

实验三 外部中断编程——蜂鸣器实验

1656038 斯提凡

一、实验目的

1. 了解 STM32F103 芯片的中断机制
2. 掌握中断编程的方法

二、实验基本要求

1. 认真阅读和掌握本实验的程序。
2. 按实验要求编写程序并调试运行。
3. 保存与记录实验结果，并进行分析总结。

三、实验要点

实验环境

硬件：PC 机一台，P4 2.06CPU/40GHD/512M RAM 以上配置，STM32F103 开发板一套。

软件：PC 机操作系统为 Windows7，程序开发调试环境为 Keil C。

四、实验学时数

本次实验共 2 学时。

五、实验内容

编写代码实现按键控制蜂鸣器（或小灯）开关。

六、实验步骤

1. 新建工程目录：打开 PC 机，在 D 盘新建目录“D:\EXTint”
2. 拷贝固件函数库：将光盘中示例代码工程目录下的“library”目录拷贝到新建的工程目录下。
3. 新建工程：打开 Keil C，Project 菜单选择 New uVision Project，新建一个工程。将工程命名为“EXTint”，并保存在 D 盘新建的目录中。
4. 选择芯片型号：在芯片型号数据库选择弹出式菜单选项中，选择通用 CPU 数据库，点击“OK”。在弹出的目标设备型号选择菜单中，选中“STMicroelectronics”公司，在出现的下拉式列表中选择“STM32F103VC”，点击“OK”。在弹出的是否添加启动文件对话框中选择“否”不添加。
5. 添加启动文件：右键点击项目资源管理器中的“Target1Source Group1”文件夹，在弹出的菜单中选择“Add File to Group' Source Group 1'”将启动文件“stm32f10x_vector.s”，“cortexm3_macro.s”添加至项目。
6. 新建用户程序：点击主菜单“File - ->New”，新建一个文件，命名为“EXTint.c”

保存至项目文件夹。重复第 5 步，将用户程序文件添加到工程中。

7. 添加 Include 语句：通过“Include”语句，将以上代码使用到的库件函数头文件添加到文件。a) #include"./library/src/stm32f10x_it.h" b)#include"./library/src/stm32f10x_nvic.h" c)#include"./library/src/stm32f10x_gpio.h"

d)#include"./library/src/stm32f10x_flash.h"
e)#include"./library/src/stm32f10x_rcc.h" f)#include"./library/src/
stm32f10x_exti.h"

g)#include "stdio.h"

h)#include "stm32f10x_lib.h"

8. 将以下库文件添加到工程： a) stm32f10x_it.c
b) stm32f10x_nvic.c c) stm32f10x_gpio.c d) stm32f10x_flash.
c e) stm32f10x_rcc.c f) stm32f10x_exti.c

9. 建立 Main 函数：输入 int main(void)，建立入口函数，并添加 while(1) 循环，使 main 函数不会退出。

10. 初始化时钟：定义“ErrorStatus”类结构体“ErrorStatus HSEStartUpStatus”，并在 main 里添加时钟系统初始化函数 RCC_Configuration();

11. 配置 GPIO 端口：添加代码，定义“GPIO_InitTypeDef”类结构体“GPIO_InitStructure”，将 GPIO 口 C 的第 9 号管脚配置为浮空输入模式，速度为 50M 并初始化端口。

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;  
GPIO_InitStructure.GPIO_Mode =  
GPIO_Mode_IN_FLOATING; GPIO_InitStructure.GPIO_Speed  
= GPIO_Speed_50MHz; GPIO_Init(GPIOC,  
&GPIO_InitStructure);
```

将 GPIO 口 B 的第 2 号管脚配置为推挽输出模式，速度为 50M 并初始化端口。

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;  
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;  
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;  
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

12. 配置 GPIO 端口为下降沿中断模式：添加代码，定义

“EXTI_InitTypeDef”类结构体“EXTI_InitStructure”，将 GPIO 口 C 的第 9 号管脚配置为下降沿中断模式，并初始化端口。GPIO_EXTILineConfig(GPIO_PortSourceGPIOC, GPIO_PinSource9); EXTI_InitStructure.EXTI_Line=EXTI_Line9;

