

## D/A Converter by Using PWM

### Using PWM to Generate Analog Output

#### DESCRIPTION

Pulse Width Modulation (PWM) modules, which is kind of digital waveforms, can be used as cheap D/A converters only with a few external components. A wide variety of micro controller applications need analog output but do not require high resolution D/A converters. Some applications such as a speech synthesis systems in toy, controlling the speed of a DC motor and generation variable voltages in a power supply, also do not require high resolution D/A converters. For these applications, Pulse Width Modulated outputs may be converted to analog output.

Conversion of PWM waveforms to analog signals involves the use of analog low pass filters. This application note describes the design criteria of the analog filter necessary and the requirements of the PWM frequency.

In a typical PWM signal, the base frequency is fixed, but the pulse width is a variable or fixed. The pulse width is directly proportional to the amplitude of the original unmodulated signal. In other words, in a PWM signal, the frequency of the waveform is a constant while the duty cycle varies according to the amplitude of the original signal. A simple low pass filter is used to generate an output voltage directly proportional to the average time spent in the high level.

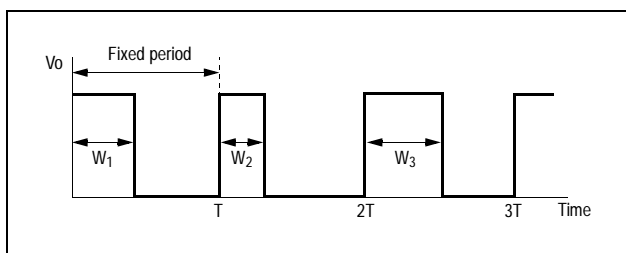


Figure 1. A Typical PWM Waveform with Variable Pulse Width

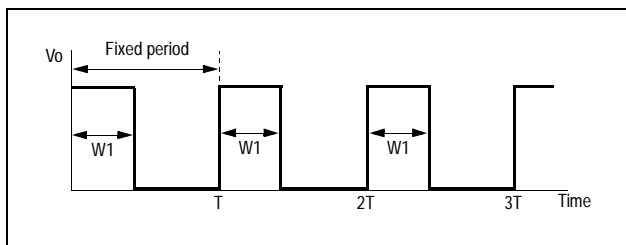


Figure 2. A Typical PWM Waveform with Fixed Pulse Width

A Fourier analysis of a typical PWM signal (such as the one depicted in Figure 1 or 2) shows that there is a strong peak at frequency  $F_n = 1/T$ . Other strong harmonics also exist at  $F=K/T$ , where K is an integer. These peaks are unwanted noise which is a AC value and should be eliminated to get a DC value. This requires that the PWM signal be low-pass filtered, thus eliminating these inherent noise components as shown in Figure 3.

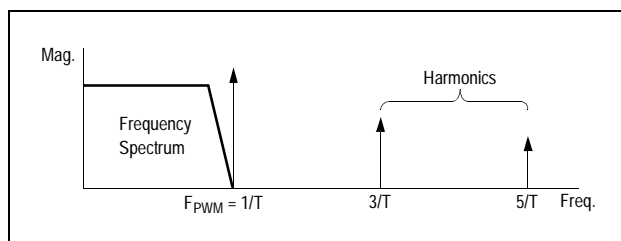


Figure 3. Power Spectrum of A PWM Signal

The band-width of the desired signal should be

$$F_{BW} \leq \left( F_{PWM} = \frac{1}{T} \right)$$

If  $F_{BW}$  is selected such that  $F_{BW} = F_{PWM}$ , then the external low-pass filter should be a brick-wall type filter. Brick-wall type analog filter are very difficult and expensive to build. So, for practical purpose, the external low-pass filter, that has much smaller cut-off bandwidth than PWM bandwidth, should be used as shown in Figure 4.

