14. 4×4 矩阵式键盘识别技术

1. 实验任务

如图 4.14.2 所示，用 AT89S51 的并行口 P1 接 4×4 矩阵键盘，以 P1.0—P1.3 作输入线，以 P1.4—P1.7 作输出线；在数码管上显示每个按键的“0—F”序号。对应的按键的序号排列如图 4.14.1 所示

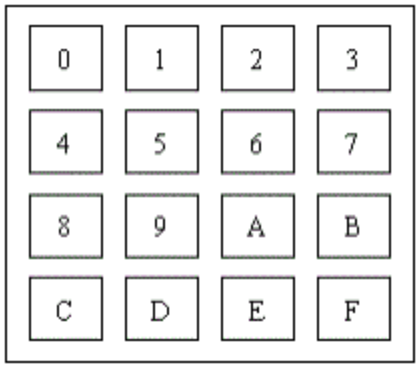

图 4.14.1

2. 硬件电路原理图

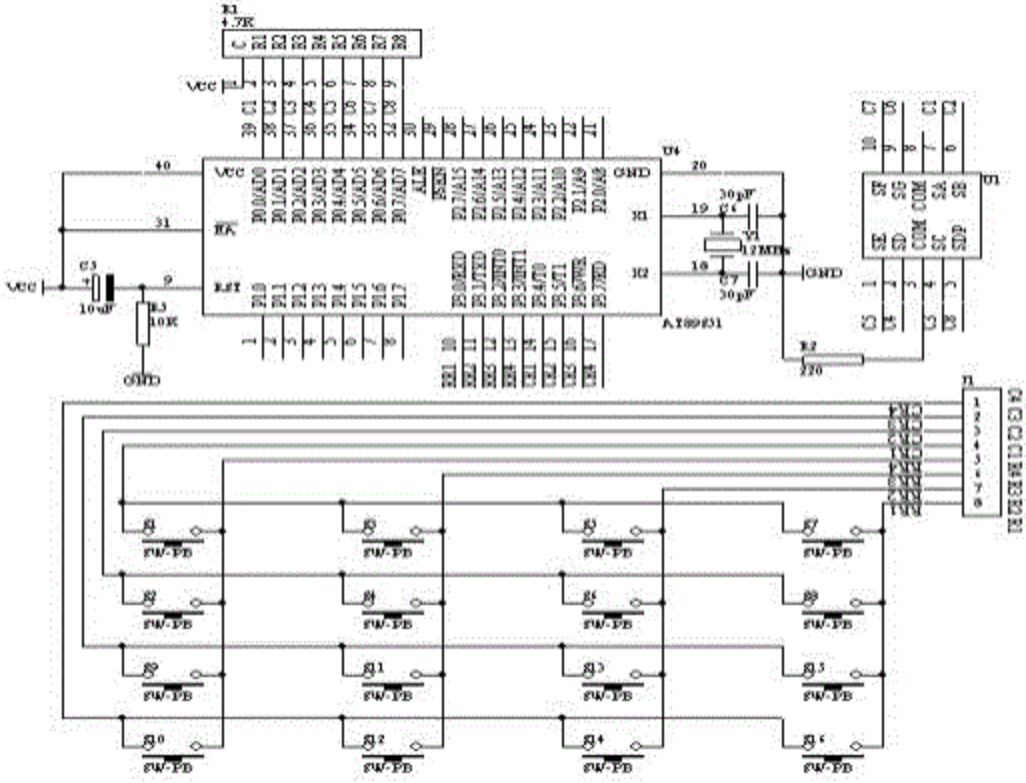

图 4.14.2

3. 系统板上硬件连线

（1. 把“单片机系统“区域中的 P3.0—P3.7 端口用 8 芯排线连接到“4X4 行列式键盘”区域中的 C1—C4　R1—R4 端口上；

（2. 把“单片机系统”区域中的 P0.0/AD0—P0.7/AD7 端口用 8 芯排线连接到“四路静态数码显示模块”区域中的任一个 a—h 端口上；要求：P0.0/AD0 对应着 a，P0.1/AD1 对应着 b，……，P0.7/AD7 对应着 h。

