STM32 的 IO 口在本章第一节有详细介绍，而外部中断在第二章也有详细的阐述。这里我

们将介绍如何将这两者结合起来，实现外部中断输入。

STM32 的每个 IO口都可以作为中断输入，这点很好用。要把 IO口作为外部中断输入，有

以下几个步骤：

**1）初始化 IO 口为输入。**

这一步设置你要作为外部中断输入的 IO 口的状态，可以设置为上拉/下拉输入，也可以设

置为浮空输入，但浮空的时候外部一定要带上拉，或者下拉电阻。否则可能导致中断不停的触

发。在干扰较大的地方，就算使用了上拉/下拉，也建议使用外部上拉/下拉电阻，这样可以一

定程度防止外部干扰带来的影响。

**2）开启 IO 口复用时钟，设置 IO 口与中断线的映射关系。**

STM32 的 IO 口与中断线的对应关系需要配置外部中断配置寄存器 EXTICR，这样我们要

先开启复用时钟，然后配置 IO口与中断线的对应关系。才能把外部中断与中断线连接起来。

**3）开启与该 IO 口相对的线上中断/事件，设置触发条件。**

这一步，我们要配置中断产生的条件，STM32可以配置成上升沿触发，下降沿触发，或者

任意电平变化触发，但是不能配置成高电平触发和低电平触发。这里根据自己的实际情况来配

置。同时要开启中断线上的中断，这里需要注意的是：如果使用外部中断，并设置该中断的EMR

位的话，会引起软件仿真不能跳到中断，而硬件上是可以的。而不设置 EMR，软件仿真就可以

进入中断服务函数，并且硬件上也是可以的。建议不要配置 EMR 位。

**4）配置中断分组（NVIC），并使能中断。**

这一步，我们就是配置中断的分组，以及使能，对 STM32的中断来说，只有配置了 NVIC

的设置，并开启才能被执行，否则是不会执行到中断服务函数里面去的。关于 NVIC 的详细介

绍，请参考前面章节。

**5）编写中断服务函数。**

这是中断设置的最后一步，中断服务函数，是必不可少的，如果在代码里面开启了中断，

但是没编写中断服务函数，就可能引起硬件错误，从而导致程序崩溃！所以在开启了某个中断

后，一定要记得为该中断编写服务函数。在终端服务函数里面编写你要执行的中断后的操作。

通过以上几个步骤的设置，我们就可以正常使用外部中断了。

这一节，我们将实现同第二节差不多的功能，但是这里我们使用的是中断来检测按键，还

是通过WK\_UP按键实现按一次DS0和DS1同时翻转，按KEY0翻转DS0，按KEY1翻转DS1。

void EXTI4\_IRQHandler(void)

{

EXTI\_ClearFlag(EXTI\_Line4); delay\_ms(10);//消抖

if(KEY2==1) //按键2

{

LED1=!LED1;

printf(" 中断 OK\n");

// LED2=!LED2;

}

}

//外部中断15~10服务程序

void EXTI9\_5\_IRQHandler(void)

{

EXTI\_ClearFlag(EXTI\_Line5|EXTI\_Line6|EXTI\_Line7);

delay\_ms(10); //消抖

LED3=!LED3;

LED4=!LED4;

printf(" 中断9-5 OK\n");

}

//外部中断初始化程序

//初始化PA0,PA13,PA15为中断输入.

void EXTIX\_Init(void)

{

GPIO\_InitTypeDef GPIO\_InitStructure;

EXTI\_InitTypeDef EXTI\_InitStructure;

RCC\_APB2PeriphClockCmd( RCC\_APB2Periph\_GPIOA | RCC\_APB2Periph\_AFIO , ENABLE);

GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_4|GPIO\_Pin\_5|GPIO\_Pin\_6|GPIO\_Pin\_7;

GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;

GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_IPU; 1）//初始化 IO 口为输入。

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

GPIO\_EXTILineConfig(GPIO\_PortSourceGPIOA, GPIO\_PinSource4|GPIO\_PinSource5|GPIO\_PinSource6|GPIO\_PinSource7);

EXTI\_InitStructure.EXTI\_Line = EXTI\_Line4|EXTI\_Line5|EXTI\_Line6|EXTI\_Line7;

EXTI\_InitStructure.EXTI\_Mode = EXTI\_Mode\_Interrupt;

EXTI\_InitStructure.EXTI\_Trigger = EXTI\_Trigger\_Falling;

EXTI\_InitStructure.EXTI\_LineCmd = ENABLE;

EXTI\_Init(&EXTI\_InitStructure);

MY\_NVIC\_Init(2,2,EXTI4\_IRQChannel,2); //抢占2，子优先级2，组2

MY\_NVIC\_Init(2,2,EXTI9\_5\_IRQChannel,2); //抢占2，子优先级2，组2

}