GPIO输入输出

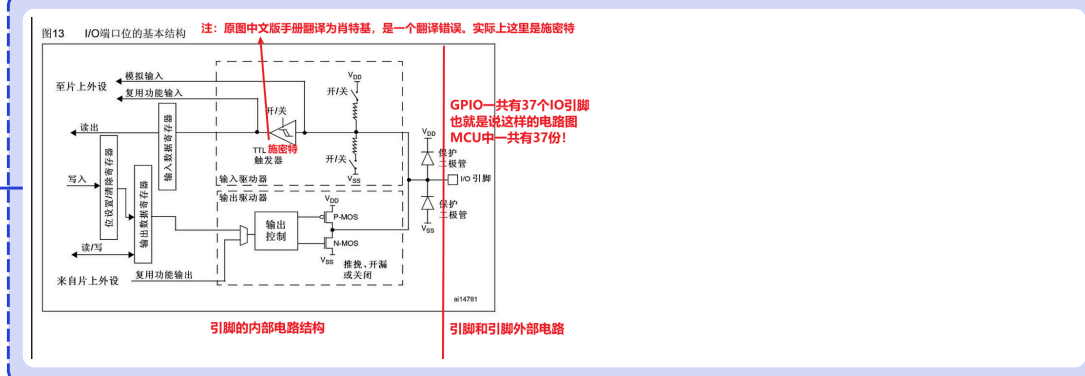
概念：是General Purpose Input Output的首字母缩写直接翻译成中文就是通用输入输出端口，是MCU(单片机)中最基础、且最常用的外设部分

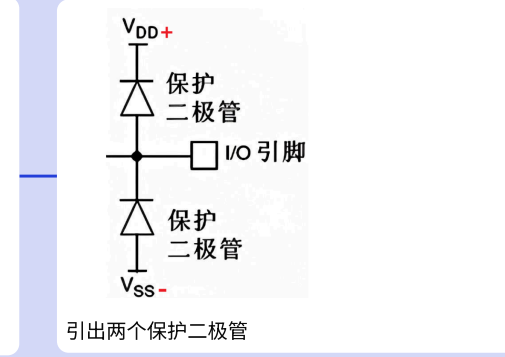
STM32F103C8T6共有四大类GPIO端口

- GPIOA：PA0、PA1、PA2、PA3、PA4、PA5、PA6、PA7、PA8、PA9、PA10、PA11、PA12、PA13、PA14、PA15
- GPIOB：PB0、PB1、PB2、PB3、PB4、PB5、PB6、PB7、PB8、PB9、PB10、PB11、PB12、PB13、PB14、PB15
- GPIOC：PC13、PC14、PC15
- GPIOD：PD0、PD1
  - 其中PD0和PD1引脚常用于配置启用外部晶振，通常都不能用于普通IO引脚，最小系统板上也没有将这两个引脚引出

STM32F103C8T6一共有48个引脚，而GPIO下的普通IO引脚就占据了37个

如何操作引脚？？





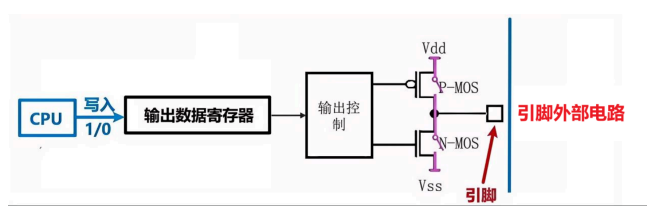
引出两个保护二极管

上面的二极管，我们称之为"正向保护二极管"，它的作用是保护引脚内部电路不会因为过高的电压而损坏它的正极和IO引脚连接，负极接入电源正极。

如果引脚突然输入了一个很高的电压（大于Vdd的电压）时，引脚电压高于VDD，形成电势差，此时上方的正向保护二极管正向偏置，大电流会从引脚流向VDD，从而实现保护内部电路的作用。

下方的二极管，称之为"反向保护二极管"，其原理也是大同小异的。它的正极接入电源负极/接地，负极和IO引脚连接。

如果引脚外部输入的电压非常低，甚至是一个负电压时，引脚电压低于VSS，形成电势差，此时下方的反向保护二极管也会正向偏置（上方的二极管反向偏置，无法形成电流），会形成从VSS到IO引脚的电流，避免引脚过低的电压对内部电路造成损害。



