外部中断

1. 实验目的

- 掌握 STC15W4K32S4 系列 MCU 外部中断及中断优先级的原理。
- 掌握外部中断配置及中断优先级配置的程序设计。

2. 实验内容

- 编写程序实现对单个外部中断触发的控制设计。
- 编写程序实现对多个外部中断触发控制及中断优先级的设计。

3. 硬件电路设计

3.1. 开发板 IO 口外部中断硬件电路

进取者 STC15 开发板上设计了 1 个电容式触摸按键和 1 个用户轻触按键，可以为外部中断引脚提供触发信号，电路如下。

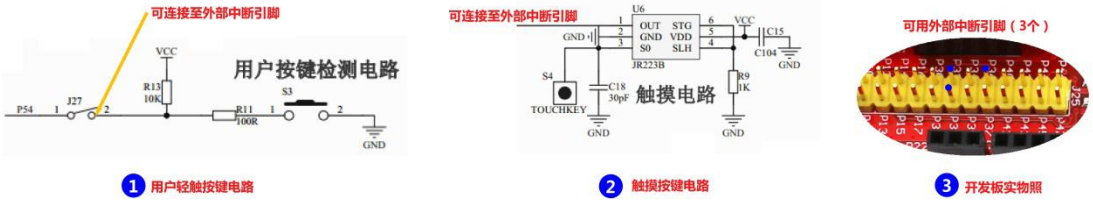

图 1：开发板电路

■ 开发板 3 个可用外部中断引脚如下表：

表 1：开发板外部中断引脚分配

INT	对应 IO 口	功能描述	说明	备注
INT0	P3.2	外部中断 0	非独立 GPIO	W5500 模块接口 J16 使用。
INT1	P3.3	外部中断 1	非独立 GPIO	nRF24L01 模块接口 J11 使用。
INT2	P3.6	外部中断 2	非独立 GPIO	RC522 接口 J16、DHT11 接口 J13 使用。

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。针对非独立 GPIO 使用时需特别注意。

3.2. STC15W4K32S4 系列 MCU 外部中断

STC15W4K32S4 系列 MCU 不是每一个 GPIO 口都可以作为外部中断输入使用，只有特定的 GPIO 口才可以作为特定的外部中断输入使用。STC15W4K32S4 系列 MCU 具有 5 个外部中断引脚，列表如下。

表 2：STC15W4K32S4 系列外部中断引脚分配

序号	INT	对应 IO 口	功能描述	备注
1	INT0	P3.2	外部中断 0	中断优先级有高低之分。
2	INT1	P3.3	外部中断 1	中断优先级有高低之分。
3	INT2	P3.6	外部中断 2	只能为低中断优先级。
4	INT3	P3.7	外部中断 3	只能为低中断优先级。
5	INT4	P3.0	外部中断 4	只能为低中断优先级。

单片机外部中断引脚电平变化了，就有可能触发中断，进而执行中断服务程序。电平变化有可能是高电平到低电平（即产生下降沿），也可能是低电平到高电平（即产生上升沿）。STC15W4K32S4 系列外部中断引脚支持的触发方式是不一样的。如下表。

表 3：STC15W4K32S4 系列外部中断引脚触发方式

序号	触发方式	INT	备注
1	下降沿触发	INT0、INT1、INT2、INT3、INT4	
2	上升沿触发	无	指只可上升沿触发的意思。
3	上升沿或下降沿触发	INT0、INT1	

实现 GPIO 口电平变化的方式有很多，使用外部按键改变 GPIO 口电平是一种简单直观的方式。但在实际使用中单片机的外部中断功能是应用于一些模块或芯片的中断引脚上，比如 W5500 以太网模块、nRF24L01 无线模块等。

下面介绍下带有中断引脚（IRQ）的模块或传感器与单片机连接示意图。

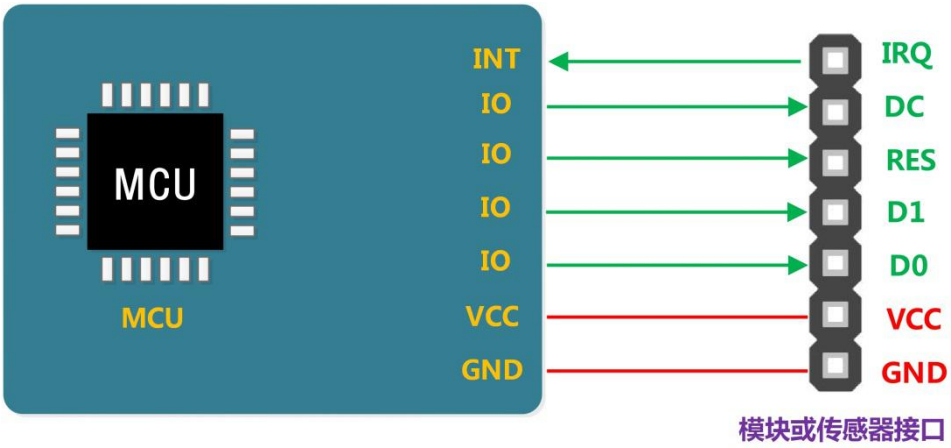

图 2：带中断引脚的模块或传感器与单片机连接示意图

■ 需要弄清楚的问题。

1） 模块或传感器 IRQ 引脚的有效电平（低电平有效还是高电平有效）。

2） 模块或传感器 IRQ 引脚一旦跳变到有效电平后需要单片机执行的操作。

■ 延伸出的问题。

1） 如果模块或传感器 IRQ 引脚是低有效电平，那么产生的就是下降沿；如果模块或传感器 IRQ 引脚是高有效电平，那么产生的就是上升沿。这就需要单片机外部中断引脚有匹配的触发方式才可，如果没有怎么办呢？

