实验 4　中断技术

一．实验目的

1. 了解中断原理，包括对中断源、中断向量、中断类型号、中断程序以及中断响应过程的理解；
2. 掌握单片机 C 语言中断程序设计方法。

二. 实验任务

1．中断响应过程的理解

阅读下面 C 语言中断程序 L4_int.C（见后页），说明程序执行的流程和实现功能。上机实践，**思考下面问题，**掌握用 C 语言编写中断程序的方法。

1) 从程序**如何**判断用的是哪个中断源？该中断的中断类型号是多少？将实验板上某一按键与该中断源对应的引脚相连，运行程序，操作按键，观察现象。

2）main 函数中无调用函数 port_ISR 的语句，函数 port_ISR 如何能被执行？何时会被执行？ 据此思考和理解中断响应过程。

3) 如果函数 port_ISR 中不清分中断标志 P2IFG 的后果是什么？为什么？

4) 函数 port_ISR 的名称可以用其他的符号，比如 KeyInt 吗？符号 PORT2_VECTOR 可用其他吗？为什么？

5) 如果 L4_int.c 中的 PORT2_VECTOR 改为 PORT1_VECTOR， 其他不变，程序执行的后果是什么？为什么？

6) 程序中的“P2OUT |=BIT1;　P2REN |=BIT1;”这两条语句可否不要？为什么？

7) 若中断源用 P1.3，按键用 K1，请连线，修改 L4_int.c 程序完成以中断方式响应 K1 的操作。

注意: 1) 查看 msp430G2553.h 文件末尾处有关中断向量偏址的符号定义。

方法是：在 main.c 中，按下 ctl 键，并用鼠标双击语句#include “msp430.h”，在打开的 msp430.h 中，同时按下 ctl+F 查找 g2553，找到语句 #include "msp430g2553.h"，然后再按下 ctl 键，并用鼠标点击该语句即可打开 msp430G2553.h 文件，按下 ctl+END 即可到文件末尾，向上翻看即可查看有关中断向量相关的定义。

2) 为便于了解程序执行流程，可在中断函数入口处设置一断点，然后连续运行程序（F5），观察操作和不操作按键两不同情况下程序执行的现象有何不同。思考为什么？

3) 观察在执行中断函数时操作按键，与在执行完中断函数后按键，现象有什么不同。思考为什么？

2. 中断程序编程练习

在实验板上用导线将按键 K2、K4 分别与单片机的 P1.2 和 P1.6 相连，编程以中断方式响应按键 K2 和 K4 的请求：当按下一次 K2 键，实验板上的发光二极管 L3 闪 3 次；当按下一次 K4 键，实验板上的蜂鸣器发出一声警报。主循环中控制 L8 循环闪亮。通过蜂鸣器、以及不同 LED 的状态了解程序的执行。

思考：

1) 若按键 K2、K4 分别连接在 P2.2 和 P2.6 上，如何修改程序以实现任务 2 功能？
2) 若按键 K2、K4 分别连接在 P1.2 和 P2.6 上，如何修改程序以实现任务 2 功能？

3. 采用事件标志处理中断

阅读程序 L4_intA.c 和 L4_intB.c（见后），描述其实现功能。回答：

1） 比较两种方法的不同，特别注意蜂鸣器响时，对按键响应的不同。
（注意实际按键次数与程序控制可能不同，原因可参看任务 4 的按键抖动。）

2） 理解程序用 define 宏定义按键、蜂鸣器引脚的益处，并改写 L4_intB 程序中对 LED 灯的部分。

4. （提高）按键抖动问题

程序 L4_Key.C 见后页， 其功能是用中断方式响应与 P1.2 连接的按键，计数按键的次数，并将所计的次数用 8 个发光二极管显示出来。运行该程序，并操作按键，观察实际操作的次数与显示值之间的关系。编程改进 L4_Key.C 程序，用软件方式去除按键抖动的影响。

说明：通常的按键所用开关为机械弹性开关，当机械触点断开、闭合时，由于机械触点的弹性作用，一个按键开关在闭合时不会马上稳定地接通，在断开时也不会一下子断开。因而在闭合(按下)及断开(释放)的瞬间均伴随有一连串的抖动，产生电压毛刺，见图4-1机械按键的电压变化图。在一次按键过程中，因为电压毛刺的产生，会有若干次下降沿和上升沿。采用下降沿判断时，只有第一次下降沿是真正的按键事件，其它是由于按键抖动带来的毛刺，不是按键事件。去除这些毛刺带来的影响，称为按键消抖或去抖。软件编程处理比较简单的方法是，在响应了第一次下降沿后，在中断函数中加入一定的延时，躲过其它电压毛刺的产生时间。

思考:

1.延时函数加在按键中断程序的什么位置？为什么？

2.延时函数的延时长短如何控制？简单的延时方法能完全解决抖动问题吗？

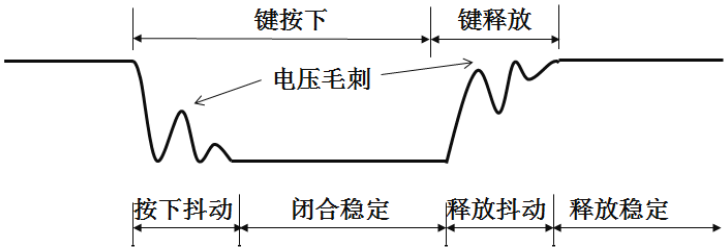

图 4-1 机械按键的电压变化图

注意：目前实验板使用的按键，抖动现象不是很严重。但可以通过上面分析，了解中断源信号对中断响应的影响，以及可以从中断函数的执行次数来反推中断源信号的稳定性，从而找到问题和解决问题的方法。以后在使用外部的一些传感器触发中断时，也会有类似的信号抖动问题，使用时需要注意。

5. （提高）红外避障模块的抖动研究

采用任务 4 的方法，检测红外避障模块的抖动问题，并做消抖处理。

6. 数字示波器的使用（为实验 5 做准备）

1）将信号源的波形在示波器上显示出来，掌握测量周期、频率、峰峰值的方法；

2）用导线将实验板的地信号与示波器的地信号相连，测量实验板上的 Vcc 电源信号是否正常。

L4_int.C 程序清单(提供电子文件)

```
#include "msp430.h"

void delay(unsigned int i)          //延时函数
{      unsigned int j;                    //定义局部变量
       for (j=0; j<i; j++);
}

void Blink( )                    //LED闪
{        P2OUT &=~BIT3;
         delay(0xf000);
         P2OUT |= BIT3;
         delay(0xf000);
}

void Buzz( )                      //蜂鸣响
{ unsigned int i;
   for (i=0; i<8; i++)
   {      P2OUT ^=BIT4;           //对引脚输出求反
          delay(0xf000);
   };
}

void main ( void )
{      WDTCTL = WDTPW + WDTHOLD;            //关闭看门狗

//设置引脚P2.4、P2.3输出，P2.3连接LED，P2.4连接蜂鸣器
       P2SEL &=~(BIT3+BIT4);
       P2SEL2 &=~(BIT3+BIT4);
       P2OUT|=(BIT3+BIT4);
       P2DIR|=(BIT3+BIT4);

//中断相关设置
       P2SEL &=~BIT1;
       P2SEL2 &=~BIT1;
       P2OUT |=BIT1;
       P2REN |=BIT1;
       P2DIR &=~BIT1;
       P2IES |= BIT1;
       P2IFG &=~BIT1;
       P2IE |= BIT1;

       _EINT();       //总中断允许

       for (;;)    //主循环
       {      Blink();       };
}

#pragma    vector=PORT2_VECTOR
__interrupt    void port_ISR( )
{    if ( (P2IFG&BIT1)!=0 )       //判断是P2.1上的中断
      Buzz();
       P2IFG &=~BIT1;
}
```

L4_intA.c 程序清单(提供电子文件):

```
#include "msp430.h"

//定义蜂鸣器连接的引脚和端口
#define buzz_PSEL         P2SEL
#define buzz_PSEL2        P2SEL2
#define buzz_PDIR         P2DIR
#define buzz_POUT         P2OUT
#define buzz_Pin          BIT4

//定义控制蜂鸣器响的按键次数
#define num_Buzz       6

//定义中断相关的引脚和端口
#define key_PSEL        P1SEL
#define key_PSEL2       P1SEL2
#define key_POUT        P1OUT
#define key_PREN        P1REN
#define key_PDIR        P1DIR
#define key_PIES        P1IES
#define key_PIFG        P1IFG
#define key_PIE         P1IE
#define key_Pin         BIT5
#define key_Vector      PORT1_VECTOR

unsigned int num_Key=0;     //计数按键次数

void delay(unsigned int i)     //延时函数
{     unsigned int j;
      for (j=0;j<i;j++);
}

void Buzz( )                   //蜂鸣响函数
{ unsigned int i;
   for (i=0;i<8;i++)
   {     buzz_POUT ^=buzz_Pin;
         delay(0x6000);
   };
}

void main ( void )
{     WDTCTL = WDTPW + WDTHOLD;          //关闭看门狗

//初始化 LED 引脚，基本输出，初值 LED 灭
      P2SEL &=~BIT0;
      P2SEL2 &=~BIT0;
      P2OUT|=BIT0;
      P2DIR|=BIT0;

//初始化蜂鸣器引脚，基本输出，初值蜂鸣器不响
      buzz_PSEL &=~buzz_Pin;
      buzz_PSEL2 &=~buzz_Pin;
      buzz_POUT|=buzz_Pin;
      buzz_PDIR|=buzz_Pin;

//中断引脚的相关设置
      key_PSEL &=~key_Pin;
      key_PSEL2 &=~key_Pin;
      key_POUT |=key_Pin;
      key_PREN |=key_Pin;
      key_PDIR &=~key_Pin;
      key_PIES |= key_Pin;
      key_PIFG &=~key_Pin;
      key_PIE |= key_Pin;

      _EINT();      //总中断允许

      while (1);   //主循环

}

#pragma    vector=key_Vector
__interrupt    void port_ISR( )
{          num_Key++;                          //按键次数加 1
           P2OUT^=BIT0;                        //LED 变化一次
           if (num_Key==num_Buzz)              //蜂鸣响的条件到
                 {   Buzz( );                  //蜂鸣响;
                     num_Key=0;                //按键次数回零，重新计数
                 };
            key_PIFG &=~key_Pin;               //清除中断标志
}
```