如上图就能看出程序员只需要控制0/1到输出数据寄存器就能控制引脚的高低电平

1.所以在向输出数据寄存器写入0/1控制引脚输出之前，需要先配置引脚的输出模式，这同时是通过寄存器配置来实现的。

2.通过配置寄存器，配置不同的引脚输出模式，两个MOS管会有不同的通断表现。

通用输出模式

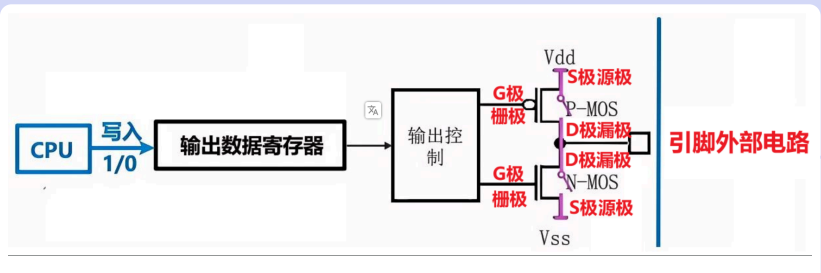
通用推挽输出模式

首先，通用推挽输出模式是最容易理解的输出模式，因为它的名字就表示它的作用：

1.推，也就是push，你可以想象成将电流从引脚"推"出去，也就是引脚输出高电平。
2.挽，一个文艺好听的翻译，但使得它有点不太好理解。挽，其实就是拉，也就是pull，你可以想象成将电流从引脚"拉"回来，电流从高电势流向低电势，所以"拉"电流就意味着引脚输出低电平。
3.向输出数据寄存器中写入0/1就恰好对应这两种状态，而且：
向输出数据寄存器中写入0，表示引脚输出低电平。
向输出数据寄存器中写入1，表示引脚输出高电平。

那么两个MOS该如何开闭才能达成这样的效果呢?

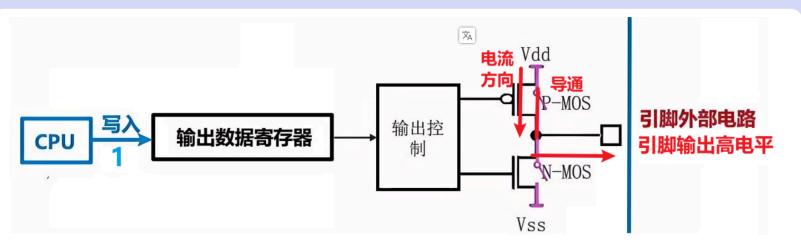
引脚输出高电平



引脚输出高电平-示意图

很明显，这两个MOS管是不可能同时导通的，如果同时导通，PMOS的源极就会直连NMOS的源极，相当于电源正负极直接连在一起就短路。

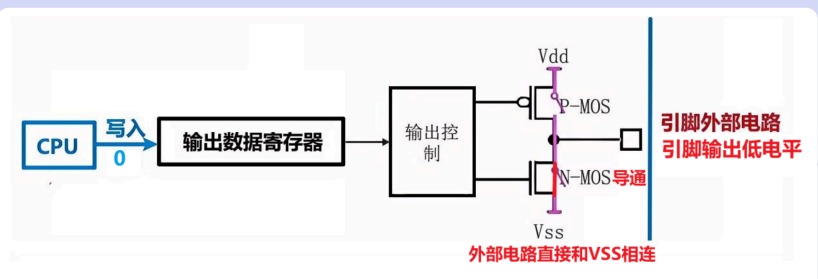
在通用推挽输出模式下，上下两个MOS管是交替闭合的，这就是通用推挽输出模式的MOS管通断表现。



为了引脚能够输出高电平，电路MOS的导通状态一定如上图所示，即上面的PMOS管导通

1.PMOS管导通，所以向PMOS管施加的是低电平，栅极G和源极S存在电势差，PMOS管导通。
2.NMOS管断开，所以向NMOS管施加的也是低电平，栅极G和源极S没有电势差，NMOS管断开。
3.当然有些人会觉得向输出数据寄存器写1，相当于"高电平"，应该在两个MOS的栅极施加高电平，但实际上输出控制电路会将这个高电平转换成低电平施加在栅极。

引脚输出低电平



引脚输出低电平-示意图

1.PMOS管断开，所以向PMOS管施加的是高电平，栅极G和源极S没有电势差，PMOS管断开。
2.NMOS管导通，所以向NMOS管施加的也是高电平，栅极G和源极S有电势差，NMOS管闭合。
3.当然有些人会觉得向输出数据寄存器写0，相当于"低电平"，该在两个MOS的栅极施加低电平，但实际上输出控制电路会将这个低电平转换成高电平施加在栅极。

