


STM32F103 使用TIM3产生四路PWM

STM32F103 使用TIM3产生四路PWM

程序如下：



```

/*****
* 程序说明      : 思路PWM波生成函数
* 函数功能      : 使用TIM3的PWM功能生成思路PWM,
* 输    入      : 无
* 输    出      : 四路PWM, 通过GPIO引脚复用, 对TIM3的四个输出通道引脚重映射为PC6、PC7、PC8、PC9
*****/

#include "stm32f10x.h"

void RCC_Cfg(void);
void GPIO_Cfg(void);
void TIM_Cfg(void);
void NVIC_Cfg(void);
void delay_ms(u32 i);
void PWM_Cfg(float dutyfactor1,float dutyfactor2,float dutyfactor3,float dutyfactor4);

int main()
{
    u8 flag = 1;
    float ooo=0.5;
    RCC_Cfg();
    NVIC_Cfg();
    GPIO_Cfg();
    TIM_Cfg();

    //开启定时器2
    TIM_Cmd(TIM3,ENABLE);

    //呼吸灯
    while(1){
        PWM_Cfg(ooo,10,50+0.5*ooo,200-2*ooo);

        if(flag == 1)
        {
            ooo=ooo+0.002;
        }
        if(flag == 0)
        {
            ooo=ooo-0.002;
        }
        if(ooo>100){
            flag = 0;
        }
        if(ooo<0.5)
        {
            flag = 1;
        }
    }
}
```

公告

昵称: jiwangbujia
园龄: 3年7个月
粉丝: 25
关注: 30
+加关注

<		2019年10月						>		
日	一	二	三	四	五	六				
29	30	1	2	3	4	5				
6	7	8	9	10	11	12				
13	14	15	16	17	18	19				
20	21	22	23	24	25	26				
27	28	29	30	31	1	2				
3	4	5	6	7	8	9				

搜索

常用链接

我的随笔
我的评论
我的参与
最新评论
我的标签

我的标签

IOS (29) IOS开发 (21)
UI (15) C语言 (13) oc (10)
STM32 (8) VB.NET (8)
MySQL (7) 数据库 (5)
单片机 (4) 更多

随笔档案

2017年4月(1)
2016年10月(1)
2016年8月(2)
2016年7月(1)
2016年6月(17)
2016年5月(27)
2016年4月(23)
2016年3月(39)
2016年2月(1)

最新评论

1. Re:超市RFID结算系统项...
好!!!

```

    }
}

void GPIO_Cfg(void)
{

    GPIO_InitTypeDef GPIO_InitStructure;

//RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC|RCC_APB2Periph_GPIOA|RCC_APB2Periph_AFIO,
ENABLE);

//全部映射, 将TIM3_CH2映射到PB5
//根据STM32中文参考手册2010中第119页可知:
//当没有重映射时, TIM3的四个通道CH1, CH2, CH3, CH4分别对应PA6, PA7,PB0,PB1
//当部分重映射时, TIM3的四个通道CH1, CH2, CH3, CH4分别对应PB4, PB5,PB0,PB1
//当完全重映射时, TIM3的四个通道CH1, CH2, CH3, CH4分别对应PC6, PC7,PC8,PC9
//也即是说, 完全重映射之后, 四个通道的PWM输出引脚分别为PC6, PC7,PC8,PC9, 我们用到了通道1和通道2, 所以对应引脚为PC6, PC7,PC8,PC9, 我们用到了通道1和通道2, 所以对应引脚为
GPIO_PinRemapConfig(GPIO_FullRemap_TIM3, ENABLE);

//部分重映射的参数
//GPIO_PinRemapConfig(GPIO_PartialRemap_TIM3, ENABLE);

//设置PC6、PC7、PC8、PC9为复用输出, 输出4路PWM
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_6|GPIO_Pin_7|GPIO_Pin_8|GPIO_Pin_9;
GPIO_Init(GPIOC,&GPIO_InitStructure);

}

void TIM_Cfg(void)
{
//定义结构体
TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;

//重新将Timer设置为缺省值
TIM_DeInit(TIM3);
//采用内部时钟给TIM2提供时钟源
TIM_InternalClockConfig(TIM3);

//预分频系数为0, 即不进行预分频, 此时TIMER的频率为72MHzre.TIM_Prescaler =0;
TIM_TimeBaseStructure.TIM_Prescaler = 0;
//设置计数溢出大小, 每计20000个数就产生一个更新事件
TIM_TimeBaseStructure.TIM_Period = 7200 - 1;
//设置时钟分割
TIM_TimeBaseStructure.TIM_ClockDivision = TIM_CKD_DIV1;
//设置计数器模式为向上计数模式
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;

//将配置应用到TIM2中
TIM_TimeBaseInit(TIM3,&TIM_TimeBaseStructure);
//清除溢出中断标志
//TIM_ClearFlag(TIM2, TIM_FLAG_Update);
//禁止ARR预装载缓冲器
//TIM_ARRPreloadConfig(TIM2, DISABLE);
//开启TIM2的中断
//TIM_ITConfig(TIM2,TIM_IT_Update,ENABLE);

}

/*****
* 函 数 名      : PWM波产生配置函数
* 函数功能      : PWM_Cfg
* 输    入      : dutyfactor 占空比数值, 大小从0.014到100
* 输    出      : 无
*****/
void PWM_Cfg(float dutyfactor1,float dutyfactor2,float dutyfactor3,float dutyfactor4)
{

```

2. Re:开发底层硬件应该怎...
有专门写接口文档的工具的, 易文档

--dongdonggo

3. Re:VS2010环境下使用V...
我是一个程序小白, 尝试学习你的程序, 调试时串口有加载, 但不能打开串口, , 点击调试查看显示无可来源, ***调用堆栈位置: > 串口测试.exe!串口测试.My.MyApplication.Main(Str...

