# LED点亮的原理

**简单说来**：led就是一颗小灯，普通led两个脚，正端接电源正极（接反了不亮），另一端接地就可以亮。在led可接受电压范围内，调节电压高低可以实现发光强度的调节；也可以通过反复通电断电（频率要足够快），或者控制通电断电时间来实现亮度调节。RGB全彩LED灯（开发板配备），实质上是一种把红、绿、蓝单色发光体集成到小面积区域中的LED灯（相当于三个led集成到一个壳里），控制时对这三种颜色的灯管输出不同的光照强度，即可混合得到不同的颜色，其混色原理与光的三原色混合原理一致。

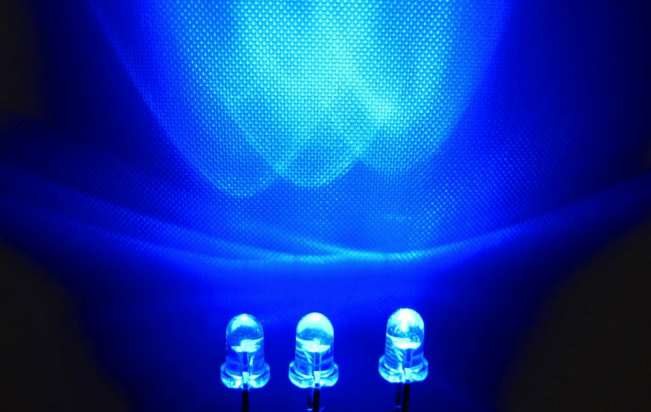
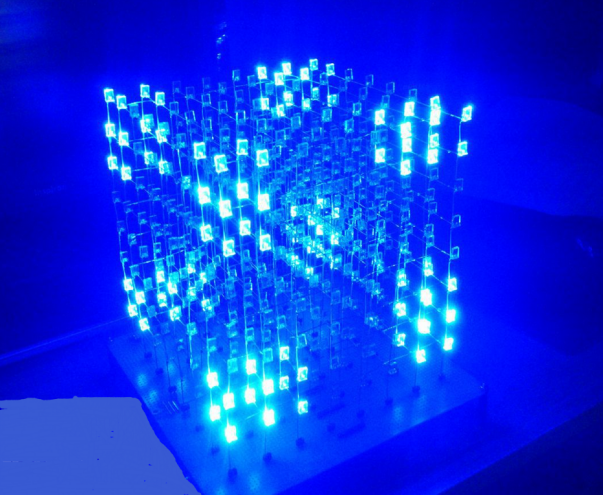
**复杂说来（了解即可）：**

LED（Light Emitting Diode），[发光二极管](https://baike.baidu.com/item/%E5%8F%91%E5%85%89%E4%BA%8C%E6%9E%81%E7%AE%A1" \t "https://baike.baidu.com/item/LED%E7%81%AF/_blank)，是一种能够将[电能](https://baike.baidu.com/item/%E7%94%B5%E8%83%BD" \t "https://baike.baidu.com/item/LED%E7%81%AF/_blank)转化为[可见光](https://baike.baidu.com/item/%E5%8F%AF%E8%A7%81%E5%85%89" \t "https://baike.baidu.com/item/LED%E7%81%AF/_blank)的固态的[半导体](https://baike.baidu.com/item/%E5%8D%8A%E5%AF%BC%E4%BD%93" \t "https://baike.baidu.com/item/LED%E7%81%AF/_blank)器件，它可以直接把电转化为光。LED的心脏是一个半导体的晶片，晶片的一端附在一个支架上，一端是负极，另一端连接电源的正极，使整个晶片被环氧树脂封装起来。

半导体晶片由两部分组成，一部分是[P型半导体](https://baike.baidu.com/item/P%E5%9E%8B%E5%8D%8A%E5%AF%BC%E4%BD%93" \t "https://baike.baidu.com/item/LED%E7%81%AF/_blank)，在它里面空穴占主导地位，另一端是[N型半导体](https://baike.baidu.com/item/N%E5%9E%8B%E5%8D%8A%E5%AF%BC%E4%BD%93" \t "https://baike.baidu.com/item/LED%E7%81%AF/_blank)，在这边主要是电子。但这两种半导体连接起来的时候，它们之间就形成一个P-N结。当电流通过导线作用于这个晶片的时候，电子就会被推向P区，在P区里电子跟空穴复合，然后就会以光子的形式发出能量，这就是LED灯发光的原理。而光的波长也就是光的颜色，是由[形成](https://baike.baidu.com/item/%E5%BD%A2%E6%88%90" \t "https://baike.baidu.com/item/LED%E7%81%AF/_blank)P-N结的材料决定的。

