中断

> 外部引脚中断

这里使用LPC824来说明怎么配置外部引脚中断

1. 寄存器功能

在LPC824的引脚中断控制中，一共使用了13个寄存器。下面是这些寄存器组所对应的结构体形式（位于头文件 `LPC82x.h` 中）。

```
typedef struct {
  __IO uint32_t  ISEL;
  __IO uint32_t  IENR;
  __O  uint32_t  SIENR;
  __O  uint32_t  CIENR;
  __IO uint32_t  IENF;
  __O  uint32_t  SIENF;
  __O  uint32_t  CIENF;
  __IO uint32_t  RISE;
  __IO uint32_t  FALL;
  __IO uint32_t  IST;
  __IO uint32_t  PMCTRL;
  __IO uint32_t  PMSRC;
  __IO uint32_t  PMCFG;
} LPC_PIN_INT_Type;
```

ISEL 引脚中断模式寄存器

> 指示PMODE

1. 7:0 为8个外部中断引脚的模式选择位，
    1. 置位0，表示中断设置为边沿触发；
    2. 置位1，表示中断设置为电平触发；
    3. 默认为0，边沿触发；
2. 31:8 为保留位。

IENR 引脚电平中断或者上升沿中断使能寄存器

> 符号是ENRL enable rising edge or level interrupt

1. 7:0 为对应引脚的中断使能信号
    1. 置位0，对应外部中断金庸上升或者电平中断；
    2. 置位1，使能上升沿或者电平中断；

    3. 默认为0，禁用；
2. 31:8 保留位；

SIENR 引脚电平中断或者上升沿中断置位寄存器

> 是对 IENR 的单独置位操作，当按byte操作时，直接使用IENR就可以；
> 符号时SETENRL set

CIENR clear IENR

> 为了对IENR寄存器进行单独的清零操作而设立的，当按byte操作时，直接使用IENR就可以；

1. 7:0 写入1时会清零IENR中对应的位，从而禁用
2. 置位0无影响；
3. 31:8 保留位；

IENF 引脚电平中断或者下降沿中断使能寄存器

SIENF 和 CIENF 类似

RISE 引脚中断上升沿寄存器

> 上升沿检测，Rising edge DETect，每个bit对应每个引脚

1. 7:0 RDET
    1. Read 0 ： 对应引脚没有检测到上升沿；
    2. Write 0：无影响；
    3. Read 1：自从reset或者上次该位写入1起，对应的引脚上检测到了上升沿；
    4. Write 1：清除对应引脚的上升沿检测标记，为下一次上升沿检测做准备；
2. 31:8 保留位；

FALL 引脚中断下降沿寄存器

1. 7:0
    1. 在第0到7位中写入1会清除相应引脚的下降沿检测标记，从而为下一次下降沿检测作准备；
    2. 写0时无影响。
    3. 若在相应的位上读取到1，则表示自复位或上一次向该位写1清除起，对应的引脚上检测到了下降沿；
    4. 读取到0，则表示对应引脚上未检测到下降沿。
2. 第8到31位为保留位。

IST 引脚中断状态寄存器

1. 7:0
    1. 写入1时，对于边沿触发型中断，将会清除对应引脚的上升和下降沿检测，对于电平触发型中断，将会切换对应引脚上的有效电平;
    2. 写0时无影响。
    3. 若在相应的位上读取到1，则表示对应的中断引脚上有正在请求的外部中断，
    4. 读取到0，则表示对应的中断引脚上无正在请求的外部中断。
2. 31:8 保留位

注意，电平触发的中断会由硬件在有效电平消失时自动清除标志，**并没有软件清除电平中断标志的操作**。

其他 PINTSEL 引脚中断选择寄存器

> 以上10个寄存器就是LPC824中与外部引脚中断相关寄存器，其实除了这10个寄存器以外，还有1个名为 `PINTSEL` 的寄存器也与外部引脚中断密切相关，只不过它不在上述寄存器组中，而是位于 `SYSCON` 模块中。下表就给出了引脚中断选择寄存器 `PINTSEL` 的全部位结构，这样的寄存器一共有8个。

1. 5:0 用来选择外部中断的引脚编号，0~28可选；
2. 31:6 保留位

注意，PINTSEL寄存器一共有8个，**也即LPC824的外部引脚中断同时最多只能使用8个**，但每一个外部中断都可以在全部PIO0_0到PIO0_28端口中选择引脚。 相当于PINTSEL0~PINTSEL7是8个外部中断源，具体哪个中断源使用哪个引脚，是根据PINTSEL寄存器的配置来确定的。

