ITS_第一次电控培训

Author：zyt

认识单片机

在一片集成电路芯片上集成中央处理器、存储器、I/O接口电路，即一块芯片就构成了一个完整的计算机系统，称为`单片微型计算机`，因此就简称为单片机（Single Chip Microcomputer）。简单地理解，单片机就是一个微型计算机。

而引出芯片所提供的所有接口并将外设和元器件集成在PCB板上，就得到了一块最小系统板。

STM32就是一款应用十分广泛的单片机，它是ST公司开发的基于Arm® Cortex®-M处理器的32位微控制器`（内核/芯片/开发板）`

STM32有许多系列的产品，分别适用于不同的场景和需求，我们目前最常用到的是f1和f4两个系列，其他系列感兴趣的可以自行了解　　官网：st.com/content/st_com/en.html

开发方式：

- 直接读写寄存器：非常底层，效率高
- 标准库：对读写寄存器的操作进行一定程度的封装（目前已停止更新）
- HAL库：全称`Hardware Abstraction Layer`（硬件抽象层），是ST公司目前主力推的开发方式。

`tips:在写代码的时候，我们到底是在控制什么呢`

在正式开始讲GPIO之前，我们可以先来看一看这块板子上都有些什么，我们应该用怎样的顺序去认识这块单片机：

芯片：stm32f103c8t6

外设：GPIO，板载LED，晶振，UART，IIC，SPI，CAN.....

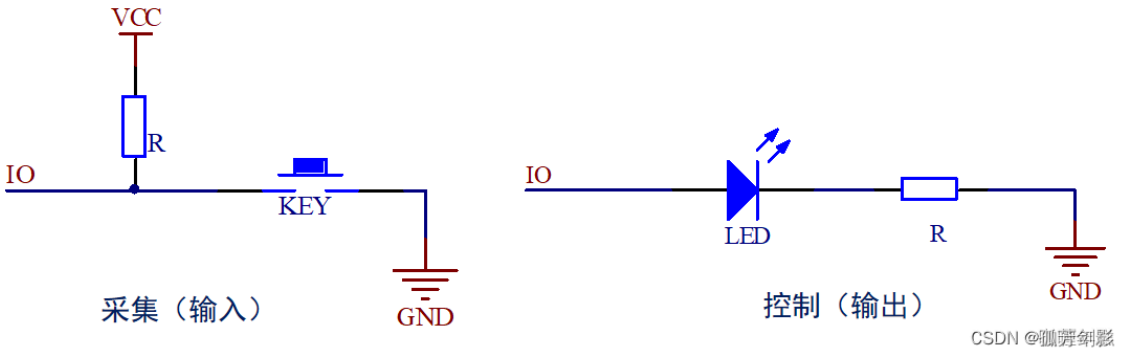
GPIO

简介

GPIO（General Purpose Input Output）即通用I/O端口，是一种通用的数字输入/输出端口。在嵌入式系统中，GPIO被设计为灵活的引脚，可以被配置为输入或输出，以满足不同的应用需求

