本章教程使用定时器1通道1（PA8），通过PA8引脚产生PWM控制LED灯LED1的亮度。

1、PWM简介及相关函数介绍

脉冲宽度调制（PWM）是一种模拟控制方式，其是利用微处理器的数字输出对模拟电路进行控制的一种非常有效的技术，广泛应用在从测量、通信到功率控制与变换的许多领域中。CH32V103的高级定时器TIM1和通用定时器TIM2/3/4均可产生PWM输出。

PWM输出模式是定时器的基本功能之一。PWM输出模式最常见的是使用重装值确定PWM频率，使用捕获比较寄存器确定占空比的方法。将OCxM域（比较捕获通道x模式设置域）中置110b或者111b使用PWM模式1或者模式2，置OCxPE位（比较捕获寄存器x预装载使能位）使能预装载寄存器，最后置ARPE 位（自动重装预装使能位）使能预装载寄存器的自动重装载。在发生一个更新事件时，预装载寄存器的值才能被送到影子寄存器，所以在核心计数器开始计数之前，需要置UG位（更新事件产生位）来初始化所有寄存器。在PWM模式下，核心计数器和比较捕获寄存器一直在进行比较，根据CMS（中央对齐模式选择）位，定时器能够输出边沿对齐或者中央对齐的PWM信号

- 边沿对齐：使用边沿对齐时，核心计数器增计数或者减计数，在PWM模式1的情景下，在核心计数器的值大于比较捕获寄存器时，OCxREF上升为高；当核心计数器的值小于比较捕获寄存器时，OCxREF下降为低。
- 中央对齐：使用中央对齐模式时，核心计数器运行在增计数和减计数交替进行的模式下，OCxREF在核心计数器和比较捕获寄存器的值一致时进行上升和下降的跳变。但比较标志在三种中央对齐模式下，置位的时机有所不同。在使用中央对齐模式时，最好在启动核心计数器之前产生一个软件更新标志 （置UG位）。

关于PWM输出具体信息，可参考CH32V103应用手册。本章教程通过

定时器1通道1产生PWM控制LED，其程序所用库函数在定时器中断教程中均已介绍，在此不再赘述。

2、硬件设计

本章教程通过定时器1通道1（PA8）产生PWM控制LED灯，其中，定时器为CH32V103内部资源，无需进行硬件设计，只需进行LED连接即可，LED与GPIO引脚连接方式如下：

- LED1与PA8连接。

3、软件设计

本章教程主要通过定时器产生PWM输出控制LED灯，其中，led.c文件与led.h文件前面GPIO教程中已讲解，在此不再介绍，本章主要介绍pwm.h文件、pwm.c文件以及main.c文件，具体程序如下：
pwm.h文件

```
#ifndef __PWM_H
#define __PWM_H

#include "ch32v10x_conf.h"

void TIM1_PWMOut_Init( u16 arr, u16 psc, u16 ccp );

#endif

