cortex_m3_stm32嵌入式学习笔记（四）：外部中断实验

原创

_lyl 于 2015-01-20 15:46:15 发布 4278 收藏 6
分类专栏： 嵌入式_stm32

本章学习将STM32的IO口作为外部中断输入（实现和按键扫描一样的功能）

STM32 的每个 IO 都可以作为外部中断的中断输入口，这点也是 STM32 的强大之处。 STM32F103 的中断控制器支持 19 个外部中断/事件请求。每个中断设有状位，每个中断/事件都有独立的触发和屏蔽设置。 STM32F103 的19 个外部中断为：

线 0~15：对应外部 IO 口的输入中断。（本章只学习这一种）

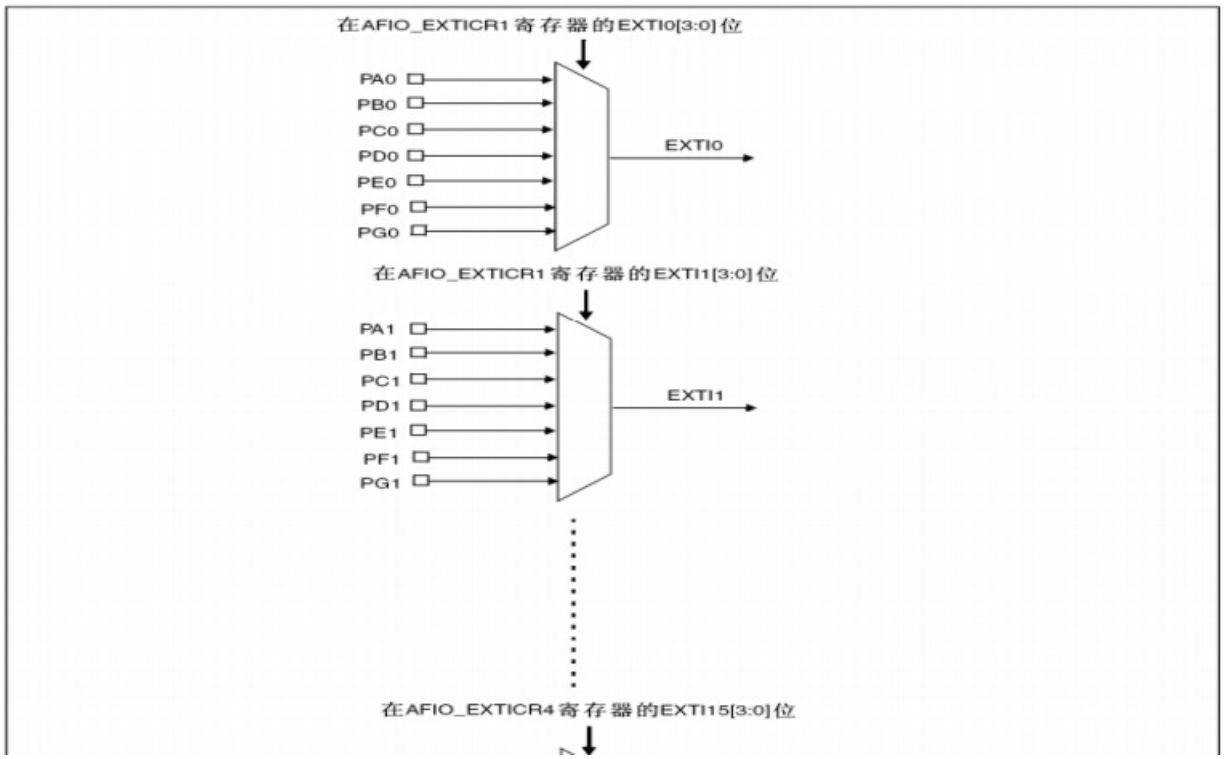
线 16：连接到 PVD 输出。

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 口使用的中断线只有 16 个，但是 STM32 的 IO 口却远远不止 16 个，那么 STM32 是怎么把 16 个中断线和 IO 口一一对应起来的呢？于是 STM32 就这样设计， GPIO 的管教 GPIOx.0~GPIOx.15(x=A,B,C,D,E， F,G)分别对应中断线 0~15。（原文是15-0 我感觉它写错了QAQ）

