第四天笔记

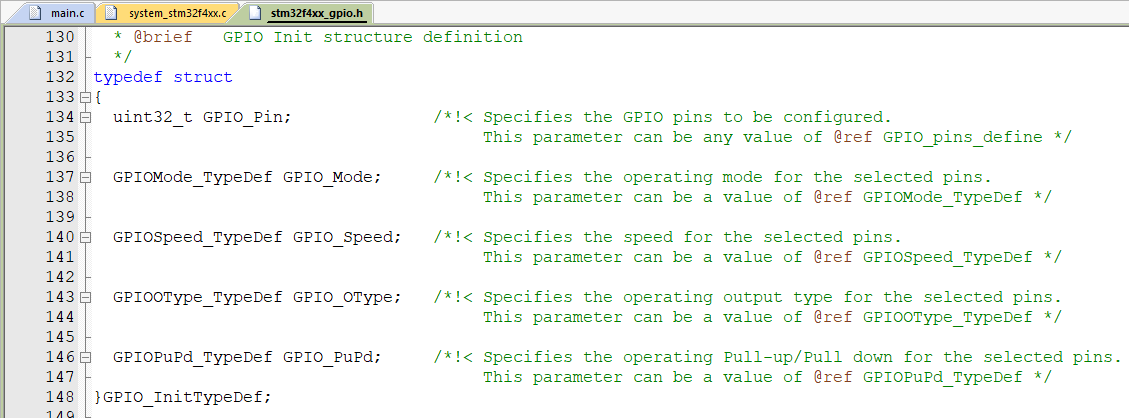
1. GPIO外设详解



大家可以看到，函数库开发的时候外设的使用流程都是一样的，接下来就讲解一下细节

* 定义一个外设的结构体变量 变量命名规则 ***PPP\_InitTypeDef PPP\_InitStructure;***

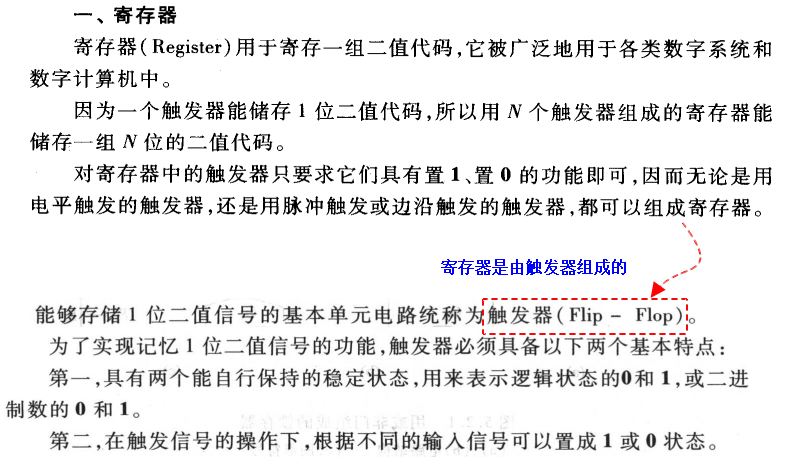
每个外设都有对应的结构体，结构体的定义一般都是存放在每个外设的头文件内，比如GPIO外设的初始化结构体就定义在stm32f4xx\_gpio.h中



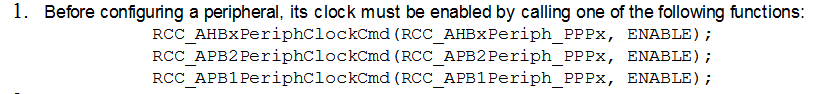
* 在配置外设结构体之前，需要调用库函数打开外设的时钟 如RCC\_AHB1PeriphClockCmd

