JLX200-01701-BN 使用说明书

（插接式 FPC）

目 录

1．概述

晶联讯电子专注于液晶屏及液晶模块的研发、制造。所生产 JLX200-01701-BN 型液晶模块由于使用方便、显示清晰，广泛应用于各种人机交流面板。

JLX200-01701-BN 可以显示 176 列*220 行点阵彩色图片。

2．JLX200-01701-BN 彩色图像型点阵液晶模块的特性

2.1 结构轻、薄、带背光、插接式 FPC。
2.2 IC 采用 ST7775R,功能强大，稳定性好
2.3 指令功能强：例如可以用指令控制显示内容顺时针旋转 90、逆时针旋转 90°或倒立竖放。
2.4 接口方式:并口、串口。
2.5 工作温度宽:-20℃ ~ 70℃;
2.6 储存温度宽:-30℃ ~ 80℃;

3. 外形尺寸及接口引脚功能

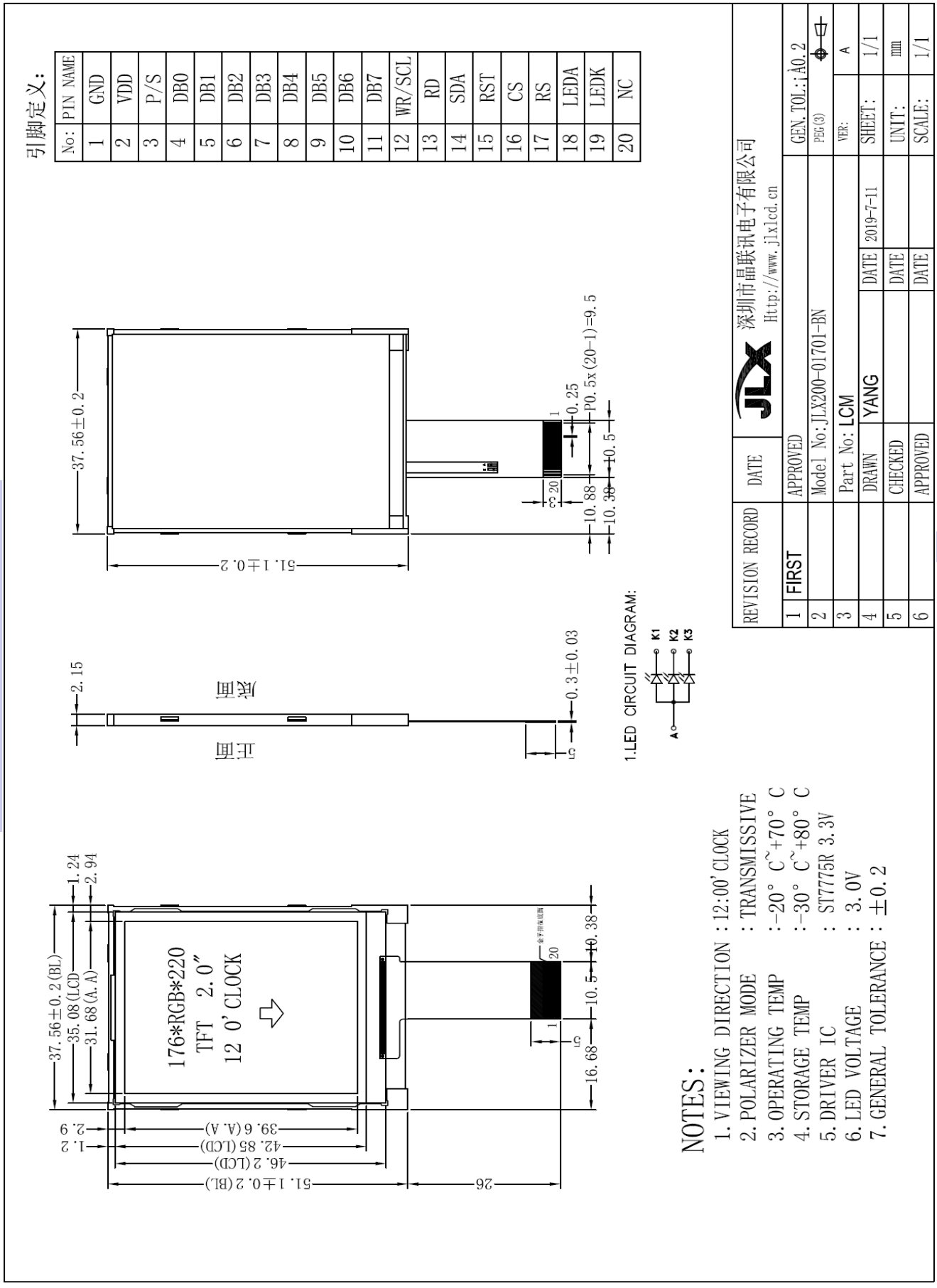


图 1. 外形尺寸

模块的接口引脚功能

引 线 号	符 号	名 称	功 能
1	GND	接地	0V
2	VDD	电路电源	3.3V
3	PS	串并选择	并口接地，串口接高
4	DB0	I/O	数据总线，（串口时空）
5	DB1	I/O	数据总线，（串口时空）
6	DB2	I/O	数据总线，（串口时空）
7	DB3	I/O	数据总线，（串口时空）
8	DB4	I/O	数据总线，（串口时空）
9	DB5	I/O	数据总线，（串口时空）
10	DB6	I/O	数据总线，（串口时空）
11	DB7	I/O	数据总线，（串口时空）
12	WR/SCL	I/O	并口做为 WR 用，串口做为 SCK 用
13	RD	I/O	使能信号，（串口时空）
14	SDA	I/O	串行数据（并口时，空）
15	RST	复位	低电平复位，复位完成后，回到高电平，液晶模块开始工作
16	CS	片选	低电平片选
17	RS	寄存器选择信号	H:数据寄存器 0:指令寄存器（IC 资料上所写为”A0”）
18	LDEA	背光电源正极	背光电源正极,接 3.0V
19	LEDK	背光电源负极	背光电源负极，接地
20	NC	空脚	空脚

表 1：模块的接口引脚功能

4. 基本原理

4.1 背光参数

该型号液晶模块带 LED 背光源。它的性能参数如下:

工作温度:-20∽+70° C;

存储温度:-30∽+80° C;

背光板是白色。

正常工作电流为：24∽60mA（LED 灯数共 3 颗,每颗灯是 8～20 mA）

工作电压：3.0V。

5. 技术参数

5.1 最大极限参数（超过极限参数则会损坏液晶模块）

名称	符号	标准值			单位
		最小	典型	最大	
电路电源	VDD	-0.3	3.3	3.3	V
工作温度		-20		+70	℃
储存温度		-30		+80	℃

表 2：最大极限参数

5.2 直流（DC）参数

名 称	符 号	测 试 条 件	标 准 值			单位
			最小	典型值	最大	
工作电压	VDD		2.8	3.0	3.3	V
背光工作电压	VLED		2.9	3.0	3.1	V
背光工作电流	ILED	VLED=3.0V， 共 3 颗 LED 灯并联	24	45	60	mA