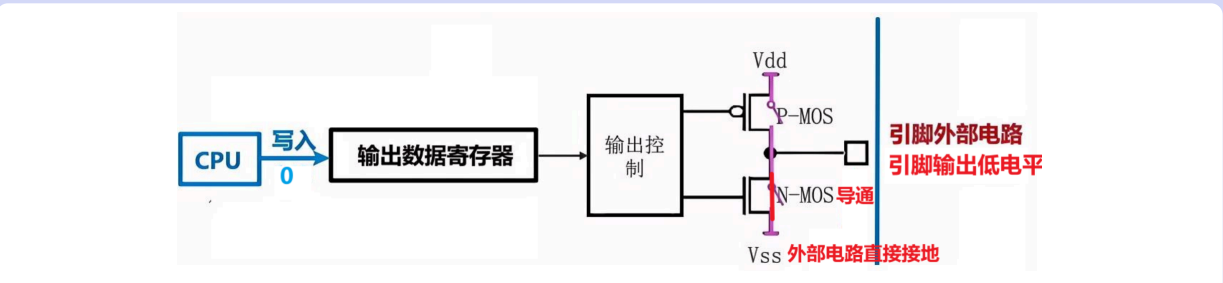
通用开漏输出模式

电路图中，有什么东西和漏有关?

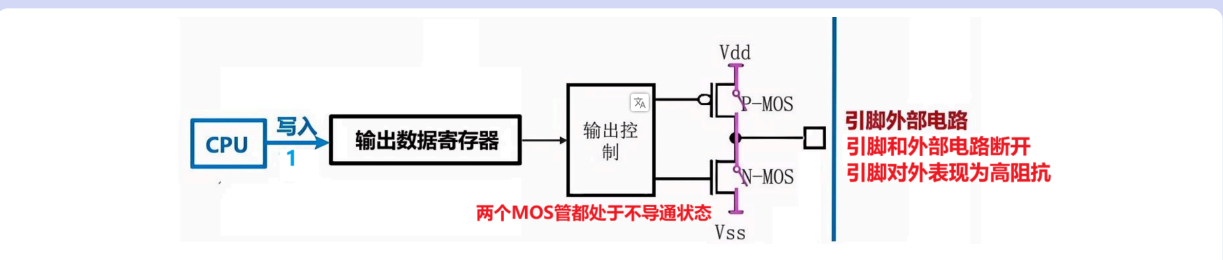
那自然是MOS管的漏极，所以开漏指的是MOS管的漏极处于开路状态，即MOS管的漏极不能和电源正极接在一起。

我们分析电路，发现：
若想实现这一点，就必须让PMOS管始终保持断开状态，也就是让电源正极不能接入电路。

所以当引脚处在通用开漏输出模式时，两个MOS管的通断表现为：
1.PMOS管始终处于断开状态。
2.向输出数据寄存器写入0时，表示引脚输出低电平，此时NMOS管闭合。
3.向输出数据寄存器写入1时，引脚对外部电路表现为高阻抗，也就是电路断开，此时NMOS管也断开。



通用开漏输出模式-引脚输出低电平



通用开漏输出模式-引脚表现为断路悬空，高阻抗

总结：对于 GPIO 普通 IO 引脚 的操作，本质上都是对 GPIO 外设内部寄存器 的读写操作。

对于引脚通用输出模式的使用，需要以下两个步骤：

1.配置引脚的功能和工作模式，涉及GPIO外设的配置寄存器。
2.控制引脚输出状态，涉及向GPIO外设的输出数据寄存器写0或1。
当然，我们采用的是SPL标准外设库的开发方式，寄存器操作都被隐藏在底层了。上述操作在编码实现时，只需调用对应函数即可完成。

现在我们已经学习了两种最常用的GPIO通用输出模式：

通用推挽输出模式：

1.向输出数据寄存器写1，表示引脚输出高电平
2.向输出数据寄存器写0，表示引脚输出低电平
通用开漏输出模式：

1.向输出数据寄存器写1，表示引脚悬空，高阻态。
2.向输出数据寄存器写0，表示引脚输出低电平
很明显可以得出下列结论：两种通用输出模式，都可以输出低电平。只有通用推挽输出模式，可以输出高电平。

引脚的输出模式主要用于驱动外部电路，进一步可以得出下列重要结论：

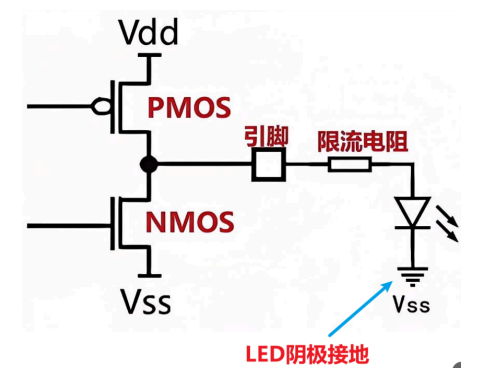
1.如果外部电路需要高电平驱动，则必须选择通用推挽输出模式。
2.如果外部电路只需要低电平就可以驱动，则两种通用输出模式实际上都可以。

电路推挽和开漏输出模式选择

为了点亮PC13指示灯，我们需要进行什么样的引脚操作呢?

