

|      |               |       |  |
|------|---------------|-------|--|
| 学生学号 | 0121618380615 | 实验课成绩 |  |
|------|---------------|-------|--|

# 武汉理工大学 学 生 实 验 报 告 书

|         |            |
|---------|------------|
| 实验课程名称  | 单片机原理及接口技术 |
| 开 课 学 院 | 物流工程学院     |
| 指导教师姓名  | 袁 兵        |
| 学 生 姓 名 | 付清晨        |
| 学生专业班级  | 机设 1606    |

2018 -- 2019 学年 第 1 学期

## 实验课程名称：单片机原理及接口技术

|        |            |      |         |      |                 |
|--------|------------|------|---------|------|-----------------|
| 实验项目名称 | 电话键盘及拨号的模拟 |      |         | 实验成绩 |                 |
| 实 验 者  | 付清晨        | 专业班级 | 机设 1606 | 组 别  |                 |
| 同 组 者  | 无          |      |         | 实验日期 | 2018 年 11 月 2 日 |

### 1. 实验要求

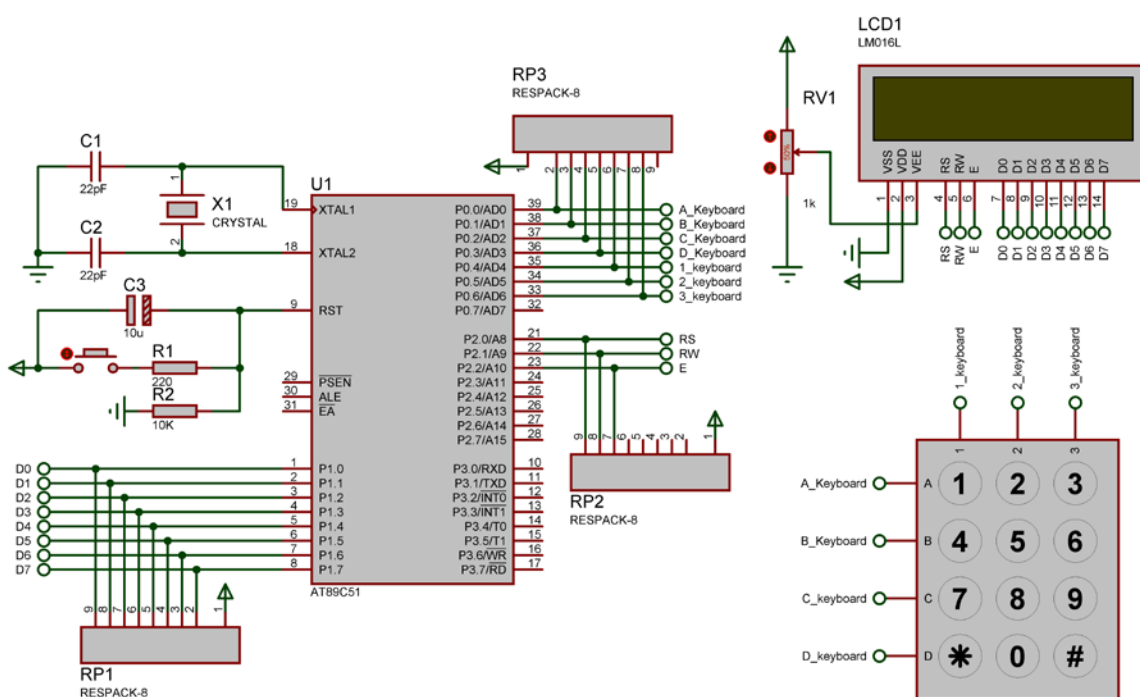
设计一个模拟电话拨号的显示装置，即把电话键盘中拨出的某一电话号码，显示在 LCD 显示屏上。电话键盘共有 12 个键，除了 0~9 的 10 个数字键外，还有“\*”键用于实现删除功能，即删除位最后输入的号码；“#”键用于清除显示屏上所有的数字显示。还要求每按下一个键要发出声响，以表示按下该键。

### 2. 实验原理

本题目涉及单片机与 4x3 矩阵式键盘的接口设计以及与 16x2 的液晶显示屏的接口设计，以及如何驱动蜂鸣器。液晶显示屏采用 LM016L(LCD1602)LCD,显示共 2 行，每行 16 个字符。第 1 行为设计者信息，第 2 行开始显示所拨的电话号码，最多为 16 位(因为 LCD 的-行能显示 16 个字符)。

### 3. 系统电路的设计

#### 3.1 电路原理图



### 3.2 电子元器件型号和数量

| 名称              | 数量 | 位号          |
|-----------------|----|-------------|
| 9C04021A2200FLH | 1  | R1          |
| 9C08052A1002JLH | 1  | R2          |
| 02013A220JAT2A  | 2  | C1,C2       |
| AT89C51         | 1  | U1          |
| CRYSTAL         | 1  | X1          |
| B45196H4106K309 | 1  | C3          |
| BUTTON          | 1  | -           |
| KEYPAD-PHONE    | 1  | -           |
| LM016L          | 1  | LCD1        |
| POT-HG          | 1  | RV1         |
| RESPACK-8       | 3  | RP1,RP2,RP3 |