LED可以直接发出红、黄、蓝、绿、青、橙、紫、白色的光。

光立方是一学习单片机非常好的练习材料。拿8\*8\*8光立方举例，它是由512个发光二极管按照立方体的方式搭建焊接起来的，有层共阴束共阳和层共阳束共阴两种方案，每一层有8\*8个发光二极管，共8层。用c语言编译单片机程序使自定义动画效果得以实现。（后续讲解）

图一:蓝光二极管 图二：发光二极管阵列——光立方

# GPIO外设

GPIO （General Purpose Input Output ）是通用输入输出端口的简称(其中输入叫GPI,输出叫GPO)，简单来说就是 STM32 可控制的引脚，STM32 芯片的 GPIO 引脚与外部设备连接起来，从而实现与外部通讯、控制以及数据采集的功能。

STM32芯片的 GPIO 被分成很多组，每组有 16 个引脚，如型号为 STM32F103VET6 型号的芯片有 GPIOA、GPIOB、GPIOC至 GPIOE共 5组GPIO，芯片一共 100个引脚，其中 GPIO

就占了一大部分，所有的 GPIO引脚都有基本的输入输出功能。

最基本的输出功能是由 STM32 控制引脚输出高、低电平，实现开关控制，如把 GPIO引脚接入到LED灯，那就可以控制LED灯的亮灭（后有实例讲解），引脚接入到继电器或三极管，那就可以通过继电器或三极管控制外部大功率电路的通断。

最基本的输入功能是检测外部输入电平，如把 GPIO 引脚连接到按键，通过电平高低

区分按键是否被按下。

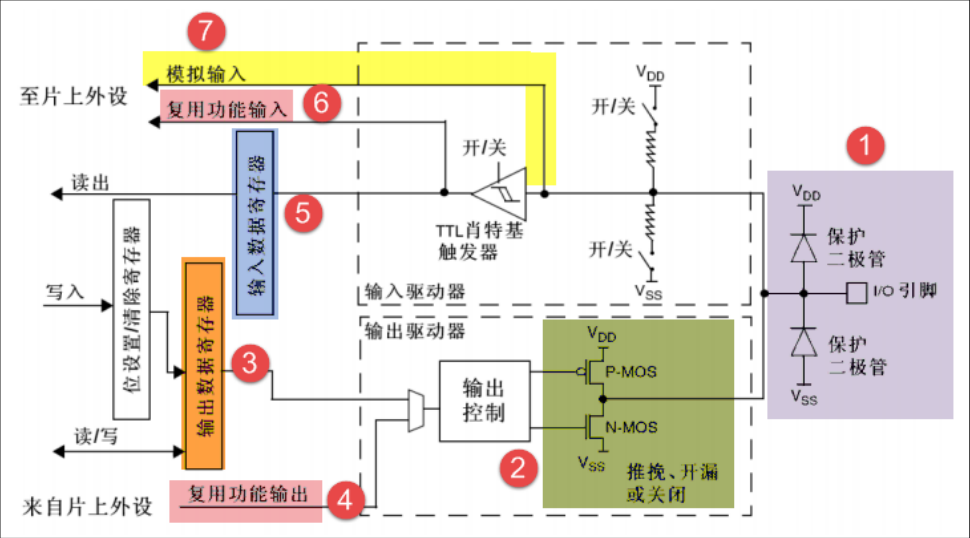
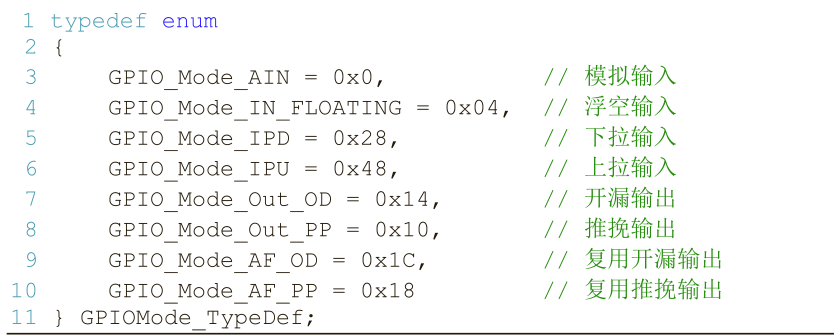