```
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;  
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;  
EXTI_InitStructure.EXTI_LineCmd = ENABLE;  
EXTI_Init(&EXTI_InitStructure);
```

13. 配置 5 - 9 号中断线优先级：添加代码，定义“NVIC_InitTypeDef”类结构“NVIC_InitStruct”，将 5 - 9 号线的抢占中断优先级与响应优先级都设置为 0，并初始化中断。

```
NVIC_InitStruct.NVIC_IRQChannel =  
EXTI9_5_IRQChannel;  
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority  
= 0;  
NVIC_InitStruct.NVIC_IRQChannelSubPriority  
= 0;  
NVIC_InitStruct.NVIC_IRQChannelCmd  
= ENABLE; NVIC_Init(&NVIC_InitStruct);
```

14. 添加中断响应函数：打开“stm32f10x_it.c”文件，找到“void

EXTI9_5_IRQHandler(void) ” 函 数。在函 数中添加 中断响应 代码： `extern void delay_nms(unsigned long n);` (在文件开头定义外部引用函

```

数)
if(GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_2)==0)
GPIO_SetBits(GPIOB, GPIO_Pin_2);
else
GPIO_ResetBits(GPIOB,
GPIO_Pin_2); delay_nms(10);
EXTI_ClearITPendingBit( EXTI_Li
ne9) ;

```

15. 添加 delay_nms 函数：

```

void delay_nus(unsigned long n)
{
    unsigned long j;
    while(n--)
    {
        j=8;
        while(j--);
    }
}

void delay_nms(unsigned long n)
{
    while(n--)
        delay_nus(1100);
}

```

16. 连接硬件：取出开发板，用 J - Link 将开发板连接至 PC 机，并给开发板上电。

17. 跳线设置：将开发板的 Beep 跳线跳至 On，连接蜂鸣器。

18. 配置编译环境：点击项目 Option 菜单，在弹出式菜单中选择 Debug 菜单，配置 J - Llink。选择 Utilities 菜单配置 J - Llink 并选择 Flash 类型为“STM32F10x Med - density Flash”。

19. 编译并下载：编译程序，编译完成后点击下载按钮，将程序下载至开发板。

20. 察看结果：按动按键 PC9，察看蜂鸣器响应情况。

```

七、实验代码#include"./library/src/s
tm32f10x_it.h" #include"./library/sr
c/stm32f10x_nvic.h" #include"./libr
ary/src/stm32f10x_gpio.h" #includ
e"./library/src/stm32f10x_flash.h" #i
nclude"./library/src/stm32f10x_rcc.
h" #include"./library/src/stm32f10x_
exti.h" #include "stdio.h"
#include "stm32f10x_lib.h"
ErrorStatus

```

```

HSEStartUpStatus;
GPIO_InitTypeDef

```

```
GPIO_InitStructure;  
EXTI_InitTypeDef EXTI_InitStructure;  
NVIC_InitTypeDef NVIC_InitStruct;
```

```

void RCC_Configuration(void)    //时钟系统初始化函数
{
    RCC_DeInit();
    RCC_HSEConfig(RCC_HSE_
ON);
    HSEStartUpStatus =
    RCC_WaitForHSEStartUp(); if
    (HSEStartUpStatus == SUCCESS)
    {
        RCC_HCLKConfig(RCC_SYSCLK_Div1);
        RCC_PCLK2Config(RCC_HCLK_Div1);
        RCC_PCLK1Config(RCC_HCLK_Div2);
        FLASH_SetLatency(FLASH_Latency_2);
        FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_En
able); RCC_PLLConfig(RCC_PLLSource_HSE_Div1,
        RCC_PLLMul_9); RCC_PLLCmd(ENABLE);
        while (RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET);
        RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
        while (RCC_GetSYSCLKSource() != 0x08);
    }
    /* Enable GPIOA~E and AFIO clocks */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA |
        RCC_APB2Periph_GPIOB |
        RCC_APB2Periph_GPIOC | RCC_APB2Periph_GPIOD |
        RCC_APB2Periph_GPIOE | RCC_APB2Periph_AFIO, ENABLE);

    RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);
    /* TIM1 clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1,
        ENABLE);

    /* TIM2 clock enable */
    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2,
        ENABLE);

