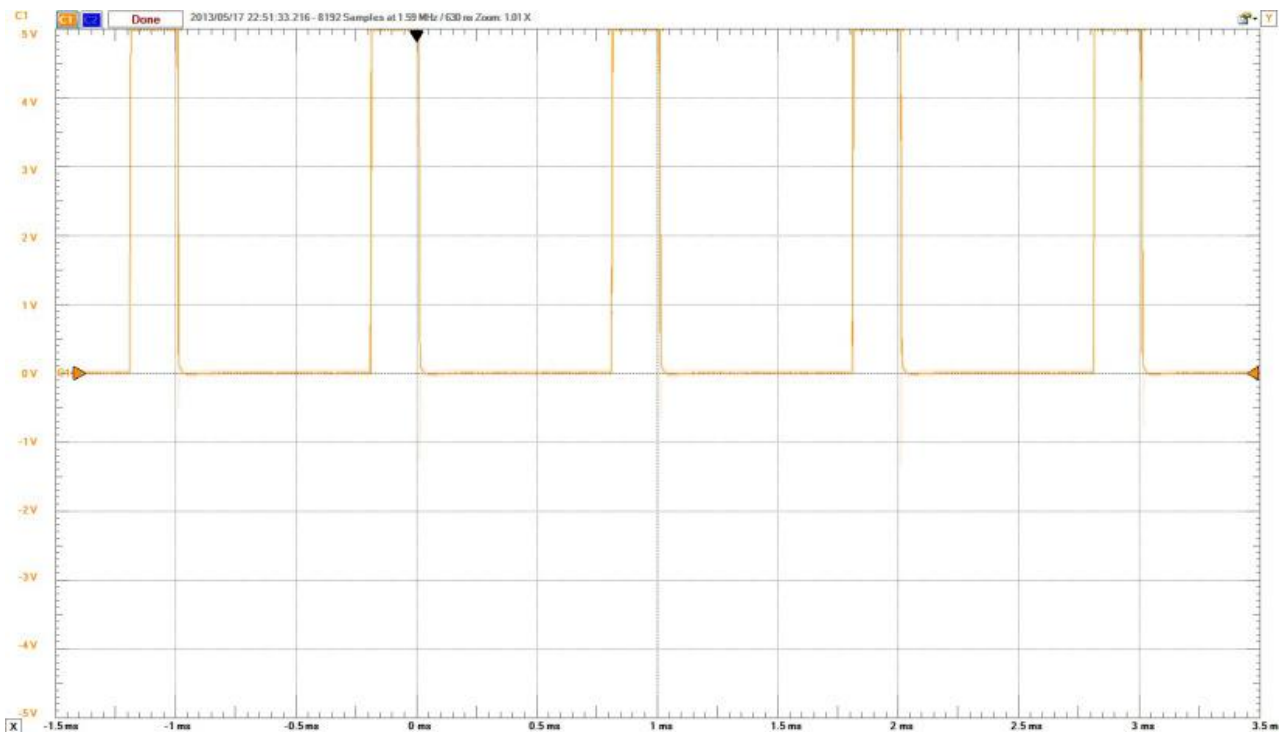


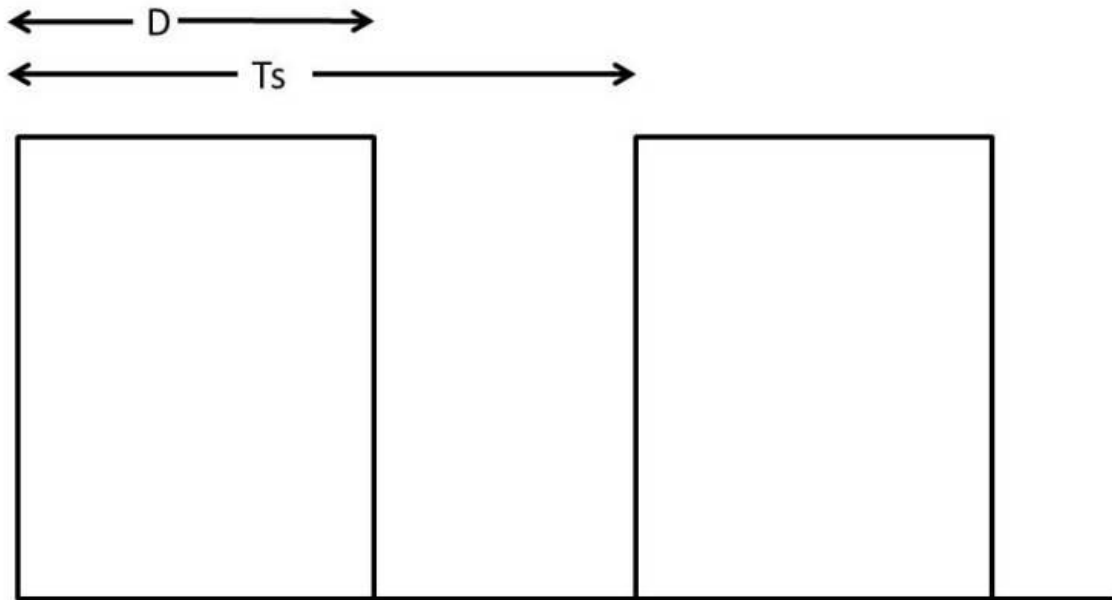
# Implementing PWM On PIC18F Microcontroller