### 3.3 电路工作原理的重点说明

#### 3.3.1 时钟电路与复位电路

电容与晶振片并联接在 XTAL1，XTAL2 构成时钟电路

电容与按钮并联接在 RST 上

#### 3.3.2 键盘控制模块

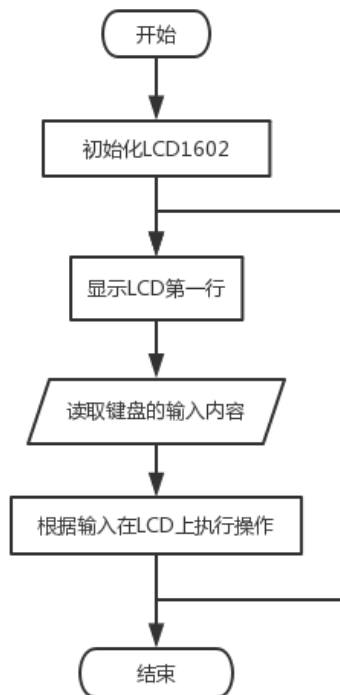
键盘 7 个接口接在 P0 上，通过 P0 读取输入内容

#### 3.3.3 LCD 控制模块

LCD 的 D0~7 接在 P1 上，RS、RW、E 接在 P2 上，通过单片机控制

## 4. 系统控制程序的设计

### 4.1 控制程序的工作流程图



## 4.2 控制程序的源代码

```

/*****
* @Author: fuqingchen@whut.edu.cn
* @Date: 2018-11-12 20:06:29
* @Description: 电话键盘&拨号的模拟
*****/

#include <reg51.h>
#include <intrins.h>
#include <string.h>
#define KEY P0 //分别读取A,B,C,D,1,2,3
#define LCD P1 //分别控制D0,D1,D2,D3,D4,D5,D6,D7
sbit RS = P2 ^ 0; //控制RS
sbit RW = P2 ^ 1; //控制RW
sbit E = P2 ^ 2; //控制E
unsigned char str[16]; //LCD1602 第二行输出内容

/*****
* @Description: 延时函数
* @Input: t:延长的时间
*****/

void delay(unsigned int t) {
    unsigned char i;
    while (t--) {

```

```

        for (i = 0; i<120; i++);
    }
}

/*****
 * @Description: LCD1602 读状态(查忙)
 *****/

void check_busy_LCD() {
    unsigned char pin;
    do
    {
        pin = 0xff;
        E = 0;
        RS = 0; RW = 1; E = 1;    //LCD 读状态控制信号
        pin = LCD;
    } while (pin & 0x80);
    E = 0;
}

/*****
 * @Description: LCD1602 写命令
 * @Input: com:要写入的命令
 *****/

void write_command_LCD(unsigned char com) {
    check_busy_LCD();
    RS = 0; RW = 0;    //LCD 写命令控制信号
    E = 0; LCD = com; E = 1;    //写命令正脉冲
    _nop(); E = 0; delay(1);
}

/*****
 * @Description: LCD1602 写数据
 * @Input: date:要写入的数据
 *****/

void write_data_LCD(unsigned char date) {
    check_busy_LCD();
    RS = 1; RW = 0;    //LCD 写数据控制信号
    E = 0; LCD = date; E = 1;    //写数据正脉冲
    _nop(); E = 0; delay(1);
}

/*****
 * @Description: LCD1602 初始化
 *****/

```

```

void initial_LCD() {
    write_command_LCD(0x38); //两行显示, 5*7 点阵, 8 位数据接口
    _nop_();
    write_command_LCD(0x0C); //整体显示, 光标关, 无显示
    _nop_();
    write_command_LCD(0x04); //整屏不移位
    _nop_();
    write_command_LCD(0x01); //显示清屏
}

/*****
* @Description: LCD1602 显示字符串
* @Input: address:字符串位置;*s:字符串;Length:字符串长度
*****/

void show_string_LCD(unsigned char address, unsigned char *s, unsigned int length) {
    unsigned int i;
    write_command_LCD(address);
    delay(5);
    for (i = 0; i < length; i++)
    {
        write_data_LCD(*(s + i));
    }
}

/*****
* @Description: 根据输入在LCD 上的操作
* @Input: no:读取的键盘字符[0,1,2,3,4,5,6,7,8,9,*,#]
*****/

void get_No(int no) {
    char str2;
    if (no < 10) //输入数字时, 加一位数字
    {
        str2 = no + 48; //将no 给str2,48 为ASCII 码
        if (strlen(str)<16)
        {
            strcat(str, &str2); //加一位数字
        }
    }
    else if (no == 10) //输入 * 时, 去一位数字
    {
        write_command_LCD(0x01); //显示清屏
        delay(5);
        show_string_LCD(0x82, "#FU_QINGCHEN", 12);
        *(str + strlen(str) - 1) = 0;
    }
}

