MSP430程序库<十一>定时器 TA 的 PWM 输出

定时器是单片机常用的其本设备，用来产生精确计时或是其他功能；msp430的定时器不仅可以完成精确定时，还能产生 PWM 波形输出，和捕获时刻值(上升沿或是下降沿到来的时候)。这里完成一个比较通用的 PWM 波形产生程序。

1.硬件介绍：

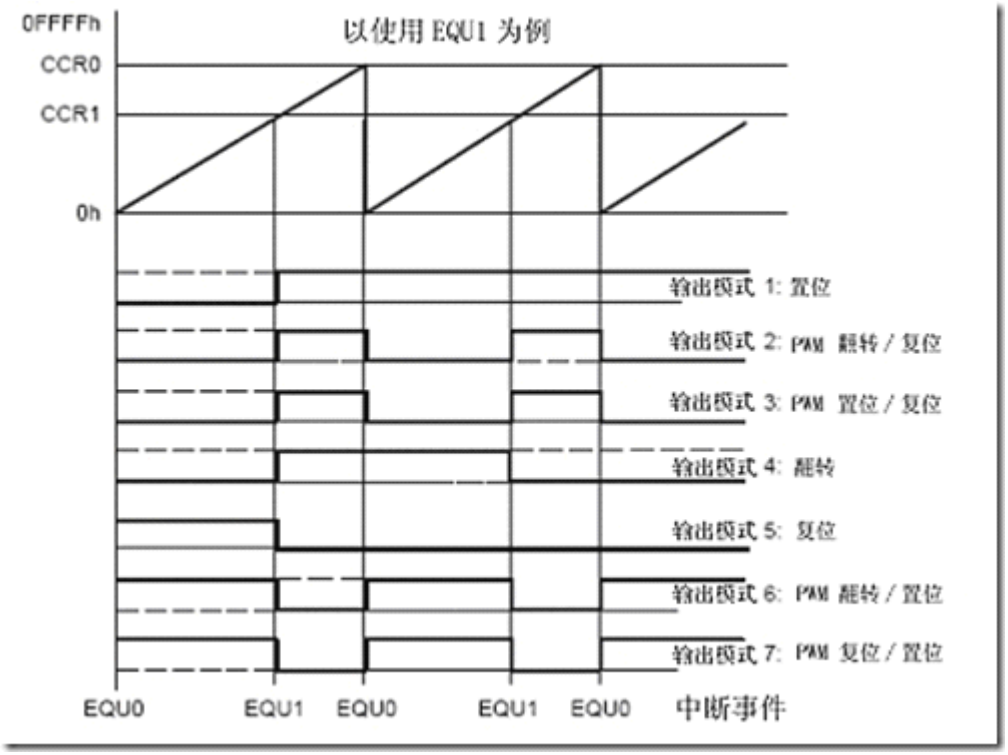
MSP430系列单片机的 TimerA 结构复杂，功能强大，适合应用于工业控制，如数字化电机控制，电表和手持式仪表的理想配置。它给开发人员提供了较多灵活的选择余地。当 PWM 不需要修改占空比和时间时，TimerA 能自动输出 PWM，而不需利用中断维持 PWM 输出。

MSP430F16x 和 MSP430F14x 单片机内部均含有两个定时器，TA 和 TB；TA 有三个模块，CCR0-CCR2；TB 含有 CCR0-CCR67个模块；其中 CCR0模块不能完整的输出 PWM 波形(只有三种输出模式可用);TA 可以输出完整的2路 PWM 波形；TB 可以输出6路完整的 PWM 波形。

定时器的 PWM 输出有有8种模式：

输出模式0　输出模式：输出信号 OUTx 由每个捕获/比较模块的控制寄存器 CCTLx 中的 OUTx 位定义，并在写入该寄存器后立即更新。最终位 OUTx 直通。
输出模式1 置位模式：输出信号在 TAR 等于 CCRx 时置位，并保持置位到定时器复位或选择另一种输出模式为止。
输出模式2 PWM 翻转/复位模式：输出在 TAR 的值等于 CCRx 时翻转，当 TAR 的值等于 CCR0时复位。
输出模式3 PWM 置位/复位模式：输出在 TAR 的值等于 CCRx 时置位，当 TAR 的值等于 CCR0时复位。
输出模式4 翻转模式：输出电平在 TAR 的值等于 CCRx 时翻转，输出周期是定时器周期的2倍。
输出模式5复位模式：输出在 TAR 的值等于 CCRx 时复位，并保持低电平直到选择另一种输出模式。
输出模式6PWM 翻转/置位模式：输出电平在 TAR 的值等于 CCRx 时翻转，当 TAR 值等于 CCR0时置位。
输出模式7PWM 复位/置位模式：输出电平在 TAR 的值等于 CCRx 时复位，当 TAR 的值等于 CCR0时置位。