任何一个外设都是由时钟控制的，**时钟就相当于外设的开关**，因为底层寄存器一般内部结构中都是由触发器组成，而触发器需要触发信号，而触发信号是由时钟提供的。

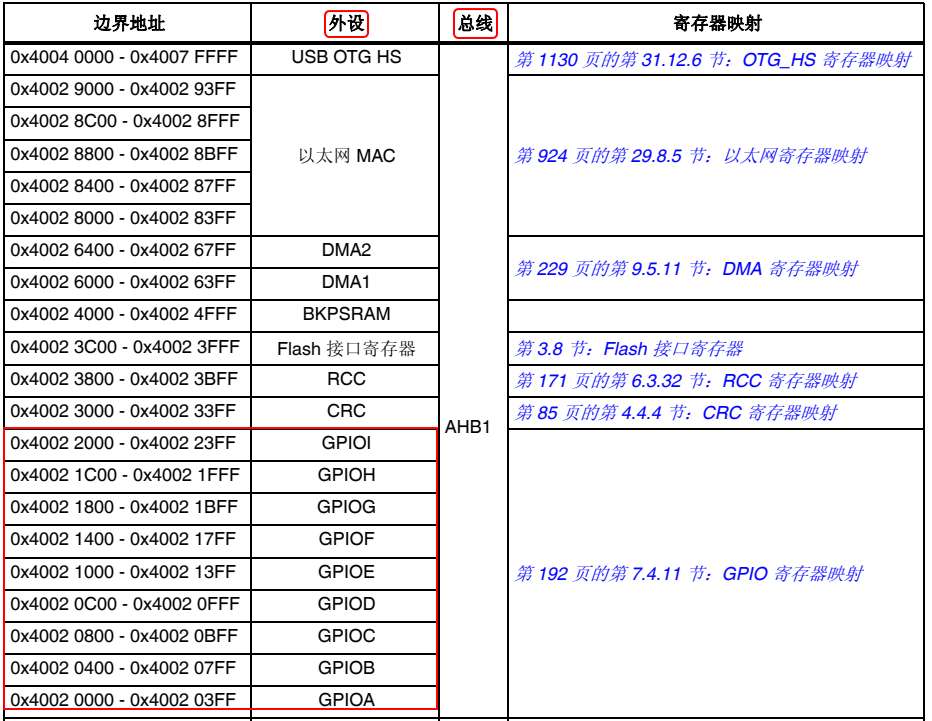
STM32低功耗的根本原因就是在**芯片复位之后**，所有的外设的**时钟**（开关）都是**默认关闭**，需要用到哪个外设就打开哪个外设的时钟，不需要的就关闭，**有利于降低功耗**。



**可以看到，函数库中提供了多个可以打开外设时钟的函数，如下图所示**



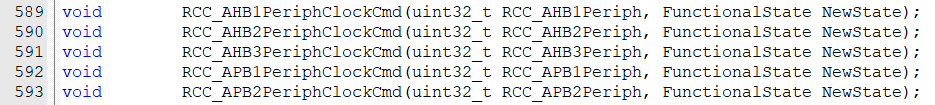
思考：想要打开GPIO外设的时钟，应该用哪个函数？可以参考STM32中文参考手册的2.3章节 存储器映射

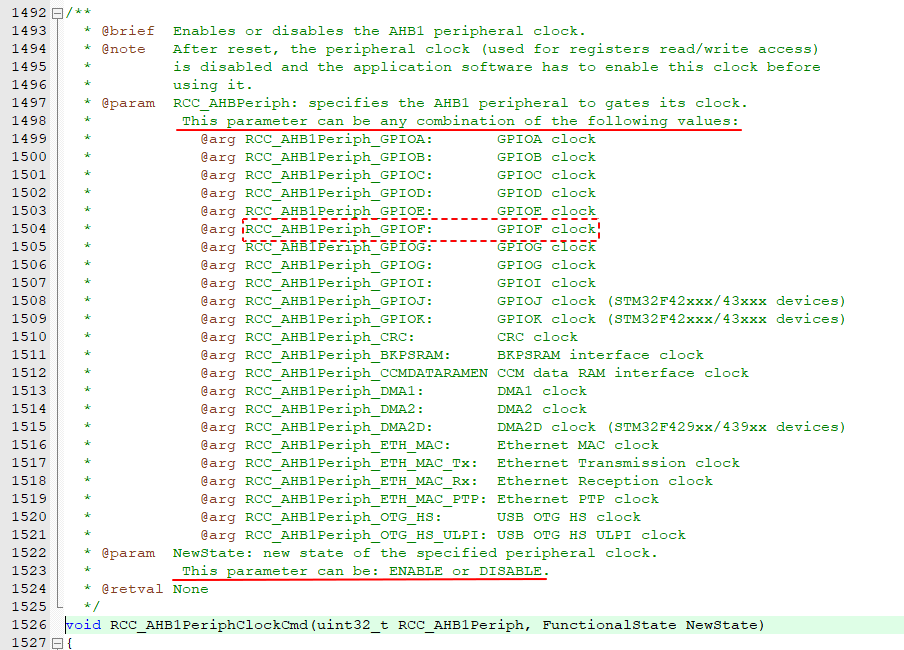


可以看到，内核中的外设都会挂载在不同的总线下面，**AHB总线叫做高级高性能总线**，**APB总线叫做高级外设总线（APB1和APB2）**。**每个总线的时钟频率都不同**，因为不同类型的外设需要的时钟频率不同，所以为了降低STM32的功耗，把不同性能的外设挂载在不同频率的总线下。