L4_intB.c 程序清单(提供电子文件)，红色部分是和 L4_intA.c 不同的地方:

```
#include "msp430.h"
#define buzz_PSEL       P2SEL     //定义蜂鸣器连接的引脚和端口
#define buzz_PSEL2      P2SEL2
#define buzz_PDIR       P2DIR
#define buzz_POUT       P2OUT
#define buzz_Pin        BIT4
#define num_Buzz        6         //定义控制蜂鸣器响的按键次数
#define key_PSEL        P1SEL     //定义中断相关的引脚和端口
#define key_PSEL2       P1SEL2
#define key_POUT        P1OUT
#define key_PREN        P1REN
#define key_PDIR        P1DIR
#define key_PIES        P1IES
#define key_PIFG        P1IFG
#define key_PIE         P1IE
#define key_Pin         BIT5
#define key_Vector      PORT1_VECTOR
unsigned int num_Key=0;       //计数按键次数
unsigned int flag_Buzz=0;     //蜂鸣器响标志

void delay(unsigned int i)       //延时函数
{   unsigned int j;      for (j=0; j<i; j++);   }

void Buzz( )                     //蜂鸣响函数
{   unsigned int i;
    for ( i=0;i<8;i++ ) {   buzz_POUT ^=buzz_Pin; delay(0x6000); };
}

void main ( void )
{    WDTCTL = WDTPW + WDTHOLD;          //关闭看门狗

     P2SEL &=~BIT0;                     //初始化 LED 引脚，基本输出，初值 LED 灭
     P2SEL2 &=~BIT0;
     P2OUT|=BIT0;
     P2DIR|=BIT0;

     buzz_PSEL &=~buzz_Pin;             //初始化蜂鸣器引脚，基本输出，初值蜂鸣器不响
     buzz_PSEL2 &=~buzz_Pin;
     buzz_POUT|=buzz_Pin;
     buzz_PDIR|=buzz_Pin;

     key_PSEL &=~key_Pin;               //中断引脚的相关设置
     key_PSEL2 &=~key_Pin;
     key_POUT |=key_Pin;
     key_PREN |=key_Pin;
     key_PDIR &=~key_Pin;
     key_PIES |= key_Pin;
     key_PIFG &=~key_Pin;
     key_PIE |= key_Pin;
     _EINT();                           //总中断允许

     while (1)                          //主循环
     {    if ( flag_Buzz==1 )           //蜂鸣器响标志成立
          {      Buzz();                //蜂鸣响
                 flag_Buzz=0;           //清蜂鸣器响标志
          };
      }
}

#pragma   vector=key_Vector
__interrupt   void port_ISR( )
{        num_Key++;                          //按键次数加 1
         P2OUT^=BIT0;                        //LED 变化一次
         if (num_Key==num_Buzz)              //蜂鸣响的条件到
               {   //Buzz( );                 //蜂鸣响;
                   flag_Buzz=1;              //设置蜂鸣器响标志
                    num_Key=0;               //按键次数回零，重新计数
               };
          key_PIFG &=~key_Pin;               //清除中断标志
}
```

L4_Key.c 程序清单(提供电子文件):

```
#include "msp430.h"
unsigned int number=0;              //记录响应按键次数
int main( void )
{       WDTCTL = WDTPW + WDTHOLD;           //关闭看门狗
        __disable_interrupt();          //_DINT(); 禁止总中断
        P2SEL=0;                    //置 P2 为基本 I/O 功能
        P2SEL2=0;                       //
        P2OUT=0xFF;                 //置 P2 输出的初值
        P2DIR=0xFF;                     //置 P2 为输出方向
        P1SEL &= BIT2;
        P1SEL2 &= BIT2;
        P1OUT |=BIT2;;
        P1REN |=BIT2;
        P1DIR &=~BIT2;
        P1IES |= BIT2;
        P1IFG &=~BIT2;
        P1IE |= BIT2;
        __enable_interrupt();           //_EINT(); 总中断运行
        while(1){ };
}
#pragma    vector=PORT1_VECTOR
__interrupt void port_int(void)
{       unsigned int i;
        if ( (P1IFG&BIT2)!=0 )
        {       number++;
                P2OUT=~number;
        }
        P1IFG &=~BIT2;
}
```