项目三

电子密码锁设计

密码锁是锁的一种，由一系列数字或符号构成密码，开启时输入密码，当密码正确时，锁便可以打开。电子密码锁以键盘、指纹、瞳孔扫描等方式作为密码输入，以单片机为核心，具有密码设置、存储、识别、显示、报警等功能，在楼宇控制、金融、保险等领域获得了广泛的应用。

项目任务描述

本项目采用 STC89C51 单片机为核心，以 4×3 非编码键盘为密码输入媒介，以 1602 点阵字符型 LCD 显示器为显示介质设计一个电子密码锁。本密码锁开机后，LCD 显示主菜单；当用户输入密码后，以字符“*”代替。如果密码正确，则继电器开启，绿灯亮；否则，继电器关闭，红灯亮。如果密码输入不正确，发出报警，直到密码输入正确后，解除报警。

本项目对知识和能力的要求如表 3-1 所示。

表 3-1　项目对知识和能力的要求

项目名称	学习任务单元划分	任务支撑知识点	项目支撑工作能力
电子密码锁设计	键盘检测	①非编码键盘的工作原理；②独立按键的检测；③矩阵非编码键盘的检测	资料筛查与利用能力；新知识、新技能获取能力；组织、决策能力；独立思考能力；交流、合作与协商能力；语言表达与沟通能力；规范办事能力；批评与自我批评能力
	通用型 1602 液晶	①通用型 1602 液晶的工作原理；②通用型 1602 液晶的控制字；③通用型 1602 液晶显示字符和数据	
	电子密码锁设计实践	①继电器认知和控制；②蜂鸣器认知；③电子密码锁仿真电路设计；④电子密码锁程序设计；⑤电子密码锁 PCB 电路图设计和电路制作	

3.1　键盘检测

3.1.1　键盘工作原理

键盘是日常生活中常用的输入设备，在电脑、手机、PDA、ATM 柜员机等设备中应用广泛。键盘按照结构原理来划分，分为触点式开关键盘和非触点式开关键盘；按编码方式，分为编码键盘和非编码键盘。触点式键盘由机械式开关、导电橡胶式按键组成，非触点式键盘由电气式按键、磁感应按键等构成。编码键盘按键的闭合识别由专用硬件编码器实现，如计算机键盘；非编码键盘按键的闭合识别靠软件编程来实现。非编码式键盘又分独立式键盘和矩阵式（又称行列式）键盘。编码键盘所需软件简单，但是电路复杂，成本高。相对而言，非编码键盘具有电路简单、成本低的特点，因此在单片机应用系统中主要采用非编码键盘。

键盘是由一系列按键组成的，在单片机应用系统中往往采用机械触点式按键。当按键按下时，线路导通，按键弹起，线路断开。由于机械触点的弹性作用，按键在按下的过程中存在触点在闭合和断开瞬间接触不稳定的情况，造成了电压信号不稳定的现象（如图 3-1 所示），因此，在实际应用中需要消除按键的抖动。按键的抖动时间一般为 5～10ms，而稳定闭合时间一般超过 20ms，如果不对按键进行去抖动处理，会引起单片机对一次按键操作进行多次处理。在单片机应用中，一般采用的措施是：当第一次检测到按键按下后，延时 10～20ms，再次检测按键是否按下；如果此时按键还是处于按下状态，则确认有按键按下，否则取消此次检测结果。键盘检测程序流程图如图 3-2 所示。