表 3：直流（DC）参数

6. 读写时序特性

详见 IC 资料“ST7775R”，请找相关客服人员索要。

并行接口，8080 时序

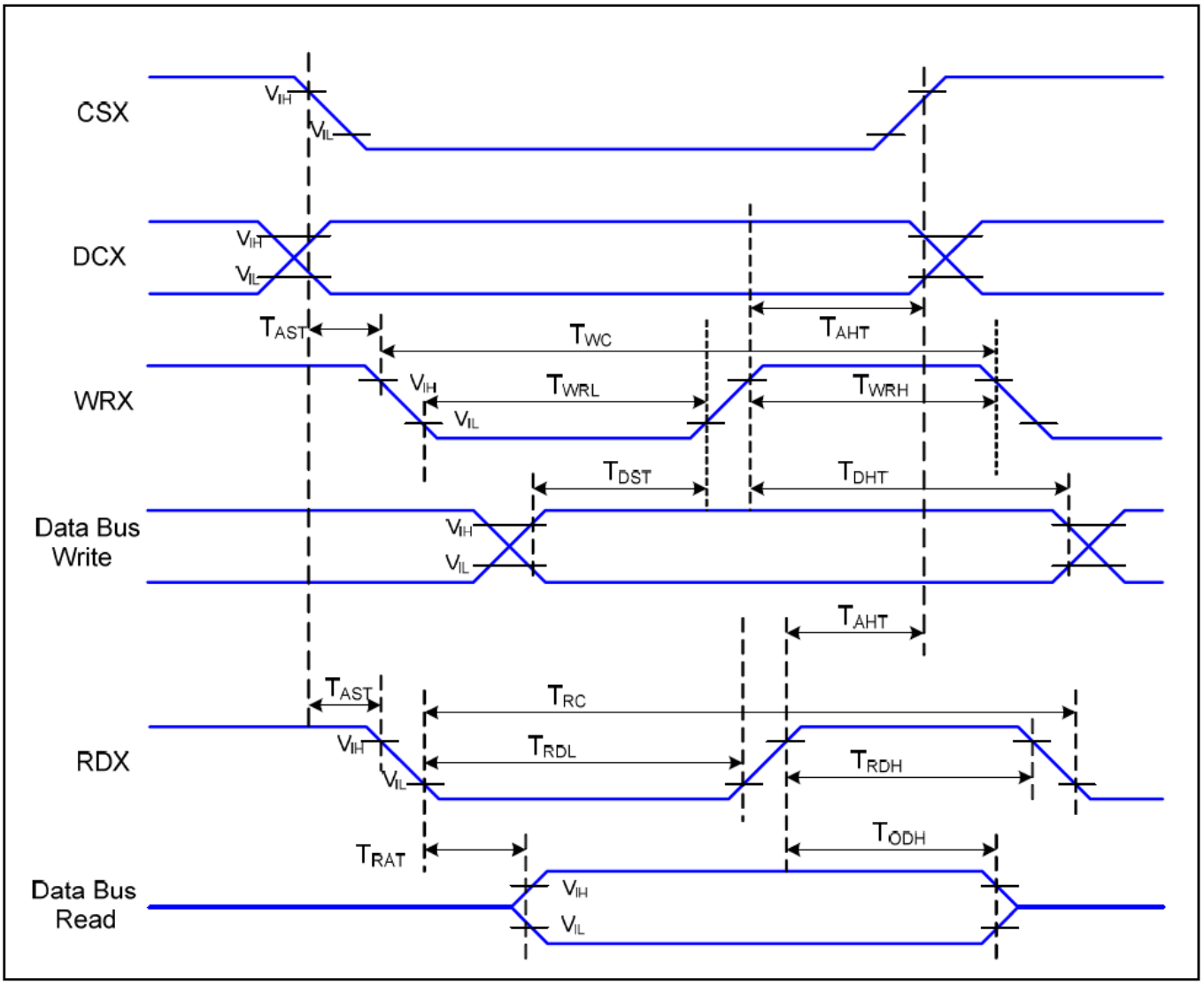


图 2

Signal	Symbol	Parameter	Min	Max	Unit	Description
DCX	TAST	Address Setup Time	TBD		ns	
	TAHT	Address Hold Time (Write/Read)	TBD		ns	
WRX	TWC	Write Cycle	TBD		ns	
	TWRH	Control Pulse “H” Duration	TBD		ns	
	TWRL	Control Pulse “L” Duration	TBD		ns	
RDX	TRC	Read Cycle (ID)	TBD		ns	When Read ID Data
	TRDH	Control Pulse “H” Duration (ID)	TBD		ns	
	TRDL	Control Pulse “L” Duration (ID)	TBD		ns	
DB[17:0]	TDST	Data Setup Time	TBD		ns	TRAT, TRATFM: 3K ohm Pull up or Down
	TDHT	Data Hold Time	TBD		ns	

表 4

6800 时序

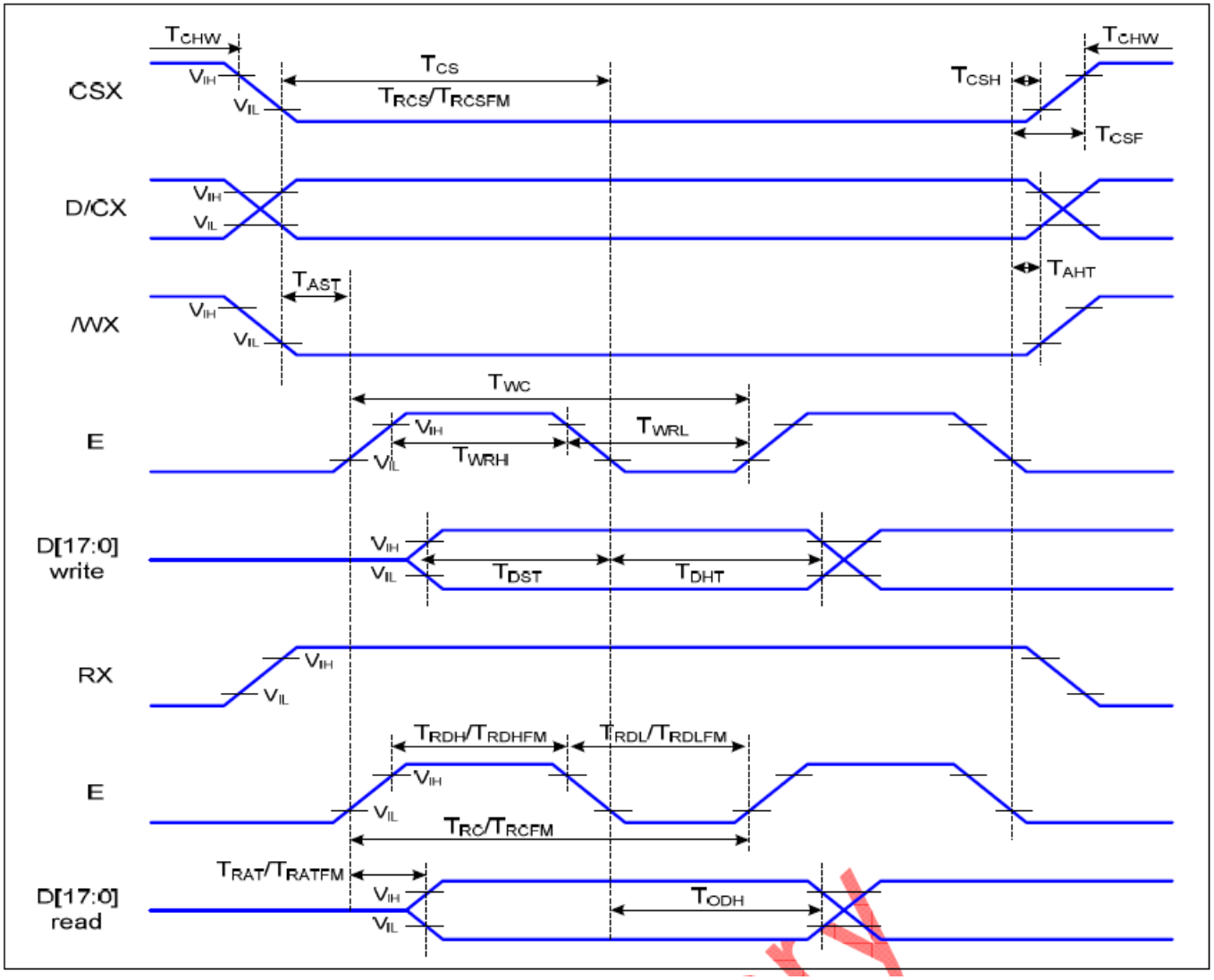


图 3

Signal	Symbol	Parameter	Min	Max	Unit	Description
DCX	T_{AST}	Address setup time	TBD		ns	-
	T_{AHT}	Address hold time (Write/Read)	TBD		ns	
E	T_{WC}	Write cycle	TBD		ns	
	T_{WRH}	Control pulse “H” duration	TBD		ns	
	T_{WRL}	Control pulse “L” duration	TBD		ns	
RDX (ID)	T_{RC}	Read cycle (ID)	TBD		ns	When read ID data
	T_{RDH}	Control pulse “H” duration (ID)	TBD		ns	
	T_{RDL}	Control pulse “L” duration (ID)	TBD		ns	
RDX (FM)	T_{RCFM}	Read cycle (FM)	TBD		ns	When read from frame memory
	T_{RDHFM}	Control pulse “H” duration (FM)	TBD		ns	
	T_{RDLFM}	Control pulse “L” duration (FM)	TBD		ns	
DB[17:0]	T_{DST}	Data setup time	TBD		ns	For maximum CL=30pF
	T_{DHT}	Data hold time	TBD		ns	
	T_{ODH}	Output disable time	TBD	TBD	ns	For minimum CL=8pF