4. 程序设计内容

（1. 4×4 矩阵键盘识别处理

（2. 每个按键有它的行值和列值　，行值和列值的组合就是识别这个按键的编码。矩阵的行线和列线分别通过两并行接口和 CPU 通信。每个按键的状态同样需变成数字量“0”和“1”，开关的一端（列线）通过电阻接 VCC，而接地是通过程序输出数字“0”实现的。键盘处理程序的任务是：确定有无键按下，判断哪一个键按下，键的功能是什么；还要消除按键在闭合或断开时的抖动。两个并行口中，一个输出扫描码，使按键逐行动态接地，另一个并行口输入按键状态，由行扫描值和回馈信号共同形成键编码而识别按键，通过软件查表，查出该键的功能。

5. 程序框图

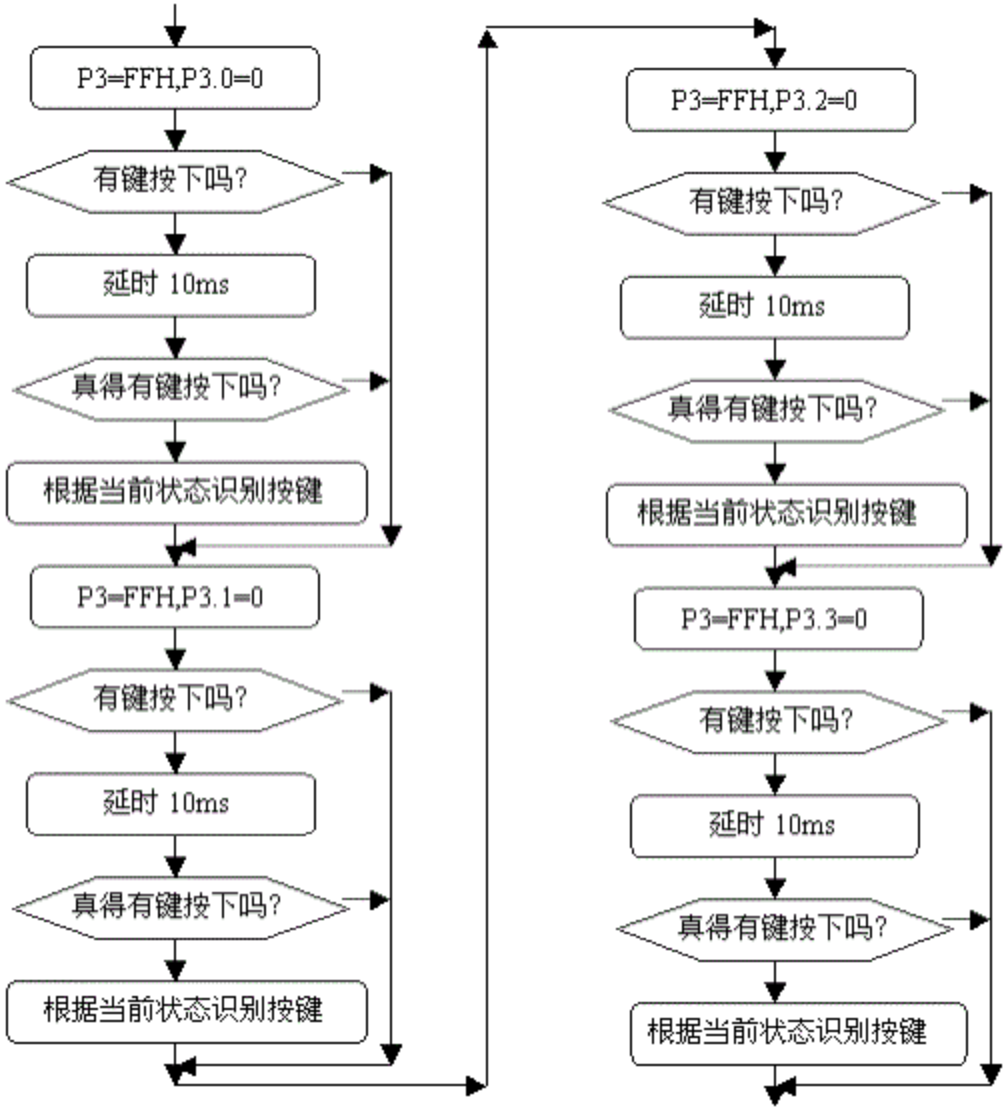

图 4.14.3

6. 汇编源程序

```
KEYBUF EQU 30H
ORG 00H
START: MOV KEYBUF,#2
WAIT:
MOV P3,#0FFH
CLR P3.4
MOV A,P3
ANL A,#0FH
XRL A,#0FH
JZ NOKEY1
LCALL DELY10MS
MOV A,P3
```

```
ANL A,#0FH
XRL A,#0FH
JZ NOKEY1
MOV A,P3
ANL A,#0FH
CJNE A,#0EH,NK1
MOV KEYBUF,#0
LJMP DK1
NK1: CJNE A,#0DH,NK2
MOV KEYBUF,#1
LJMP DK1
NK2: CJNE A,#0BH,NK3
MOV KEYBUF,#2
LJMP DK1
NK3: CJNE A,#07H,NK4
MOV KEYBUF,#3
LJMP DK1
NK4: NOP
DK1:
MOV A,KEYBUF
MOV DPTR,#TABLE
MOVC A,@A+DPTR
MOV P0,A

DK1A: MOV A,P3
ANL A,#0FH
XRL A,#0FH
JNZ DK1A
NOKEY1:
MOV P3,#0FFH
CLR P3.5
MOV A,P3
ANL A,#0FH
XRL A,#0FH
JZ NOKEY2
LCALL DELY10MS
MOV A,P3
ANL A,#0FH
XRL A,#0FH
JZ NOKEY2
MOV A,P3
ANL A,#0FH
CJNE A,#0EH,NK5
MOV KEYBUF,#4
```

```
LJMP DK2
NK5: CJNE A,#0DH,NK6
MOV KEYBUF,#5
LJMP DK2
NK6: CJNE A,#0BH,NK7
MOV KEYBUF,#6
LJMP DK2
NK7: CJNE A,#07H,NK8
MOV KEYBUF,#7
LJMP DK2
NK8: NOP