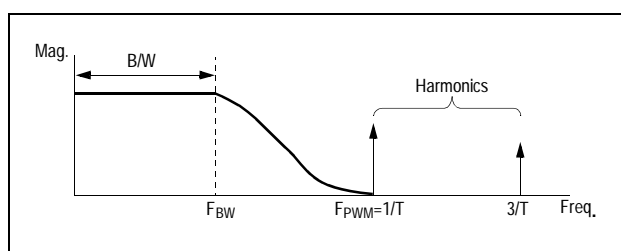


Figure 4. External Low-Pass Filter

From Figure 4, We can know that

$$F_{BW} \ll F_{PWM} \quad (1)$$

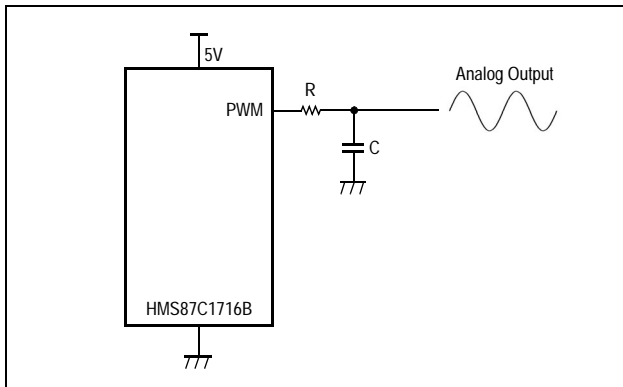
$$F_{PWM} = K \times F_{BW}$$

where, K is a constant such that  $K \gg 1$

The value of K should be chosen dependant upon the number dB the inherent fundamental noise component of PWM will be rejected.

**EXAMPLE 1**

It is required to design a simple RC low-pass filter to generation variable voltages from a pulse width modulated signal of bandwidth 4KHz. Figure 5 shows a simple voltage generation circuit. The PWM output generates variable voltages.



**Figure 5. RC Filter Connected To PWM of HMS87C1716B**

From the equ. (1), choosing arbitrarily  $K = 5$ ,

$$F_{PWM} = K \times F_{BW} = 5 \times 4 = 20 \text{ KHz}$$

Choosing the -3dB point at 4KHz, and using the relation  $RC = 1 / (2\pi f)$ , we get  $R = 4K\Omega$ , if C is chosen as  $0.01\mu\text{F}$ .

$$f = \frac{1}{(2\pi RC)} = 4 \text{ KHz}$$

that is, cut-off bandwidth is 4KHz.

Since the PWM frequency is selected as 20KHz, the fundamental noise peak to be filtered is at 20KHz. So how many dB the main peak of PWM signal is cut-off at 20KHz.

$$dB = 10 \log \left( \frac{V_{out}}{V_{in}} \right)^2 = 10 \log |H(f)|^2$$

$$|H(f)| = \frac{1}{\sqrt{1 + (2\pi \times f_{PWM} \times RC)^2}}$$

$$dB = 10 \log [1 + (2\pi \times f_{PWM} \times RC)^2] = -14 \text{ dB}$$

From this calculation, if pulse width is 50% of a period with 5V, we can calculate cut-offed fundamental noise and low-pass filter output.

$$V_{main} = 2 \times PWM_{high} \times \left[ \frac{PWM_{duty}}{PWM_{period}} \right] \times \frac{\sin \left( \frac{PWM_{duty}}{PWM_{period}} \right)}{\left( \frac{PWM_{duty}}{PWM_{period}} \right)}$$

$$V_{main} = 2 \times 5 \times \left[ \frac{2.5}{5} \right] \times \frac{\sin \left( \frac{2.5}{5} \right)}{\left( \frac{2.5}{5} \right)} = 3.18 \text{ V}$$

$$dB = 10 \log \left( \frac{V_{noise}}{V_{main}} \right)^2$$

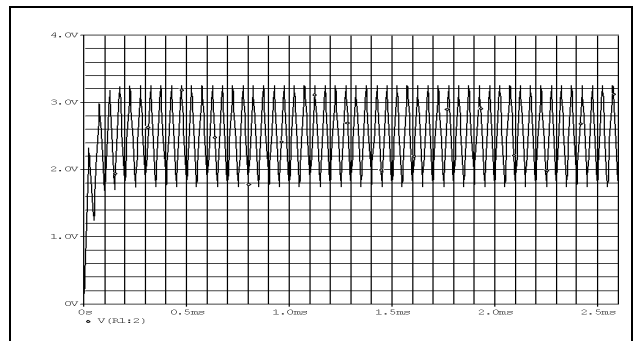
$$V_{noise} = 10^{\left( \frac{dB}{20} \right)} \times V_{main} = 0.62 \text{ V}$$

$$V_{lowpass} \leq 5 \times \left[ \frac{PWM_{duty}}{PWM_{period}} \right] \pm V_{noise} = 2.5 \pm 0.62$$

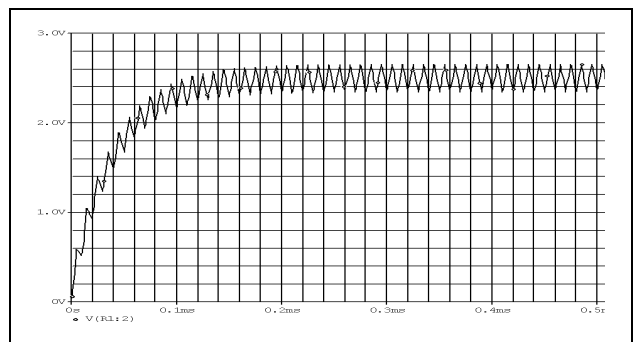
where, 2.5V is DC value at zero frequency and  $\pm 0.62$  is cut-offed AC value at main peak of the  $F_{PWM}$ . The result value of  $V_{low-pass}$ , which is sum of DC and AC value, is described in Figure 6.