PWM output with 20% duty cycle.

## What is PWM?

As you guys know, microcontrollers, FPGAs, and other embedded processors can only output a high or low signal based on the power supplied to it. In other words, embedded processors cannot output a variety of voltages. Despite this, we have the processor “pretend” to output different voltages using a series of pulses and varying the width of the pulses. The technique of varying the width of a series of pulses is called Pulse Width Modulation (PWM).

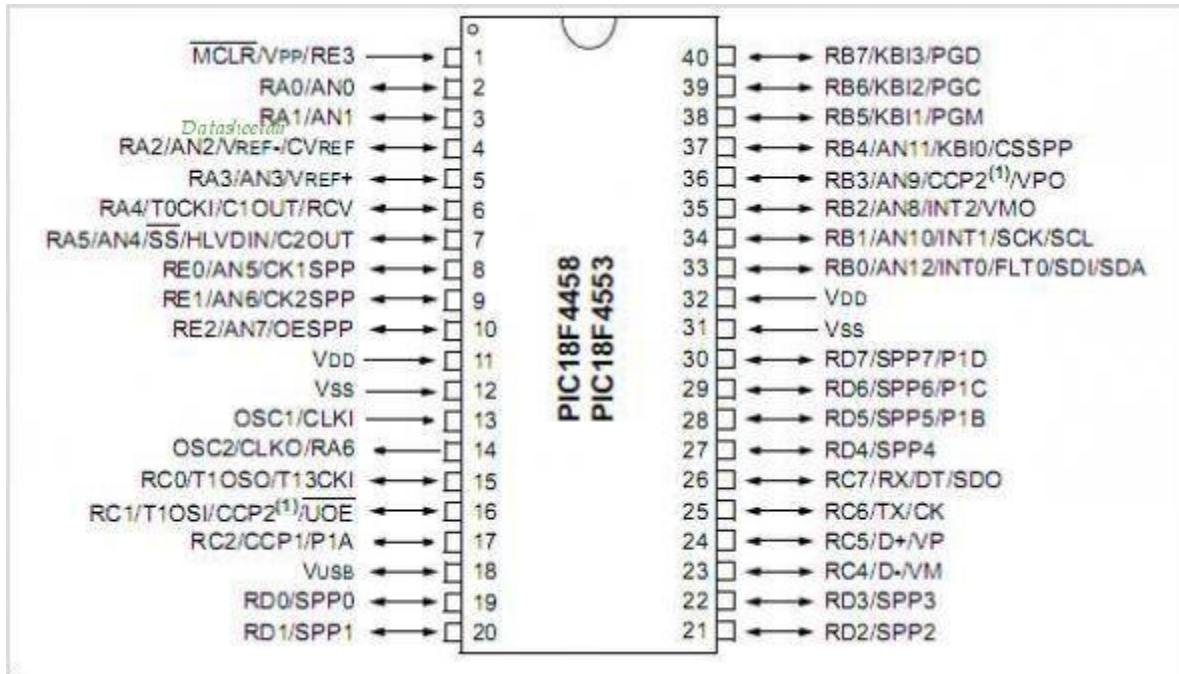


The picture above shows an example of a simple pulse train. The time it takes for each pulse to start again is referred to as its period ( $T_s$ ). In each pulse train, the pulse is on for a certain amount of time before turning off. The percentage which the pulse's on time compared to its time off is called the duty cycle. Duty Cycle is expressed as...

$$D = \frac{T_{on}}{T_{on} + T_{off}}$$

### PIC18F And PWM

As you're already aware of, all microcontrollers feature a set number of PWM pins. But what pins are specifically designed for PWM? When reading your PIC18F datasheet, look for pins that have CCP (compare capture pwm) labeled. So let's look at a picture featuring the PIC18F4553



As you see in the picture, RC1, RC2 and RB3 can be configured as PWM pins.

