May 23, 2015

EE 354
Notes on PWM for the ARM Cortex M4 Discovery Board

The STM32F407VG has 14 timers and 12 of them can be used to generate Pulse Width Modulation (PWM) signals. (Timers 6 and 7 cannot be used for PWM)

In PWM we start with a base frequency of say 1,000Hz and each cycle we output a single pulse whose width varies from 0% of the period to 100% of the period.

![Figure 1](image)

**Figure 1**

Pulse width modulation. In this case the signal is 50% of maximum.

The base frequency of the PWM is \( f = \frac{1}{T} \) so that the frequency content of a PWM signal will have frequencies at \( f \) and higher due to the base frequency. The signal information is transmitted by the duty cycle and will be, generally, a much lower frequency signal. Thus a relatively simple low pass filter can eliminate the frequencies at the base frequency and above and recover an analog signal from the PWM.

The ARM processor generates PWM signals using three registers: The timer count register `TIMx_CNT`, the auto-reload register `TIMx_ARR`, and the capture/compare register `TIMx_CCRx`. The CNT register counts up from zero until it reaches the value in the ARR register. This amount of time establishes the base frequency. The output goes from low to high at the start of each cycle. When the CNT register gets to the value in the CCR register the signal set back to low. Figure 2 shows the functional diagram for counters 2 to 5. From the figure we note that there are four outputs associated with the timer. Each of these outputs has a capture register but they all share a common counter register and a common auto-reload register. This means the timer can generate four PWM signals as long as they all have the same base frequency. Further, Timers 2 to 5 can run simultaneously and independently of one another so these four timers could generate 16 PWM signals at 4 different base frequencies.

**Example 1**
Use Timer 3 to use PC6 (Channel 1) to generate a PWM signal which has a 12-bit resolution and produces a ramp function that runs from 0 to 4095. (Note that the pin assignments for each timer channel are listed in the data sheet\(^1\).) This example uses an integer variable called `tmp` which goes from 0 to 4095. After `tmp` is changed to a new value it is loaded into `TIM3_CCR1` which holds the pulse width count.

The auto-reload register is loaded with 4096 which is the maximum count for one cycle. The value of tmp is set back to zero when it reaches this maximum. A software delay has been added to slow the time it takes the ramp to grow.

Example 2
Create a PWM signal using Timer 2 to output a sinusoid at 1 KHz which has a sample frequency of 20000 Hz. The PWM output will appear on PA5 which is channel 1 of Timer 2 is chosen. See the data sheet. Since the sample frequency is 20000 Hz and the signal frequency is 1000 Hz there will be 20 samples per signal cycle. The while loop increments $t$ by the sample time, $T$ twenty times before starting it again at 0.

A sinusoid varies from -1 to +1. We add one to the sine wave so that it goes from 0 to 2. We then multiply by 2000 so that it ranges from 0 to 4000. Since the auto-reload register has the number 4200 in it we are very near the maximum. Since 4000/4200 is about 95% this will be the maximum duty cycle of the PWM output.

May 23, 2015  

Example 1

//PWMEx1.c

/* This program generates a PWM signal on PC6 using Timer 3. The signal has 12-bits of resolution and outputs a ramp function which goes from 0 to 4095. */

#include "stm32f407vg.h"

int main()
{
    int tmp, i;

    //Clock bits
    RCC_AHB1ENR |= 4; //Bit 2 is GPIOC clock enable bit
    RCC_APB1ENR |= 2; //Enable peripheral timer for timer 3 (bit 1)

    //I/O bits
    GPIOC_MODER |= 0x2000; //Bits 13-12 = 10 for Alt Funct Mode on PC6
    //OTYPER register resets to 0 so it is push/pull by default
    GPIOC_OSPEEDER |= 0x3000; //Bits 13-12 = 11 for high speed on PC6
    //PUPDR defaults to no pull up no pull down
    //Timer 3 bits
    GPIOC_AFRL = 0x02000000; //Sets PC6 to Timer 3
    TIM3_CCMR1 |= 0x60; //Timer 3 in PWM Mode bits 6,5,4 = 110
    TIM3_CCR1 |= 0x0C; //Timer 3 Preload enable and fast enable
    TIM3_CR1 |= (1 << 7); //Auto reload is buffered
    TIM3_PSC = 0; //Don't use prescaling
    TIM3_ARR = 4096; //((168 MHz/2)/4096 = 20508 Hz
    TIM3_CCR1 = 0; //Duty cycle starts at 0
    TIM3_CCR1 |= 1; //Compare and capture output enable
    TIM3_EGR |= 1; //Enable event
    TIM3_CR1 |= 1; //Enable Timer 3

    //Main program loop
    tmp = 0;
    while(1)
    {
        TIM3_CCR1 = tmp;
        for(i=0;i<10000;i++); //Delay
        tmp++;
        if(tmp >= TIM3_ARR)
        {
            tmp = 0;
        }
    }
}
/* Create a PWM signal using Timer 2 to output a sinusoid at 1 KHz which has a sample frequency of 20000 Hz. The PWM output appears on PA5. */
#include "stm32f407vg.h"
#include <math.h>        //Need this for sine function
const float w = 2*3.14159265359*1000;
const float T = 1.0/20000;
int main()
{
    float t;
    //Clock bits
    RCC_AHB1ENR |= 1;         //Bit 0 is GPIOA clock enable bit
    RCC_APB1ENR |= 1;         //Enable peripheral timer for timer 2 (bit 0)
    //I/O bits
    GPIOA_MODER |= 0x800;    //Bits 11-10 = 10 for Alt Funct Mode on PA5
    GPIOA_OSPEEDER |= 0xC00;  //Bits 11-10 = 11 for high speed on PA5
    //Timer 2 bits
    GPIOA_AFRL = 0x0100000;  //Sets PA5 to Timer 2
    TIM2_CCMR1 |= 0x60;      //Timer 2 in PWM Mode bits 6,5,4 = 110
    TIM2_CCMR1 |= 0x0C;      //Timer 2 Preload enable and fast enable
    TIM2_CR1 |= (1 << 7);    //Auto reload is buffered
    TIM2_PSC = 0;            //Don't use prescaling
    TIM2_ARR = 4200;         //((168 MHz/2)/4200 = 20000 Hz
    TIM2_CCR1 = 0;           //Duty cycle starts at 0
    TIM2_CCER |= 1;          //Compare and capture output enable
    TIM2_EGR |= 1;           //Clear counter on update
    TIM2_CR1 |= 1;           //Enable Timer 2
    //Main program loop
    t = 0;
    while(1)
    {
        t = 0;
        while(t < 20*T)     //20 samples make one cycle at 1 KHz
        {
            //maximum number of counts is 4200
            TIM2_CCR1 = (int)(2000*(1 + sin(w*t)));
            t += T;
        }
    }
}