第 2-4 讲：外部中断(INT0~INT4)

1. 学习目的

1. 学习中断的相关概念。
2. 掌握外部中断配置及中断优先级配置的程序设计。
3. 掌握中断服务程序的编写。

2. 中断相关概念

1. 什么是中断

中断系统是为使 CPU 具有对外界紧急事件的实时处理能力而设置的。

CPU 在处理某一事件 A 时，发生了另一事件 B 请求 CPU 迅速去处理（中断发生）；CPU 暂时中断当前的工作，转去处理事件 B（中断响应和中断服务）；待 CPU 将事件 B 处理完毕后，再回到原来事件 A 被中断的地方继续处理事件 A（中断返回），这一过程称为中断。

中断机制能让紧急的事情得到及时的响应，从而大大的提高了程序的实时响应能力。

2. 中断优先级

单片机通常会有多个中断源，当多个中断源同时发出中断请求时，CPU 应该先响应哪一个中断？

为了解决这个问题，单片机规定了中断优先级，各个中断可以根据事情的轻重缓急分配不同的优先级。这样，当多个中断源同时发出中断请求时，优先级高的中断能先被响应，优先级高的中断执行完成之后再去执行优先级低的中断请求。

IAP15F2K61S2 单片机提供了 14 个中断请求源，分别是 INT0 中断、定时器 0 中断、INT1 中断、定时器 1 中断、串口 1 中断、ADC 中断、低压检测中断、PCA 中断、串口 2 中断、SPI 中断、INT2 中断、INT3 中断、定时器 2 中断和 INT4 中断，其中除 INT2 中断、INT3 中断、定时器 2 中断和 INT4 中断外，其他中断均有 2 级中断优先级可设置。

IAP15W4K61S4 单片机提供了 21 个中断请求源，分别是 INT0 中断、定时器 0 中断、INT1 中断、定时器 1 中断、串口 1 中断、ADC 中断、低压检测中断、PCA 中断、串口 2 中断、SPI 中断、INT2 中断、INT3 中断、定时器 2 中断、INT4 中断、串口 3 中断、串口 4 中断、定时器 3 中断、定时器 4 中断、比较器中断、PWM 中断和 PWM 异常检测中断，其中除 INT2 中断、INT3 中断、定时器 2 中断、INT4 中断、串口 3 中断、串口 4 中断、定时器 3 中断、定时器 4 中断和比较器中断外，其他中断均有 2 级中断优先级可设置。

高优先级的中断请求可以打断低优先级的中断，反之，低优先级的中断请求不可以打断高优先级的中断。当两个相同优先级的中断同时产生时，将由查询次序（自然优先级）来决定系统先响应哪个中断。

3. 中断嵌套

当 CPU 响应某一中断时，若有优先级更高的中断源发出中断请求，则 CPU 会中断正在进行的中断服务程序，并保护现场，响应更高优先级的中断，高优先级的中断处理完成后，再继续进行被中断的中断服务程序，这个过程称为中断嵌套。如果发出新的中断请求的中断源的优先权级别与正在处理的中断源同级或更低时，CPU 不会响应这个中断请求，直至正在处理的中断服务程序执行完以后才会去处理新的中断请求。

1. 中断的开启和关闭

单片机的中断通常可以开启和关闭，用户通过编程可以控制中断的开启和关闭，这样，用户就可以根据自己的需求使用相应的中断。

使用 STC 单片机时，用户可以用关总中断允许位（EA/IE.7）或相应中断的允许位屏蔽相应的中断请求，也可以用打开相应的中断允许位来使 CPU 响应相应的中断申请， 每一个中断源可以用软件独立地控制为开中断或关中断。

3. 单片机外部中断

IAP15F2K61S2/IAP15W4K61S4 单片机有 5 个外部中断 INT0~INT4，如下表所示。

表 1：外部中断

GPIO	外部中断	中断号	说明
P3.2	INT0	0	外部中断 0，支持上升沿和下降沿中断，中断优先级可配置为 0、1。
P3.3	INT1	2	外部中断 1，支持上升沿和下降沿中断，中断优先级可配置为 0、1。
P3.6	INT2	10	外部中断 2，只支持下降沿中断，中断优先级只能为最低优先级 0。
P3.7	INT3	11	外部中断 3，只支持下降沿中断，中断优先级只能为最低优先级 0。
P3.0	INT4	16	外部中断 4，只支持下降沿中断，中断优先级只能为最低优先级 0。