表 5

串行接口

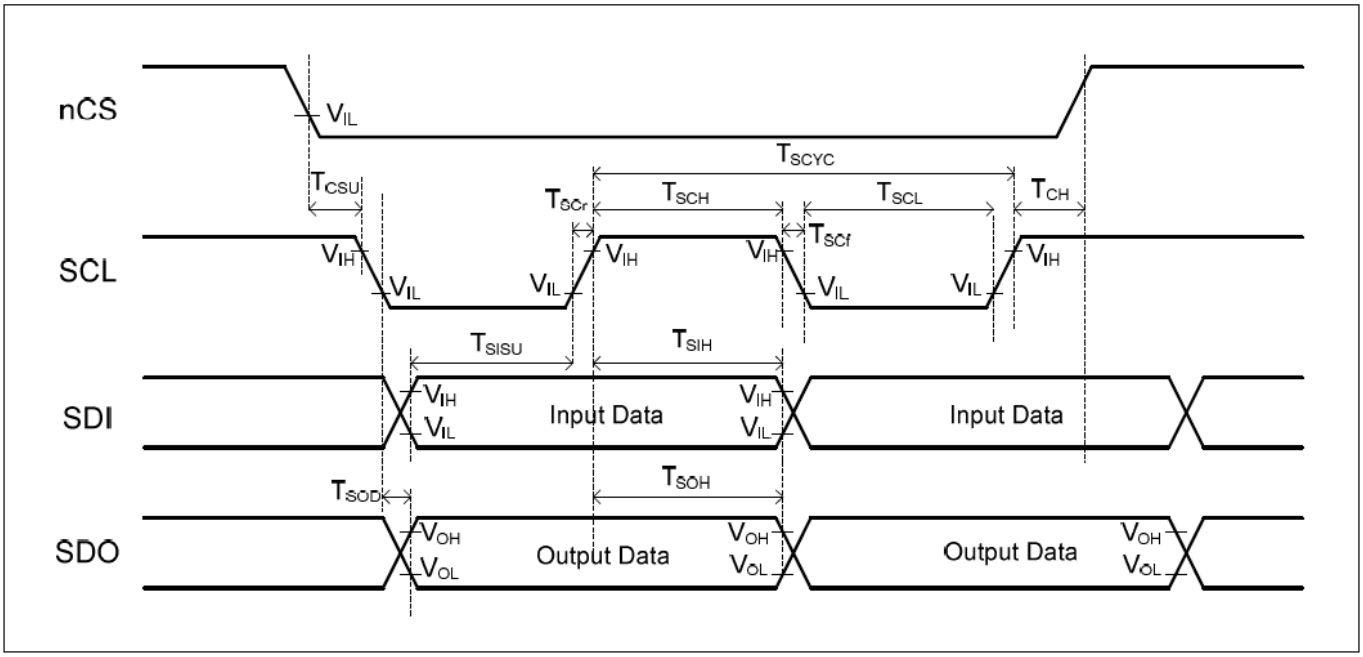


图 4

Signal	Symbol	Parameter	Min	Max	Unit	Description
CSX	TCSU	Chip Select Setup Time	TBD		ns	-
	TCH	Chip Select Hold Time	TBD		ns	
SCL	TSCr ,TSCf	Serial clock rise/fall time	TBD		ns	
	TSCH	SCL “H” pulse width (Write)	TBD		ns	
	TSCH	SCL “H” pulse width (Read)	TBD		ns	
	TSCYC	Serial clock cycle (Write)	TBD		μs	
	TSCYC	Serial clock cycle (Read)	TBD		μs	
	TSCL	SCL “L” pulse width (Write)	TBD		ns	
	TSCL	SCL “L” pulse width (Read)	TBD		ns	
SDI	TSISU	Serial Input Data Setup Time	TBD		ns	
	TSIH	Serial Input Data Hold Time	TBD		ns	
SDO	TSOD	Serial Output Data Setup Time	TBD		ns	
	TSOH	Serial Output Data Hold Time	TBD	TBD	ns	

表 6

6.1 电源启动后复位的时序要求（RESET CONDITION AFTER POWER UP）：

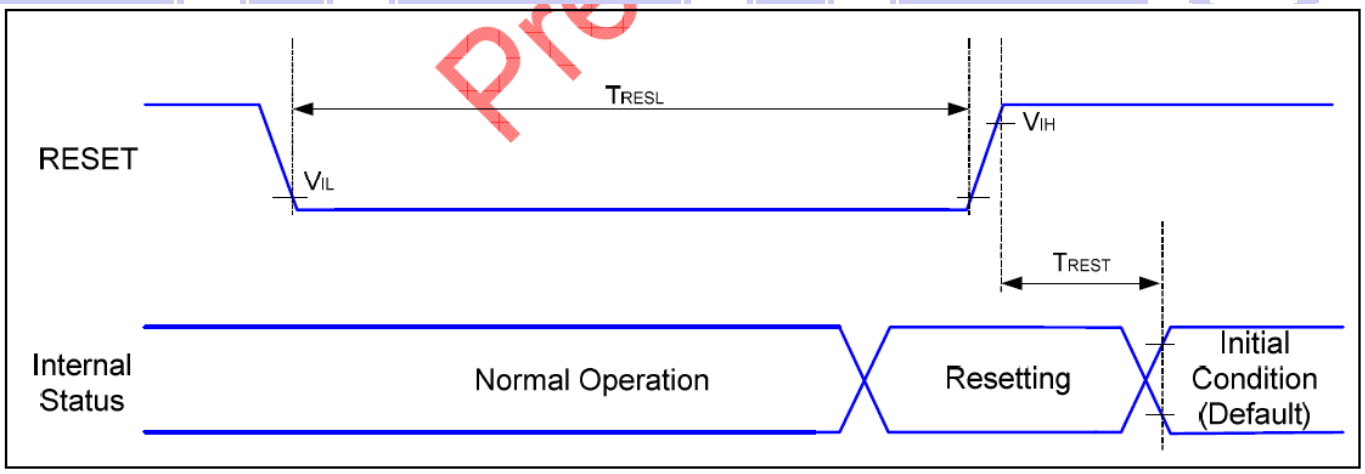


图 5 为电源启动后复位的时序

电源启动后复位的时序要求

Signal	Symbol	Parameter	Min	Max	Unit	Description
RESET	TRESL	Reset Low Level Width	1	-	ms	-
	TREST	Reset Complete Time	1		ms	

