STM32之呼吸灯实验

原创

Andrew_Qian 于 2017-05-08 11:44:54 发布 4713 收藏 19
分类专栏： stm32

本文链接：https://blog.csdn.net/m0_37655357/article/details/71404843

首先，我想引用一下在一片博文里 看到 的一段话，写的很详细，

首先来说，你要使用PWM模式你得先选择用那个定时器来输出PWM吧！除了TIM6、TIM7这两个普通的定时器无法输出PWM外，其余的定时器都可以输出PWM，每个通用定时器可以输出4路PWM，高级定时器TIM1、TIM8每个可输出7路PWM，这里为了方便起见，我们选择与实验相同的TIM3的通道2来说明。选好定时器及通道后，下一步就是要使能定时器的时钟，根据需要看看是否需要重映射IO,然后就是配置输出PWM的IO及定时器，到这里原子的视频及例程都有详细的介绍，这里只需要提一点有些网友疑惑的TIM_TimeBaseStructure.TIM_ClockDivision = 0;这句话是什么作用？其实仔细看过技术手册后发现这句话与PWM输出实验其实是没关系的，这句话是设置定时器时钟(CK_INT)频率与数字滤波器(ETR，TIx)使用的采样频率之间的分频比例的（与输入捕获相关），0表示滤波器的频率和定时器的频率是一样的。至于其余部分，我就不再赘述。

