# STM32 第三天

## 库函数与寄存器开发区别

### 1.1 寄存器

**寄存器开发优点**

直接操作寄存器，运行效率高。

**寄存器开发缺点**

1. 开发难度大，开发周期长
2. 代码可阅读性差，可移植差
3. 后期维护难度高

### 1.2库函数

**库函数开发优点**

1. 开发难度较小，开发周期短
2. 代码可阅读性强，可移植高
3. 后期维护难度低

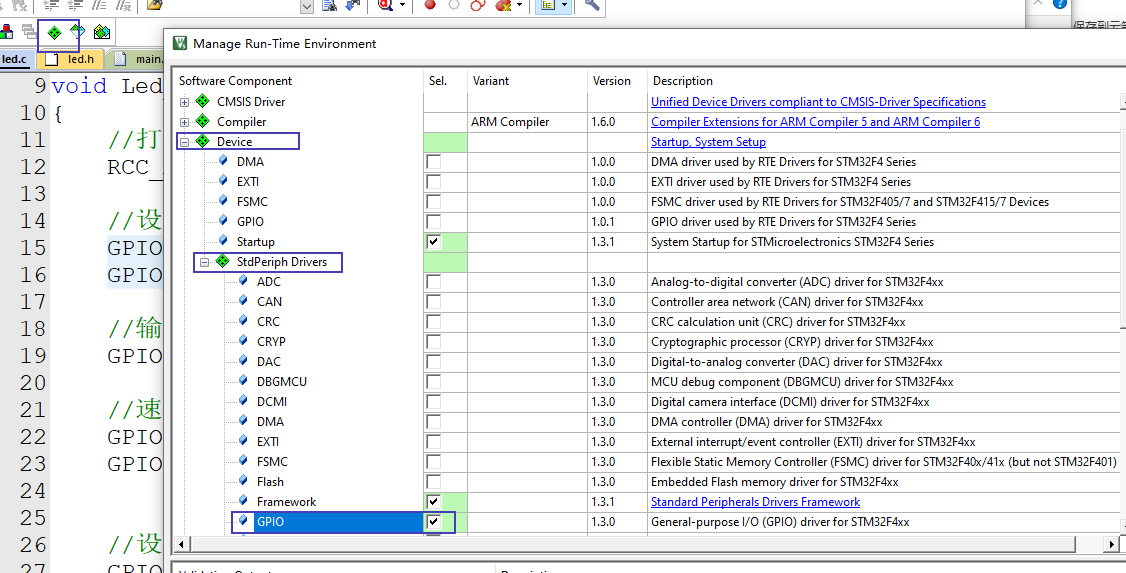
**库函数开发缺点**

相对于寄存器开发，运行效率略低

**注意：库函数其实是ST公司对寄存器的进一步封装。**

## 库函数开发LED

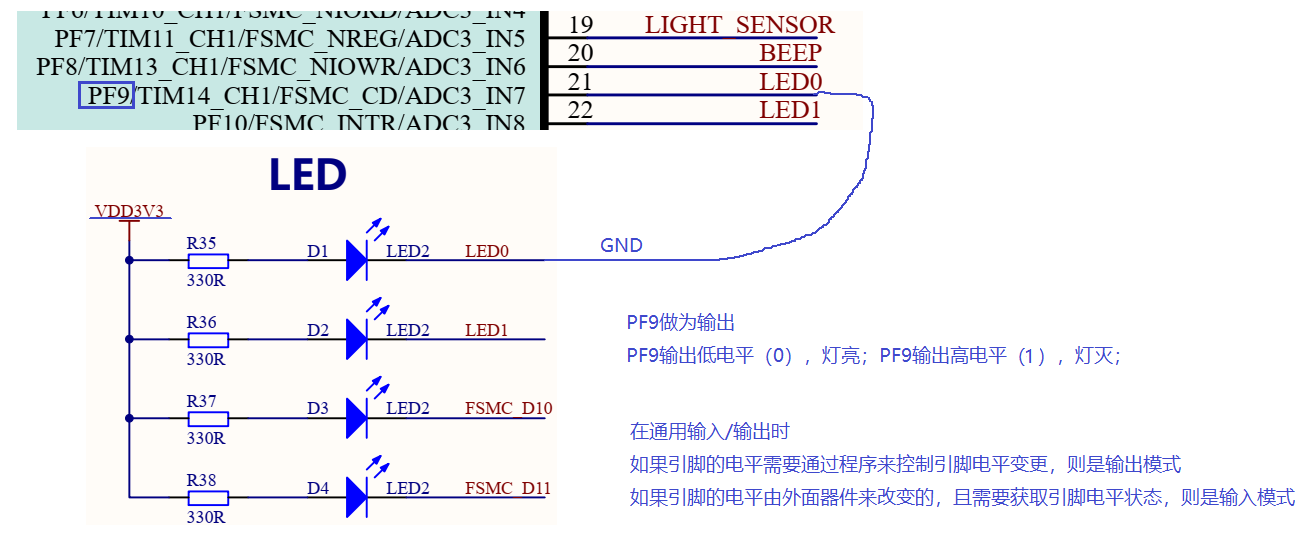
库函数开发LED要添加的库函数文件：stm32f4xx\_gpio.c



1. **理解led灯原理图**

LED0连接在PF9

PF9输出低电平（0），灯亮；PF9输出高电平（0），灯灭；



1. **打开GPIOF组时钟**

**在STM32芯片中，所有的外设时钟是不打开，为了降低功耗**

//打开GPIOF组时钟

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOF, ENABLE);

1. **设置PF9灯为输出模式 输出推挽 上拉 速度（50MHZ）**

GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_9; //引脚

GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_OUT; //输出模式

GPIO\_InitStruct.GPIO\_OType = GPIO\_OType\_PP; //输出推挽

GPIO\_InitStruct.GPIO\_Speed = GPIO\_Speed\_25MHz; //速度25MHZ

GPIO\_InitStruct.GPIO\_PuPd = GPIO\_PuPd\_UP; //上拉

GPIO\_Init(GPIOF, &GPIO\_InitStruct);