```

复制代码
pwm.h文件主要是PWM相关函数的声明；
pwm.c文件

```
#include "pwm.h"

void TIM1_PWMOut_Init( u16 arr, u16 psc, u16 ccp )
{
    GPIO_InitTypeDef GPIO_InitStructure;
    TIM_TimeBaseInitTypeDef TIM_TimeBaseInitStructure;
    TIM_OCInitTypeDef TIM_OCInitStructure;

    RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Periph_TIM1, ENABLE );//使能GPIOA外设时钟和TIM1时钟

    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;        //配置PA8引脚
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  //设置为复用推挽输出
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//设置输出速度：50MHz
    GPIO_Init( GPIOA, &GPIO_InitStructure );         //GPIO初始化
```

```
15.
16.     TIM_TimeBaseInitStructure.TIM_Period = arr;      //指定下次更新事件时要加载到活动自动重新加载寄存器中的周期值。
17.     TIM_TimeBaseInitStructure.TIM_Prescaler = psc;   //指定用于划分TIM时钟的预分频器值。
18.     TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;  //时钟分频因子
19.     TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up;//TIM计数模式，向上计数模式
20.     TIM_TimeBaseInit( TIM1, &TIM_TimeBaseInitStructure); //根据指定的参数初始化TIMx的时间基数单位
21.
22.     TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;    //指定TIM模式
23.     TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;//指定TIM输出比较状态，即使能比较输出
24.     TIM_OCInitStructure.TIM_Pulse = ccp;                 //指定要加载到捕获比较寄存器中的脉冲值。
25.     TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;    //指定输出极性。
26.     TIM_OC1Init( TIM1, &TIM_OCInitStructure ); //根据TIM_OCInitStruct中指定的参数初始化TIM1 Channel1。
27.
28.     TIM_CtrlPWMOutputs(TIM1, ENABLE );               //启用定时器1PWM输出
29.     TIM_OC1PreloadConfig( TIM1, TIM_OCPreload_Disable );//使能CCR1上的TIM1外设预加载寄存器
30.     TIM_ARRPreloadConfig( TIM1, ENABLE );            //使能ARR上TIM1外设预加载寄存器
31.     TIM_Cmd( TIM1, ENABLE );                         //使能TIM1
32. }
```

复制代码

pwm.c文件主要包含定时器的PWM输出配置，其具体配置步骤如下：

1、使能GPIO时钟和定时器时钟；

```
1. RCC_APB2PeriphClockCmd( RCC_APB2Periph_GPIOA | RCC_APB2Periph_TIM1, ENABLE );//使能GPIOA外设时钟和TIM1时钟
```

复制代码

2、初始化GPIOA，并进行相关参数配置；

```
1.     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;        //配置PA8引脚
2.     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_AF_PP;  //设置为复用推挽输出
3.     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;//设置输出速度：50MHz
4.     GPIO_Init( GPIOA, &GPIO_InitStructure );         //GPIO初始化
```

复制代码

3、初始化TIM1时基结构体，对相关参数进行配置；

```
1.  TIM_TimeBaseInitStructure.TIM_Period = arr;  //指定下次更新事件时要加载到活动自动重新加载寄存器中的周期值。
2.  TIM_TimeBaseInitStructure.TIM_Prescaler = psc;  //指定用于划分TIM时钟的预分频器值。
3.  TIM_TimeBaseInitStructure.TIM_ClockDivision = TIM_CKD_DIV1;  //时钟分频因子
4.  TIM_TimeBaseInitStructure.TIM_CounterMode = TIM_CounterMode_Up;//TIM计数模式，向上计数模式
5.  TIM_TimeBaseInit( TIM1, &TIM_TimeBaseInitStructure); //根据指定的参数初始化TIMx的时间基数单位
```

复制代码

4、初始化TIM1 Channel1，并对TIM_OCInitStruct中指定的参数进行配置；

```
1.  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM2;  //指定TIM模式
2.  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;//指定TIM输出比较状态，即使能比较输出
3.  TIM_OCInitStructure.TIM_Pulse = ccp;  //指定要加载到捕获比较寄存器中的脉冲值。
4.  TIM_OCInitStructure.TIM_OCPolarity = TIM_OCPolarity_High;  //指定输出极性。
5.  TIM_OC1Init( TIM1, &TIM_OCInitStructure ); //根据TIM_OCInitStruct中指定的参数初始化TIM1 Channel1。
```

复制代码

5、启用定时器1 PWM输出；

```
1. TIM_CtrlPWMOutputs(TIM1, ENABLE );  //启用定时器1PWM输出
```

复制代码

6、使能CCR1上的TIM1外设预加载寄存器和ARR上TIM1外设预加载寄存器；

```
1. TIM_OC1PreloadConfig( TIM1, TIM_OCPreload_Disable );//使能CCR1上的TIM1外设预加载寄存器
2. TIM_ARRPreloadConfig( TIM1, ENABLE );  //使能ARR上TIM1外设预加载寄存器
```

