基于LGTSDK Builder

LGT8F690A 快速开发系列教程

第十篇: TIMER1的使用之PWM篇



本篇为系列教程的第10篇。如果需要了解教程相关的软件硬件环境,请参考本系列教程的第一篇:《LGT8F690A快速开发系列教程第一篇.急速上手》

LGT8F690A的TIMR1是一个功能完整的16位定时/计数器。拥有一个16位的计数器TMR1,一个16位的周期寄存器PR1。TIMER1的时钟源有5种选择:可以同步指令周期,可以直接来自系统时钟,可以来自外部低速晶振输入,可以来自外部引脚T1CKI的输入时钟或者来自内部32KHz低速RC振荡器。TIMER1拥有一个3位预分频器,可以支持1/2,1/4以及1/8分频。

TIMER1的计数器支持门控模式。门控模式下,TIMER1的计数受门控信号的控制。 门控信号可以为外部T1G引脚输入或者来自LGT8F690A内部比较器的输出。可以利用门控功能实现简单的Delta-Sigma ADC转换以及其他多种模拟量的测试。

TIMER1支持编码器模式,可以根据外部输入信号的跳变进行向上或者向下的计数。编码器模式针对伺服电机的应用。

另外,TIMER1是ECCP1的频率源。TIMER1与ECCP1配合工作,实现多路PWM输出或者输入俘获功能。ECCP1支持3通道PWM输出或3通道输入俘获。可以以极少的代码实现多路电机驱动或外部事件的测量。

TIMER1编程

下面我们将使用LGTSDK Builder, 展示如何简单快速的实现TIMER1/ECCP1的相关功能。

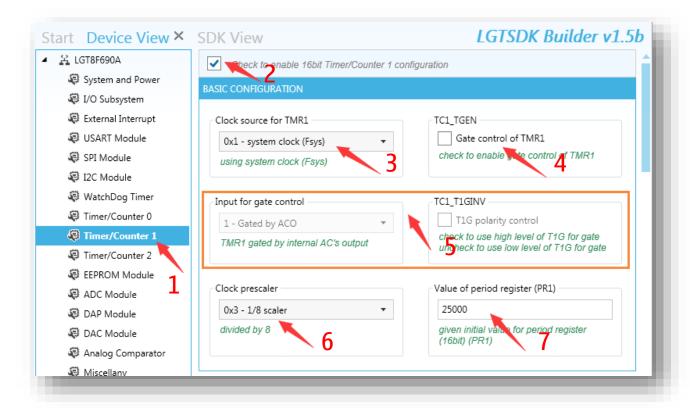
我们将分2个例程演示。 首先一个是最简单的PWM输出与控制。

启动LGTSDK Builder, 新建一个工程, 选择芯片: LGT8F690A, 工程名称: lgt8f690a_tc1

第一个例程是实现一个PWM输出,可外接一个LED闪灯。然后通过使用RCO作为输入,来控制PWM输出与关闭。系统时钟我们使用默认的配置:主时钟源为16MHz内部RC振荡器,主时钟预分频为4,即系统工作时钟为4MHz。

我们计划使用这个4MHz的系统时钟作为TIMER1的计数时钟。 为实现闪灯, 我们实现100ms的 PWM周期,定时器的周期就是50ms, PWM的占空比暂定为50%, 也就是输出一个方波。 下面算下定时器周期PR1的设置: 4MHz的计数频率,就是250ns。 相对于50ms还是比较快的。 我们这里为TIMER1选择1/8的预分频, 这样计数频率就是500KHz, 也就是2us。 50ms的周期需要50000/2 = 25000。占空比寄存器就是12500。