    /* ADC1 clock enable */
    RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,
        ENABLE);
}

void GPIO_Configuration()        //配置 GPIO 端口
{
    //将 GPIO 口 C 的第 9 号管脚配置为浮空输入，速度为 50M 并初始化端口
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Mode =
    GPIO_Mode_IN_FLOATING;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);
    //将 GPIO 口 B 的第 2 号管脚配置为推挽输出模式，速度为 50M 并初始化端

```

□

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);
//将 GPIO 口 C 的第 9 号管脚配置为下降沿中断模式，并初始化端
□ GPIO_EXTILineConfig(GPIO_PortSourceGPIOC,
GPIO_PinSource9); EXTI_InitStructure.EXTI_Line = EXTI_Line9;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);
```

```
//将 5-9 号线的抢占中断先优先级与响应优先级都设置为 0，并初始化中断
NVIC_InitStruct.NVIC_IRQChannel = EXTI9_5_IRQChannel;
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStruct.NVIC_IRQChannelSubPriority
y = 0;
NVIC_InitStruct.NVIC_IRQChannelCmd =
ENABLE; NVIC_Init(&NVIC_InitStruct);
void delay_nus(unsigned long n) //添加 delay_nms 函数
{
    unsigned long
    j; while (n--) {
        j = 8;
        while (j--);
    }
}
void delay_nms(unsigned long n) //延时 n ms
{
    while (n--) // 外部晶振：8M； PLL：9；
        8M*9=72MHz delay_nus(1100); // 1ms 延时补偿
}
```

```
int main(void) {

    RCC_Configuration();
    GPIO_Configuration();
    while (1) {
        delay_nms(500);
    }
}
```

```
}
```

中断处理程序：

八、进阶实验

1. 增加一组按钮与 LED 灯的配合（按钮控制 LED 灯亮和灭）。
2. 在中断响应程序中增加时延，并给两个按钮设置不同的中断权限，看两组按钮间的中断响应关系。注：NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1); 语句，用于将 4 位优先级设置分组，从而实现中断嵌套。

代码如下：

```
#include"./library/src/stm32f10x_it.h" #include"./library/src/stm32f10x_nvic.h" #include"./library/src/stm32f10x_gpio.h" #include"./library/src/stm32f10x_flash.h" #include"./library/src/stm32f10x_rcc.h" #include"./library/src/stm32f10x_exti.h" #include "stdio.h" #include "stm32f10x_lib.h"
```

ErrorStatus

HSEStartUpStatus;
GPIO_InitTypeDef

GPIO_InitStructure;
EXTI_InitTypeDef EXTI_InitStructure;
NVIC_InitTypeDef NVIC_InitStruct;

```
void RCC_Configuration(void) //时钟系统初始化函数
{
    RCC_DeInit();
    RCC_HSEConfig(RCC_HSE_ON);
    HSEStartUpStatus = RCC_WaitForHSEStartUp();
    if(HSEStartUpStatus == SUCCESS)
    {
        RCC_HCLKConfig(RCC_SYSCLK_Div1);
        RCC_PCLK2Config(RCC_HCLK_Div1);
        RCC_PCLK1Config(RCC_HCLK_Div2);
        FLASH_SetLatency(FLASH_Latency_2);
        FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
        RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
        RCC_PLLCmd(ENABLE);
        while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET) ;
    }
}
```



```

    RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
    while(RCC_GetSYSCLKSource() != 0x08);
}

/* Enable GPIOA~E and AFIO clocks */ RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|RCC_APB2Periph_GPIOB
|RCC_APB2Periph_GPIOC|RCC_APB2Periph_GPIOD|RCC_APB2Periph_GPIOE|
RCC_APB2Periph_AFIO, ENABLE);

RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1, ENABLE);

/* TIM1 clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1,
ENABLE);

/* TIM2 clock enable */
RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2,
ENABLE);

/* ADC1 clock enable */
RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,
ENABLE);
}

void GPIO_Configuration()      //配置 GPIO 端口
{
    //将 GPIO 口 C 的第 9 号管脚配置为浮空输入，速度为 50M 并初始化端口
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
    GPIO_InitStructure.GPIO_Mode =
    GPIO_Mode_IN_FLOATING;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    // 进阶灯按钮 C10
    GPIO_InitStructure.GPIO_Pin =
    GPIO_Pin_10;
    GPIO_InitStructure.GPIO_Mode =
    GPIO_Mode_IN_FLOATING;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOC, &GPIO_InitStructure);

    //将 GPIO 口 B 的第 2 号管脚配置为推挽输出模式，速度为 50M 并初始化端
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
    GPIO_Init(GPIOB, &GPIO_InitStructure);

    //GPIO C12 led 灯
    GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;