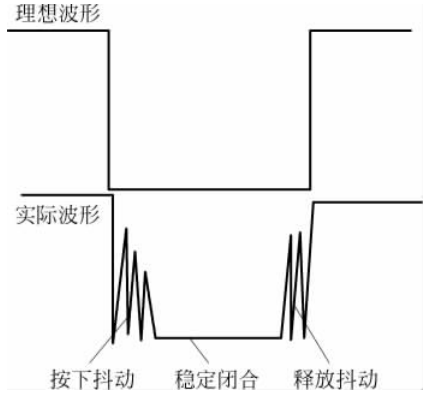

图 3-1　按键触点抖动引起的电平变化

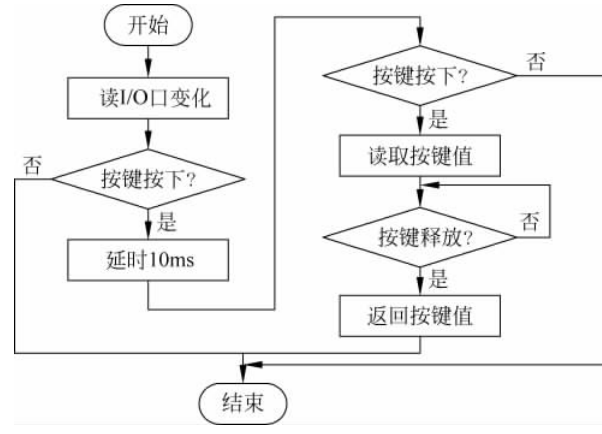

图 3-2　按键检测程序流程图

【例 3.1】　按键点控制 LED 亮灭灯。

仿真电路图如图 3-3 所示，编程实现按键 K_1 控制 LED 灯的亮灭。即开机 D_1 熄灭，在 D_1 熄灭状态按下 K_1，D_1 亮；在 D_1 亮状态，按下 K_1，D_1 熄灭。当按键按下时，P1.4 接

地，为低电平；当按键弹开时，P1.4 通过 R_3 接＋5V 电源，为高电平。D_1 LED 灯通过 R_2 接 P2.0，R_2 起限流作用。

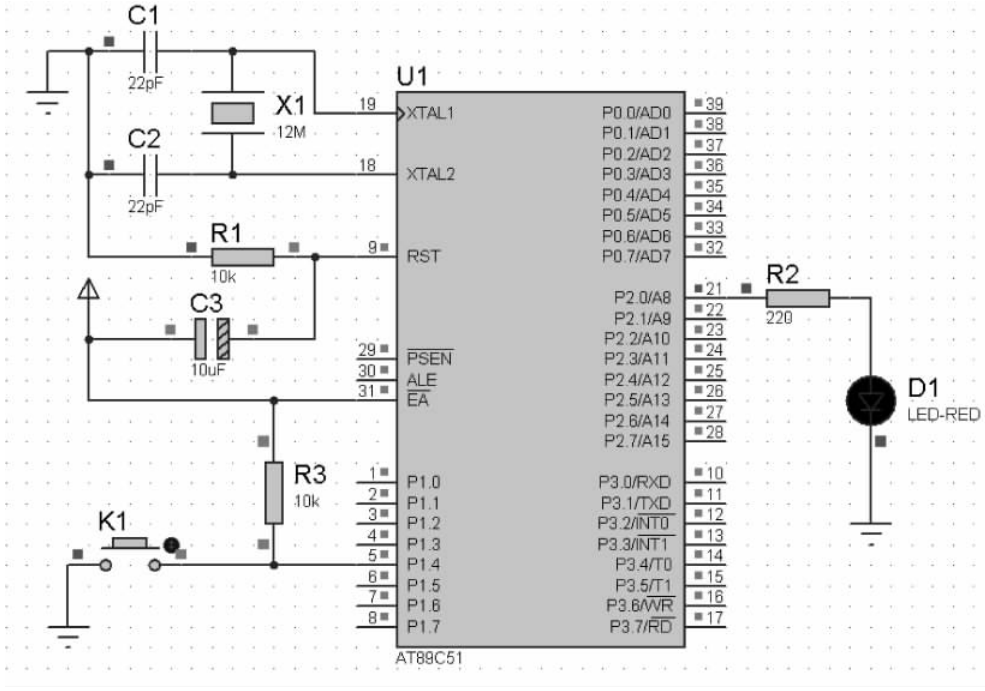

图 3-3　例 3.1 仿真电路图

参考程序如下：

```
#include <reg51.h>
#define uchar unsigned char                    //宏定义
sbit Key = P1^4;                               //声明按键接口
sbit D1 = P2^0;                                //声明 D1 LED 灯接口
void Delay(int x)                              //延时函数，延时 xms
{
    while(x--)
    {
        uchar i;
        for(i = 0; i < 110; i++);              //内部延时 1ms
    }
}
uchar KeyScand()
{
    if(0 == Key)                               //如果按键按下
    {
        Delay(10);                             //延时 10ms，去抖动
        if(0 == Key)                           //确实有按键按下
        {
            while(0 == Key);                   //等待按键释放
```

```
            return 1;                                    //返回按键值 1
        }
        else                                  //否则返回 0,表示按键按下时间不足,取消本次按键操作
        {
            return 0;
        }
    }
    else
    {
        return 0;
    }
}
void main()
{
    D1 = 0;
    while(1)
    {
        if(KeyScand())
        {
            D1 = ~D1;
        }
    }
}
```

