深 圳 大 学 实 验 报 告

课程名称：微型计算机技术

实验项目名称：外部中断实验

学院：医学院

专业：生物医学工程

指导教师：徐海华、刘昕宇

报告人：陈焕鑫 学号：2016222042 班级：生工 2 班

实验时间：2018-10-30

实验报告提交时间：2018-11-6

教务处制

一、实验目的

1．了解中断、中断入口地址、中断服务程序

2．了解外部中断的两种方式（边沿触发、电平触发），并用按键实现

3．使用汇编语言和 C 语言进行编程，了解汇编语言和 C 语言对中断处理的不同之处

二、实验仪器

微机原理实验板

三、实验内容

1、在主程序中编写循环跑马灯的程序，使用 KEY1 外部中断功能，当 KEY1 按下时，程序进入外部中断 0，在中断程序内让蜂鸣器响三声（哔-哔-哔），每一次响 500ms，停 500ms。

2、分别使用边沿触发方式和电平触发方式实现 1）中的功能

3、分别用汇编和 C 语言写出以上程序，故一共有 4 个程序

四、实验原理

外部中断是单片机实时地处理外部事件的一种内部机制。当某种外部事件发生时，单片机的中断系统将迫使 CPU 暂停正在执行的程序，转而去进行中断事件的处理；中断处理完毕后．又返回被中断的程序处，继续执行下去。

在 C 语言中，中断服务函数的模板如下所示

```
void main()
{
  ISR_Init();
  while(1);
}
void INIT0_ISR()interrupt 0
{
  中断服务程序
}
```

以上的程序，中断初始化完成后，程序就会一直在 main()的 while(1)中循环，而当来了一个外部中断 0 的中断信号时，此时程序就会暂停 while(1)的循环，自动跳入 INT1_ISR()程序中执行，执行完中断服务程序后，程序指针又会重新返回刚才停止的断点处，继续执行。

在 C 语言中，中断的入口程序，全部都要加 interrupt 限定，而后面的数字，则表示外部中断 0.

表6-1　中断查询次序

中断源	中断向量地址	相同优先级内的查询次序	中断优先级设置 (IPH,IP)	优先级0（最低）	优先级1	优先级2	优先级3（最高）	中断请求标志位	中断允许控制位
$\overline{\text{INT0}}$ (外部中断 0)	0003H	0 (highest)	PX0H, PX0	0, 0	0, 1	1, 0	1, 1	IE0	EX0/EA
Timer 0	000BH	1	PT0H, PT0	0, 0	0, 1	1, 0	1, 1	TF0	ET0/EA
$\overline{\text{INT1}}$ (外部中断1)	0013H	2	PX1H, PX1	0, 0	0, 1	1, 0	1, 1	IE1	EX1/EA
Timer1	001BH	3	PT1H, PT1	0, 0	0, 1	1, 0	1, 1	TF1	ET1/EA
UART1	0023H	4	PSH, PS	0, 0	0, 1	1, 0	1, 1	RI+TI	
ADC	002BH	5	PADCH,PADC	0, 0	0, 1	1, 0	1, 1	ADC_FLAG	EADC/EA
LVD	0033H	6	PLVDH,PLVD	0, 0	0, 1	1, 0	1, 1	LVDF	ELVD/EA
PCA	003BH	7	PPCAH,PPCA	0, 0	0, 1	1, 0	1, 1	CF+CCF0 + CCF1	(ECF+ECCF0 +ECCF1)/EA
S2(UART2)	0043H	8	PS2H, PS2	0, 0	0, 1	1, 0	1, 1	S2TI+S2RI	ES2/EA
SPI	004BH	9 (lowest)	PSPIH, PSPI	0, 0	0, 1	1, 0	1, 1	SPIF	ESPI/EA

在汇编语言中，中断服务函数的进入方法：

在汇编语言中，中断程序有固定的入口地址，入口地址可以从；表 6-1 中得知。

下面给出一个例程

```
ORG 0000H
LJMP MAIN
ORG 0003H
LJMP INT0_ISR

ORG 1000H
MAIN:
``````````````````````````;中断初始化程序
JMP $

ORG 2000H
INT0_ISR:
``````````````````````````;中断服务程序
RETI
```

我们可以看看上面的程序，开机之后，程序跳到 MAIN 标记处，运行中断初始化程序之后，在 JMP $这里不停的原地跳转循环。

而当来了一个外部中断 0 的中断信号时，此时程序就会暂停 JMP $的循环，自动跳入

0003H 的地址中，从此处开始执行，于是程序就会执行一个长跳转，跳转到 INT0_ISR 程序中执行，执行完中断服务程序后，也就是执行到 RETI 命令（中断返回）的时候，程序指针就会停止中断服务程序的执行，重新返回刚才停止的断点处（JMP $），继续执行循环，等待下一次中断。