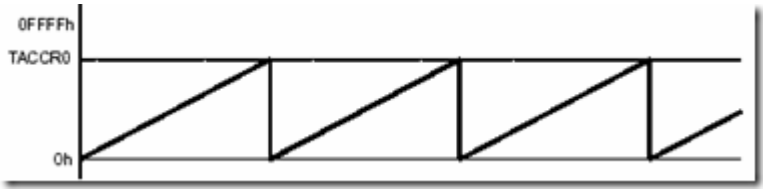
下图是增计数模式下的输出波形(本程序使用的是增模式3和7)：



计数模式：

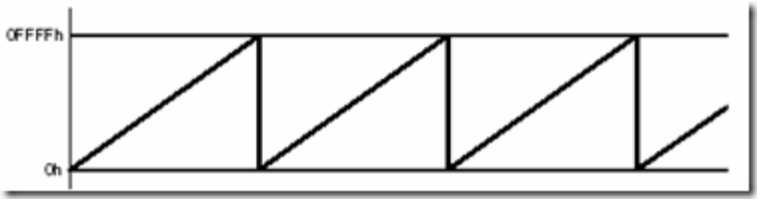
增计数模式

捕获/比较寄存器 CCR0用作 Timer_A 增计数模式的周期寄存器，因为 CCR0为16位寄存器，所以该模式适用于定时周期小于65 536的连续计数情况。计数器 TAR 可以增计数到CCR0的值，当计数值与 CCR0的值相等(或定时器值大于 CCR0的值)时，定时器复位并从0开始重新计数。



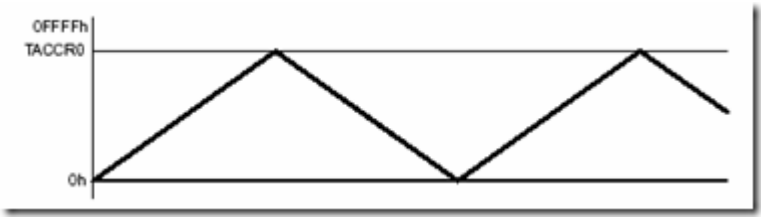
连续计数模式

在需要65 536个时钟周期的定时应用场合常用连续计数模式。定时器从当前值计数到0FFFFH 后，又从0开始重新计数



增/减计数模式

需要对称波形的情况经常可以使用增/减计数模式，该模式下，定时器先增计数到 CCR0的值，然后反向减计数到0。计数周期仍由 CCR0定义，它是 CCR0计数器数值的2倍。



TA 定时器有比较、捕获两种工作方式；比较可以产生 PWM 波形等，捕获可以精确的测量时间；这里用的是比较输出。

硬件介绍就这么多了，其他的可以参考 msp430x1xx_family_users_guide(用户指南)。

2.程序实现：

本程序是直接从 msp430f42x 移植的，只改动了端口就能正常使用了。由此，430的模块在不同的系列中是通用的，有关寄存器是一样的；只是也许外部端口不太一样。