读者在使用这些外部中断的时候，要注意下面两点：

1) 中断触发方式：INT0 和 INT1 支持上升沿和下降沿触发中断，而 INT2、INT3 和 INT4 仅支持下降沿触发中断。对于 INT0 和 INT1，触发方式通过 TCON 寄存器中的 IT0 位和 IT1 位配置，如外部中断 0：

- IT0=0：上升沿或下降沿均可触发外部中断 0。
- IT0=1，下降沿触发外部中断 0。

这里，我们可以看到 INT0 和 INT1 是无法配置为单独的上升沿触发中断的，这一点也是编程时需要特别注意的事项。

2) 中断优先级：INT0 和 INT1 的中断优先级是可配置的，可配置的优先级别为 0~1，而 INT2、INT3 和 INT4 的中断优先级是固定的，只能是最低优先级 0。

4. 软件设计

4.1. 外部中断应用步骤

外部中断的应用流程如下图所示，本节，我们重点描述外部中断初始化和中断服务程序的编写。

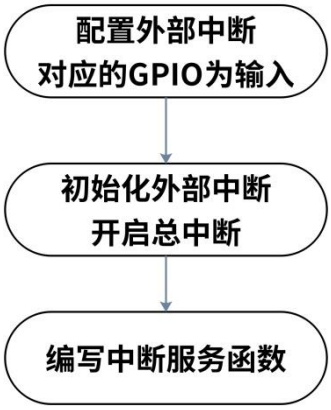

图 1：外部中断应用步骤

4.1.1. 初始化外部中断

初始化外部中断所要做的工作主要包括使能需要使用的外部中断、配置外部中断的触发方式以及配置中断优先级，最后开启总中断。

1. 使能需要使用的外部中断

外部中断中的 INT0 和 INT1 的开启和关闭由中断使能寄存器 IE 的位 0（EX0）和位 2（EX1）控制，如下图所示。

IE（中断使能寄存器）

符号	地址	B7	B6	B5	B4	B3	B2	B1	B0
IE	A8H	EA	ELVD	EADC	ES	ET1	EX1	ET0	EX0

总中断允许位　外部中断1允许位　外部中断0允许位

图 2：IE 寄存器

- EA：总中断允许控制位。EA 的作用是使中断允许形成多级控制。即各中断源首先受 EA 控制，其次还受各中断源自己的中断允许控制位控制。
  0：CPU 屏蔽所有的中断申请
  1：CPU 开放中断
- EX0：外部中断 0 中断允许位。
  0：禁止 INT0 中断。
  1：允许 INT0 中断。
- EX1：外部中断 1 中断允许位。
  0：禁止 INT1 中断。

1：允许 INT1 中断。

外部中断里面的 INT2、INT3 和 INT4 的开启和关闭由外部中断与时钟输出控制寄存器 INTCLKO 的位 4（EX2）、位 5（EX3）和位 6（EX4）控制，如下图所示。

INT_CLKO (AUXR2)：外部中断允许和时钟输出寄存器

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
INT_CLKO (AUXR2)	8FH	name	-	EX4	EX3	EX2	MCKO_S2	T2CLKO	T1CLKO	T0CLKO

外部中断4允许位 外部中断3允许位 外部中断2允许位

图 3：中断与时钟输出控制寄存器

- EX2：外部中断 2 中断允许位。
  0：禁止 INT2 中断。
  1：允许 INT2 中断。
- EX3：外部中断 3 中断允许位。
  0：禁止 INT3 中断。
  1：允许 INT3 中断。
- EX4：外部中断 4 中断允许位。
  0：禁止 INT4 中断。
  1：允许 INT4 中断。

