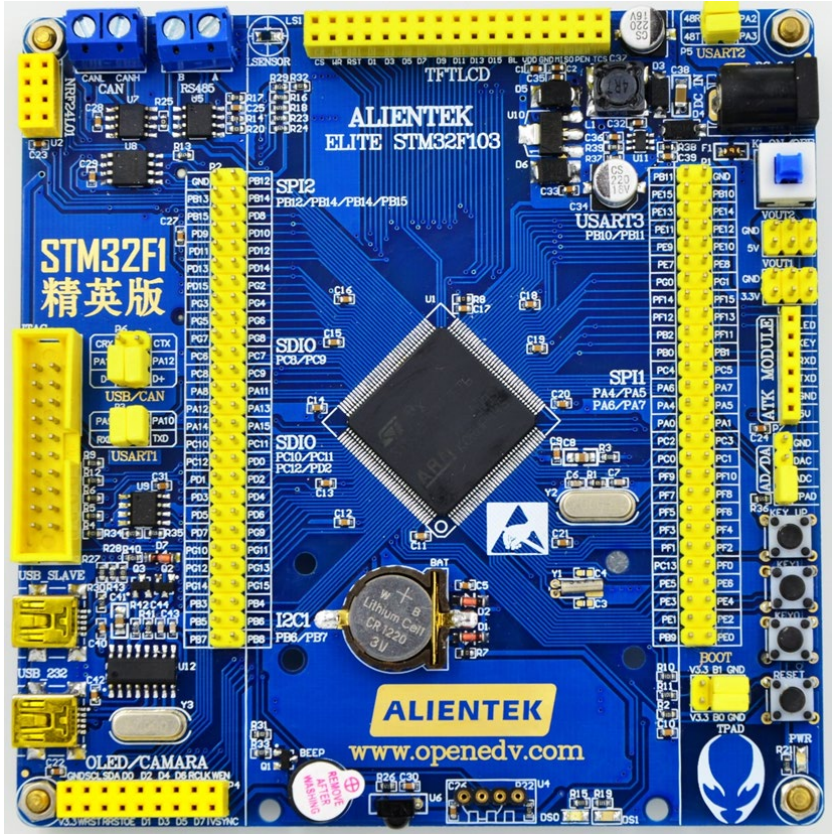
嵌入式系统实验

——STM32开发板实验

浙江大学机械工程学院 实验教学中心

第 2 次实验

- **1. 按键输入实验（必做）**
- 2. 外部中断实验（必做）



STM32F103ZET6 开发板

1.1 实验任务

利用按键扫描方式实现以下功能：

1、KEY0按下：DS0、DS1长亮

2、KEY1按下：DS0、DS1均不亮

3、KEY_UP按下：蜂鸣器状态翻转（按键做防抖处理、区分长短按）

4、建议先仿真再实测

1.2 GPIO使用流程

01

第一步

使能按键对应IO口时钟。调用函数：

RCC_APB2PeriphClockCmd();

易错

- GPIO_InitTypeDef GPIO_InitStructure;
- RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE|RCC_APB2Periph_GPIOA,ENABLE);

02

第二步

初始化IO口：包括引脚名称、传输速率、引脚工作模式调用函数：

GPIO_Init();

易错

- GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IPU;
  GPIO_InitStructure.GPIO_Pin=GPIO_Pin_3|GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
  GPIO_Init(GPIOE,&GPIO_InitStructure);

03

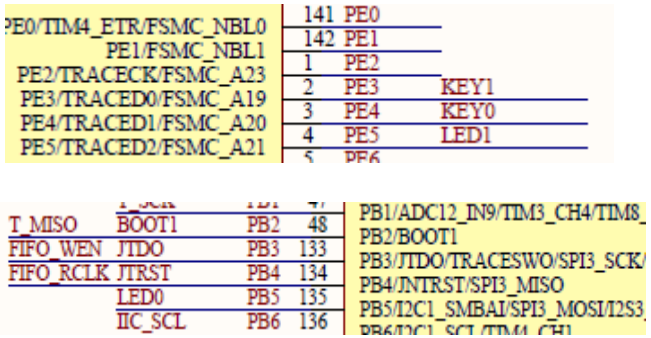
第三步

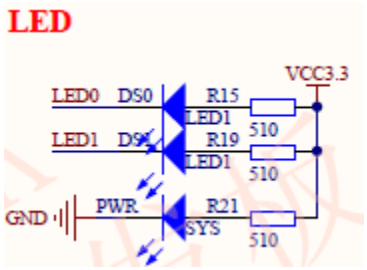
循环检测：

uint8_t GPIO_ReadInputDataBit();

- GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_4)

1.3 实验原理-硬件连接

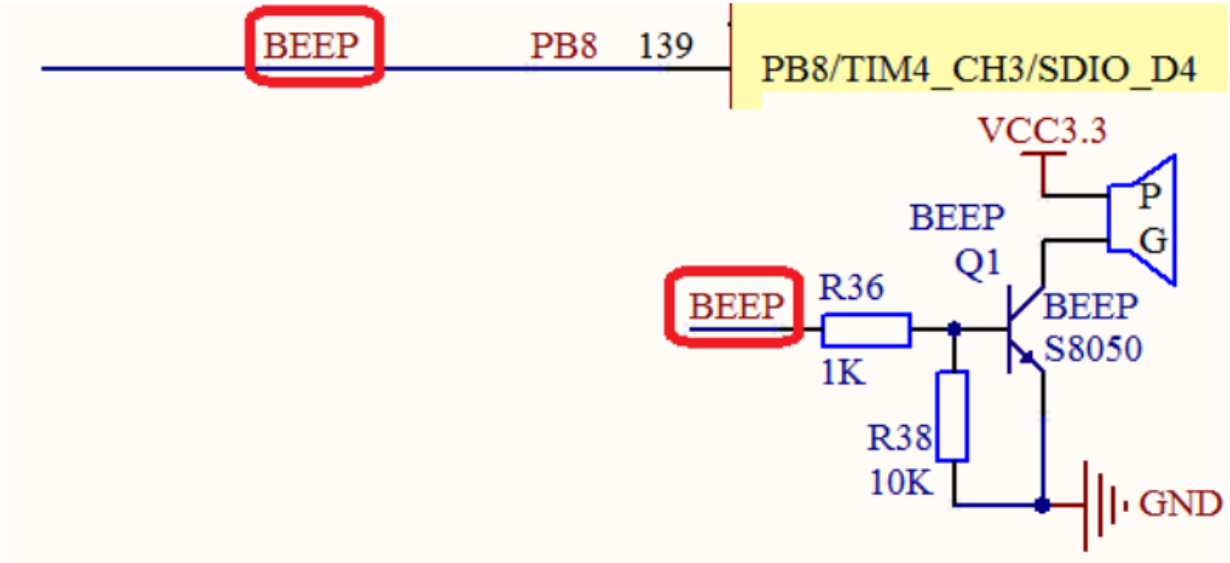




LED的IO口

—PB5、PE5口

— 按上节课设置为推挽输出模式

