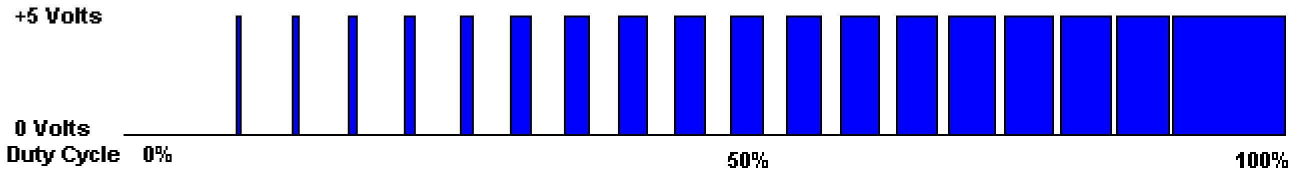
# Pulse Width Modulation (PWM) Tutorial

**Pulse Width Modulation** – Using digital pulses to create some analog value other than just 'high' and 'low' signal levels. Many digital systems are powered by a 5-Volt power supply, so if you filter a signal that has a 50% duty cycle you get an average voltage of 2.5 Volts. Other duty cycles produce any voltage in the range of 0 to 100% of the 'high' voltage, depending upon the PWM resolution.

The duty cycle is defined as the percentage of digital 'high' to digital 'low' signals present during a PWM period.

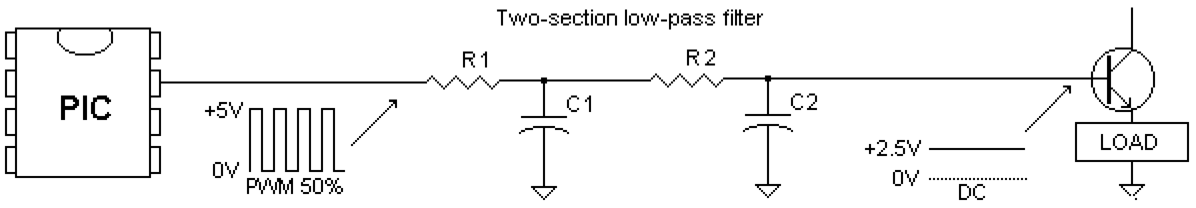
The PWM resolution is defined as the maximum number of pulses that you can pack into a PWM period.

The PWM period is an arbitrarily time period in which PWM takes place. It is chosen to give best results for your particular use.



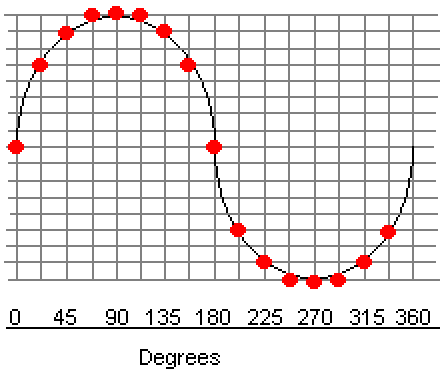
**Uses for PWM:** 1) To digitally create an analog output voltage level for control functions and power supplies.
2) To digitally create analog signals for arbitrary waveforms, sounds, music and speech.

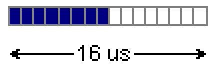
**Using PWM to generate an analog voltage level:** A common use is in power supplies. The PWM resolution is selected to be equal to or greater than the resolution requirements of the power supply. A 5-Volt power supply that can be adjusted to +/- 1 milli-Volt should use a PWM resolution of 5,000 or greater. The PWM output is then filtered to obtain acceptable ripple. The filter can be a simple low-pass filter.



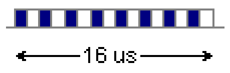
The above figure shows a PIC microcontroller generating a 50% duty cycle PWM signal at 5,000 Hz, a two-section 5,000 Hz low-pass filter and a pass-transistor with a direct current input of +2.5 Volts. The filter frequency = 1/2 pi RC for each section.

**Using PWM to generate an analog waveform:** Any shape waveform can be generated by outputting a sequence of PWM values that correspond to multiple points on the waveform. When you use more points, heavier filtering and greater PWM resolution, you can represent fast waveforms with great accuracy. In practice, a PIC microcontroller can easily output reasonably decent sine waveforms in the voice frequency range. The 4 KHz sine waveform below uses a 16 level resolution PWM signal and 16 points on the sine wave. The PWM frequency is about 30 KHz using an 'on/off' cycle, and about 500 KHz using a 'distributed' cycle. The 'distributed' cycle produces smoother results, as shown below.





**'On/off' PWM**
**50% duty cycle**



**'Distributed' PWM**
**50% duty cycle**
**(Easier to filter)**

**The 'on/off' PWM duty cycle:** Starts with eight pulses 'high' for the first part of the PWM cycle, then finishes with eight 'low' pulses for the remainder of the PWM cycle. The ratio of the percent 'high' time to the percent 'low' time is called the duty cycle. Note that the PWM frequency of the 'on/off' PWM for any duty cycle in the drawing above is: F = 1/P = 1 / 16us = 62.5KHz.

**The 'distributed' PWM cycle:** Also uses eight 'high' pulses and eight 'low' pulses as does the previous example, but spreads them out during the entire PWM cycle. Note that the PWM frequency of the 'distributed' PWM 50% duty cycle in the drawing above is: F = 1/P = 1 / 2us = 500 KHz. This is great if most of your PWM is around 50% duty cycle but the PWM frequency gradually slows to 62.5KHz as you approach 0% or 100%.

**Why the PWM frequency is important:** The PWM is a large amplitude digital signal that swings from one voltage extreme to the other. And, this wide voltage swing takes a lot of filtering to smooth out. When the PWM frequency is close to the frequency of the waveform that you are generating, then any PWM filter will also smooth out your generated waveform and drastically reduce its amplitude. So, a good rule of thumb is to keep the PWM frequency much higher than the frequency of any waveform you generate. Finally, filtering pulses is not just about the pulse frequency but about the duty cycle and how much energy is in the pulse. The same filter will do better on a low or high duty cycle pulse compared to a 50% duty cycle pulse. Because the wider pulse has more time to integrate to a stable filter voltage and the smaller pulse has less time to disturb it.

**Conclusion:** PWM is the poor mans' digital-to-analog converter (DAC). It has problems not shared by other DACs, such as speed and instantaneous voltage output, but it is the least expensive way to get an analog voltage output from a microcontroller. Some other uses are to operate relays and solenoids that require high 'pull-in' current and more moderate 'hold' current. Or for devices that require a lower operating voltage than your microcontroller, such as a 1.5-Volt lamp. There are additional ways to improve PWM, such as using LC filters. Although they cost more, they will not drop the voltage as do RC filters. Also, you can use two microcontroller I/O pins to mix 'on/off' PWM with 'distributed' PWM to obtain twice the resolution. These and other methods are discussed in the Datadog Systems PIC10F2XX tutorials. I hope this PWM tutorial was useful to you.

Best regards,
John Massa