那么，下面就贴上我自己的代码

这个代码可是我改了一天的结果啊。。。。

```
/*****************************************
利用pwm控制led亮度，
其实我们看到的是灯在呼吸，实际上灯是一直在以很高的频率在闪烁，每次闪烁的亮度都不一样，越来越亮，或者暗
由于人的视觉暂留效应，我们看到灯一直在亮，而且亮度在渐变。

日期：2016.2.25

*****************************************/

#include "stm32f10x.h"



/* LED亮度等级 PWM表 */
uint8_t indexWave[] = {1,1,2,2,3,4,6,8,10,14,19,25,33,44,59,80,
    107,143,191,255,255,191,143,107,80,59,44,33,25,19,14,10,8,6,4,3,2,2,1,1};


//函数申明

        void Init_LED(void);
        void NVIC_Config_PWM(void);
        void Init_TIMER(void);
        void Init_PWM(void);
        void Delay_Ms(uint16_t time);
```

```
        void Delay_ms(uint16_t time);
        void Delay_Us(uint16_t time);



int main(void)
{
            SystemInit();                    //系统时钟配置

            Init_LED();                        //LED初始化
            NVIC_Config_PWM();
            Init_TIMER();                    //定时器初始化
            Init_PWM();            //PWM初始化设置
            GPIO_SetBits(GPIOG,GPIO_Pin_14);    // LED D2 输出为高

            while(1);

}



void Init_LED(void)
{
      GPIO_InitTypeDef GPIO_InitStructure;                    //定义一个GPIO结构体变量

      RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD| RCC_APB2Periph_GPIOG
|RCC_APB2Periph_AFIO,ENABLE);
                                                          //使能各个端口时钟，重要！！！

      GPIO_InitStructure.GPIO_Pin =  GPIO_Pin_14;             //配置LED D2端口挂接到PG14端口
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;           //通用输出推挽
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;           //配置端口速度为50M
      GPIO_Init(GPIOG, &GPIO_InitStructure);                        //将端口GPIOD进行初始化配置

      GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13 ;             //配置LED D5端口挂接到13端口
      GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;               //复用功能输出推挽，这是重映射必要地，AF表示
复用
      GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;           //配置端口速度为50M
      GPIO_Init(GPIOD, &GPIO_InitStructure);                        //将端口GPIOD进行初始化配置

      GPIO_PinRemapConfig(GPIO_Remap_TIM4,ENABLE);            //将定时器4通道2重映射到PD13引脚，重要！！
}                                                       //我看很多资料都是这样映射的



//还是野火比较专业，人的吸气呼气通常用时为3秒，算出来是这样的
//TIM_BaseInitStructure.TIM_Period = 256-1;
//TIM_BaseInitStructure.TIM_Prescaler = 2000-1;


void Init_TIMER(void)
{

    TIM_TimeBaseInitTypeDef     TIM_BaseInitStructure;            //定义一个定时器结构体变量

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM4, ENABLE);       //使能定时器4，重要！！

//    TIM_DeInit(TIM4);                                          //将TIM4定时器初始化位复位值
```

```
//    TIM_InternalClockConfig(TIM4);                                    //配置 TIM4 内部时钟

    //TIM_BaseInitStructure.TIM_Period = 7200-1;          //设置自动重载寄存器值为最大值     0~65535之间
1000000/1000=1000us=1ms
                                                                        //TIM_Period（TIM1_ARR）=7200，计数
器向上计数到7200后产生更新事件，
                                                                    //计数值归零 也就是 1MS产生更新事件一次

    TIM_BaseInitStructure.TIM_Period = 256-1;

    TIM_BaseInitStructure.TIM_Prescaler = 2000-1;  //自定义预分频系数为0，即定时器的时钟频率为72M提供给定时器的时钟
0~65535之间


    TIM_BaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1; //时钟分割为0

    TIM_BaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up;   //TIM向上计数模式 从0开始向上计数，计数到1000
后产生更新事件

    TIM_TimeBaseInit(TIM4, &TIM_BaseInitStructure);          //根据指定参数初始化TIM时间基数寄存器

     TIM_ARRPreloadConfig(TIM4, ENABLE);        //使能TIMx在 ARR 上的预装载寄存器

    TIM_Cmd(TIM4, ENABLE);          //TIM4总开关：开启

    TIM_ITConfig(TIM4,TIM_IT_Update,ENABLE);    //使能定时器中断

    NVIC_Config_PWM();   //配置中断优先级


}



void Init_PWM()
{
    TIM_OCInitTypeDef  TIM_OCInitStructure;                    //定义一个通道输出结构

    TIM_OCStructInit(&TIM_OCInitStructure);                    //设置缺省值

    TIM_OCInitStructure.TIM_Pulse = 0;              //设置占空比，占空比=(CCRx/ARR)*100%或
(TIM_Pulse/TIM_Period)*100%
                                                                    //PWM的输出频率为Fpwm=72M/7200=1Mhz；

    /* 下面五句话就把PWM 基本上配置完成，再加上上面定时器的使能，TIM_Cmd(TIM4,ENABLE)   */

    TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;           //PWM 模式 1 输出      ,

    TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;    //使能输出状态  需要PWM输出才需要这行代码

    TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;    //TIM 输出比较极性高 ，就是开始     计数到ccr之
前都是high，因为这里的led是正逻辑点亮

  TIM_OC2Init(TIM4, &TIM_OCInitStructure);              //根据参数初始化PWM寄存器 ，使能通道CCR2

    TIM_OC2PreloadConfig(TIM4,TIM_OCPreload_Enable);           //使能 TIMx在 CCR2 上的预装载寄存器,很关键和容易
遗漏的一部

  //TIM_CtrlPWMOutputs(TIM4,ENABLE);                          //设置TIM4 的PWM 输出为使能
```

```
}


void NVIC_Config_PWM(void)      //配置嵌套向量中断控制器NVIC
{

      NVIC_InitTypeDef NVIC_InitStructure;

        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);    //设置中断优先级分组2

        NVIC_InitStructure.NVIC_IRQChannel = TIM4_IRQn;    //设定中断源为PC13

        NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;    //中断占优先级为0

        NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;    //副优先级为0

        NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;    //使能中断

        NVIC_Init(&NVIC_InitStructure);    //根据参数初始化中断寄存器

}


void TIM4_IRQHandler(void)    //中断入口函数
{
    static uint8_t pwm_index = 0;            //用于PWM查表
    static uint8_t period_cnt = 0;        //用于计算周期数

    if (TIM_GetITStatus(TIM4, TIM_IT_Update) != RESET)    //TIM_IT_Update
     {
            period_cnt++;
            if(period_cnt >= 10)                                        //若输出的周期数大于10，输出下一种脉冲宽
的PWM波,在这里野火说他也不知道为什么大于10
            {

                TIM4->CCR2 = indexWave[pwm_index];    //根据PWM表修改定时器的比较寄存器值
                pwm_index++;                                                //标志PWM表的下一个元素

                if( pwm_index >=  40)                                    //若PWM脉冲表已经输出完成一遍，重置PWM查表
标志
                {
                    pwm_index=0;
                }

                period_cnt=0;                                                //重置周期计数标志
        }

        TIM_ClearITPendingBit (TIM4, TIM_IT_Update);    //必须要清除中断标志位

    }
}



    /*:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
** 函数名称: Delay_Ms_Ms
** 功能描述: 延时1MS (可通过仿真来判断他的准确度)
** 参数描述: time (ms) 注意time<65535
```

```
** 作    者: Dream
** 日    期: 2011年6月20日
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
void Delay_Ms(uint16_t time)  //延时函数
{
    uint16_t i,j;
    for(i=0;i<time;i++)
          for(j=0;j<10260;j++);
}
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
** 函数名称: Delay_Ms_Us
** 功能描述: 延时1us (可通过仿真来判断他的准确度)
** 参数描述: time (us) 注意time<65535
** 作    者: Dream
** 日    期: 2011年6月20日
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
void Delay_Us(uint16_t time)  //延时函数
{
    uint16_t i,j;
    for(i=0;i<time;i++)
          for(j=0;j<9;j++);
}
/*::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
End:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D:-D
::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::*/
```