## PWM Calculations

So there are three crucial registers you must take in account for PWM: PR2, and CCPRXL:CCPXCON[5:4]. Configuring the PR2 register, will allow you to configure the PWM period while selecting a proper the value for CCPRXL:CCPXCON[5:4] will allow you to choose the right duty cycle. For choosing the right value for the PR2 register, use the following equation...

$$PR2 = \frac{PWMPeriod}{4 * T_{osc} * TMRPrescaleValue} - 1$$

Keep in mind that  $T_{osc}$  is the the inverse of the frequency that your microcontroller is running at. Also, we will use a prescale of 16 as we will use this in the upcoming code example. For CCPRXL:CCPXCON[5:4], we will use the following equation...

$$CCPRXL : CCPXCON[5 : 4] = \frac{PWMDutyCycle}{T_{osc} * TMRPrescale}$$

CCPRXL:CCPXCON[5:4] is a 10 bit number. To make things easier on you guys, make sure you convert your number to a 10 bit binary number. After you convert the number to a binary number, set your CCPRXL register to the first 8 MSB numbers. Finally, take the last 2 LSB and you use it for your CCPXCON[5:4].

## PWM Coding Procedure

Now that I talked about the calculations, let's talk about how to code your PIC18F microcontroller to use PWM. For this tutorial, we will use the PIC18F4553's CCP1 pin, set the PWM period at 1KHZ, a time prescale of 16, a clock frequency of  $8 * 10^6$  and a

50% duty cycle. Parts of the final code can be found in my Beginner's Guide To PIC18F Microcontroller.

First we need to configure the CCP1 pin as an output pin, which is RC2. To do this we set the TRISCbits.RC2 to 0 and PORTCbits.RC2 to 0.

```
1 TRISCbits.RC2=0;
2 PORTCbits.RC2=0;
```

Now, we need to choose a value for PR2 register. Here is my calculation for PR2.

$$PR2 = \frac{10^{-3}}{4 \cdot 16 \cdot \frac{1}{8 \cdot 10^6}} - 1 = 124$$

Now that we found the number, let's convert it to a binary number and use it.

```
1 PR2 = 0b1111011;
```

PWM uses timer2 to work. So we need to not only turn on timer2, but we must set the time scale to 16. This can be done by properly configuring T2CON by setting T2CON's TMR2ON to 1, and T2CKPS1 to 1.

```
1 T2CON = 0b00000111 ;
```

CCPR1L:CCP1CON[5:4] is the last thing to compute.  
 $CCPR1L : CCP1CON[5 : 4] = \frac{\frac{1}{10^3} \cdot .5}{\frac{1}{8 \cdot 10^6} \cdot 16} = 250$

When we convert 250 to its binary value, we get 0011111010. However, we take the first 8 MSB for CCPR1L and the last 2 LSB for CCP1CON bit 5 and CCP1CON bit 4.

```
1 CCPR1L = 0b00111110;
2 CCP1CON = 0b00101100;
```

Since you guys been so swell, here's the final code for this example.

```
1 #include <p18f4553.h>
```

```
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <usart.h>
5  #include <delays.h>
6  #include <portb.h>
7
8  #pragma config FOSC = HS
9  #pragma config WDT = OFF
10 #pragma config PBADEN = OFF
11 #pragma config LVP = OFF
12 #pragma config PWRT = ON
13 #pragma config DEBUG=OFF
14
15 int main(void)
16 {
17     TRISCbits.RC2=0;
18     PORTCbits.RC2=0;
```

```
19     PR2 = 0b01111100 ;
20     T2CON = 0b00000111 ;
21     CCPR1L = 0b00111110;
22     CCP1CON = 0b00101100;
23
24     while(1);
25 }
```

Source: <http://coolcapengineer.wordpress.com/2013/05/27/implementing-pwm-on-pic18f-microcontroller/>