

JLX12864OLED-096X 中文使用说明书

目 录

序号	内 容 标 题	页码
1	概述	2
2	特点	2
3	外形及接口引脚功能	3~5
4	基本原理	6
5	技术参数	6
6	时序特性	6~9
7	指令功能及硬件接口与编程案例	9~页末

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 JLX12864OLED-096X 型液晶模块由于使用方便、无需背光、视角宽、显示清晰、超薄，广泛应用于各种人机交流面板。

JLX12864OLED-096X 可以显示 128 列*64 行点阵单色图片,或显示 16*16 点阵的汉字 8 个*4 行,或显示 8*16 点阵的英文、数字、符号 16 个*4 行。或显示 5*8 点阵的英文、数字、符号 21 个*8 行。

2. JLX12864OLED-096X 图像型点阵液晶模块的特性

2.1 结构牢：焊接式 FPC。

2.2 IC 采用 SSD1315, 功能强大，稳定性好

2.3 功耗低。

2.4 显示内容：

- 128*64 点阵单色图片；

- 可選用 16*16 点阵或其他点阵的图片来自编汉字，按照 16*16 点阵汉字来计算可显示 8 字/行*4 行。按照 12*12 点阵汉字来计算可显示 10 字/行*4 行。

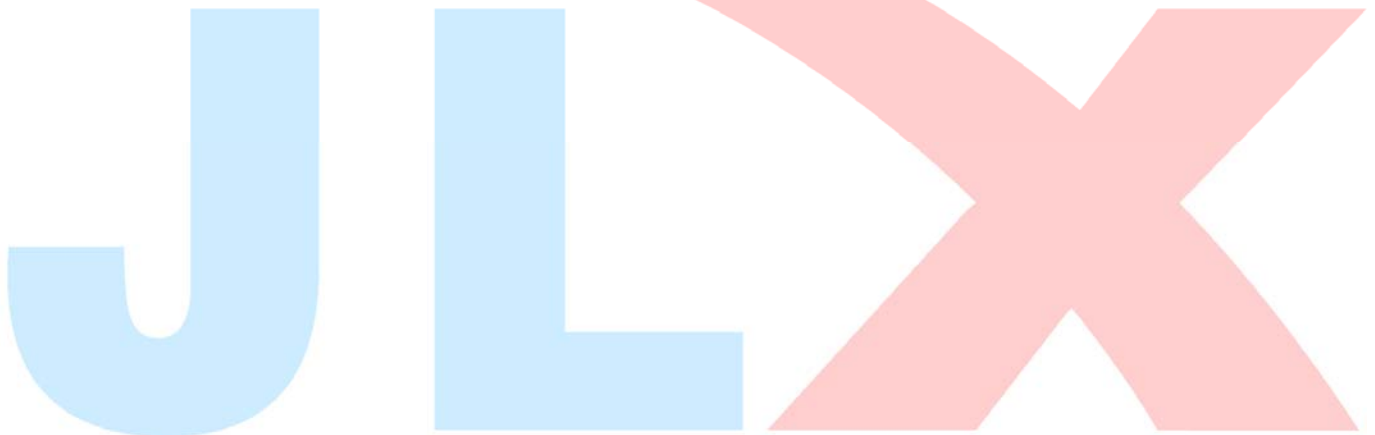
2.5 指令功能强:可组合成各种输入、显示、移位方式以满足不同的要求；

2.6 接口方式：4 线 SPI 串行接口、I²C 接口。

2.7 工作温度宽:-20℃ - 70℃；

2.8 储存温度宽：-30℃-80℃；

2.9 底色可选：蓝色、白色。



模块的接口引脚功能

3.2.1 串行时接口引脚功能

引线号	符号	名称	功能
1	NC (GND)	NC	空脚或接地
2	C2P	升压电容	
3	C2N	升压电容	
4	C1P	升压电容	
5	C1N	升压电容	
6	VBAT	VBAT	接 VDD
7	NC	NC	空脚
8	VSS	接地	0V
9	VDD	电源电路	3.3V
10	BS0	BS0	接低电平或 VSS
11	BS1	BS1	接低电平或 VSS
12	NC	NC	悬空
13	CS#	片选	低电平片选
14	RES#	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
15	D/C#	寄存选择信号	H: 数据存储器 0: 指令存储 (IC 资料上缩写为 "A0")
16	NC	NC	悬空
17	NC	NC	悬空
