目录

[目录 1](#_Toc16005870)

[第2章 STM32F407VG-GPIO 2](#_Toc16005871)

[2.1 STM32F407VG-GPIO-介绍 2](#_Toc16005872)

[2.2 STM32F407VG-GPIO-工作原理 2](#_Toc16005873)

[2.2.1 芯片IO口框架分析 2](#_Toc16005874)

[2.2.2 芯片IO口普通输出功能分析 3](#_Toc16005875)

[2.2.3 芯片IO口普通输入功能分析 3](#_Toc16005876)

[2.2.4 芯片IO口复用功能分析 4](#_Toc16005877)

[2.2.5 芯片IO口模拟功能分析 5](#_Toc16005878)

[2.3 STM32F407VG-GPIO-相关寄存器 5](#_Toc16005879)

[2.4 STM32F407VG-GPIO-实验 10](#_Toc16005880)

[2.4.1 点亮LED灯 10](#_Toc16005881)

# STM32F407VG-GPIO

## STM32F407VG-GPIO-介绍

GPIO作用：输出信号控制外部器件工作；读取外部信号，用来进行数据处理从而达到控制外部器件的作用。

GPIO口就是芯片的引脚，不是所有的引脚都是GPIO口。

STM32的GPIO口是按照分组的形式存在，每组IO口数量都是：16个，编号都是PX0~15.

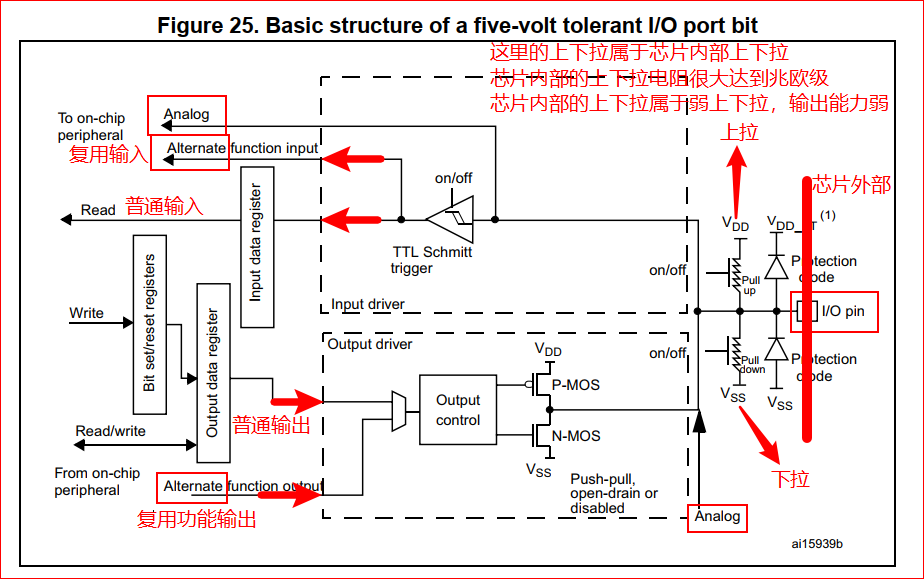
GPIO口作为GPIO外设使用的时候是普通IO功能；GPIO口作为其它外设使用的时候是复用IO功能。

概念补充：

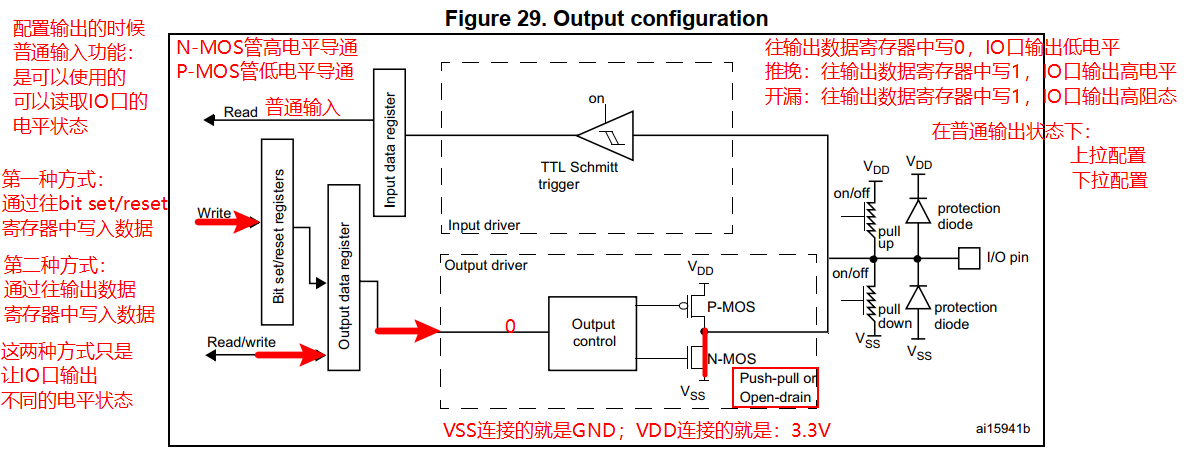
|  |
| --- |
| 推挽/开漏：  推挽：输出能力强，输出电流大，驱动能力强；一般用来驱动外部的三极管、MOS管、LED灯、蜂鸣器等需要大电流设备。  开漏：输出能力弱，输出电流小，驱动能力弱；一般用来作为总线通信中的输入数据线。(I2C总线)  输出速度：  速度越快，芯片的功耗越高，配置IO口速度的时候必须能够按照合理的速度进行配置，否则会浪费资源。  上拉/下拉：  上拉：利用一个电阻与电源正极连接(VCC)。    下拉：利用一个电阻与电源负极连接(GND)。 |

