本章教程通过PA0引脚输入信号产生中断控制运营中断服务函数，并将程序运行状态通过串口调试助手打印显示，具体如下：
当无中断产生，串口调试助手打印显示：Run at main；
当有中断产生，串口调试助手打印显示：Run at EXTI。

# 1、EXTI简介及相关函数介绍

EXTI（External interrupt/event controller）—外部中断/事件控制器，可分为产生中断和产生事件两大功能。每个中断/事件线都对应一个边沿检测器，可实现输入信号的上升沿检测和下降沿检测。EXTI可对每个中断/事件线进行单独配置，可单独配置为中断或者事件，以及相应触发事件的属性。

EXTI可分为产生中断和产生事件两大功能。产生中断线路目的是把输入信号输入到NVIC，从而执行中断服务函数，实现相应功能，属于软件级；产生事件线路目的是传输脉冲信号给对应外设使用，为电路级别信号传输，属于硬件级。

CH32V103外部中断的触发源既可以是软件中断（SWIEVR），也可以是外部中断通道。当触发源为外部中断通道时，其输入信号会经过边沿检测电路进行筛选。只要软件中断和外部中断二者信号产生一个，即可通过或门电路并输出给事件使能和中断使能两个与门电路，只要有中断被使能或者事件被使能，就会产生中断或者事件。关于CH32V103 EXTI具体说明，可参考CH32V103应用手册。

外部中断库函数初始化结构体及相关声明定义在ch32v10x_exti.h文件中，初始化库函数定义在ch32v10x_exti.c文件中，进行外部中断程序编程需结合上述两个文件使用，其主要内容介绍如下：

```
1. void EXTI_DeInit(void);
2. void EXTI_Init(EXTI_InitTypeDef* EXTI_InitStruct);
3. void EXTI_StructInit(EXTI_InitTypeDef* EXTI_InitStruct);
4. void EXTI_GenerateSWInterrupt(uint32_t EXTI_Line);
5. FlagStatus EXTI_GetFlagStatus(uint32_t EXTI_Line);
6. void EXTI_ClearFlag(uint32_t EXTI_Line);
7. ITStatus EXTI_GetITStatus(uint32_t EXTI_Line);
8. void EXTI_ClearITPendingBit(uint32_t EXTI_Line);
```

复制代码

1.1、void EXTI_DeInit(void)
功 能：将EXTI外围寄存器初始化为其默认重置值。
输 入：无。

1.2、void EXTI_Init(EXTI_InitTypeDef* EXTI_InitStruct)
功 能：根据EXTI_InitStruct中指定的参数初始化EXTI外围设备。
输 入：EXTI_InitStruct：指向EXTI_InitTypeDef结构的指针。

1.3、void EXTI_StructInit(EXTI_InitTypeDef* EXTI_InitStruct)
功 能：用重置值填充每个EXTI_InitStruct成员。
输 入：EXTI_InitStruct：指向EXTI_InitTypeDef结构的指针。

1.4、void EXTI_GenerateSWInterrupt(uint32_t EXTI_Line)
功 能：生成软件中断。
输 入：EXTI_Line：指定要启用或禁用的EXTI行。

1.5、FlagStatus EXTI_GetFlagStatus(uint32_t EXTI_Line)
功 能：检查是否设置了指定的EXTI行标志。
输 入：EXTI_Line：指定要启用或禁用的EXTI行。

1.6、void EXTI_ClearFlag(uint32_t EXTI_Line)
功 能：清除EXTI的行挂起标志。
输 入：EXTI_Line：指定要启用或禁用的EXTI行。

1.7、ITStatus EXTI_GetITStatus(uint32_t EXTI_Line)
功 能：检查指定的EXTI行是否被断言。
输 入：EXTI_Line：指定要启用或禁用的EXTI行。

1.8、ITStatus EXTI_GetITStatus(uint32_t EXTI_Line)
功 能：清除EXTI的行挂起位
输 入：EXTI_Line：指定要启用或禁用的EXTI行。

在进行外部中断程序编写时，若需用到上述函数，直接在程序中进行调用即可。

## 2、硬件设计

本教程通过PA0引脚接地产生中断从而执行中断服务函数，连接方式如下：

- 用杜邦线将PA0引脚与开发板GND引脚连接，产生中断之后拔下。

## 3、软件设计

外部中断使用步骤通常如下：

- 配置GPIO操作；
- 配置对应的外部中断通道的中断使能位；
- 配置触发沿，选择相应触发方式；
- 在内核NVIC中配置EXTI中断，以保证其可以正确响应

根据上述步骤，编写相关程序，具体如下：
exti.h文件

```
#ifndef __EXTI_H
#define __EXTI_H

#include "ch32v10x_conf.h"

void EXTI0_INT_INIT(void);

#endif