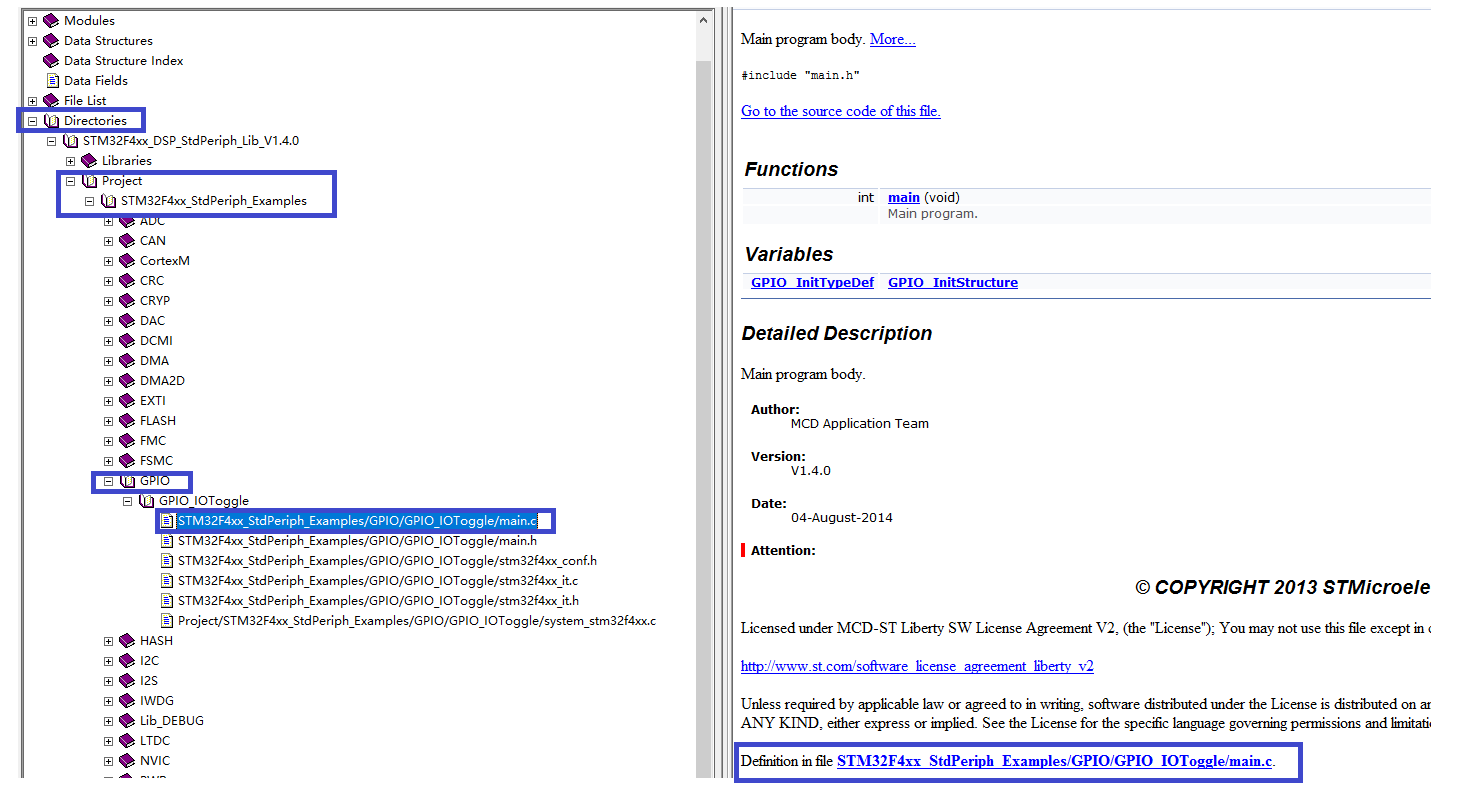
**4、通过下面控制LED灯亮与灭**

GPIO\_SetBits() 引脚置1

GPIO\_ResetBits() 引脚置0

**官方参考案例**

路径：D:\GZFX2302\STM32\STM32上课资料\STM32F4xx固件库\stm32f4\_dsp\_stdperiph\_lib\STM32F4xx\_DSP\_StdPeriph\_Lib\_V1.4.0\stm32f4xx\_dsp\_stdperiph\_lib\_um.chm



## 按键库函数开发

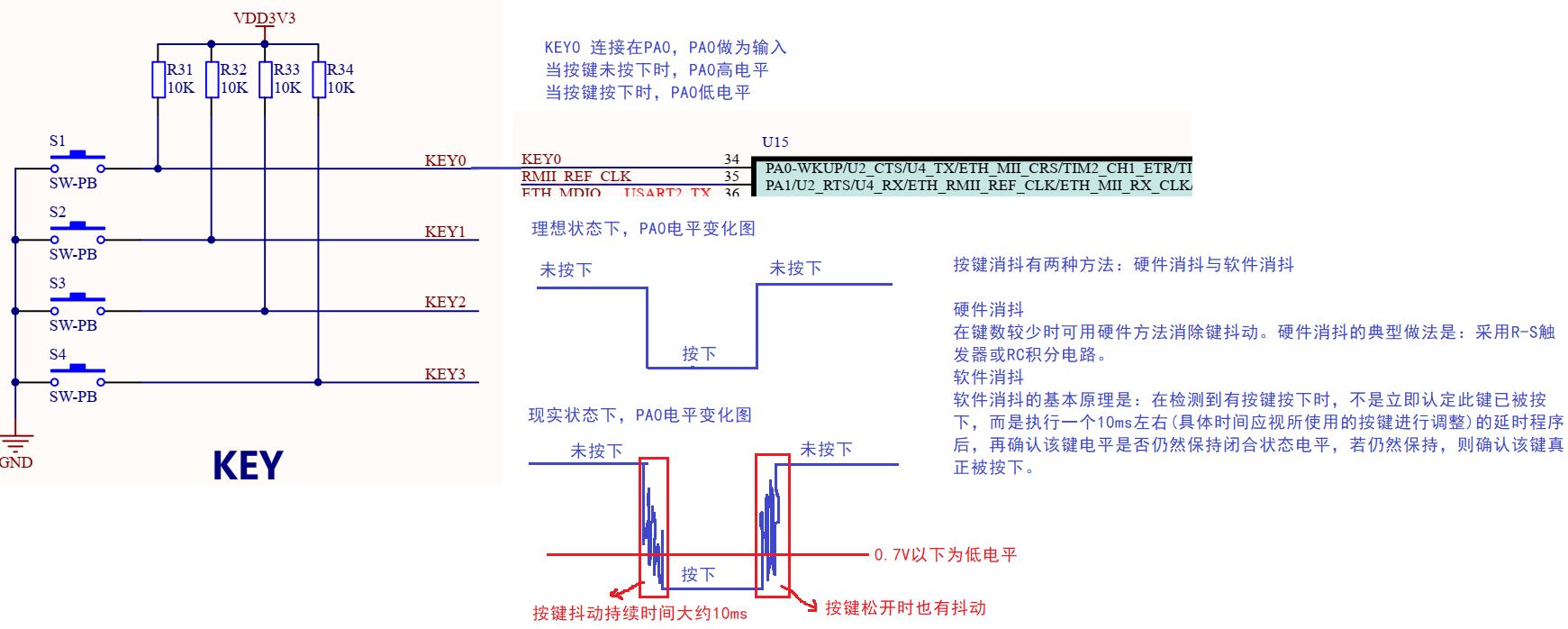
库函数开发按键要添加的库函数文件：stm32f4xx\_gpio.c

**1、理解按键控制原理**

KEY0连接在PA0

按键未按下，PA0为高电平(1)

按键 按下，PA0为低电平(0)



1. **打开GPIOA组时钟**

//打开GPIOA组时钟

RCC\_AHB1PeriphClockCmd(RCC\_AHB1Periph\_GPIOA, ENABLE);

1. **设置引脚为输入，上拉。**

GPIO\_InitTypeDef GPIO\_InitStruct;

GPIO\_InitStruct.GPIO\_Pin = GPIO\_Pin\_0; //引脚0

GPIO\_InitStruct.GPIO\_Mode = GPIO\_Mode\_IN; //输入模式

GPIO\_InitStruct.GPIO\_PuPd = GPIO\_PuPd\_UP; //上拉

GPIO\_Init(GPIOA, &GPIO\_InitStruct);