```

```

        *(str + strlen(str)) = 0;
    }
    else    //输入 # 时, 清屏
    {
        *str = 0;
        write_command_LCD(0x01); //显示清屏
        delay(5);
        show_string_LCD(0x82, "#FU_QINGCHEN", 12);
        _nop_();
        delay(500);
    }
    show_string_LCD(0xC0, str, strlen(str));
    delay(5);
}

/*****
 * @Description: 输入 # 时清屏
 *****/

void get_sign() {
    //write_command_LCD(0x01);    //显示清屏
    delay(50);
    show_string_LCD(0x82, "#FU_QINGCHEN", 12);
    _nop_();
    delay(100);
}

/*****
 * @Description: 读取键盘的输入内容
 *****/

void getKeyValue() {
    int i;
    unsigned char R; //行扫描值
    unsigned char C; //列扫描值
    unsigned char RC;    //坐标
    unsigned char code_key;
    KEY = 0x0F; //ABCD 取高电平, 123 取低电平
    if (KEY!=0x0F)    //KEY 出现变化说明输入
    {
        delay(15);    //去抖动
        if ((KEY|0xF0)!=0xFF)
        {
            R = (~KEY) & 0x0F;    //获取行值
            code_key = 0xEF;
            C = 0x10;

```

```

        for (i = 0; i < 3; i++)
        {
            KEY = code_key;
            if ((KEY|0xF0)!=0xFF)
            {
                RC = R | C;
                switch (RC)
                {
                    case 0x28:get_No(0); break;
                    case 0x11:get_No(1); break;
                    case 0x21:get_No(2); break;
                    case 0x41:get_No(3); break;
                    case 0x12:get_No(4); break;
                    case 0x22:get_No(5); break;
                    case 0x42:get_No(6); break;
                    case 0x14:get_No(7); break;
                    case 0x24:get_No(8); break;
                    case 0x44:get_No(9); break;
                    case 0x18:get_No(10); break;
                    case 0x48:get_No(11); break;
                }
                break;
            }
            code_key = _crol_(code_key, 1);
            C = _crol_(C, 1);
        }
    }
}

void main() {
    initial_LCD();
    while (1)
    {
        get_sign();
        getKeyValue();
        //write_command_LCD(0x01);    //清屏
        delay(100);
    }
}

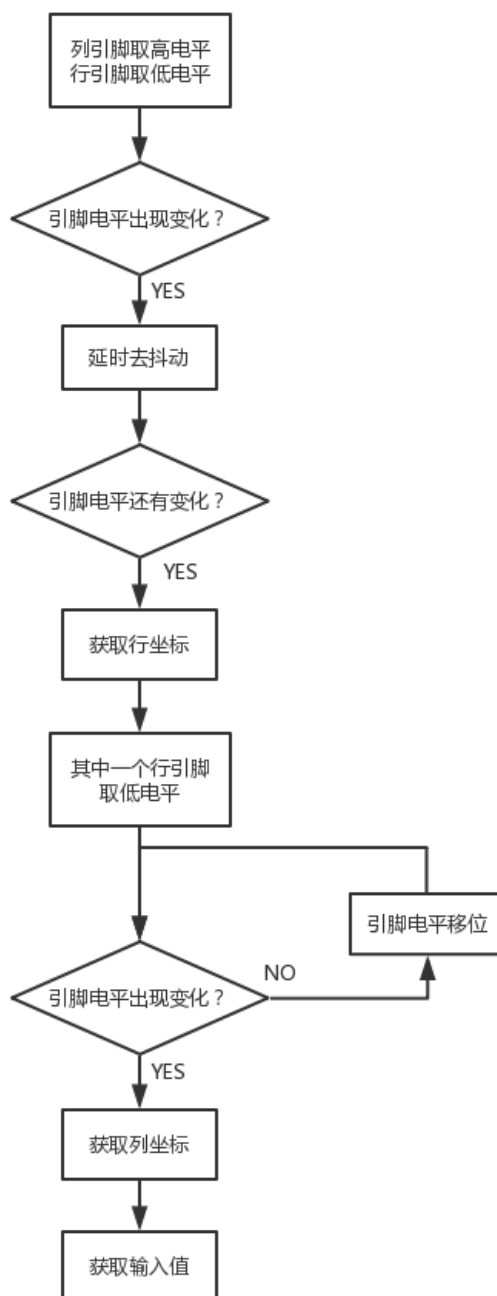
```

