一、GPIO配置  
（1）GPIO\_Mode\_AIN 模拟输入   
（2）GPIO\_Mode\_IN\_FLOATING 浮空输入  
（3）GPIO\_Mode\_IPD 下拉输入   
（4）GPIO\_Mode\_IPU 上拉输入   
（5）GPIO\_Mode\_Out\_OD 开漏输出  
（6）GPIO\_Mode\_Out\_PP 推挽输出  
（7）GPIO\_Mode\_AF\_OD 复用开漏输出   
（8）GPIO\_Mode\_AF\_PP 复用推挽输出  
GPIO\_Speed\_10MHz 最高输出速率10MHz   
GPIO\_Speed\_2MHz 最高输出速率2MHz   
GPIO\_Speed\_50MHz 最高输出速率50MHz

1.1 I/O口的输出模式下，有3种输出速度可选(2MHz、10MHz和50MHz)，这个速度是指I/O口驱动电路的响应速度而不是输出信号的速度，输出信号的速度与程序有关（芯片内部在I/O口 的输出部分安排了多个响应速度不同的输出驱动电路，用户可以根据自己的需要选择合适的驱动电路）。通过选择速度来选择不同的输出驱动模块，达到最佳的噪声 控制和降低功耗的目的。高频的驱动电路，噪声也高，当不需要高的输出频率时，请选用低频驱动电路，这样非常有利于提高系统的EMI性能。当然如果要输出较高频率的信号，但却选用了较低频率的驱动模块，很可能会得到失真的输出信号。

关键是GPIO的引脚速度跟应用匹配（推荐10倍以上？）。比如：  
1.1.1 对于串口，假如最大波特率只需115.2k，那么用2M的GPIO的引脚速度就够了，既省电也噪声小。  
1.1.2 对于I2C接口，假如使用400k波特率，若想把余量留大些，那么用2M的GPIO的引脚速度或许不够，这时可以选用10M的GPIO引脚速度。  
1.1.3 对于SPI接口，假如使用18M或9M波特率，用10M的GPIO的引脚速度显然不够了，需要选用50M的GPIO的引脚速度。  
1.2 GPIO口设为输入时，输出驱动电路与端口是断开，所以输出速度配置无意义。  
1.3 在复位期间和刚复位后，复用功能未开启，I/O端口被配置成浮空输入模式。  
1.4 所有端口都有外部中断能力。为了使用外部中断线，端口必须配置成输入模式。  
1.5 GPIO口的配置具有上锁功能，当配置好GPIO口后，可以通过程序锁住配置组合，直到下次芯片复位才能解锁。

2、推挽输出与开漏输出的区别

推挽输出:可以输出高,低电平,连接数字器件;开漏输出:输出端相当于三极管的集电极. 要得到高电平状态需要上拉电阻才行. 适合于做电流型的驱动,其吸收电流的能力相对强(一般20ma以内).  
推挽结构一般是指两个三极管分别受两互补信号的控制,总是在一个三极管导通的时候另一个截止.  
要实现 线与 需要用OC(open collector)门电路.是两个参数相同的三极管或MOSFET,以推挽方式存在于电路中,各负责正负半周的波形放大任务,电路工作时，两只对称的功率开关管每次只有一个导通，所以导通损耗小,效率高。输出既可以向负载灌电流，也可以从负载抽取电流

当端口配置为输出时：   
开漏模式：输出 0 时，N-MOS 导通，P-MOS 不被激活，输出0。  
输出 1 时，N-MOS 高阻， P-MOS 不被激活，输出1（需要外部上拉电路）；此模式可以把端口作为双向IO使用。  
推挽模式：输出 0 时，N-MOS 导通，P-MOS 高阻 ，输出0。  
输出 1 时，N-MOS 高阻，P-MOS 导通，输出1（不需要外部上拉电路）。

简单来说开漏是0的时候接GND 1的时候浮空 推挽是0的时候接GND 1的时候接VCC

3、在STM32中选用IO模式  
（1） 浮空输入\_IN\_FLOATING ——浮空输入，可以做KEY识别，RX1  
（2）带上拉输入\_IPU——IO内部上拉电阻输入  
（3）带下拉输入\_IPD—— IO内部下拉电阻输入   
（4） 模拟输入\_AIN ——应用ADC模拟输入，或者低功耗下省电  
（5）开漏输出\_OUT\_OD ——IO输出0接GND，IO输出1，悬空，需要外接上拉电阻，才能实现输出高电平。当输出为1时，IO口的状态由上拉电阻拉高电平，但由于是开漏输出模式，这样IO口也就可以由外部电路改变为低电平或不变 。可以读IO输入电平变化，实现C51的IO双向功能  
（6）推挽输出\_OUT\_PP ——IO输出0-接GND， IO输出1 -接VCC，读输入值是未知的  
（7）复用功能的推挽输出\_AF\_PP ——片内外设功能（I2C的SCL,SDA）  
（8）复用功能的开漏输出\_AF\_OD——片内外设功能（TX1,MOSI,MISO.SCK.SS）

实例总结：

（1）模拟I2C使用推挽复用\_OUT\_OD，接上拉电阻，能够正确输出0和1；读值时先  
GPIO\_SetBits(GPIOB, GPIO\_Pin\_0)；拉高，然后可以读IO的值；使用  
GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_0)；

（2）如果是无上拉电阻，IO默认是高电平；需要读取IO的值，可以使用  
带上拉输入\_IPU和浮空输入\_IN\_FLOATING和 开漏输出\_OUT\_OD；

4、IO低功耗：

关于模拟输入&低功耗，根据STM32的低功耗AN（AN2629）及其源文件，在STOP模式下，为了得到尽量低的功耗，确实把所有的IO（包括非A/D输入的GPIO）都设置为模拟输入

5、程序

（1）时钟：  
RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_GPIOB |  
RCC\_APB2Periph\_GPIOC, ENABLE);

（2）IO配置：  
GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_8; // IR 输入  
GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_10MHz;  
GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_OD;

GPIO\_Init(GPIOC, &GPIO\_InitStructure);  
GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_0|GPIO\_Pin\_1|GPIO\_Pin\_15;   
GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_2MHz;  
GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;  
GPIO\_Init(GPIOB, &GPIO\_InitStructure);

（3）输出输入：

输出0：GPIO\_ResetBits(GPIOB, GPIO\_Pin\_0)  
输出1：GPIO\_SetBits(GPIOB, GPIO\_Pin\_0)  
输入： GPIO\_ReadInputDataBit(GPIOB,GPIO\_Pin\_7)