## STM32F407VG-GPIO-工作原理

### 芯片IO口框架分析



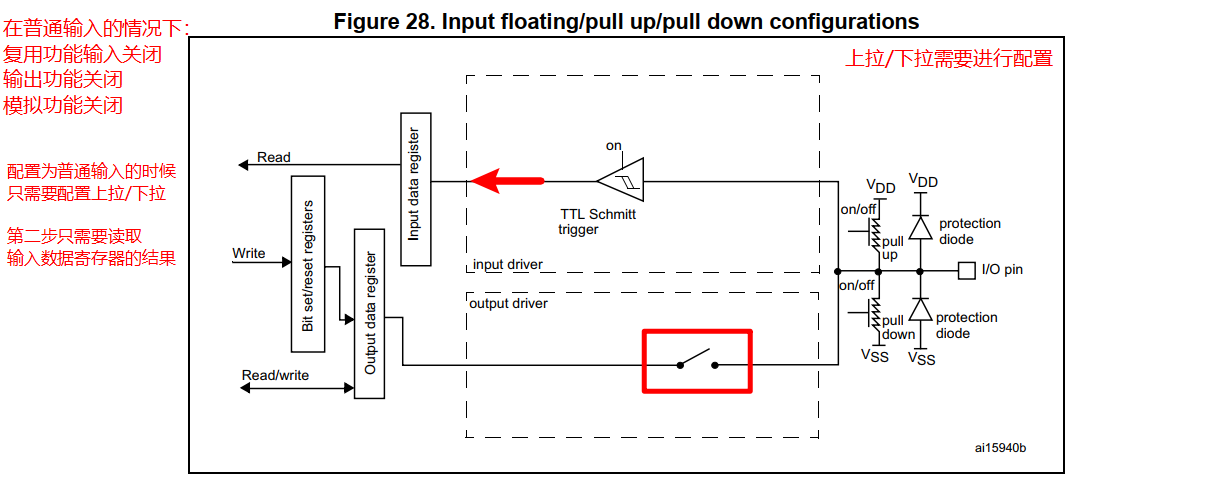
### 芯片IO口普通输出功能分析



想要配置IO口为普通输出，输出一定的电平状态，IO口初始配置要有以下几步：

1. 配置IO口模式，使得IO口进入普通输出功能
2. 配置IO口输出类型，使得IO口为推挽/开漏
3. 配置IO口上拉或下拉，根据实际情况选择；一般来说，推挽功能不用上拉也不用下拉
4. 配置IO口的速度，根据实际情况选择
5. 配置IO口的初始电平状态