```

```

GPIO_InitStructure.GPIO_Mode                =
GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed              =
GPIO_Speed_50MHz;          GPIO_Init(GPIOC,
&GPIO_InitStructure);

/* #ifdef VECT_TAB_RAM
    NVIC_SetVectorTable(NVIC_VectTab_RAM
,0X0); #else
    NVIC_SetVectorTable(NVIC_VectTab_FLASH,0
X0); #endif */
    NVIC_PriorityGroupConfig(NVIC_PriorityGroup_1);

//将 GPIO 口 C 的第 9 号管脚配置为下降沿中断模式，并初始化端
口 GPIO_EXTI_LineConfig(GPIO_PortSourceGPIOC,
GPIO_PinSource9); EXTI_InitStructure.EXTI_Line = EXTI_Line9;
EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_InitStructure.EXTI_Trigger = EXTI_Trigger_Falling;
EXTI_InitStructure.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_InitStructure);

//将 5-9 号线的抢占中断先优先级与响应优先级都设置为 0，并初始化中断
NVIC_InitStruct.NVIC_IRQChannel = EXTI9_5_IRQChannel;
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority =0;
NVIC_InitStruct.NVIC_IRQChannelSubPriority
y                =                0;
NVIC_InitStruct.NVIC_IRQChannelCmd
=ENABLE; NVIC_Init(&NVIC_InitStruct);

GPIO_EXTI_LineConfig(GPIO_PortSourceGPIOC,
GPIO_PinSource10); EXTI_InitStructure.EXTI_Line =
EXTI_Line10; EXTI_InitStructure.EXTI_Mode =
EXTI_Mode_Interrupt; EXTI_InitStructure.EXTI_Trigger =
EXTI_Trigger_Falling; EXTI_InitStructure.EXTI_LineCmd =
ENABLE; EXTI_Init(&EXTI_InitStructure);

NVIC_InitStruct.NVIC_IRQChannel =
EXTI15_10_IRQChannel;
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority
=1;
NVIC_InitStruct.NVIC_IRQChannelSubPriority
y                =                0;
NVIC_InitStruct.NVIC_IRQChannelCmd
=ENABLE; NVIC_Init(&NVIC_InitStruct);

}

```

void delay_nus(unsigned long n)//添加 delay_nms 函数

```
{
    unsigned long
    j; while(n--){
        j=8;
        while(j--);
    }
}
```

void delay_nms(unsigned long n) //延时 n ms

```
{
    while(n--) // 外部晶振：8M； PLL：9；
        8M*9=72MHz delay_nus(1100); // 1ms 延时补偿
}
```

int main(void){

```
    RCC_Configuration();
    GPIO_Configuration();
    while(1)
    { delay_nms(500);
    }
```

}

中断处理程序代码：

void EXTI9_5_IRQHandler(void)

```
{
    extern void delay_nms(unsigned long n);
    if(GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_2)==0)
        GPIO_SetBits(GPIOB, GPIO_Pin_2);
    else
        GPIO_ResetBits(GPIOB, GPIO_Pin_2);
    delay_nms(5000);
    EXTI_ClearITPendingBit( EXTI_Line9 );
    //EXTI_ClearITPendingBit( EXTI_Line9 );//中断结束时清中断标志位
    //EXTI_ClearITPendingBit( EXTI_Line5 );//中断结束时清中断标志位
}
```

void EXTI15_10_IRQHandler(void)

```
{
    extern void delay_nms(unsigned long n);
    if(GPIO_ReadOutputDataBit(GPIOC, GPIO_Pin_12)==0)
        GPIO_SetBits(GPIOC, GPIO_Pin_12);
    else
        GPIO_ResetBits(GPIOC, GPIO_Pin_12);
}
```

```
delay_nms(5000);  
EXTI_ClearITPendingBit( EXTI_Line10);
```

九、实验现象

- 1, 按动按键 PC9, 蜂鸣器响起, 再按一下, 蜂鸣器不响。
- 2, 进阶实验: 增加延时后, 中断响应处理时间增加; 改变优先级顺序后, 先响应优先级高的中断程序。