**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。
3. **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pin\_ISR的语句，函数pin\_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) 请试一试函数pin\_ISR中不清分中断标志P1IFG的后果是什么？分析为什么？

4) 函数pin\_ISR的名称可以用其他的符号，比如 KeyInt吗？符号PORT1\_VECTOR可用其他的符号吗？为什么？

5) 如果L4\_int.c中的PORT1\_VECTOR改为PORT2\_VECTOR， 其他不变，程序执行的后果是什么？为什么？

6) 程序中的“P1OUT |=BIT5; P1REN |=BIT5;”这两条语句可否不要？为什么？

7) 若中断源用P2.3，按键用K2，请连线，修改L4\_int.c程序完成以中断方式响应K2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，了解符号与中断类型号的对应关系。

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，同时按下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符号的定义。

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（(F8，Resume），观察操作和不操作按键，两种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**执行中断函数中**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K1、K3分别与单片机的P1.5和P1.6相连，发光二极管L7和L2、蜂鸣器buzz分别与P2.2、P2.3、P2.4连接。编程以中断方式响应按键K1和K3的请求：当按下一次K1键，发光二极管L2闪3次；当按下一次K3键，蜂鸣器发出3声警报。主循环中发光二极管L7循环闪亮。通过蜂鸣、以及不同LED的闪烁状态了解程序的执行。

***思考：*** 1) 按键K1、K3分别连接在P2.5和P2.6上，如何修改程序以实现任务2功能？

1. 按键K1、K3分别连接在P1.5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（见后页），描述其实现功能。回答：

1. 比较两种方法有什么不同，为什么？(提示：注意蜂鸣器响时，对按键是否响应，并分析。)

（操作时，**实际按键次数**与**程序中要求的次数**可能不同，原因可参看任务4的**按键抖动**说明）

1. 比较在L4\_intB.c的主循环中，把清除FLag\_Buzz标志放在调用Buzz()函数前和后，对按键响应有什么不同，为什么？
2. 理解程序用define宏定义按键、蜂鸣器引脚的益处，用define改写程序中LED灯的相关代码。

**4. (提高)按键抖动处理**

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3-1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？

3.除延时方法去抖外，有没有其他的方法可去抖？可上网查找相关资料了解其他处理方法。

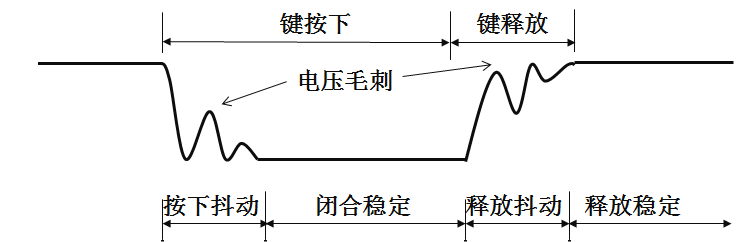


图3-1 机械按键的电压变化图

**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，用红外避障模块的输出信号，作为中断申请信号，检测红外避障模块的抖动问题，并做消抖处理。

1. **数字示波器的使用(为明天实验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 &=~BIT2;

delay(0x8000);

P2OUT |= BIT2;

delay(0x8000);

}

void Buzz( ) //蜂鸣响，声音高低、大小适当

{ unsigned int i;

for ( i=0; i<6; i++)

{

P2OUT &=~BIT4; //响

delay(0x100); //响的时长

P2OUT |=BIT4;; //停响

delay(0xfff0); //停响的时长

};

}

void main ( void )

{ WDTCTL = WDTPW + WDTHOLD; //关闭看门狗

//设置引脚P2.4、P2.2输出，P2.2连接LED，P2.4连接蜂鸣器

P2SEL &=~(BIT2+BIT4);

P2SEL2 &=~(BIT2+BIT4);

P2OUT|=(BIT2+BIT4); //设置引脚输出的初始值

P2DIR|=(BIT2+BIT4);

//中断相关设置

P1SEL &=~BIT5;

P1SEL2 &=~BIT5;

P1OUT |=BIT5;

P1REN |=BIT5;

P1DIR &=~BIT5;

P1IES |= BIT5;

P1IFG &=~BIT5;

P1IE |= BIT5; //分中断允许

\_EINT(); //总中断允许

for (;;) //主循环，看到LED闪，表示在执行主循环部分

{ Blink(); };

}

#pragma vector=PORT1\_VECTOR //设置端口P1中断向量

\_\_interrupt void pin\_ISR( ) //中断函数

{ if ( ( P1IFG&BIT5)!=0 ) //判断中断具体来自哪个引脚

{ Buzz(); //听到蜂鸣器响，表示在执行中断函数

P1IFG &=~BIT5; //清除对应的中断标志

}

}

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 BIT5

//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 BIT4

#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，声音高低、大小适当

{ unsigned int i;

for ( i=0; i<3; i++)

{

buzz\_POUT &=~buzz\_Pin; //响

delay(0x100); //响的时长

buzz\_POUT |=buzz\_Pin; //停响

delay(0xfff0); //停响的时长

};

}

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 pin\_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"

//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 BIT5

//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 BIT4

#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，声音高低、大小适当

{ unsigned int i;

for ( i=0; i<3; i++)

{ buzz\_POUT &=~buzz\_Pin; //响

delay(0x100); //响的时长

buzz\_POUT |=buzz\_Pin; //停响

delay(0xfff0); //停响的时长

};

}

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; //分中断允许

\_EINT(); //总中断允许

while (1) //主循环

{

**if (flag\_Buzz==1) //蜂鸣器响标志成立**

**{ // flag\_Buzz=0; //清蜂鸣器响标志**

**Buzz(); //蜂鸣响**

**flag\_Buzz=0; //清蜂鸣器响标志**

**};**

}

}

#pragma vector=key\_Vector

\_\_interrupt void pin\_ISR( )

{ num\_Key++; //按键次数加1

P2OUT^=BIT0; //LED变化一次

if (num\_Key==num\_Buzz) //蜂鸣器响的条件到

{ **//Buzz( ); //蜂鸣器响;**

**flag\_Buzz=1; //清蜂鸣器响标志**

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

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;

\_EINT(); //总中断允许

while(1){ };

}

#pragma vector=PORT1\_VECTOR

\_\_interrupt void port\_int(void)

{ unsigned int i;

if ( ( P1IFG&BIT2)!=0 )

{ number++;

P2OUT=~number;

}

P1IFG &=~BIT2;

}