

实验名称：中断技术、基本时钟和定时功能（实验 4、5）

姓名____ 学号____
实验班号 21 机器号__

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. 了解 MSP430G2553 基本时钟模块的工作原理，掌握其控制方法；
4. 掌握利用时钟信号和中断技术实现定时功能的方法

二、实验基本任务

1. 中断响应过程的理解

阅读下面 C 语言中断程序，说明程序 [L4_int.c](#) 执行的流程和实现功能。上机实践，回答下面问题，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？其中断类型号是多少？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。

答：（注：源程序主函数中第 7、8 行有误，应为 `P1SEL &=~ BIT5; P1SEL2 &=~ BIT5;` ）

① P1.1 为中断源，中断类型号为 2；

② 现象：LED 灯一直在闪烁，按下按键后闪烁暂停，蜂鸣器响三声，然后 LED 灯继续闪烁；

2) main 函数中无调用函数 Buzz 的语句，函数 Buzz 如何能被执行？何时会被执行？据此描述中断响应过程。

答：

① 当按下按键时函数 Buzz 被执行；

② 中断响应的过程：P1.1 处发出中断请求→判断是否满足响应条件→若满足，则 CPU 在执行完当前指令后，硬件自动完成保护现场的操作→从中断向量表中取中断向量至 PC→转去执行中断服务子程；

3) 如果 port_int 函数中不清分中断标志 P1IFG 的后果是什么？

答：中断将一直重复进行下去；

4) 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？（可在中断处加一断点，运行程序，看现象，分析原因）

答：PORT1_VECTOR 改为 PORT2_VECTOR，其他不变，程序将会无法进入中断。因为程序中的中断属于 P1 引脚的中断，中断向量与 P2 引脚的中断向量不同，所进行的改动则是把中断程序写入到了 P2 引脚的中断向量对应的地址中，而 P1 引脚的中断向量对应的地址上没有程序，因此无法执行原先的中断子程。

5) 如果中断源采用的是 P1.5，按键用 K7，请设计连线，修改程序完成以中断方式响应 K7 的操作。

答：

① 只需将程序中对 P1.1 的操作改为对 P1.5 的操作即可，程序见附录[程序 1](#)；

② 连线：将引脚与 K6 相连，其他连线不变；

2. 中断程序编程练习

在实验板上用跳线将按键 K5、K6 分别与单片机的 P1.4、P1.5 相连，编程以中断方式响应按键 K5 和 K6 的请求：当按一次 K5 键，实验板上的蜂鸣器发出一声警报声；当按下一次 K6 键，实验板上的发光二极管 L1 闪 3 次。主循环中控制 L7 循环闪亮。

思考：如果用长导线将按键 K5、K6 分别连接在 P2.2 和 P2.5 上，如何修改程序以实现任务 2 功能？

答：经过分析，

① 本程序需要有四个函数，分别是延时函数、L1 闪烁 3 次的函数、L7 循环闪烁、以及蜂鸣器响一声的函数；

② 其次 P2 端口为输出端口，分别控制 L1、L7 和蜂鸣器，本程序中选用 P2.1、P2.3、P2.4 引脚；P1.4 与 P1.5 作为中断源；

③ 应注意到本程序有两个中断源，须利用 P1IFG 判断产生中断的中断源引脚。

④ 程序见附录[程序 2](#)。

思考题：

如果用长导线将按键 K5、K6 分别连接在 P2.2 和 P2.5 上，则应设置 P2.2、P2.5 为中断源，设置中断向量时也应注意将对 P1 端口的操作改为对 P2 端口操作。具体程序见附录[程序 2 之思考题](#)。

3. 数字示波器的使用

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) 用孔孔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。

答：

1)

a. 信号源的峰峰值为 3.20V；

b. 周期为 1.000ms；

c. 频率为 1.000kHz.