为了讲清楚这一点，我们不妨先来看一看若希望使用引脚操作来控制一盏LED的亮灭，需要设计怎样的电路。我们可以直接将简化的引脚电路和LED相连接起来，按照输出模式的不同，分为两种情况：

1.推挽输出模式接法
2.开漏输出模式接法



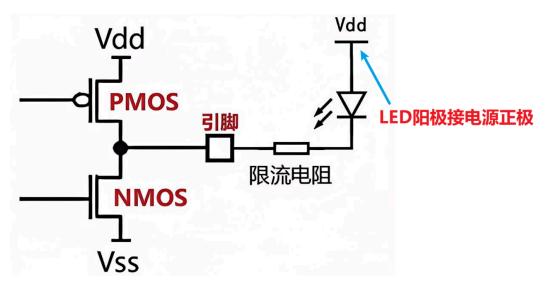
推挽输出点亮LED

推挽输出模式下，引脚可以输出高电平和低电平。在这种情况下，为了能够控制LED的亮灭，LED的另一端必须接入低电平（接地）。

这样：

1.引脚输出高电平，LED亮
2.引脚输出低电平，LED灭
结合的寄存器的操作，如果采用通用推挽输出模式接法，操作流程是这样的：

1.通过配置引脚寄存器，将IO引脚的工作模式设置为推挽输出模式。
2.向输出数据寄存器中写1，使得引脚输出高电平，于是点亮LED
3.向输出数据寄存器中写0，使得引脚输出低电平，于是熄灭LED



开漏输出点亮LED

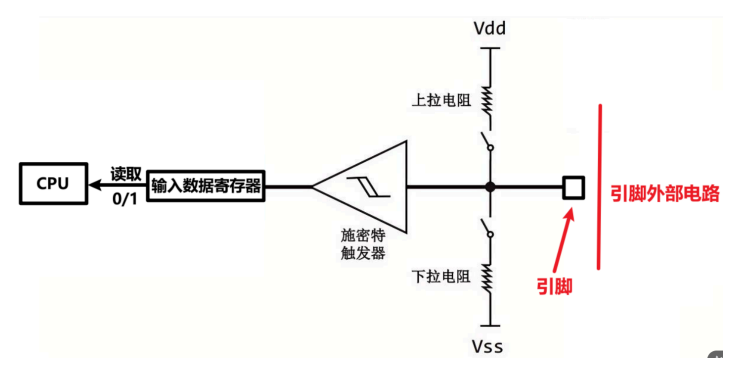
开漏输出模式下，引脚可以输出低电平和高阻抗。在这种情况下，为了能够控制LED的亮灭，LED的另一端必须接入高电平（电源正极）。

这样：

1.引脚输出低电平，LED亮
2.引脚输出高阻抗，LED灭
结合的寄存器的操作，如果采用通用开漏输出模式接法，操作流程是这样的：

1.通过配置引脚寄存器，将IO引脚的工作模式设置为开漏输出模式。
2.向输出数据寄存器中写0，使得引脚输出低电平，即点亮LED
3.向输出数据寄存器中写1，引脚对外表现为高阻态，此时LED熄灭。
也就是说，使用推挽输出和开漏输出两种输出模式都可以实现点灯，但需要连接的电路是不同的，寄存器的操作也稍有区别。

通用输入模式



输入模式电路简化

施密特触发器

施密特触发器是电平转换电路的核心器件，我们先来简单了解一下。

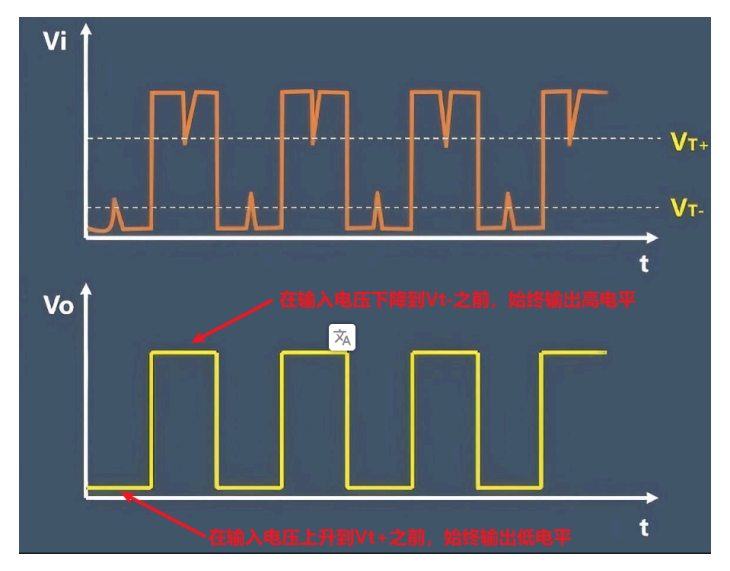
简单来说，施密特触发器会把输入的连续的、模拟电压信号，转换成高低电平的1和0数字信号，并写入输入数据寄存器。