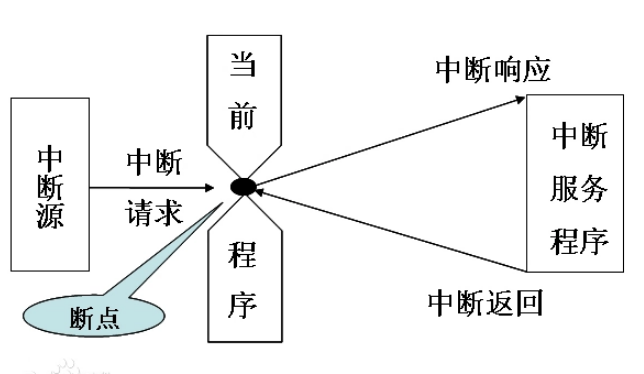
**4、通过下面函数获取引脚电平**

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

## 中断

**中断概念**

中断是指计算机运行过程中，出现某些意外情况需主机干预时，机器能自动停止正在运行的程序并转入处理新情况的程序，处理完毕后又返回原被暂停的程序继续运行（面试题）。

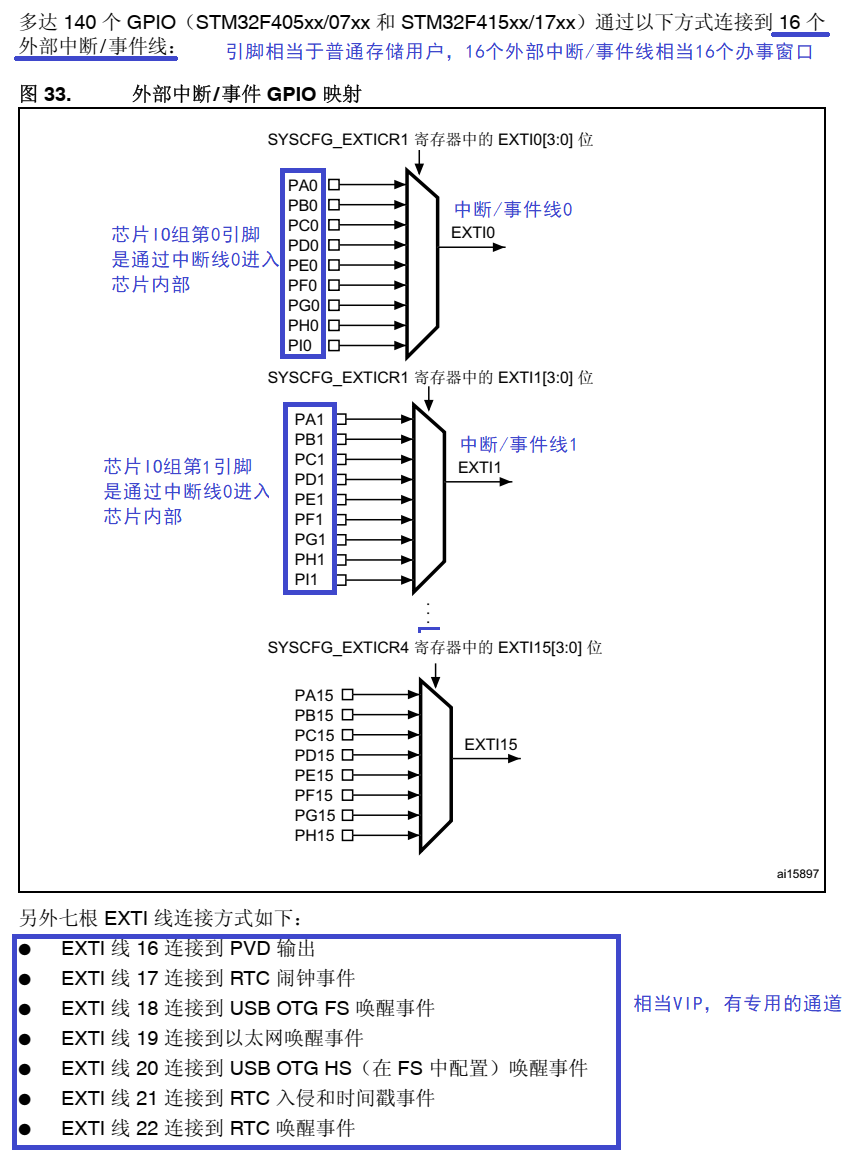


STM32外部中部需要注重知识点EXTI与NVIC,STM32所有引脚均可以做为外部中断。

### **4.1、外部中断**/**事件控制器** (EXTI)

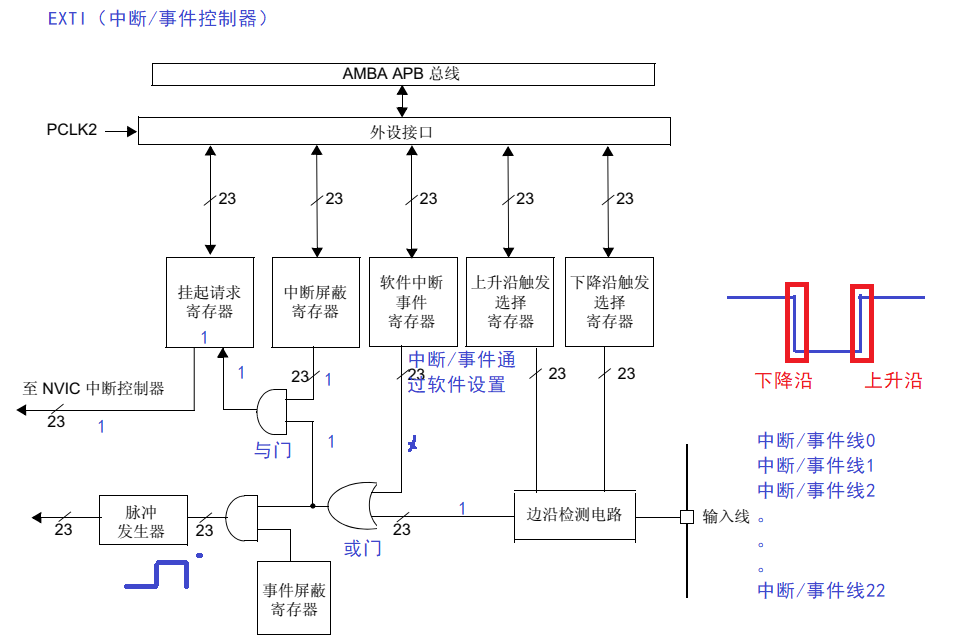