From Figure 6, we can intuitively know that this rejection of -14dB is not enough to get DC value at  $K=5$ . Therefore a higher order active low-pass filter is necessary to reject AC value. Or, if the micro controller is able to modulate higher PWM frequencies, the rejection of noise(AC) will be greater. The spice simulation results of using higher PWM frequencies are shown in Figure 7 and 8.

The HMS800 series which have 10bit high speed PWM make it easier to generate analog output.



**Figure 6. K=5, Low-pass filter Output of PWM**



**Figure 7. K=20, Low-pass filter Output of PWM**

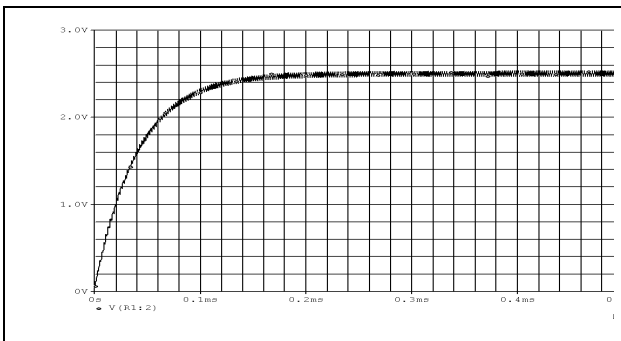


Figure 8. K=100, Low-pass filter Output of PWM

**EXAMPLE 2**

It is required to design a simple RC low-pass filter to generation 60Hz sine waveform from a pulse width modulated signal.

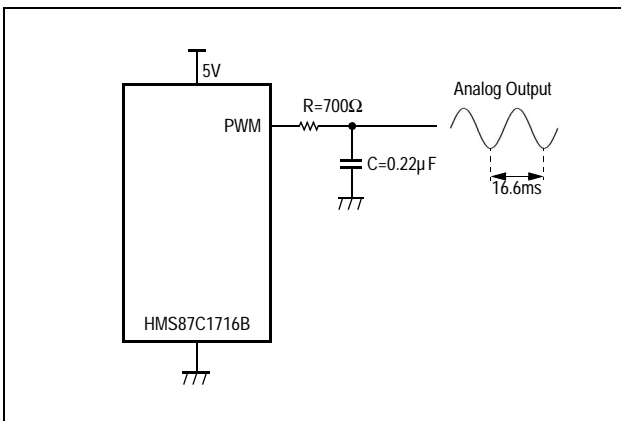


Figure 9. Sinewave Generation using PWM of HMS87C1716B

The software `sin_fun.c`, written for the HMS87C1716B, generates 60Hz sine function which has 60 sample points for a period in my design. The bandwidth of RC filter, which have  $R=700\Omega$  and  $C=0.22\mu F$ , is 1KHz and a period of PWM is 30KHz at 8MHz source,  $PWM1HR=0$  and  $T3PPR=0xff$ . For each step of sine wave is generated by using PWM which is over 10 times higher frequency than low-pass bandwidth to settle one.

**Appendix A:**

```

/*****
Programmer : Jungmin Choi
Data      : 2003.12.10
Subject   : generation sine waveform using PWM
Device    : HMS87C1716B
File name : sin_fun.c
*****/
/*
Bandwidth : 1/(2*pi*R*C) = 1KHz
    
```

The frequency of PWM is decided by number of sampling points and required sine frequency. The designer should experiment with this value to find the most optimal number for the each application. These two criteria decide to the required PWM frequency..

$$f_{PWM} = N_{sample} \times f_{sin} \times K \quad (K > 10)$$

And how to extract sample point of sine wave and how to match each sample point to pulse width of PWM. Let's assume that a period of PWM is `0xff` and a pulse width is matched like below the equation.

$$t = (0 \rightarrow 2\pi) \quad step = \frac{\pi}{N_{sample} - 1}$$

$$P_{WIDTH} = \left\lfloor \frac{255 \times (\sin t + 1)}{2} \right\rfloor$$

where,  $\pi/(N_{SAMPLE} - 1)$  is interval from 0 to  $2\pi$ ,  $t$  are sample points of sine wave and  $P_{WIDTH}$  is pulse width of PWM.