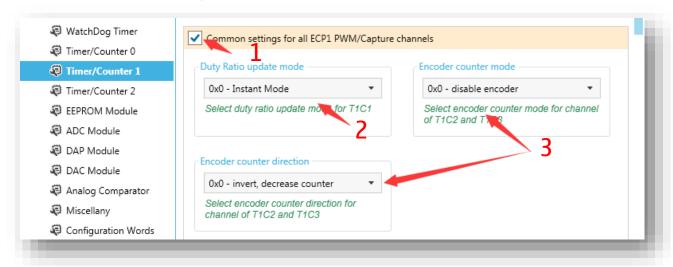
下面我们就直接讲入TIMER1的配置页:



首先是TIMER1的基本配置。 配置说明如下:

- 3:选择TMER1的计数时钟源。这里我们选择是直接来自系统时钟;
- 4: TIMER1计数器的门控使能, 我们此处不用门控, 不需要勾选;
- 5: 这两项都是和门控相关的, 分别是选择门控输入源和门控信号的极性;
- 6: 这里是选择TIMER1预分频,按我们之前的计划,选择8分频;
- 7: 这里是设置TIMER1的16位定时周期, 对于PWM, 这里也相当于设置PWM的频率。根据之前我们的计算, 这里填25000;

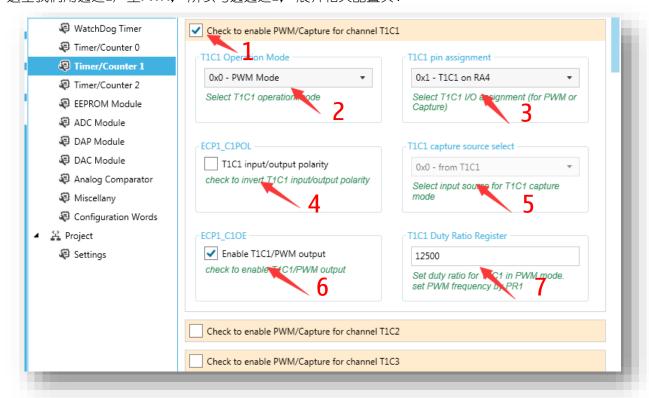
基本设置完成。下面是产生PWM相关的设置。TIMER1的ECCP支持三个PWM/俘获通道,我们可以同时配置输出三路PWM,这里我们只输出一路:



在TIMER1基本配置下面,首先是勾选[Common settings for all ECP1 PWM/Capture channels], 这里是 ECCP1的公用设置, 这里的设置对下面的三个通道都有效。目前我们这里使用默认配置即可, 下面简单介绍下这些设置:

- 1: 首先要勾选这里, 然后我们才能看到相关的设置;
- 2: 这里是设置PWM占空比更新的模式,分为立即模式和溢出更新模式。立即模式就是软件写占空比 寄存器后,立即更新目前的占空比。溢出更新模式是软件写占空比寄存器后,需要在下一个定时 器溢出发生时,才更新占空比寄存器; 这里我们用默认的立即模式;
- 3: 这两项是编码器相关的设置, 我们暂时不用, 所以这里就保持默认;

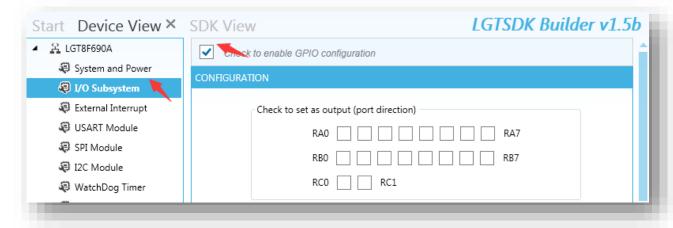
通用设置完成后,下面是PWM/俘获相关的设置。设置分为三个通道,用那个通道勾选那个通道。 这里我们用通道1产牛PWM. 所以勾选通道1. 展开相关配置页:



勾选[Check to enable PWM/Capture of channel T1C1], 这将使能ECCP1的通道1, 并展开相关配置:

- 2: 这里是选择通道1的工作模式, 我们选择PWM模式。这个模式下T1C1将输出PWM;
- 3: 这里选择T1C1的引脚分频,可以选择RB4或者RA4, 此处我们选择RA4;
- 4: 设置T1C1输出的极性, 这个选项对PWM模式和俘获模式都有效;
- 5: 这里是选择通道1俘获模式是的输入源。 此处我们不用俘获模式, 因此选项无效;
- 6: 勾选这里,使能PWM输出。这是PWM输出到端□的控制,必须勾选。勾选后我们可以在 [PINOUT View]视图里看到RA4引脚上的T1C1被标注为输出的颜色;
- 7: 这里是设置PWM的占空比。按之前的计数,我们填12500,是定时器1周期(PR1)的一般,输出的PWM是一个方波。这里你可以可以尝试其他值(不能大于PR1),占空比将会影响LED的亮度。

最后,因为我们需要通过RCO的输入来控制PWM,因此我们需要使用到I/O相关的接口函数。很简单,我们只需要打开[I/O Subsystem]配置也, 勾选它即可, 不用做其他任何选择, 这样就可把I/O相关的 SDK函数接口包含产生的工作中:



所有的配置已完成,下面是代码部分。代码很简单,首先我们不需要为产生PWM写任何代码。SDK已经完成这部分工作。我们只需要完成通过RCO控制PWM输出的部分:

```
LGTSDK Builder v1.5b
SDK View X
main.c ×
7
    // Import external definitions
    extern void init_modules(void);
8
9
    int main(void)
10
11 □ {
        // Device initialization
12
13
        init modules();
14
15
        // global pullup enable
        sysPullupEnable();
16
17
18
        // RC0 pullup enable
19
        gppPullupEnable(GPPC, P0);
                                                 使能RCO内部上拉
20
21
        // Add your code from here
22
23
        while(1)
24 🖃
25
26
            if(gppReadSingle(GPPC, P0) == 0)
27
                t1c1OutputDisable();
28
                                              读RCO的状态,如果RCO为低,关闭PWM
29
                t1c1OutputEnable();
                                              输出;否则打开PWM输出
30
         }
31
```

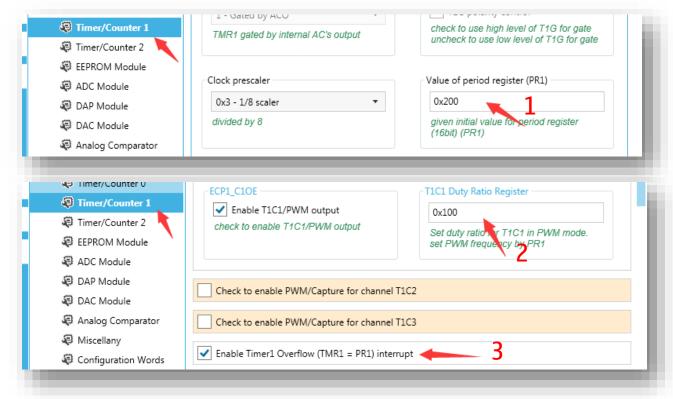
代码完成,编译通过,下载到最小板测试!注意,PWM是在RA4上,请将LED接到RA4上。

下面我们将稍作更改,用TIMER1输出一个占空比可变的PWM,实现一个正儿八经的呼吸灯。

要实现呼吸等,我们需要每个PWM周期逐次更新占空比,这个时间就是定时器的溢出点。我们可以开启TIMER1的溢出中断,在中断中完成占空比的更新!

另外,呼吸等的呼吸是一个周期,这个周期包含了PWM占空比从0调整到最大的全部PWM周期。因此我们还需要更改一下PWM的频率。因为TIMER1是16位的计数器,这次我们可以把PWM的分辨率设置高一点,同时根据之前TIMER0实现呼吸等的计算(请参考本系列教程第四篇),需要需要PWM的周期在4ms左右,在我们之前的系统时钟配置下,我们将定时周期(PR1)设置为0x200。

因为要更新TIMER1的配置,我们回到[Device View]下,对Timer1的配置进行更新:



只需要更新以上三处。 其中第3处, 我们勾选使能定时器1的溢出中断!