举个例子：早上起床刷牙，用水龙头接杯水就可以了，不需要跑到三峡打开水闸然后接杯水。

**函数库中提供的时钟使能的函数在stm32f4xx\_rcc.h中声明，在stm32f4xx\_rcc.c定义的**





函数原型

void RCC\_AHB1PeriphClockCmd(uint32\_t RCC\_AHB1Periph, FunctionalState NewState)

函数参数

参数一：RCC\_AHB1Periph 指的是外设的时钟 如 RCC\_AHB1Periph\_GPIOF

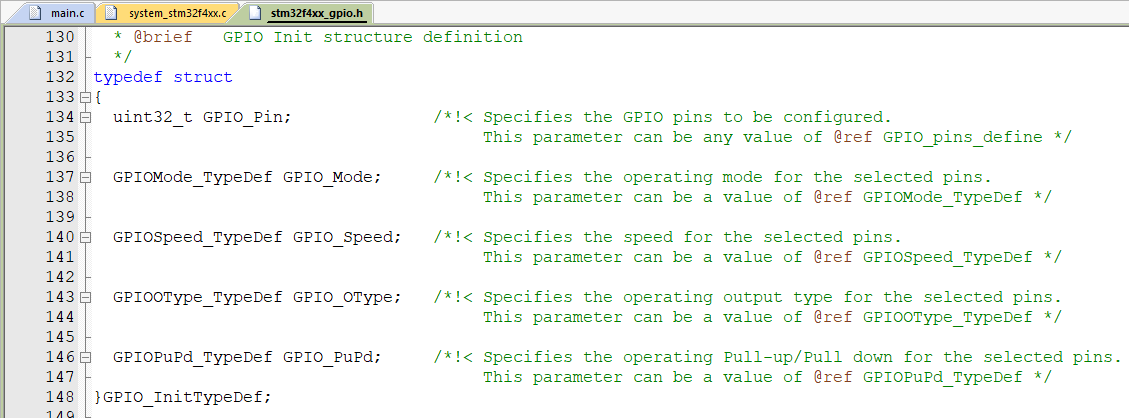
参数二：NewState 指的是打开或者关闭 如 ENABLE or DISABLE

返回值：None

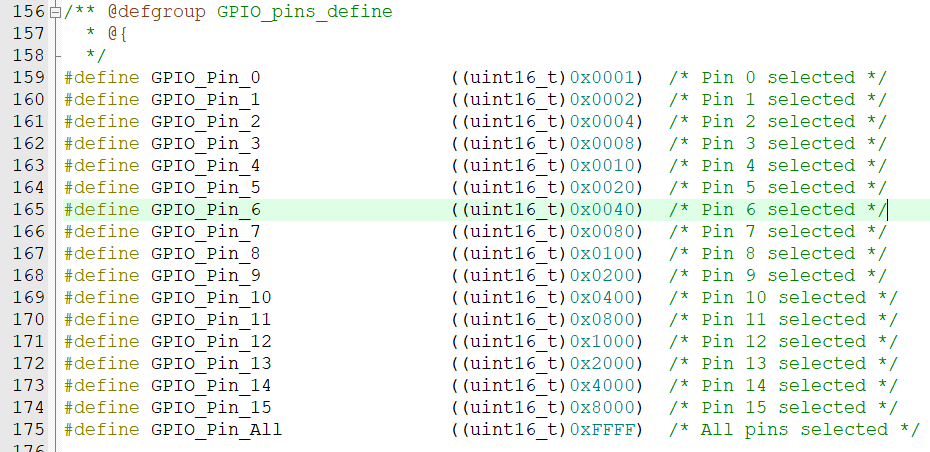
需要注意：在调用函数库的时候，大家不需要关注函数内部的实现过程，只需要掌握函数的传参和返回值即可。