#### 4.3 控制程序的重点说明



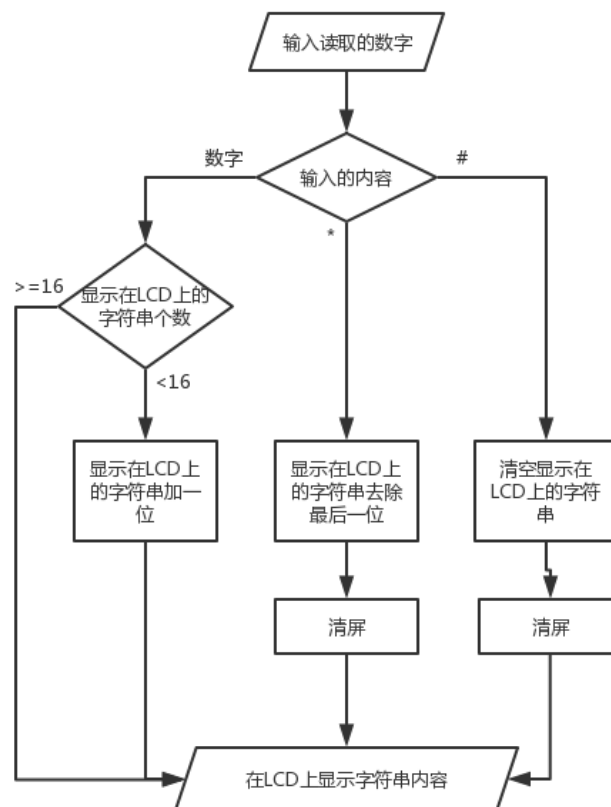
#### 4.3.1 读取键盘输入内容的函数 (getKeyValue 函数)

```
/*  
*****  
* @Description: 读取键盘的输入内容  
*****  
*/
```



#### 4.3.2 根据输入在 LCD 进行操作的函数 (get\_No 函数)

```
/*  
*****  
* @Description: 根据输入在 LCD 上的操作  
* @Input: no: 读取的键盘字符[0,1,2,3,4,5,6,7,8,9,*,#]  
*****  
*/
```