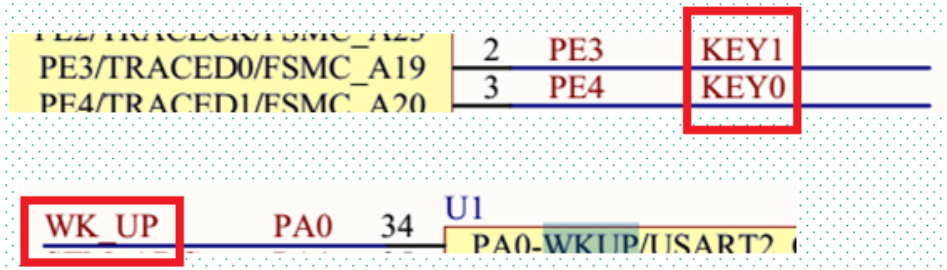


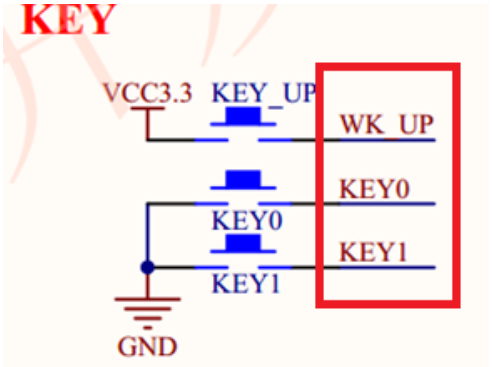
BEEP的IO口

—PB8口

—推挽输出模式

1.3 实验原理-硬件连接



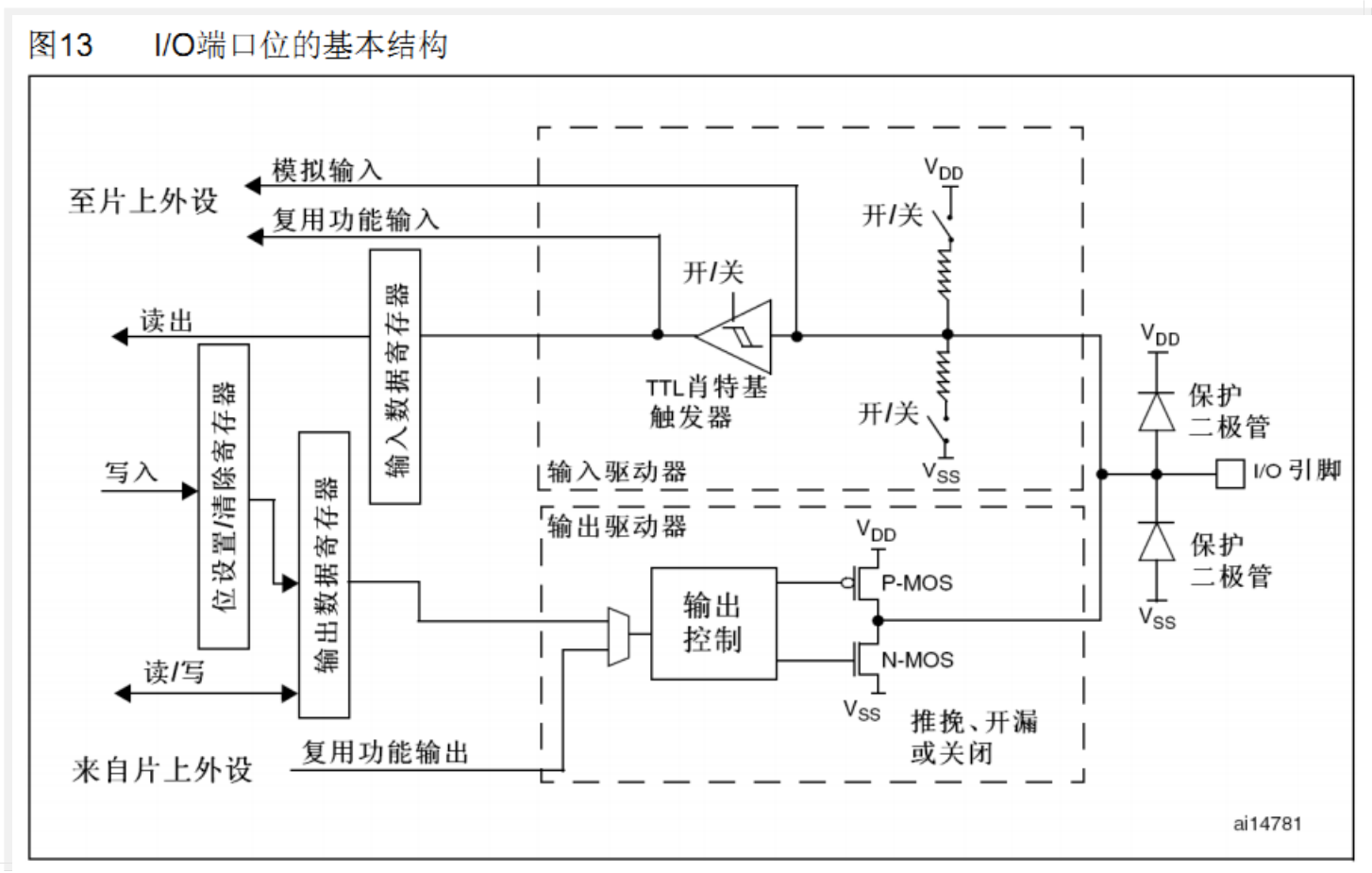


按键的IO口

—KEY0、KEY1:PE3、PE4口