图3 GPIO 结构框图

通过 GPIO硬件结构框图，就可以从整体上深入了解 GPIO外设及它的各种应用模式。该图从最右端看起，最右端就是代表 STM32 芯片引出的 GPIO 引脚，其余部件都位于芯片内部。其中1-7大概了解即可，等你们对单片机有一个更深入的了解了再去研究即可。

**GPIO工作模式**

由 GPIO 的结构决定了 GPIO可以配置成以下8种工作模式：



在固件库中，GPIO 总共有 8种细分的工作模式，稍加整理可以大致归类为以下三类（尽力理解，很重要）：

1. 输入模式(模拟/浮空/上拉/下拉)

在输入模式时，施密特触发器打开，输出被禁止，可通过输入数据寄存器 GPIOx\_IDR

读取 I/O 状态。其中输入模式，可设置为上拉、下拉、浮空和模拟输入四种。上拉和下拉

输入很好理解，默认的电平由上拉或者下拉决定。浮空输入的电平是不确定的，完全由外部的输入决定，一般接按键的时候用的是这个模式。模拟输入则用于 ADC 采集。

2. 输出模式(推挽/开漏)

在输出模式中，推挽模式时双 MOS 管以轮流方式工作，输出数据寄存器 GPIOx\_ODR

可控制 I/O 输出高低电平。开漏模式时，只有 N-MOS 管工作，输出数据寄存器可控制 I/O

输出高阻态或低电平。输出速度可配置，有2MHz\10MHz\50MHz的选项。此处的输出速度

即 I/O 支持的高低电平状态最高切换频率，支持的频率越高，功耗越大，如果功耗要求不

严格，把速度设置成最大即可。在输出模式时施密特触发器是打开的，即输入可用，通过输入数据寄存器 GPIOx\_IDR可读取 I/O 的实际状态。

3. 复用功能(推挽/开漏)

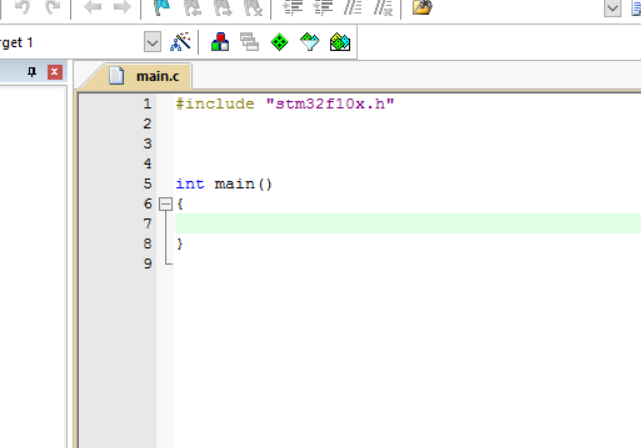
“复用功能”中的“复用”是指 STM32 的其它片上外设对 GPIO 引脚进行控制，例如我们使用 USART 串口通讯时（后续讲解），需要用到某个 GPIO 引脚作为通讯发送引脚，这个时候就可以把该 GPIO引脚配置成 USART 串口复用功能，由串口外设控制该引脚，发送数据。

此时 GPIO 引脚用作该外设功能的一部分，算是第二用途。复用功能模式中，输出使能，输出速度可配置，可工作在开漏及推挽模式，但是输出信号源于其它外设，输出数据寄存器 GPIOx\_ODR 无效；输入可用，通过输入数据寄存器可获取 I/O 实际状态，但一般直接用外设的寄存器来获取该数据信号。

