一起学习 CC3200 系列教程之定时器模式

阿汤哥

序：

能力有限，难免有错，有问题请联系我，

QQ1519256298　　hytga@163.com

Pdf 下载 http://pan.baidu.com/s/1hqiWB56

现在介绍 CC3200 的定时器。

CC3200 的计数器可以作为普通的有 4 组，每组有 2 个计数器，每个计数器的位数是 16BIT，而这两个可以组成一个 32bit 的计数器。

计数器可以根据功能分成定时器模式，捕获模式，PWM 模式...

计数器框图：

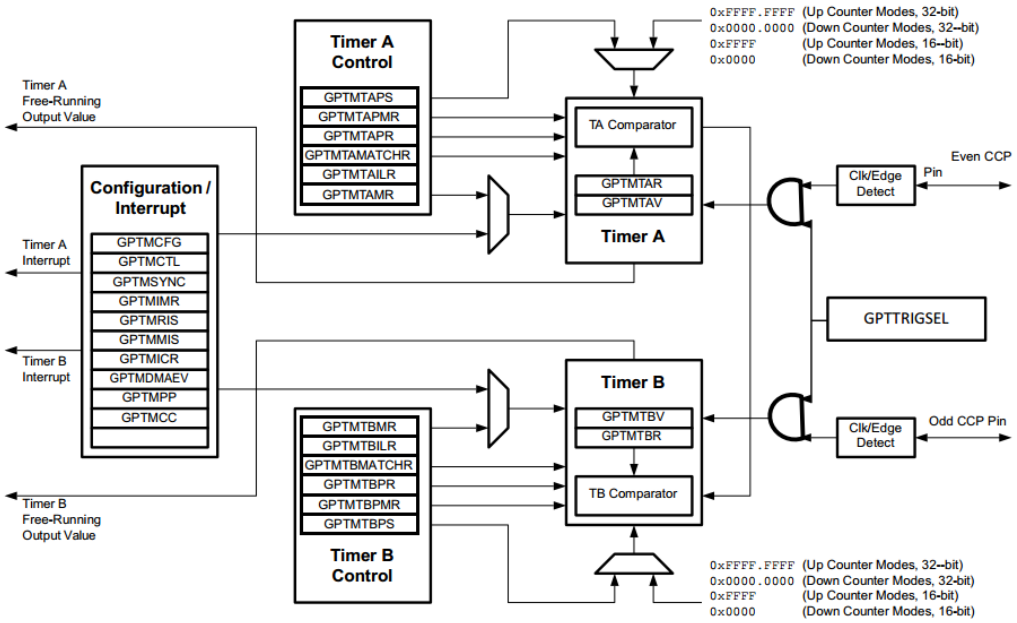

Figure 9-1. GPTM Module Block Diagram

Table 9-1. Available CCP Pins and PWM Outputs/Signals Pins

Timer	Up/Down Counter	Even CCP Pin	Odd CCP Pin	PWM Outputs/Signals
16/32-Bit Timer 0	Timer A	GT_CCP00	-	PWM_OUT0
	Timer B	-	GT_CCP01	PWM_OUT1
16/32-Bit Timer 1	Timer A	GT_CCP02	-	PWM_OUT2
	Timer B	-	GT_CCP03	PWM_OUT3
16/32-Bit Timer 2	Timer A	GT_CCP04	-	
	Timer B	-	GT_CCP05	PWM_OUT5
16/32-Bit Timer 3	Timer A	GT_CCP06	-	PWM_OUT6
	Timer B	-	GT_CCP07	PWM_OUT7

一般定时器，我们关心的参数：

位数：16bit 和 32bit

时钟：80MHz

分频系数：

计数方向：向下或者向上

周期性：单次的还是周期性触发的。

中断设置。

软件功能：实现简单的周期性的向下计数方式的定时器功能，实现了 16bit 定时器和 32bit 定时器。16bit 定时器支持分频功能，32bit 定时器支持不分频功能，在中断函数我们实现了 IO 口翻转功能，可以使用示波器等仪器观察，这里我使用的是逻辑分析仪。

