本章通过点亮LED作为CH32V103系列MCU应用开发的第一个教程，其LED灯控制使用到GPIO外设的基本输出功能，本章将通过点亮LED对CH32V103的GPIO进行基本的学习了解。

1、GPIO简介及其库函数介绍

GPIO，全称为通用输入输出端口，其可与外部设备连接实现MCU与外部设备的通讯、控制、信号采集等功能。本教程即通过CH32V103的GPIO与LED连接，实现MCU对LED的输出控制。关于CH32V103 GPIO的具体介绍，可参考CH32V103应用手册和数据手册。

进行LED点亮程序编写之前，需对GPIO固件库进行了解。GPIO相关的函数和定义分布在固件库文件ch32v10x_gpio.c和头文件ch32v10x_gpio.h文件中，LED点亮程序需要调用GPIO固件库文件中某些函数，GPIO库函数相关函数如下：

1. void GPIO_DeInit(GPIO_TypeDef* GPIOx);
2. void GPIO_AFIODeInit(void);
3. void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct);
4. void GPIO_StructInit(GPIO_InitTypeDef* GPIO_InitStruct);
5. uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
6. uint16_t GPIO_ReadInputData(GPIO_TypeDef* GPIOx);
7. uint8_t GPIO_ReadOutputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
8. uint16_t GPIO_ReadOutputData(GPIO_TypeDef* GPIOx);
9. void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
10. void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
11. void GPIO_WriteBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, BitAction BitVal);
12. void GPIO_Write(GPIO_TypeDef* GPIOx, uint16_t PortVal);
13. void GPIO_PinLockConfig(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
14. void GPIO_EventOutputConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource);
15. void GPIO_EventOutputCmd(FunctionalState NewState);

16. void GPIO_PinRemapConfig(uint32_t GPIO_Remap, FunctionalState NewState);
17. void GPIO_EXTILineConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource);

复制代码

1.1、void GPIO_DeInit(GPIO_TypeDef* GPIOx)

功 能：将GPIOx外围寄存器初始化为其默认重置值。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG。

1.2、void GPIO_AFIODeInit(void)

功 能：将复用功能（重映射，事件控制与EXTI设置）重设为默认值。通俗理解可认为此函数功能为初始化所有的复用功能。
参 数：无。

1.3、void GPIO_Init(GPIO_TypeDef* GPIOx, GPIO_InitTypeDef* GPIO_InitStruct)

功 能： GPIO初始化函数，根据GPIO_InitStructure中的指定参数初始化GPIO外设寄存器。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；GPIO_InitStructure为GPIO_InitTypedef类型结构体指针，指向包含GPIO外设配置信息的GPIO_InitTypedef结构体。

1.4、void GPIO_StructInit(GPIO_InitTypeDef* GPIO_InitStruct)

功 能：初始化结构体成员，即用其默认值填充每个GPIO_StructInit成员，包括GPIO_Pin、GPIO_Speed、GPIO_Mode等。
参 数：GPIO_StructInit为指向结构体GPIO_InitTypedef的指针，待初始化。

1.5、uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)

功 能：读取指定端口管脚的输入（0或1）。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；GPIO_Pin用来选择待读取的端口位。

1.6、uint16_t GPIO_ReadInputData(GPIO_TypeDef* GPIOx)

功 能：读取指定的GPIO输入数据端口。
参 数：无。

1.7、uint8_t GPIO_ReadOutputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)

功 能：读取指定端口管脚的输出（0或1）。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；

GPIO_Pin用来选择待读取的端口位。

1.8、uint16_t GPIO_ReadOutputData(GPIO_TypeDef* GPIOx)

功 能：读取指定的GPIO输出数据端口。
参 数：无。

1.9、void GPIO_SetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)

功 能：设置指定的数据端口位，可理解为将指定的引脚设置为高电平。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；
GPIO_Pin用来选择待设置的端口位。

1.10、void GPIO_ResetBits(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)

功 能：清除指定的数据端口位，可理解为将指定的引脚设置为低电平。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；
GPIO_Pin用来选择待设置的端口位。

1.11、void GPIO_WriteBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin, BitAction BitVal)

功 能：设置或清除指定的数据端口位，可理解为将指定的引脚设置为高电平或低电平。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；
GPIO_Pin用来选择待设置的端口位；BitVal为指定待写入的值，该参数必须取枚举BitAction的其中一个值。

1.12、void GPIO_Write(GPIO_TypeDef* GPIOx, uint16_t PortVal)

功 能：向指定的GPIO端口写入数据
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；
PortVal为待写入端口数据寄存器（ODR寄存器）的值。

1.13、void GPIO_PinLockConfig(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)

功 能：锁定GPIO管脚配置寄存器。
参 数：GPIOx用来选择GPIO外设，取值可为GPIOA-GPIOG；，
GPIO_Pin为待锁定的端口位。

1.14、void GPIO_EventOutputConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource)

功 能：选择GPIO管脚用作事件输出
参 数：GPIO_PortSource为选择用作事件输出的GPIO端口；

GPIO_PinSource为事件输出的管脚。

1.15、void GPIO_EventOutputCmd(FunctionalState NewState)

功 能：使能或失能事件输出
参 数：NewState为管脚重映射的新状态，该参数可以取ENABLE或DISABLE。

1.16、void GPIO_PinRemapConfig(uint32_t GPIO_Remap, FunctionalState NewState)