--zengjiadong

4. Re:开发底层硬件应该怎...
邮箱: 774578884@qq.com

--本尊... | 蛋疼 |

5. Re:开发底层硬件应该怎...
高手你好, 把你用的开发包SDK可以给我传一份吗?

--本尊... | 蛋疼 |

阅读排行榜

- STM32精确延迟1us和1...
- STM32F103 使用TIM3...
- C语言 - if语句(14095)
- IOS开发中重写init方法使...
- C#中Console.WriteLine...

评论排行榜

- 小规模软件开发团队现存...
- 开发底层硬件应该怎么编...
- 数据可视化-使用EXCEL和...
- IOS开发中重写init方法使...
- 超市RFID结算系统项目进...

推荐排行榜

- IOS开发中重写init方法使...
- 小规模软件开发团队现存...
- IOS开发中使用CNConta...
- IOS开发中UITableView...
- C语言 - switch语句(1)

```

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

//TIM3的CH1输出
TIM_OCInitStructure.TIM_OCMode      = TIM_OCMode_PWM1; //设置是PWM模式还是比较模式
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //比较输出使能, 使能PWM
输出到端口
TIM_OCInitStructure.TIM_OCPolarity  = TIM_OCPolarity_High; //设置极性是高还是低
//设置占空比, 占空比=(CCRx/ARR)*100%或(TIM_Pulse/TIM_Period)*100%
TIM_OCInitStructure.TIM_Pulse = dutyfactor1 * 7200 / 100;
TIM_OC1Init(TIM3, &TIM_OCInitStructure);

//TIM3的CH2输出
TIM_OCInitStructure.TIM_OCMode      = TIM_OCMode_PWM1; //设置是PWM模式还是比较模式
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //比较输出使能, 使能PWM
输出到端口
TIM_OCInitStructure.TIM_OCPolarity  = TIM_OCPolarity_High; //设置极性是高还是低
//设置占空比, 占空比=(CCRx/ARR)*100%或(TIM_Pulse/TIM_Period)*100%
TIM_OCInitStructure.TIM_Pulse = dutyfactor2 * 7200 / 100;
TIM_OC2Init(TIM3, &TIM_OCInitStructure);

//TIM3的CH3输出
TIM_OCInitStructure.TIM_OCMode      = TIM_OCMode_PWM1; //设置是PWM模式还是比较模式
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //比较输出使能, 使能PWM
输出到端口
TIM_OCInitStructure.TIM_OCPolarity  = TIM_OCPolarity_High; //设置极性是高还是低
//设置占空比, 占空比=(CCRx/ARR)*100%或(TIM_Pulse/TIM_Period)*100%
TIM_OCInitStructure.TIM_Pulse = dutyfactor3 * 7200 / 100;
TIM_OC3Init(TIM3, &TIM_OCInitStructure);

//TIM3的CH4输出
TIM_OCInitStructure.TIM_OCMode      = TIM_OCMode_PWM1; //设置是PWM模式还是比较模式
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable; //比较输出使能, 使能PWM
输出到端口
TIM_OCInitStructure.TIM_OCPolarity  = TIM_OCPolarity_High; //设置极性是高还是低
//设置占空比, 占空比=(CCRx/ARR)*100%或(TIM_Pulse/TIM_Period)*100%
TIM_OCInitStructure.TIM_Pulse = dutyfactor4 * 7200 / 100;
TIM_OC4Init(TIM3, &TIM_OCInitStructure);

//使能输出状态
TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;

//设置TIM3的PWM输出为使能
TIM_CtrlPWMOutputs(TIM3, ENABLE);
}

void NVIC_Cfg(void)
{
//定义结构体
NVIC_InitTypeDef NVIC_InitStructure;

//选择中断分组1
NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

//选择TIM2的中断通道
NVIC_InitStructure.NVIC_IRQChannel = TIM2_IRQn;
//抢占式中断优先级设置为0
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0;
//响应式中断优先级设置为0
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0;
//使能中断
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;

NVIC_Init(&NVIC_InitStructure);
}

void RCC_Cfg(void)