2)经测量，Vcc=3.64V，正常

4. 测试上电复位系统的 ACLK、和 SMCLK 时钟频率

编程输出单片机上电复位后的 ACLK、和 SMCLK 时钟，用示波器测量其频率，并记录下来。

思考：上电复位后，CPU 工作的时钟信号 MCLK 频率值是多少？

答：经分析

① 本程序需要置引脚 P1.0、P1.4 分别输出 ACLK、SMCLK；

② 需要确认外部晶振连上；

③ 具体程序见附录[程序 5 测试程序](#)。

④ 测得 $f(\text{ACLK})=32.79\text{kHz}$, $f(\text{SMCLK})=1.044\text{MHz}$.

思考题

上电复位后，通过观察寄存器，发现 BCSCTL2 寄存器上 SELM 位为 00，SELS 位为 0，说明 MCLK 与 SMCLK 均由 DCO 振荡器控制，所以 MCLK 的频率与 SMCLK 相同，也为 1.044MHz。

5. 掌握基本时钟模块的编程控制

参看附录 A 实验板原理图，用跳线将 JP8 中的插针信号接到晶振 32.768Khz 侧，使晶振

与单片机的 P2.6 和 P2.7 相连。编程控制基本时钟模块，设置 ACLK 分别为下面时钟频

率，并通过 P1.0 输出 ACLK，用示波器观察：

1) ACLK=4096Hz；（时钟源外部晶振，32768Hz/8）

2) ACLK=3KHz；（时钟源 VLOCLK, 12KHz/4）

思考：可否编程在引脚 P2.0 上输出 ACLK？为什么？

答：

a. 1) 中，应使单片机接外部晶振，并使 ACLK 的输出为 8 分频；具体程序见附录[程序 5 之 1](#)）

b. 2) 中，应通过 BCSCTL3 寄存器选择时钟源 VLOCLK，并使其输出为 4 分频，具体程序见附录[程序 5 之 2](#)）

思考题

不能在引脚 P2.0 上输出 ACLK，因为，各引脚的特殊功能是由单片机结构所决定的，P2.0 引脚不具有输出辅助时钟的功能。

6. DCO 出厂校验值的频率检测

1) 利用出厂校验值，编程使 DCO 分别为 1MHz、8MHz、12MHz、16MHz，通过 P1.4 输出，用示波器测量实际值。

答：经分析知

① P1.4 输出的是 SMCLK；

② 1MHz、8MHz、12MHz、16MHz 的情况类似，此处仅以 1MHz 为例，具体程序见附录[程序 6](#)

③ 测得真实值分别为 1.002MHz、7.752MHz、12.08MHz、15.94MHz。

7. 利用输出的时钟信号做中断源，实现定时功能

将任务 3 中 P1.0 输出的 3KHz ACLK 时钟信号，作为 P1.5 的中断申请信号，用导线将

P1.5 与 P1.0 相连即可，在中断子程中设置一个计数变量，计数中断子程被执行的次数，中断子程每被执行 3000 次表示一秒时间到。利用该定时功能，将 8 个发光二极管设计成一个秒表，显示秒值，每秒改变一次 8 个发光二极管的显示。

答：根据分析可知

a. 需要在实验 5 之 2) 的基础上进行编程；

b. 须用跳线板将 P2 的引脚与 LED 灯相连；

c. 具体程序见附录[程序 7](#)

思考：如果要每隔 10 秒蜂鸣器响一声，如何在任务 5 的基础上编程实现？

答：将 P1.7 与蜂鸣器相连，增加一个延时函数以及每 10 秒操作一次 P1.7 上电位；具体程序见附录[程序 7 之思考题](#)

三、实验选做任务

1. 中断响应的理解

6)（选做）去掉 [L4_int.c](#) 程序最后的那条无限循环语句，看看有什么现象？可以去掉吗？

答：

① 现象：程序很快运行结束，无法响应中断；

② 不可去掉无限循环语句

注意：1) 查看 io430G2553.h 文件末尾处有关中断向量偏址的符号定义。

2) 为便于了解程序执行流程，可在中断子程入口处设置一断点，然后连续运行程序（F5），观察操作按键和不操作按键两种情况下程序执行的现象有何不同。

2. (选做) 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。在实验板上将 P1.0 与

一个按键的控制端相连, P1.7 与蜂鸣器的控制端相连。比较 [L4_intA.c](#) 和 [L4_intB.c](#) 二者在编程实现上有何不同。注意各自中断子程执行时间的长短。用 [L4_intB.c](#) 的方法, 改写任务 2 的编程。

