a Fast PWM with variable frequency. Due to lack of availability of time, it is not possible for me to create another illustration. So, kindly bear with me. However, you can at least get the concept of variable TOP from the diagram.



Thus, for this, we need Frequency and Phase Correct PWM. Since in most cases the value of TOP remains same, it doesn't matter which one we are choosing – Phase Correct or Frequency and Phase Correct PWM.

Making Choices

Now that you are familiar with all the PWM concepts, it's upto you to decide

- Which timer to choose?
- Which mode of operation to choose?
- Which compare output mode to choose?

Choosing Timer

In AVR, PWM Mode is available in all timers. TIMER0 and TIMER2 provide 8bit accuracy whereas TIMER1 provides 16bit accuracy. In 8bit accuracy, you have 256 individual steps, whereas in 16bit accuracy, you have 65536 steps.

Now suppose you want to control the speed of a DC motor. In this case, having 65536 steps is totally useless! Thus we can use an 8bit timer for this. Even 8bit is too much, but there is no other choice. Obviously there isn't much difference in speed between 123/256th and 124/256th of full speed in case of a motor! But if you use servo motors, you have to use 16bit timer. More on that later.

If you need quite high resolution in your application, go for 16bit timer.

Choosing Mode of Operation

If you want to control the speed of DC motors or brightness of LEDs, go for any one of them. But if you are using it for telecommunication purposes, or for signal sampling, fast PWM would be better. For general applications, phase correct PWM would do.

Choosing Compare Output Modes

inverted mode is the most reasonable. This is because upon increasing the compare voltage, the duty cycle increases. However, you can choose any of them. Regarding toggle mode, I wonder if there is any practical application of it.

So this is it! In the next post, we will learn how to implement it in AVRs! So till then, **catch up with me by subscribing to my blog or by grabbing RSS Feeds! And don't forget to post your comments! :) Enjoy!**

Loved it? Share it!

 Facebook 52


 Twitter







 Google

 LinkedIn

 More

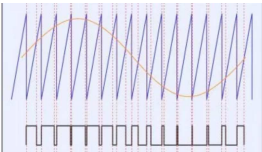
Like this:

 Like



6 bloggers like this.

Related



AVR Timers - PWM Mode - Part II

January 7, 2012

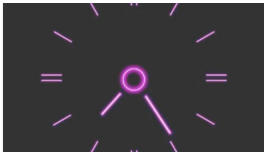
In "Atmel AVR"

Mode	REGISTER	REGISTER	REGISTER	Function/Mode of	Mode	Mode of	Mode of
0	0	0	0	Normal	Normal	Normal	Normal
1	1	1	1	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
2	2	2	2	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
3	3	3	3	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
4	4	4	4	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
5	5	5	5	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
6	6	6	6	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
7	7	7	7	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
8	8	8	8	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
9	9	9	9	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
10	10	10	10	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
11	11	11	11	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
12	12	12	12	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
13	13	13	13	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
14	14	14	14	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
15	15	15	15	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
16	16	16	16	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM
17	17	17	17	Fast PWM (Clear on Compare Match)	Fast PWM	Fast PWM	Fast PWM
18	18	18	18	Fast PWM (Set on Compare Match)	Fast PWM	Fast PWM	Fast PWM

AVR Timers - CTC Mode

July 14, 2011

In "Atmel AVR"



Introduction to AVR Timers


June 22, 2011

In "Atmel AVR"



51 Comments

← Older Comments

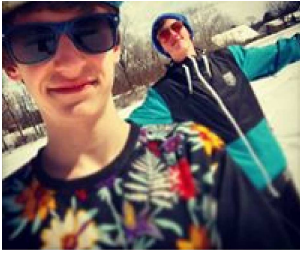


Jake Hemmerle February 26, 2014

Hey Max, did you ever finish your Part two tutorial? If so, send me a link! I would be ecstatic to read it!

REPLY

Jake Hemmerle February 26, 2014



My apologies, I just finished reading part two... Great job Max, thanks again!

[REPLY](#)

nabil March 30, 2014

can a variable voltage get generated from a variable frequency?

[REPLY](#)

Kesava Prasad April 28, 2014

Hi,

Excellent posts.

Is it possible to find the frequency of an analog signal using timers and analog comparator available in ATMEGA16/32? If so how to implement this?

[REPLY](#)

Max April 30, 2014

Hello Kesava,

If the analog signal is periodic and symmetric about the time axis, you can use the zero cross detector, and then count the number of zero crossings within a second (use timers for this). You can calculate frequency from this data. If it is periodic and not symmetric, you'll need to use comparator instead of zero cross detector. If the signal is aperiodic, then it's really difficult to find its frequency this way – you'll need to perform FFT analysis in that case, which is out of scope of AVR processors. If you have any other idea, please let me know!

Zero cross detector: <http://www.atmel.com/images/doc2508.pdf>

[REPLY](#)



Parth Katti November 13, 2014

excellent!!!!