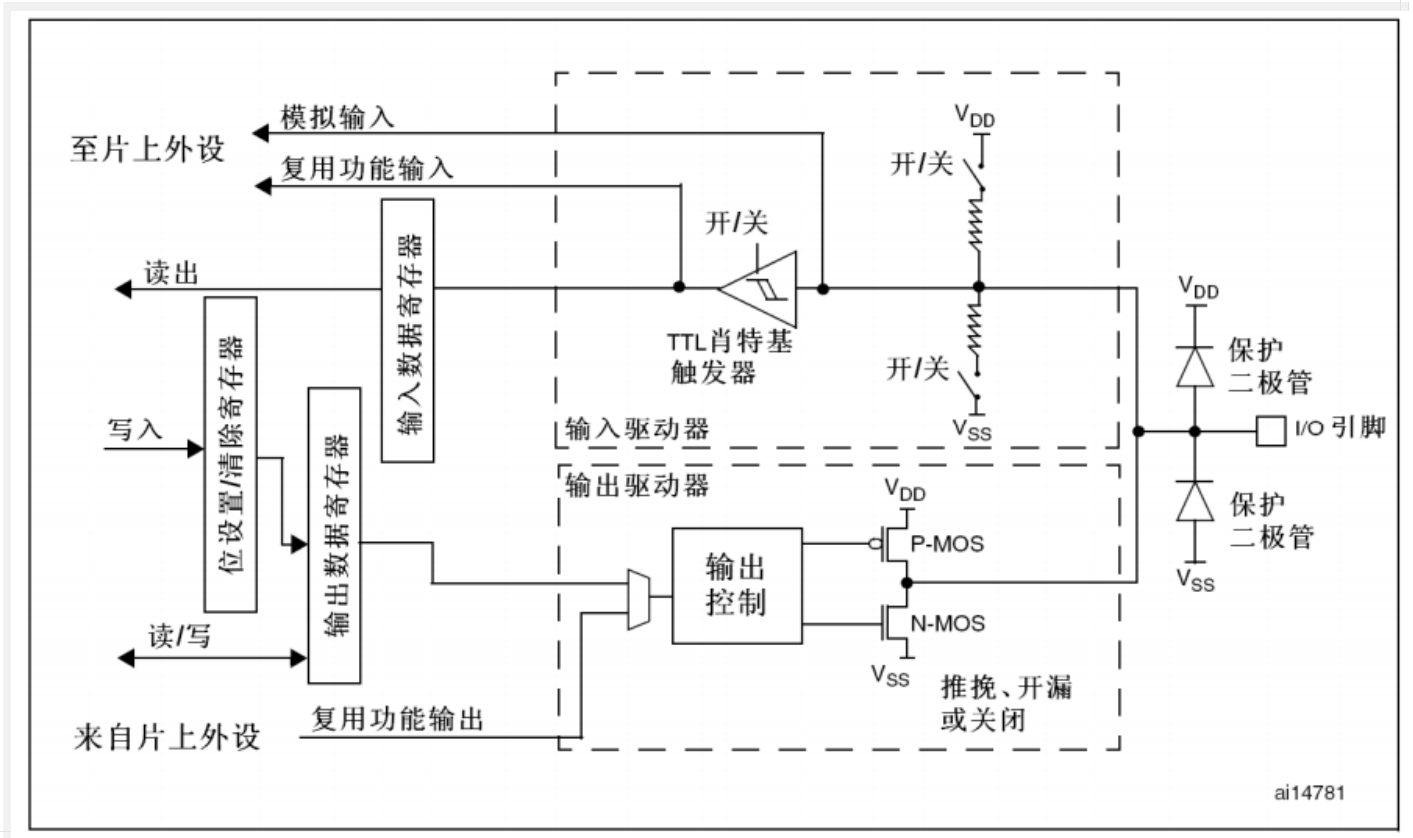
1.4 实验原理-GPIO工作方式

- **4种输入模式：**
  - 输入浮空
  - 输入上拉
  - 输入下拉
  - 模拟输入
- **4种输出模式：**
  - 开漏输出
  - 开漏复用功能
  - 推挽式输出
  - 推挽式复用功能