General Purpose Input Output，即通用输入输出端口，简称GPIO

作用：负责采集外部器件的信息或者控制外部器件工作，即输入输出



（网图勿喷）

stm32内部的GPIO框图

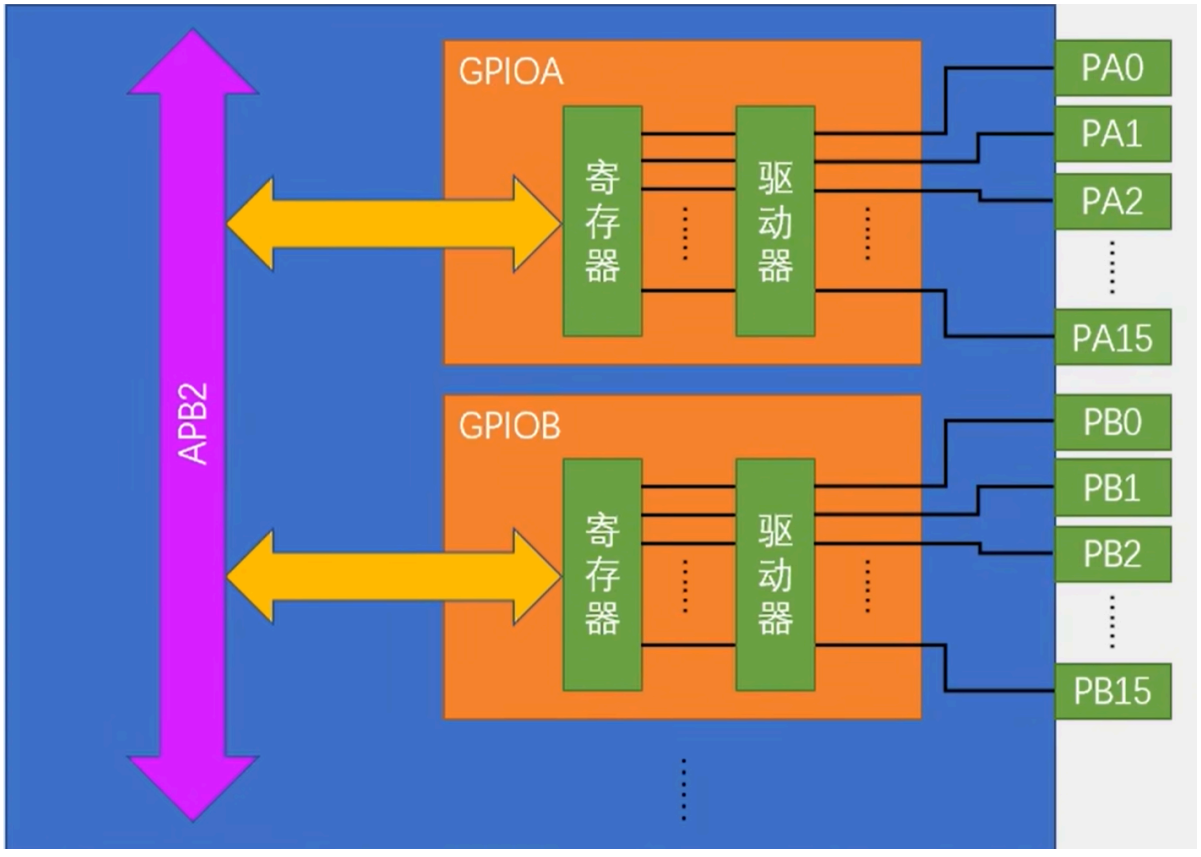
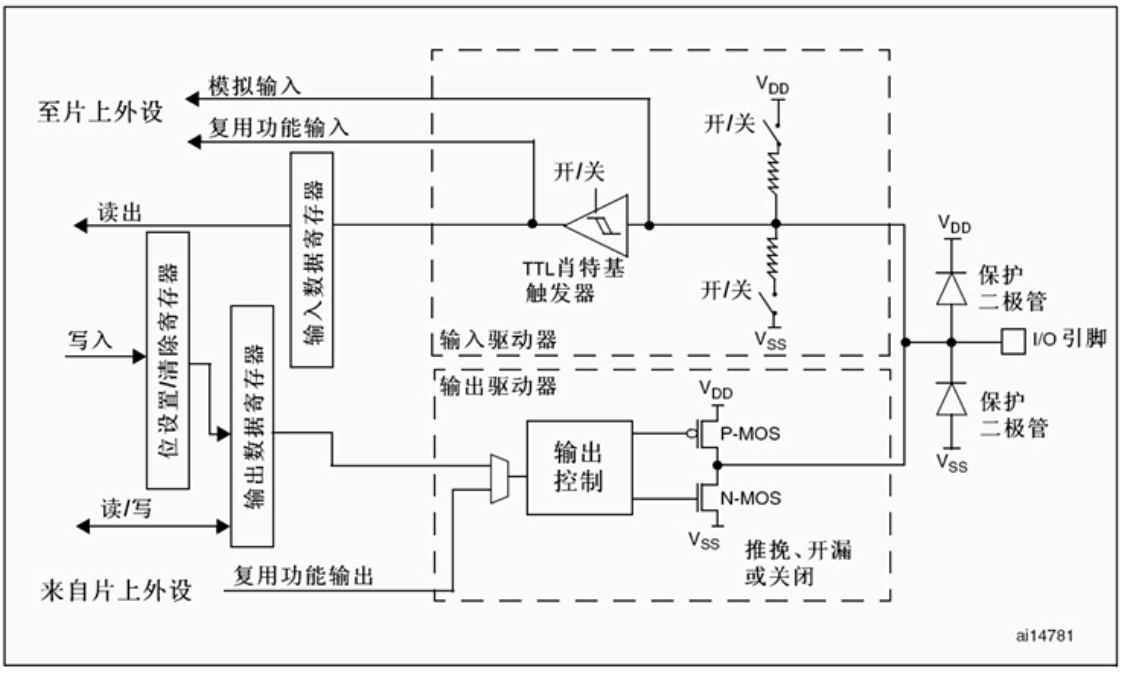