2） 与模块或传感器 IRQ 引脚相连的单片机引脚可以不是外部中断引脚，而是普通的 GPIO 口吗？

■ 回答延伸出的问题。

1） 从 STC15W4K32S4 系列外部中断引脚触发方式表格中可以获悉，单片机 INT0 和 INT1 可选择为上升沿或下降沿触发方式，这基本是可以满足大部分的应用需求的。如果单片机只有 INT2、INT3 或 INT4（下降沿触发方式）的外部中断引脚可用，比较简便的方式是对模块或传感器的 IRQ 信号进行取反处理，硬件上加个反相器是一种做法，但务必考虑会有一个信号延时的问题。

2） 模块或传感器 IRQ 引脚一般是可以和单片机 GPIO 口引脚相连的，这在程序设计时采用的就是查询方式。也就是单片机程序设计时不断循环查询与模块或传感器 IRQ 引脚相连的 GPIO 口状态，若检测到 GPIO 口跳变到有效电平后才会执行需要的操作。当单片机使用中断引脚与模块或传感器 IRQ 引脚相连时，设计程序便可采用中断方式，后面会详细介绍查询方式和中断方式的区别。

3.3. 单片机中断方式和查询方式

单片机中断方式，是事件触发的，占用的 CPU 资源相对很少。换言之打开中断之后，CPU 可以做其他事情，只有当符合条件的事件产生时才会进入中断，去执行中断服务程序。举例，针对 STC15W4K32S4 系列单片机，前面介绍的下降沿中断或上升沿中断都是触发中断的条件，在这个条件不满足时单片机是不用理会外部中断引脚的，而一旦条件满足，如果程序中没有其他同时发生的更高优先级的中断的话，CPU 会立即执行该外部中断对应的中断服务函数。

单片机查询方式，就是在主函数里面不停循环，查询端口状态，这会占用大量 CPU 资源。这时模块或传感器 IRQ 引脚是可以和单片机普通 IO 口引脚相连的。这种方式的弊端：

1） 占用过多 CPU 资源，影响响应速度。

2） 在处理事件多，处理流程复杂，函数嵌套执行的情况下，由于 CPU 处理不过来可能会丢失事件。（往往模块或传感器 IRQ 引脚跳变到有效电平的时间也是有限的，如果在有限的时间内没有捕获到有效信号就会丢失事件）

✧ 注：单片机查询方式的缺点就是单片机中断方式的优点。当然，单片机查询方式的优点是可以使用普通 IO 口甚至不使用 IO 口。

3.4. 外部中断配置步骤

针对 STC15W4K32S4 系列单片机 5 个外部中断引脚，软件的配置过程如下：

图 3：外部中断软件配置步骤

✧ 注：实验例程即是按照上述配置步骤操作寄存器相关位实现，后有详述。

4. 软件设计

4.1. 寄存器解析

首先普及一个常用知识点：在操作单片机寄存器时常说有的寄存器支持位寻址，有的寄存器不支持位寻址？这里的位寻址究竟是什么含义？

位寻址定义：对位地址中的内容进行位操作的寻址方式称为位寻址。由于单片机中只有内部 RAM 和特殊功能寄存器的部分单元有位地址，因此位寻址只能对有位地址的这两个空间进行寻址操作。

单片机中不是每个寄存器都支持位寻址的，一般如果 SFR（特殊功能寄存器）的地址值能被 8 整除，则该 SFR 可以进行位寻址。举例 P0 端口数据寄存器是支持位寻址的。

图 4：P0 端口数据寄存器

✧ 注：单片机中寄存器地址值如不能被 8 整除，则该寄存器是不支持位寻址的。针对 STC15W4K32S4 系列有些 SFR 是位于扩展 RAM 中，这些 SFR 不可以按照这个总结的规则去判定是否支持位寻址。访问位于扩展 RAM 中的 SFR 需要先操作 P_SW2 的 BIT7 才可，这个在用到时讲解。

4.1.1. 中断允许寄存器 IE

中断允许寄存器 IE 支持位寻址，该寄存器的 B7 位是单片机所有中断的总闸，即其他中断位使能后，最后还必须将这个总中断打开才能开启中断，与外部中断有关的还有 B0 和 B2 位，含义如下图。

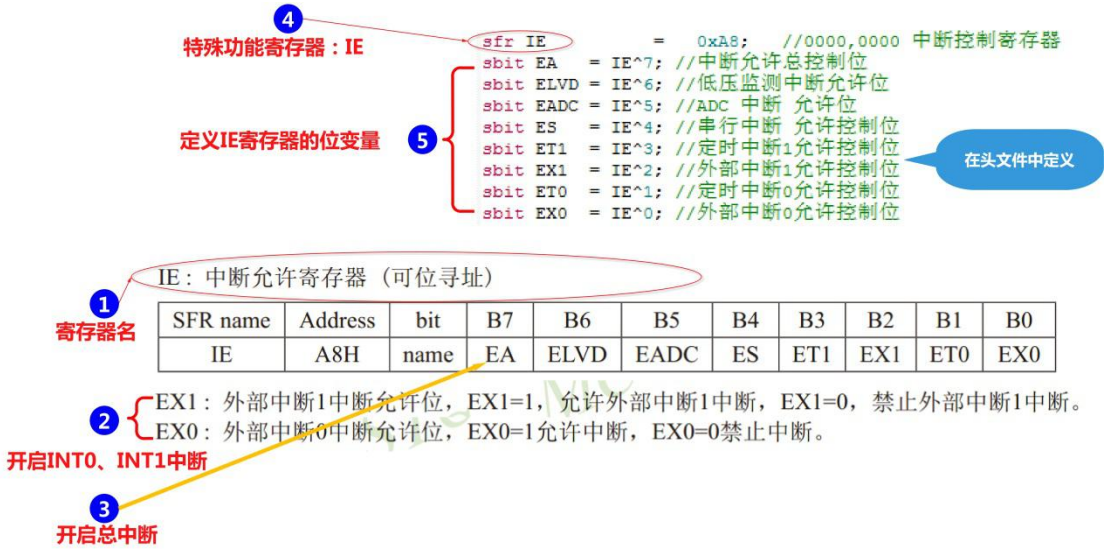

图 5：中断允许寄存器

4.1.2. 外部中断允许和时钟输出寄存器 AUXR2

外部中断允许和时钟输出寄存器 AUXR2 不支持位寻址，该寄存器的 B4、B5 和 B6 位是外部中断 2、外部中断 3 和外部中断 4 的中断允许位。因为 AUXR2 寄存器不支持位寻址，所以操作该寄存器位时，不可以直接“EX2=0;”进行操作 INT2 中断允许位，参考下图。