答:

不同之处:

① 采用事件标志处理中断时, 按键之后机器做出的反应相对较慢, 这是因为它需要现在中断子程中设置 **flag**, 再回到主程序根据 **flag** 做相应改动, 耗费时间较长。

② 但是采用事件标志中断的一个好处就是, 由于它在中断子程中只是进行了一个标记, 你可以在标记 **flag** 在主程序中发挥作用之前再次进行中断, 将 **flag** 改为其它值。它相当于解决了一般中断过程中“中断执行时, 屏蔽其他中断”的问题。

任务 2 改写后的程序[程序 3](#)

3. (选做) 按键抖动处理

程序 [L4_Key.C](#) 见后页, 其功能是用中断方式相应与 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。

答: 根据分析, 需要在响应了第一次下降沿后, 加入一定的延时, 躲过其它电压毛刺的产生时间。具体程序见附录[程序 4](#)

通过实验观察可以发现, 改进之前, 二进制显示的数值明显大于实际按键数, 而改进之后, 两者数值大致相等。

4. DCO 出厂校验值的频率检测

2) (选做) 控制发光二极管通过延时闪亮, 编程分别使主系统时钟工作在

(1) $MCLK = \text{复位频率} / 8 \approx 100\text{KHz}$;

(2) $MCLK = DCO = 16\text{MHz}$;

两种不同频率下, 观察灯的亮灭速度有何不同, 掌握主系统时钟的变化对程序执行速度的影响

答: 根据分析

- (1) 须选择 8 分频;
- (1) 须具备延时函数与 LED 灯亮函数;
- (2) 须选择 DCO 时钟源, 并使用出厂校验值 16MHz ;
- (2) 须具备延时函数与 LED 灯亮函数;
- 具体程序见附录[选作 4 之 1](#)) 和[选作 4 之 2](#))
- 实验观察到两种不同频率下, 第一种情形下灯的亮灭速度非常慢, 而第二种情形下灯的亮灭速度非常快, 则说明: 主系统时钟频率的加快会加速执行速度。

5. (选做) 改用 4 个数码管显示秒值, 重新完成必做任务 7

答: 本题需要用到数码管的知识, 需要标志值, 具体程序见附录[选作 5 程序](#)

四、小结

总的来说, 我觉得这次的实验内容很多, 但是收获也是很大的; 并且在这个过程中重温了课本, 加深了对课本上抽象的内容的理解, 比如说实验前对于时钟, 我的理解一直很模糊, 似是而非, 经过实验, 发现其实它不过是三个寄存器, 三个时钟信号, 四个时钟源之间的问题, 虽然过程比较繁琐, 脉络却是清晰的; 而且关于中断, 我也有了进一步的理解, 特别是

最后几个选作实验和必做实验是对这两节知识的综合应用，难度有点大，但是写出来之后还是很有成就感的。

【附录】

一、基本任务程序

1. [必做第 1 题](#)

5)

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

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

void Buzz( )       // 蜂鸣响
{
    unsigned int i;
    for (i=0;i<3;i++)
    {
        P2OUT &=~BIT4;
        delay();
        P2OUT|= BIT4;
        delay();
    };
}

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);
    // 设置端口 P1.5  允许中断
    P1SEL &=~ BIT5;
    P1SEL2 &=~ BIT5;
    P1OUT |=BIT5;
    P1REN |=BIT5;
    P1DIR &=~BIT5;
    P1IES |= BIT5;
    P1IFG &=~BIT5;
    P1IE |= BIT5;
    _EINT();      // 总中断允许
```

```

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

#pragma vector=PORT1_VECTOR
__interrupt void port_ISR( )
{
    Buzz();
    P1IFG &=~BIT5;
}

```

2. [必做第 2 题](#)