EXTI 控制器的主要特性如下：  
● 每个中断/事件线上都具有独立的触发和屏蔽  
● 每个中断线都具有专用的状态位  
● 支持多达 23 个软件事件/中断请求  
● 检测脉冲宽度低于 APB2 时钟宽度的外部信号。有关此参数的详细信息，请参见  
STM32F4xx 数据手册的电气特性部分。

**引脚中断线**



**上升沿：数字电路中，数字电平从低电平（数字“0”）变为高电平（数字“1”）的那一瞬间（时刻）叫作上升沿。**

**下降沿：数字电路中，数字电平从高电平（数字“1”）变为低电平（数字“0”）的那一瞬间叫作下降沿。**



### **4.2 NVIC**

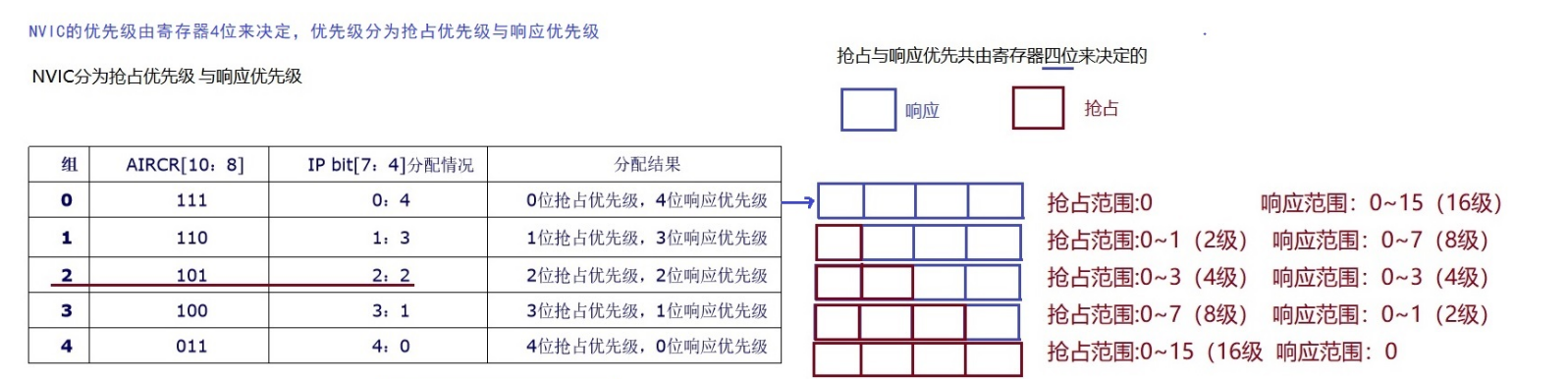
嵌套向量中断控制器 (NVIC)