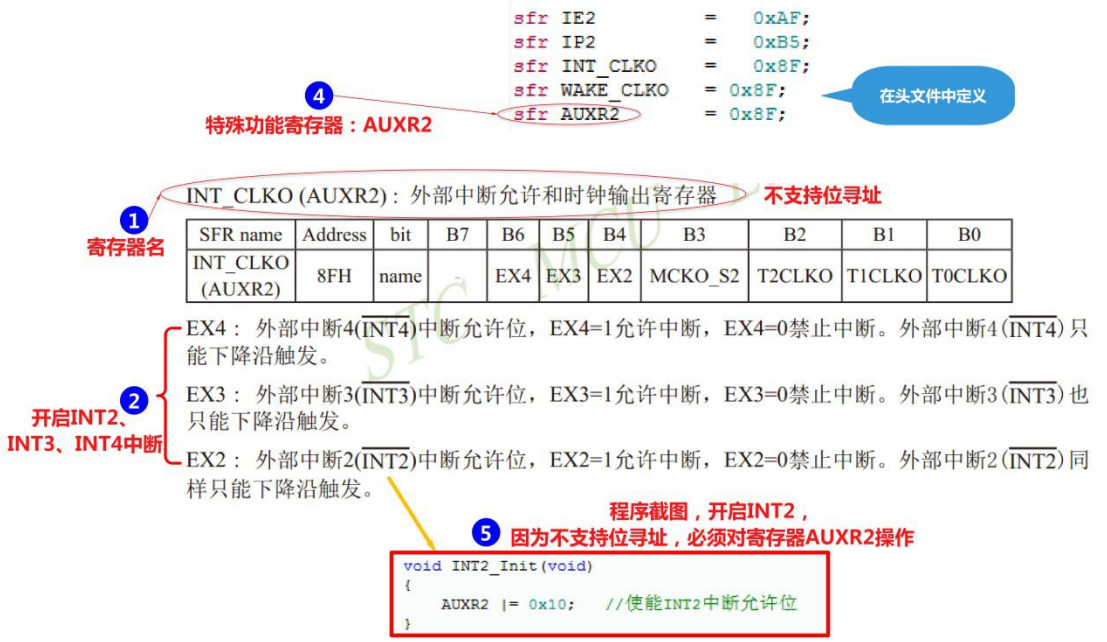

图 6：外部中断允许和时钟输出寄存器

4.1.3. 定时器/计数器中断控制寄存器 TCON

定时器/计数器中断控制寄存器 TCON 支持位寻址，该寄存器的 B0 位和 B2 位是选择 INT0 和 INT1 中断源类型的(可理解成中断触发方式)，寄存器 B1 位和 B3 位是 INT0 和 INT1 中断请求标志，含义如下图。

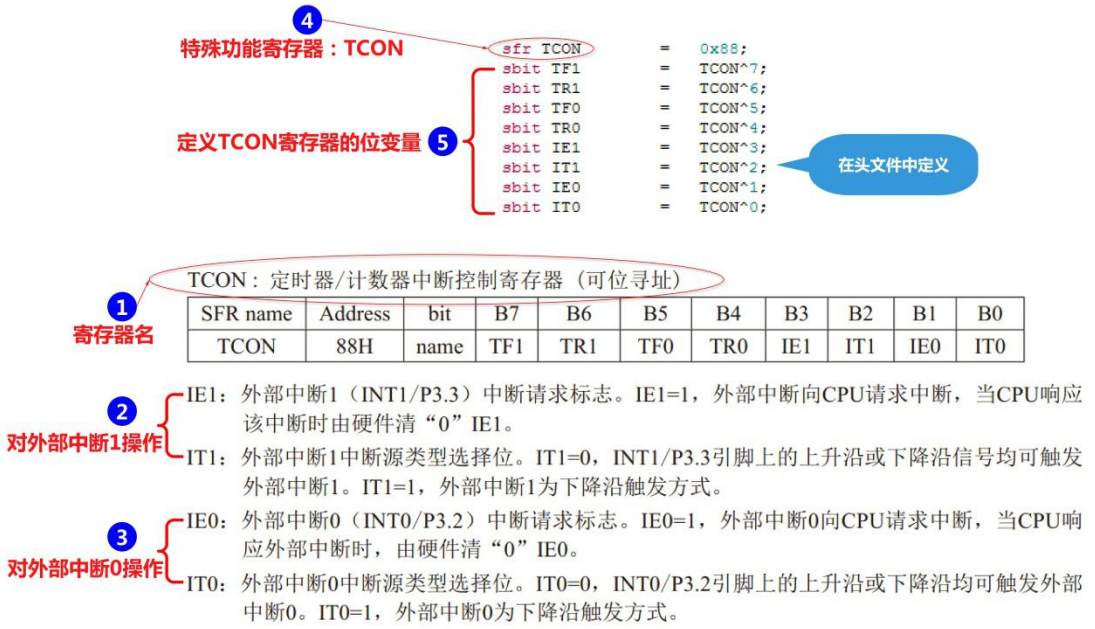

图 7：定时器/计数器中断控制寄存器

✧ 注：单片机中很多寄存器都像 TCON 寄存器一样，寄存器关联的单片机外设不止一个，这就需要我们在操作寄存器时一定要按位操作，用不到的位千万不要操作，否则在程序比较复杂、用到的外设比较多时，往往会遇到不可知问题。

4.1.4. 中断优先级控制寄存器 IP

中断优先级控制寄存器 IP 支持位寻址，该寄存器的 B0 位和 B2 位是设置 INT0 和 INT1 中断优先级的，含义如下图。需要说明的是 INT2、INT3 和 INT4 是没有中断优先级的。