也就是说施密特触发器是一种将模拟信号转换成数字信号的一种元器件。

具体来说，施密特触发器有两个核心关键的参数：

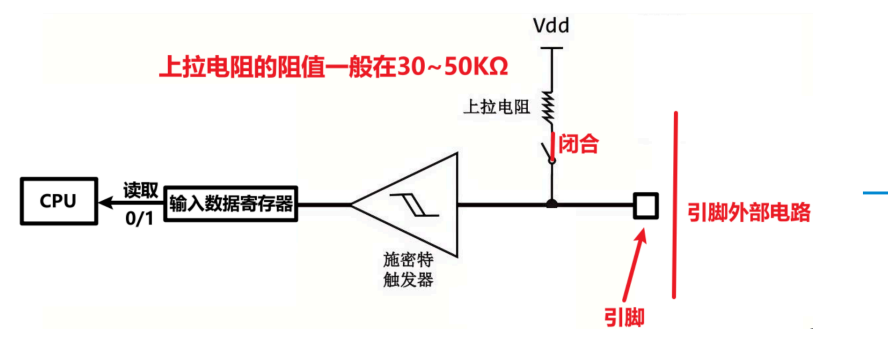
1.正向阈值电压Vt+: 当输入电压大于Vt+时，施密特触发器输出高电平，也就是向输入数据寄存器写1。
2.负向阈值电压Vt-：当输入电压小于Vt-时，施密特触发器输出低电平，也就是向输入数据寄存器写0。
也就是说：

1.当输入电压从0V开始一直持续上升，在上升到正向阈值电压Vt+之前，施密特触发器始终输出低电平。只有当输入电压超过Vt+时，输出才变为高电平。
2.当输入电压从Vt+开始一直持续下降，在下降到负向阈值电压Vt-之前，施密特触发器始终输出高电平。只有当输入电压低于Vt-时，输出才变为低电平。



施密特触发器

上拉输入模式



上拉输入模式

当我们将IO引脚配置为上拉输入模式时，上拉电阻部分的开关就会闭合，上拉电阻接入电路。而下拉电阻部分的开关并不会闭合，此时下拉电阻就没有接入电路 。 上面我们已经讲过施密特触发器的作用了，那么电路中的这个上拉电阻有什么作用呢?

上拉电阻的主要作用是为引脚提供一个默认的高电平输入，即便引脚未连接任何有效的外部信号源。

上拉电阻一端连接电源Vdd，另一端连接引脚，通过闭合的开关构成通路。

此时电流从电源经过上拉电阻流向引脚，使得引脚的输入电压接近电源电压，呈现高电平状态，此时输入数据寄存器中的值就是高电平"1"，这种设计就避免了引脚处于悬空状态而导致的电平不确定（或者电平的反复横跳）。

由于上拉电阻的阻值很大，Vdd电压也不高（3.3V），上拉支路中的电流十分微弱。

当 GPIO 引脚外部由强驱动能力的电路（如推挽输出）稳定输出高或低电平时，电平主要由外部电路决定。

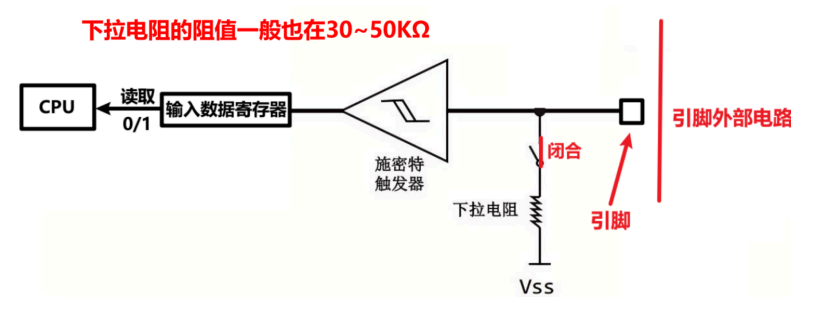
上拉电阻几乎不影响电平状态，其作用通常可忽略不计。

总之，当引脚设置为上拉输入模式时：

1.若引脚输入了高电平，此时读取输入数据寄存器，可以读到代表高电平的"1"信号。
2.若引脚输入了低电平或者引脚接地，此时读取输入数据寄存器，可以读到代表低电平的"0"信号。
3.若引脚处于悬空状态，无外部信号输入时，由于上拉电阻的影响，此时读取输入数据寄存器，仍然可以读到代表高电平的"1"信号。
什么时候使用上拉输入模式呢?