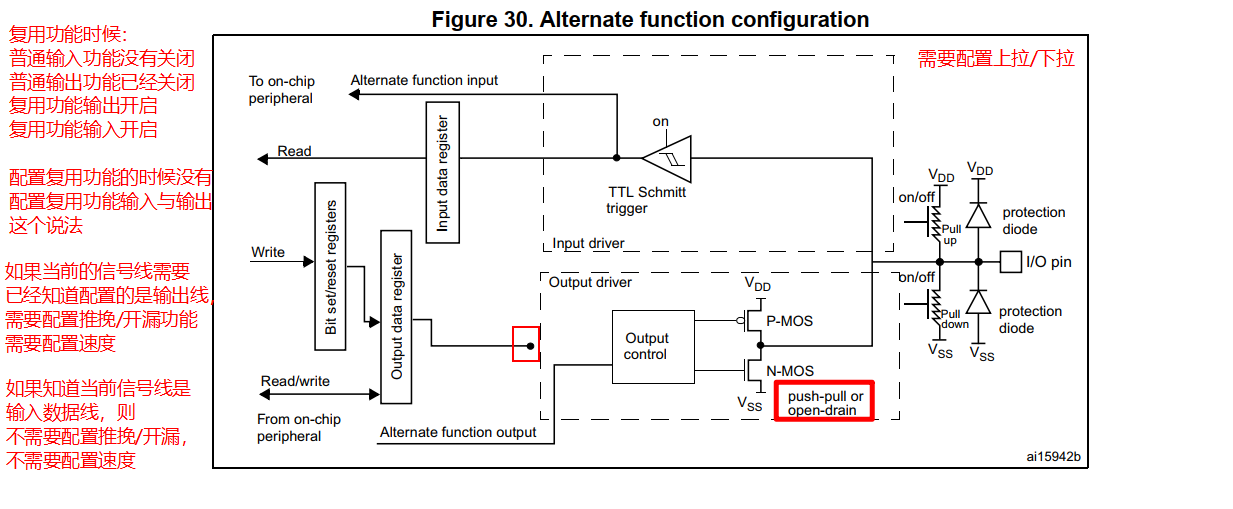
### 芯片IO口普通输入功能分析

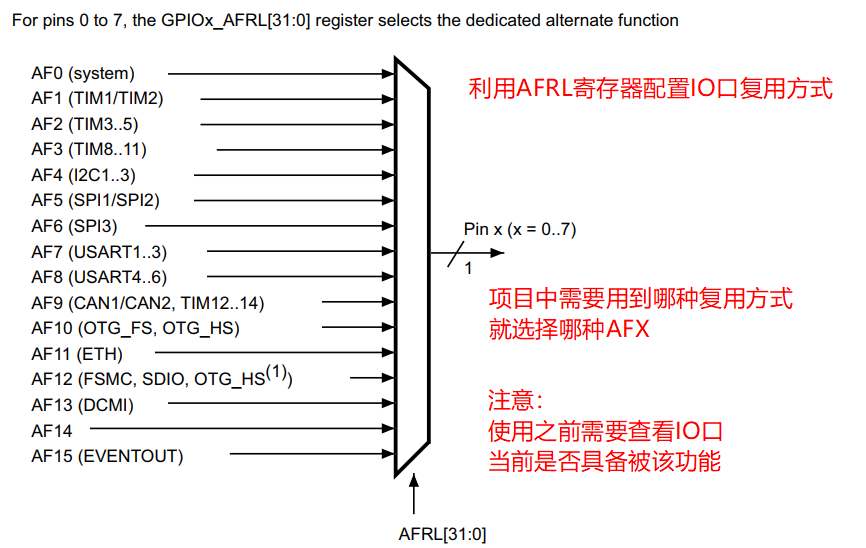


想要配置普通输入功能，需要完成的初始步骤：

1. 配置IO口的模式，选择IO口的模式为普通输入
2. 配置上拉/下拉；根据实际情况选择；如果外部有了硬件上拉/下拉，就不需要配置上拉/下拉。如果没有外部上拉/下拉，一般根据实际情况选择上拉/下拉，目的：给定一个初始状态

### 芯片IO口复用功能分析





使用复用功能的前提：当前IO口需要使用到芯片内部的其它外设(这个外设不是IO口外设功能)