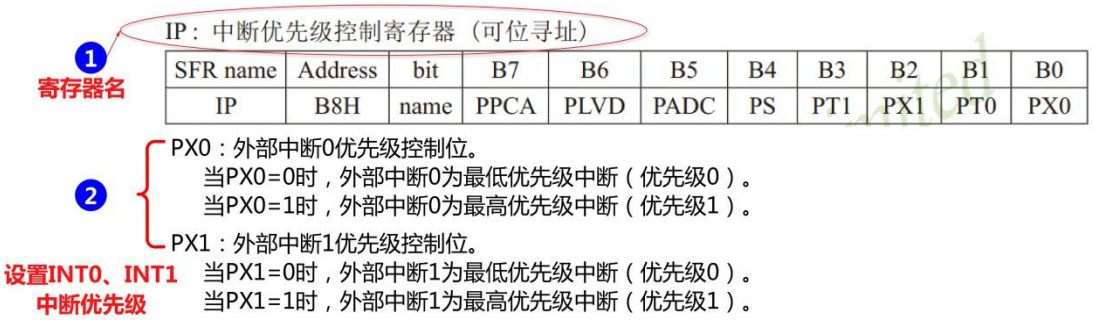

图 8：中断优先级控制寄存器

4.2.外部中断 0 实验（下降沿中断方式）

✧ 注：本节的实验源码是在“实验 2-2-2： GPIO 输入按键检测（多个 c 文件）”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 0（下降沿中断方式）”。

4.2.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 4：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	led	.c	包含与用户 led 控制有关的用户自定义函数。
2	exint	.c	外部中断有关的用户自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.2.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "led.h"
3. #include "exint.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 5：头文件包含路径

序号	路径	描述
1	..\ Source	led.h、exint.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

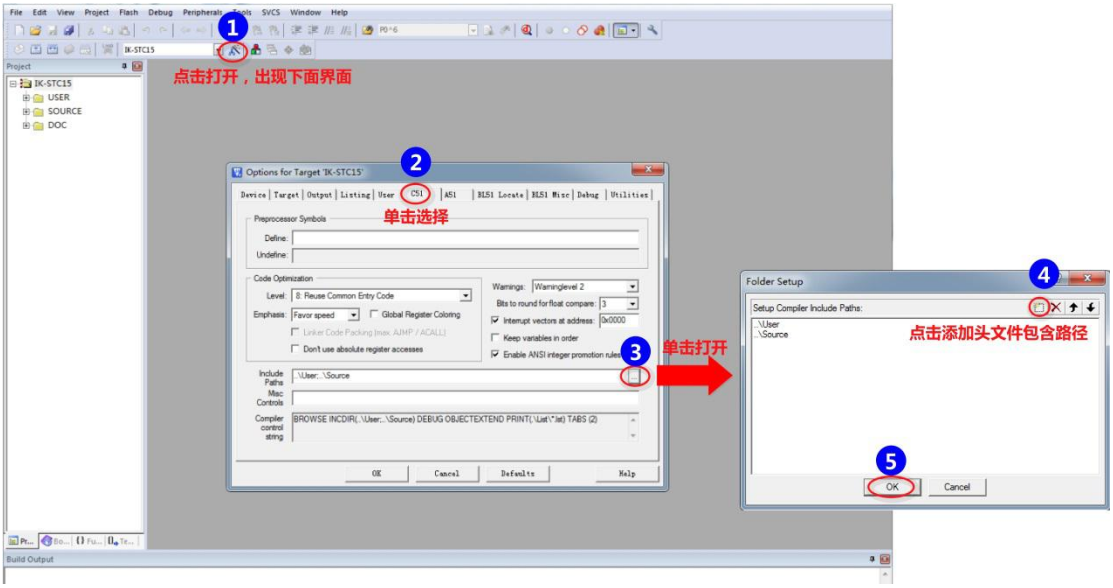

图 9：添加头文件包含路径

4.2.3. 编写代码

首先，在 exint.c 文件中编写外部中断 0 初始化函数 INT0_Init，代码如下。

程序清单：外部中断 0 初始化函数

```
/*************************************************************
功能描述：外部中断 0 初始化
入口参数：无
返回值：无
*************************************************************/
void INT0_Init(void)
{
    IE0  = 0;          //将 INT0 中断请求标志位清"0"
    EX0 = 1;           //使能 INT0 中断允许位
    IT0 = 1;           //选择 INT0 为下降沿触发方式
}
```

然后，编写外部中断服务函数，一旦进入中断会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：外部中断服务程序
入口参数：无
返回值：无
*************************************************************/
void INT0_Isr (void) interrupt INT0_VECTOR
{
    led_toggle(LED_1);         //翻转蓝色指示灯 DS1
}
```

最后，在主函数中对 P0.6 口进行模式配置，调用 INT0 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////////////
```

```
9.      P0M1 &= 0x3F;   P0M0 &= 0x3F;      //设置 P0.6、P0.7 为准双向口
10.
11.     INT0_Init();      //外部中断 0 的初始化配置
12.     EA = 1;             //允许总中断
13.
14.   while (1)
15.     {
16.         ;               //无任务，说明 LED 亮灭来自于中断
17.     }
18. }
```

4.2.4. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，使用 P3.2（外部中断 0 引脚）连接到按键 S3，因此需要按下图所示连接杜邦线和短路帽。