复制代码

7、使能TIM1。

```
1. TIM_Cmd( TIM1, ENABLE );  //使能TIM1
```

复制代码

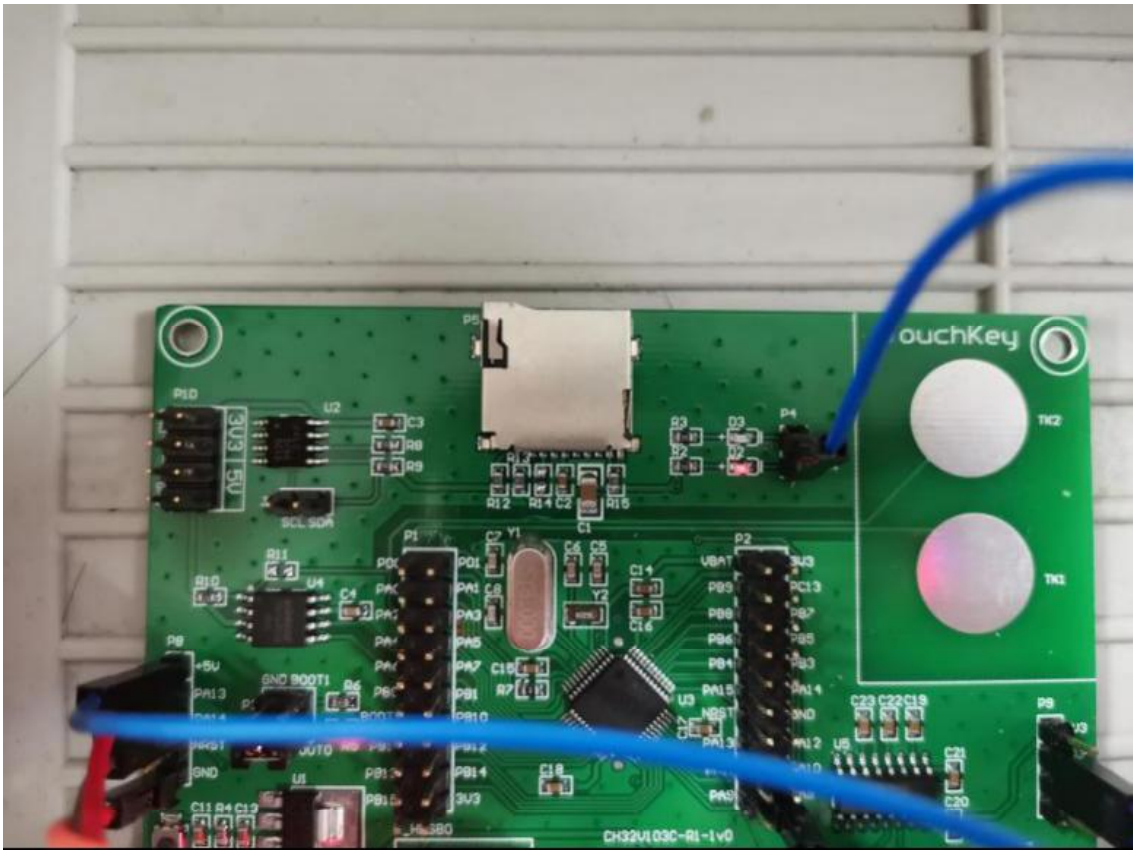
pwm.c文件主要是对PWM输出的相关配置，通过以上步骤进行配置，即可产生PWM输出。
main.c文件

```
1. int main(void)
2. {
3.     u16 pwmval=0;
4.     u8   a=1;
5.
6.     Delay_Init();
7.     USART_Printf_Init(115200);
8.     printf("SystemClk:%d\r\n",SystemCoreClock);
9.
10.    TIM1_PWMOut_Init( 899, 0, 500 );
11.
12.    while(1)
13.    {
14.        Delay_Ms(10);
15.        if(a)  pwmval++;
16.        else  pwmval--;
17.
18.        if(pwmval>300) a=0;
19.        if(pwmval==0)  a=1;
20.        TIM_SetCompare1(TIM1,pwmval); //设置TIM1捕获比较1寄存器值,，用于修改占空比
21.    }
22. }
```