```

```

{
//定义错误状态变量
ErrorStatus HSEStartUpStatus;

//将RCC寄存器重新设置为默认值
RCC_DeInit();

//打开外部高速时钟晶振
RCC_HSEConfig(RCC_HSE_ON);

//等待外部高速时钟晶振工作
HSEStartUpStatus = RCC_WaitForHSEStartUp();

if(HSEStartUpStatus == SUCCESS)
{
//设置AHB时钟(HCLK)为系统时钟
RCC_HCLKConfig(RCC_SYSCLK_Div1);

//设置高速AHB时钟(APB2)为HCLK时钟
RCC_PCLK2Config(RCC_HCLK_Div1);

//设置低速AHB时钟(APB1)为HCLK的2分频
RCC_PCLK1Config(RCC_HCLK_Div2);

//设置FLASH代码延时
FLASH_SetLatency(FLASH_Latency_2);

//使能预取指缓存
FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);

//设置PLL时钟,为HSE的9倍频 8MHz * 9 = 72MHz
RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);

//使能PLL
RCC_PLLCmd(ENABLE);

//等待PLL准备就绪
while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);

//设置PLL为系统时钟源
RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);

//判断PLL是否是系统时钟
while(RCC_GetSYSCLKSource() != 0x08);

}

//允许TIM2的时钟
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3,ENABLE);

//允许GPIO的时钟
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB|RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOC|RCC_APB2Periph_AFIO,ENABLE);

}

void TIM2_IRQHandler(void)
{
    ul6 aa=10;
    if(TIM_GetFlagStatus(TIM2,TIM_IT_Update) !=RESET)
    {
        //清除TIM2的中断待处理位
        TIM_ClearITPendingBit(TIM2 , TIM_FLAG_Update);

        TIM_Cmd(TIM2,DISABLE);
        //通过循环让灯闪烁
        while (aa){
            GPIO_SetBits(GPIOC,GPIO_Pin_3);
            delay_ms(10);
            GPIO_ResetBits(GPIOC,GPIO_Pin_3);
            delay_ms(10);
            aa--;
        }
        //使灯的状态为灭
        GPIO_SetBits(GPIOC,GPIO_Pin_3);
    }
}

```

```
TIM_Cmd(TIM2,ENABLE);
}

}

void delay_ms(u32 i)
{
    u32 temp;
    SysTick->LOAD=9000*i;           //设置重装数值，72MHz时
    SysTick->CTRL=0x01;             //使能，减到零是无动作，采用外部时钟源
    SysTick->VAL=0;                 //清零计数器
    do
    {
        temp=SysTick->CTRL;         //读取当前倒计数值
    }
    while((temp&0x01)&&!(temp&(1<<16)))); //等待时间到达
    SysTick->CTRL=0;                //关闭计数器
    SysTick->VAL=0;                 //清空计数器
}

```

在产生PWM时，如果输出引脚已经被使用，就要对引脚进行重映射，阅读《STM32中文参考手册2010》第119页可知：

表42 TIM3复用功能重映像

复用功能	TIM3_REMAP[1:0] = 00 (没有重映像)	TIM3_REMAP[1:0] = 10 (部分重映像)	TIM3_F (完
TIM3_CH1	PA6	PB4	
TIM3_CH2	PA7	PB5	
TIM3_CH3	PB0		
TIM3_CH4	PB1		

1. 重映像只适用于 64、100 和 144 脚的封装

对TIM3而言：

- 1、当没有重映射时，TIM3的四个通道CH1，CH2，CH3，CH4分别对应PA6，PA7,PB0,PB1
- 2、当部分重映射时，TIM3的四个通道CH1，CH2，CH3，CH4分别对应PB4，PB5,PB0,PB1
- 3、当完全重映射时，TIM3的四个通道CH1，CH2，CH3，CH4分别对应PC6，PC7,PC8,PC9

为了整齐，我们选择完全重映射，使用的函数是：

```
GPIO_PinRemapConfig(GPIO_FullRemap_TIM3, ENABLE);
```

如果想使用部分映射，参数用GPIO_PartialRemap_TIM3：

```
GPIO_PinRemapConfig(GPIO_PartialRemap_TIM3, ENABLE);
```



标签：STM32，定时器，PWM

好文要顶 关注我 收藏该文



jiwangbujiu
关注 - 30
粉丝 - 25
[+加关注](#)

00

- « 上一篇： STM32F103外部中断编程
- » 下一篇： STM32的时钟树深入详解以及RCC配置

posted @ 2016-06-25 15:18 jiwangbujiu 阅读(24990) 评论(0) 编辑 收藏

[刷新评论](#) [刷新页面](#) [返回顶部](#)

注册用户登录后才能发表评论，请 [登录](#) 或 [注册](#)，[访问](#) [网站首页](#)。

- 【推荐】超50万行VC++源码：大型组态工控、电力仿真CAD与GIS源码库
- 【活动】京东云服务器_云主机低于1折，低价高性能产品备战双11
- 【推荐】天翼云新用户专享，0元体验数十款云产品，立即开通

【活动】魔程社区技术沙龙—移动测试应用专场等你报名

【福利】学AI有奖：博客园&华为云 Modelarts 有奖训练营

相关博文：

- [STM32F103]定时器PWM输入
- STM32之PWM君
- STM32之PWM波形输出配置总结
- STM32F103输入捕获的实现
- stm32之PWM学习