实验四 外部中断处理

一、实验目的

1. 掌握STM32F4外部中断的基本原理和配置方法。
2. 学会使用外部中断控制数码管和LED的反转。

二、实验原理

本实验通过STM32F4的外部中断功能，实现对数码管和LED的控制。当按下KEY0、KEY1时，数码管清空，对应的LED反转；当按下KEY-UP时，数码管显示“ON”，同时开启KEY0-2的中断源具体效果如下。

1. KEY0 数码管清空，LED1反转
2. KEY1 数码管清空，LED2反转
3. KEY2 数码管显示“OFF”，并且关闭KEY0-2的中断源
4. KEY-UP 数码管显示“ON”，同时开启KEY0-2的中断源

三、实验硬件连接

1. STM32F4单片机
2. LED1、LED2
3. 数码管
4. 按键KEY0、KEY1、KEY2、KEY-UP

四、实验步骤

1. 配置系统时钟。
2. 初始化GPIO，设置按键输入模式。
3. 初始化EXTI，配置外部中断。
4. 编写中断服务函数，实现按键功能。
5. 编写主函数，循环检测按键状态。

五、实验步骤及代码

EXTI的初始化方法

1.在HARDWARE创建EXTI的头文件和源文件，用来写我们的中断配置程序

包含常用的头文件

```
#include "exti.h"
#include "key.h"
#include "sys.h"
#include "delay.h"
#include "led.h"
```

CSDN @宁子希

2.创建EXTI的初始化函数，在函数中定义NVIC 和 EXTI 结构体

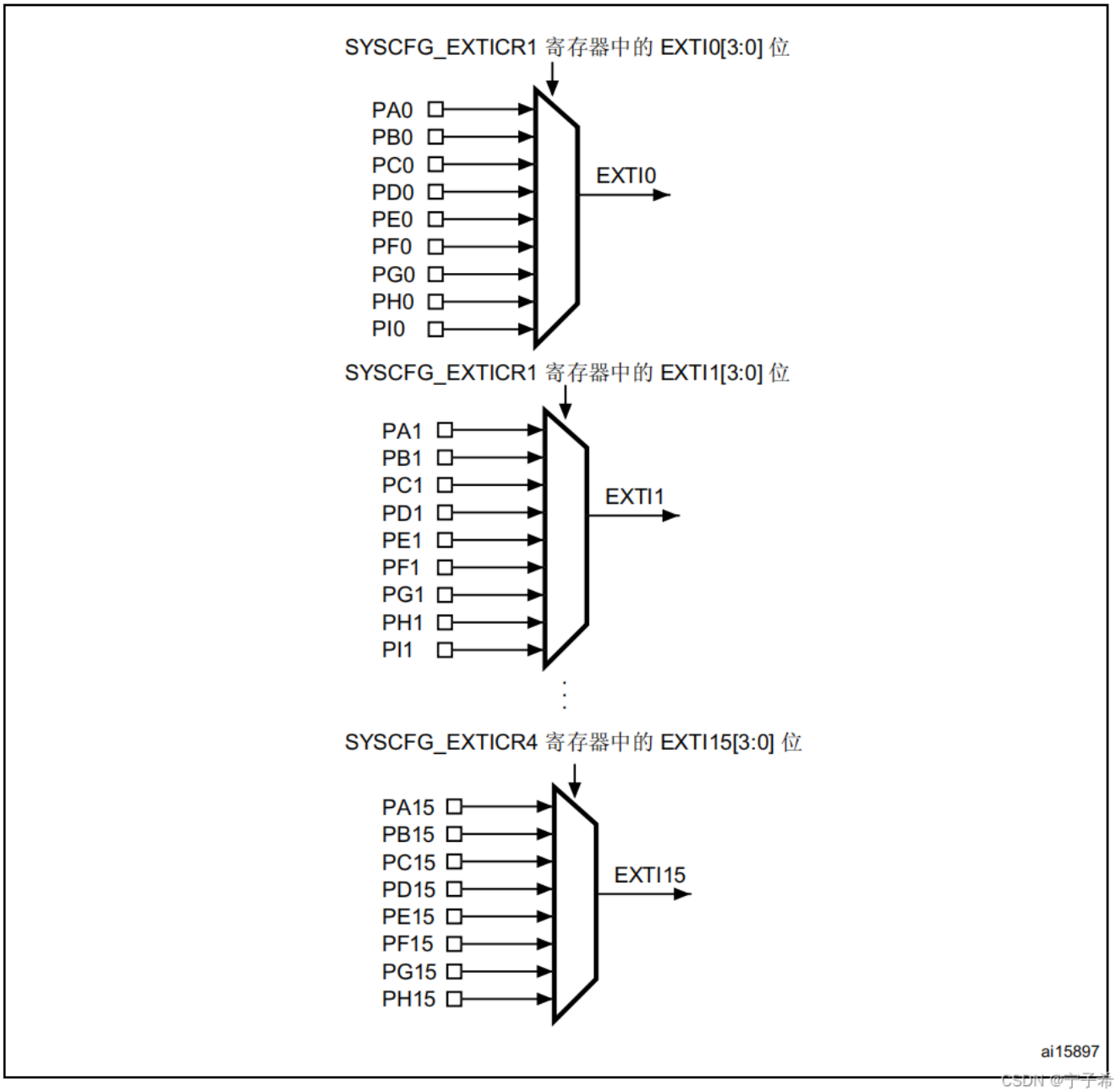
```
void EXTIX_Init(void)
{
    //定义NVIC 和 EXTI 结构体
    NVIC_InitTypeDef   NVIC_InitStructure;
    EXTI_InitTypeDef   EXTI_InitStructure;
```