**NVIC** 特性  
嵌套向量中断控制器 NVIC 包含以下特性：  
● STM32F405xx/07xx 和 STM32F415xx/17xx 具有 82 个可屏蔽中断通道， STM32F42xxx  
和 STM32F43xxx 具有多达 86 个可屏蔽中断通道（不包括 Cortex™-M4F 的 16 根中  
断线）  
● 16 个可编程优先级（使用了 4 位中断优先级）  
● 低延迟异常和中断处理  
● 电源管理控制  
● 系统控制寄存器的实现  
嵌套向量中断控制器 (NVIC) 和处理器内核接口紧密配合，可以实现低延迟的中断处理和晚  
到中断的高效处理。

NVIC优先级分为抢占优先级与响应优先级。

**数字越小，优先级别越高**

**优先级分组**



抢占优先级 & 响应优先级区别： 抢占优先级 > 响应优先级

高优先级的抢占优先级是可以打断正在进行的低抢占优先级中断的。

抢占优先级相同的中断，高响应优先级不可以打断低响应优先级的中断。

抢占优先级相同的中断，当两个中断同时发生的情况下，哪个响应优先级高，哪个先执行。

如果两个中断的抢占优先级和响应优先级都是一样的话，则看哪个中断先发生就先执行；

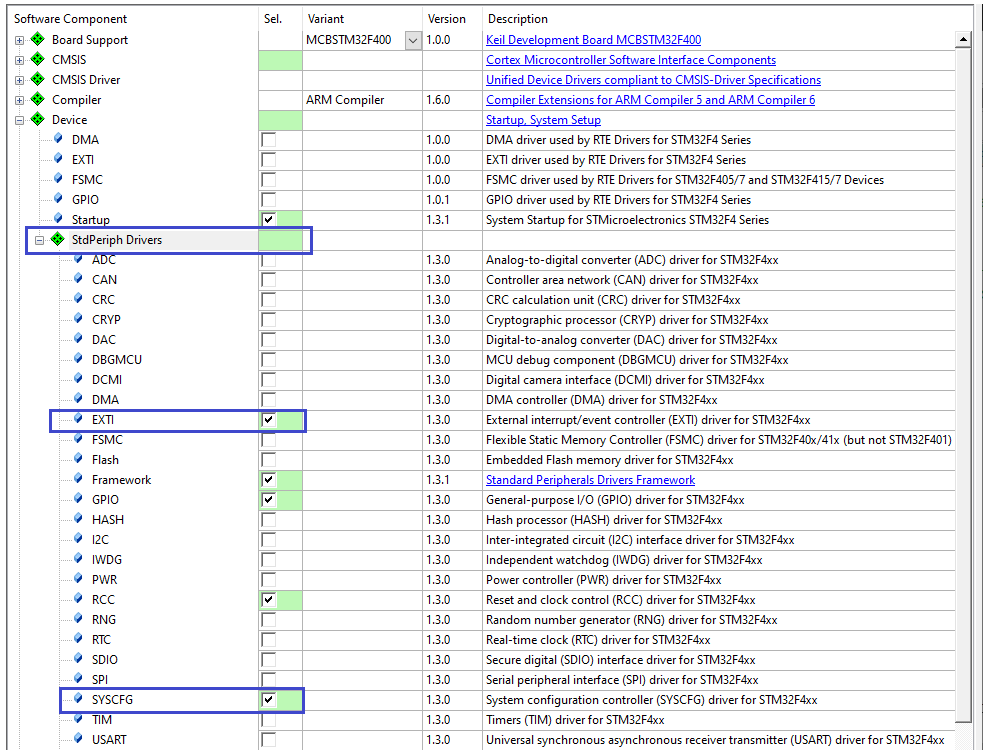
例：

假定设置中断优先级组为2，然后设置中断3(RTC中断)的抢占优先级为2，响应优先级为1。 中断6（外部中断0）的抢占优先级为3，响应优先级为0。中断7（外部中断1）的抢占优先级为2，响应优先级为0。