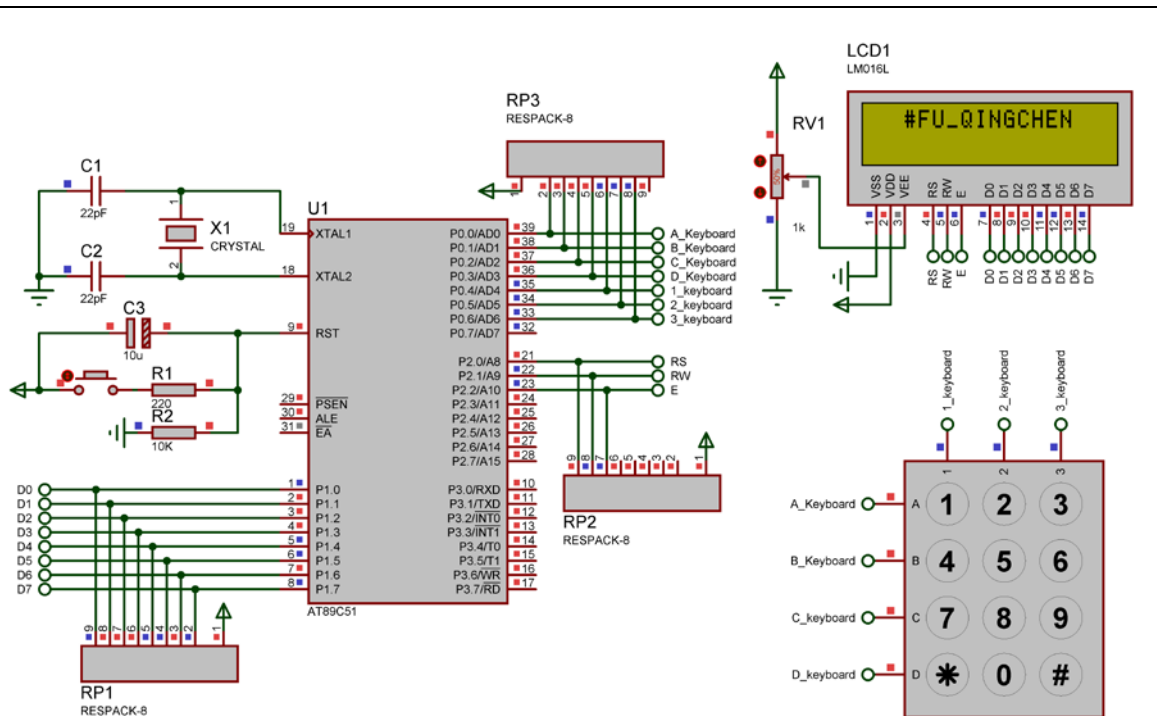
## 5. 系统的调试和结果

### 5.1 系统调试的方法

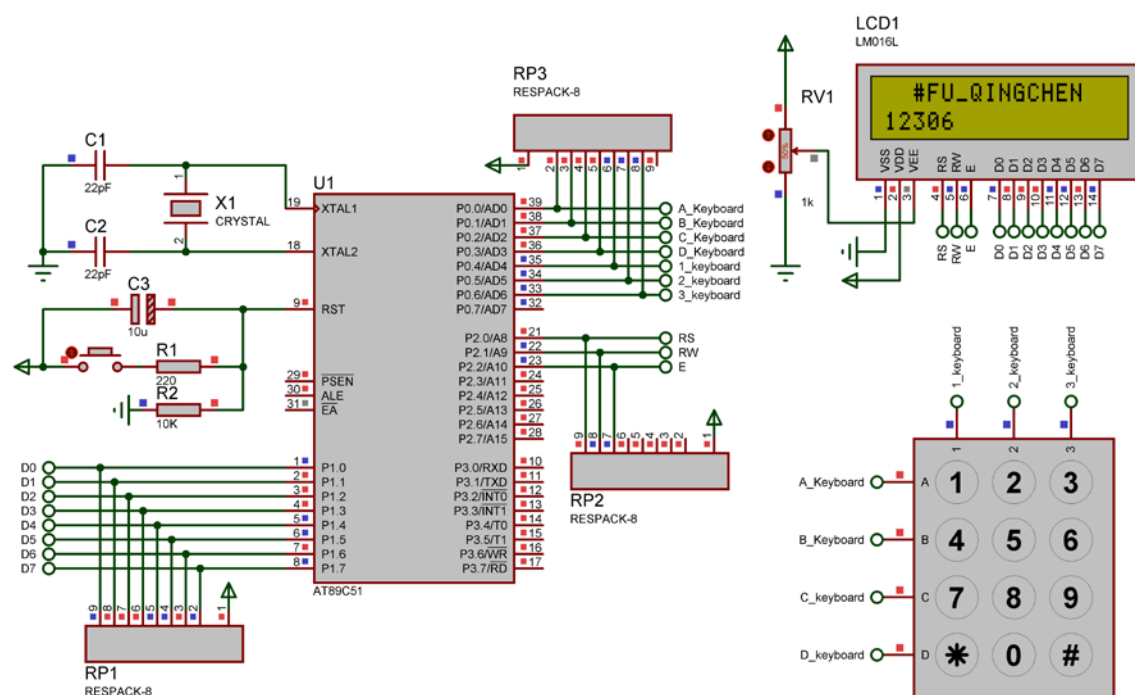
1. 在 Keil uVision 中将写好的控制程序编译生成 hex 文件
2. 在 Proteus 画好电路图后，将软件中的模拟单片机与生成的 hex 文件链接
3. 在 Proteus 中进行仿真，观察现象。
4. 若不满足要求，重新修改控制程序并生成 hex 文件