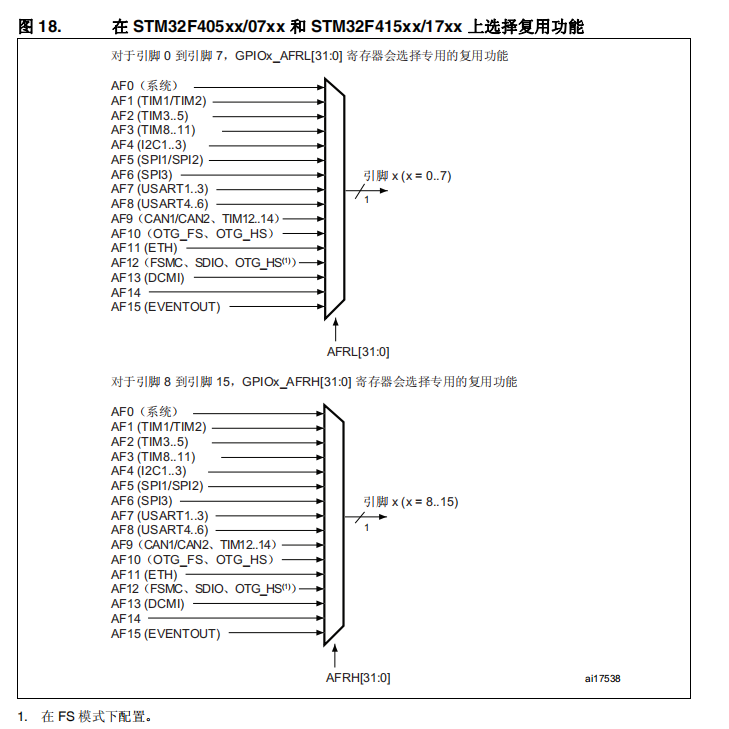
想要配置复用功能，操作步骤如下：

复用功能输出配置：

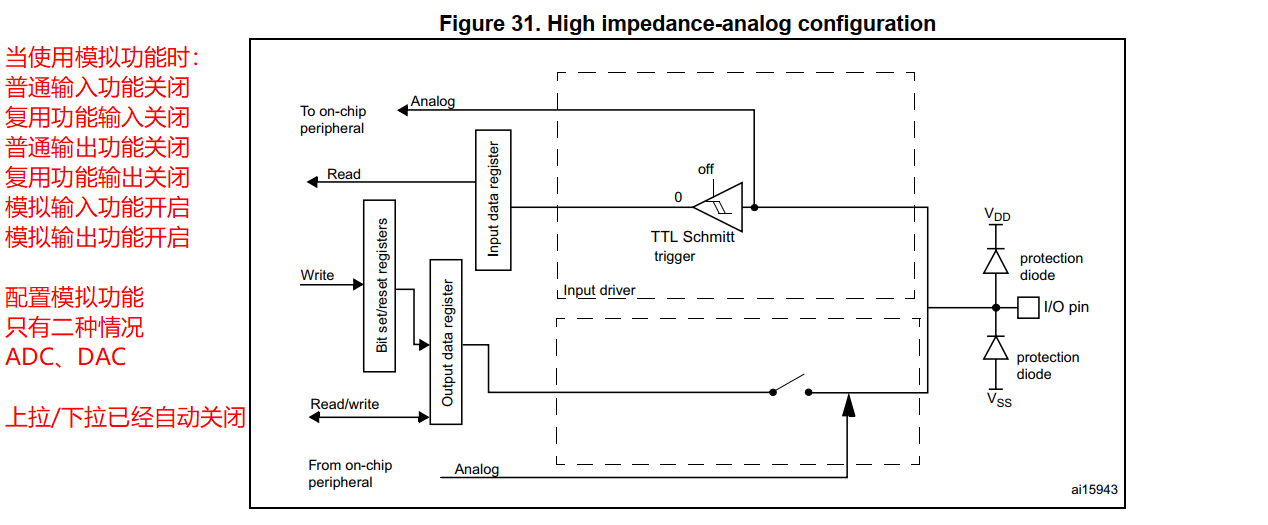
1. 配置IO口模式，使得IO口为复用功能
2. 配置IO口类型，选择推挽/开漏
3. 配置IO口上拉/下拉，根据实际情况选择
4. 配置IO口速度，根据实际情况选择
5. 配置IO口复用类型，根据使用的片上外设资源以及AFR寄存器中的连接关系选择

复用功能输入配置：

1. 配置IO口模式，使得IO口为复用功能
2. 配置IO口上拉/下拉，根据实际情况选择
3. 配置IO口复用类型，根据使用的片上外设资源以及AFR寄存器的连接关系选择



