YC3121PWM 拓展使用说明

1. 3121 PWM/TIMER 实现原理

PWM 有 PCNT 和 NCNT 两个寄存器,PCNT 作为 PWM 的高电平持续时间,NCNT 作为 PWM 的低电平持续时间当启动 PWM 后 PWM 会重载 PCNT 或 NCNT 的值到 CNT 里面(由初始电平决定),并开始用系统时钟进行计数,当 CNT 为 0 的时候,翻转 IO 电平,并重载另一个 CNT 值(PCNT 或 NCNT)到 CNT 里面,pwm 的初始电平可以配置为高电平或低电平,当配置好高低电平之后,IO 会立马变成高电平或低电平(此时可能 PWM 还未启动,用于在不使用 PWM 的时候来控制 IO 电平),然后在 PWM 启动之后重载高电平的持续时间,(所以若是高低电平的配置和 PWM 的启动不是在一个时刻,则会观察到 pwm 的第一个周期会比预测的要大)

TIMER 模式则是会重载 PCNT 的值到 CNT 里面,当 CNT 为 0 的时候会触发 TIMER 中断。若是配置了 TIMER 的自动重载，则当 CNT 为 0 的时候会自动重载 PCNT 里面的值到 CNT 里面，从而实现定时多次连续触发。所以改 PCNT 的值需要在下一个周期才能实现。

Q:多个 PWM 为什么不能同步?

A:若是 PWM 不是在同一时间启动的则不能同步,PWM0~PWM7 的控制寄存器在同一个 32BIT 内,可以先配置好 PWM,然后再同时启动

Q:PWM 改了占空比过后为什么不能与未改占空比的 PWM 同步?

A:PWM 若是改占空比在一个周期的头部(比如高电平持续时间 9000 个 CLK,低电平持续时间 600 个 clk,需要把占空比改变为高电平持续时间 600 个 CLK,低电平持续时间 9000 个 CLK)，则若是在高电平持续时间内，改变 PCNT 和 NCNT 的值，就会出现改变后的第一个周期高电平持续时间和低电平持续时间都为 9000 个 CLK 从而造成偏移。

解决方案：(1)在一个周期尾部改变占空比（举上面的例子，则是需要在低电平持续时间的 600 个 CLK 期间改变占空比，可用 TIMER 实现，或者通过读取 IO 电平来判断现在处于什么周期再改变占空比）

(2) 关闭 PWM 并把 PWM 设置为 TIMER 模式(这样可以清除 CNT 值),然后配置好占空比再启动 PWM 即可

Q:PWM 互补怎么实现?

A:两个 PWM PCNT 和 NCNT 设置一样,然后一个 PWM 初始化为高电平,一个 PWM 初始化为低电平,再同时启动即可

Q:PWM 如何实现定时定点 ADC 值采样

A:需要使用一个 TIMER,(timer 的周期需要比 pwm 的周期多一个 CLK),然后在 TIMER 里面采样,使用 ADC 采样

2. 参考代码

```
/*本示例代码使用了 PWM0~2 共三个,TIMER3 共一个,实现 PWM 的定时定点采样*/
void PWM_enable()
{
    uint32_t buff = 0;
    TIM_CTRL = 0x4444;//关闭所有 PWM0~2 和 TIMER3 并全部初始化为 TIMER 以清空 CNT 寄存器的值
    g_rpcnt = 9000 ;     g_rncnt = 600 ;//全局变量用于传递 PWM 占空比的参数
    g_rpcnt1 = g_rpcnt | (g_rncnt<<16);/*全局变量用于传递 PWM 占空比的参数(实际传递给 TIMER 的值在这里上面的参数用于在其他地方修改)
                                        这里这样写的目的在于能够只使用一条指令就能够传递出参数,防止被中断打断,产生误差*/
    TIM_PCNT(TIM0) = g_rpcnt;   TIM_NCNT(TIM0) = g_rncnt;
    TIM_PCNT(TIM1) = g_rpcnt;   TIM_NCNT(TIM1) = g_rncnt;
    TIM_PCNT(TIM2) = g_rpcnt;   TIM_NCNT(TIM2) = g_rncnt;//初始化 PWM0~2 的占空比参数
    TIM_PCNT(TIM3) = 9601;//PWM 周期为 9600,则 timer 周期需为 9601
    buff =TIM_CTRL;
    buff |=0x8222;//开启 TIMER 的自动重载,并初始化所有 PWM 为低电平
    buff &= ~0x444;//将 TIMER0~2 初始化为 PWM0~2
    buff |=0x1000;//开启 TIMER
    TIM_CTRL  = buff;//将上面的所有配置一起写入寄存器,TIMER 开始运行    注意 :若是需要实现 PWM 互补,请参考上面的语言描述,类似这样操作即可(配置好寄存器 同时写下去)
    delay_us(5);//延时 5us,此时 MCU 的时钟为 96m,计算出 TIMER 的周期为 100us,计算好时序,这里配置不用的延时,则可实现在任意一点实现 PWM 的 ADC 采样
    TIM_CTRL  |= 0x111;//启动 PWM
}
/*此函数用于改变 PWM 的占空比,实际修改在 TIMER 中断里面,这里只用作传参*/
void PWM_RSable()
{
    if( g_rpcnt <= 9598 ){
        g_rpcnt += 1 ;     g_rncnt = 9600 - g_rpcnt;
        g_rpcnt1 = g_rpcnt | (g_rncnt<<16);
    }else{
        g_rpcnt = 600 ;     g_rncnt = 9000 ;
        g_rpcnt1 = g_rpcnt | (g_rncnt<<16);
    }
}
void TIMER3_IRQHandler(void)
{
    uint32_t g_rpcnt2, g_rncnt2;
    GPIO_CONFIG(30) = OUTPUT_HIGH;//拉 IO 示波器查看
```

```
38.     g_rpcnt2 = g_rpcnt1&0xffff;
39.     g_rncnt2 = (g_rpcnt1&0xffff0000)>>16;//解析出参数 并配置好 PWM 实现修改 PWM 占空比
40.     TIM_PCNT(TIM0) = g_rpcnt2;
41.     TIM_NCNT(TIM0) = g_rncnt2;
42.     TIM_PCNT(TIM1) = g_rpcnt2;
43.     TIM_NCNT(TIM1) = g_rncnt2;
44.     TIM_PCNT(TIM2) = g_rpcnt2;
45.     TIM_NCNT(TIM2) = g_rncnt2;
46.     delay_us(1);//模式 ADC 采样
47.     GPIO_CONFIG(30) = OUTPUT_LOW;
48. }
```

3. 实现效果