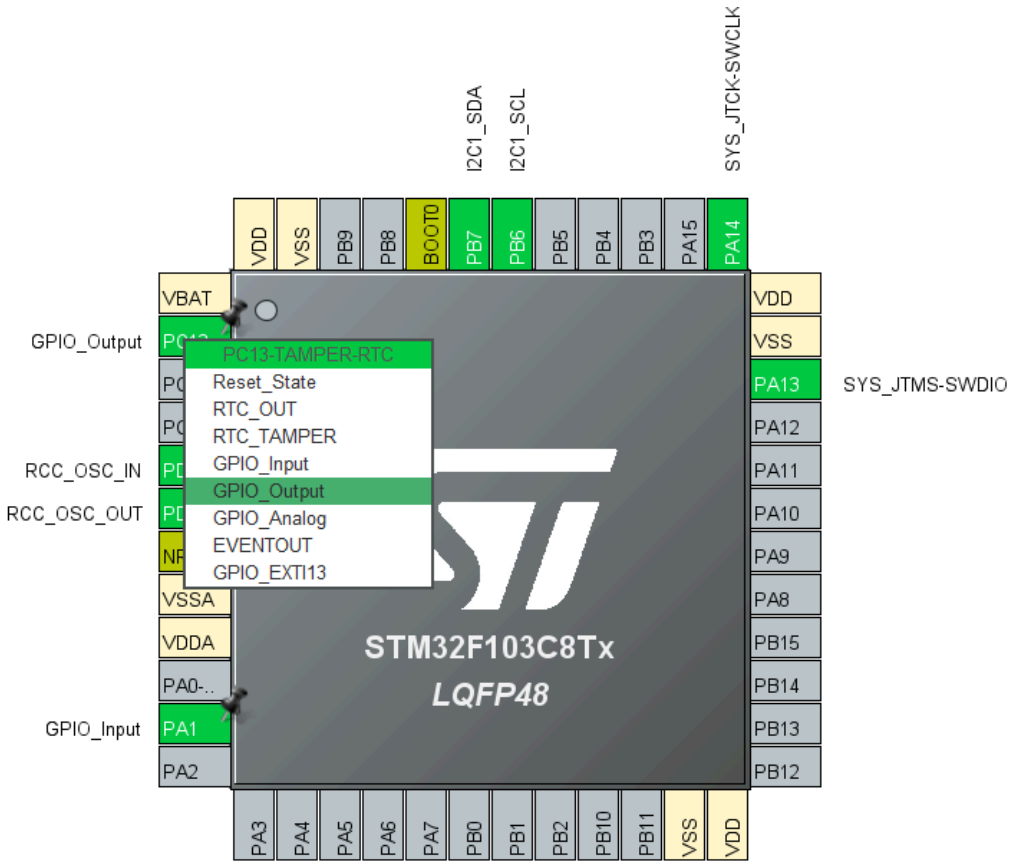
图13　I/O端口位的基本结构

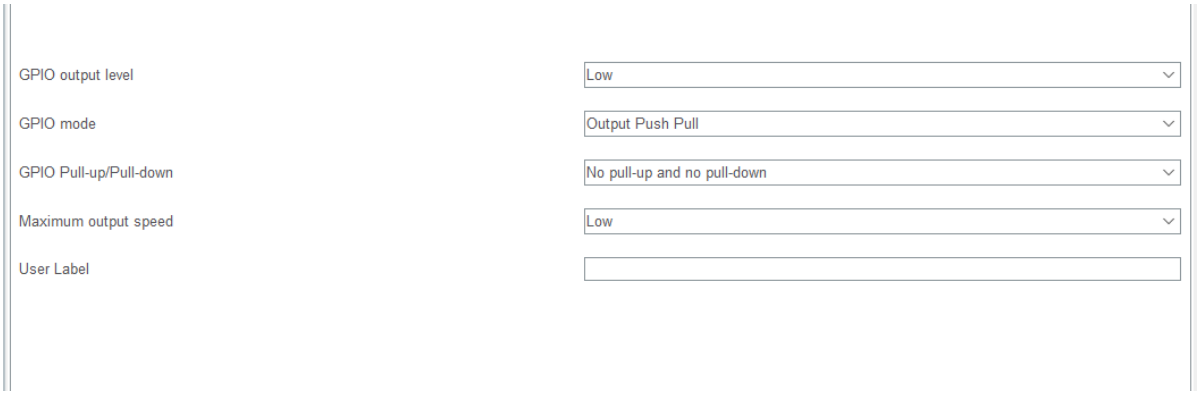


(此处肖特基触发器其实应该是施密特触发器，用于整流 (类似于一个滞回比较器))

几种常用的工作模式

cube中的配置卡:

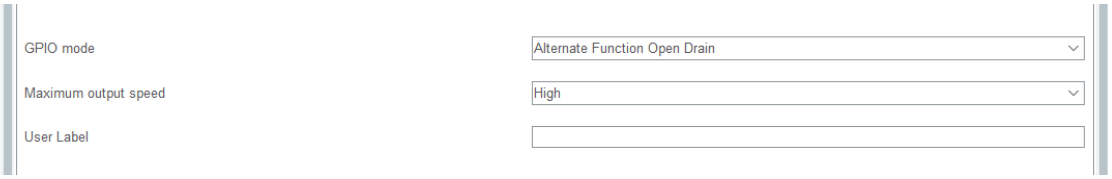




输出

- `Output Push Pull`：推挽输出，芯片绝对控制芯片输出电平高低（最常用！）
- `Output Open Drain`：开漏输出，只有下拉的NMOS起作用，通常用于某些通信中（不过使用板上的硬件时也不需要自己配置这个玩意）

  例如iic



输入（读取外部电平）

- `No pull-up and no pull-down`：默认不上拉也不下拉，会让引脚处于不确定的状态，一般不选
- `Pull-up`：默认弱上拉
- `Pull-down`：默认弱下拉

常用函数

```
void HAL_GPIO_WritePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin, GPIO_PinState
PinState)
/**GPIO写函数
  *参数1：GPIO_TypeDef *GPIOx，GPIO的端口，如GPIOA
  *参数2：GPIO_Pin，GPIO的引脚号，如GPIO_PIN_13
  *参数3：写入电平 如GPIO_PIN_RESET
  */
```

```
GPIO_PinState HAL_GPIO_ReadPin(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
/**GPIO读函数
  *参数1：GPIO端口
  *参数2：GPIO引脚号
  *返回值：引脚电平
  */
```

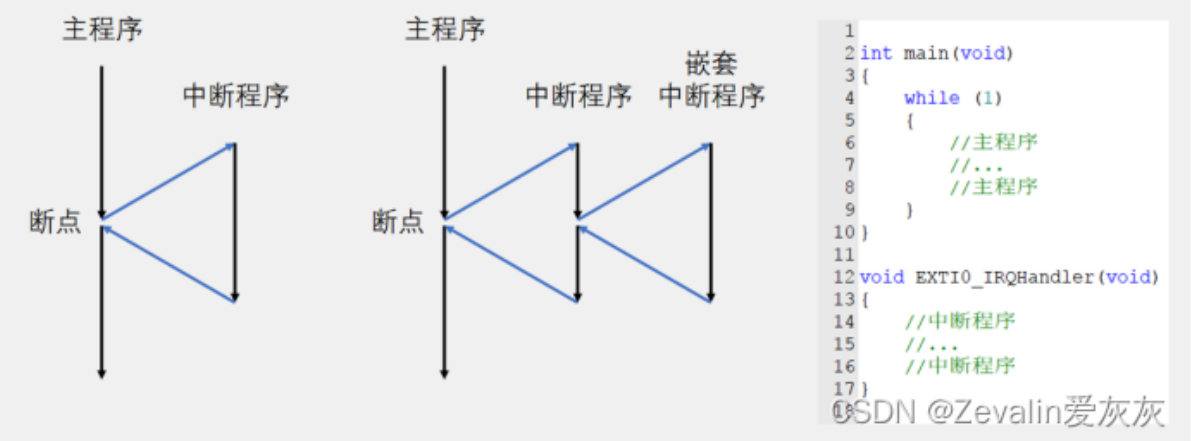
```
void HAL_GPIO_TogglePin(GPIO_TypeDef *GPIOx, uint16_t GPIO_Pin)
/**GPIO电平翻转函数
  *参数1：GPIO端口
  *参数2：GPIO引脚号
  */
```