希望IO引脚悬空，外部电路断开时，引脚具有默认的高电平时都可以使用上拉输入模式。

下拉输入模式



下拉输入模式

当我们将IO引脚配置为下拉输入模式时，下拉电阻部分的开关就会闭合，下拉电阻接入电路。而上拉电阻则不会接入电路 。 在这个电路中，下拉电阻的主要作用是为引脚提供一个默认的低电平输入，即便引脚未连接任何有效的外部信号源。

下拉电阻的一端接地，另一端连接引脚，当引脚悬空时，下拉电阻使引脚电位接近地电平，呈现低电平状态。

此时输入数据寄存器就能够得到一个稳定的电平信号"0"，这种设计就避免了引脚处于悬空状态而导致的电平不确定（或者电平的反复横跳）。

由于下拉电阻阻值也非常大，其下拉支路中形成的电流也非常小。

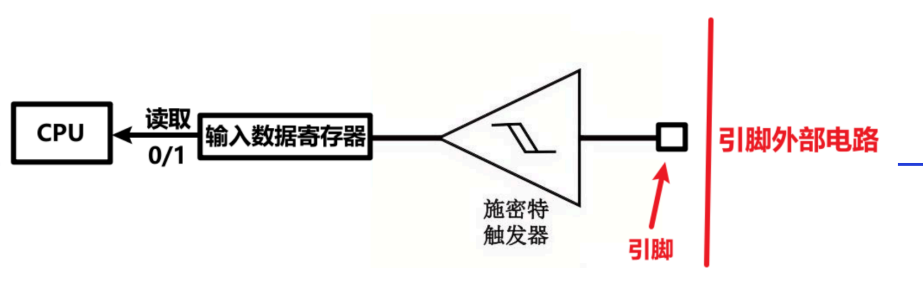
当 GPIO 引脚由强驱动电路（如推挽输出）稳定输入高电平或低电平时，电平由外部主导，下拉电阻对引脚输入电平的影响可以忽略。

总之，当引脚设置为下拉输入模式时：

1.若引脚输入了高电压，此时读取输入数据寄存器，可以读到代表高电平的"1"信号。
2.若引脚输入了低电压或者引脚接地，此时读取输入数据寄存器，可以读到代表低电平的"0"信号。
3.若引脚处于悬空状态，无外部信号输入时，由于下拉电阻的影响，此时读取输入数据寄存器，仍然可以读到代表低电平的"0"信号。
什么时候使用下拉输入模式呢?

希望IO引脚悬空，外部电路断开时，引脚具有默认的低电平时都可以使用下拉输入模式。

浮空输入模式



浮空输入模式

浮空输入模式意味着上、下两个电阻都不会接入电路，那么直接将它们从电路中删除 。 此时引脚直接与施密特触发器相连，然后连接到输入数据寄存器，CPU 通过读取输入数据寄存器来获取引脚的输入状态信息。

此时引脚的电平状态完全由外部连接的电路决定，内部没有默认的上拉或下拉机制来设定默认引脚电平。也就是说：

1.如果外部电路稳定输入高电平，那么输入数据寄存器中存储的数据就稳定是1。
2.如果外部电路稳定输入低电平，那么输入数据寄存器中存储的数据就稳定是0。
3.如果外部电路是断路悬空的，那么外部的信号是无法确定的，此时输入数据寄存器中的数值是无意义的"垃圾值"。
浮空输入模式常用于需要检测外部电路真实电平状态，且外部电路能够提供稳定的高电平或低电平信号的场景。
比如外部电路是推挽输出的。