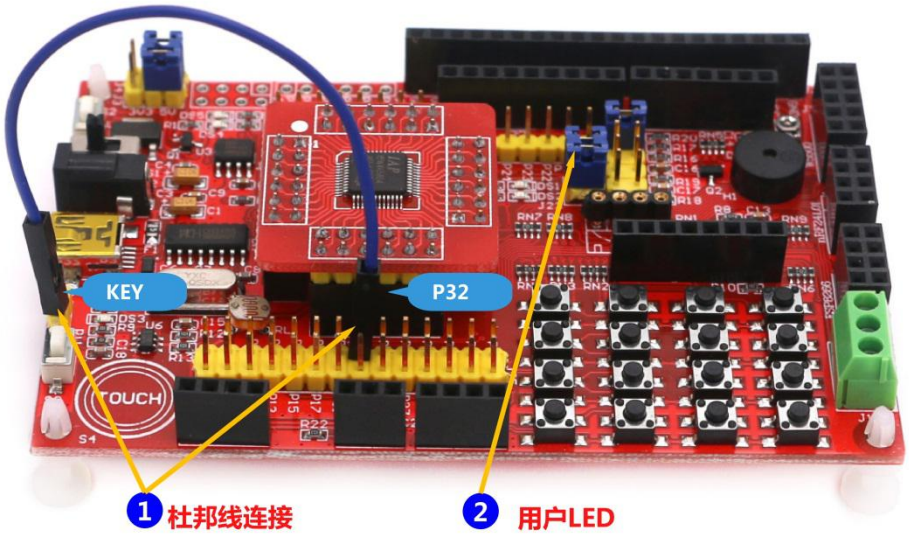

图 10：开发板连接图

4.2.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-4-1：外部中断 0（下降沿中断方式）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\INT0\projec”目录下的工程“INT0.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程

序，直到编译成功为止。编译后生成的 HEX 文件“INT0.hex”位于工程目录下的“Output”文件夹中。

5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，按下一次用户按键 S3 可观察到蓝色 LED 灯状态翻转一次（由亮变灭或由灭变亮）。

4.3.外部中断 0 实验（上升沿或下降沿中断方式）

✧ 注：本节的实验源码是在“实验 2-4-1：外部中断 0（下降沿中断方式）”的基础上修改。本节对应的实验源码是：“实验 2-4-2：外部中断 0（上升沿或下降沿中断方式）”。

4.3.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-4-1：外部中断 0（下降沿中断方式）”部分。

4.3.2. 编写代码

首先，在 exint.c 文件中编写外部中断 0 初始化函数 INT0_Init，代码如下。

程序清单：外部中断 0 初始化函数

```
/*************************************************************
功能描述：外部中断 0 初始化
入口参数：无
返回值：无
*************************************************************/
void INT0_Init(void)
{
    IE0  = 0;          //将 INT0 中断请求标志位清"0"
    EX0 = 1;           //使能 INT0 中断允许位
    IT0 = 0;           //选择 INT0 为上升沿或下降沿触发方式
}
```

然后，编写外部中断服务函数，一旦进入中断会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：外部中断服务程序
入口参数：无
返回值：无
*************************************************************/
void INT0_Isr (void) interrupt INT0_VECTOR
{
    led_toggle(LED_1);         //翻转蓝色指示灯 DS1
```

```
}
```

最后，在主函数中对 P0.6 口进行模式配置，调用 INT0 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////////////
    P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口

    INT0_Init();     //外部中断 0 的初始化配置
    EA = 1;            //允许总中断

  while (1)
    {
        ;              //无任务，说明 LED 亮灭来自于中断
    }
}
```

4.3.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，使用 P3.2（外部中断 0 引脚）连接到按键 S3，因此需要按下图所示连接杜邦线和短路帽。

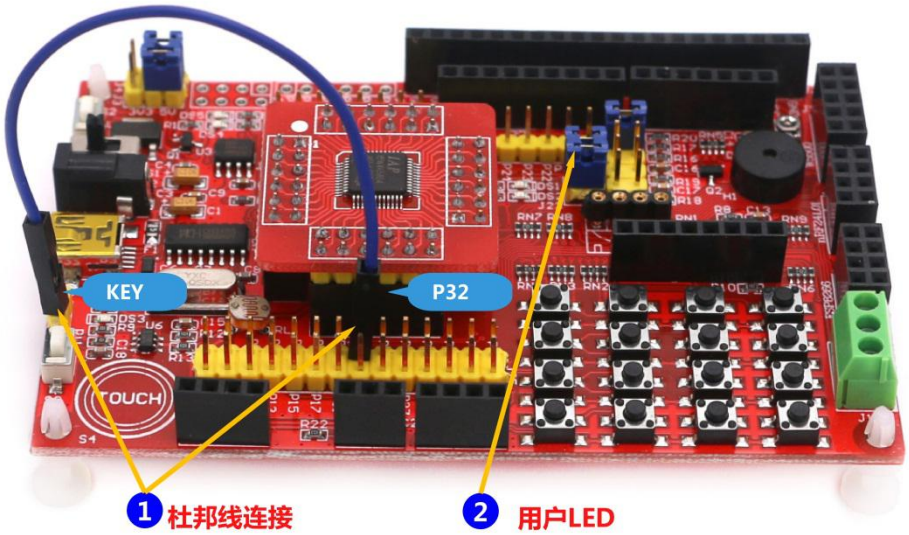

图 11：开发板连接图

4.3.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-4-2：外部中断 0（上升沿或下降沿中断方式）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\INT0\projec”目录下的工程“INT0.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“INT0.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，按下用户按键 S3 不松开，可观察到蓝色 LED 灯由亮变灭；之后松开用户按键 S3，蓝色 LED 灯由灭变亮。

■ **思考题：本例中的实验现象和“实验 2-4-1：外部中断 0（下降沿中断方式）”有何不同，为什么？**

4.4.外部中断 1 实验（下降沿中断方式）

✧ 注：本节的实验源码是在“实验 2-4-1：外部中断 0（下降沿中断方式）”的基础上修改。本节对应的实验源码是：“实验 2-4-3：外部中断 1（下降沿中断方式）”。

4.4.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-4-1：外部中断 0（下降沿中断方式）”部分。

4.4.2. 编写代码

首先，在 exint.c 文件中编写外部中断 1 初始化函数 INT1_Init，代码如下。

程序清单：外部中断 1 初始化函数

```
/*************************************************************
功能描述：外部中断 1 初始化
入口参数：无
返回值：无
*************************************************************/
void INT1_Init(void)
{
    IE1  = 0;          //将 INT1 中断请求标志位清"0"
    EX1 = 1;           //使能 INT1 中断允许位
    IT1 = 1;           //选择 INT1 为下降沿触发方式
}
```