参考： 《STM32中文参考手册V10》-第8章GPIO和AFIO或datasheet《stm32F103ZET6》

1.4 实验原理-GPIO工作方式

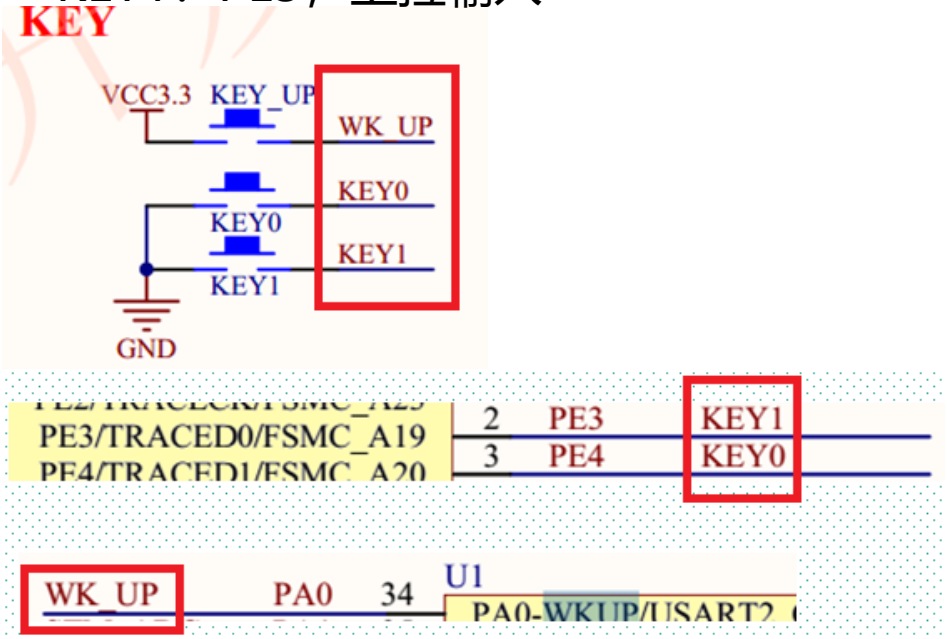


KEY的IO口

— KEY_UP：PA0,下拉输入

— KEY0：PE4，上拉输入

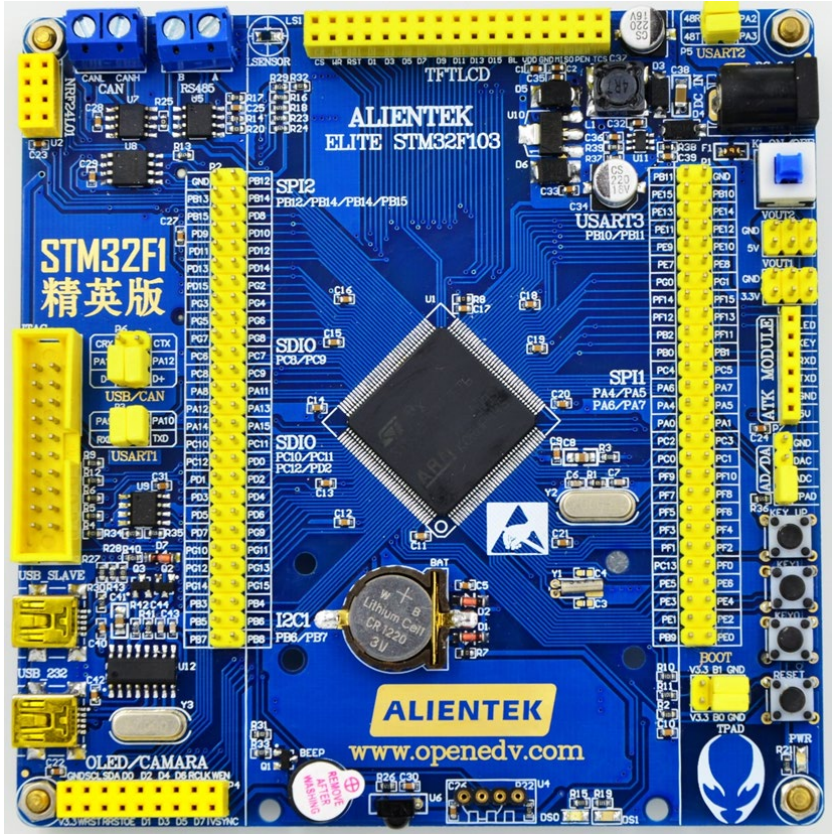
—KEY1：PE3，上拉输入



参考： 《STM32中文参考手册V10》-第8章GPIO和AFIO或datasheet《stm32F103ZET6》

第 2 次实验

- 1. 按键输入实验
- **2. 外部中断实验**
- 3. 按键扫描（选做）



STM32F103ZET6 开发板

2.1 实验任务

任务

1、主程序：DS0闪亮

2、KEY1按下产生中断：DS1亮5s

3、KEY_UP按下产生中断：蜂鸣器啸叫5s

4、要求：KEY1、KEY_UP的中断设在group2，KEY_UP优先级高于KEY1

2.2 IO中断使用流程

01

第一步

开启复用时钟、IO口时钟。调用函数：

RCC_APB2PeriphClockCmd();

- RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO|RCC_APB2Periph_GPIOA,ENABLE);

02

第二步

配置IO口，包括引脚名称、传输速率、引脚工作模式调用函数：

GPIO_Init();