2. 配置外部中断的触发方式

外部中断 0 和 1 的触发方式由“定时器 0/1 控制寄存器 TCON”中的位 0（IT0）和位 2（IT1）控制，如下图所示。

读者需要注意的是：INT2、INT3 和 INT4 只有下降沿触发方式，因此，他们的触发方式是无需配置的（单片机也没有提供可配置的寄存器）。

符号	地址	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

外部中断1触发控制位 外部中断0触发控制位

图 4：TCON 寄存器

- IT0：外部中断源 0 触发控制位。 IT0=0，上升沿或下降沿均可触发外部中断 0。
  IT0=1，外部中断 0 为下降沿触发方式。
- IT1：外部中断源 1 触发控制位。 IT1=0，上升沿或下降沿均可触发外部中断 1。
  IT1=1，外部中断 1 为下降沿触发方式。

3. 配置中断优先级

IAP15F2K61S2/IAP15W4K61S4 的 5 个外部中断里面，INT0、INT1 的中断优先级是可

配置为 0~1 级别的，INT2、INT3 和 INT4 的中断优先级是固定为最低级别 0 的，是不能配置的。中断优先级控制寄存器如下图所示。

IP：中断优先级控制寄存器（可位寻址）

SFR name	Address	bit	B7	B6	B5	B4	B3	B2	B1	B0
IP	B8H	name	PPCA	PLVD	PADC	PS	PT1	PX1	PT0	PX0

外部中断1优先级控制位　　外部中断0优先级控制位

图 5：中断优先级控制寄存器

- PX0：外部中断 0 中断优先级控制位，复位值为 0，即最低优先级。
  0：INT0 中断优先级为 最低优先级（优先级 0）。
  1：INT0 中断优先级为最高优先 级（优先级 1）。
- PX1：外部中断 1 中断优先级控制位，复位值为 0，即最低优先级。
  0：INT1 中断优先级为 最低优先级（优先级 0）。
  1：INT1 中断优先级为最高优先 级（优先级 1）。

- 优先级配置示例：配置 INT0 的中断优先级为 0

代码清单：INT0 中断优先级配置示例

```
1.  PX0  = 1;   //将 INT0 优先级设置为高优先级
2.  // PX0  = 0;    //将 INT0 优先级设置为低优先级
```

4.1.2. 编写中断服务函数

中断服务函数是由硬件自动调用的，应用程序是不能调用中断服务函数的。中断服务函数使用关键字“interrupt”声明，其格式如下：

代码清单：中断服务函数声明格式

```
1.  void 函数名(void) interrupt n using m {
2.  //用户代码
3.
4.  }
```

上面的中断服务函数声明中：

- void：中断服务函数返回值类型必须为 void 类型，因此，这里必须是 void，指明函数无返回值。
- 函数名：开发人员命名，和普通函数命名方式一样。
- (void)：中断服务函数不能调用参数，因此，这里必须是 void，指明函数无参数。
- interrupt：声明中断服务函数的关键字，指明该函数是一个中断服务函数。
- n：中断号，就是中断查询次序号。必须是整数，取值范围 0-31。中断号和中断源是对应的（如外部中断 0 的中断号是 0），因此，中断号表示的是该中断服务函数所对应

的中断源。单片机的各个中断源的中断号可以在单片机的数据手册中查询。

- using m：用指定该中断服务程序要使用的工作寄存器组号，通常使用缺省值，即中断服务函数中省略“using m”。

■ **示例：**下面的代码是外部中断 0 的中断服务函数，其中断号为 0，该函数中翻转指示灯 D2 状态。当我们初始化（开启 INT0 中断，配置触发方式和优先级）外部中断 0 并且开启了总中断之后，如果外部中断 0 发生，单片机硬件会自动调用该中断服务函数。

代码清单：外部中断 0 的中断服务函数

```
1.  void ext_int0_isr (void) interrupt 0
2.  {
3.      led_toggle(LED_2);          //翻转用户指示灯 D2
4.  }
```