程序初始化部分：完成 TA 相关寄存器的初始化。

```
charTAPwmInit(charClk,charDiv,charMode1,charMode2)
{
 TACTL = 0;                    //清除以前设置
TACTL |= MC_1;                 //定时器 TA 设为增计数模式
switch(Clk)                   //选择时钟源
  {
    case'A': case'a':   TACTL|=TASSEL_1; break;     //ACLK
    case'S': case's':   TACTL|=TASSEL_2; break;     //SMCLK
    case'E':              TACTL|=TASSEL_0; break;     //外部输入(TACLK)
    case'e':              TACTL|=TASSEL_3; break;     //外部输入(TACLK 取反)
    default:   return(0);                             //参数有误
  }
 switch(Div)                  //选择分频系数
  {
    case1:    TACTL|=ID_0; break;    //1
    case2:    TACTL|=ID_1; break;    //2
    case4:    TACTL|=ID_2; break;    //4
    case8:    TACTL|=ID_3; break;    //8
    default:   return(0);            //参数有误
  }
 switch(Mode1)                //设置 PWM 通道1的输出模式。
  {
    case'P':case'p':          //如果设置为高电平模式
      TACCTL1 = OUTMOD_7;      //高电平 PWM 输出
      P1SEL |= BIT2;           //从 P1.2输出 (不同型号单片机可能不一样)
      P1DIR |= BIT2;           //从 P1.2输出 (不同型号单片机可能不一样)
        break;
    case'N':case'n':          //如果设置为低电平模式
      TACCTL1 = OUTMOD_3;      //低电平 PWM 输出
```

```
            P1SEL |= BIT2;            //从 P1.2输出 (不同型号单片机可能不一样)
            P1DIR |= BIT2;            //从 P1.2输出 (不同型号单片机可能不一样)
               break;
        case'0':case0:              //如果设置为禁用
            P1SEL &= ~BIT2;           //P1.2恢复为普通 IO 口
               break;
        default:   return(0);        //参数有误
      }
    switch(Mode2)                     //设置 PWM 通道1的输出模式。
      {
        case'P':case'p':            //如果设置为高电平模式
            TACCTL2 =OUTMOD_7;        //高电平 PWM 输出
            P1SEL |= BIT3;            //从 P1.3输出 (不同型号单片机可能不一样)
            P1DIR |= BIT3;            //从 P1.3输出 (不同型号单片机可能不一样)
               break;
        case'N':case'n':            //如果设置为低电平模式
            TACCTL2 =OUTMOD_3;        //低电平 PWM 输出
            P1SEL |= BIT3;            //从 P1.3输出 (不同型号单片机可能不一样)
            P1DIR |= BIT3;            //从 P1.3输出 (不同型号单片机可能不一样)
               break;
        case'0':case0:              //如果设置为禁用
            P1SEL &= ~BIT3;           //P1.3恢复为普通 IO 口
               break;
        default:   return(0);        //参数有误
      }
       return(1);
}
```

主要是设置 TACTL 寄存器，让 TA 工作于增模式，设置时钟源和分频；CCTLx 设置对应的输出模式；并且打开相应端口的第二功能。

设置周期函数：设置 PWM 波形的周期，单位是多少个 TACLK 周期。

```
voidTAPwmSetPeriod(unsigned intPeriod)
{
    TACCR0 = Period;
}
```

工作于增模式时，TA 计数到 TACCR0,设 CCR0就完成了周期的设置。

设置占空比：设置 TA 的 PWM 输出的有效电平的时间。

```
voidTAPwmSetDuty(charChannel,unsigned intDuty)
{
    switch(Channel)
    {
        case1: TACCR1=Duty; break;
        case2: TACCR2=Duty; break;
    }
```

```
}
```

根据参数分别设置每一路的参数。

设置占空比，用千分比设置：

```
* 入口参数：Channel：当前设置的通道号  1/2
            Percent：PWM 有效时间的千分比 (0~1000)
* 出口参数：无
* 说    明：1000=100.0%  500=50.0% ，依次类推
* 范    例：TAPwmSetPermill(1,300)设置 PWM 通道1方波的占空比为30.0%
            TAPwmSetPermill(2,825)设置 PWM 通道2方波的占空比为82.5%
            */
voidTAPwmSetPermill(charChannel,unsigned intPercent)
{
    unsigned long intPeriod;
    unsigned intDuty;
    Period = TACCR0;
    Duty = Period * Percent / 1000;
    TAPwmSetDuty(Channel,Duty);
}
```

这个函数用千分比来设置 PWM 输出的有效时间。方便程序的使用。

有关定时器，TI 提供的大量的例程，这些历程都很简洁、清晰。需要其他功能可以自己根据例程编写对应的程序。程序实现就这么多了，下面说下本程序的使用方法。

3.使用示例：

使用方式：依然是在工程中加入 c 文件；文件包含 h 头文件；然后就可以正常使用本函数了。详细参考示例工程和 main.c。

main 主要程序如下：

```
#include"msp430x16x.h"      //430寄存器头文件
#include"TAPwm.h"           //TA PWM 输出程序库头文件

voidmain()
{
 // Stop watchdog timer to prevent time out reset
   WDTCTL = WDTPW + WDTHOLD;
    ClkInit();

 TAPwmInit('A',1,'P','P');    //将定时器 TA 初始化成为 PWM 发生器
                  //时钟源=ACLK ；无分频；  通道1和通道2均设为高电平模式。
TAPwmSetPeriod(500);          //通道1/2的 PWM 方波周期均设为500个时钟周期
TAPwmSetDuty(1,200);          //1通道  有效200个时钟周期
TAPwmSetPermill(2,200);       //2通道  20.0%

   LPM0;
}
```

本程序调用程序库，产生两路 PWM 波形。

TA 的 PWM 输出就到这儿了，如果需要更多路的 PWM 波，可以使用 TB，他可以产生6路完整的 PWM 波形；可以参考本程序编写 TB 的波形输出程序。

相关文章及附件下载：http://www.ideyi.org/bbs/article_1077_372721.html