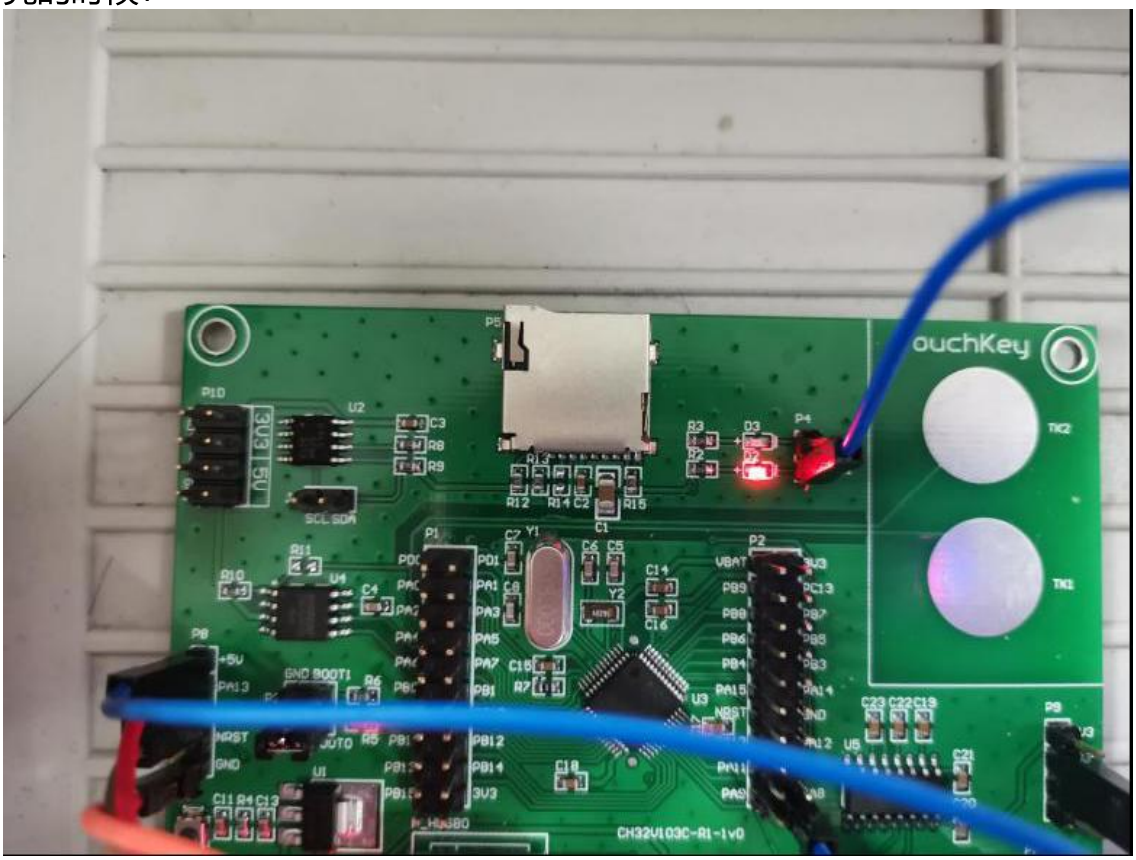
复制代码
main.c文件主要包含相关函数的初始化以及while循环函数，在while循环中，将pwmval这个值设置为PWM比较值，也就是通过pwmval来控制PWM的占空比，然后控制 pwmval的值从0变到300，然后又从300变到0，如此循环，因此 LED0 的亮度也会跟着从暗变到亮，然后又从亮变到暗。TIM_SetCompare1用于修改占空比。

4、下载验证

将编译好的程序下载到开发板并复位，可以看到开发板LED1开始不停由暗变亮，再由亮变暗。如图所示：
暗的时候：


ouchKey
TK2
TK1
CH32V103C-R1-1v0

亮的时候：


ouchKey
TK2
TK1
CH32V103C-R1-1v0