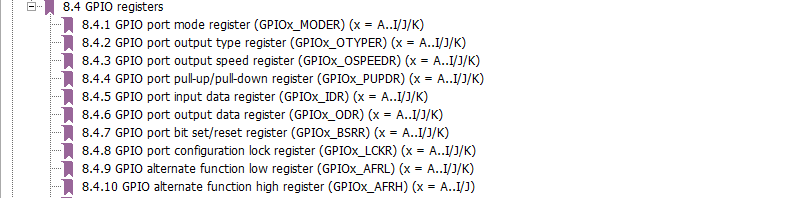
### 芯片IO口模拟功能分析

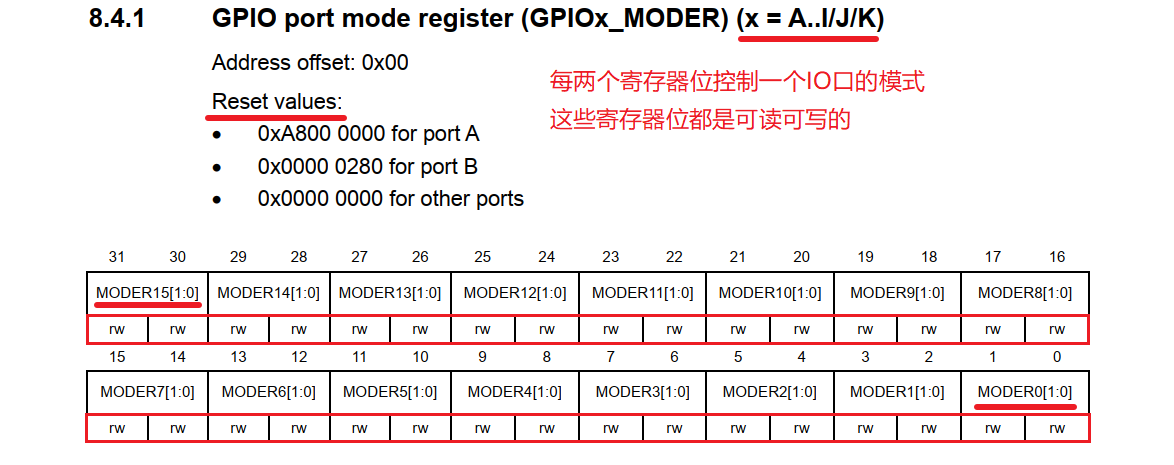


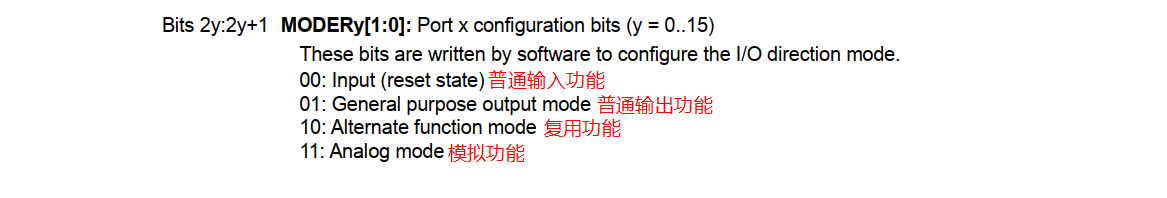
想要配置模拟功能，需要进行如下操作：

1. 配置IO口模式，使得IO口模式为模拟功能

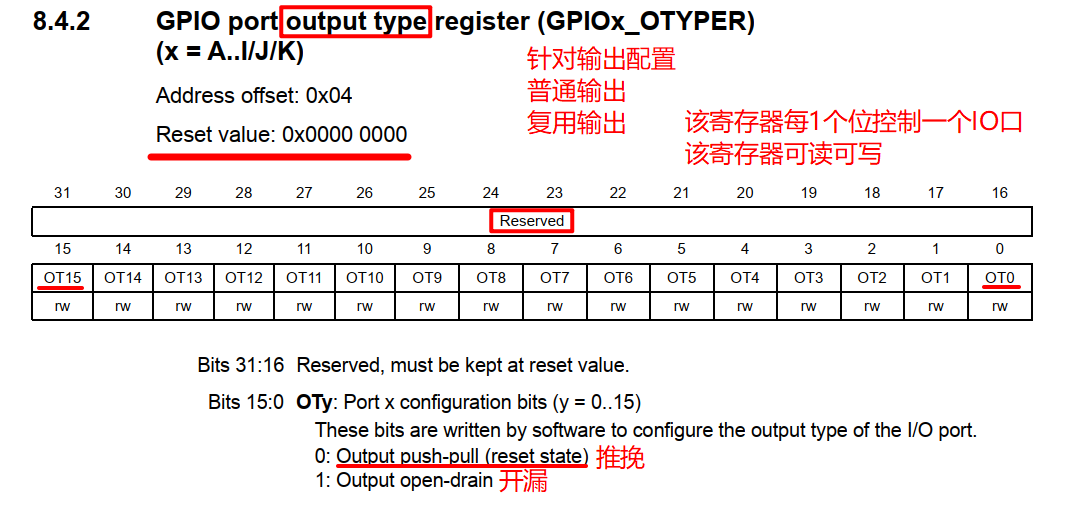