18	D0 (SCL)	I/O	串行时钟 (SCL)
19-20	D1-D2 (SDA)	I/O	串行数据 (SDA), D1、D2 连接一起做 SDA
21	NC	NC	悬空
22	NC	NC	悬空
23	NC	NC	悬空
24	NC	NC	悬空
25	NC	NC	悬空
26	IREF	IREF	悬空
27	VCOMH	VCOMH	
28	VCC	面板电源	
29	VLSS	VLSS	接地
30	NC (GND)	NC	空脚或接地

3.2.2 I²C 接口引脚功能

引线号	符号	名称	功能
1	NC (GND)	NC	空脚或接地
2	C2P	升压电容	
3	C2N	升压电容	
4	C1P	升压电容	
5	C1N	升压电容	
6	VBAT	VBAT	接 VDD
7	NC	NC	空脚
8	VSS	接地	0V
9	VDD	电源电路	3.3V
10	BS0	BS0	接低电平 (VSS)
11	BS1	BS1	接高电平 (VDD)
12	NC	NC	悬空
13	CS#	片选	接 VSS
14	RES#	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
15	D/C#	从属地址选择 SA0	接 VSS
16	NC	NC	悬空
17	NC	NC	悬空
18	D0 (SCL)	I/O	串行时钟 (SCL)
19-20	D1-D2 (SDA)	I/O	串行数据 (SDA),
20	D2 (SDA)	I/O	串行数据 (SDA), D1、D2 连接一起做 SDA
21	NC	NC	悬空
22	NC	NC	悬空
23	NC	NC	悬空
24	NC	NC	悬空
25	NC	NC	悬空
26	IREF	IREF	悬空
27	VCOMH	VCOMH	
28	VCC	面板电源	
29	VLSS	VLSS	接地
30	NC (GND)	NC	空脚或接地

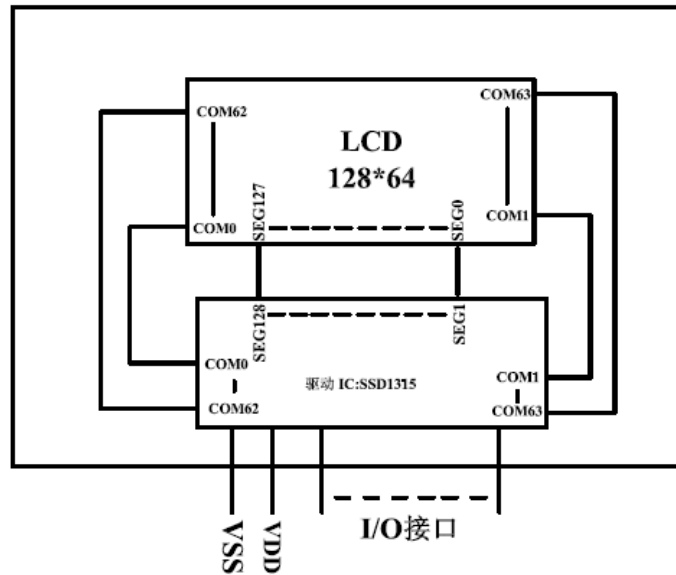
表 1: 模块的接口引脚功能

4. 基本原理

4.1 液晶屏 (LCD)

在 LCD 上排列着 128×64 点阵, 128 个列信号与驱动 IC 相连, 64 个行信号也与驱动 IC 相连, IC 邦定在 LCD 玻璃上 (这种加工工艺叫 COG)。

电路框图



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 - VSS	-0.3		4	V
LCD 驱动电压	VCC	0		16.5	V
静电电压		—	—	100	V
工作温度		-20		+70	°C
储存温度		-30		+80	°C