表 7

7. 指令功能：

7.1 指令表

指 令 表

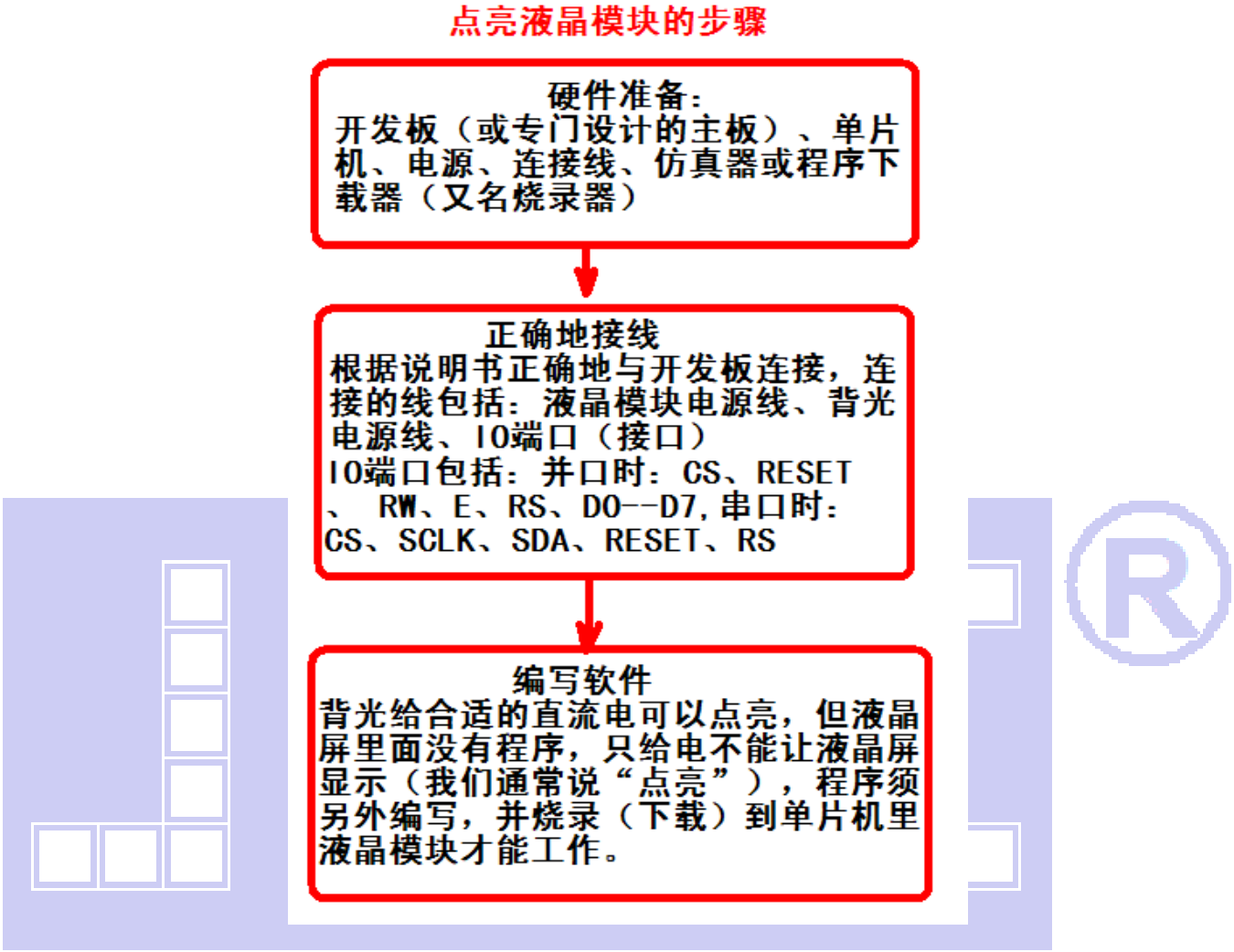
表 8

No	Registers	W/R	RS	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
IR	Index Register	W	0	-	-	-	-	-	-	-	-	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
00h	Driver Code Read	R	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	0	1
01h	Driver Output Control	W/R	1	VSPL	HSPL	DPL	EPL	0	SM	GS	SS	0	0	0	NL4	NL3	NL2	NL1	NL0
02h	LCD Driving Control	W/R	1	0	0	0	0	0	0	0	INV	0	0	0	0	0	0	0	0
03h	Entry Mode	W/R	1	0	0	0	BGR	0	0	MDT1	MDT0	0	0	I/D1	I/D0	AM	0	0	0
07h	Display Control 1	W/R	1	0	0	0	TEMON	0	0	0	0	0	0	0	GON	CL	REV	D1	D0
08h	Display control 2	W/R	0	0	0	0	FP3	FP2	FP1	FP0	0	0	0	0	BP3	BP2	BP1	BP0	0
0Bh	Display Control 4	W/R	1	0	0	0	0	0	0	0	0	0	0	0	0	RTN3	RTN2	RTN1	RTN0
0Ch	RGB Display Interface Control 1	W/R	1	0	0	0	0	0	0	0	RM	0	0	0	DM	0	0	RIM1	RIM0
0Fh	Frame Marker Position	W/R	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	OSC_EN
10h	Power Control 1	W/R	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	SLP	STB
11h	Power Control 2	W/R	1	0	0	0	APON	0	0	0	0	0	0	0	0	0	0	0	0
20h	Horizontal DRAM Address Set	W/R	1	0	0	0	0	0	0	0	0	AD7	AD6	AD5	AD4	AD3	AD2	AD1	AD0
21h	Vertical DRAM Address Set	W/R	1	0	0	0	0	0	0	0	0	AD15	AD14	AD13	AD12	AD11	AD10	AD9	AD8
22h	Write Data to GRAM	W	1	DRAM Write Data (WD17-0) / Read Data (RD17-0)															
22h	Read Data from GRAM	R	1																
28h	Software Reset	W	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30h	Gate Scan Control	W/R	1	0	0	0	0	0	0	0	0	0	0	0	SCN4	SCN3	SCN2	SCN1	SCN0
31h	Vertical Scroll Control 1	W/R	1	0	0	0	0	0	0	0	0	SEA7	SEA6	SEA5	SEA4	SEA3	SEA2	SEA1	SEA0
32h	Vertical Scroll Control 2	W/R	1	0	0	0	0	0	0	0	0	SSA7	SSA6	SSA5	SSA4	SSA3	SSA2	SSA1	SSA0
33h	Vertical Scroll Control 3	W/R	1	0	0	0	0	0	0	0	0	SST7	SST6	SST5	SST4	SST3	SST2	SST1	SST0
34h	Partial Driving Control 1	W/R	1	0	0	0	0	0	0	0	0	SE17	SE16	SE15	SE14	SE13	SE12	SE11	SE10
35h	Partial Driving Control 2	W/R	1	0	0	0	0	0	0	0	0	SS17	SS16	SS15	SS14	SS13	SS12	SS11	SS10
36h	Horizontal Address End Position	W/R	1	0	0	0	0	0	0	0	0	HEA7	HEA6	HEA5	HEA4	HEA3	HEA2	HEA1	HEA0
37h	Horizontal Address Start Position	W/R	1	0	0	0	0	0	0	0	0	HSA7	HSA6	HSA5	HSA4	HSA3	HSA2	HSA1	HSA0
38h	Vertical Address End Position	W/R	1	0	0	0	0	0	0	0	0	VEA7	VEA6	VEA5	VEA4	VEA3	VEA2	VEA1	VEA0

39h	Vertical Address Start Position	W/R	1	0	0	0	0	0	0	0	0	VSA7	VSA6	VSA5	VSA4	VSA3	VSA2	VSA1	VSA0
50h	Gamma Control 1	W/R	1	0	0	0	0	0	KP1[2]	KP1[1]	KP1[0]	0	0	0	0	0	KP0[2]	KP0[1]	KP0[0]
51h	Gamma Control 2	W/R	1	0	0	0	0	KP3[3]	KP3[2]	KP3[1]	KP3[0]	0	0	0	0	KP2[3]	KP2[2]	KP2[1]	KP2[0]
52h	Gamma Control 3	W/R	1	0	0	0	0	0	KP5[2]	KP5[1]	KP5[0]	0	0	0	0	KP4[3]	KP4[2]	KP4[1]	KP4[0]
53h	Gamma Control 4	W/R	1	0	0	SELV63 P[2]	SELV63 P[1]	SELV63 P[0]	SELV62 P[2]	SELV62 P[1]	SELV62 P[0]	SELV1 P[3]	SELV1 P[2]	SELV1 P[1]	SELV1 P[0]	SELV0 P[3]	SELV0 P[2]	SELV0 P[1]	SELV0 P[0]
54h	Gamma Control 5	W/R	1	0	0	0	0	VOS0 P[3]	VOS0 P[2]	VOS0 P[1]	VOS0 P[0]	0	0	0	0	VRF0 P[3]	VRF0 P[2]	VRF0 P[1]	VRF0 P[0]
55h	Gamma Control 6	W/R	1	0	0	0	0	0	KN1[2]	KN1[1]	KN1[0]	0	0	0	0	0	KN0[2]	KN0[1]	KN0[0]
56h	Gamma Control 7	W/R	1	0	0	0	0	KN3[3]	KN3[2]	KN3[1]	KN3[0]	0	0	0	0	KN2[3]	KN2[2]	KN2[1]	KN2[0]
57h	Gamma Control 8	W/R	1	0	0	0	0	0	KN5[2]	KN5[1]	KN5[0]	0	0	0	0	KN4[3]	PKN4[2]	KN4[1]	KN4[0]
58h	Gamma Control 9	W/R	1	0	0	SELV63 N[2]	SELV63 N[1]	SELV63 N[0]	SELV62 N[2]	SELV62 N[1]	SELV62 N[0]	SELV1 N[3]	SELV1 N[2]	SELV1 N[1]	SELV1 N[0]	SELV0 N[3]	SELV0 N[2]	SELV0 N[1]	SELV0 N[0]
59h	Gamma Control 10	W/R	1	0	0	0	0	VOS0 N[3]	VOS0 N[2]	VOS0 N[1]	VOS0 N[0]	0	0	0	0	VRF0 N[3]	VRF0 N[2]	VRF0 N[1]	VRF0 N[0]
65h	ID code	R		0	0	0	0	0	0	0	0	0	0	0	0	ID3	ID2	ID1	ID0
66h	SPI Read/Write Control	W/R	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	R/WX
B0h	Power Control 3	W/R	1	0	0	VCM5	VCM4	VCM3	VCM2	VCM1	VCM0	0	0	VGLSEL 1	VGLSEL 0	0	0	VGHBT1	VGHBT0
B1h	Power Control 4	W/R	1	0	0	0	VRHN4	VRHN3	VRHN2	VRHN1	VRHN0	0	0	0	VRHP4	VRHP3	VRHP2	VRHP1	VRHP0
B2h	Power Control 5	W/R	1	0	0	0	0	0	AVCLS2	AVCLS1	AVCLS0	0	0	BCLK_DIV1	BCLK_DIV0	0	AVDDS2	AVDDS1	AVDDS0
D2h	NVM ID Code	W/R	1	0	0	0	0	0	0	0	0	0	0	0	0	ID3	ID2	ID1	ID0
D9h	NVM Control Status	W/R	1	0	0	0	0	0	0	0	0	0	VMF_EN	0	0	0	0	0	0
DFh	NVM Write Command	W	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	1
FAh	NVM Enable	W/R	1	0	0	0	0	0	0	0	0	PROG _MODE	0	0	0	0	1	MTP _PROG	0
FEh	NVM VCOM Offset	W/R	1	0	0	0	0	0	0	0	0	1	0	0	VMF4	VMF3	VMF2	VMF1	VMF0
FFh	NVM Command Enable	W	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CMD1_ EN	CMD2_ EN