4.2. 外部中断实验

✧ 注：本节的实验是在“实验 2-3-1：独立按键检测”的基础上修改，本节对应的实验源码是：“实验 2-4-1：外部中断”。

4.2.1. 实验内容

1. 编写 INT0~INT4，5 个外部中断的初始化和中断服务函数。
2. 主函数中初始化 INT0（下降沿中断）、INT1（下降沿中断）和 INT4（下降沿中断），即本实验演示 INT0、INT1 和 INT4。这里使用 INT0、INT1 和 INT4，是因为他们的引脚连接到了独立按键 S5、独立按键 S4 和独立按键 S7，方便测试。

4.2.2. 代码编写

1. 新建一个名称为“exint.c”的文件及其头文件“exint.h”并保存到工程的“Source”文件夹，并将“exint.c”加入到 Keil 工程中的“SOURCE”组。
2. 引用头文件

因为在“main.c”文件中使用了“exint.c”文件中的函数，所以需要引用下面的头文件“exint.h”。

代码清单：引用头文件

```
1.  //引用外部中断头文件
2.  #include  "exint.h"
```

3. 编写 5 个外部中断的初始化和中断服务函数

中断服务函数中执行的功能均为翻转指示灯的状态的样例代码，读者可以在中断服务函数中根据自己的实际需求编写功能代码，但是需要注意的是：中断服务函数中的功能代码要做到“短小精悍”，尽可能的减少占用中断的时间，花费时间较多的程序代码可以放到主程序中运行。

下面的代码列出了 INT0、INT1 和 INT4 的中断初始化和中断服务函数，其他外部中断的代码读者可以在本节的例子程序里面查看。

1) 外部中断 0 的初始化函数和中断服务函数

外部中断 0 的触发方式配置为下降沿触发，中断优先级设置为最低优先级 0，如下。

代码清单：外部中断 0 初始化函数

```
1. /************************************************************
2. 功能描述：外部中断 0 初始化，外部中断 0 的引脚：P3.2，INT0 可以配置为
3.          ：上升沿或下降沿均可触发外部中断（IT0=0）
4.          ：下降沿触发外部中断（IT0=1）
5. 参     数：无
6. 返 回 值：无
7. ************************************************************/
8. void ext_int0_init(void)
9. {
10.    INT0_Enable();       //使能 INT0 中断
11.    IT0 = 1;             //选择 INT0 为下降沿触发方式
12.    PX0 = 0;             //将 INT0 优先级设置为低优先级
13.}
```

代码清单：外部中断 0 中断服务函数

```
1. /************************************************************
2. 功能描述：外部中断 0 的中断服务程序
3. 参     数：无
4. 返 回 值：无
5. ************************************************************/
6. void ext_int0_isr (void) interrupt 0
7. {
8.     //可以在这里加入自己的应用代码，但是注意：中断服务函数中占用的时间尽可能的短
9.     led_toggle(LED_2);         //翻转用户指示灯 D2
10.}
```

2) 外部中断 1 的初始化函数和中断服务函数

外部中断 1 的触发方式配置为下降沿触发，中断优先级设置为最低优先级 0，如下。

代码清单：外部中断 1 初始化函数

```
1. /************************************************************
2. 功能描述：外部中断 1 初始化，外部中断 1 的引脚：P3.3，INT1 可以配置为
3.          ：上升沿或下降沿均可触发外部中断（IT0=0）
4.          ：下降沿触发外部中断（IT0=1）
5. 参     数：无
6. 返 回 值：无
7. ************************************************************/
8. void ext_int1_init(void)
9. {
10.    INT1_Enable();      //使能 INT0 中断
```

```
    IT1 = 1;            //选择 INT0 为下降沿触发方式
    PX1 = 0;            //将 INT1 优先级设置为低优先级
}
```

代码清单：外部中断 1 中断服务函数

```
/*************************************************************
功能描述：外部中断 1 的中断服务程序
参    数：无
返 回 值：无
*************************************************************/
void ext_int1_isr (void) interrupt 2
{
    //可以在这里加入自己的应用代码，但是注意：中断服务函数中占用的时间尽可能的短
    led_toggle(LED_1);        //翻转用户指示灯 D1
}
```