对应图如下：



有了映射关系 很明显 KEY0（PC5）对应中断线5（EXTI5）KEY1（PA15）对应中断线15（EXTI15） WK_UP（PA0）对应中断线0（EXTI0）

接下来就是配置中断了，编写exti.c 添加到工程

```
#include "exti.h"
#include "led.h"
#include "key.h"
#include "delay.h"
#include "usart.h"
void EXTIx_Init(void)
{
 EXTI_InitTypeDef EXTI_ist;
  NVIC_InitTypeDef NVIC_ist;
 RCC_APB2PeriphClockCmd(RCC_APB2Periph_AFIO,ENABLE);//外部中断，需要使能 AFIO 时钟
 KEY_Init();
 //GPIOC.5 中断线以及中断初始化配置
 GPIO_EXTILineConfig(GPIO_PortSourceGPIOC,GPIO_PinSource5);
 EXTI_ist.EXTI_Line=EXTI_Line5;
  EXTI_ist.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_ist.EXTI_Trigger = EXTI_Trigger_Falling;//下降沿触发
  EXTI_ist.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_ist);
 //GPIOA.15 中断线以及中断初始化配置
 GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource15);
 EXTI_ist.EXTI_Line=EXTI_Line15;
  EXTI_ist.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_ist.EXTI_Trigger = EXTI_Trigger_Falling;//下降沿触发（1->0）
  EXTI_ist.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_ist);
 //GPIOA.0 中断线以及中断初始化配置
 GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource0);
 EXTI_ist.EXTI_Line=EXTI_Line0;
  EXTI_ist.EXTI_Mode = EXTI_Mode_Interrupt;
  EXTI_ist.EXTI_Trigger = EXTI_Trigger_Rising;//上升沿触发（0->1）
  EXTI_ist.EXTI_LineCmd = ENABLE;
  EXTI_Init(&EXTI_ist);

 //配置NVIC
 NVIC_ist.NVIC_IRQChannel = EXTI0_IRQn;//使能按键所在的外部中断通道
 NVIC_ist.NVIC_IRQChannelPreemptionPriority = 0x02; //抢占优先级 2
  NVIC_ist.NVIC_IRQChannelSubPriority = 0x01; //子优先级 1
  NVIC_ist.NVIC_IRQChannelCmd = ENABLE; //使能外部中断通道
  NVIC_Init(&NVIC_ist);

 NVIC_ist.NVIC_IRQChannel = EXTI9_5_IRQn;//使能按键所在的外部中断通道
 //NVIC_ist.NVIC_IRQChannelPreemptionPriority = 0x02; //抢占优先级 2
  //NVIC_ist.NVIC_IRQChannelSubPriority = 0x01; //子优先级 1
  //NVIC_ist.NVIC_IRQChannelCmd = ENABLE; //使能外部中断通道
  NVIC_Init(&NVIC_ist);

 NVIC_ist.NVIC_IRQChannel = EXTI15_10_IRQn;//使能按键所在的外部中断通道
 //NVIC_ist.NVIC_IRQChannelPreemptionPriority = 0x02; //抢占优先级 2
  //NVIC_ist.NVIC_IRQChannelSubPriority = 0x01; //子优先级 1
  //NVIC_ist.NVIC_IRQChannelCmd = ENABLE; //使能外部中断通道
  NVIC_Init(&NVIC_ist);
}
void EXTI0_IRQHandler(void)//中断服务函数
{
 delay_ms(10); //消抖
```

```
 if(WK_UP==1)
 {
  LED0=!LED0;
  LED1=!LED1;
 }
 EXTI_ClearITPendingBit(EXTI_Line0); //清除 EXTI0 线路挂起位
}
void EXTI9_5_IRQHandler(void)
{
 delay_ms(10); //消抖
 if(KEY0==0)
 {
  LED0=!LED0;
 }
 EXTI_ClearITPendingBit(EXTI_Line5); //清除 LINE5 上的中断标志位
}
void EXTI15_10_IRQHandler(void)
{
 delay_ms(10); //消抖
 if(KEY1==0)
 {
  LED1=!LED1;
 }
 EXTI_ClearITPendingBit(EXTI_Line15); //清除 LINE15 上的中断标志位
}
```