* 打开外设时钟后，需要对外设的结构体成员进行赋值操作（引脚、模式、速率、类型....）

GPIO外设的结构体成员有5个，可以参考stm32f4xx\_gpio.h

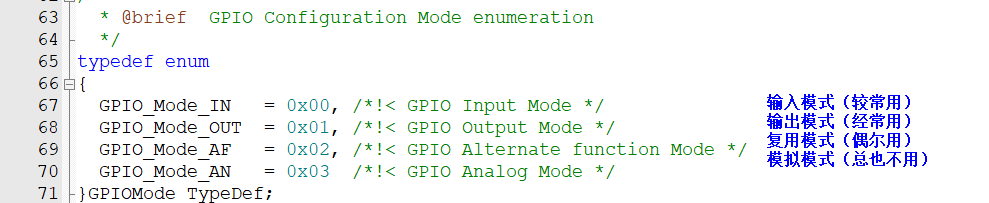


1. 需要指定配置的引脚 GPIO\_Pin 定义在GPIO\_pins\_define （跳转 or 查找）

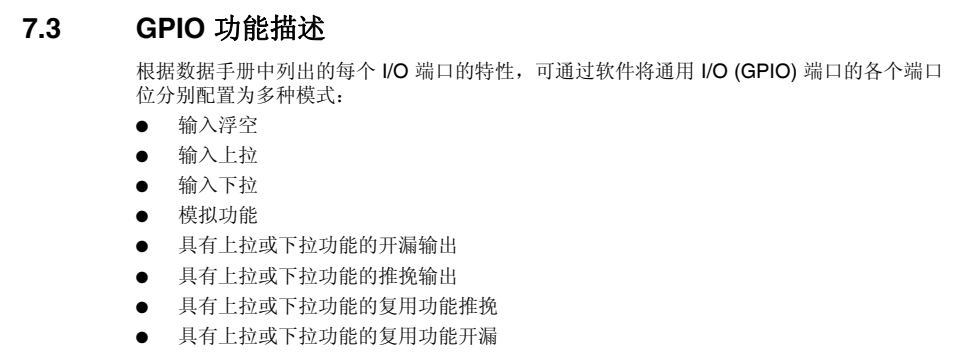


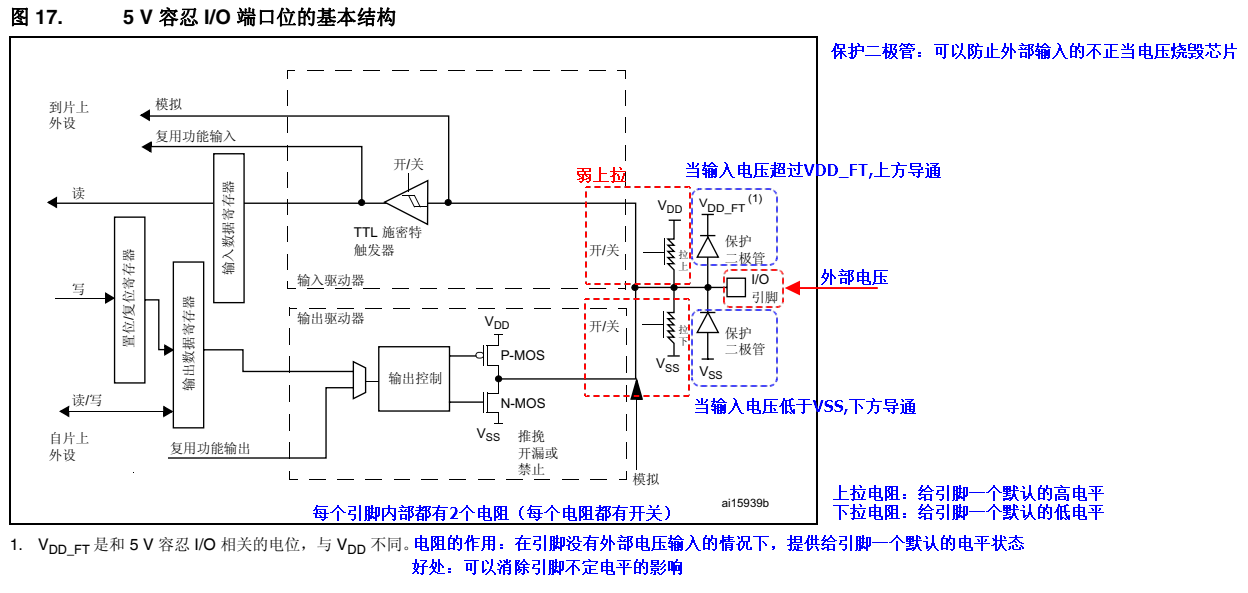
一个端口有16个引脚，在操作端口的某个引脚的时候，需要说清楚用哪个引脚，这样才可以对寄存器的某些位进行操作。