## STM32F407VG-GPIO-相关寄存器

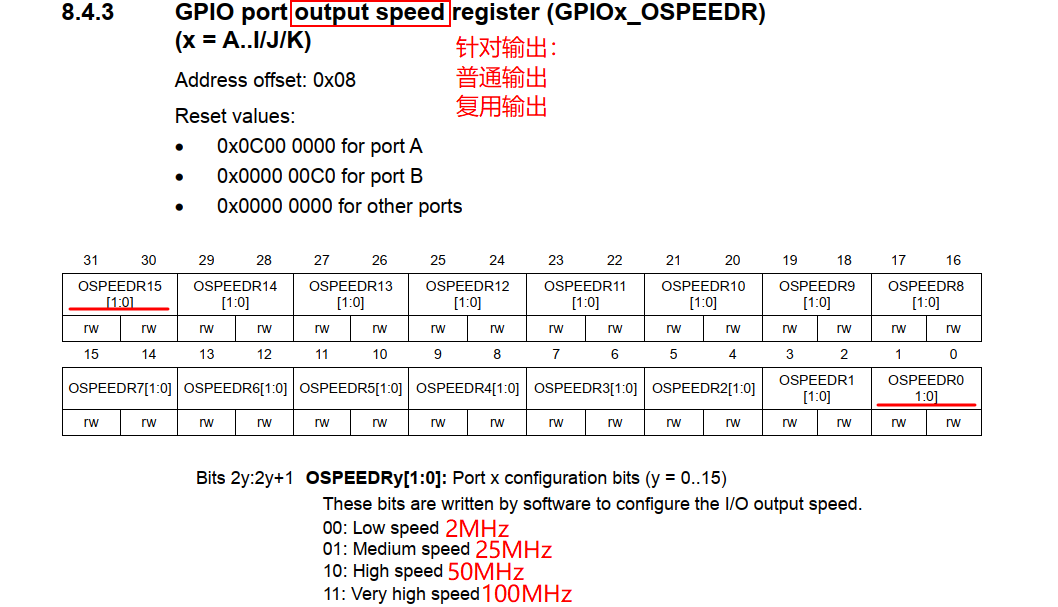


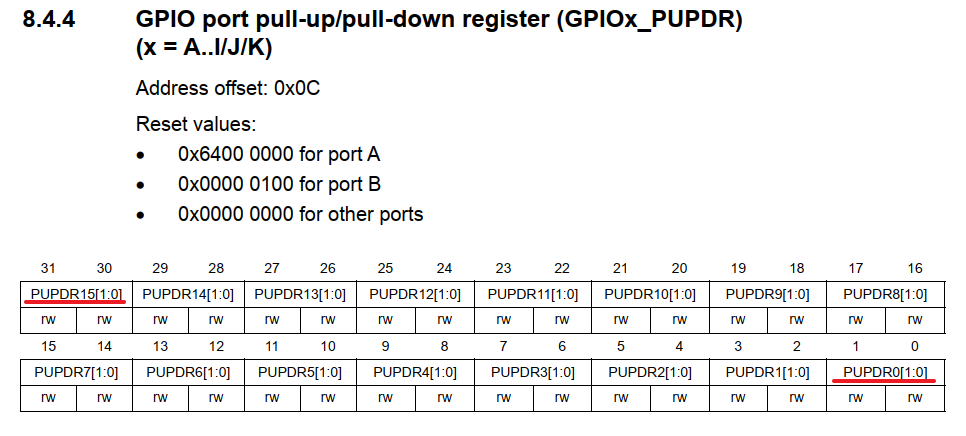


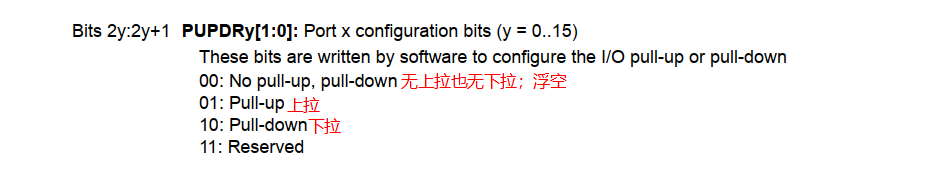


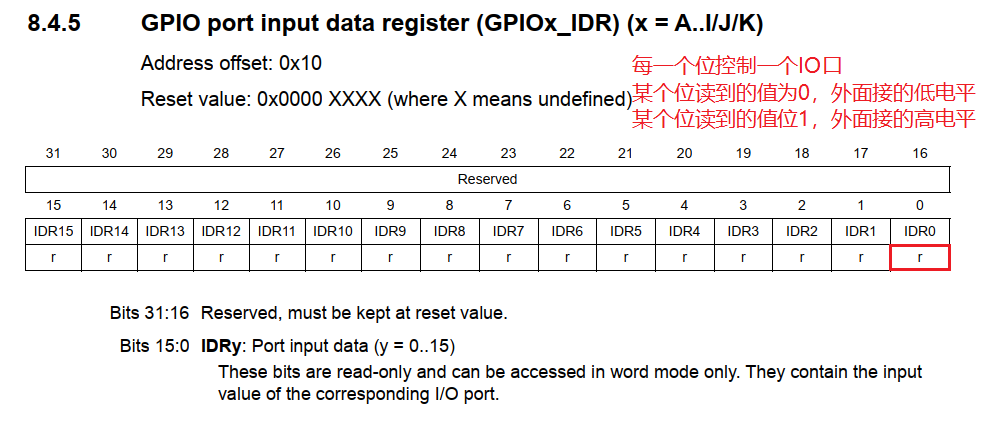
|  |
| --- |
| GPIOX\_MODER &= ~(0x3 << 2 \* n);  GPIOX\_MODER |= (0x2 << 2 \* n); |