3.1.2　线性键盘检测

当按键数目不多的时候，可以将按键排成一行或一列（因此称为线性键盘），一端接单片机 I/O 口的引脚，同时接上拉电阻，另一端串接在一起接公共端（接地），如图 3-4 所示。线性键盘电路配置灵活，结构简单，但每个按键都必须占用一个单片机 I/O 口，占用单片机硬件资源比较多，因此适合于按键数目不多，单片机硬件资源不紧张的应用场合。

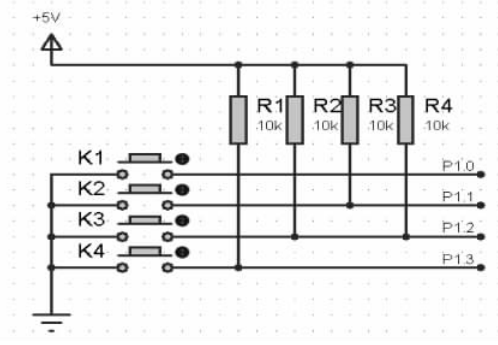

图 3-4　线性键盘

【例 3.2】 检测线性键盘值并显示。

仿真电路如图 3-5 所示，P0 口接七段共阴极数码管显示器（7SEG-COM-CATHODE），

P1.0～P1.3 分别接按键 K_1～K_4，同时接 10kΩ 上拉电阻。当按键没有按下时，P1.0～P1.3 电平为高电平；当按键按下时，相应端口电平变为低电平，通过按键接地。因此，通过读取 P1 口低 4 位电平变化，可获知按键是否按下。如果有按键按下，则 P1 口低 4 位必然有 1 位为低电平，延时 10ms 去抖动，再次读取 P1 口低 4 位值；如果不为 0x0f，表示确实有按键按下，然后通过一个 while 循环等待按键释放。如图 3-5 所示，按键 K_1～K_4 单独按下时对应的按键码值分别为 0x0e、0x0d、0x0b、0x07；如果有两个以上按键同时按下，认为按键无效。系统初始显示按键值为 0；当有按键按下时，显示相应的按键值；两个以上按键同时按下，则显示 0。