```

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

void L1Blink( )    //L1  闪 3 次
{
    unsigned int i;
    for (i=0;i<3;i++)
    {
        P2OUT &=~BIT1;
        delay();
        P2OUT|= BIT1;
        delay();
    };
}

void L7Blink( )    //L7  闪
{
    P2OUT &=~BIT3;
    delay();
    P2OUT |= BIT3;
    delay();
}

void Buzz( )    // 蜂鸣响
{
    P2OUT &=~BIT4;
    delay();
    P2OUT|= BIT4;
    delay();
}

void main ( void )
{
    WDTCTL = WDTPW + WDTHOLD;    // 关闭看门狗
    // 设置引脚 P2.1 、 P2.3 、 P2.4 输出， P2.1 、 P2.3  分别连接 L1 和 L7， P2.4  连接蜂鸣器
    P2SEL &=~(BIT1+BIT3+BIT4);
    P2SEL2 &=~(BIT1+BIT3+BIT4);
}

```

```

    P2OUT|=(BIT1+BIT3+BIT4);
    P2DIR|=(BIT1+BIT3+BIT4);
// 设置端口 P1.4 、 P1.5  允许中断
    P1SEL &=~ (BIT4+BIT5);
    P1SEL2 &=~ (BIT4+BIT5);
    P1OUT |=(BIT4+BIT5);
    P1REN |=(BIT4+BIT5);
    P1DIR &=~(BIT4+BIT5);
    P1IES |= (BIT4+BIT5);
    P1IFG &=~(BIT4+BIT5);
    P1IE |= (BIT4+BIT5);
    _EINT();    // 总中断允许
for (;;)    // 主循环
    {    L7Blink();    };
}

```

```

#pragma    vector=PORT1_VECTOR
__interrupt    void    port_ISR( )
{
if ((P1IFG&BIT4)!=0)
    {Buzz();
    P1IFG &=~BIT4; };
if ((P1IFG&BIT5)!=0)
{
L1Blink();
P1IFG&=~BIT5;};
}

```

2. 思考题

```

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

void L1Blink( )    //L1  闪 3 次
{ unsigned int i;
    for (i=0;i<3;i++)
    {    P2OUT &=~BIT1;
    delay();
    P2OUT|= BIT1;
    delay();
    };
}

```

```

void L7Blink( )      //L7  闪
{
    P2OUT &=~BIT3;
    delay();
    P2OUT |= BIT3;
    delay();
}

void Buzz( )        // 蜂鸣响
{ P2OUT &=~BIT4;
  delay();
  P2OUT|= BIT4;
  delay();
}

void main ( void )
{ WDTCTL = WDTPW + WDTHOLD;      // 关闭看门狗
// 设置引脚 P2.1 、 P2.3 、 P2.4 输出， P2.1 、 P2.3  分别连接 L1 和 L7， P2.4  连接蜂鸣器
    P2SEL &=~(BIT1+BIT3+BIT4);
    P2SEL2 &=~(BIT1+BIT3+BIT4);
    P2OUT|=(BIT1+BIT3+BIT4);
    P2DIR|=(BIT1+BIT3+BIT4);
// 设置端口 P2.2 、 P2.5  允许中断
    P2SEL &=~ (BIT2+BIT5);
    P2SEL2 &=~ (BIT2+BIT5);
    P2OUT |=(BIT2+BIT5);
    P2REN |=(BIT2+BIT5);
    P2DIR &=~(BIT2+BIT5);
    P2IES |= (BIT2+BIT5);
    P2IFG &=~(BIT2+BIT5);
    P2IE |= (BIT2+BIT5);
    _EINT();      // 总中断允许
    for (;;)      // 主循环
    {   L7Blink();   };
}

#pragma    vector=PORT2_VECTOR
__interrupt    void    port_ISR( )
{
    if ((P2IFG&BIT2)!=0)
    {Buzz();
      P2IFG &=~BIT2; };
    if ((P2IFG&BIT5)!=0)
    {
        L1Blink();
        P2IFG&=~BIT5;};
}

```