DK2:
MOV A,KEYBUF
MOV DPTR,#TABLE
MOVC A,@A+DPTR
MOV P0,A

DK2A: MOV A,P3
ANL A,#0FH
XRL A,#0FH
JNZ DK2A
NOKEY2:
MOV P3,#0FFH
CLR P3.6
MOV A,P3
ANL A,#0FH
XRL A,#0FH
JZ NOKEY3
LCALL DELY10MS
MOV A,P3
ANL A,#0FH
XRL A,#0FH
JZ NOKEY3
MOV A,P3
ANL A,#0FH
CJNE A,#0EH,NK9
MOV KEYBUF,#8
LJMP DK3
NK9: CJNE A,#0DH,NK10
MOV KEYBUF,#9
LJMP DK3
NK10: CJNE A,#0BH,NK11
MOV KEYBUF,#10
LJMP DK3
```

```
NK11: CJNE A,#07H,NK12
MOV KEYBUF,#11
LJMP DK3
NK12: NOP
DK3:
MOV A,KEYBUF
MOV DPTR,#TABLE
MOVC A,@A+DPTR
MOV P0,A

DK3A: MOV A,P3
ANL A,#0FH
XRL A,#0FH
JNZ DK3A
NOKEY3:
MOV P3,#0FFH
CLR P3.7
MOV A,P3
ANL A,#0FH
XRL A,#0FH
JZ NOKEY4
LCALL DELY10MS
MOV A,P3
ANL A,#0FH
XRL A,#0FH
JZ NOKEY4
MOV A,P3
ANL A,#0FH
CJNE A,#0EH,NK13
MOV KEYBUF,#12
LJMP DK4
NK13: CJNE A,#0DH,NK14
MOV KEYBUF,#13
LJMP DK4
NK14: CJNE A,#0BH,NK15
MOV KEYBUF,#14
LJMP DK4
NK15: CJNE A,#07H,NK16
MOV KEYBUF,#15
LJMP DK4
NK16: NOP
DK4:
MOV A,KEYBUF
MOV DPTR,#TABLE
```

```
MOVC A,@A+DPTR
MOV P0,A

DK4A: MOV A,P3
ANL A,#0FH
XRL A,#0FH
JNZ DK4A
NOKEY4:
LJMP WAIT
DELY10MS:
MOV R6,#10
D1: MOV R7,#248
DJNZ R7,$
DJNZ R6,D1
RET
TABLE: DB 3FH,06H,5BH,4FH,66H,6DH,7DH,07H
DB 7FH,6FH,77H,7CH,39H,5EH,79H,71H
END
```