GPIO的外部中断

什么是中断

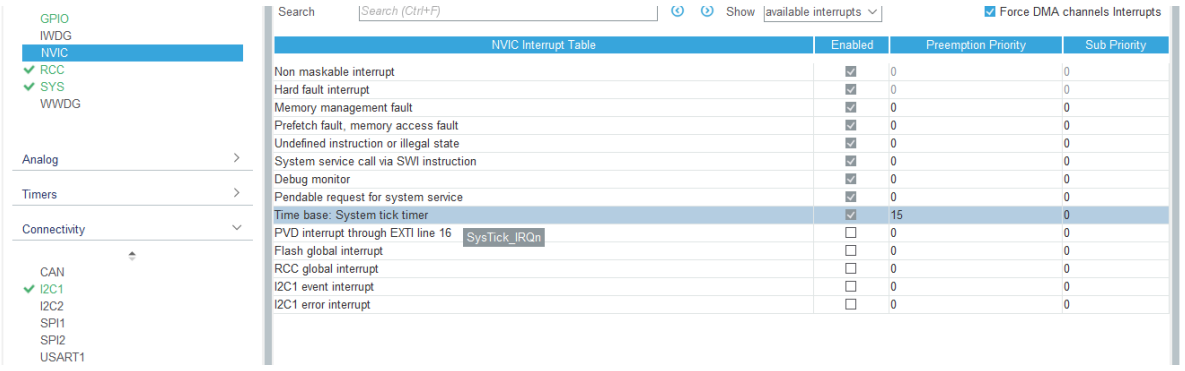
在嵌入式系统中，特别是在实时系统中，对一些事件的即时响应至关重要。例如：传感器数据的读取、定时器溢出、外部输入信号等都是需要及时处理的事件。

而中断就是在程序执行期间中断当前任务，转而去处理被触发的任务



中断机制允许系统对实时事件做出及时响应，而不必用循环去等待特定事件的发生，从而可以确保系统能够在这些事件发生时立即作出响应，而不会因为等待而造成延迟或丢失数据。

中断系统由中断向量表、中断优先级机制和中断处理程序组成。中断向量表存储中断服务程序的地址，中断优先级机制决定中断处理的顺序，其中NVIC可以在cube中配置（序号越小优先级越高）



Preemption Priority(抢占优先级)：顾名思义能够抢先执行任务，即打断当前的主程序或者中断程序的运行，前去完成抢断中断。也称中断嵌套。

SubPriority(子优先级)：也是从优先级或者副优先级，在抢占优先级相同的情况下不能发生中断嵌套，高级的子优先级先执行。如果低级的子优先级在执行，需要等待完成才能执行高级的子优先级，即不发生抢断。

而我们学习中断，重点关注两个方面：一是中断信号的来源（什么触发了中断），二是中断回调函数的内容（中断完了之后要干什么）。

STM32中有68个可屏蔽中断通道，包含EXTI、TIM、ADC、USART、SPI、I2C、RTC等多个外设。

我们今天要解决的就是EXIT（External interrupt）即GPIO外部中断

EXTI简介

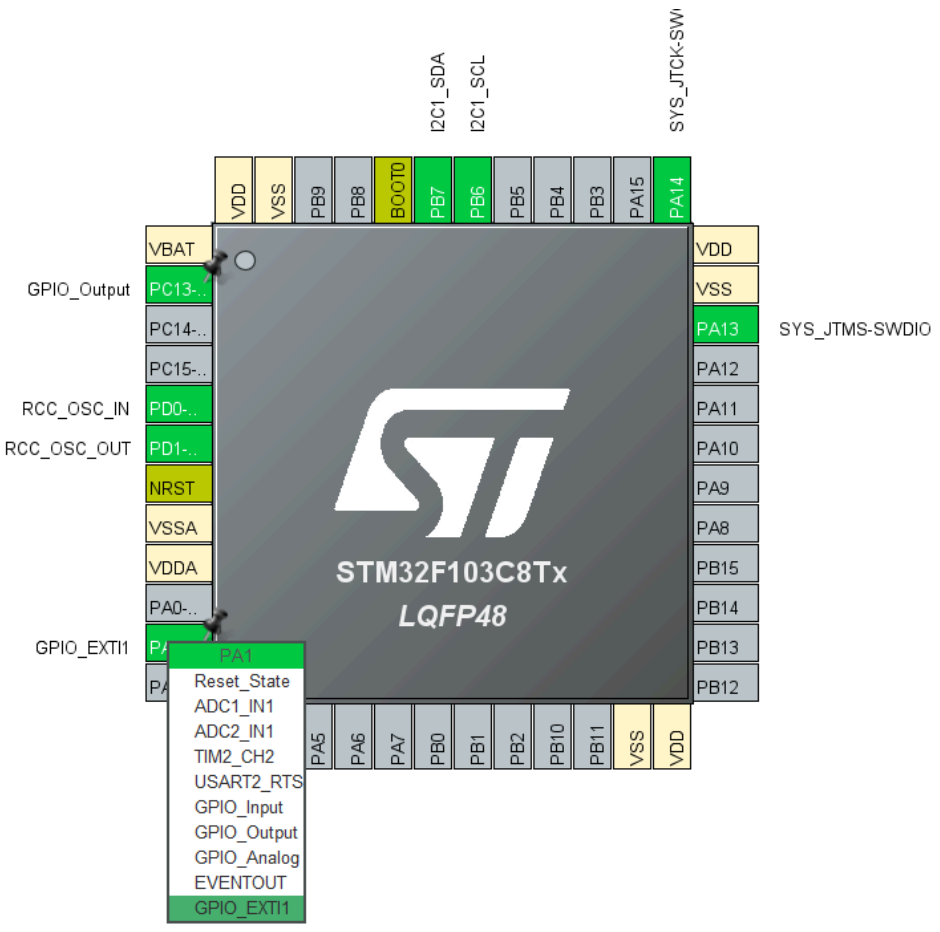
EXTI（Extern Interrupt）可以监测指定GPIO口的电平信号，当其指定的GPIO口产生电平变化时，EXTI将立即向NVIC发出中断申请，经过NVIC裁决后即可中断CPU主程序，使CPU执行EXTI对应的中断程序。