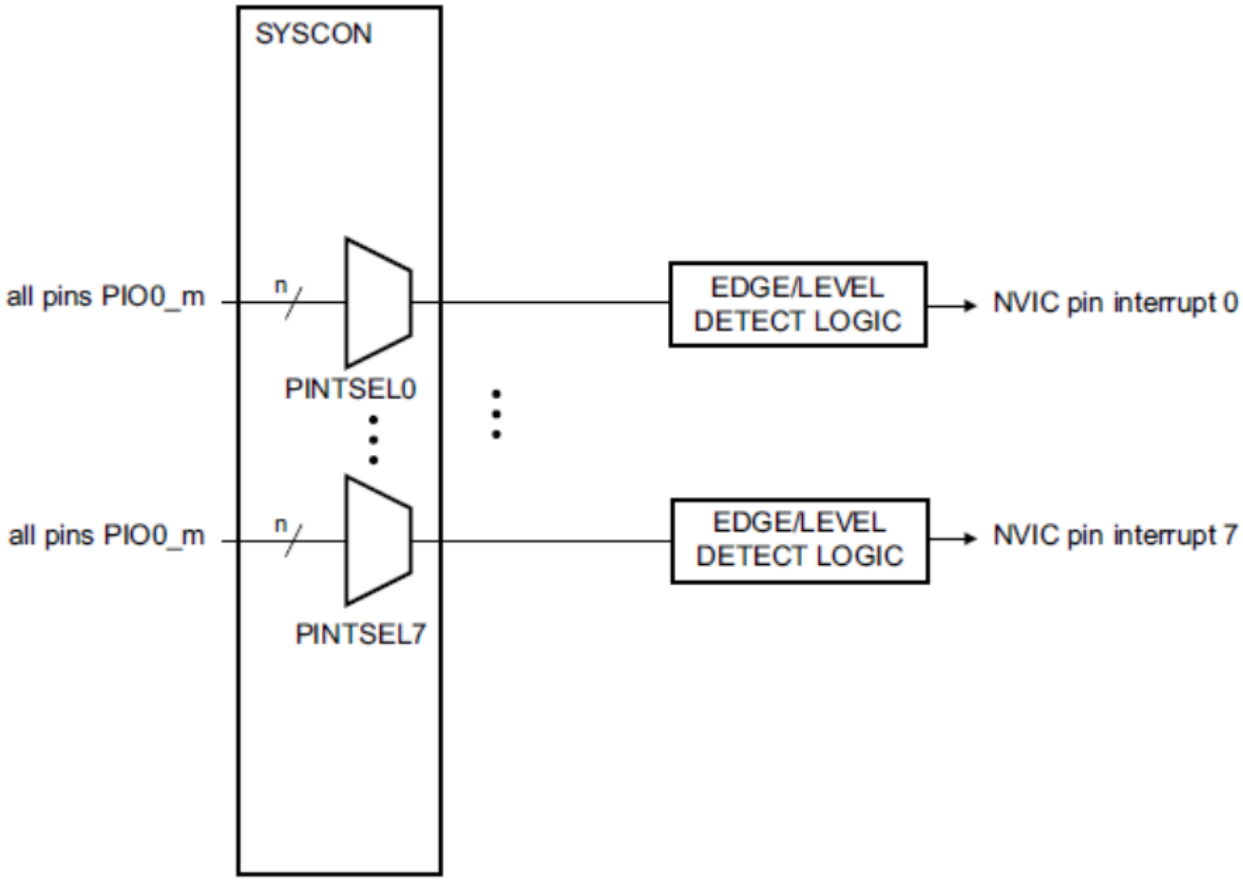
2. 具体配置步骤

下面就来分解一下**把某个引脚配置为外部中断模式的具体步骤**:

1. 确定要配置为中断模式的引脚，然后在SYSCON模块的PINTSEL寄存器中进行选择设置，一共可以配置8个外部中断引脚。例如，执行 `LPC_SYSCON->PINTSEL0 |= 0x01;` 语句后，就把PIO0_1引脚设置为了外部中断引脚。
2. 确定是电平触发还是边沿触发，通过ISEL寄存器进行选择配置。例如，执行 `LPC_PIN_INT->ISEL &= ~0x01;` 语句后，就把PIO0_1引脚设置为边沿触发方式(其实默认就是边沿触发方式，此句也可不写)。
3. 若上一步配置成电平触发，则需要确定是低电平触发还是高电平触发，若是边沿触发，则需要确定是上升沿触发还是下降沿触发，通过IENR或IENF寄存器进行执行配置。例如，执行 `LPC_PIN_INT->IENR |= 0x01;` 语句后，就把PIO0_1引脚设置为上升沿触发方式；执行`LPC_PIN_INT->IENF |= 0x01;` 语句后，就把PIO0_1引脚设置为下降沿触发方式。
4. 在第3步中，还可以通过访问SIENR和CIENR寄存器来更改IENR寄存器中的某一位，通过访问SIENF和CIENF寄存器来更改IENF寄存器中的某一位。SIENR、CIENR和SIENF、CIENF这四个寄存器其实是IENR和IENF寄存器的伴侣寄存器，用来优化位操作，以避免对IENR和IENF寄存器直接执行“读—改—写”的操作，提高效率。