7. C语言源程序

```
#include <AT89X51.H>
unsigned char code table[]={0x3f,0x06,0x5b,0x4f,
0x66,0x6d,0x7d,0x07,
0x7f,0x6f,0x77,0x7c,
0x39,0x5e,0x79,0x71};
unsigned char temp;
unsigned char key;
unsigned char i,j;


void main(void)
{
while(1)
{
P3=0xff;
P3_4=0;
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
{
for(i=50;i>0;i--)
for(j=200;j>0;j--);
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
```

```
{
temp=P3;
temp=temp & 0x0f;
switch(temp)
{
case 0x0e:
key=7;
break;
case 0x0d:
key=8;
break;
case 0x0b:
key=9;
break;
case 0x07:
key=10;
break;
}
temp=P3;
P1_0=~P1_0;
P0=table[key];
temp=temp & 0x0f;
while(temp!=0x0f)
{
temp=P3;
temp=temp & 0x0f;
}
}
}

P3=0xff;
P3_5=0;
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
{
for(i=50;i>0;i--)
for(j=200;j>0;j--);
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
{
temp=P3;
temp=temp & 0x0f;
```

```
switch(temp)
{
case 0x0e:
key=4;
break;
case 0x0d:
key=5;
break;
case 0x0b:
key=6;
break;
case 0x07:
key=11;
break;
}
temp=P3;
P1_0=~P1_0;
P0=table[key];
temp=temp & 0x0f;
while(temp!=0x0f)
{
temp=P3;
temp=temp & 0x0f;
}
}
}

P3=0xff;
P3_6=0;
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
{
for(i=50;i>0;i--)
for(j=200;j>0;j--);
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
{
temp=P3;
temp=temp & 0x0f;
switch(temp)
{
case 0x0e:
```

```
key=1;
break;
case 0x0d:
key=2;
break;
case 0x0b:
key=3;
break;
case 0x07:
key=12;
break;
}
temp=P3;
P1_0=~P1_0;
P0=table[key];
temp=temp & 0x0f;
while(temp!=0x0f)
{
temp=P3;
temp=temp & 0x0f;
}
}
}

P3=0xff;
P3_7=0;
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
{
for(i=50;i>0;i--)
for(j=200;j>0;j--);
temp=P3;
temp=temp & 0x0f;
if (temp!=0x0f)
{
temp=P3;
temp=temp & 0x0f;
switch(temp)
{
case 0x0e:
key=0;
break;
case 0x0d:
```

```
key=13;
break;
case 0x0b:
key=14;
break;
case 0x07:
key=15;
break;
}
temp=P3;
P1_0=~P1_0;
P0=table[key];
temp=temp & 0x0f;
while(temp!=0x0f)
{
temp=P3;
temp=temp & 0x0f;
}
}
}
}
}
```