重新生产代码。回到[SDK View]。因为我们使能了中断,所以多了一个isr_auto.c文件,这里面包含了中断服务中断,我们产生可变占空比的代码,都将在这个中断服务中经行。

因为是重新生产SDK代码,main.c文件将不会被更新。因此我们还需要在主程序手动增加使能全局中断的代码,这里需要特别注意!!

下面是读main.c的改动, 只需要增加下面一句, 使能全局中断:

```
main.c X
            isr_auto.c
 5
     #include "allinone.h"
 6
 7
     // Import external definitions
 8
     extern void init modules(void);
 9
10
     int main(void)
11
   □{
12
         // Device initialization
13
         init modules();
14
15
         // global interrupt enable
                                          使能全局中断
16
         SEI(); ←
17
18
         // global pullup enable
19
         sysPullupEnable();
```

然后是中断服务程序部分,双击打开isr_auto.c文件,代码如下:

```
LGTSDK Builder v1.5b
                          SDK View X
Start Device View

    Project 'lgt8f690a_tc1'

                                      isr_auto.c X
    Drivers
                                                            占空比增减的方向控制, 对应等的呼或吸
                               static u8 pdir = 0;
                               static u16 duty = 0; ← 调整占空比的变量,注意为16位
    Includes
                           8
                           9
       allinone.h
                          10
                              // Enable Timer1 Overflow (TMR1 = PR1) interrupt
       FSYSCLK.h
                               void interrupt _L_HISR(void)
                          11
                          12 🗏 {
       🗓 global.h
                          13
                                   // TODO: Add your code here
                                  if(tc1InterruptFlag())
                          14
                                                          ◆ 检查是否为TIMER1的溢出中断
       Igt8f690a.h
                          15 🗀
       Igt8f690a_gpp.h
                                       tc1InterruptClear(); ← 清中断标记位
                          16
                          17
       🗓 lgt8f690a_misc.h
                                       if(duty == 0x1ff)
                          18
                                          pdir = 1;
                          19
       🖺 lgt8f690a_sys.h
                                                              占空比增减方向更新
                          20
                                       else if(duty == 0)
       Igt8f690a_tc1.h
                          21
                                          pdir = 0;
                          22
       macros_auto.h
                          23
                                       if(pdir == 0)
      Sources
                          24
                                          t1c1SetDuty(++duty);
                                                                 根据方向标志位设置当前占空比
                          25
                                       else
        Igt8f690a_gpp.c
                                          t1c1SetDuty(--duty);
                          26
                          27
                                  }
        Igt8f690a_misc.c
                          28
                          29 -}
        🗓 lgt8f690a_sys.c
       igt8f690a_tc1.c
    init auto.c
                           xic2mic: code convert successful!
    isr_auto.c
                           xic2mic: generate mic8-hex file successful!
    main.c
                           xic2mic: generate mic8-abin file successful!
```

TIMER1编程

代码编写完成, 编译通过后, 下载到最小开发板板上测试!!

最后,我们列一下本篇教程中新使用的到的SDK接□函数:

函数名称	功能描述	使用方法
t1c10utputEnable()	使能T1C1上的PWM输出	t1c10utputEnable();
t1c1OutputDisable()	关闭T1C1上的PWM输出	t1c1OutputDisable();
tc1InterruptFlag()	返回定时器1的溢出中断标志位	返回1: TIMER1发送溢出中断
tc1InterruptClear()	清定时器1的溢出中断标志位	tc1InterruptClear()
t1c1SetDuty()	设置T1C1通道PWM输出的占空比	t1c1SetDuty(0x100) 设置t1c1的PWM输出占空比寄存器为0x100 占空比的设置不能大于定时器周期设置
SEI()	使能全局中断	也可以使用SDK中的另外一个函数: gie_enable()
CLI()	禁止全局中断	也可以使用SDK中的另外一个函数: gie_disable()