这里面有4个函数，一个初始化中断的函数，3个中断服务函数（就是中断了之后你让他去干什么）

初始化中比较麻烦

因为3个按键用到了3条中断线，所以3条中断线都要初始化，但3条线的初始化很相似，我们就以 EXTI0 为例吧 将代码抠出来

```
GPIO_EXTILineConfig(GPIO_PortSourceGPIOA,GPIO_PinSource0);
EXTI_ist.EXTI_Line=EXTI_Line0;
EXTI_ist.EXTI_Mode = EXTI_Mode_Interrupt;
EXTI_ist.EXTI_Trigger = EXTI_Trigger_Rising;//上升沿触发（0->1）
EXTI_ist.EXTI_LineCmd = ENABLE;
EXTI_Init(&EXTI_ist);
```

第一句是 将 GPIO 端口与中断线映射起来，然后选中断线，选择中断模式（还有一种模式为事件模式，去论坛搜了一下没太明白。。算了学到再说吧）

接下来选择触发模式，一般有上升沿触发（电平从0->1），下降沿触发（电平从1->0），和任意触发模式。根据按键的有效性（因为WK_UP是高电平有效，所以选择上升沿触发，而另外两个按键选择下降沿触发模式），最后使能外部中断通道（不太懂。。强记吧先）

在接下来就要配置NVIC中断优先级，NVIC又是一个陌生词，度娘了一下：提供中断控制器，用于总体管理异常，称之为“嵌套向量中断控制器:Nested Vectored Interrupt Controller (NVIC)”。

就是用来管理中断优先级的

中断优先级有两项：抢占优先级和响应优先级，手册上原话：

第一，如果两个中断的抢占优先级和响应优先级都是一样的话，则看哪个中断先发生就先执行；第二，高优先级的抢占优先级是可以打断正在进行的低抢占优先级
中断的。而抢占优先级相同的中断，高优先级的响应优先级不可以打断低响应优先级的中断。
意思是好像抢占优先级的地位比响应优先级的地位高？

但我没实现优先级的实验，因为好像是只有在中断同时进行的时候才会用到优先级，而次实验中断是靠按键触发的，也就是说要两个按键同时按下。。这几乎是不可能的

中断服务函数就相对简单了 注意最后清除中短线上中断标志位那条语句，一定要写。。没有的后果是你按下键后他可能不灵敏，就是有可能需要按多次才能看到应有的效果

exti.h 基本没东西

```
#ifndef  _EXTI_H
#define  _EXIT_H
#include "sys.h"
void EXTIx_Init(void);
#endif
```

主函数就是调用一堆初始化的函数。。

```
#include "led.h"
#include "key.h"
#include "sys.h"
#include "delay.h"
#include "exti.h"
#include "usart.h"
void init(void)
{
 delay_init();
 LED_Init();
 EXTIx_Init();
 NVIC_Configuration();
 uart_init(9600);

}
int main(void)
{
 init();
 LED0=0;
 while(1)
 {
  //检测程序是否正常运行，没什么用 去掉也行
  printf("OK\n");
  delay_ms(100);
 }
}
```

注意。。有一些相关的文件没贴。。那都是之前写过的。。像led.c key.c 什么的