1. 需要指定引脚的模式 GPIO\_Mode 定义在GPIOMode\_TypeDef （跳转 or 查找）



GPIO的引脚模式可以参考中文参考手册的7.3章节



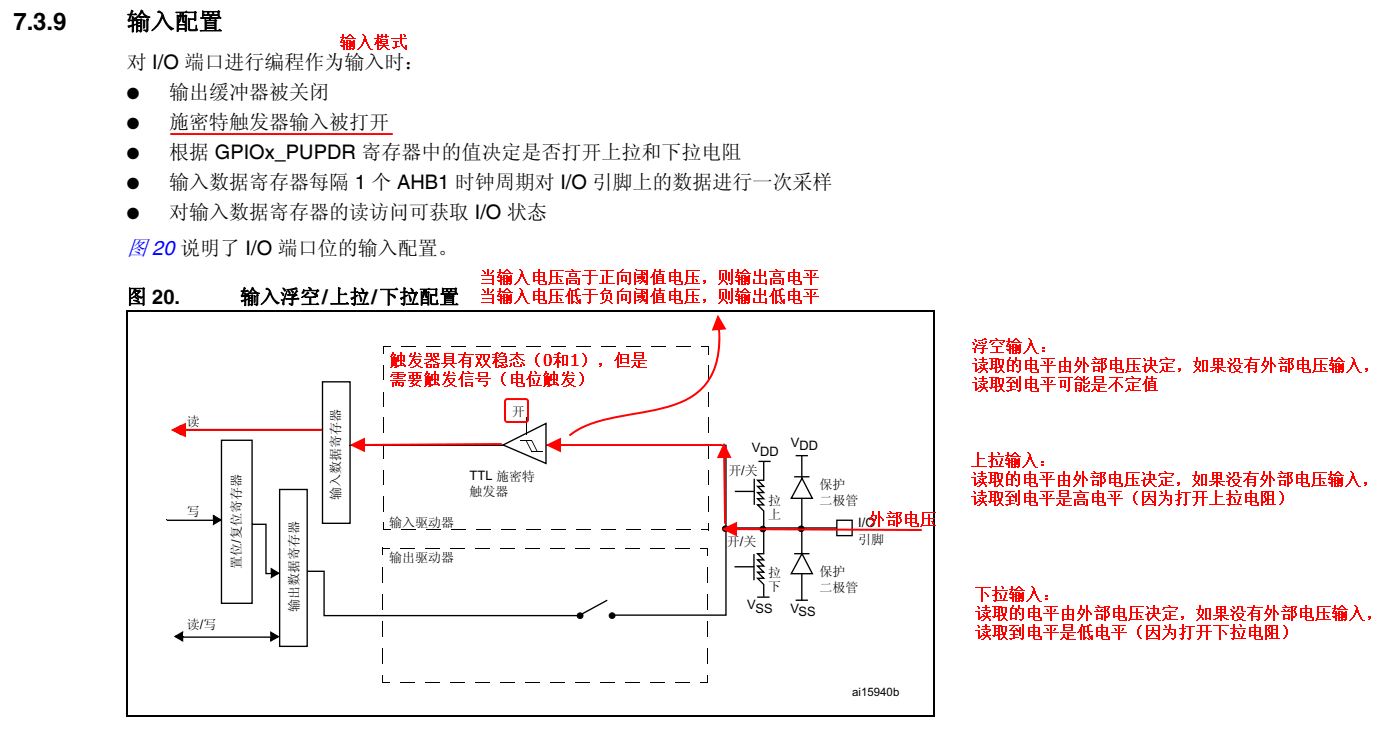


可以看到每个引脚内部都有保护二极管，这也是STM32的安全机制，但是需要注意STM32引脚不是万能的，一般是作为控制使用，而不是作为驱动使用。

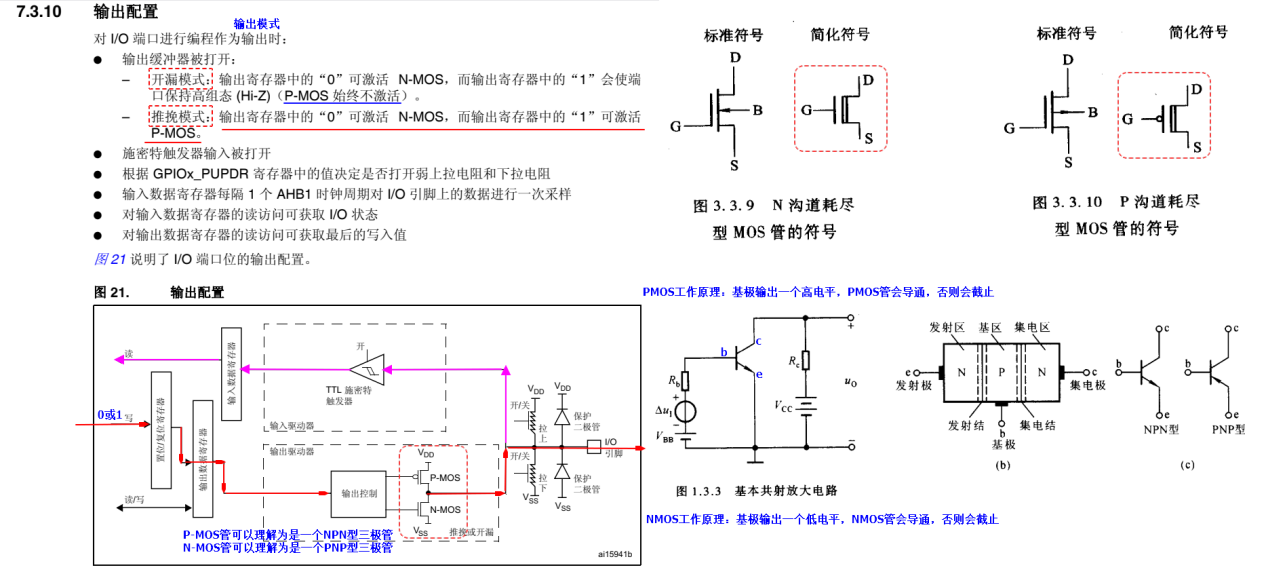
每个引脚内部都有上拉电阻和下拉电阻，但是上拉电阻是一个弱上拉（驱动能力较弱），如果想要提供给引脚一个确定的电平，建议外接上拉电阻。