### 5.2 系统正确运行的图片

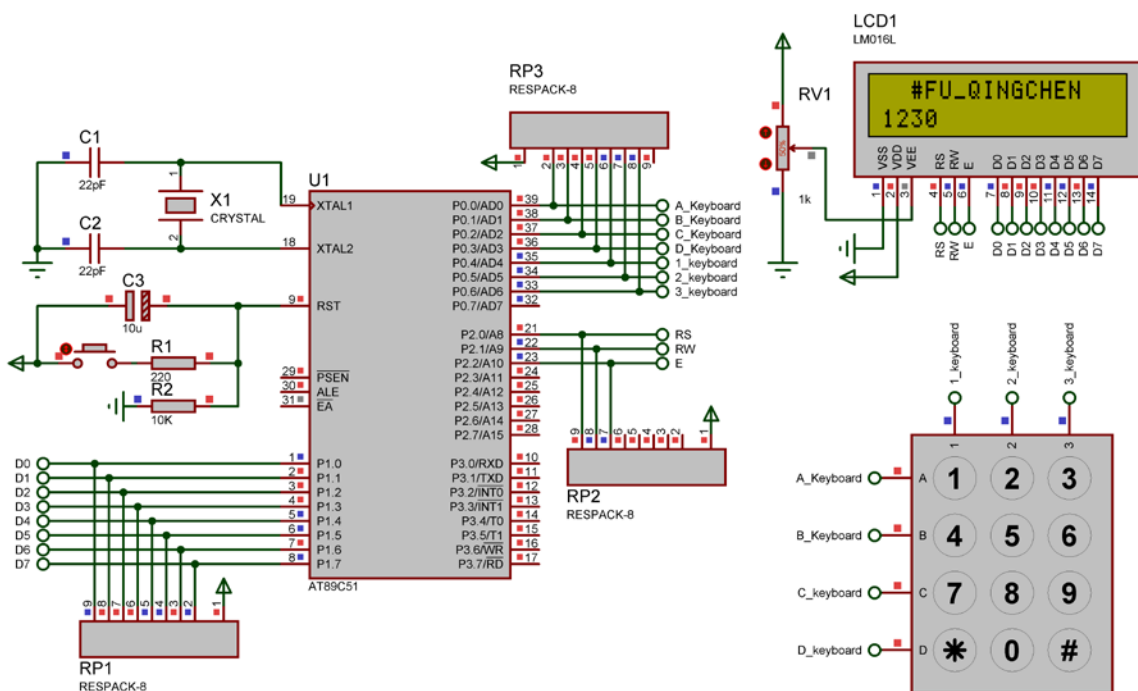
#### 5.2.1 电话键盘开启时：



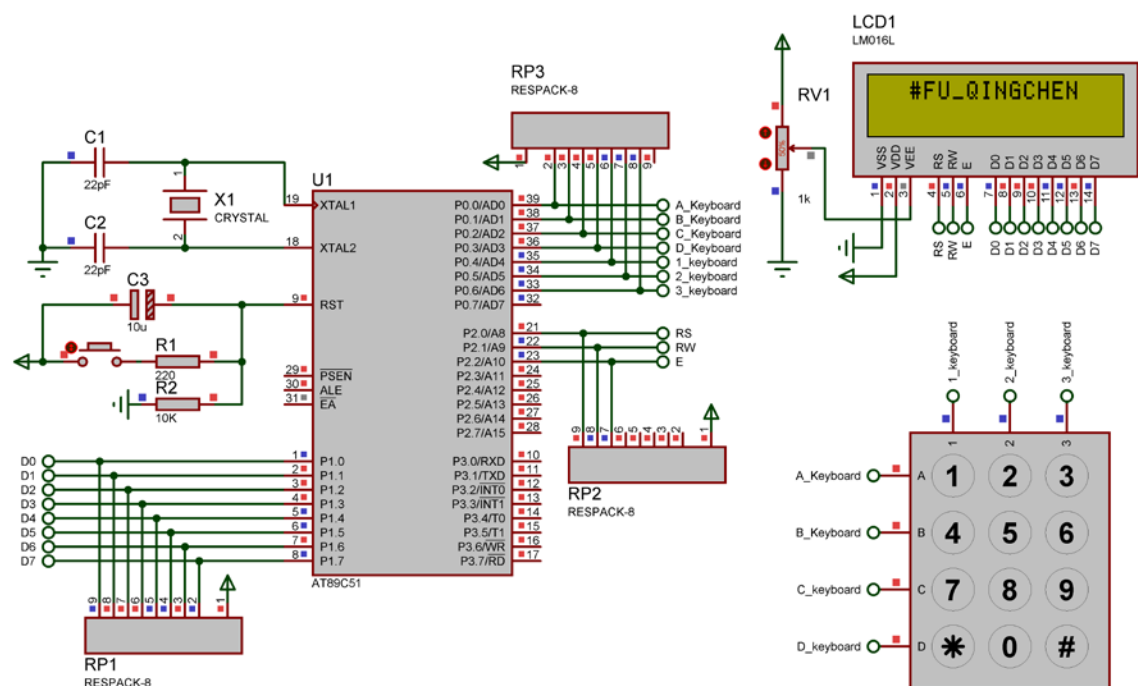
5.2.2 电话键盘输入[数字]时，显示输入的数字：



5.2.3 电话键盘输入[\*]时，删除最后一位输入的数字：



5.2.4 电话键盘输入[#]时，清楚显示屏上所以的数字显示：



## 6. 实验过程中所遇到的问题，解决方法和建议

### 6.1 【C 语言语法】变量的申明

#### 6.1.1 问题的描述

我一开始的循环是这样写的：

```
for (int i = 0; i < length; i++){
    //Do Some Thing
```

```
}
```

但是我发现标准 C 语言不支持临时变量在 for 循环中定义

### 6.1.2 解决方法

在函数的开始申明变量，之后再使用变量

```
int i;
for (i = 0; i < length; i++){
    //Do Some Thing
}
```

## 6.2 【LCD1602】显示器字符的去除

### 6.2.1 问题的描述

写字符串的函数

```
/*
*****
* @Description: LCD1602 显示字符串
* @Input: address: 字符串位置; *s: 字符串; length: 字符串长度
*****
*/
void show_string_LCD(unsigned char address, unsigned char *s, unsigned int length) {
    unsigned int i;
    write_command_LCD(address);
    delay(5);
    for (i = 0; i < length; i++)
    {
        write_data_LCD(*(s + i));
    }
}
```

只能对 LCD1602 上面的字符进行改变，但是无法删除。

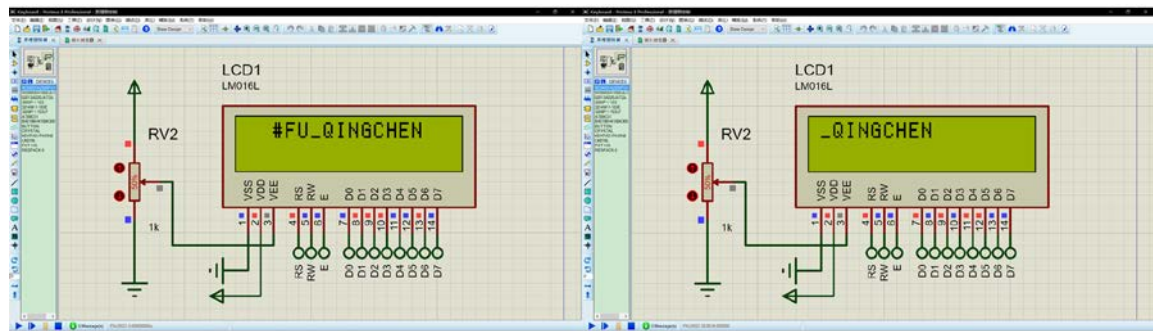
### 6.2.2 解决方法

当有删除字符的操作时，先**清除屏幕内容**，再**加载删除后的内容**。

## 6.3 【LCD1602】屏幕的移动

### 6.3.1 问题的提出

按照课本上的代码，每当我输入一个字符时，整个屏幕都会向左移动。但是按照要求，第 1 行为设计者信息，这个是不希望移动的，第二行是可以移动的



添加数字前

添加数字后

### 6.3.2 解决方法

在 LCD1602 初始化时，**设置为整屏不移动**，以保持第一行正常移动；记录每次输入值，**通过软件算法给出需要显示在 LCD 上的数字**，实现第二行的移动。