然后，编写外部中断服务函数，一旦进入中断会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：外部中断服务程序
入口参数：无
返回值：无
*************************************************************/
void INT1_Isr (void) interrupt INT1_VECTOR
{
    led_toggle(LED_1);         //翻转蓝色指示灯 DS1
}
```

最后，在主函数中对 P0.6 口进行模式配置，调用 INT1 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
```

```
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////////////
    P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口

    INT1_Init();     //外部中断 1 的初始化配置
    EA = 1;            //允许总中断

  while (1)
    {
        ;              //无任务，说明 LED 亮灭来自于中断
    }
}
```

4.4.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，使用 P3.3（外部中断 1 引脚）连接到按键 S3，因此需要按下图所示连接杜邦线和短路帽。

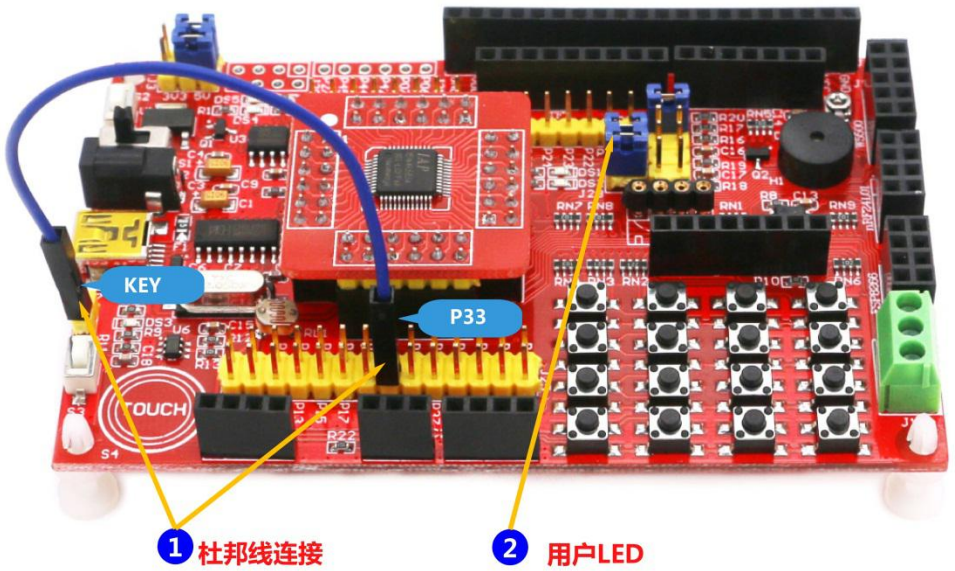

图 12：开发板连接图

4.4.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-4-3：外部中断 1（下降沿中断方式）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。

3. 在 Keil C51 中执行“Project→Open Project”打开“…\INT1\projec”目录下的工程“INT1.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“INT1.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，按下用户按键 S3 不松开，可观察到蓝色 LED 灯状态翻转一次（由亮变灭或由灭变亮）。

4.5.外部中断 1 实验（上升沿或下降沿中断方式）

✧ 注：本节的实验源码是在“实验 2-4-3：外部中断 1（下降沿中断方式）”的基础上修改。本节对应的实验源码是：“实验 2-4-4：外部中断 1（上升沿或下降沿中断方式）”。

4.5.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-4-1：外部中断 0（下降沿中断方式）”部分。

4.5.2. 编写代码

首先，在 exint.c 文件中编写外部中断 1 初始化函数 INT1_Init，代码如下。

程序清单：外部中断 1 初始化函数

```
/*************************************************************
功能描述：外部中断 1 初始化
入口参数：无
返回值：无
*************************************************************/
void INT1_Init(void)
{
    IE1  = 0;          //将 INT1 中断请求标志位清"0"
    EX1 = 1;           //使能 INT1 中断允许位
    IT1 = 0;          //选择 INT1 为上升沿或下降沿触发方式
}
```

然后，编写外部中断服务函数，一旦进入中断会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：外部中断服务程序
入口参数：无
返回值：无
```

```
5.  ************************************************************/
6.  void INT1_Isr (void) interrupt INT1_VECTOR
7.  {
8.      led_toggle(LED_1);          //翻转蓝色指示灯 DS1
9.  }
```

最后，在主函数中对 P0.6 口进行模式配置，调用 INT1 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
1.  int main(void)
2.  {
3.  ///////////////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ///////////////////////////////////////////////////////////
9.      P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口
10.
11.     INT1_Init();     //外部中断 1 的初始化配置
12.     EA = 1;            //允许总中断
13.
14.   while (1)
15.     {
16.         ;              //无任务，说明 LED 亮灭来自于中断
17.     }
18. }
```

4.5.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，使用 P3.3（外部中断 1 引脚）连接到按键 S3，因此需要按下图所示连接杜邦线和短路帽。

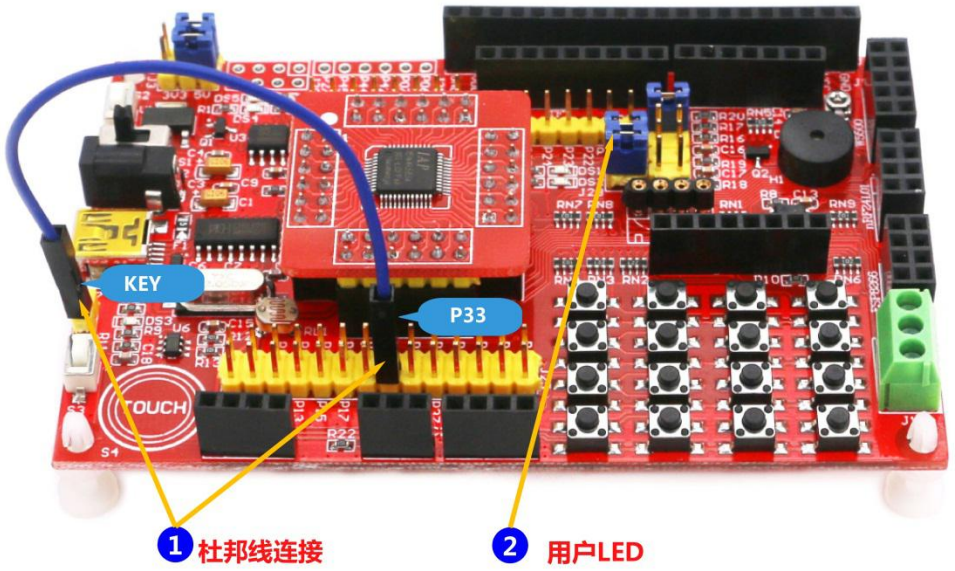

图 13：开发板连接图

4.5.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-4-4：外部中断 1（上升沿或下降沿中断方式）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\INT1\projec”目录下的工程“INT1.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“INT1.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，按下用户按键 S3 不松开，可观察到蓝色 LED 灯由亮变灭；之后松开用户按键 S3，蓝色 LED 灯由灭变亮。