3) 外部中断 4 的初始化函数和中断服务函数

外部中断 4 只支持下降沿触发，中断优先级只能是最低优先级 0，因此，初始化函数里面没有触发方式和中断优先级的配置。

代码清单：外部中断 4 初始化函数

```
/*************************************************************
功能描述：外部中断 4 初始化，外部中断 4 的引脚：P3.0
        ：INT4 只能下降沿触发外部中断
参    数：无
返 回 值：无
*************************************************************/
void ext_int4_init(void)
{
    INT4_Enable();   //使能 INT4 中断
}
```

代码清单：外部中断 4 中断服务函数

```
/*************************************************************
功能描述：外部中断 4 的中断服务程序
参    数：无
返 回 值：无
*************************************************************/
void ext_int4_isr (void) interrupt 16
{
    //可以在这里加入自己的应用代码，但是注意：中断服务函数中占用的时间尽可能的短
    led_toggle(LED_3);        //翻转用户指示灯 D3
}
```

4. 主函数

主函数中配置 INT0、INT1 和 INT4 的引脚为准双向输入，接着调用 INT0、INT1 和 INT4 的初始化函数 ext_int0_init()、ext_int1_init()和 ext_int4_init()完成 INT0、INT1 和 INT4 的初始化，之后开启总中断即可。

代码清单：主函数

```
/*******************************************************************************
功能描述：主函数
入口参数：无
返 回 值：int 类型
*******************************************************************************/
int main(void)
{
    P2M1 &= 0x1F;   P2M0 |= 0xE0;     //设置 P2.5、P2.6、P2.7 为推挽输出
    P0M1 &= 0x00;   P0M0 |= 0xFF;     //设置 P0.0 ~ P0.7 为推挽输出
    P3M1 &= 0xF0;   P3M0 &= 0xF0;     //设置 P3.0 ~ P0.3 为准双向口

    SEG_off();          //控制 8 位数码管/点阵不显示
    ULN2003_off();      //控制步进电机、蜂鸣器、继电器等不工作
    leds_off();         //熄灭 D1~D8 指示灯
    delay_ms(10);       //延时

    ext_int0_init();    //初始化外部中断 0
    ext_int1_init();    //初始化外部中断 1
    ext_int4_init();    //初始化外部中断 4
    EA = 1;             //允许总中断
    delay_ms(10);       //延时

    while(1)
    {
        ;
    }
}
```

4.2.3. 硬件连接

本实验程序的编写都是基于 IO 模式，所以 J24 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择为 IO 模式。同时 INT0、INT1 和 INT4 对应的是独立按键 S5、独立按键 S4 和独立按键 S7，所以 J6 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择独立按键。