```
}
```

4. 测试上电复位系统的 ACLK、和 SMCLK 时钟频率

```
int main( void )
{
    // 关闭看门狗
    WDTCTL = WDTPW + WDTHOLD;
    // 设置 P2.6 、 P2.7 连接外部晶振
    P2SEL|=(BIT6+BIT7);
    P2SEL2&=~(BIT6+BIT7);
    P2DIR&=~BIT6;
    P2DIR|=BIT7;
    // 设置 P1.0 、 P1.4 输出 ACLK、 SMCLK
    P1SEL|=BIT0;
    P1SEL2&=~BIT0;
    P1DIR|=BIT0;
    P1SEL|=BIT4;
    P1SEL2&=~BIT4;
    P1DIR|=BIT4;
    while(1);
}
```

5. 掌握基本时钟模块的编程控制

1) ACLK=4096Hz; (时钟源外部晶振, 32768HZ/8)

```
int main( void )
{
    // 关闭看门狗
    WDTCTL = WDTPW + WDTHOLD;
    // 设置 P2.6 、 P2.7 连接外部晶振
    P2SEL|=(BIT6+BIT7);
    P2SEL2&=~(BIT6+BIT7);
    P2DIR&=~BIT6;
    P2DIR|=BIT7;
    // 设置 P1.0 输出 ACLK
    P1SEL|=BIT0;
    P1SEL2&=~BIT0;
    P1DIR|=BIT0;
    BCCTL3&=~(LFXT1S0+LFXT1S1);// 选择外部晶振
    BCCTL1|=(DIVA0+DIVA1);// 选择 8 分频
    while(1);
}
```

2) ACLK=3KHz ; (时钟源 VLOCLK, 12KHz/4)

```
int main( void )
```

```

// 关闭看门狗
WDTCTL = WDTPW + WDTHOLD;
// 设置 P1.0 输出 ACLK
P1SEL|=BIT0;
P1SEL2&=~BIT0;
P1DIR|=BIT0;
BCSCTL3&=~LFXT1S0;// 选择时钟源 VLOCLK
BCSCTL3|=LFXT1S1;
BCSCTL1&=~DIVA0;// 选择 4 分频
BCSCTL1|=DIVA1;
while(1);
}

```

第 6 题必做之 DCO 出厂校验值的频率检测

```

int main( void )
{
// 关闭看门狗
WDTCTL = WDTPW + WDTHOLD;
P1SEL|=BIT4;// 设置 P1.4 输出 SMCLK
P1SEL2&=~BIT4;
P1DIR|=BIT4;
BCSCTL2&=~SELS;// 选择 DCO 时钟源
BCSCTL1=CALBC1_1MHZ; // 其他情况改变 1MHz 就可以了
DCOCTL=CALDCO_1MHZ;
while(1);
}

```

必做第 7 题之利用输出的时钟信号做中断源，实现定时功能

1)

```

unsigned char time=0;
unsigned int i=0;
int main( void )
{
// 关闭看门狗
WDTCTL = WDTPW + WDTHOLD;

// 设置端口 P1.0 输出 3KHzACLK 时钟信号
P1SEL|=BIT0;
P1SEL2&=~BIT0;
P1DIR|=BIT0;
BCSCTL3&=~LFXT1S0;
BCSCTL3|=LFXT1S1;

```

```

BCSCTL1&=~DIVA0;
BCSCTL1|=DIVA1;

// 设置 P2.0~P2.7 为输出状态
P2SEL&=0;
P2SEL2&=0;
P2DIR|=0Xff;
P2OUT|=0Xff;

// 设置端口 P1.5 允许中断
P1SEL&=~BIT5;
P1SEL2&=~BIT5;
P1REN|=BIT5;
P1OUT&=~BIT5;
P1DIR&=~BIT5;
P1IES&=~BIT5;
P1IFG&=~BIT5;
P1IE|=BIT5;
_EINT();

while(1){
    if (i>=3000) {
        time+=1;
        i=0;}
    P2OUT=~time;          //LED 灯显示输出秒表的值
    }
}

#pragma vector=PORT1_VECTOR
__interrupt void port_ISR(){
    i++;
    P1IFG&=~(BIT5);      // 清中断标志
}

```

2) 必做 7 之思考题

```

unsigned char time=0;
unsigned int i=0;
void delay();
int main( void )
{
    // 关闭看门狗
    WDTCTL = WDTPW + WDTTHOLD;

```

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/086024122012010102>