分频后的时钟频率 = 80Mhz/(1+分频系数)，最大的分频系数为 0xff；

周期 = （1/时钟频率）×（1+溢出值）

向下模式介绍： 定时器给一个初值，定时器的值一直向下递减，到达 0，触发了 time—out 中断，在周期性中，会自动导入初值。

向上模式介绍：不怎么了解

在 CCS debug 中我发现总是会先触发中断（定时器刚开始运行，按

道理是不可能会触发中断的），但是我把代码下载 flash 中就没有这个问题。这个问题困扰了我好久，楞是想不明白怎么会这样。。。

附上代码

Main 函数

```
//是否使用32bit的计数器
#define USE_TIMER_32BIT  0
int main(void) {
    unsigned int i;
    unsigned int arr[50];
    //
    // Initialize board configurations
    BoardInit();
    GPIO_INIT();
    uart0_Init();
    //使能timer的时钟
    PRCMPeripheralClkEnable(PRCM_TIMERA0, PRCM_RUN_MODE_CLK);
    //软复位定时器
    PRCMPeripheralReset(PRCM_TIMERA0);
#if USE_TIMER_32BIT
    //设置定时器A的位数为32BIT，向上计数,周期性触发
    TimerConfigure(TIMERA0_BASE,TIMER_CFG_PERIODIC);
    //32bit不支持分频，所以这里设置0
    //TimerPrescaleSet(TIMERA0_BASE,TIMER_A,79);
    TimerPrescaleSet(TIMERA0_BASE,TIMER_A,0);
    //100000 × 0.0125us == 1.25ms周期
    TimerLoadSet(TIMERA0_BASE,TIMER_A, 99999);
#else
    //设置定时器A的位数为16BIT，向下计数，周期性触发
    TimerConfigure(TIMERA0_BASE,TIMER_CFG_A_PERIODIC
|TIMER_CFG_SPLIT_PAIR);
    //设置分频系数：79,,80Mhz 时钟频率为1Mhz
    TimerPrescaleSet(TIMERA0_BASE,TIMER_A,79);
    //设置溢出值，10000.10ms触发一次中断
    //10000 * 1us = 10ms
    TimerLoadSet(TIMERA0_BASE,TIMER_A, 9999);
#endif

    //设置中断函数
    TimerIntRegister(TIMERA0_BASE,TIMER_A,TimerBaseIntHandler);
```

```
    //使能定时器A溢出中断
    TimerIntEnable(TIMERA0_BASE,TIMER_TIMA_TIMEOUT);
    //使能定时器A
    TimerEnable(TIMERA0_BASE,TIMER_A);
#if 0
    for ( i = 0;i < 50;i++) {
        arr[i]= TimerValueGet(TIMERA0_BASE,TIMER_A);

    }
    for ( i = 0;i < 50;i++) {
        my_printf("%d\r\n",arr[i]);

    }
#endif
    //my_printf("%d\r\n",TimerValueGet(TIMERA0_BASE,TIMER_A));
    while (1) {
        //delay(0xf);
        my_printf("%d\r\n",TimerValueGet(TIMERA0_BASE,TIMER_A));
        //TimerDisable(TIMERA0_BASE,TIMER_A);

    }
}
```

中断处理函数

```
TimerBaseIntHandler(void)
{
 int i;
    unsigned long ulInts;
    ulInts = TimerIntStatus(TIMERA0_BASE,true);
    TimerIntClear(TIMERA0_BASE,ulInts);
#if 1
    ulInts = (GPIOPinRead(GPIOA1_BASE,GPIO_PIN_1)&GPIO_PIN_1)?1:0;
    if (ulInts == 1) {

     GPIOPinWrite(GPIOA1_BASE,GPIO_PIN_1,0);
    }else {

     GPIOPinWrite(GPIOA1_BASE,GPIO_PIN_1,GPIO_PIN_1);
    }
   // i = TimerValueGet(TIMERA0_BASE,TIMER_A);
```

```
#endif
   // my_printf("+%d+\r\n",ulInts);
        //TimerDisable(TIMERA0_BASE,TIMER_A);
    //  my_printf("+++%d\r\n",i);
}
```