建议大家在没有使用引脚的时候给引脚一个默认的电平状态，可以消除引脚不定值的影响。

GPIO提供了3种输入模式（浮空输入、上拉输入、下拉输入），参考GPIO内部结构示意图

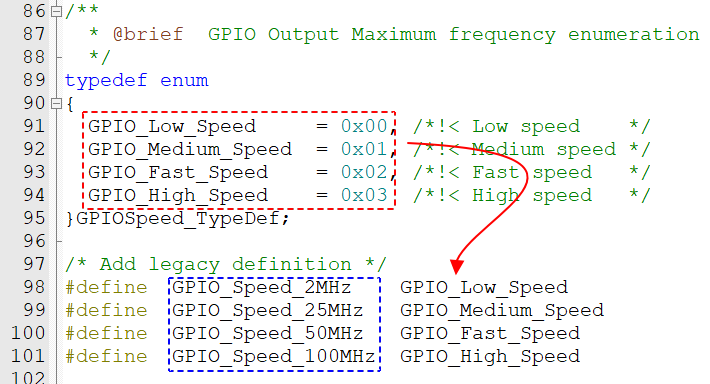


GPIO提供了4种输出模式（上拉+下拉推挽输出、上拉+下拉开漏输出），参考GPIO内部结构示意图



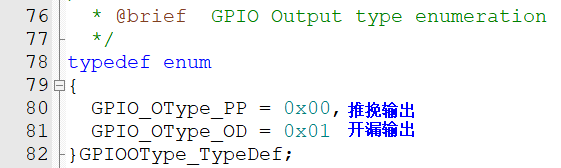
PMOS管和NMOS管循环依次导通，好处是可以提高电路的负载能力和切换速度，并且可以降低功耗。

1. 设置引脚的输出速率 GPIO\_Speed 定义在GPIOSpeed\_TypeDef



指的是IO驱动电路的输出能力（其实就是在不失真的情况下能输出的最大速率），引脚的输出速率影响GPIO的引脚的功耗和噪声，一般大家可以根据实际需要来选择，速率越高，则功耗越高，噪声越大，如果对产品的实时性要求不高，则选择一个低速的即可，有利于降低EMI（电磁干扰），一般指的是影响引脚在切换高低电平的过程中上升沿和下降沿的陡度。如果自己也不知道选哪个，就选高的。

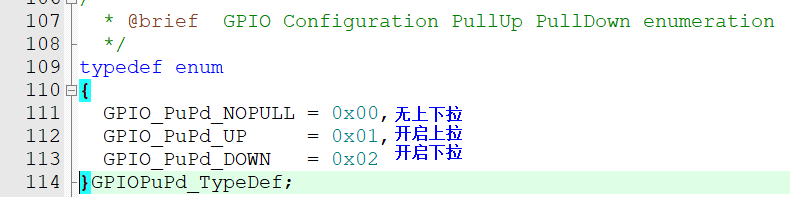
1. 设置引脚的输出类型 GPIO\_OType 定义在GPIOOType\_TypeDef