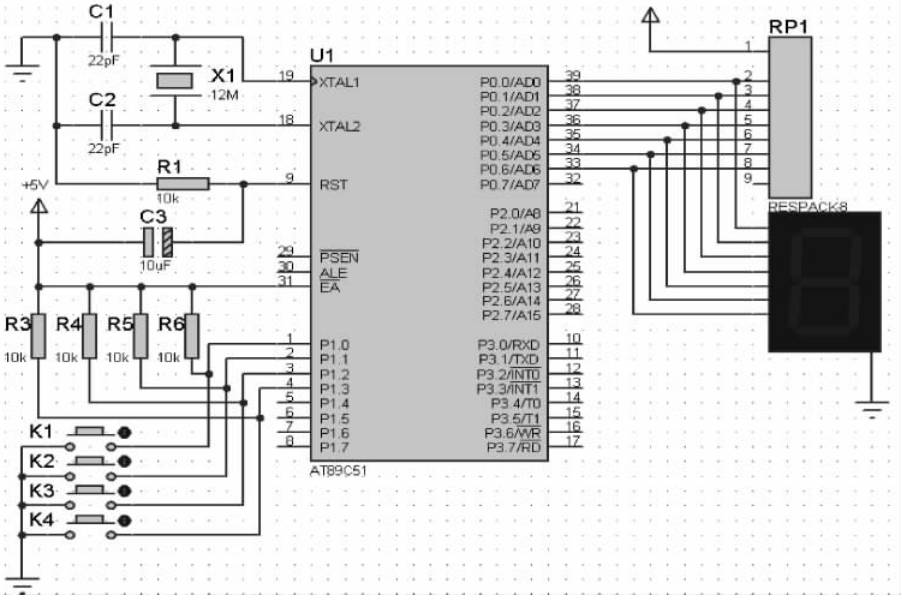

图 3-5　线性键盘检测

参考程序如下：

```
/ ************************************************************************
文件名称：例 3.2.c
程序功能：读取线性键盘信息并显示
程序作者：果子冰
创建时间：2011-06-12
 ************************************************************************ /
#include <reg51.h>
#define uchar unsigned char                    //宏定义
uchar SegCode[5]=                              //显示字符码表,0～4
{0x3f,0x06,0x5b,0x4f,0x66};
/ ************************************************************************
函数名称：void delay(uint x)
函数功能：实现 xms 延时
入口参数：x,定时时间形参
返回值：无
```

```
函数作者：果子冰
创建时间：2011-06-12
*************************************************************************/
void delay(int x)                                    //延时函数，延时 xms
{
    while(x--)
    {
        uchar i;
        for(i = 0; i < 110; i++);                    //内部延时 1ms
    }
}
/*************************************************************************
函数名称：uchar KeyScand()
函数功能：按键检测，检测哪个按键按下
入口参数：无
返回值：按下的按键值
函数作者：果子冰
创建时间：2011-06-12
*************************************************************************/
uchar KeyScand()
{
    uchar temp = 0;
    uchar key = 0;
    temp = P1 & 0x0f;                                //取低 4 位值
    if(0x0f != temp)                                 //判断是否有按键按下
    {
        delay(10);                                   //延时 10ms，去抖动
        temp = P1 & 0x0f;
        if(0x0f != temp)
        {
            key = temp;
            switch(key)                              //取得按键值
            {
                case 0x0e : key = 1;break;
                case 0x0d : key = 2;break;
                case 0x0b : key = 3;break;
                case 0x07 : key = 4;break;
                default : key = 0;break;
            }
            while(0x0f != temp) temp = P1 & 0x0f; //等待按键释放

        }
    }
    return key;
}
/*************************************************************************
函数名称：void main()
函数功能：获得线性键盘信息并显示
入口参数：无
```

```
返回值：无
函数作者：果子冰
创建时间：2011-06-12
**************************************************************************/
void main()
{
    uchar key = 0;
    P0 = SegCode[0];
    while(1)
    {

        key = KeyScand();                          //扫描按键
        if(key != 0)
        {
            P0 = SegCode[key];
        }

    }
}
```

3.1.3 矩阵键盘检测

由于线性键盘的每一个按键都是单独与单片机的 I/O 相连，每一个按键都需要单片机的 I/O 口，占用单片机的硬件资源较多。特别是当按键数量很多的时候，如果每个按键都占用单片机的一个 I/O 口，势必造成单片机硬件资源紧张。因此，在按键数量较多的情况下，将按键开关设置在行线和列线的交叉点上，行线和列线分别连接在按键的两端，构成矩阵键盘，以节约单片机的 I/O 口。如图 3-6 所示便是一个 4×4 的矩阵非编码键盘。

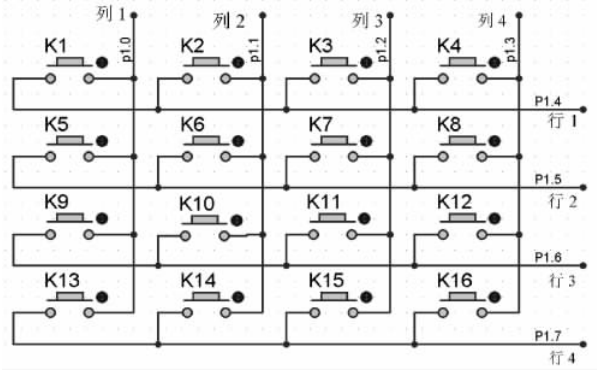

图 3-6 矩阵非编码键盘

矩阵非编码键盘和线性非编码键盘按键的工作原理是一样的，即当有按键按下的时候，按键所连接的 I/O 口电平将发生变化。通过查询 I/O 口电平的变化，便可获知是哪个键按下。对于矩阵非编码键盘来说，通过 I/O 口电平的变化，便可知道按键所在的行值和列值，而每一个按键都对应一对行值和列值。例如，K_1 键所在位置为第 1 行第 1 列，则对应的键值编码可设为 0x11。因此，矩阵非编码键盘检测的实质就是确定按键所在的