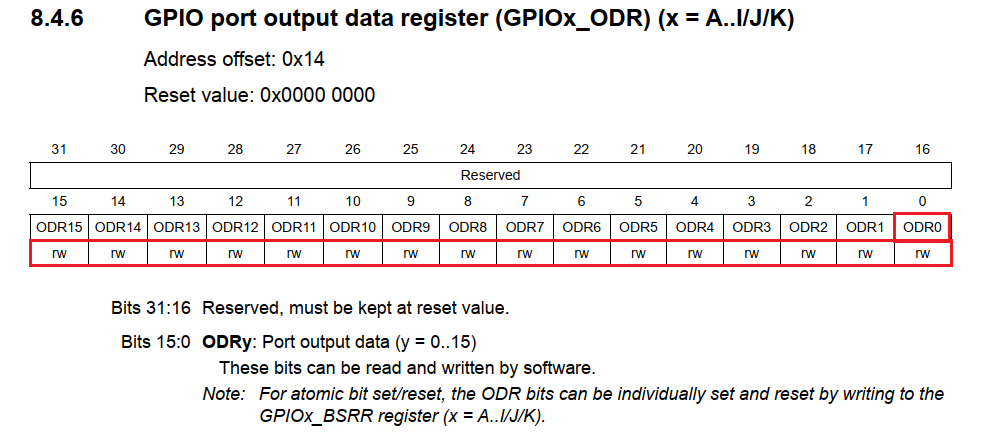


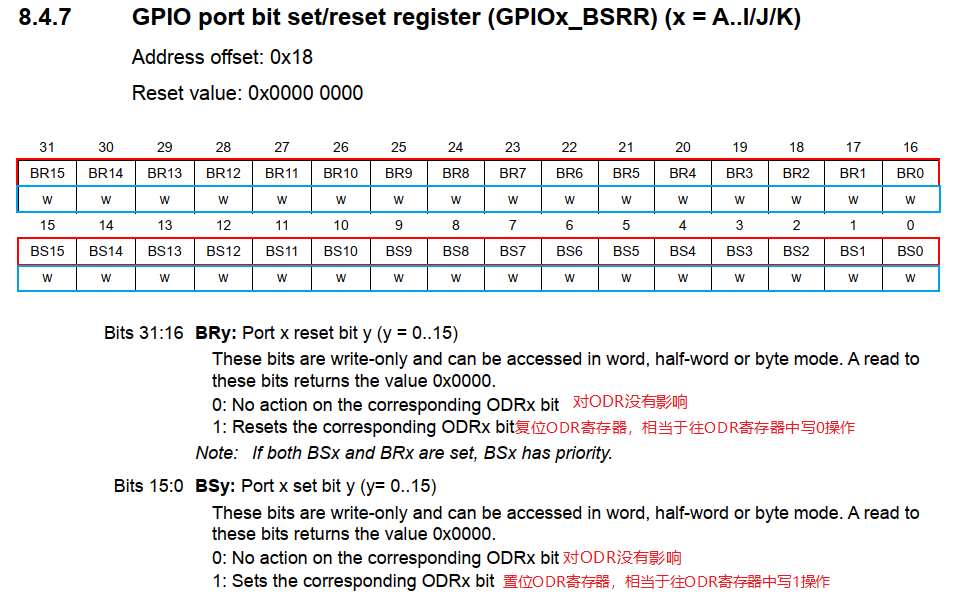


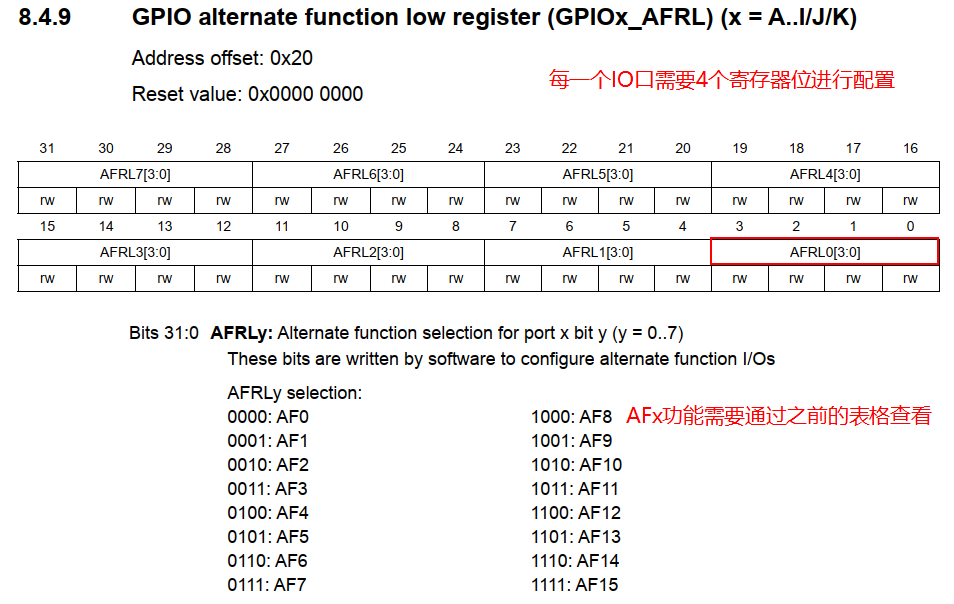
输入数据寄存器，获取寄存器的值读取整个一组IO口的所有电平状态。

如果只需要判断某一个位状态，如何操作？

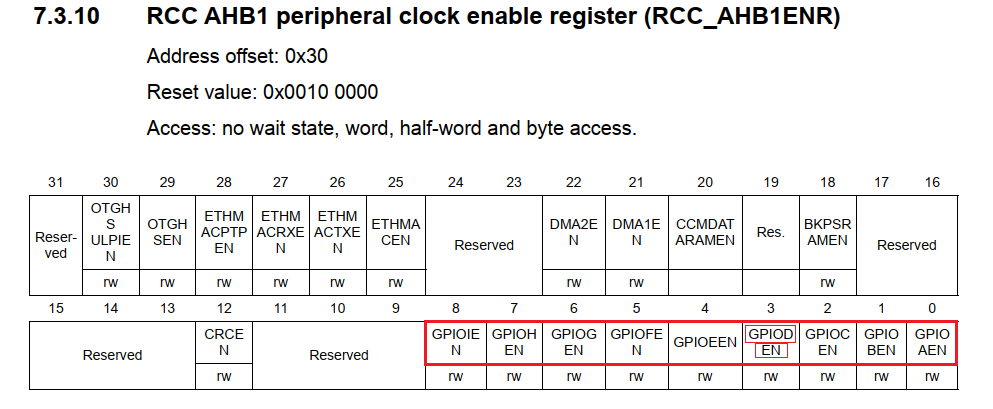
|  |
| --- |
| 如果不需要二值性，操作方法如下：  //如果某一个位为高电平，则条件成立；如果某一个位为低电平，则条件不成立  if(GPIOX\_IDR & (0x1 << n))  {  //高电平成立需要执行的程序  }  else  {  //低电平成立需要执行的程序  }  如果需要二值性，操作如下：  #define XXXXXX (!!(GPIOX\_IDR & (0x1 << n)))  //如果某一个位为低电平，则条件成立，如果某一个位为高电平，则条件不成立  if(XXXXXX == 0)  {  //低电平成立需要执行的程序  }  else  {  //高电平成立需要执行的程序  } |