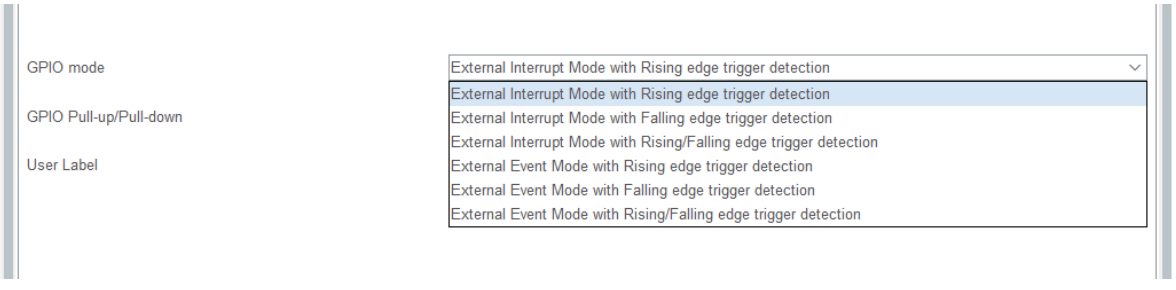
EXTI支持所有GPIO口，但注意引脚数字相同的Pin不能同时触发中断（比如PA0和PB0不能同时使用，但PA0和PA1可以同时使用），这是由其内部结构决定的。

cube配置与代码

中断的配置尤其需要注意，中断需要使能，否则无法发挥作用，一定要记住在cube中需要勾选哪些选项，在代码中需要什么使能语句

将引脚选为外部中断模式

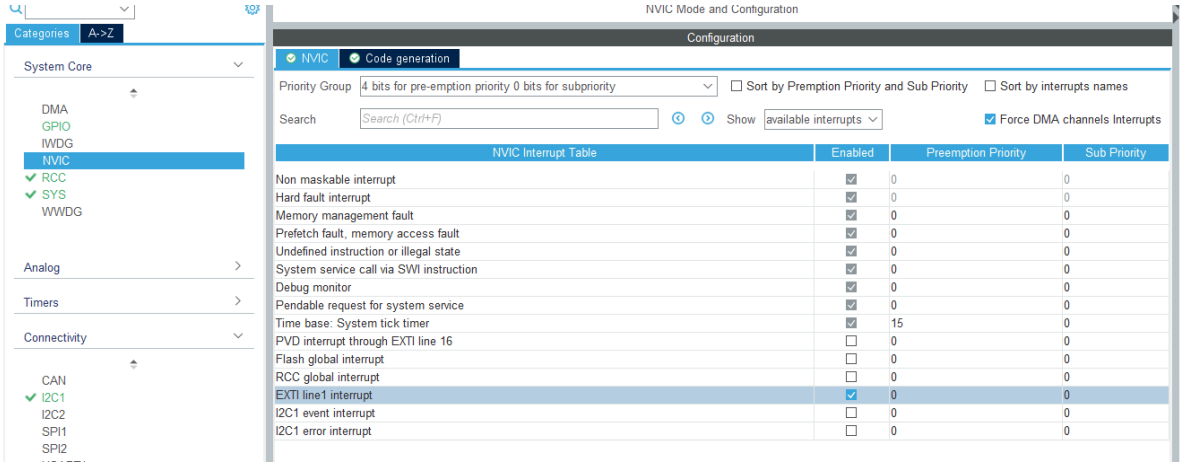




- 边沿检测方式：上升沿/下降沿
- 引脚拉高/拉低：与上述同理，一般依边沿检测方式而定

> 中断和事件的区别
> 产生中断线路目的是把输入信号输入到 NVIC，进一步会运行中断服务函数，实现功能，这样是软件级的。而产生事件线路目的就是传输一个脉冲信号给其他外设使用，并且是电路级别的信号传输，属于硬件级的。
>
> 简单点来说，就是中断最终会自动进入中断服务函数，而事件则不会有对应的服务函数。中断是软件级的，而事件是硬件级的。
> ————————————————
>

然后一定要记得在NVIC这里把对应的EXIT使能给勾上！！



中断回调函数：（本身是一个弱定义，可以重新定义）

注意中断函数里面不要出现HAL_Delay()和死循环！！！！！

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if(GPIO_Pin==GPIO_PIN_1)                                    //因为所有外部中断最后都会调用这个函数，所以要判断一下是哪个引脚的
  {