5. 使能NVIC中的相关外部中断。例如，执行 `NVIC_EnableIRQ(PIN_INT0_IRQn);` 语句后，就使能了PIO0_1上的外部引脚中断。
6. 在中断服务程序中，需要清除原有的外部中断标记，以保证下一次外部中断顺利触发，通过访问RISE寄存器来清除上升沿中断标记，通过访问FALL寄存器来清除下降沿标记。例如，执行 `LPC_PIN_INT->RISE |= 0x01;` 语句后，PIO0_1原来的上升沿中断标记就被清除了。执行 `LPC_PIN_INT->FALL |= 0x01;` 语句后，PIO0_1原来的下降沿中断标记就被清除了。
7. 在第6步中，也可以通过访问IST寄存器来清除边沿（包括上升沿和下降沿）触发的标记。执行 `LPC_PIN_INT->IST |= 0x01;` 语句后，PIO0_1原来的边沿中断标记就被清除了。

上述步骤也可通过下图来描述:



3. addition

最后需要说明一点，即使某个引脚并不作为GPIO使用，也能配接到PININT的中断源上。比如：要自动探测4个I2C接口中哪一个被连接到主机上，即可以把它们4个的SCL线所在的IO引脚分别配接到4路PININT中断上。通信时，发生哪个中断，就认为连接到了哪个I2C接口上。而此时，IO引脚并没有作为GPIO使用，而是作为I2C的SCL信号接口。

此外，对于外部引脚中断，在MDK5的开发环境中有特定的入口函数形式，比如对于PINTSEL0的中断，函数形式如下所示。

```
 void PIN_INT0_IRQHandler(void)
 {
        PINTSEL0中断服务程序部分
 }
```

下面来看一个外部引脚中断的实际例子，电路原理图如下所示:

```
#include <LPC82x.h>
//************************端口初始化**********************************
void Port_init(void)
{
LPC_GPIO_PORT->DIRSET0 = (1<<7) | (1<<13); //设置端口为输出方向
LPC_GPIO_PORT->SET0 = (1<<7) | (1<<13); //熄灭LED
}
//**************************主函数**********************************
int main(void)
{
Port_init(); //调用端口初始化
LPC_SYSCON->PINTSEL0 = 0x1;//选择PIO0_1作为外部中断引脚
LPC_SYSCON->PINTSEL1 = 0x4;//选择PIO0_4作为外部中断引脚
// LPC_PIN_INT->ISEL &= ~0x3; //边沿触发
LPC_PIN_INT->IENR |= 0x1; //PINTSEL0上升沿使能
LPC_PIN_INT->IENF |= 0x2; //PINTSEL1下降沿使能
NVIC_EnableIRQ(PIN_INT0_IRQn);//使能PINTSEL0中断
NVIC_EnableIRQ(PIN_INT1_IRQn);//使能PINTSEL1中断
while(1)
{
;
}
}
//*********************PINTSEL0中断（S2）*****************************
void PIN_INT0_IRQHandler(void)
{
LPC_GPIO_PORT->NOT0 = 0x2000; //LED2取反
LPC_PIN_INT->RISE |= 0x1;
}
//*********************PINTSEL1中断（S1）*****************************
void PIN_INT1_IRQHandler(void)
{
LPC_GPIO_PORT->NOT0 = 0x80; //LED1取反
LPC_PIN_INT->FALL |= 0x2;
}
```

把程序编译后下载到LPC824中，初始状态LED都不亮，按下S1，LED1仍然不亮，松开S1，LED1亮，证明上升沿有效，再次按下S1，LED1仍然亮，松开S1，LED1熄灭，证明上升沿有效。 按下S2，LED2亮，松开S2，LED2仍然亮，证明下降沿有效，再次按下S2，LED2熄灭，松开S2，LED2仍然熄灭，证明下降沿有效。