4.6. 外部中断 2 实验（下降沿中断方式）

✧ 注：本节的实验源码是在“实验 2-4-1：外部中断 0（下降沿中断方式）”的基础上修改。本节对应的实验源码是：“实验 2-4-5：外部中断 2（下降沿中断方式）”。

4.6.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-4-1：外部中断 0（下降沿中断方式）”部分。

4.6.2. 编写代码

首先，在 exint.c 文件中编写外部中断 2 初始化函数 INT2_Init，代码如下。

程序清单：外部中断 2 初始化函数

```
1.  int main(void)
2.  /************************************************************
3.  功能描述：外部中断 2 初始化
4.  入口参数：无
5.  返回值：无
6.  ************************************************************/
7.  void INT2_Init(void)
8.  {
9.      AUXR2 |= 0x10;    //使能 INT2 中断允许位
10. }
```

然后，编写外部中断服务函数，一旦进入中断会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
1.  /************************************************************
2.  功能描述：外部中断服务程序
3.  入口参数：无
4.  返回值：无
5.  ************************************************************/
6.  void INT2_Isr (void) interrupt INT2_VECTOR
7.  {
8.      led_toggle(LED_1);         //翻转蓝色指示灯 DS1
9.  }
```

最后，在主函数中对 P0.6 口进行模式配置，调用 INT2 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
1.  int main(void)
2.  {
3.  ///////////////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ///////////////////////////////////////////////////////////
9.      P0M1 &= 0x3F;   P0M0 &= 0x3F;      //设置 P0.6、P0.7 为准双向口
```

```
10.
11.     INT2_Init();      //外部中断 2 的初始化配置
12.     EA = 1;           //允许总中断
13.
14.   while (1)
15.     {
16.         ;               //无任务，说明 LED 亮灭来自于中断
17.     }
18. }
```

4.6.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，使用 P3.6（外部中断 2 引脚）连接到按键 S3，因此需要按下图所示连接杜邦线和短路帽。

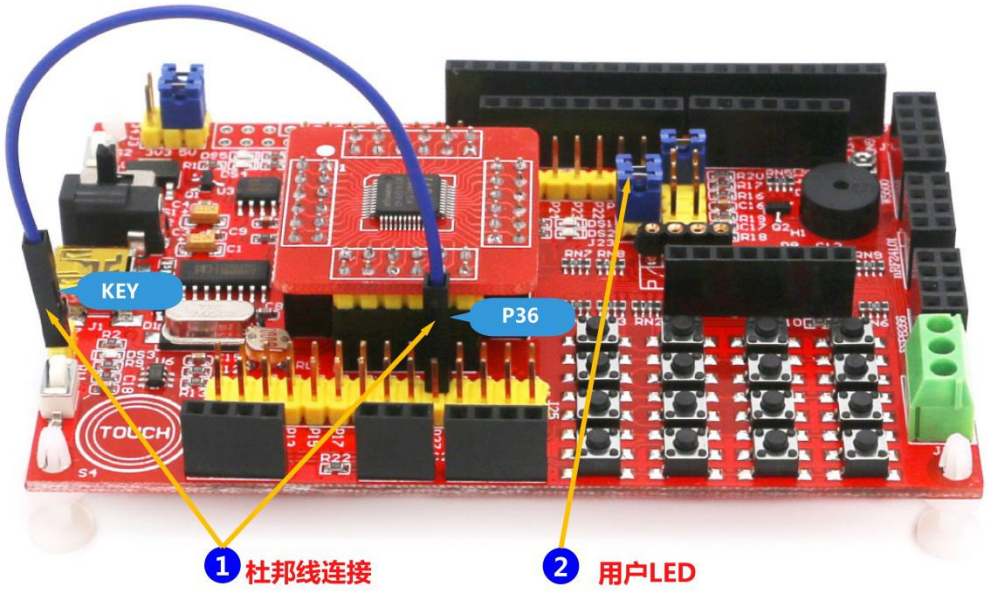

图 14：开发板连接图

4.6.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-4-2：外部中断 0（上升沿或下降沿中断方式）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\INT2\projec”目录下的工程“INT2.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“INT2.hex”位于工程目录下的“Output”

文件夹中。

5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，按下用户按键 S3 不松开，可观察到蓝色 LED 灯状态翻转一次（由亮变灭或由灭变亮）。

4.7. 多个外部中断实验

✧ 注：本节的实验源码是在“实验 2-4-1：外部中断 0（下降沿中断方式）”的基础上修改。本节对应的实验源码是：“实验 2-4-6：多个外部中断”。

4.7.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-4-1：外部中断 0（下降沿中断方式）”部分。

4.7.2. 编写代码

首先，在 exint.c 文件中编写外部中断 0 初始化函数 INT0_Init 和外部中断 1 初始化函数 INT1_Init，代码如下。

程序清单：外部中断 0 初始化函数

```
/*************************************************************
功能描述：外部中断 0 初始化
入口参数：无
返回值：无
*************************************************************/
void INT0_Init(void)
{
    IE0 = 0;          //将 INT0 中断请求标志位清"0"
    EX0 = 1;          //使能 INT0 中断允许位
    IT0 = 1;            //选择 INT0 为下降沿触发方式
    PX0 = 0;          //将 INT0 优先级设置为低优先级
}
```