```

复制代码
exti.h文件用于保存exti相关函数的声明；
exti.c文件

```
#include "exti.h"

void EXTI0_IRQHandler(void) __attribute__((interrupt("WCH-Interrupt-fast")));

void EXTI0_INT_INIT(void)
{
  GPIO_InitTypeDef  GPIO_InitStructure;
  EXTI_InitTypeDef EXTI_InitStructure;
  NVIC_InitTypeDef NVIC_InitStructure;

  RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO|RCC_APB2Periph_GPIOA,ENABLE); //使能复用功能时钟和GPIOA时钟

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;                 //配置GPIO引脚
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;             //配置GPIO上拉输入模式
  GPIO_Init(GPIOA, &GPIO_InitStructure);                    //初始化GPIOA

   /* GPIOA ----> EXTI_Line0 */
  GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource0); //指定中断/事件线的输入源，实际上是设定外部中断配置寄存器AFIO_EXTICRx的值，此处为PA0
  EXTI_InitStructure.EXTI_Line=EXTI_Line0;                  //EXTI中断/事件线选择，此处选择EXTI_Line0
  EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;       //EXTI模式选择，此处选择为产生中断模式
  EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;   //EXTI边沿触发事件，此处选择为下降沿触发
  EXTI_InitStructure.EXTI_LineCmd = ENABLE;                 //使能EXTI线
  EXTI_Init(&EXTI_InitStructure);

  NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;          //使能EXTI0中断通道
  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1; //设置抢占优先级为1
```

```
27.   NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;        //设置子优先级为2
28.   NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;           //使能外部中断通道
29.   NVIC_Init(&NVIC_InitStructure);                           //中断优先级分组初始化
30. }
31.
32. void EXTI0_IRQHandler(void)
33. {
34.   if(EXTI_GetITStatus(EXTI_Line0)==SET)  //EXTI_GetITStatus用来获取中断标志位状态，如果EXTI线产生中断则返回SET，否则返回RESET
35.   {
36.     Delay_Ms(1000);
37.     printf("Run at EXTI\r\n");
38.     EXTI_ClearITPendingBit(EXTI_Line0);  //清除中断标志位
39.   }
40. }
```

复制代码

exti.c文件是外部中断的配置程序，对外部中断进行具体配置，其具体配置流程如下：

1、使能复用功能时钟和GPIO时钟；

```
1. RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO|RCC_APB2Periph_GPIOA,ENABLE);  //使能复用功能时钟和GPIOA时钟
```

复制代码

2、初始化产生中断的GPIO口

```
1. GPIO_InitStructure.GPIO_Pin = GPIO_Pin_0;                //配置GPIO引脚
2. GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;            //配置GPIO上拉输入模式
3. GPIO_Init(GPIOA, &GPIO_InitStructure);                   //初始化GPIOA
```

复制代码

3、初始化EXTI配置；

```
1.   /* GPIOA ----> EXTI_Line0 */
2. GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource0); //指定中断/事件线的输入源，实际上是设定外部中断配置寄存器AFIO_EXTICRx的值，此处为PA0
3. EXTI_InitStructure.EXTI_Line=EXTI_Line0;                 //EXTI中断/事件线选择，此处选择EXTI_Line0
4. EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;      //EXTI模式选择，此处选择为产生中断模式
5. EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;  //EXTI边沿触发事件，此处选择为下降沿触发
6. EXTI_InitStructure.EXTI_LineCmd = ENABLE;                //使能EXTI线
7. EXTI_Init(&EXTI_InitStructure);
```

复制代码

4、初始化NVIC配置

```
1.   NVIC_InitStructure.NVIC_IRQChannel = EXTI0_IRQn;          //使能EXTI0中断通道
```

```
2.  NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 1; //设置抢占优先级为1
3.  NVIC_InitStructure.NVIC_IRQChannelSubPriority = 2;        //设置子优先级为2
4.  NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;           //使能外部中断通道
5.  NVIC_Init(&NVIC_InitStructure);                           //中断优先级分组初始化
```