- GPIO_InitStructure.GPIO_Mode=GPIO_Mode_IPU;
  GPIO_InitStructure.GPIO_Pin=GPIO_Pin_3|GPIO_Pin_4;
  GPIO_InitStructure.GPIO_Speed=GPIO_Speed_50MHz;
  GPIO_Init(GPIOE,&GPIO_InitStructure);

03

第三步

建立IO口与中断线的映射

GPIO_EXTILineConfig();

- GPIO_EXTILineConfig(GPIO_PortSourceGPIOE,GPIO_PinSource4)

2.2 IO中断使用流程

外部中断概念

- STM32的每个IO都可以作为外部中断输入。
- STM32的中断控制器支持19个外部中断/事件请求：

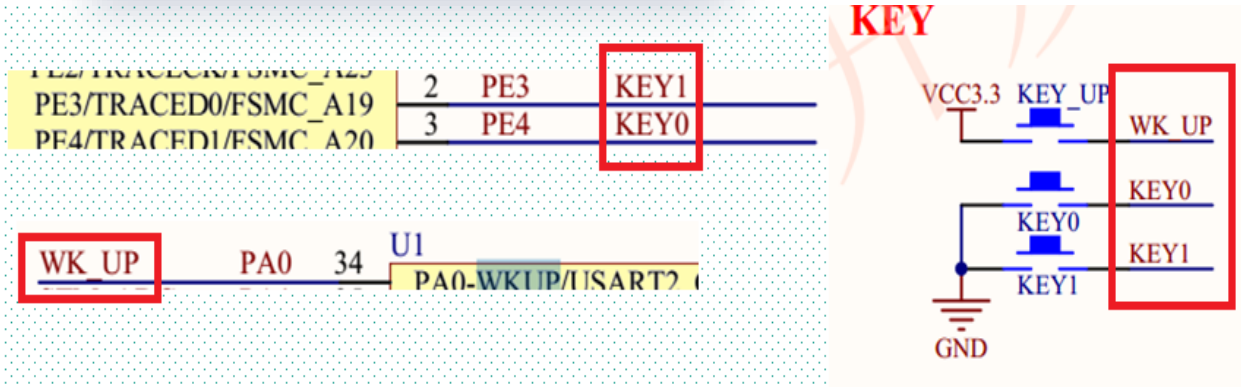
  线0~15：对应外部IO口的输入中断。
  线16：连接到PVD输出。
  线17：连接到RTC闹钟事件。
  线18：连接到USB唤醒事件。

  从上面可以看出，STM32供IO使用的中断线只有16个，但是STM32F10x系列的IO口多达上百个，STM32F103ZET6(112), STM32F103RCT6(51)，那么中断线怎么跟io口对应呢？