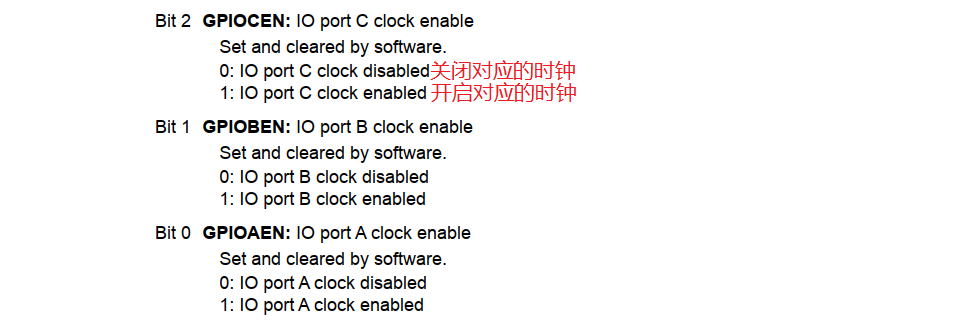






由于STM32处理器属于低功耗类型的处理器，每个外设功能都进行功能保护，如果外设功能不开启对应的时钟，该外设不会进行工作。需要GPIO外设工作需要开启对应的时钟。GPIO外设时钟开启，GPIO外设属于AHB1总线上。从AHB1相关寄存器中查找。





## STM32F407VG-GPIO-实验

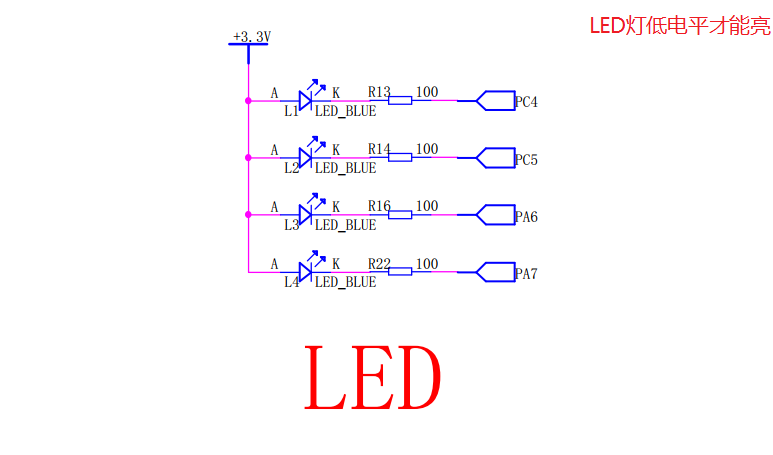
### 点亮LED灯

第一步：初始化配置IO口

1. 开启GPIO口时钟
2. 按照普通输出配置

第二步：点亮LED灯

1. 根据LED灯需要什么电平进行设置



### 按键识别实验

第一步：初始化按键接口

1. 开启接口时钟
2. 配置模式为普通输入
3. 配置上拉/下拉为无上拉无下拉

第二步：识别按键

1. 读取输入数据寄存器的值
2. 延时消抖
3. 再次读取输入数据寄存器的值