CSDN @宁子希

3.按键对应的IO口初始化
使能SYSCFG时钟，使用APB
db5779511386fe036.png)

4.中断线以及中断初始化初始化配置

图 33.　　外部中断/事件 GPIO 映射



CSDN @宁子希

```
SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource11);
SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource12);
SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOC, EXTI_PinSource13);
SYSCFG_EXTILineConfig(EXTI_PortSourceGPIOA, EXTI_PinSource0);//PA0 连接到中断线0
```

CSDN @宁子希

这段代码是用于配置外部中断线（EXTI）的。库函数 `SYSCFG_EXTILineConfig` 来配置外部中断线。

这些引脚被配置为外部中断源，当对应的引脚上的电平发生变化时，会触发相应的中断事件

5.配置EXTI_Line0的结构体

```
/* 配置EXTI_Line0 */
EXTI_InitStructure.EXTI_Line = EXTI_Line0;//LINE0
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断事件
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Rising; //上升沿触发
EXTI_InitStructure.EXTI_LineCmd = ENABLE;//使能LINE0
EXTI_Init(&EXTI_InitStructure);//配置
```

CSDN @宁子希

这段代码是用于配置STM32的外部中断（EXTI）的初始化结构体。

这段代码的作用是配置STM32的外部中断，当GPIOC的第13引脚上的电平从低变高时，会触发一个中断事件。

```
/* 配置EXTI_Line2,3,4 */
EXTI_InitStructure.EXTI_Line = EXTI_Line11 | EXTI_Line12 | EXTI_Line13;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;//中断事件
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling; //下降沿触发
EXTI_InitStructure.EXTI_LineCmd = ENABLE;//中断线使能
EXTI_Init(&EXTI_InitStructure);//配置
```

CSDN @宁子希

同理继续配置EXTI_Line2,3,4

6.定义NVIC的结构体

```
NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;//外部中断0
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;//抢占优先级为0
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x03;//响应优先级为3
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;//使能外部中断通道
NVIC_Init(&NVIC_InitStructure);//配置

NVIC_InitStructure.NVIC_IRQChannel = EXTI15_10_IRQn;//外部中断2
NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0x00;
NVIC_InitStructure.NVIC_IRQChannelSubPriority = 0x02;
NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;//使能外部中断通道
NVIC_Init(&NVIC_InitStructure);//配置
}
```

CSDN @宁子希

这段代码是用于配置两个外部中断的初始化结构体（**注意：只有EXTI0~4有独立的中断向量，5 ~ 9和10 ~ 15共享中断向量EXTI9_5_IRQn和EXTI15_10_IRQn，**），并使能这两个中断通道。