#### 1. 更改 LCD1602 初始化设置

```

/*****
@Description: LCD1602 初始化
*****/

void initial_LCD() {
    write_command_LCD(0x38); //两行显示, 5*7 点阵, 8 位数据接口
    _nop_();
    write_command_LCD(0x0C); //整体显示, 光标关, 无显示
    _nop_();
    write_command_LCD(0x04); //整屏不移位
    _nop_();
    write_command_LCD(0x01); //显示清屏
}

```

#### 2. 通过软件算法给出需要显示在 LCD 第二排的内容

```

/*****
* @Description: 根据输入在 LCD 上的操作
* @Input: no: 读取的键盘字符[0,1,2,3,4,5,6,7,8,9,*,#]
*****/

void get_No(int no) {
    char str2;
    if (no < 10) //输入数字时, 加一位数字
    {
        str2 = no + 48; //将no 给 str2, 48 为 ASCII 码
        if (strlen(str)<16)
        {
            strcat(str, &str2); //加一位数字
        }
    }
}

```

```

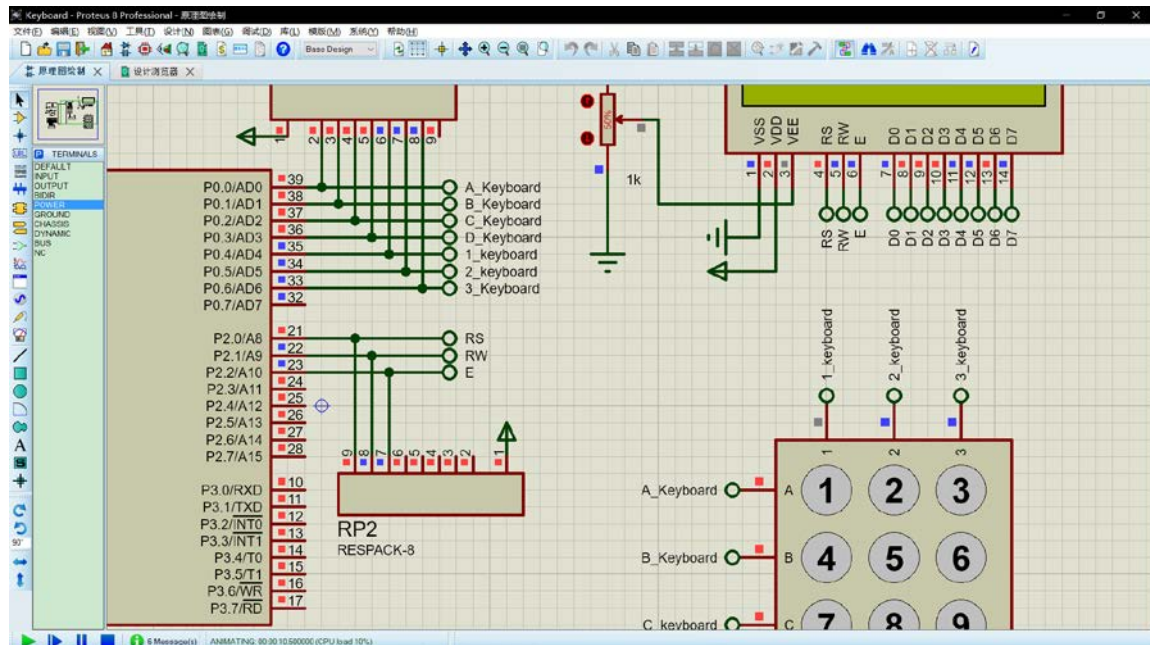
else if (no == 10)    //输入 * 时，去一位数字
{
    write_command_LCD(0x01); //显示清屏
    delay(5);
    show_string_LCD(0x82, "#FU_QINGCHEN", 12);
    *(str + strlen(str) - 1) = 0;
    *(str + strlen(str)) = 0;
}
else    //输入 # 时，清屏
{
    *str = 0;
    write_command_LCD(0x01); //显示清屏
    delay(5);
    show_string_LCD(0x82, "#FU_QINGCHEN", 12);
    _nop();
    delay(500);
}
show_string_LCD(0xC0, str, strlen(str));
delay(5);
}