提示：中断知识可以看《STM32中文参考手册_V10》的第9章和《STM32F1开发指南(精英版)-库函数版本_V1.3》第10章

2.2 IO中断使用流程

GPIO与中断映射关系



易错

- 本次实验用到中断线：

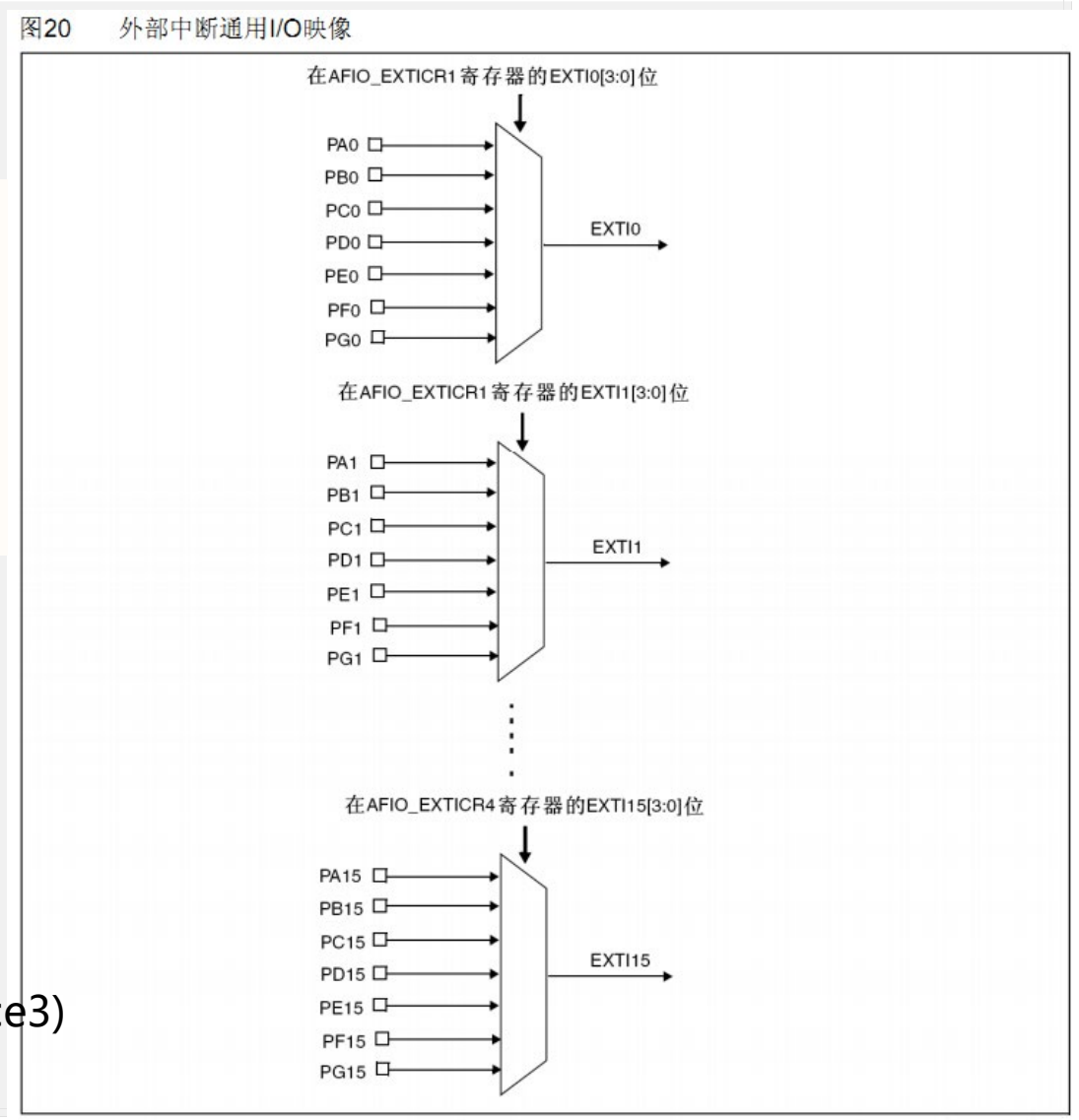
1)WK_UP：PA0映射到EXTI0

2)KEY1：PE3映射到EXTI3

3)KEY0：PE4映射到EXTI4

- 中断线映射函数

GPIO_EXTILineConfig(GPIO_PortSourceGPIOE,GPIO_PinSource3)



2.2 IO中断使用流程

04

第四步

中断初始化：

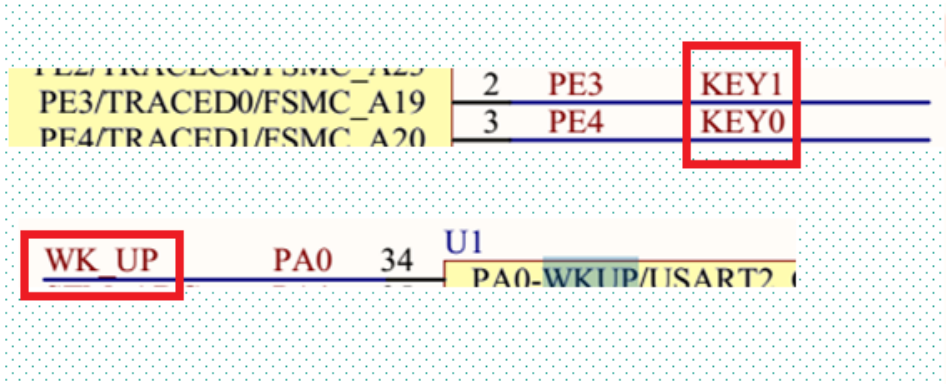
配置中断线、中断模式

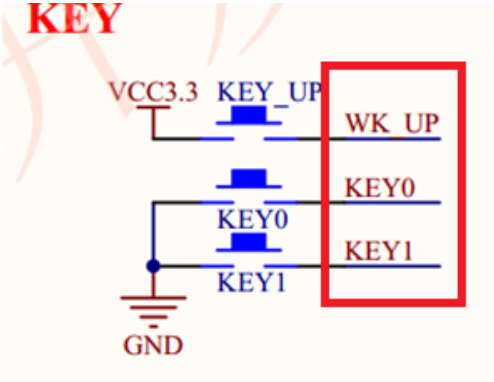
中断触发方式、中断使能

函数：EXTI_Init();