   // __HAL_GPIO_EXTI_CLEAR_IT(GPIO_PIN_1);
  }
  else if(GPIO_Pin==GPIO_PIN_2)
  {

   // __HAL_GPIO_EXTI_CLEAR_IT(GPIO_PIN_2);
  }
}
```

特别提醒：板子的一些电气特性

加在器件上的载荷如果超过'绝对最大额定值'列表(表6、表7、表8)中给出的值，可能会导致器件永久 性地损坏。这里只是给出能承受的最大载荷，并不意味在此条件下器件的功能性操作无误。器件长 期工作在最大值条件下会影响器件的可靠性。

表6　电压特性

符 号	描 述	最小值	最大值	单 位
V_{DD} - V_{SS}	外部主供电电压(包含V_{DDA}和V_{DD})(1)	-0.3	4.0	V
V_{IN}	在5V容忍的引脚上的输入电压(2)	V_{SS}-0.3	5.5	
	在其它引脚上的输入电压(2)	V_{SS}-0.3	V_{DD} + 0.3	
| ΔV_{DDx} |	不同供电引脚之间的电压差		50	mV
| V_{SSx} - V_{SS} |	不同接地引脚之间的电压差		50	
$V_{ESD(HBM)}$	ESD静电放电电压(人体模型)	参见第5.3.11节		

1．所有的电源(V_{DD}, V_{DDA})和地(V_{SS}, V_{SSA})引脚必须始终连接到外部允许范围内的供电系统上。
2．$I_{INJ(PIN)}$绝对不可以超过它的极限(见表7)，即保证V_{IN}不超过其最大值。如果不能保证V_{IN}不超过其最大值，也要保证在外部限制$I_{INJ(PIN)}$不超过其最大值。当V_{IN}>V_{IN}max时，有一个正向注入电流；当V_{IN}<V_{SS}时，有一个反向注入电流。

表7　电流特性

符号	描述	最大值	单位
I_{VDD}	经过V_{DD}/V_{DDA}电源线的总电流(供应电流)(1)	150	mA
I_{VSS}	经过V_{SS}地线的总电流(流出电流)(1)	150	
I_{IO}	任意I/O和控制引脚上的输出灌电流	25	
	任意I/O和控制引脚上的输出电流	-25	
$I_{INJ(PIN)}$ (2) (3)	NRST引脚的注入电流	±5	
	HSE的OSC_IN引脚和LSE的OSC_IN引脚的注入电流	±5	
	其他引脚的注入电流(4)	±5	
$\Sigma I_{INJ(PIN)}$(2)	所有I/O和控制引脚上的总注入电流(4)	±25	

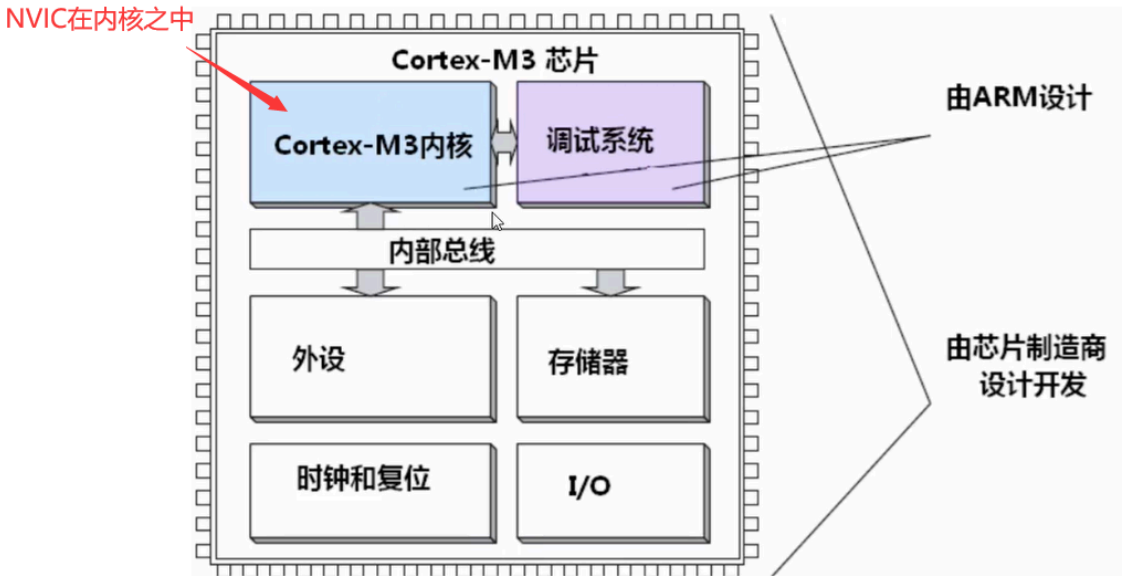
1. 所有的电源(V_{DD}，V_{DDA})和地(V_{SS}，V_{SSA})引脚必须始终连接到外部允许范围内的供电系统上。