```

## 6.4 【Proteus】接口序号

### 6.4.1 问题的描述

按照电路图，[1\_keyboard] 引脚应该为低电平。但是测试时出现了 [1\_keyboard] 引脚点位不明确的现象

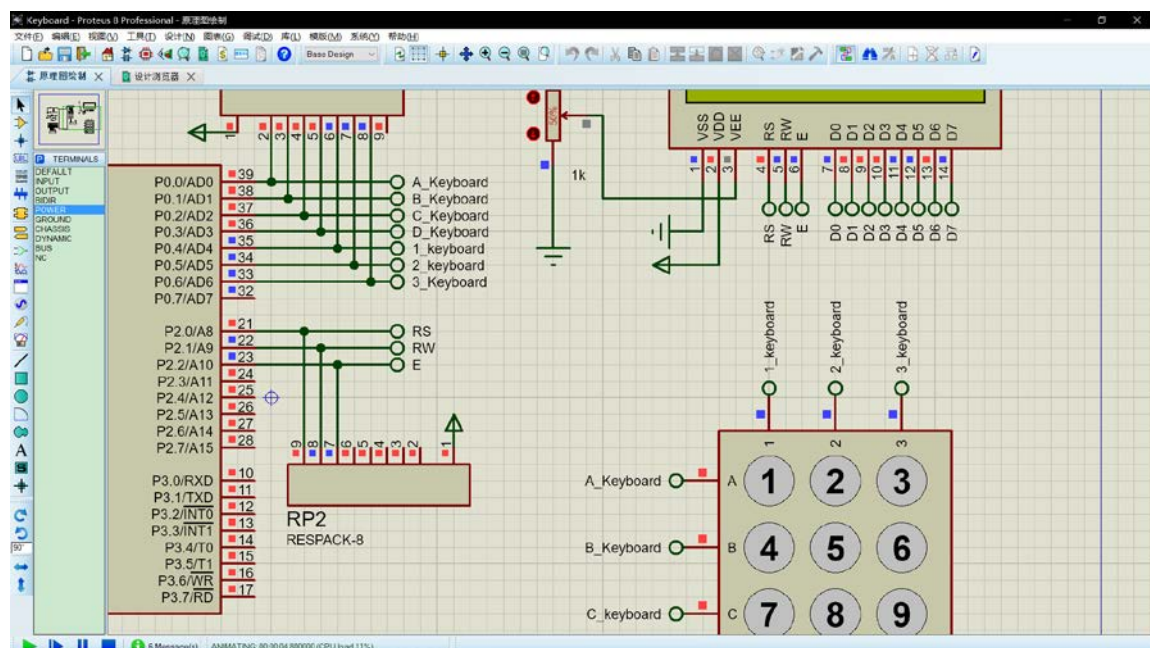


### 6.4.2 解决方法

将[AT89C51]上的接口名称由[1\_keyboard]改为[1\_keyboard ]



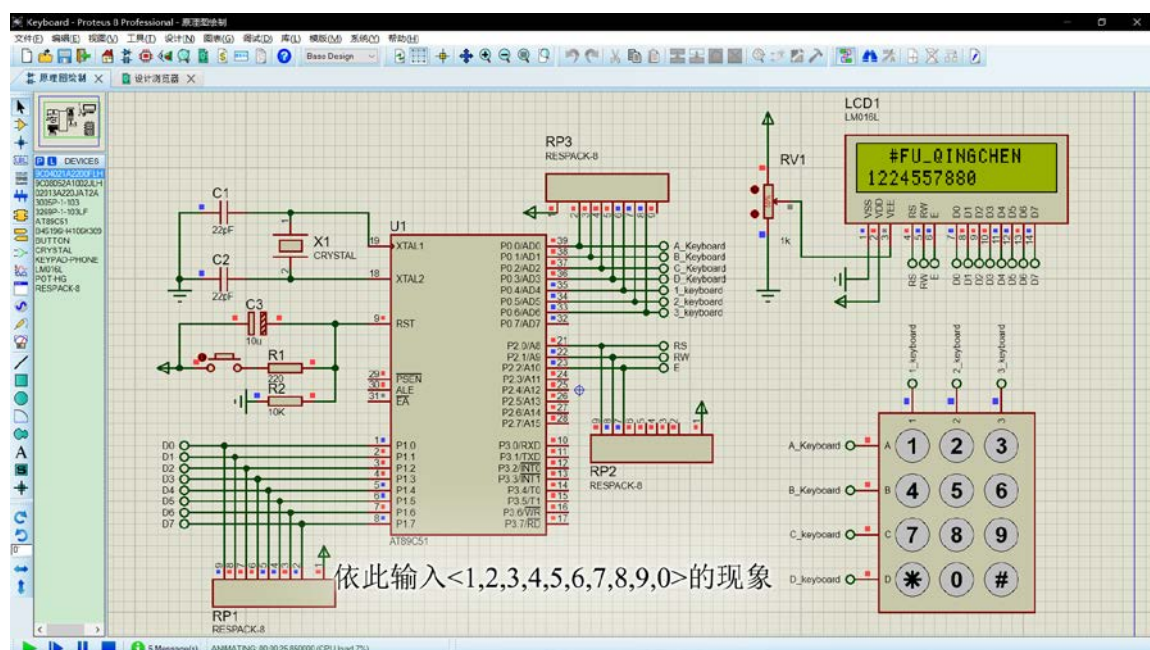
(在后面多加一个空格)



## 6.5 【C 语言程序】第 3 列输入与第 2 列相同

### 6.5.1 问题的描述

当键盘按下第三列的键（[3],[6],[9],[#]）时，读取的不是第三列的值而是第二列的值。



### 6.5.2 解决方法

给 P0 接口设置一个临时变量储存信息并调用后，恢复正常。