模拟输入模式

扩展：上拉电阻和下拉电阻的作用

在上面我们已经讲过，如果外部电路是强推挽输出，能够主动、稳定地推高和拉低电平：

那么无论是上拉电阻还是下拉电阻，其产生的影响都相对较小，在多数情况下可以忽略它们的作用。

你可以理解成：强推挽是一条壮汉，无论上拉电阻还是下拉电阻，都是一名弱女子，力量相比较于壮汉都还太弱小了。

但如果排除外界条件，只讨论它们本身对电路产生的影响，那差别就比较大了。

我们可以从两个角度来分析它们的不同。

第一，通俗的理解。

上拉的本质可以通俗理解成，给节点电路充电，或者说补充电荷。

下拉的本质可以通俗理解成，给节点电路放点，或者说释放电荷。

很浅显的道理：

放电总比充电容易多了。

所以下拉往往更容易对电路产生影响。

比如电路本身是输出高电平的，接入一个下拉电阻，有可能就会使得电路不能稳定输出高电平。

反之，上拉电阻要想实现上拉是很困难的。

比如电路本身输出低电平，好比电路中完全没电，现在接一个上拉电阻，指望上拉电阻把这个低电平拉成高电平，是几乎不可能的。

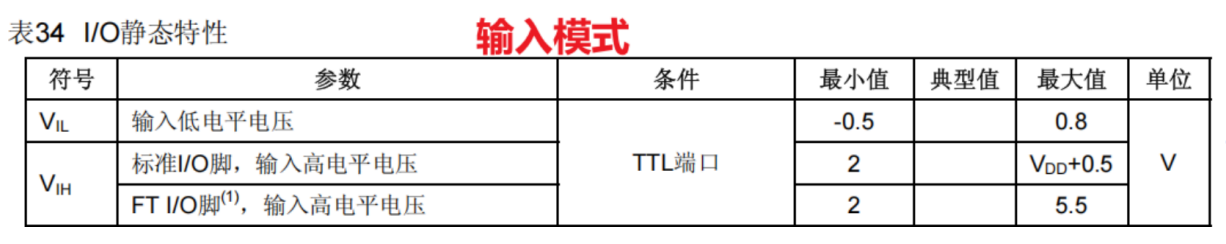


表34 I/O静态特性 **输入模式**

符号	参数	条件	最小值	典型值	最大值	单位
V_{IL}	输入低电平电压	TTL端口	-0.5		0.8	V
V_{IH}	标准I/O脚，输入高电平电压		2		V_{DD}+0.5	
	FT I/O脚[1]，输入高电平电压		2		5.5	

电平阈值

低电平的范围是：-0.5V ~ 0.8V，阈值是0.8，低于0.8V都视为低电平。

高电平的范围是：2V ~ 3.8V，阈值是2，高于2V都视为高电平。

假如，电路输出3.3V，输出高电平，此时接入下拉电阻，只需要把输出电压拉到2V以下，就不再是稳定高电平状态了。

反之，电路接地输出低电平0V，此时接入上拉电阻，需要把电压拉到2V才可能变成变成高电平，这显然困难多了。

总结：

在电路中接入一个上拉电阻，不会影响电路原本的高电平状态，也很难破坏电路本来的低电平状态。
在电路中接入一个下拉电阻，不会影响电路的低电平状态，却可能破坏电路本身的高电平状态。
对于实践使用的建议：

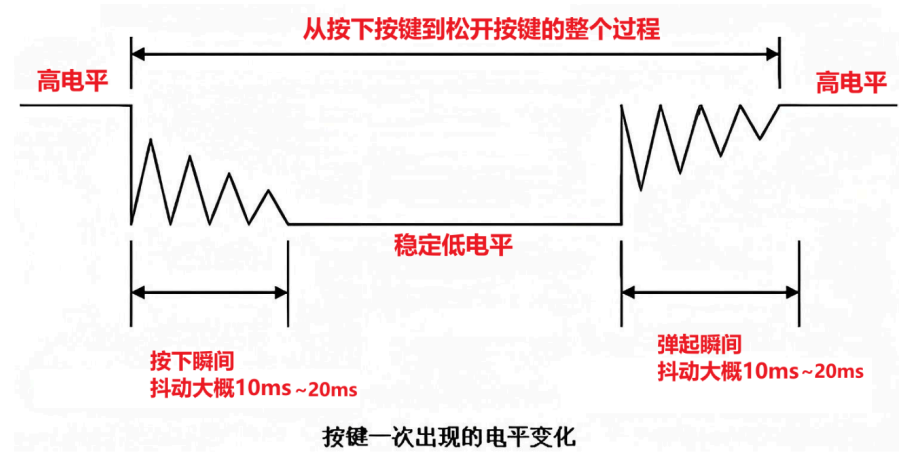
上拉电阻，以及对应的上拉输入模式，在使用时可以更随意一些、使用场景也更广泛一些。

因为上拉电阻的作用就是在电路断路时，提供稳定高电平，其余情况下作用不明显。
很多时候，只要能用浮空输入，往往也能用上拉输入。
下拉电阻，以及对应的下拉输入模式，在使用时则要更加小心谨慎。

因为下拉电阻除了可以在电路断路时提供稳定低电平外，还可能破坏原本电路的高电平状态。
很多时候，下拉电阻对高电平状态的破坏，都会使得下拉输入模式不可用!