程序清单：外部中断 1 初始化函数

```
/*************************************************************
功能描述：外部中断 1 初始化
入口参数：无
返回值：无
*************************************************************/
void INT1_Init(void)
{
    IE1  = 0;         //将 INT1 中断请求标志位清"0"
    EX1 = 1;          //使能 INT1 中断允许位
```

```
    IT1 = 1;            //选择 INT1 为下降沿触发方式
    PX1 = 1;          //将 INT1 优先级设置为高优先级
}
```

然后，编写外部中断服务函数，一旦进入外部中断 0 会执行翻转 5 次蓝色指示灯 DS1 的任务，进入外部中断 1 会执行翻转 4 次红色指示灯 DS2 的任务，代码如下。

程序清单：中断服务函数

```
/************************************************************
功能描述：外部中断服务程序
入口参数：无
返回值：无
************************************************************/
void INT0_Isr (void) interrupt INT0_VECTOR
{
    uint8 i;
    for(i=0;i<5;i++)
    {
        led_toggle(LED_1);        //翻转蓝色指示灯 DS1
        delay_ms(200);
    }
}
```

程序清单：中断服务函数

```
/************************************************************
功能描述：外部中断服务程序
入口参数：无
返回值：无
************************************************************/
void INT1_Isr (void) interrupt INT1_VECTOR
{
    led_on(LED_2);          //点亮红色指示灯 DS2
    delay_ms(200);
    led_off(LED_2);         //熄灭红色指示灯 DS2
    delay_ms(200);
    led_on(LED_2);          //点亮红色指示灯 DS2
    delay_ms(200);
    led_off(LED_2);         //熄灭红色指示灯 DS2
    delay_ms(200);
    led_on(LED_2);          //点亮红色指示灯 DS2
    delay_ms(200);
```

```
    led_off(LED_2);         //熄灭红色指示灯 DS2
    delay_ms(200);
    led_on(LED_2);          //点亮红色指示灯 DS2
    delay_ms(200);
    led_off(LED_2);         //熄灭红色指示灯 DS2
    delay_ms(200);
}
```

最后，在主函数中对 P0.6、P0.7 口进行模式配置，调用 INT0 和 INT1 初始化函数，开启总中断。在主循环中没有任务，蓝色指示灯 DS1 和红色指示灯 DS2 变化来自于中断。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////////////
  P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口

  INT0_Init();     //外部中断 0 的初始化配置
  INT1_Init();     //外部中断 1 的初始化配置
  EA = 1;            //允许总中断

  while (1)
    {
        ;              //无任务，说明 LED 亮灭来自于中断
    }
}
```

4.7.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，使用 P0.7 驱动红色指示灯 DS2，使用 P3.2（外部中断 0 引脚）连接到按键 S3，使用 P3.3（外部中断 1 引脚）连接到触摸按键 S4，因此需要按下图所示连接杜邦线和短路帽。

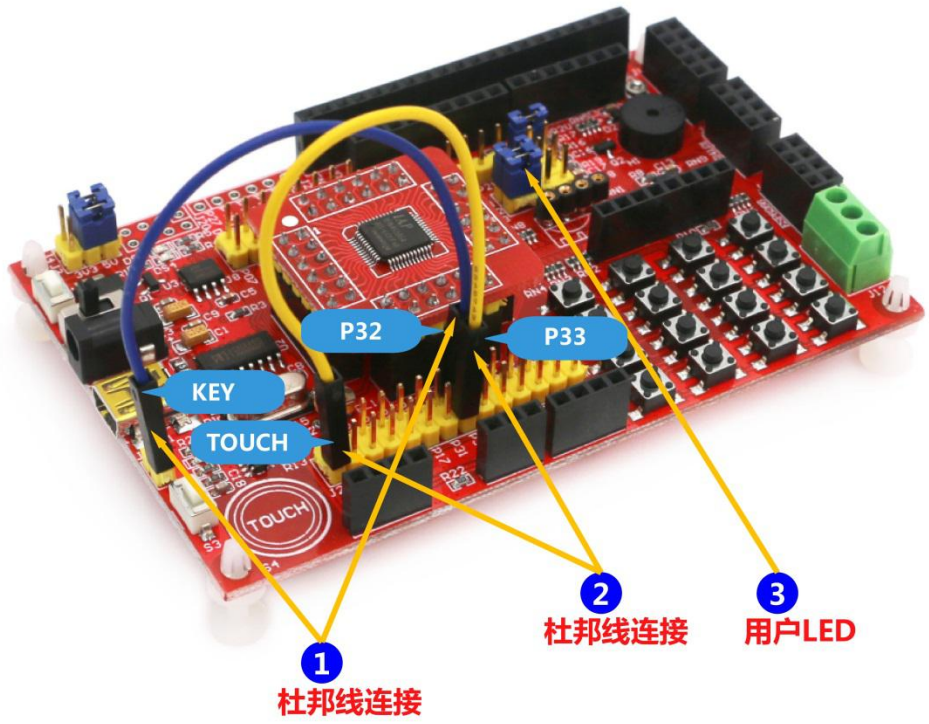

图 15：开发板连接图

4.7.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-4-6：多个外部中断”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\INTs\projec”目录下的工程“INTs.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“INTs.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，请按下面步骤操作：

   1）按下用户按键 S3，可观察蓝色指示灯 DS1 闪烁 5 次；

   2）按下触摸按键 S4，可观察红色指示灯 DS2 闪烁 4 次；

   3）按下用户按键 S3，在蓝色指示灯 DS1 闪烁 1 次时按下触摸按键 S4，可观察蓝色指示灯 DS1 停止闪烁，待红色指示灯 DS2 闪烁 4 次后蓝色指示灯 DS1 接着闪烁 4 次。

   4）按下触摸按键 S4，在红色指示灯 DS2 闪烁 1 次时按下用户按键 S3，可观察待红色指示灯 DS2 接着闪烁 3 次后蓝色指示灯 DS1 才开始闪烁 5 次。

■ **思考题：根据本例实验现象分析 INT0 和 INT1 谁的中断优先级高？为什么？**