行值和列值。矩阵非编码键盘的检测方法主要有反转法和扫描法。

1. 反转法

所谓反转法，是指依次通过向行线和列线输入相反的电平，然后通过单片机 I/O 电平的变化，确定按键所在的行和列。反转法矩阵非编码键盘检测的程序流程图如图 3-7 所示。反转法矩阵非编码键盘检测步骤如下：

（1）向行线输出低电平，列线输出高电平，然后读取列线所在的 I/O 口电平。如果有按键按下，则按键所在的列 I/O 口将变为低电平。如图 3-6 所示，首先置 P1.4～P1.7 输出低电平，P1.0～P1.3 输出高电平，然后读取 P1 口的电平。如果此时有按键按下，则 P1 口的值必然不为 0x0f。例如，如果 K_1 按下，则 P1.0 电平将变为低电平，读 P1 口的值为 0x0e，将该值取反得 0xf1；0xf1 与 0x0f 做按位与运算，保留低 4 位值，得 0x01。此值就是 K_1 按键所在的列值。

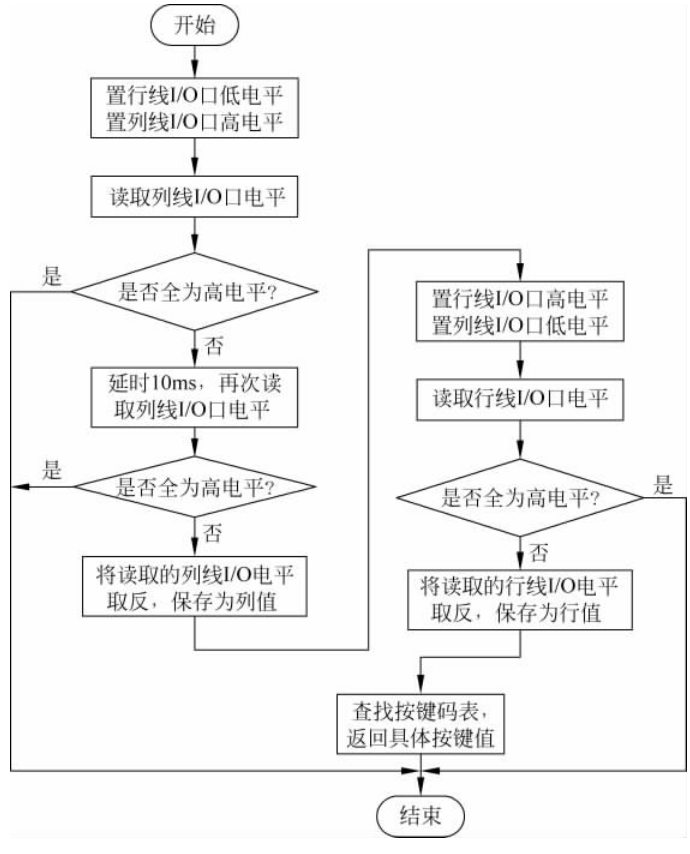

图 3-7　矩阵非编码键盘反转法识别程序流程图

（2）向行线输出高电平，列线输出低电平，然后读取行线所在的 I/O 口电平。如果有按键按下，则按键所在的行线 I/O 口将变为低电平。如图 3-6 所示，置 P1.0～P1.3 输出低电平，P1.4～P1.7 输出高电平，然后读取 P1 口的电平。如果此时有按键按下，则 P1 口的值必然不为 0xf0。例如，如果 K_1 按下，则 P1.4 电平将变为低电平，读 P1 口的值为

0xe0，将该值取反得 0x1f；0x1f 与 0xf0 做按位与运算，保留高 4 位值，得 0x10。此值就是 K_1 按键所在的行值。

（3）将按键的行与列相加，获得按键的码值，高 4 位保存行值，低 4 位保存列值，如 K_1 键对应的按键码值为 0x11。将按键的码值与非编码键盘按键码值表进行对照，就可以获得按键的键值。图 3-6 所示 4×4 非编码键盘对应的按键码值表如表 3-2 所示。

表 3-2　4×4 非编码键盘码值表

按键名称	码值	按键名称	码值
K_1	0x11	K_9	0x41
K_2	0x12	K_{10}	0x42
K_3	0x14	K_{11}	0x44
K_4	0x18	K_{12}	0x48
K_5	0x21	K_{13}	0x81
K_6	0x22	K_{14}	0x82
K_7	0x24	K_{15}	0x84
K_8	0x28	K_{16}	0x88