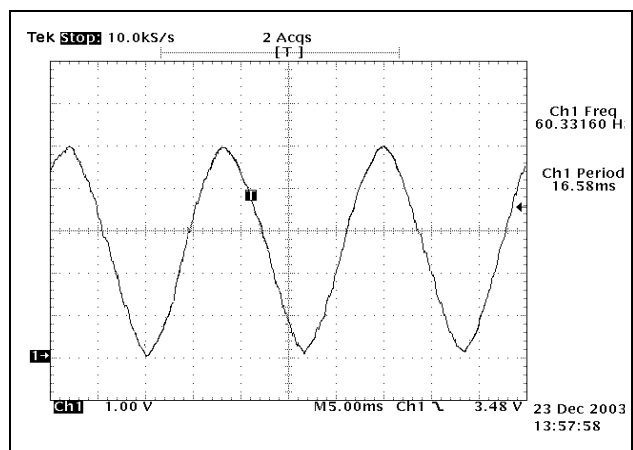
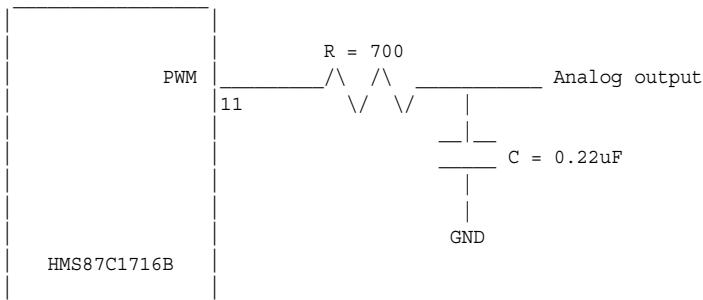


Figure 10. The result of PWM output

Figure 10 is the result of example 2 and Appendix A is the source code for example 2.

Author: June Choi  
 MCU application team  
 e-mail: jungmin1.choi@hynix.com



```

*/
#include "HMS87C1716B.h"

unsigned char SIN_TABLE60[] __attribute__((section(".text"))) = {
    128, 141, 154, 168, 180, 192, 204, 214, 223, 232,
    239, 245, 250, 253, 255, 255, 254, 251, 247, 242,
    236, 228, 219, 209, 198, 186, 174, 161, 148, 134,
    121, 107, 94, 81, 69, 57, 46, 36, 27, 19,
    13, 8, 4, 1, 0, 0, 2, 5, 10, 16,
    23, 32, 41, 51, 63, 75, 87, 101, 114, 127 };

unsigned char p_count;

// flag setting
struct flag0
{
    char f_msec:1;
} f0;

// Timer0 interrupt
void __attribute__((interrupt("11")))
Timer0_Int(void)
{
    f0.f_msec = 1;
}

// Initial I/O and Peri.
void Initial(void)
{
    RAO = 0xff;
    RAFUNC = 0;
    RA = 0;
    RBIO = 0xff;
    RBFUNC = 0x3C;           // PWM1 setting
    RB = 0;
    RCIO = 0xff;
    RC = 0;
    RDIO = 0xff;
    RDFUNC = 0;
    RD = 0;
    REIO = 0xff;
    RE = 0;

    //Initialize control reg.
    CKCTRLR = 0x3e;         // initial watch dog timer source
    WDTR = 0xff;
    TM0 = 0x0f;             // initial timer0 (4us)
    TDR0 = 0x44;           // 4us * 0x44 = 272us * 60 = 16.32ms
    TM1 = 0x0;
    TM3 = 0x23;             // initial pwm source
    IENH = 0x20;           // timer0 interrupt enable
}

```

```

    IENL = 0;
    IRQH = 0;
    IRQL = 0;
}

// PWM function
void PWM_fun(void)
{
    TM3 = 0;
    TM3 = 0x20;           // source = 8Mhz = 125nS
    PWM1HR = 0;
    T3PPR = 0xff;       // 125nS*0x00ff= 30KHz
    T3PDR = SIN_TABLE60[p_count];
    TM3 = 0x23;         // start timer3
}

// main function
main()
{
    Initial();           // initial I/O and peri.
    asm(" EI ");
    while(1)
    {
        WDTR = 0xff;    // watch dog timer initial again
        if ( f0.f_msec )
        {
            f0.f_msec = 0;
            PWM_fun();
            p_count++;
            if(p_count>59)
                p_count = 0; // end of sine table
        }
    }
}

```