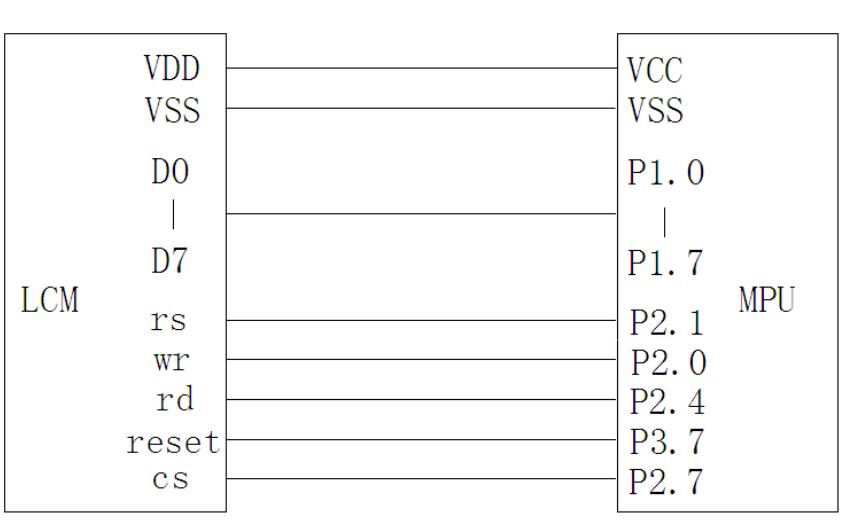
7.2 初始化方法

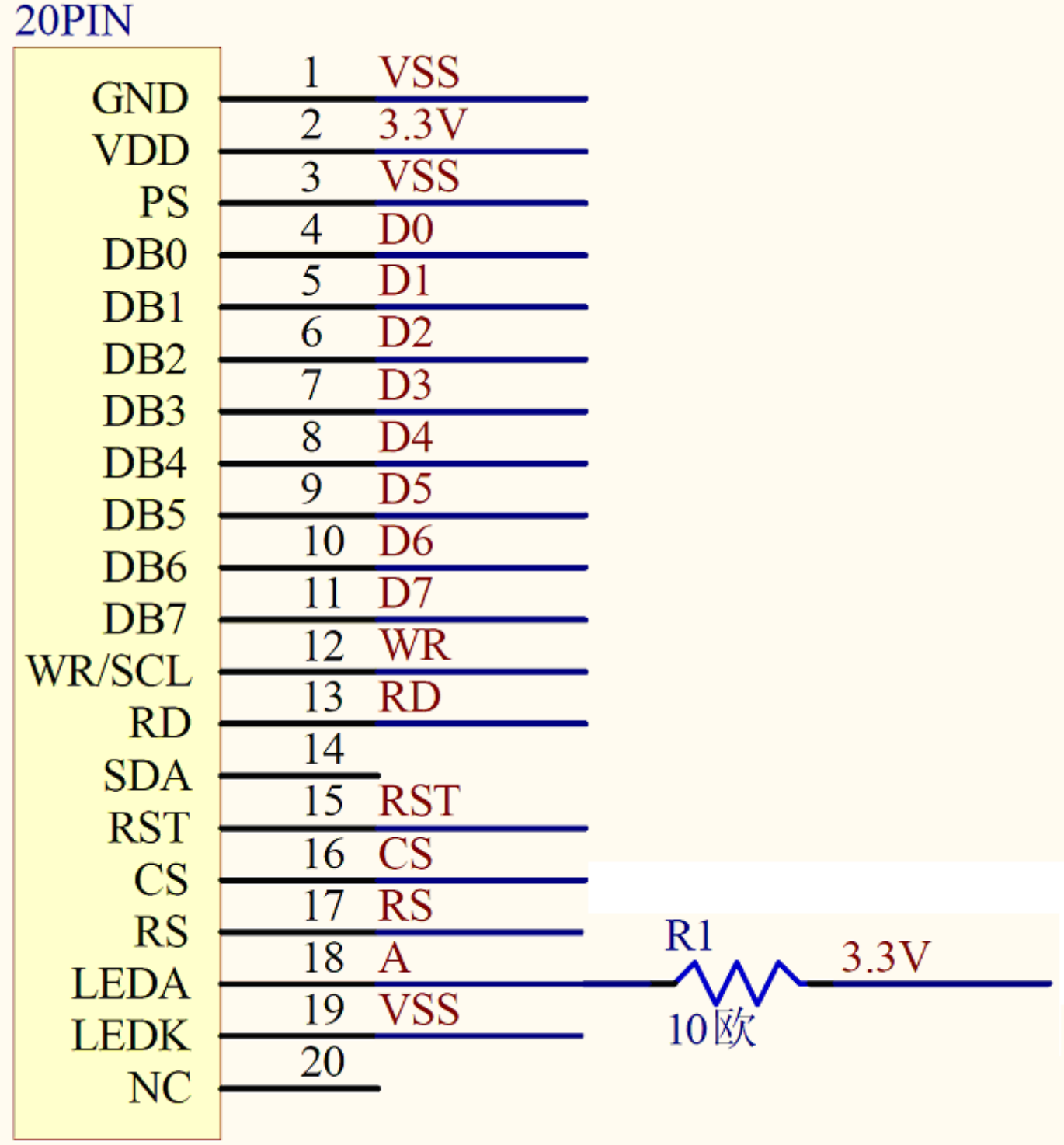
用户所编的显示程序，开始必须进行初始化，否则模块无法正常显示，过程请参考程序



7.3 程序举例：

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:并行接口





详细例程请找销售索要

```
#include <reg51.h>
#include <chinese_code.h>

sbit RS = P2^1;    //液晶屏接口定义
sbit WR = P2^0;    //液晶屏接口定义
sbit RD = P2^4;    //液晶屏接口定义
sbit CS = P2^7;    //液晶屏接口定义
sbit RESET = P3^7; //液晶屏接口定义
```

```
sbit key =P2^0;     //按键：低电平有效

#define uint unsigned int

void delay(long i)
{
     int j,k;
     for(j=0;j<i;j++)
     for(k=0;k<110;k++);
}

//void delay_us(long i)
//{
//   int j,k;
//   for(j=0;j<i;j++);
//   for(k=0;k<1;k++);
//}

void Switch()
{
 repeat:
     if (key==1) goto repeat;
     else delay(1000);
     if (key) goto repeat;
     else ;
}

void transfer_command(int com1)
{
     CS = 0;
     RS = 0;
     RD = 1;
     P1=com1;
     WR = 0;
     WR = 1;
     CS = 1;
}

void transfer_data(int data1)
{
    CS = 0;
     RS = 1;
     RD = 1;
     P1=data1;
     WR = 0;
     WR = 1;
```

```
    CS = 1;
}


//===传 16 位指令，16 位指令一起赋值
void transfer_command_16(uint com_16bit)
{
    transfer_command(com_16bit >>8);     //先传高 8 位
    transfer_command(com_16bit );        //再传低 8 位
}

//连写 2 个字节（即 16 位）数据到 LCD 模块
void transfer_data_16(uint data_16bit)
{
    transfer_data(data_16bit>>8);
    transfer_data(data_16bit);
}

void lcd_initial()
{
    RESET=1;
    delay(200);
    RESET=0;
    delay(200);
    RESET=1;
    delay(200);
//-------Display Control Setting------------//

    transfer_command_16(0x0001);
    transfer_data_16(0x011c);
    transfer_command_16(0x0002);
    transfer_data_16(0x0100);
    transfer_command_16(0x0003);     //Entry Mode
    transfer_data_16(0x1030);
    transfer_command_16(0x0008);
    transfer_data_16(0x0808);
    transfer_command_16(0x000c);
    transfer_data_16(0x0000);
    transfer_command_16(0x000f);
    transfer_data_16(0x0001);
    transfer_command_16(0x0020);
    transfer_data_16(0x0000);
    transfer_command_16(0x0021);
    transfer_data_16(0x0000);

//-------Power Control Registers Initial------------//
```

```
    transfer_command_16(0x0010);
    transfer_data_16(0x0000);
    transfer_command_16(0x0011);
    transfer_data_16(0x1000);
    delay(100);
//-------Display Windows 176x220------------//
    transfer_command_16(0x0030);
    transfer_data_16(0x0000);
    transfer_command_16(0x0031);
    transfer_data_16(0x00db);
    transfer_command_16(0x0032);
    transfer_data_16(0x0000);
    transfer_command_16(0x0033);
    transfer_data_16(0x0000);
    transfer_command_16(0x0034);
    transfer_data_16(0x00db);
    transfer_command_16(0x0035);
    transfer_data_16(0x0000);
    transfer_command_16(0x0036);
    transfer_data_16(0x00af);
    transfer_command_16(0x0037);
    transfer_data_16(0x0000);
    transfer_command_16(0x0038);
    transfer_data_16(0x00db);
    transfer_command_16(0x0039);
    transfer_data_16(0x0000);
    delay(10);
//-------End Display Windows 176x220-----------//
    transfer_command_16(0x00ff);
    transfer_data_16(0x0003);
//-------Gamma Cluster Setting------------//
    transfer_command_16(0x0050);