2. $I_{INJ(PIN)}$绝对不可以超过它的极限，即保证V_{IN}不超过其最大值。如果不能保证V_{IN}不超过其最大值，也要保证在外部限制$I_{INJ(PIN)}$不超过其最大值。当V_{IN}>V_{DD}时，有一个正向注入电流；当V_{IN}<V_{SS}时，有一个反向注入电流。
3. 反向注入电流会干扰器件的模拟性能。参看第5.3.17节。
4. 当几个I/O口同时有注入电流时，$\Sigma I_{INJ(PIN)}$的最大值为正向注入电流与反向注入电流的即时绝对值之和。该结果基于在器件4个I/O端口上$\Sigma I_{INJ(PIN)}$最大值的特性。

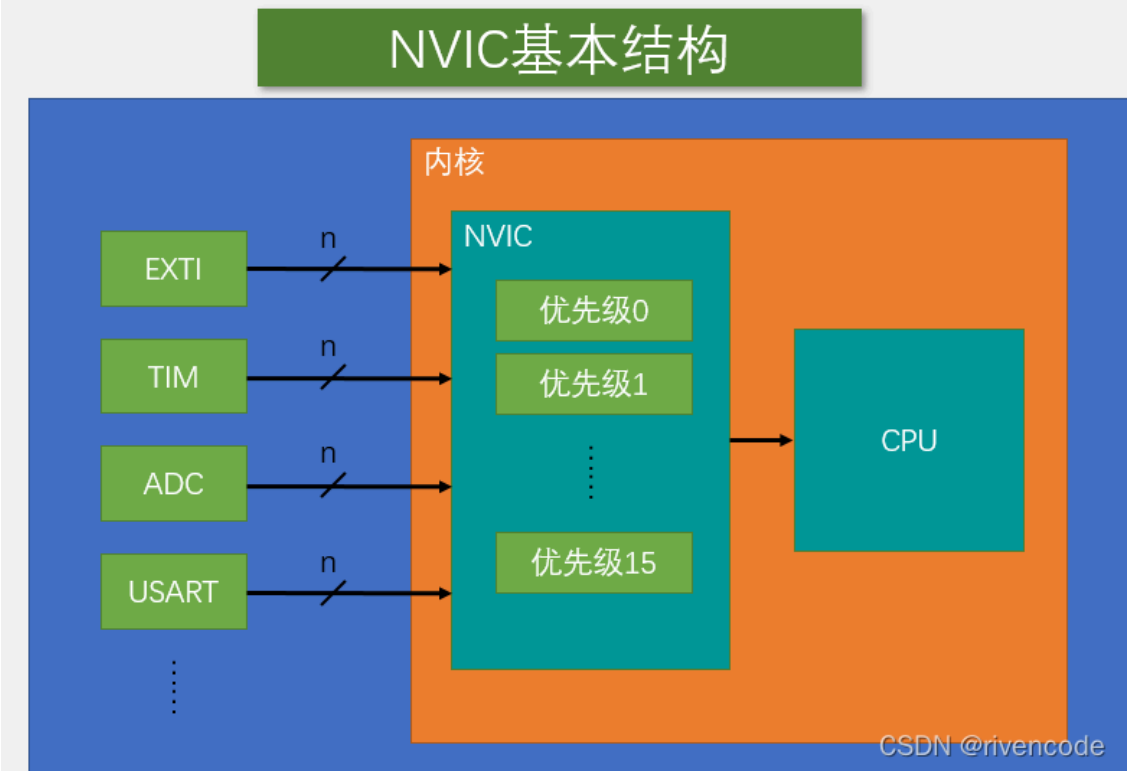
表8　温度特性

符号	描述	数值	单位
T_{STG}	储存温度范围	-65 ~ + 150	℃
T_J	最大结温度	150	℃

原理后置

> 中断优先级的配置：NVIC





什么是NVIC？即嵌套向量中断控制器(Nested Vectored Interrupt Controller)。CM3的中有一个强大而方便的NVIC，它是属于Cortex内核的器件，中断向量表中60个中断都由它来处理。NVIC是Cortex-M3核心的一部分，关于它的资料不在《STM32的技术参考手册》中，应查阅ARM公司的《Cortex-M3技术参考手册》。Cortex-M3的向量中断统一由NVIC管理。

NVIC的核心功能是中断优先级分组、中断优先级的配置、读中断请求标志、清除中断请求标志、使能中断、清除中断等，它控制着STM32中断向量表中中断号为0-59的60个中断！！外部中断信号从核外发出，信号最终要传递到NVIC(嵌套向量中断控制器)。NVIC跟内核紧密耦合，它控制着整个芯片中断的相关功能。

EXTI（External interrupt/event controller）—外部中断/事件控制器，管理了控制器的 20 个中断/事件线。每个中断/事件线都对应有一个边沿检测器，可以实现输入信号的上升沿检测和下降沿的检测。EXTI 可以实现对每个中断/事件线进行单独配置，可以单独配置为中断或者事件，以及触发事件的属性。

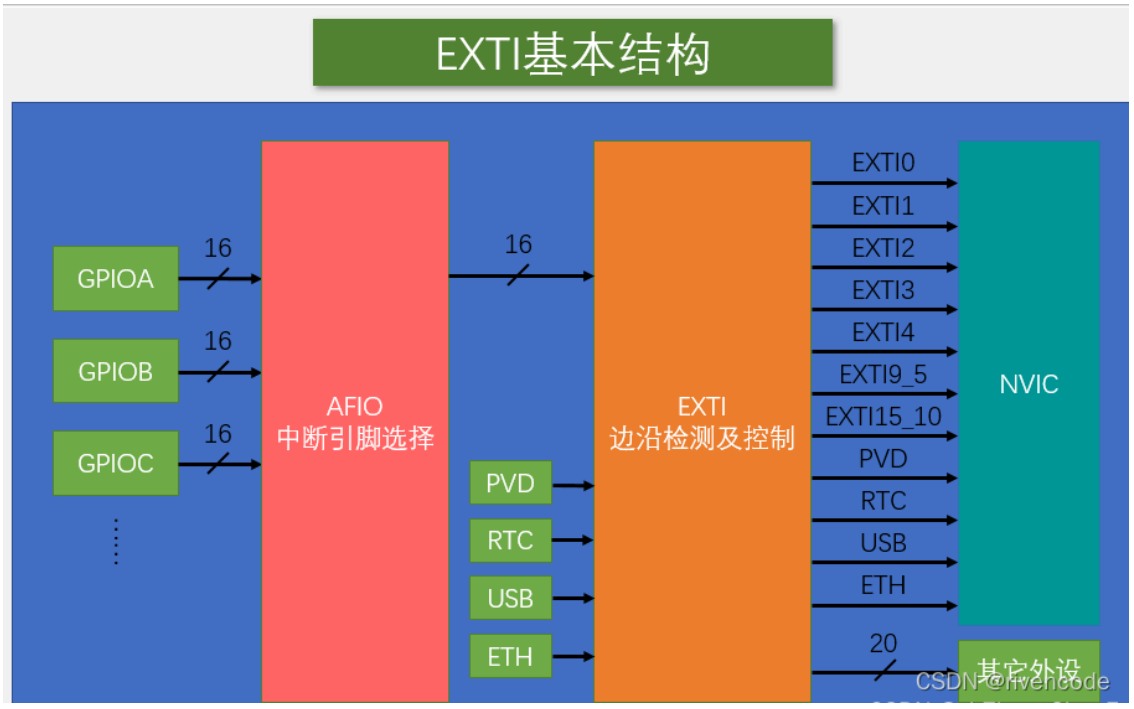
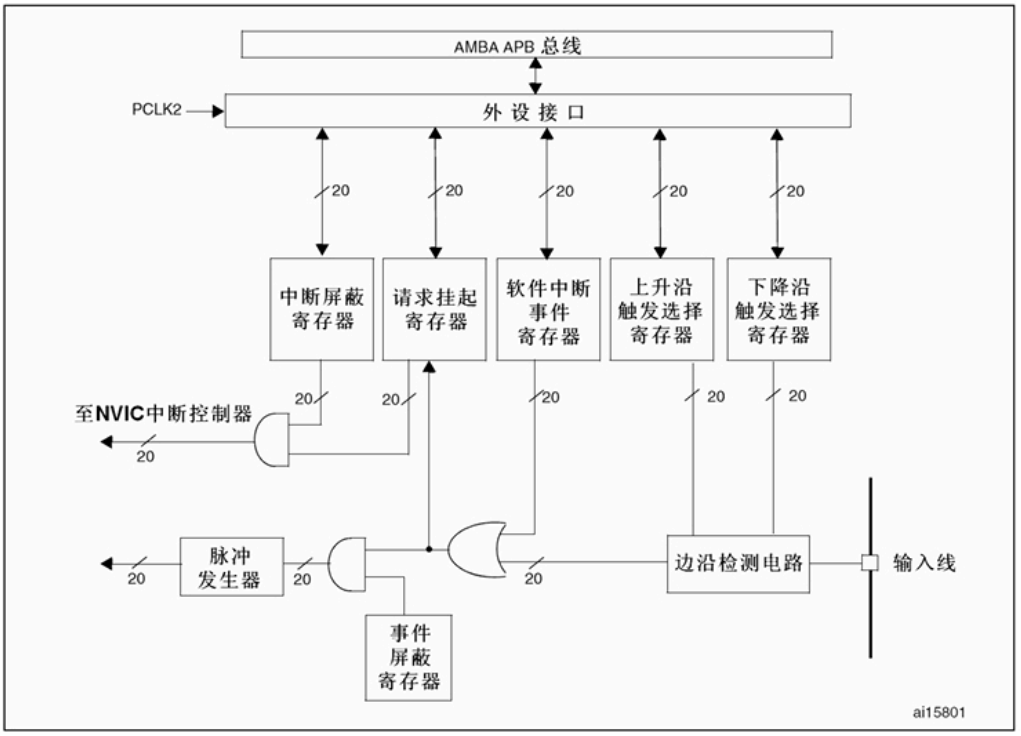


图19　外部中断/事件控制器框图



输入线--->边沿检测电路--->边沿触发寄存器--->请求挂起寄存器（即PR寄存器，对应位置一）---与上中断屏蔽寄存器（若使能了则对应位为1，表示不被屏蔽）------>到达NVIC，调用中断服务函数

中断流程：

触发中断

进入`void EXTI1_IRQHandler(void)`中断服务函数