[REPLY](#)

Manuel Marcano February 6, 2015

pana me gustaria si colocaras los códigos para realizar todos esos modos pwm en ensamblador, por que me descargo el datasheet at90usb1286, que es el que utilizo y no ofrece ejemplos de ningun tipo. Los que ofrece no me sirven me arrojan error en el avr studio 4.0, y en el atmel studio 6

[REPLY](#)

Max February 6, 2015

English?!

[REPLY](#)

venkat February 13, 2015

Can u help me out how to interface atmega16 with bldc motor how to generate pule for this motor ??

[REPLY](#)

Max March 20, 2015

Check this similar tutorial:

<http://www.atmel.com/images/doc8138.pdf>[REPLY](#)

maysam May 8, 2015

Ur posts are super excellent! tnxx

[REPLY](#)



sameh June 20, 2015

thank you max for this excellent post i have learned something new so
thank you very much

[REPLY](#)



Prashant Agarwal September 23, 2015

Thanks! ;)

Keep Reading, Keep Sharing!

[REPLY](#)



kewl August 7, 2015

I have gone through this logic again and again but really not able figure out how the absolute and relative offset is being considered. Feels like I am missing something very silly here to understand,...

```
void TimerInit(void)
{
    DISABLE_TIMER_INT; // Disable timer interrupt

    m_nGlobalTime = 0; // Reset system time

    OCR1A += TICKS_PER_MSECOND; // Set first clock period

    TCCR1A = 0; // Set TimerMode to Normal
    TCCR1B |= (1 << CS10); // Ckcklo, no pre-scaler; set TimerMode

    ENABLE_INTERRUPTS;
    ENABLE_TIMER_INT; // Enable send timer interrupt (1 ms)
}

ISR( TIMER1_COMPA_vect ) {
    uint16_t nTemp;

    nTemp = TCNT1; // Get current time
    nTemp -= OCR1A; // Subtract interrupt time

    if (nTemp < (TICKS_PER_MSECOND / 2)) // If more than half period left
    {
        OCR1A += (TICKS_PER_MSECOND); // Add Offset to OCR1A relative
    }
    else
    {
        OCR1A = TCNT1 + (TICKS_PER_MSECOND); // Set OCR1A to 1 ms absolute
    }
}
```

}

[REPLY](#)

nonko September 15, 2015

In die dia. "Phase Correct PWM" there is no connect in amplitude between the upper and lower diagram, it's very confusing.

Where the B-line crosses the trinangular graph (5V), it whould be 4 blocks up from 0V.

Yet it is 3

Where the A-line crosses the trinangular graph (5V), it whould be 2 blocks up from 0V.

Yet it is 3

So what should it be?

[REPLY](#)

blackjellyfish2015 September 20, 2015

excellent post!

[REPLY](#)

Prashant Agarwal September 23, 2015

Thanks ;)

Keep Reading, Keep Sharing!

[REPLY](#)

Puru September 24, 2015

"But if you are using it for telecommunication purposes, or for signal sampling, fast PWM would be better." My question is why ? What advantages will it offer over phase correct PWM

[REPLY](#)

sushant January 12, 2016

Hi max, I did not understood why we use only ramp signal for Normal or fast PWM mode? and Why Traingular wave for Phase correct PWM mode?

REPLY



Vishnu February 28, 2016

Sir you are a great help to newbies like us.
Thank you for your efforts.

REPLY

← Older Comments

Leave a Reply

Enter your comment here...

Copyright



maxEmbedded by [Mayank Prasad](#) is licensed under a [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Unported License](#).

© Copyright 2011-2014 by maxEmbedded. Some rights reserved.

Tags

[accelerometer](#) [adapter](#) [adc](#) [assembly](#) [avr](#) [adc](#) [avr](#) [basics](#) [avr](#) [eeprom](#) [avr](#) [interrupts](#) [avr](#) [serial](#) [avr](#) [timers](#) [basics](#) [bootloader](#) [code](#) [gallery](#) [code](#) [optimization](#) [color](#) [spaces](#) [counters](#) [cv](#) [basics](#) [filter](#) [circuit](#) [i2c](#) [ir](#) [sensor](#) [matlab](#) [msp430](#) [basics](#) [opencv](#) [pcb](#) [design](#) [power](#) [supply](#) [printed](#) [circuit](#) [boards](#) [pwm](#) [raspberry](#) [pi](#) [rf](#) [communication](#) [rf](#) [module](#) [robotics](#) [robot](#) [locomotion](#) [rs232](#) [sbc](#) [basics](#) [sensors](#) [Serial](#) [spi](#) [ssd](#) [timers](#) [toolchain](#) [transformer](#) [uart](#) [usart](#) [voltage](#) [regulator](#) [wireless](#)

Top Posts & Pages

- The ADC of the AVR
- AVR Timers - TIMERO
- The USART of the AVR
- The SPI of the AVR
- How to build your own Power Supply
- AVR Timers - CTC Mode
- AVR Timers - PWM Mode - Part I

maxEmbedded is a free and open source platform to share knowledge and learn about the concepts of robotics, embedded systems and computer vision. People from around the world who are enthusiastic about these topics and willing to support the open source community are invited to share their information, knowledge and expertise by means of written tutorials and videos on the website.

»