    transfer_data_16(0x0103);
    transfer_command_16(0x0051);
    transfer_data_16(0x0808);
    transfer_command_16(0x0052);
    transfer_data_16(0x0207);
    transfer_command_16(0x0053);
    transfer_data_16(0x2222);
    transfer_command_16(0x0054);
    transfer_data_16(0x0703);
    transfer_command_16(0x0055);
    transfer_data_16(0x0103);
    transfer_command_16(0x0056);
    transfer_data_16(0x0808);
    transfer_command_16(0x0057);
```

```
    transfer_data_16(0x0207);
    transfer_command_16(0x0058);
    transfer_data_16(0x2222);
    transfer_command_16(0x0059);
    transfer_data_16(0x0603);

//-------Vcom Setting------------//
    transfer_command_16(0x00b0);
    transfer_data_16(0x1201);
//      transfer_command_16(0x00b1);
//  transfer_data_16(0x0202);
        transfer_command_16(0x00b2);
    transfer_data_16(0x0707);
//-------End Vcom Setting------------//
    transfer_command_16(0x00ff);
    transfer_data_16(0x0000);

    transfer_command_16(0x0007);   //display on
    transfer_data_16(0x1017);
    transfer_command_16(0x0022);  //Write Data to DRAM
}


void mono_transfer_data_16(int mono_data,int font_color,int back_color)
{
    int i;
    for(i=0;i<8;i++)
    {
        if(mono_data&0x80)
        {
            transfer_data_16(font_color);    //当数据是 1 时，显示字体颜色
        }
        else
        {
            transfer_data_16(back_color);    //当数据是 0 时，显示底色
        }
        mono_data<<=1;
    }
}

//===设置单个像素的（x，y）座标
void lcd_address(int XS,int YS)
{
    transfer_command_16(0x0020);          // 设置 X 结束的地址
    transfer_data_16(XS);                 // X 开始地址(16 位)
    transfer_command_16(0x0021);          // 设置 Y 结束的地址
```

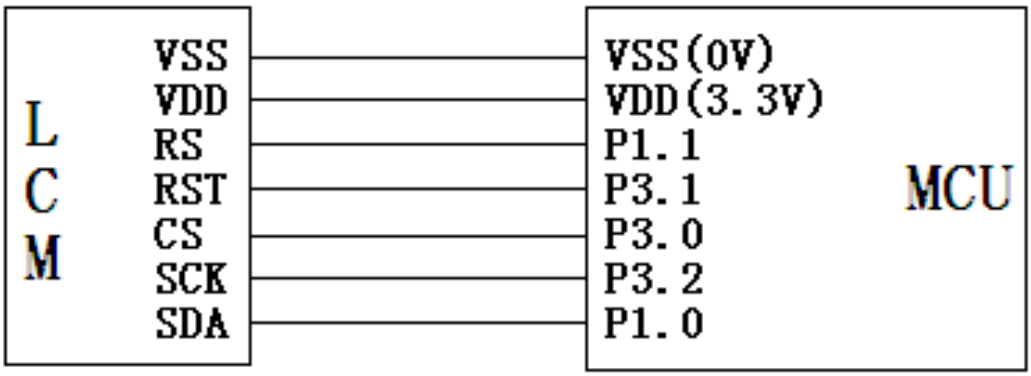
```
    transfer_data_16(YS);                // Y 开始地址(16 位)

    transfer_command_16(0x0022);         // 写数据开始
}


////定义窗口坐标：开始坐标（XS,YS)以及窗口大小（x_total,y_total)

//void set_window(int XS,int YS,int x_total,int y_total)
//{
//   int XE,YE;
//   XE=XS+x_total-1;
//   YE=YS+y_total-1;
//   transfer_command_16(0x0036);          // 设置 X 结束的地址
//   transfer_data_16(XE);                 // X 结束地址(16 位)
//   transfer_command_16(0x0037);          // 设置 X 开始的地址
//   transfer_data_16(XS);                 // X 开始地址(16 位)
//   transfer_command_16(0x0038);          // 设置 Y 结束的地址
//   transfer_data_16(YE);                 // Y 结束地址(16 位)
//   transfer_command_16(0x0039);          // 设置 Y 开始的地址
//   transfer_data_16(YS);                 // Y 开始地址(16 位)

//   transfer_command_16(0x0022);         // 写数据开始
//}


void display_string_16x16(int x,int y,uchar *text,int font_color,int back_color)
{
    uchar i,j,k;
    uint address;
    j = 0;
    while(text[j] != '\0')                                  //'\0' 字符串结束标志
    {
        i = 0;
        address = 1;
        while(Chinese_horizontal_text_16x16[i] > 0x7e)  // >0x7f 即说明不是 ASCII 码字符
        {
            if(Chinese_horizontal_text_16x16[i] == text[j])
            {
                if(Chinese_horizontal_text_16x16[i + 1] == text[j + 1])
                {
                    address = i * 16;
                    break;
                }
            }
            i += 2;
```