## 五、实验方法及步骤

首先，打开 Keil 软件，新建工程名为 Lab6Prj，在工程中添加空的 main.asm 汇编文件。

然后，对 main.asm 文件进行修改。结合原理图，我们对按键的引脚进行定义。

```
LED1 BIT P2.4   ;设置 LED1 的引脚为 P2.4
LED2 BIT P2.5   ;设置 LED2 的引脚为 P2.5
LED3 BIT P2.6   ;设置 LED3 的引脚为 P2.6
LED4 BIT P2.7   ;设置 LED4 的引脚为 P2.7
BEEP BIT P1.0   ;设置蜂鸣器的引脚为 P1.0
```

然后就是对一些重要的地址进行命名操作，比如 DelayTimes_100MS。有了这些定义之后我们就可以方便的对数据进行操作了。

结合外部中断的工作原理。主程序开始的地方，需要增加中断服务函数 0 的入口地址 0003H，在入口执行长跳转指令跳转到中断服务函数中。从 MAIN 处开始之后，首先将 SP 指针指向 70H 区，这个区是专门用来存放堆栈的。接下来是对外部中断 0 的各种初始化，SETB IT0 是将中断发生条件设置为边沿触发，CLR PX0 设置中断优先级，SETB EX0 设置外部中断使能，SETB EA 是打开外部中断的总开关。设置蜂鸣器端口为推挽输出，关闭蜂鸣器之后，运行到 START1 处，调用到 WATERLED 函数，在函数中循环点亮 LED 灯。如果此时发生了外部中断，则程序会中断当前程序的运行，跳转到中断函数中处理中断任务，处理结束后再返回原来的位置继续运行。

C 程序的方法与汇编类似，需要注意的时在中断处理函数后面需要加上标识符 interrupt 0 标识符。

边沿触发的 C 语言代码如下所示：

```
#include "STC12C5A60S2.h"

sbit LED1 =  P2^4; //设置 LED1 的引脚为 P2.4
sbit LED2 =  P2^5; //设置 LED2 的引脚为 P2.5
sbit LED3 =  P2^6; //设置 LED3 的引脚为 P2.6
sbit LED4 =  P2^7; //设置 LED4 的引脚为 P2.7
sbit Beep =  P1^0; //设置 Beep 的引脚为 P1.0
```

```c
void INT0_ISR()interrupt 0 //外部中断服务函数 0
{
  unsigned long int i; //定义延时变量 i

  Beep = 1;  //蜂鸣器响
  for(i = 0; i < 50000; i++);//延时
  Beep = 0;  //蜂鸣器不响
  for(i = 0; i < 50000; i++);//延时
  Beep = 1;   //蜂鸣器响
  for(i = 0; i < 50000; i++);//延时
  Beep = 0;   //蜂鸣器不响
  for(i = 0; i < 50000; i++);//延时
  Beep = 1;   //蜂鸣器响
  for(i = 0; i < 50000; i++);//延时
  Beep = 0;   //蜂鸣器不响
}

void ISR_Init()   //中断初始化函数
{
  IT0 = 1;  //设置边沿触发
  PX0 = 0;  //设置优先级
  EX0 = 1;  //设置外部中断使能
  EA =  1;  //中断总开关
}

void main() //主函数
{
  unsigned long int i;  //延时变量 i
  unsigned int j;       //状态变量 j
  P1M0 = 0x01;          //推挽输出
  Beep = 0;             //蜂鸣器不响

  ISR_Init();           //调用中断初始化函数

  while(1)              //死循环执行跑马灯
  {
    for(i = 0; i < 50000; i++);//延时
    switch(j)
    {
      case 0:
        LED1 = ~LED1;     //LED1 取反
        break;
      case 1:
        LED2 = ~LED2;     //LED2 取反
```

```
        break;
      case 2:
        LED3 = ~LED3;     //LED3 取反
        break;
      case 3:
        LED4 = ~LED4;     //LED4 取反
        break;
    }
    if(j >= 3)
    {
      j = 0;
    }
    else
    j++;
  }
}
```