# 点亮一盏LED

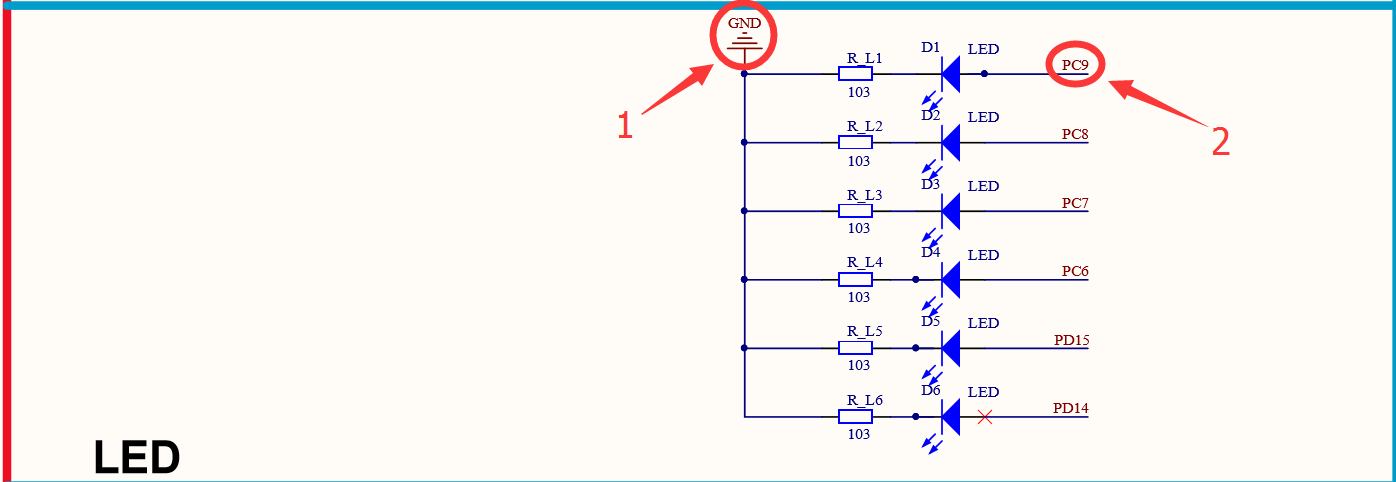
下面我们就通过点亮一盏LED来进入stm32实战的大门

1，首先，打开自己的一个空白工程，此时的界面：

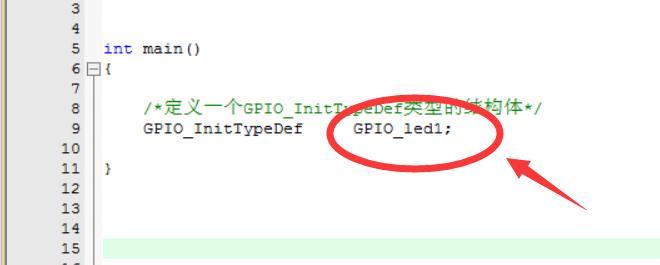


2，打开配套开发板原理图，找到led部分

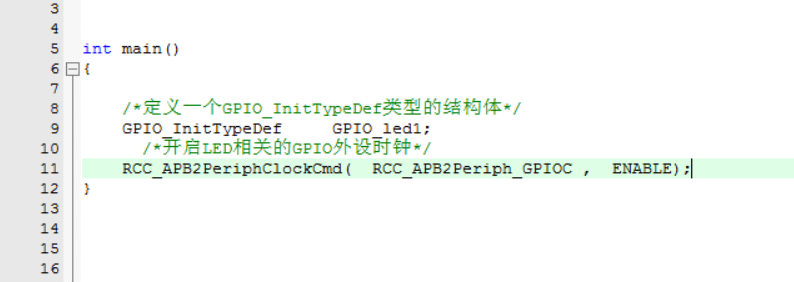
此时你会发现，LED是共阴极的（箭头1），然后你找到LED所连接引脚，比如PC9（箭头2）



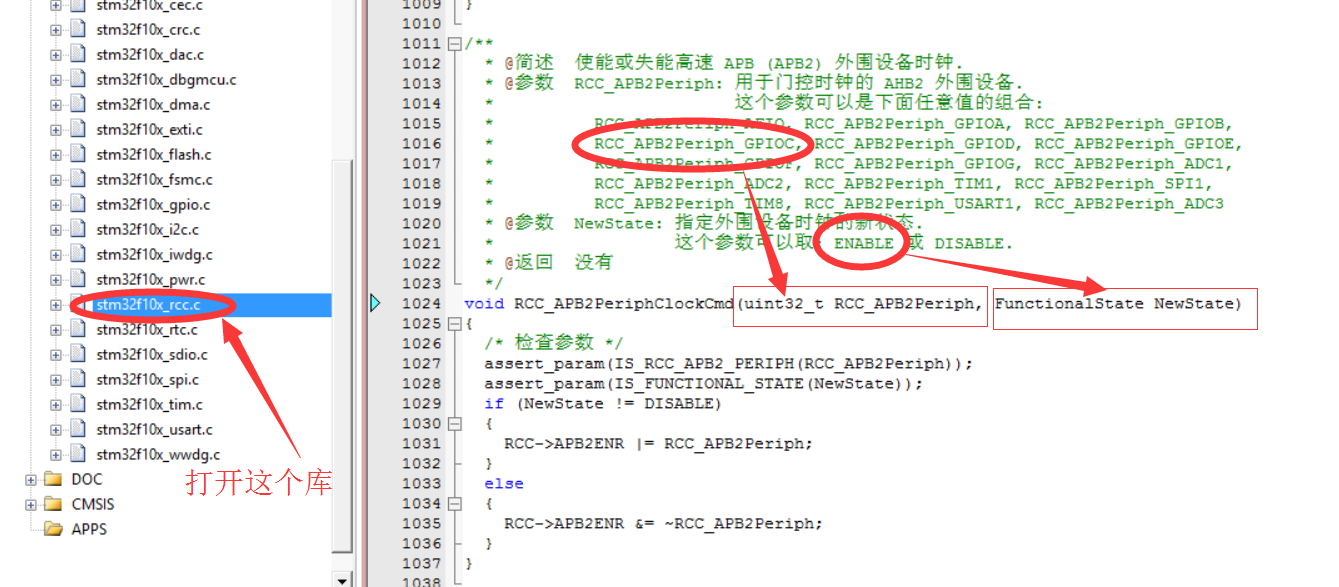
3 定义一个GPIO\_InitTypeDef 类型的结构体，名字自定义，这里取名为GPIO\_led1（箭头所示）