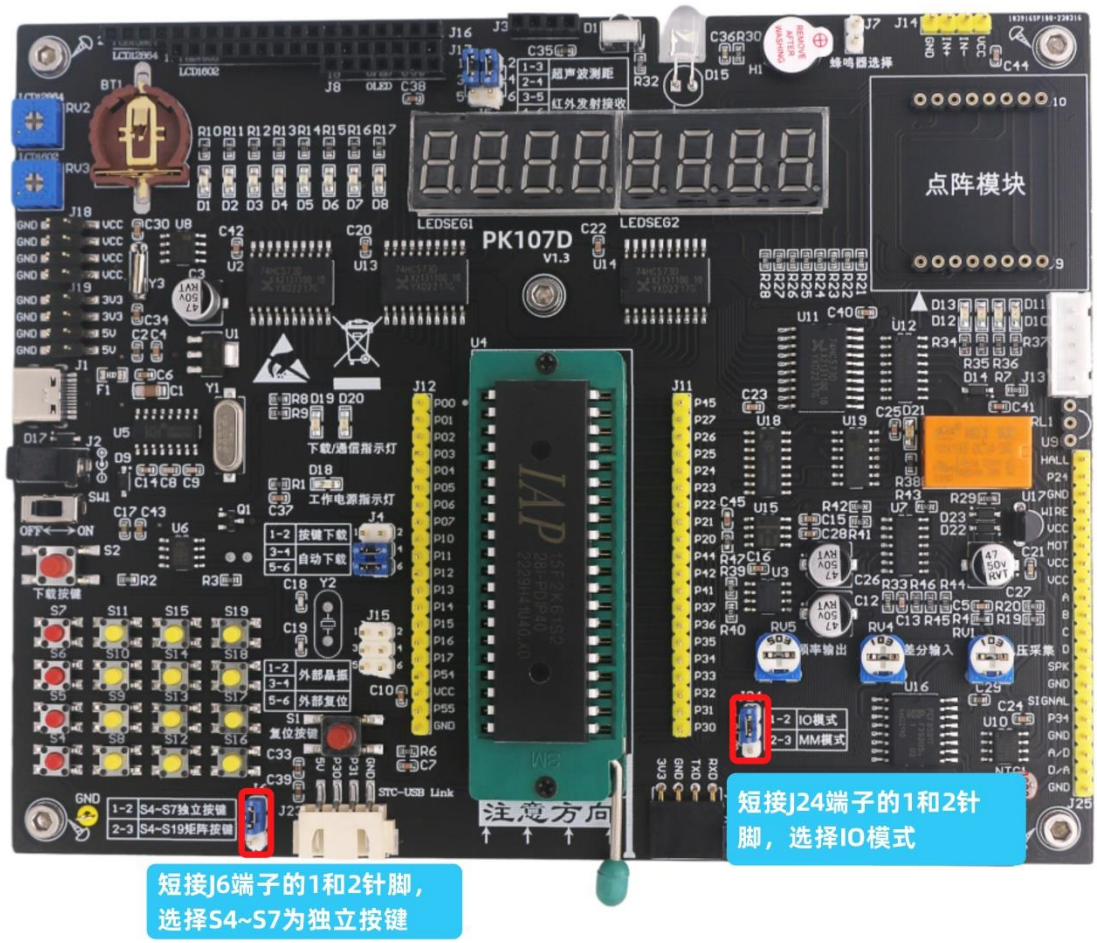

图 6：跳线帽短接

4.2.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-4-1：外部中断”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2) 双击“…\extint\project”目录下的工程文件“extint.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“extint.hex”位于工程的“…\extint\Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：12MHz。
5) 程序运行后，按下按键 S5 触发外部中断 0，中断服务函数中翻转指示灯 D2 的状态（由亮变灭或由灭变亮）。按下按键 S4 触发外部中断 1，中断服务函数中翻转指示灯 D1 的状态（由亮变灭或由灭变亮）。按下按键 S7 触发外部中断 4，中断服务函数中翻转指示灯 D3 的状态（由亮变灭或由灭变亮）。

✧ 注意事项：

因为按键存在抖动，因此一次按键可能会多次触发中断，即指示灯状态会翻转多次。

4.3. 中断优先级抢占实验

✧ 注：本节的实验是在“实验 2-4-1：外部中断”的基础上修改，本节对应的实验源码是：“实验 2-4-2：中断优先级抢占”。

4.3.1. 实验内容

1. 配置 INT0（独立按键 S5）：上升/下降沿触发中断，中断优先级为优先级 1，中断服务函数中翻转指示灯 D2 状态。
2. 配置 INT1（独立按键 S4）：下降沿中断，中断优先级为最低优先级 0，中断服务函数中翻转指示灯 D1 状态，并延时 5 秒。这里延时 5 秒是为了在 INT1 中断退出前有足够的时间去按下按键 S5 触发 INT0，演示 INT0 抢占 INT1，方便我们理解中断优先级抢占。切记，实际应用的时候不能在中断服务函数中进行长延时。

4.3.2. 代码编写

1. 编写 INT0 的初始化和中断服务函数

外部中断 0 的触发方式配置为上升/下降沿触发中断，中断优先级配置为 1，代码清单如下。

代码清单：外部中断 0 初始化函数

```
/*************************************************************
功能描述：外部中断 0 初始化，外部中断 0 的引脚：P3.2，INT0 可以配置为
        ：上升沿或下降沿均可触发外部中断（IT0=0）
        ：下降沿触发外部中断（IT0=1）
参    数：无
返 回 值：无
*************************************************************/
void ext_int0_init(void)
{
    INT0_Enable();  //使能 INT0 中断
    IT0  = 0;           //上升沿或下降沿均可触发外部中断 0
    //设置 INT0 的中断优先级为高优先级，即设置 PX0 = 1,本例中 INT1 优先级设置为低优先级，所以 INT0 可以抢占 INT1
    PX0 = 1;          //将 INT0 优先级设置为高优先级

    //设置 INT0 的中断优先级为 0，即设置 PX0 = 0,本例中 INT1 优先级设置为 0，INT0 和 INT1 优先级相同，无法抢占
    //PX0 = 0;        //将 INT0 优先级设置为低优先级
}
```