按键抖动

所谓按键抖动，指的是：由于按键内部使用的是机械式弹簧片来进行通断的，所以在按下和松手的瞬间会伴随有一连串的、不稳定的电平抖动。



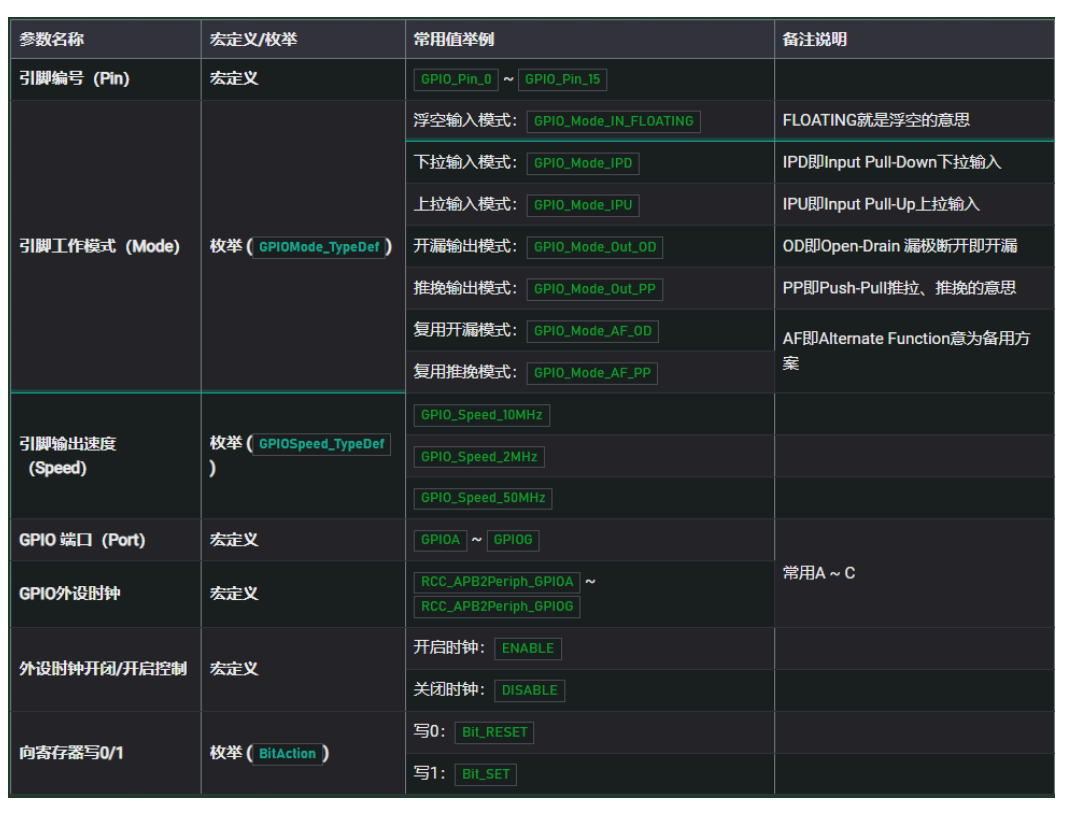
按压按键一次电平变化

按压按键一次电平变化-图 按压按键一次电平变化-图

为了规避抖动对最终电平结果的影响，一个常用好用的办法就是加延时。

于是，我们可以提取一个函数用于确定按键是否按下弹起，即返回按键状态的函数，此函数的行为是这样的：

若按键按下且已弹起，则程序需要切换LED状态，此时函数返回一个整数值1。
若按键没有按下，则程序需要保持LED当前状态，此时函数返回一个整数值0。



参数名称	宏定义/枚举	常用值举例	备注说明
引脚编号（Pin）	宏定义	GPIO_Pin_0 ~ GPIO_Pin_15	
引脚工作模式（Mode）	枚举（GPIOMode_TypeDef）	浮空输入模式：GPIO_Mode_IN_FLOATING	FLOATING就是浮空的意思
		下拉输入模式：GPIO_Mode_IPD	IPD即Input Pull-Down下拉输入
		上拉输入模式：GPIO_Mode_IPU	IPU即Input Pull-Up上拉输入
		开漏输出模式：GPIO_Mode_Out_OD	OD即Open-Drain 漏极断开即开漏
		推挽输出模式：GPIO_Mode_Out_PP	PP即Push-Pull推拉、推挽的意思
		复用开漏模式：GPIO_Mode_AF_OD	AF即Alternate Function意为备用方案
		复用推挽模式：GPIO_Mode_AF_PP	
引脚输出速度（Speed）	枚举（GPIOSpeed_TypeDef）	GPIO_Speed_10MHz	
		GPIO_Speed_2MHz	
		GPIO_Speed_50MHz	
GPIO 端口（Port）	宏定义	GPIOA ~ GPIOG	常用A ~ C
GPIO外设时钟	宏定义	RCC_APB2Periph_GPIOA ~ RCC_APB2Periph_GPIOG	
外设时钟开闭/开启控制	宏定义	开启时钟：ENABLE	
		关闭时钟：DISABLE	
向寄存器写0/1	枚举（BitAction）	写0：Bit_RESET	
		写1：Bit_SET	

Presented with xmind