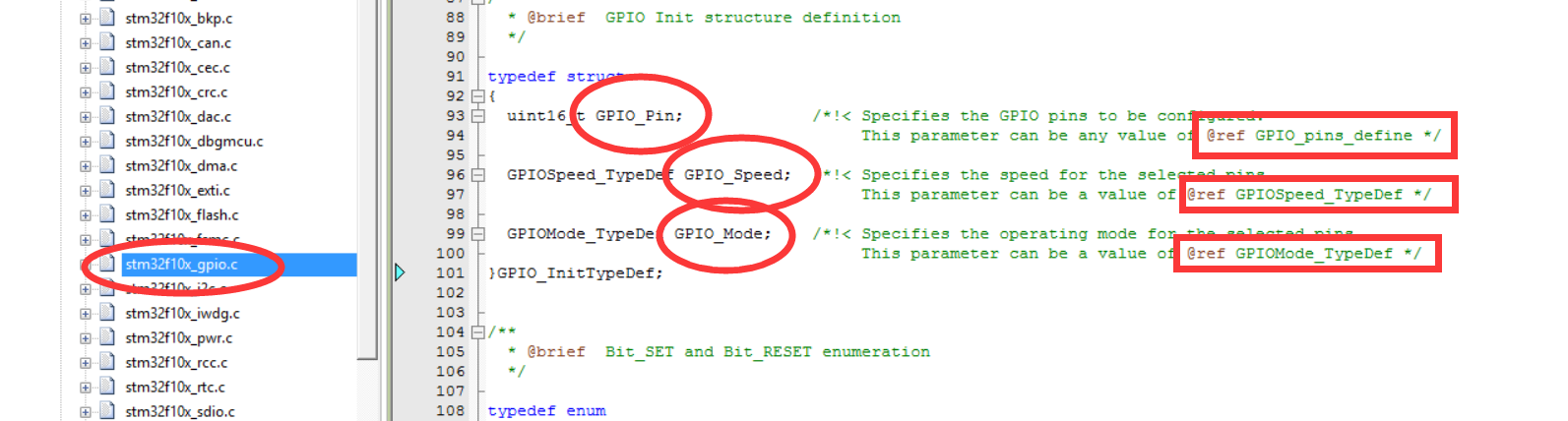
4 非常重要又容易忘记的一步，开启相应外设时钟，此时所用gpio为PC9，所以开启的时钟应该是GPIOC的时钟



