实验4-任务1

1. 可以查看断源向量的设置;该中断的中断类型号是2;将K6按键和对应引脚相连后，按下按钮蜂鸣器响起。
2. 在检测到P2的中断请求时，port\_ISR由硬件自动执行；中断响应并不体现在程序的主函数中，而是由硬件来执行。
3. 会导致重复进入中断程序，无法正常运行主程序；在中断子程结束后，运行主程序时因为没有清中断标志，又会再进入中断程序。
4. port\_ISR可以变，但PORT2\_VECTOR不可变，因为前者只是一个函数名，而后者是头文件中已经定义好的名称。
5. 程序会跑飞，因为中断向量的端口位置与实际端口位置不一样，PC指针不指向自己编写程序。
6. 不可以，因为需要一个上拉电阻才会有输入的电位变化。
7. 修改程序如下:

**#include** "msp430.h"

**void** **delay**(**unsigned** **int** i)

{ **unsigned** **int** j;

**for** ( j=0; j<i; j++ );

}

**void** **Blink**( )

{ P2OUT &=~BIT2;

delay(0x8000);

P2OUT |= BIT2;

delay(0x8000);

}

**void** **Buzz**( )

{ **unsigned** **int** i;

**for** ( i=0; i<5; i++)

{

P2OUT &=~BIT4;

delay(0x100);

P2OUT |=BIT4;;

delay(0xfff0);

};

}

**void** **main** ( **void** )

{ WDTCTL = WDTPW + WDTHOLD;

\_DINT();

P2SEL &=~(BIT2+BIT4);

P2SEL2 &=~(BIT2+BIT4);

P2OUT|=(BIT2+BIT4);

P2DIR|=(BIT2+BIT4);

P2SEL &=~BIT3;

P2SEL2 &=~BIT3;

P2OUT |=BIT3;

P2REN |=BIT3;

P2DIR &=~BIT3;

P2IES |= BIT3;

P2IFG &=~BIT3;

P2IE |= BIT3;

\_EINT();

**for** (;;)

{ Blink(); };

}

**#pragma** vector=PORT2\_VECTOR

**\_\_interrupt** **void** **pin\_ISR**( )

{ **if** ( ( P2IFG&BIT3)!=0 )

{ Buzz();

P2IFG &=~BIT3;

}

}

实验4-任务2

**#include** "msp430.h"

**unsigned** **int** k;

**void** **delay**(**unsigned** **int** i)

{ **unsigned** **int** j;

**for** ( j=0; j<i; j++ );

}

**void** **BlinkL7**( )

{

P2OUT &=~BIT2;

delay(0x8000);

P2OUT |= BIT2;

delay(0x8000);

}

**void** **BlinkL2**( )

{

**for** (k=0;k<3;k++)

{

P2OUT &=~BIT3;

delay(0x8000);

P2OUT |= BIT3;

delay(0x8000);

}

}

**void** **Buzz**( )

{

**for** ( k=0; k<3; k++)

{

P2OUT &=~BIT4;

delay(0x100);

P2OUT |=BIT4;;

delay(0xfff0);

};

}

**void** **main** ( **void** )

{ WDTCTL = WDTPW + WDTHOLD;

\_DINT();

P2SEL &=~(BIT2+BIT3+BIT4);

P2SEL2 &=~(BIT2+BIT3+BIT4);

P2OUT|=(BIT2+BIT3+BIT4);

P2DIR|=(BIT2+BIT3+BIT4); //L7 L2 bell

P1SEL &=~BIT5+BIT6;

P1SEL2 &=~BIT5+BIT6;

P1OUT |=BIT5+BIT6;

P1REN |=BIT5+BIT6;

P1DIR &=~BIT5+BIT6; //K1 K3

P1IES |= BIT5+BIT6;

P1IFG &=~(BIT5+BIT6);

P1IE |= BIT5+BIT6;

\_EINT();

**for** (;;)

{ BlinkL7(); };

}

**#pragma** vector=PORT1\_VECTOR

**\_\_interrupt** **void** **pin\_ISR**( )

{ **if** ( ( P1IFG&BIT5)!=0 )

{ BlinkL2();

P1IFG &=~BIT5;

}

**else** **if** (( P1IFG&BIT6)!=0)

{

Buzz( );

P1IFG &=~BIT6;

}

}

实验4-任务3

实现功能：每按1次按钮LED灯变化一次；每按4次按钮蜂鸣器响3下。

1)区别在于intA的蜂鸣器响在中断程序中执行，intB的蜂鸣器响在主程序中执行，因此在蜂鸣器响的时候按按钮，intA不会受按钮影响，intB会受按钮影响。

2)若把清除功能放在前面，则若在蜂鸣器响的过程中又按了4次按钮，则蜂鸣器会额外响3次；若把清除功能放在后面，则若在蜂鸣器响的过程中又按了4次按钮，蜂鸣器不会额外响3次。

3)define使得改写蜂鸣器、按键连接的接口变得方便。

LED部分的改写如下：

#include "msp430.h"

#define LED\_PSEL P2SEL

#define LED\_PESL2 P2SEL2

#define LED\_OUT P2OUT

#define LED\_DIR P2DIR

#define LED\_Pin BIT0

//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

LED\_SEL &=~LED\_Pin;

LEDSEL2 &=~ LED\_Pin;

LEDOUT |= LED\_Pin;

LEDDIR |= LED\_Pin;

//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

{

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) //蜂鸣响的条件到

{ //Buzz( ); //蜂鸣响;

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

}}