推挽输出：可以利用驱动器直接输出高电平和低电平

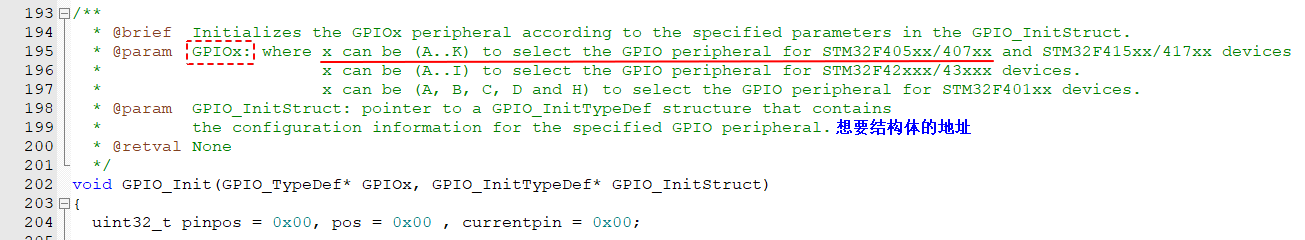
开漏输出：可以利用驱动器直接输出低电平，但是不能直接输出高电平（**需要外接上拉电阻**）

1. 设置引脚的上下拉电阻 GPIO\_PuPd 定义在GPIOPuPd\_TypeDef



一般都会在引脚未使用的情况给引脚一个默认的电平状态，防止引脚出现电平不定（高组态）的情况。

* 在对结构体成员赋值之后，调用初始化函数把结构体成员的值写入到对应外设的寄存器



函数原型

void GPIO\_Init(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* GPIO\_InitStruct)

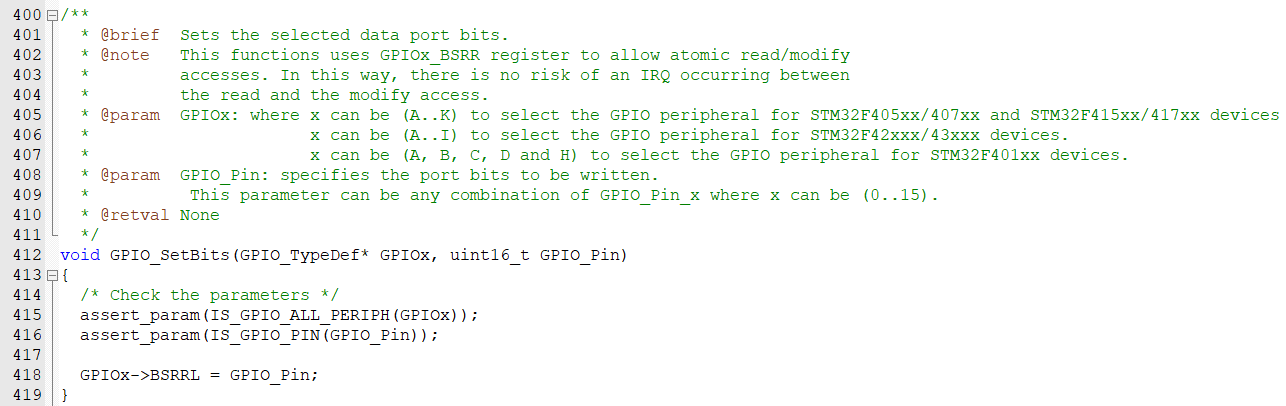
函数参数

参数一：GPIOx 指的是想要初始化的GPIO外设的端口 A~K 如 GPIOF

参数二：GPIO\_InitStruct 指的是配置好的GPIO结构体变量的地址 记得取地址 &

返回值 None

* 设置引脚输出高电平（输出模式有效） 调用 GPIO\_SetBits()