NT0 中断服务函数中翻转指示灯 D2 状态，由指示灯 D2 的状态即可判断 INT0 有没有触发，代码清单如下。

代码清单：外部中断 0 中断服务函数

```
/*******************************************************************************
功能描述：外部中断 0 的中断服务程序
参    数：无
返 回 值：无
*******************************************************************************/
void ext_int0_isr (void) interrupt 0
{
    //可以在这里加入自己的应用代码，但是注意：中断服务函数中占用的时间尽可能的短
    led_toggle(LED_2);
}
```

4) 外部中断 1 的初始化函数和中断服务函数

外部中断 1 设置为下降沿触发，中断优先级设置为优先级 0（优先级低于前面配置的外部中断 0），因此，INT0 可以抢占 INT1。

代码清单：外部中断 1 初始化函数

```
/*************************************************************
功能描述：外部中断 1 初始化，外部中断 1 的引脚：P3.3，INT1 可以配置为
        ：上升沿或下降沿均可触发外部中断（IT0=0）
        ：下降沿触发外部中断（IT0=1）
参    数：无
返 回 值：无
*************************************************************/
void ext_int1_init(void)
{
    INT1_Enable();      //使能 INT0 中断
    IT1 = 1;            //选择 INT0 为下降沿触发方式
    PX1 = 0;            //将 INT1 优先级设置为低优先级
}
```

INT1 中断服务函数中翻转指示灯 D1 状态，并延时 5 秒，以方便演示中断抢占，代码清单如下。

代码清单：外部中断 1 中断服务函数

```
/*************************************************************
功能描述：外部中断 1 的中断服务程序
参    数：无
返 回 值：无
*************************************************************/
void ext_int1_isr (void) interrupt 2
{
    led_toggle(LED_1); //翻转用户指示灯 D1
    delay_ms(5000);    //延时 5 秒，这是为了方便延时中断抢占，实际应用时不要在中断服务函数中执行长延时
}
```

5. 主函数

主函数中配置外部中断 0 和外部中断 1 的引脚为准双向输入，接着调用外部中断 0 和 1 的初始化函数 ext_int0_init()和 ext_int1_init()完成外部中断 0 和 1 的初始化，之后开启总中断即可。

代码清单：主函数

```
1. /*************************************************************************
2. 功能描述：主函数
3. 入口参数：无
4. 返 回 值：int 类型
5. *************************************************************************/
6. int main(void)
7. {
8.     P2M1 &= 0x1F;   P2M0 |= 0xE0;     //设置 P2.5、P2.6、P2.7 为推挽输出
9.     P0M1 &= 0x00;   P0M0 |= 0xFF;     //设置 P0.0 ~ P0.7 为推挽输出
10.    P3M1 &= 0xF0;   P3M0 &= 0xF0;     //设置 P3.0 ~ P0.3 为准双向口
11.
12.    SEG_off();          //控制 8 位数码管/点阵不显示
13.    ULN2003_off();      //控制步进电机、蜂鸣器、继电器等不工作
14.    leds_off();         //熄灭 D1~D8 指示灯
15.    delay_ms(10);       //延时
16.
17.    ext_int0_init();    //初始化外部中断 0
18.    ext_int1_init();    //初始化外部中断 1
19.    EA = 1;             //允许总中断
20.    delay_ms(10);       //延时
21.
22.    while(1)
23.    {
24.        ;
25.    }
26.}
```

4.3.3. 硬件连接

本实验程序的编写都是基于 IO 模式，所以 J24 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择为 IO 模式。同时 INT0、INT1 对应的是独立按键 S5、独立按键 S4，所以 J6 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择独立按键。