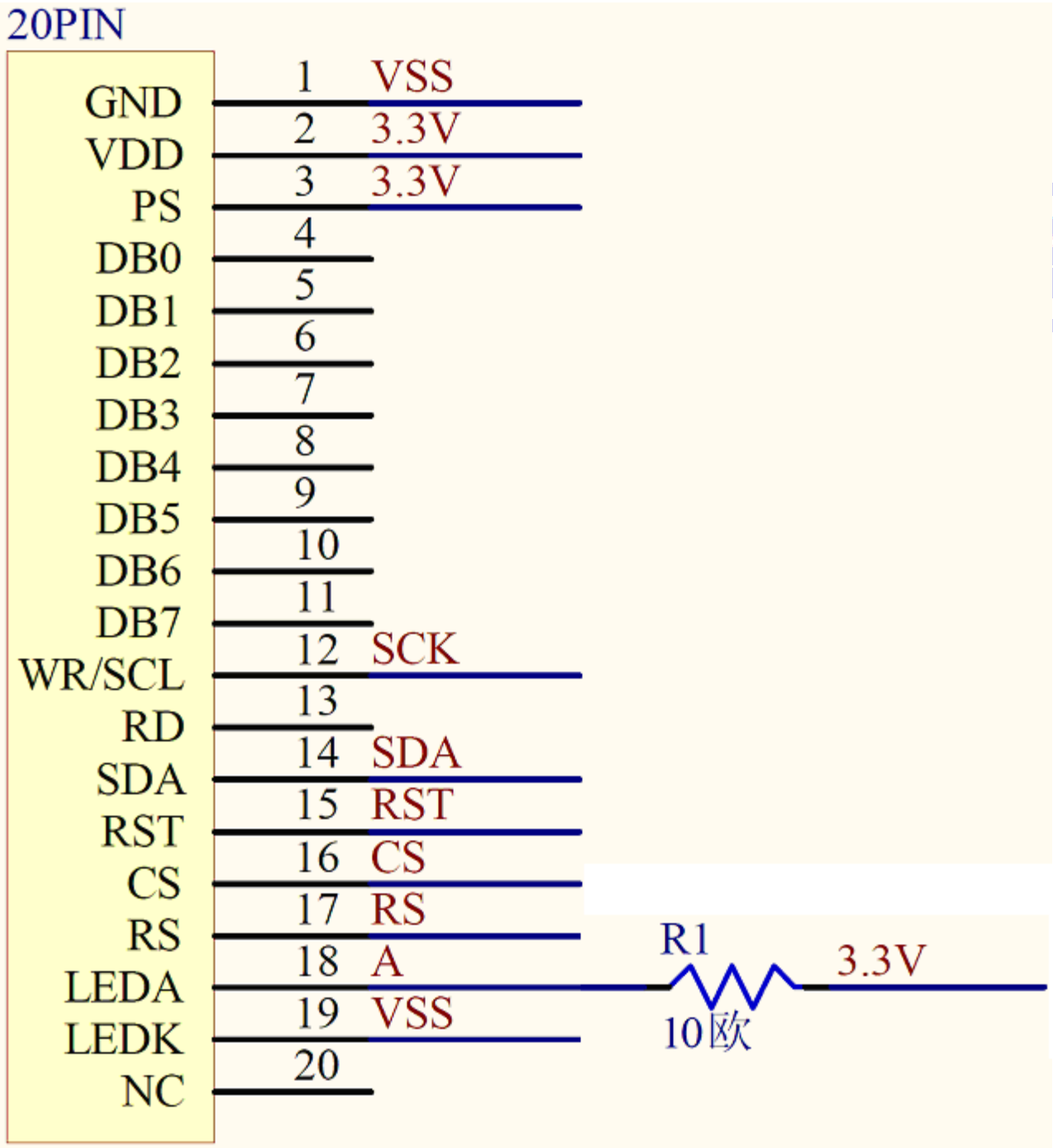
```
        }

        if(address != 1)// 显示汉字
        {
            for(k = 0; k <16; k++)
            {
                lcd_address(x,y+k);
                for(i=0;i<2;i++)
                {
                    mono_transfer_data_16(Chinese_horizontal_code_16x16[address],font_color,back_color);
                    address++;
                }
            }
            j+=2;
        }
        else              //显示空白字符
        {
            for(k = 0; k < 16; k++)
            {
                lcd_address(x,y+k);
                for(i = 0; i <2; i++)
                {
                    mono_transfer_data_16(0x00,font_color,back_color);
                }
            }
            j+=2;
        }
        x=x+16;
    }
}

//显示 32x32 点阵的单色的图像
void disp_32x32(int x,int y,char *dp,int font_color,int back_color)
{
    int i,j;
    for(i=0;i<32;i++)
    {
        lcd_address(x,y+i);
        for(j=0;j<4;j++)
        {
            mono_transfer_data_16(*dp,font_color,back_color);
            dp++;
        }
    }
}
```

```
//显示一幅 176*110 点阵彩图
void display_image(int x,int y,uchar *dp)
{
    uchar i,j;
    lcd_address(x,y);
    for(i=0;i<176;i++)
    {
        for(j=0;j<110;j++)
        {
            transfer_data(*dp);              //传一个像素的图片数据的高位
            dp++;
            transfer_data(*dp);              //传一个像素的图片数据的低位
            dp++;
        }
    }
}

//显示全屏单一色彩
void display_color(int color)
{
    int i,j;
    lcd_address(0,0);
    for(i=0;i<176;i++)
    {

        for(j=0;j<220;j++)
        {
            transfer_data_16(color);
        }
    }
}

//显示 N 个 8x16 点阵的单色的图像
void disp_8x16(int x,int y,uchar len,char *dp,int font_color,int back_color)
{
    int i,j,k;
   for(k=0;k<len;k++)
  {
    for(i=0;i<16;i++)
    {
        lcd_address(x+k*8,y+i);
        for(j=0;j<1;j++)
        {
            mono_transfer_data_16(*dp,font_color,back_color);
            dp++;
        }
```

```
    }
  }
}

void main(void)
{
    lcd_initial();
    while(1)
    {
       display_color(0x001f);
       display_image(0,0,pic1);     //显示单幅彩图，编码是通过专用取模工具获取的，取模工具请找客服人员索取。
       display_image(0,110,pic1);
       Switch();
       display_color(0x001f);
        disp_32x32(24+32*1,0,shen_32x32,white,blue);
        disp_32x32(24+32*2,0,zhen_32x32,white,blue);
       disp_32x32(24+32*1,32,shi_32x32,white,blue);
       disp_32x32(24+32*2,32,jing_32x32,white,blue);
       disp_32x32(24+32*1,64,lian_32x32,white,blue);
       disp_32x32(24+32*2,64,xun_32x32,white,blue);
       disp_32x32(24+32*1,96,dian_32x32,white,blue);
       disp_32x32(24+32*2,96,zi_32x32,white,blue);
        display_string_16x16(32,128,"深圳市晶联讯",white,blue);
       display_string_16x16(32,144,"电子有限公司",white,blue);
       disp_8x16(34,160,13,JLX200_01701_BN,white,blue);
       display_string_16x16(0,176,"接口方式：",white,blue);
       display_string_16x16(0,192,"支持串口和并口两种方式",white,blue);
        Switch();
        display_color(0xf800);
        Switch();
        display_color(0x07e0);
        Switch();
        display_color(0x001f);
        Switch();
        display_color(0x0000);
        Switch();
    }
}
```