易错

- EXTI_InitStructure.EXTI_Line=EXTI_Line3;//配中断线
- EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//模式事件or中断
- EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;//触发方式
- EXTI_InitStructure.EXTI_LineCmd=ENABLE;//使能中断
- EXTI_Init(&EXTI_InitStructure);





2.2 IO中断使用流程

05

第五步

中断优先级分组：

NVIC_PriorityGroupConfig()

- NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2)

易错

06

第六步

中断优先级初始化

NVIC_Init();

- NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn; //设所在中断线

//设抢占优先级

- NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x02;

//设子优先级

- NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x03;

//使能外部中断通道

- NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;
- NVIC_Init(&NVIC_InitStructure); //根据参数初始化

2.2 IO中断使用流程

中断优先级NVIC

首先，对STM32中断进行分组，组0~4。同时，对每个中断设置一个抢占优先级和一个响应优先级值。

分组配置是在寄存器SCB->AIRCR中配置：

组	AIRCR[10：8]	IP bit[7：4]分配情况	分配结果
0	111	0：4	0位抢占优先级，4位响应优先级
1	110	1：3	1位抢占优先级，3位响应优先级
2	101	2：2	2位抢占优先级，2位响应优先级
3	100	3：1	3位抢占优先级，1位响应优先级
4	011	4：0	4位抢占优先级，0位响应优先级

2.2 IO中断使用流程

07

第七步

写中断服务函数：

EXTIx_IRQHandler()

易错

```
void EXTI3_IRQHandler(void)
{
if(EXTI_GetITStatus(EXTI_Line3)!=RESET)//判断某个线上的中断是否发生
{
中断逻辑…
EXTI_ClearITPendingBit(EXTI_Line3); //清除LINE上的中断标志位
}
}
```

易错

0-4每个中断线对应一个中断函数，中断线5-9共用中断函数EXTI9_5_IRQHandler：

```
EXTI0_IRQHandler
EXTI1_IRQHandler
EXTI2_IRQHandler
EXTI3_IRQHandler
EXTI4_IRQHandler
EXTI9_5_IRQHandler
EXTI15_10_IRQHandler
```

2.2 IO中断使用流程

中断服务函数

0-4每个中断线对应一个中断函数，中断线5-9共用中断函数EXTI9_5_IRQHandler：

```
EXTI0_IRQHandler
EXTI1_IRQHandler
EXTI2_IRQHandler
EXTI3_IRQHandler
EXTI4_IRQHandler
EXTI9_5_IRQHandler
EXTI15_10_IRQHandler
```

易错

其他常用中断函数

```
ITStatus EXTI_GetITStatus(uint32_t EXTI_Line)；//第一个函数是判断某个中断线上的中断是否发生
EXTI_ClearITPendingBit(uint32_t EXTI_Line)；//清除某个中断线上的中断标志位：
EXTI_GetFlagStatus; //判断外部中断状态
EXTI_ClearFlag;  //清除外部状态
```

易错

其它常见错误

- 库函数保存的位置不正确、编译路径未加，导致引用头文件找不到路径
- 库函数的引用原则：用到哪个库引用哪个库的头文件，不要引用main.h
- 变量定义放在函数的最前面
- 中断服务函数名写错
- 函数名的命名可读性要强，错误案例：

  按键的初始化函数名为：KEY_init()，按键中断的初始化的函数命名为key_init()，

  在main函数里只调用了key_init()，很难发现错误。
- 提示：按键扫描要进行防抖处理，用简单的延迟10ms再检测一次的方法

  if 按键按下

  delay（10ms）

  if 按键按下

  while(GPIO_ReadInputDataBit(GPIOE,GPIO_Pin_1)==0); //等待按键松开

3.1 实验任务

任务3

使用按键扫描的方式实现：

1、主程序：DS0闪亮

2、KEY1按下产生中断：DS1长亮

2、KEY0按下下产生中断：DS1不亮

3、KEY_UP支持长按和短按：

长按使得DS1在闪亮与关闭之间切换，

短按功能是开关蜂鸣器

4、要求：KEY1、 KEY0、 KEY_UP的中断设在group2，KEY_UP优先级高于KEY1