函数原型

void GPIO\_SetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

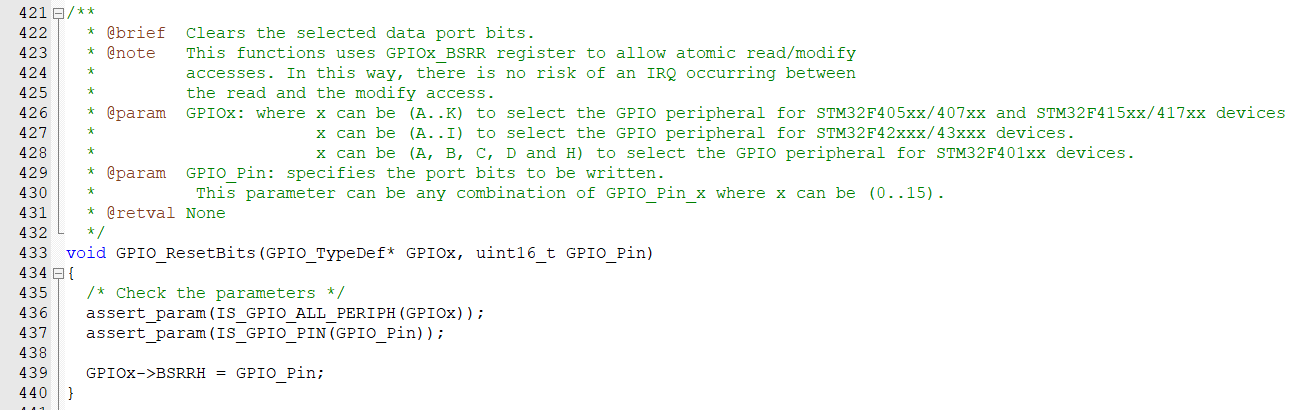
函数参数

参数一：GPIOx 想要输出高电平的引脚端口 如 GPIOx A~K 如 GPIOF

参数二：GPIO\_Pin 想要输出高电平的引脚位置 GPIO\_Pin\_x where x can be (0..15)

返回值 None

* 设置引脚输出低电平（输出模式有效） 调用 GPIO\_ResetBits()



函数原型

void GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

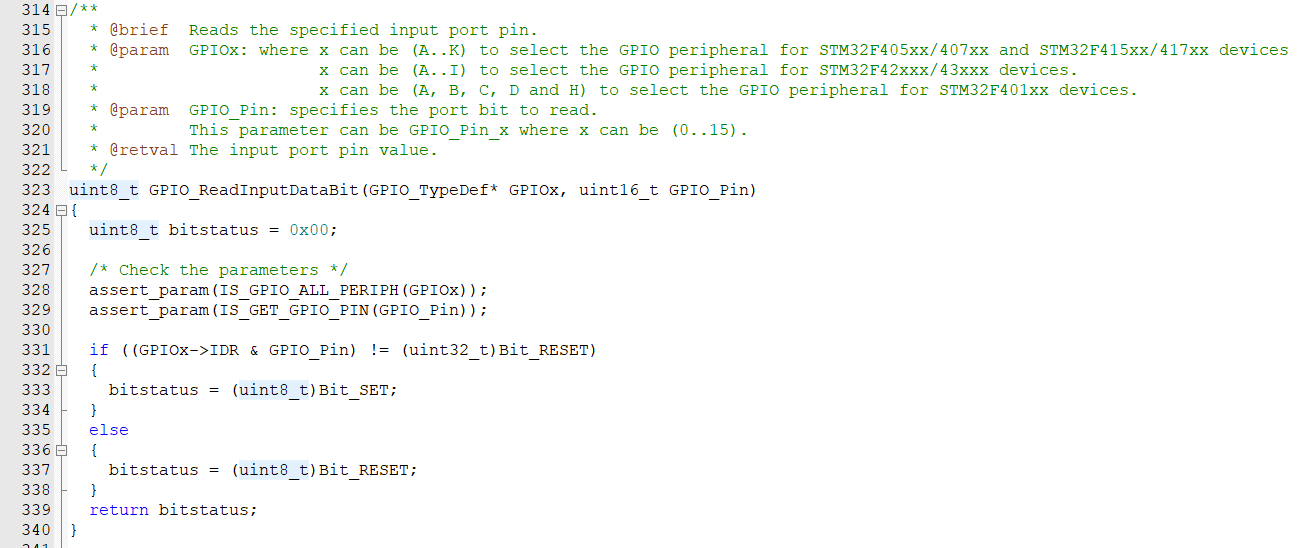
函数参数

参数一：GPIOx 想要输出高电平的引脚端口 如 GPIOx A~K 如 GPIOF

参数二：GPIO\_Pin 想要输出高电平的引脚位置 GPIO\_Pin\_x where x can be (0..15)

返回值 None

* 获取引脚的电平状态（输入模式有效） 调用GPIO\_ReadInputDataBit()



函数原型

uint8\_t GPIO\_ReadInputDataBit(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)

函数参数

参数一：GPIOx 想要读取的引脚端口 如 GPIOx A~K 如 GPIOF

参数二：GPIO\_Pin 想要读取的引脚位置 GPIO\_Pin\_x where x can be (0..15)

返回值 返回读取到的引脚的电平状态 0或者1

练习：编写代码，实现利用库函数点亮4盏LED灯 （要求代码要尽量精简）

练习：编写代码，实现利用开发板的S1按键控制LED0 要求按下点亮，松开熄灭

**作业：利用STM32CubeMx实现四个按键控制四盏灯 （基于HAL库 熟悉软件的配置流程）**

预习：STM32F4中文参考手册的第6章 RCC

晨讲：

**05 写一个函数，用冒泡排序法对整型数组进行升序排序 （奖励一杯奶茶）**

**06 请简述LSB与MSB的区别以及如何判断计算机是大端还是小端 （奖励一杯奶茶）**