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:串行接口


LCM
VSS
VDD
RS
RST
CS
SCK
SDA
VSS(0V)
VDD(3.3V)
P1.1
P3.1
P3.0
P3.2
P1.0
MCU


20PIN
GND 1 VSS
VDD 2 3.3V
PS 3 3.3V
DB0 4
DB1 5
DB2 6
DB3 7
DB4 8
DB5 9
DB6 10
DB7 11
WR/SCL 12 SCK
RD 13
SDA 14 SDA
RST 15 RST
CS 16 CS
RS 17 RS
LEDA 18 A
LEDK 19 VSS
NC 20
R1
10欧
3.3V

```
#include <reg51.h>
#include <chinese_code.h>

//液晶屏 IC 所需要的信号线的接口定义
sbit CS=P3^0;
sbit RST=P3^1;
sbit RS=P1^1;
sbit SCK=P3^2;
sbit SDA=P1^0;
sbit key=P2^0;          //P2.0 口与 GND 之间接一个按键

#define uint unsigned int

void delay(long i)
{
    int j,k;
    for(j=0;j<i;j++)
    for(k=0;k<110;k++);
}

//void delay_us(long i)
//{
//   int j,k;
//   for(j=0;j<i;j++);
//   for(k=0;k<1;k++);
//}

void Switch()
{
 repeat:
    if (key==1) goto repeat;
    else delay(1000);
    if (key) goto repeat;
    else ;
}

/*写指令到 LCD 模块*/
void transfer_command_16(int data1)
{
    char i;
    CS=0;
    RS=0;
    for(i=0;i<8;i++)
    {
        SCK=0;
        if(data1&0x80) SDA=1;
```

```
        else SDA=0;
        SCK=1;
        data1=data1<<=1;
    }
}

/*写数据到 LCD 模块*/
void transfer_data(int data1)
{
    char i;
    CS=0;
    RS=1;
    for(i=0;i<8;i++)
    {
        SCK=0;
        if(data1&0x80) SDA=1;
        else SDA=0;
        SCK=1;
        data1=data1<<=1;
    }
}

//连写 2 个字节（即 16 位）数据到 LCD 模块
void transfer_data_16(uint data2)
{
    transfer_data(data2>>8);
    transfer_data(data2);
}

void lcd_initial()
{
    RST=1;
    delay(200);
    RST=0;
    delay(200);
    RST=1;
    delay(200);
//-------Display Control Setting------------//

    transfer_command_16(0x0001);
    transfer_data_16(0x011c);
    transfer_command_16(0x0002);
    transfer_data_16(0x0100);
    transfer_command_16(0x0003);    //Entry Mode
    transfer_data_16(0x1030);
    transfer_command_16(0x0008);
```

```
    transfer_data_16(0x0808);
    transfer_command_16(0x000c);
    transfer_data_16(0x0000);
    transfer_command_16(0x000f);
    transfer_data_16(0x0001);
    transfer_command_16(0x0020);
    transfer_data_16(0x0000);
    transfer_command_16(0x0021);
    transfer_data_16(0x0000);

//-------Power Control Registers Initial------------//
    transfer_command_16(0x0010);
    transfer_data_16(0x0000);
    transfer_command_16(0x0011);
    transfer_data_16(0x1000);
    delay(100);
//-------Display Windows 176x220------------//
    transfer_command_16(0x0030);
    transfer_data_16(0x0000);
    transfer_command_16(0x0031);
    transfer_data_16(0x00db);
    transfer_command_16(0x0032);
    transfer_data_16(0x0000);
    transfer_command_16(0x0033);
    transfer_data_16(0x0000);
    transfer_command_16(0x0034);
    transfer_data_16(0x00db);
    transfer_command_16(0x0035);
    transfer_data_16(0x0000);
    transfer_command_16(0x0036);
    transfer_data_16(0x00af);
    transfer_command_16(0x0037);
    transfer_data_16(0x0000);
    transfer_command_16(0x0038);
    transfer_data_16(0x00db);
    transfer_command_16(0x0039);
    transfer_data_16(0x0000);
    delay(10);
//-------End Display Windows 176x220------------//
    transfer_command_16(0x00ff);
    transfer_data_16(0x0003);
//-------Gamma Cluster Setting------------//
    transfer_command_16(0x0050);
    transfer_data_16(0x0103);
    transfer_command_16(0x0051);
    transfer_data_16(0x0808);
```

```
    transfer_command_16(0x0052);
    transfer_data_16(0x0207);
    transfer_command_16(0x0053);
    transfer_data_16(0x2222);
    transfer_command_16(0x0054);
    transfer_data_16(0x0703);
    transfer_command_16(0x0055);
    transfer_data_16(0x0103);
    transfer_command_16(0x0056);
    transfer_data_16(0x0808);
    transfer_command_16(0x0057);
    transfer_data_16(0x0207);
    transfer_command_16(0x0058);
    transfer_data_16(0x2222);
    transfer_command_16(0x0059);
    transfer_data_16(0x0603);

//-------Vcom Setting------------//
    transfer_command_16(0x00b0);
    transfer_data_16(0x1201);
//      transfer_command_16(0x00b1);
//  transfer_data_16(0x0202);
        transfer_command_16(0x00b2);
    transfer_data_16(0x0707);
//-------End Vcom Setting------------//
    transfer_command_16(0x00ff);
    transfer_data_16(0x0000);

    transfer_command_16(0x0007);   //display on
    transfer_data_16(0x1017);
    transfer_command_16(0x0022);  //Write Data to DRAM
}


void mono_transfer_data_16(int mono_data,int font_color,int back_color)
{
    int i;
    for(i=0;i<8;i++)
    {
        if(mono_data&0x80)
        {
            transfer_data_16(font_color);    //当数据是 1 时，显示字体颜色
        }
        else
        {
            transfer_data_16(back_color);    //当数据是 0 时，显示底色
```

```
        }
        mono_data<<=1;
    }
}

//===设置单个像素的（x，y）座标
void lcd_address(int XS,int YS)
{
    transfer_command_16(0x0020);          // 设置 X 结束的地址
    transfer_data_16(XS);                 // X 开始地址(16 位)
    transfer_command_16(0x0021);          // 设置 Y 结束的地址
    transfer_data_16(YS);                 // Y 开始地址(16 位)

    transfer_command_16(0x0022);         // 写数据开始
}


////定义窗口坐标：开始坐标（XS,YS)以及窗口大小（x_total,y_total)

//void set_window(int XS,int YS,int x_total,int y_total)
//{
//  int XE,YE;
//  XE=XS+x_total-1;
//  YE=YS+y_total-1;
//  transfer_command_16(0x0036);          // 设置 X 结束的地址
//  transfer_data_16(XE);                 // X 结束地址(16 位)
//  transfer_command_16(0x0037);          // 设置 X 开始的地址
//  transfer_data_16(XS);                 // X 开始地址(16 位)
//  transfer_command_16(0x0038);          // 设置 Y 结束的地址
//  transfer_data_16(YE);                 // Y 结束地址(16 位)
//  transfer_command_16(0x0039);          // 设置 Y 开始的地址
//  transfer_data_16(YS);                 // Y 开始地址(16 位)

//  transfer_command_16(0x0022);         // 写数据开始
//}


void display_string_16x16(int x,int y,uchar *text,int font_color,int back_color)
{
    uchar i,j,k;
    uint address;
    j = 0;
    while(text[j] != '\0')                              //'\0' 字符串结束标志
    {
        i = 0;
        address = 1;
```

```
        while(Chinese_horizontal_text_16x16[i] > 0x7e)  // >0x7f 即说明不是 ASCII 码字符
        {
            if(Chinese_horizontal_text_16x16[i] == text[j])
            {
                if(Chinese_horizontal_text_16x16[i + 1] == text[j + 1])
                {
                    address = i * 16;
                    break;
                }
            }
            i += 2;
        }

        if(address != 1)// 显示汉字
        {
            for(k = 0; k <16; k++)
            {
                lcd_address(x,y+k);
                for(i=0;i<2;i++)
                {
                    mono_transfer_data_16(Chinese_horizontal_code_16x16[address],font_color,back_color);
                    address++;
                }
            }
            j+=2;
        }
        else          //显示空白字符
        {
            for(k = 0; k < 16; k++)
            {
                lcd_address(x,y+k);
                for(i = 0; i <2; i++)
                {
                    mono_transfer_data_16(0x00,font_color,back_color);
                }
            }
            j+=2;
        }
        x=x+16;
    }
}

//显示 32x32 点阵的单色的图像
void disp_32x32(int x,int y,char *dp,int font_color,int back_color)
{
    int i,j;
```

```
    for(i=0;i<32;i++)
    {
        lcd_address(x,y+i);
        for(j=0;j<4;j++)
        {
            mono_transfer_data_16(*dp,font_color,back_color);
            dp++;
        }
    }
}

//显示全屏单一色彩
void display_color(int color)
{
    int i,j;
    lcd_address(0,0);
    for(i=0;i<176;i++)
    {

        for(j=0;j<220;j++)
        {
            transfer_data_16(color);
        }
    }
}

//显示一幅 176*110 点阵彩图
void display_image(int x,int y,uchar *dp)
{
    uchar i,j;
    lcd_address(x,y);
    for(i=0;i<176;i++)
    {
        for(j=0;j<110;j++)
        {
            transfer_data(*dp);              //传一个像素的图片数据的高位
            dp++;
            transfer_data(*dp);              //传一个像素的图片数据的低位
            dp++;
        }
    }
}

//显示 N 个 8x16 点阵的单色的图像
void disp_8x16(int x,int y,uchar len,char *dp,int font_color,int back_color)
{
```

```
    int i,j,k;
   for(k=0;k<len;k++)
  {
    for(i=0;i<16;i++)
    {
         lcd_address(x+k*8,y+i);
         for(j=0;j<1;j++)
         {
              mono_transfer_data_16(*dp,font_color,back_color);
              dp++;
         }
    }
  }
}

void main(void)
{
    lcd_initial();
    while(1)
    {
         display_color(0x001f);
        display_image(0,0,pic1);     //显示单幅彩图，编码是通过专用取模工具获取的，取模工具请找客服人员索取。
        display_image(0,110,pic1);
        Switch();
        display_color(0x001f);
         disp_32x32(24+32*0,0,shen_32x32,white,blue);
         disp_32x32(24+32*1,0,zhen_32x32,white,blue);
        disp_32x32(24+32*2,0,shi_32x32,white,blue);
        disp_32x32(24+32*3,0,jing_32x32,white,blue);
        disp_32x32(24+32*0,32,lian_32x32,white,blue);
        disp_32x32(24+32*1,32,xun_32x32,white,blue);
        disp_32x32(24+32*2,32,dian_32x32,white,blue);
        disp_32x32(24+32*3,32,zi_32x32,white,blue);
         display_string_16x16(0,65,"深圳晶联讯电子有限公司",white,blue);
        disp_8x16(34,82,13,JLX200_01701_BN,white,blue);
        display_string_16x16(0,99,"支持串口和并口两种方式",white,blue);
         Switch();
         display_color(0xf800);
         Switch();
         display_color(0x07e0);
         Switch();
         display_color(0x001f);
         Switch();
         display_color(0x0000);
         Switch();
    }
```

}