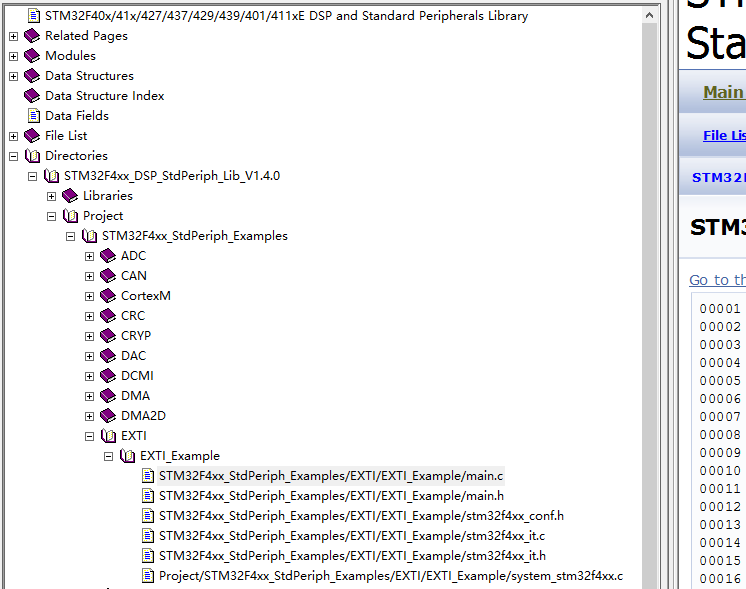
中断7>中断3>中断6。

### 4.3外部中断配置流程

外部中断配置流程要添加的库函数：stm32f4xx\_exti.c和stm32f4xx\_syscfg.c

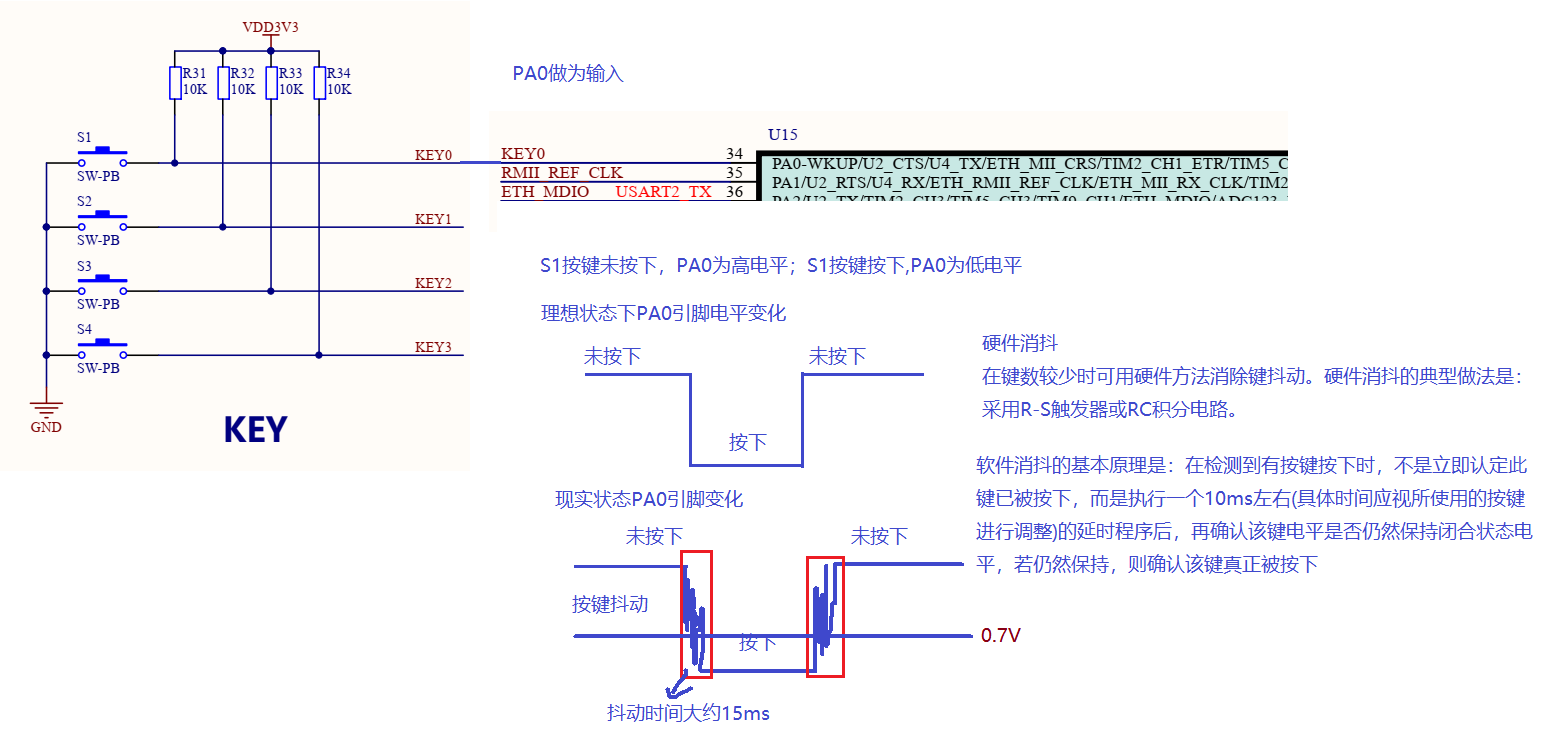


参考库函数案例



1. **理解按键原理**

**KEY0 连接PA0，选择下降沿触发**



1. **设置NVIC分组**

void NVIC\_PriorityGroupConfig(uint32\_t NVIC\_PriorityGroup);