中断服务函数

```
void EXTI0_IRQHandler(void)      // WK_UP按键 中断服务程序
{

        delay_ms(20);    //消抖
        if(KEY_UP==0)
```

```
        {
                //开启KEY0-2的中断源
                EXTI->IMR |= EXTI_Line11; //使能外部中断11 对应KEY0
                EXTI->IMR |= EXTI_Line12; //使能外部中断12 对应KEY1
                EXTI->IMR |= EXTI_Line13; //使能外部中断13 对应KEY2
                while(1)
                {
                        //显示on
                        SEG_display(17,1);
                        SEG_display(16,2);
                        delay_ms(10);
                        if(KEY0==0|KEY1==0|KEY2==0)     //其他按键按下跳出循环
                                break;
                }
        }

         EXTI_ClearITPendingBit(EXTI_Line0); //清除LINE0上的中断标志位
}

void EXTI15_10_IRQHandler(void)
{
        delay_ms(20);   //消抖
        if(KEY0==0)
        {
                //数码管清空，LED1反转
                SEG_Clear();
                LED1=!LED1;
        }
         EXTI_ClearITPendingBit(EXTI_Line11);  //清除LINE3上的中断标志位
        if(KEY1==0)
        {
                //数码管清空，LED2反转
                SEG_Clear();
                LED2=!LED2;
        }
         EXTI_ClearITPendingBit(EXTI_Line12);  //清除LINE3上的中断标志位
        if(KEY2==0)
        {
                //关闭KEY0-2的中断源前先清除LINE0上的中断标志位 防止无法进入EXTI_Line0中断线
                EXTI_ClearITPendingBit(EXTI_Line0);
                //并且关闭KEY0-2的中断源
                EXTI->IMR &= ~(EXTI_Line11); //屏蔽外部中断11
                EXTI->IMR &= ~(EXTI_Line12); //屏蔽外部中断12
                EXTI->IMR &= ~(EXTI_Line13); //屏蔽外部中断13
                //数码管先清空
                SEG_Clear();
                while(1)
                {
                        //数码管显示“OFF”
                        SEG_display(17,1);
                        SEG_display(15,2);
                        SEG_display(15,3);
                        delay_ms(10);
```

```
                        if(KEY0==0|KEY1==0|KEY_UP==0)   //其他按键按下跳出循环
                                break;
                }
        }
         EXTI_ClearITPendingBit(EXTI_Line13);  //清除LINE3上的中断标志位
}
```

这段代码是用于处理外部中断的。当WK_UP按键被按下时，会触发EXTI0_IRQHandler函数。在这个函数中，首先进行消抖操作，然后检查KEY_UP是否为0，如果为0，则开启KEY0-2的中断源，并进入一个循环，显示on，直到其他按键被按下跳出循环。最后清除LINE0上的中断标志位。

当KEY0被按下时，会触发EXTI15_10_IRQHandler函数。在这个函数中，首先进行消抖操作，然后分别检查KEY0、KEY1和KEY2是否为0，如果KEY0为0，则清空数码管并反转LED1；如果KEY1为0，则清空数码管并反转LED2；如果KEY2为0，则关闭KEY0-2的中断源前先清除LINE0上的中断标志位，防止无法进入EXTI_Line0中断线，并且关闭KEY0-2的中断源，清空数码管并显示“OFF”。最后清除LINE3上的中断标志位。

main函数

```
/*
外部中断控制数码管和LED的反转  项目完成时间2023/12/6

KEY0                    数码管清空，LED1反转
KEY1                    数码管清空，LED2反转
KEY2                    数码管显示“OFF”，并且关闭KEY0-2的中断源
KEY-UP                  数码管显示“ON”，同时开启KEY0-2的中断源
 */
#include "exti.h"
#include "key.h"
#include "sys.h"
#include "delay.h"
#include "led.h"

int main(void)
{
        NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);//设置系统中断优先级分组2 两位抢占两位响应

        delay_init(168);
        LED_Init();
        EXTIX_Init();
        while(1);
}
```

在主函数中。初始化系统中断优先级分组、延时模块、LED灯和外部中断模块，并进入一个无限循环。