边沿触发的汇编语言代码如下所示：

```
LED1 BIT P2.4   ;设置 LED1 的引脚为 P2.4
LED2 BIT P2.5   ;设置 LED2 的引脚为 P2.5
LED3 BIT P2.6   ;设置 LED3 的引脚为 P2.6
LED4 BIT P2.7   ;设置 LED4 的引脚为 P2.7

BEEP BIT P1.0   ;设置蜂鸣器的引脚为 P1.0

DelayTimes_100MS EQU 6FH  ;设置延时计数器存放的寄存器

;************************************
;--主程序开始--
ORG 0000H       ;程序入口 0000H
LJMP MAIN       ;长跳转到 MAIN
ORG 0003H       ;中断服务函数 0 在内存中的位置为 0003H
LJMP INT0_ISR   ;长跳转到中断服务函数 0

ORG 1000H       ;MAIN 函数写在 1000H 的位置
MAIN:
  MOV SP,#70H   ;修改 SP 指针的指向
  SETB IT0      ;设置为边沿触发
  CLR PX0       ;设置优先级
  SETB EX0      ;外部中断使能
  SETB EA       ;中断总开关
  MOV P1M1,#00H     ;配置为默认
  MOV P1M0,#01H     ;配置蜂鸣器端口为推挽输出
  CLR BEEP      ;关闭蜂鸣器
```

```
  LJMP START1   ;长跳转到 START1 处

START1:
  LCALL WATERLED  ;调用流水灯函数
  LJMP  START1    ;长跳转到 START1

;*************************************
;延时 100MS 函数

Delay100MS:
  MOV R7, DelayTimes_100MS
  Delay100MS_1:  MOV R6,#10
  Delay100MS_2:  MOV R5,#150
  Delay100MS_3:  MOV R4,#200
  Delay100MS_4:  DJNZ R4,Delay100MS_4
         DJNZ R5,Delay100MS_3
         DJNZ R6,Delay100MS_2
         DJNZ R7,Delay100MS_1
RET
;*************************************

;*************************************
;流水灯函数

WATERLED:
  CLR  LED1
  SETB LED2
  SETB LED3
  SETB LED4
  MOV DelayTimes_100MS,#10
  LCALL Delay100MS

  SETB  LED1
  CLR   LED2
  SETB  LED3
  SETB  LED4
  MOV DelayTimes_100MS,#10
  LCALL Delay100MS

  SETB  LED1
  SETB  LED2
  CLR   LED3
  SETB  LED4
  MOV DelayTimes_100MS,#10
```

```
  LCALL Delay100MS

  SETB  LED1
  SETB  LED2
  SETB  LED3
  CLR   LED4
  MOV DelayTimes_100MS,#10
  LCALL Delay100MS
RET
;*************************************

;*************************************
;中断服务函数 0

ORG 2000H
INT0_ISR:
  PUSH PSW    ;将 PSW 状态字压栈
  PUSH ACC    ;将 ACC 压栈
  CLR  RS0    ;修改工作寄存器组
  SETB RS1
;**********************
;中断服务函数主要内容
  MOV R0,#6
TWEET:
  CPL BEEP              ;蜂鸣器取反
  MOV DelayTimes_100MS,#5 ;延时 500ms
  LCALL Delay100MS       ;调用延时函数
  DJNZ R0,TWEET
;**********************
  POP ACC   ;将 ACC 出栈
  POP PSW   ;将 PSW 状态字出栈
RETI     ;中断服务函数结束
;*************************************


END
```

检查代码无误之后，分别编译、链接、生成 Hex 文件，将 Hex 文件通过串口烧进实验平台中，观察实验现象。

## 六、实验现象

将程序烧录到实验板之后，观察实验板的现象，可以发现，开机之后，板上的 LED 灯呈循环跑马灯流动，当轻按 Key1 按键时，跑马灯的流动被固定住，蜂鸣器鸣响三声，间隔为 0.5s 左右，响完三声之后，实验板又回到原来的状态，跑马灯又开始流动，直到再次按下 Key1 按键。

## 七、实验结论

通过这次实验的学习，我了解了中断、中断入口地址和中断服务函数如何使用，其实，中断就是指计算机在执行其他程序的过程中，当出现了某些异常事件或某种请求时，CPU 暂时中止正在执行的程序，而转去执行对异常事件或某种请求的服务程序。当服务完毕后，CPU 再回到被暂时中止的程序继续执行。了解了外部中断的两种触发的方式（边沿触发、电平触发），并用按键实现。学会了使用汇编语言和 C 语言进行编程，了解汇编语言和 C 语言对中断处理的不同之处。

| |
|---|
| 指导教师批阅意见:<br><br>成绩评定:<br><br>指导教师签字:<br><br>年　　月　　日 |
| 备注: |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。
2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。