**3、使能SYSCFG及GPIOA时钟：**

*RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_SYSCFG, ENABLE);*

**4、 初始化IO口为输入。**

*GPIO\_Init();*

**5、设置IO口与中断线的映射关系。**

*void SYSCFG\_EXTILineConfig();*

**6、初始化线上中断，设置触发条件等。**

*EXTI\_Init();*

**7、配置中断分组（NVIC），并使能中断。**

*NVIC\_Init();*

**8、 编写中断服务函数。**

*EXTIx\_IRQHandler();*

**9、清除中断标志位**

*EXTI\_ClearITPendingBit();*

## 五、函数说明

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

函数功能：使能AHB1外设时钟

返回值：无

uint32\_t RCC\_AHB1Periph：哪个外设

FunctionalState NewState：ENABLE or DISABLE

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

函数功能：GPIO初始化

返回值：无

GPIO\_TypeDef\* GPIOx：哪个GPIO组

GPIO\_InitTypeDef\* GPIO\_InitStruct：GPIO结构

typedef struct

{

uint32\_t GPIO\_Pin; //哪个引脚

GPIOMode\_TypeDef GPIO\_Mode; //模式

GPIOSpeed\_TypeDef GPIO\_Speed; //速度

GPIOOType\_TypeDef GPIO\_OType; //输出类型

GPIOPuPd\_TypeDef GPIO\_PuPd; //上下拉

}GPIO\_InitTypeDef;

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

函数说明：引脚置位

函数返回值：无

GPIO\_TypeDef\* GPIOx：哪个组IO

uint16\_t GPIO\_Pin:哪个引脚

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

函数说明：引脚复位

函数返回值：无

GPIO\_TypeDef\* GPIOx：哪个组IO

uint16\_t GPIO\_Pin:哪个引脚

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

函数说明：读某个引脚输入状态

函数返回值：

Bit\_RESET = 0 （低电平）

Bit\_SET = 1 （高电平）

GPIO\_TypeDef\* GPIOx：哪个组IO

uint16\_t GPIO\_Pin:哪个引脚

**void SYSCFG\_EXTILineConfig(uint8\_t EXTI\_PortSourceGPIOx, uint8\_t EXTI\_PinSourcex)**

函数说明：设置IO口与中断线的映射

函数返回值：无

uint8\_t EXTI\_PortSourceGPIOx：哪个组IO

uint8\_t EXTI\_PinSourcex:哪个引脚

**void EXTI\_Init(EXTI\_InitTypeDef\* EXTI\_InitStruct)**

函数说明：外部中断控制器初始化

返回值：无

typedef struct

{

uint32\_t EXTI\_Line; //中断线

EXTIMode\_TypeDef EXTI\_Mode; //模式

EXTITrigger\_TypeDef EXTI\_Trigger; //触发条件

FunctionalState EXTI\_LineCmd; //中断线命令

}EXTI\_InitTypeDef;

void NVIC\_Init(NVIC\_InitTypeDef\* NVIC\_InitStruct)

函数说明：NVIC初始化

返回值：无

NVIC\_InitTypeDef\* NVIC\_InitStruct：NVIC结构体

typedef struct

{

uint8\_t NVIC\_IRQChannel; //中断通道，可在stm32f4xx.h文件当中查找

uint8\_t NVIC\_IRQChannelPreemptionPriority; //抢占优先级

uint8\_t NVIC\_IRQChannelSubPriority; //响应优先级

FunctionalState NVIC\_IRQChannelCmd; //中断通道使能

} NVIC\_InitTypeDef;

ITStatus EXTI\_GetITStatus(uint32\_t EXTI\_Line)；

函数说明：判断中断线标志位

返回值：SET或RESET

uint32\_t EXTI\_Line：哪个中断线

void EXTI\_ClearITPendingBit(uint32\_t EXTI\_Line)；

函数说明：清空中断线标志位

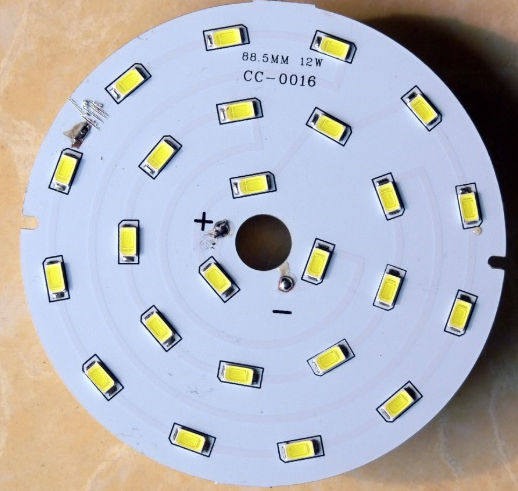
返回值：无

uint32\_t EXTI\_Line：哪个中断线

## 六、应用产品

### LED

家用LED



指示灯



### 按键

汽车按键 电梯按键