```
/**
  * @brief This function handles EXTI line1 interrupt.
  */
void EXTI1_IRQHandler(void)
{
  /* USER CODE BEGIN EXTI1_IRQn 0 */

  /* USER CODE END EXTI1_IRQn 0 */
  HAL_GPIO_EXTI_IRQHandler(GPIO_PIN_1);      //调用外部中断处理函数
  /* USER CODE BEGIN EXTI1_IRQn 1 */

  /* USER CODE END EXTI1_IRQn 1 */
}
```

```
/**
  * @brief  This function handles EXTI interrupt request.
  * @param  GPIO_Pin: Specifies the pins connected EXTI line
  * @retval None
  */
void HAL_GPIO_EXTI_IRQHandler(uint16_t GPIO_Pin)
{
  /* EXTI line interrupt detected */
  if (__HAL_GPIO_EXTI_GET_IT(GPIO_Pin) != 0x00u)     //判断是哪一条线触发的中断（尤其是对EXIT5_9,EXIT15_10这种）（查询挂起寄存器）
  {
    __HAL_GPIO_EXTI_CLEAR_IT(GPIO_Pin);      //清除标志位，防止一直触发中断（清除挂起寄存器对应的标志位）
    HAL_GPIO_EXTI_Callback(GPIO_Pin);        //中断回调函数，在里面编写中断处理内容
  }
}
```

```
555
556 /**
557   * @brief  EXTI line detection callbacks.
558   * @param  GPIO_Pin: Specifies the pins connected EXTI line
559   * @retval None
560   */
561 __weak void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
562 {
563   /* Prevent unused argument(s) compilation warning */
564   UNUSED(GPIO_Pin);
565   /* NOTE: This function Should not be modified, when the callback is needed,
566            the HAL_GPIO_EXTI_Callback could be implemented in the user file
567    */
568 }
569
```

（本身是弱定义函数，便于重新定义复写）

附资料来源：[中断-NVIC与EXTI外设详解(超全面)-CSDN博客]

https://blog.csdn.net/qq_38410730/article/details/79829983

https://blog.csdn.net/qq_45361382/article/details/113809304

STM32入门教程（EXTI外部中断篇） 请求挂起寄存器和中断标志位一样吗-CSDN博客

作业与评分

1. 点亮板载LED

   (1) 让LED长亮，并在三秒后熄灭　　（3分）

   (2) 让LED反复亮灭，时间间隔约为半秒　　（3分）

2. 用GPIO输入模式检测PA1引脚电平变化（非中断方法）
   现象要求：上电后LED先长亮，将3.3v引入PA1之后LED熄灭　　（4分）

3. 用外部中断控制LED的闪烁模式
   现象要求：上电后LED长亮，将 3.3v引入PA2后LED开始以200ms的时间间隔翻转　　（4分）

4. （进阶版）

   现象要求：上电后LED以如下规律亮暗交替：

   亮0.5s-->灭100ms-->亮0.5s-->灭200ms-->亮0.5s-->灭300ms-->亮0.5s-->灭400ms-->亮0.5s-->.....--->亮0.5s-->灭n*100ms--->.....　（2分）

   将3.3v引入PA3后 n开始递减，直到回到100ms后保持（重点在递减这个现象）　　（2分）

   将3.3v引入PA4后 LED开始以1s的时间间隔翻转　　（2分）

说明：1.作业需要提交代码和现象视频，将两者打包好后以zip形式发送到18842898017@163.com

2.每一项作业对应一个项目工程（我指的是第四项作业不要拆成两个工程写） 如果有佬佬觉得四个工程太多了太占电脑内存可以使用预编译把上述功能全部放进一个工程里面~

祝大家学习愉快~