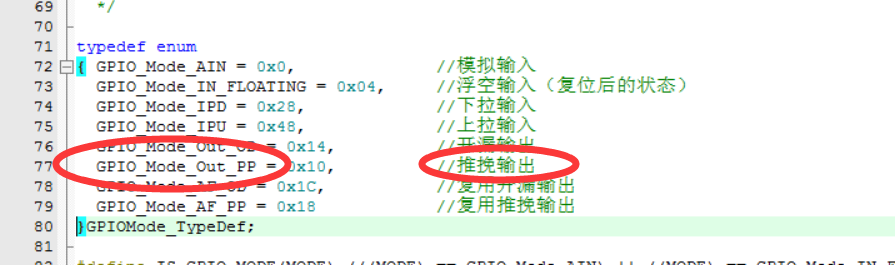
开启时钟的函数在下图所示库中，初学期间能够正确给函数传值即可。



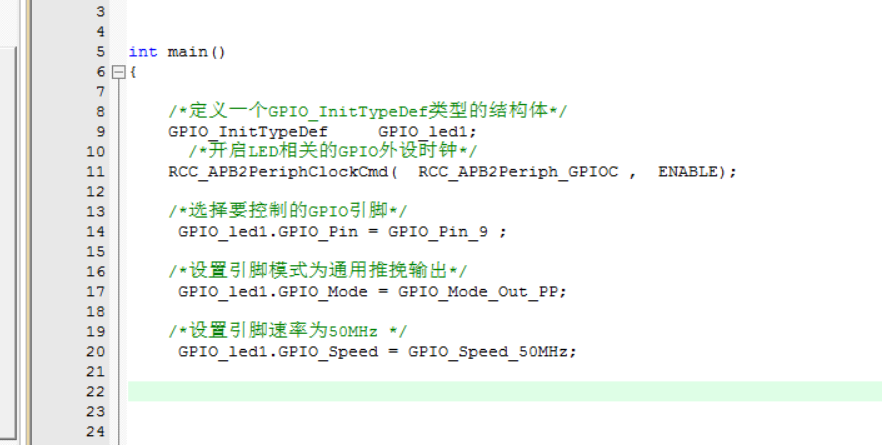
5 给GPIO结构体赋值并初始化这个结构体，结构体所在的库以及结构体成员在下图指示，其中方框为可填入值的范围



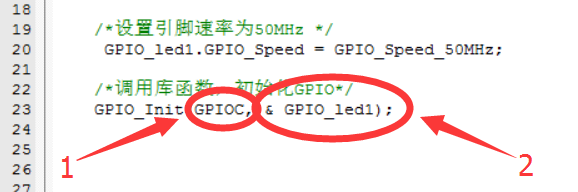
为可填入值的范围，以GPIO\_Mode为例，在这里选择推挽输出，所以填入的值应该是GPIO\_Mode\_Out\_PP



然后赋值之后是这样的



然后初始化这个结构体，让它开始发挥你想要的作用

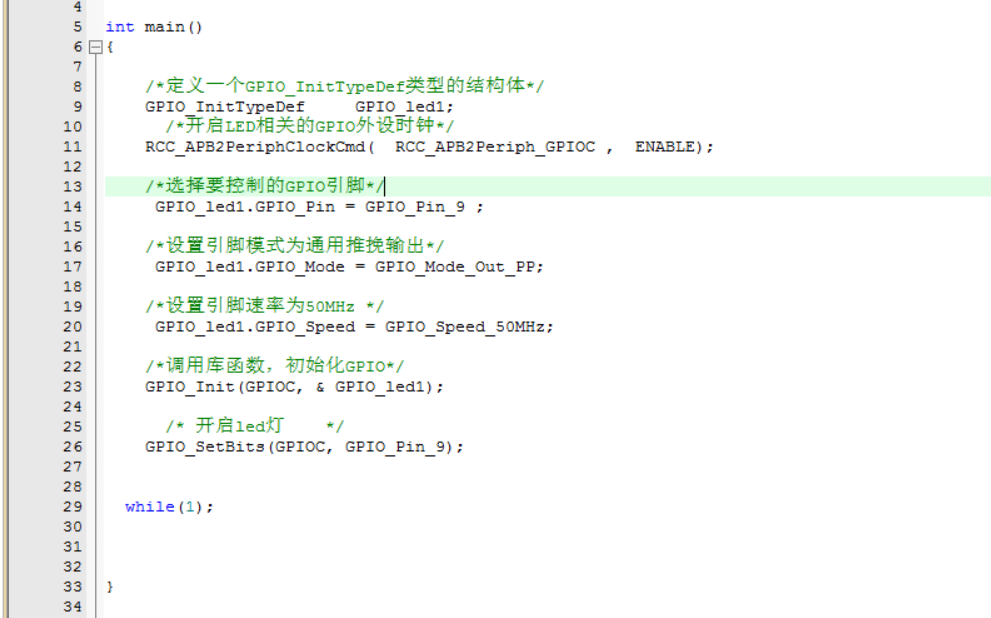


1为你要填入的外设，2为你赋值后的结构体（千万别忘了加取地址运算符&）

6，调用函数，让相应引脚输出高电平，也就是置位操作，要写入的两个值一看就懂了吧

然后加上while(1)





7最后一步 下载验证！