功 能：更改指定管脚的映射。
参 数：GPIO_Remap为选择重映射的管脚；NewState为管脚重映射的新状态，该参数可以取ENABLE或DISABLE。

1.17、void GPIO_EXTILineConfig(uint8_t GPIO_PortSource, uint8_t GPIO_PinSource)

功 能：选择GPIO管脚用作外部中断线路。
参 数：GPIO_PortSource为选择用作外部中断线源的GPIO端口;GPIO_PinSource为待设置的外部中断线路。

2、硬件设计

由于本次教程为点亮LED，需用到LED。开发板上带有两个LED灯（LED1和LED2），用两根杜邦线分别将LED1和LED2与对应GPIO引脚连接起来，此处连接方式为：

- LED1与PA0连接；
- LED2与PA1连接。

3、软件设计

LED点亮程序通过控制CH32V103 GPIO引脚的电平高低实现LED闪烁，其实现步骤如下：

- 定义一个GPIO_InitTypeDef类型结构体，结构体成员包括GPIO_Pin、GPIO_Mode、GPIO_Speed；
- 使能GPIO时钟，否则GPIO引脚不工作，本次实验使能GPIOA时钟；
- 配置GPIO_InitTypeDef类型结构体成员参数，及配置GPIO引脚为对应引脚、GPIO模式为推挽输出、GPIO口输出速度为相应值；
- 调用库函数，初始化GPIO；
- 调用库函数，设置GPIO引脚输出电平。

LED点亮程序如下：
led.h文件

```
#ifndef __LED_H
#define __LED_H

#include "ch32v10x_conf.h"

void LED_Init(void);  //初始化

#endif
```

复制代码
led.h文件是函数的声明，对LED点亮实验的GPIO配置函数进行声明；

led.c文件

```
#include "led.h"

void LED_Init(void)
{

GPIO_InitTypeDef  GPIO_InitStructure;                    //定义一个GPIO_InitTypeDef类型的结构体

RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);//使能与LED相关的GPIO端口时钟

GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;       //配置GPIO引脚
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;          //设置GPIO模式为推挽输出
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;         //设置GPIO口输出速度
GPIO_Init(GPIOA, &GPIO_InitStructure);                    //调用库函数，初始化GPIOA

GPIO_SetBits(GPIOA,GPIO_Pin_0|GPIO_Pin_1);                //设置引脚输出高电平

}
```

复制代码
led.c文件是LED点亮实验的GPIO配置程序，本教程通过PA0和PA1引脚控制LED闪烁，其配置流程如下：

1、定义一个GPIO_InitTypeDef类型的结构体，具体函数为：
GPIO_InitTypeDef GPIO_InitStructure;

2、使能GPIO时钟，使能GPIOA，具体函数为：
RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);

3、配置GPIO引脚，配置PA0和PA1，具体函数为：
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0|GPIO_Pin_1;

4、设置GPIO引脚模式，设为推挽输出，具体函数为：
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;

5、设置GPIO引脚输出速度，设为50MHz，具体函数为：
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;

6、调用库函数，初始化GPIOA，具体函数为：
GPIO_Init(GPIOA, &GPIO_InitStructure);

7、调用库函数，设置引脚输出电平，此处设置为高电平，具体函数为：
GPIO_SetBits(GPIOA,GPIO_Pin_0|GPIO_Pin_1);

完成led.c和led.h之后，对其进行编译保存，继续对main函数进行编写，main函数具体代码如下：
main.c文件

```
#include "debug.h"
#include "led.h"

int main(void)
{
        u8 i=0;
        u8 j=0;
        Delay_Init();   //延时函数初始化
        LED_Init();     //LED初始化
        while(1)
        {
            Delay_Ms(250); //延时250ms
            GPIO_WriteBit(GPIOA, GPIO_Pin_0, (i==0) ? (i=Bit_SET): (i=Bit_RESET)); //设置PA0引脚状态为低电平
            Delay_Ms(250); //延时250ms
            GPIO_WriteBit(GPIOA, GPIO_Pin_1, (j==0) ? (j=Bit_SET): (j=Bit_RESET)); //设置PA1引脚状态为低电平
        }
}
```

复制代码

main函数包含了debug.h和led.h两个头文件，使得debug.c和led.c文件内函数可在main函数中被调用。main函数详解如下：

- u8 i=0和u8 j=0即为对变量i、j进行定义并赋初值0，变量i，j在后面while循环中用到，在此先进行定义；
- Delay_Init()为延时函数初始化，因为程序中有用到延时，如Delay_Ms(250)，所以需要对延时函数进行初始化；
- LED_Init()为LED GPIO配置初始化，初始化与LED连接的硬件接口，此处为PA0和PA1引脚接口；
- while(1)为无限循环函数，只要程序运行，LED灯就一直闪烁；
- Delay_Ms(250)为延时函数，在两个GPIO_WriteBit函数之间添加延时函数，可造成跑马灯效果；
- GPIO_WriteBit函数意思为设置相应端口位的引脚状态，由(i==0) ? (i=Bit_SET):(i=Bit_RESET)和(j==0) ? (j=Bit_SET):(j=Bit_RESET)这两句代码可知，PA0和PA1置低位，由于初始化中PA0和PA1置高位，因此在while函数作用下，LED0和LED1不停闪烁，且由于延时函数存在，呈现跑马灯效果。

4、下载验证

将编译好的程序下载到开发板并复位，可以看到两个LED灯轮流不停闪烁，现象如下：