实验八 字符型背光液晶显示屏

杨庆龙
1500012956

2018.5.16

1 实验目的

1. 了解字符型液晶显示模块的使用方法
2. 掌握并行端口模拟接口时序的方法

2 实验原理

2.1 硬件接口

1. 1:VSS接地
2. 2:VDD接5V电源
3. 3:V0用于对比度调节
4. 4:RS寄存器，高电平时选择数据，低电平时选择指令
5. 5:RW寄存器，高电平为读操作，低电平为写操作
6. 6:使能端
7. 7-14:双向数据线
8. 15-16:空引脚

2.2 指令

1. 0x38:表示8位地址总线，双行显示
2. 0x08:关闭显示
3. 0x01:表示清屏
4. 0x06:表示右移光标，文字不平移
5. 0x0C:打开显示，不显示 光标
6. 0x80:设置数据储存器的位置
7. 0x02:光标位置复位

3 ““““

```
#include <C8051F020.h>
#include "../../includes/charlcd.h"
#include "../../includes/time.h"
#include "../../includes/keyboard.h"
#include "../../includes/storage.h"
#define STATE_PUTIN 0
#define STATE_DONE 1
#define CACU_ERROR 1
#define CACU_OK 0

unsigned char upbuffer[32];
unsigned char upcount;
unsigned char downbuffer[16];
unsigned char downcount;
unsigned char numbuf[16];
unsigned char cacubuf[16];
unsigned char numindex;
unsigned char cacuindex;
unsigned char level[] = {0,1,1,2,2};
void add(unsigned char putin, unsigned char up) {
     unsigned char temp;
     switch(putin) {
     case 'E':temp = 0x45;
     break;
     case 'R':temp = 0x52;
     break;
     case 'O':temp = 0x4f;
     break;
     case 0x0C:temp = 0x2B;
     break;
     case 0x0D:temp = 0x2D;
     break;
     case 0x0E:temp = 0x2A;
     break;
     case 0x0F:temp = 0x2F;
     break;
     default: temp = putin + 0x30;
     }
     temp = temp;
     if(up == 1)
             upbuffer[upcount ++] = temp;
     else
             downbuffer[downcount ++] = temp;
}
unsigned char pop() {
     while(cacuindex > 1) {
             if(level[cacubuf[cacuindex - 1]] >= level[cacubuf[cacuindex - 2]]) {
                     if(numindex > 1) {
                             switch(cacubuf[cacuindex-1]) {
                                     case 1:numbuf[numindex - 2] = numbuf[numindex - 2]
                                     break;
                                     case 2:numbuf[numindex - 2] = numbuf[numindex - 2]
                                     break;
                                     case 3:numbuf[numindex - 2] = numbuf[numindex - 2]
```

```
                                        break;
                                        case 4:numbuf[numindex - 2] = numbuf[numindex - 2]
                                        break;
                                }
                                numindex --;
                                cacuindex --;
                        }
                        else
                                return CACU_ERROR;
                }
        }
        return CACU_OK;
}
void cacu() {
        unsigned char i;
        unsigned char tempnum = 0;
        unsigned char result;
        numindex = 0;
        cacuindex = 1;
        cacubuf[0] = 0;
        for(i = 0;i < upcount;i ++) {
                if(upbuffer[i] >= 0x30 && upbuffer[i] <= 0x39) {
                        tempnum *= 10;
                        tempnum = upbuffer[i] - 0x30 + tempnum;
                }
                else {
                        switch(upbuffer[i]) {
                        case 0x2B:numbuf[numindex++] = tempnum;
                        result = pop();
                        cacubuf[cacuindex++] = 1;
                        break;
                        case 0x2D:numbuf[numindex++] = tempnum;
                        result = pop();
                        cacubuf[cacuindex++] = 2;
                        break;
                        case 0x2A:numbuf[numindex++] = tempnum;
                        cacubuf[cacuindex++] = 3;
                        break;
                        case 0x2F:numbuf[numindex++] = tempnum;
                        cacubuf[cacuindex++] = 4;
                        }
                        tempnum = 0;
                }
        }
        numbuf[numindex++] = tempnum;
        result = pop();
        if(result == CACU_ERROR) {
                add('E',0);
                add('R',0);
                add('R',0);
                add('O',0);
                add('R',0);
        }
        else {
                numindex = 1;
                do {
```

```
111                     numbuf[numindex ++] = numbuf[0] % 10;
112                     numbuf[0] /= 10;
113             }while(numbuf[0] != 0);
114             numindex --;
115             while(numindex > 0) {
116                     add(numbuf[numindex --],0);
117             }
118     }
119 }
120
121
122 void main(void) {
123     unsigned char nowkey;
124     unsigned char prekey;
125     unsigned char state;
126     unsigned char i;
127     unsigned char startpos;
128     unsigned char count;
129     WDTCN = 0xDE;
130     WDTCN = 0xAD;
131     sysclk_init();
132     P74OUT = 0x33;
133     lcd_init();
134     upcount = 0x00;
135     downcount = 0;
136     prekey = NOKEY;
137     state = STATE_PUTIN;
138     count = 0;
139     while(1) {
140             nowkey = getKey();
141             //count ++;
142             count %= 10;
143             if(nowkey != NOKEY && nowkey != prekey) {
144                     if(state == STATE_DONE) {
145                             upcount = 0;
146                             downcount = 0;
147                             cacubuf[0] = 0;
148                             lcd_init();
149                             state = STATE_PUTIN;
150                     }
151                     else {
152                             if(nowkey == 0x0A){
153                                     state = STATE_DONE;
154                                     cacu();
155                             }
156                             else if(nowkey == 0x0B) {
157                                     upcount = 0;
158                                     downcount = 0;
159                                     cacubuf[0] = 0;
160                                     lcd_init();
161                             }
162                             else
163                                     add(nowkey,1);
164                     }
165             }
166             prekey = nowkey;
```

```
167         ////////////display
168         if(count == 0){
169         if(upcount < 16)
170         startpos = 0;
171         else
172         startpos = upcount - 16;
173         lcd_write_command(0x80);
174         for(i = 0;i < upcount - startpos;i ++)
175                 lcd_write_data(upbuffer[i + startpos]);
176         if(startpos < 16)
177         startpos = 0;
178         else
179         startpos = downcount - 16;
180         lcd_write_command(0xC0);
181         for(i = 0;i < downcount - startpos;i ++)
182                 lcd_write_data(downbuffer[i + startpos]);
183         }
184         }
185
186     }
```