复制代码

5、编写中断服务函数

```
1. void EXTI0_IRQHandler(void)
2. {
3.   if(EXTI_GetITStatus(EXTI_Line0)==SET)  //EXTI_GetITStatus用来获取中断标志位状态，如果EXTI线产生中断则返回SET，否则返回RESET
4.    {
5.     Delay_Ms(1000);
6.     printf("Run at EXTI\r\n");
7.     EXTI_ClearITPendingBit(EXTI_Line0);  //清除中断标志位
8.    }
9. }
```

复制代码

中断服务函数中，EXTI_GetITStatus用来获取中断标志位状态，如果EXTI线产生中断则返回SET，否则返回RESET。当产生中断以后，串口打印输出：Run at EXTI，最后清除中断标志位。

main.c文件

```
1. int main(void)
2. {
3.  NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);
4.  Delay_Init();
5.  USART_Printf_Init(115200);
6.
7.  printf("SystemClk:%d\r\n",SystemCoreClock);
8.  printf("EXTI0 Test\r\n");
9.
10.  EXTI0_INT_INIT();
11.  while(1)
12.   {
13.    Delay_Ms(1000);
14.    printf("Run at main\r\n");
15.   }
16. }
```

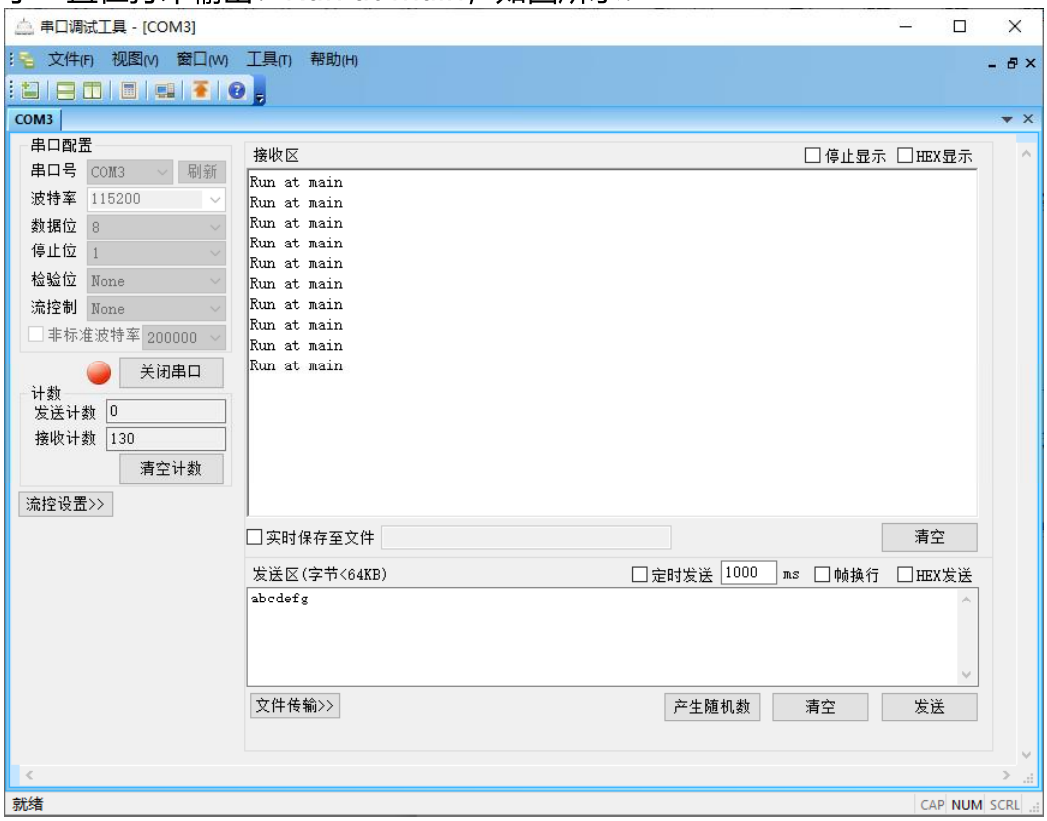
复制代码

main.c文件主要包含NVIC中断优先级分组以及EXTI中断初始化以及while循环中输出打印，提示当前运行状态。

# 4、下载验证

将编译好的程序下载到开发板并复位，打开串口调试助手，可看到串口调试助

手一直在打印输出：Run at main，如图所示：




因为程序中表示下降沿触发中断，因此用杜邦线将PA0引脚接地，可看到串口调试助手会打印出：Run at EXTI，说明程序进入中断并执行一次中断服务函数，如图所示：