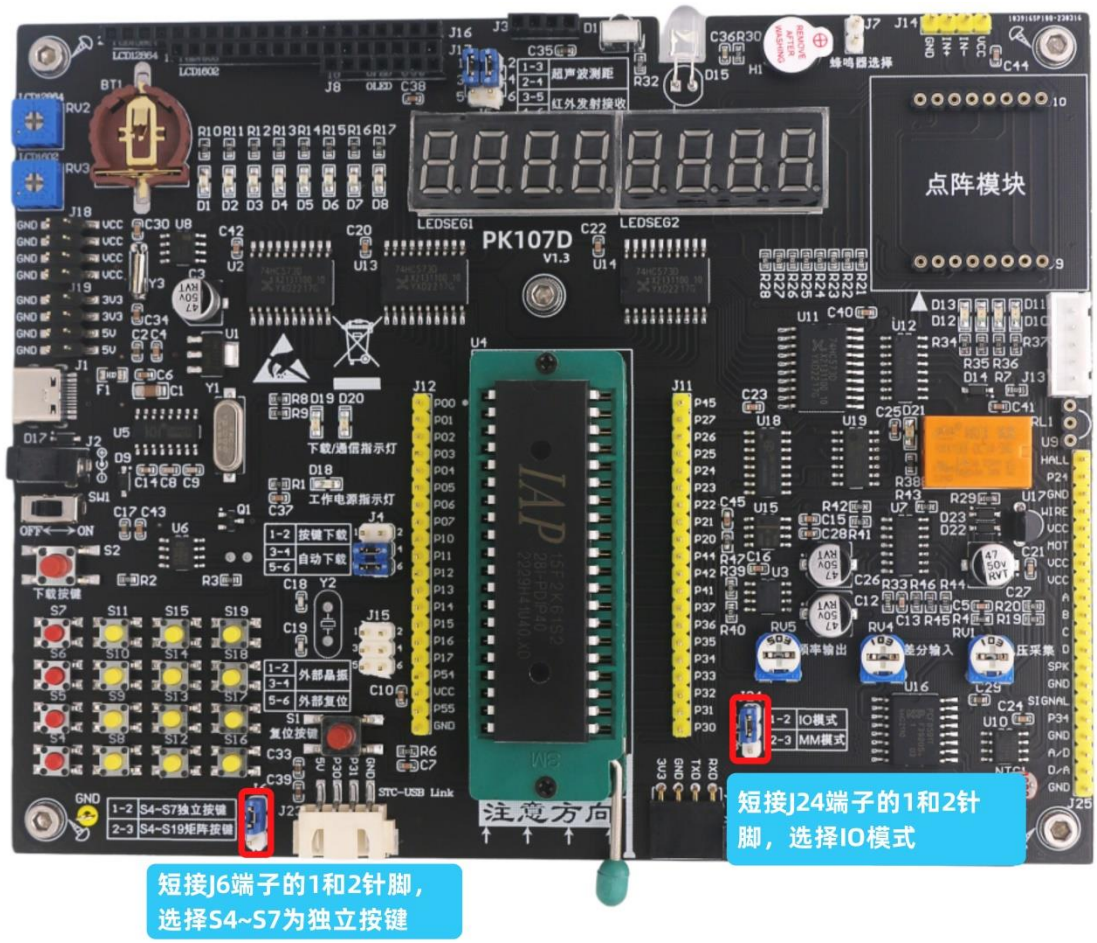

图 7：跳线帽短接

4.3.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-4-2：中断优先级抢占”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2) 双击“…\extint_pri\project”目录下的工程文件“extint_pri.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“extint_pri.hex”位于工程的“…\extint\Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：12MHz。
5) 程序运行后，按下按键 S4 触发外部中断 1，可以观察到指示灯 D1 状态改变。5 秒内即 INT1 还未退出的情况下按下按键 S5，可以观察到指示灯 D2 状态改变。这说明由于程序中配置的 INT0 的优先级高于 INT1 的优先级，高优先级的中断可以抢占低优先级的中断。

✧ 注意事项：

因为按键存在抖动，因此一次按键可能会多次触发中断，即指示灯状态会翻转多次。