【例 3.3】 利用反转法检测 4×4 非编码键盘。

仿真电路如图 3-8 所示，4×4 非编码键盘列线接 P1 口的低 4 位，即 P1.0～P1.3；行线接 P1 口的高 4 位，即 P1.4～P1.7。采用 2 位共阳极七段数码管（7SEG-MPX2-CA）显示按键值，系统初始显示值为 00。

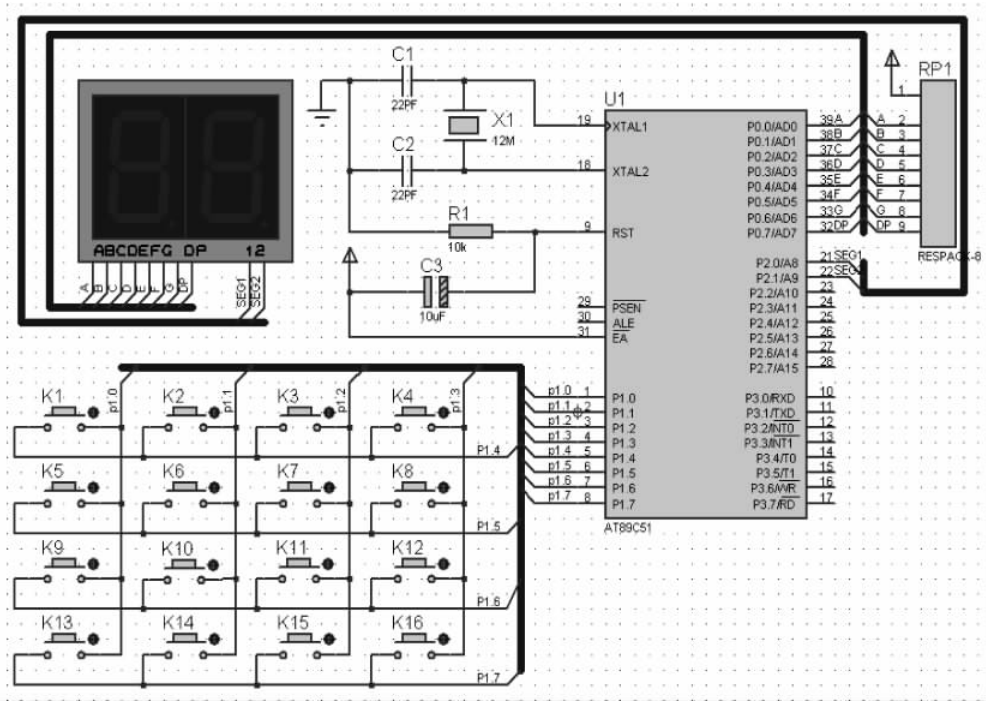

图 3-8　例 3.3 仿真电路

参考程序如下：

```
/ ************************************************************************
文件名称: 例 3.3.c
程序功能: 反转法检测非编码键盘并显示
程序作者: 果子冰
创建时间: 2011-06-12
************************************************************************ /
#include <reg51.h>
#define uchar unsigned char                          //宏定义
uchar SegCode[10]=                                   //显示字符码表,0～9
{0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90};
/ ************************************************************************
函数名称: void delay(uint x)
函数功能: 实现 xms 延时
入口参数: x,定时时间形参
返回值: 无
函数作者: 果子冰
创建时间: 2011-06-12
************************************************************************ /
void delay(int x)                                    //延时函数,延时 xms
{
    while(x--)
    {
        uchar i;
        for(i = 0; i < 110; i++);                    //内部延时 1ms
    }
}
/ ************************************************************************
函数名称: uchar KeyScand()
函数功能: 按键检测,检测哪个按键按下
入口参数: 无
返回值: 按下的按键值
函数作者: 果子冰
创建时间: 2011-06-12
************************************************************************ /
uchar KeyScand()
{
    uchar temp = 0;
    uchar key = 0;
    P1 = 0x0f;                                       //列线置高电平,行线置低电平
    temp = P1 & 0x0f;                                //读列线 I/O 电平
    if(0x0f != temp)                                 //判断是否有按键按下
    {
        delay(10);                                   //延时 10ms,去抖动
        temp = P1 & 0x0f;                            //再次读列线 I/O 电平
        if(0x0f != temp)
        {
          temp = ~temp;                              //取反
```