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
			MIN	TYPE	MAX	
工作电压 (当 3.3V 供电时)	VDD		2.4	3.3	3.6	V
工作电压 (当 5.0V 供电时)			4.8	5.0	5.2	V
输入高电平	V _{IHC}		0.8xVDD	—	VDD	V
输入低电平	V _{ILC}		VSS	—	0.2xVDD	V
输出高电平	V _{OHC}	I _{OH} = 0.2mA	0.8xVDD	—	VDD	V
输出低电平	V _{OHC}	I _{OO} = 1.2mA	VSS	—	0.2xVDD	V
模块工作电流	I _{DD}	VDD = 3.3V	—		0.3	mA

表 3: 直流 (DC) 参数

6. 读写时序特性

6.1 串行接口:

从 CPU 写到 SSD1315 (Writing Data from CPU to SSD1315)

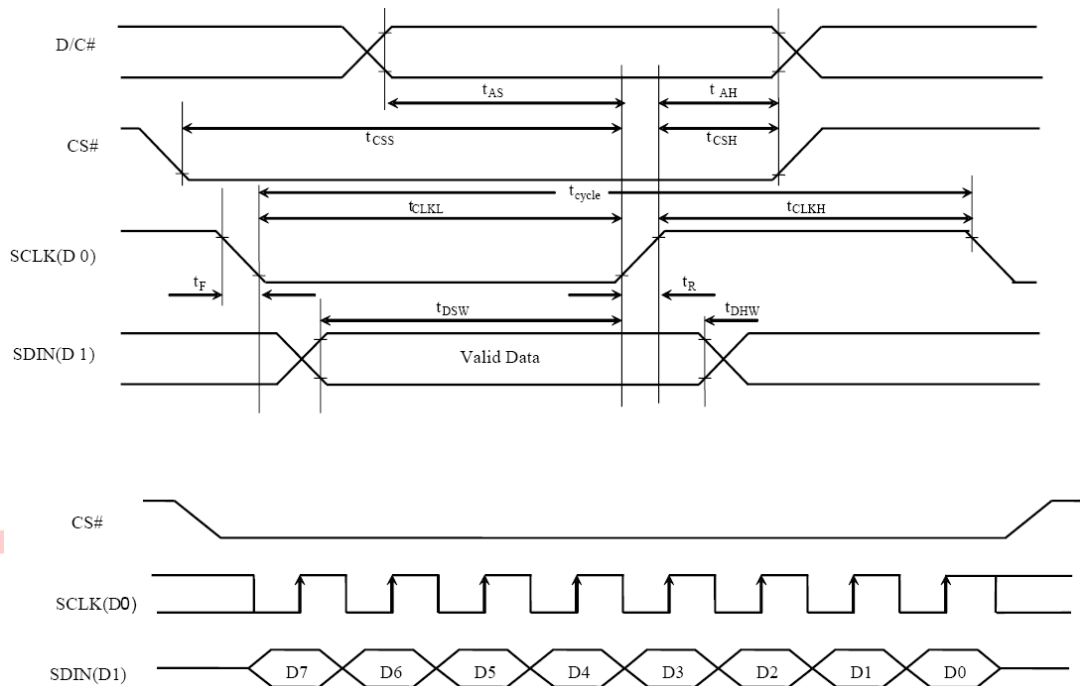


图 4. 从 CPU 写到 SSD1315 (Writing Data from CPU to SSD1315)

6.2 串行接口: 时序要求 (AC 参数):

写数据到 SSD1315 的时序要求:

表 4.

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
4线 SPI串口时钟周期 (4-line SPI Clock Period)	T_{scyc}	引脚: SCK	100	—	—	ns
保持SCK高电平脉宽 (SCK "H" pulse width)	T_{shw}	引脚: SCK	20	—	—	ns
保持SCK低电平脉宽 (SCK "L" pulse width)	T_{slw}	引脚: SCK	20	—	—	ns
地址建立时间 (Address setup time)	T_{sas}	引脚: RS	15	—	—	ns
地址保持时间 (Address hold time)	T_{sah}	引脚: RS	15	—	—	ns
数据建立时间 (Data setup time)	T_{sds}	引脚: SI	15	—	—	ns
数据保持时间 (Data hold time)	T_{sdh}	引脚: SI	15	—	—	ns
片选信号建立时间 (CS-SCL time)	T_{css}	引脚: CS	20	—	—	ns
片选信号保持时间 (CS-SCL time)	T_{csh}	引脚: CS	10	—	—	ns

* (VDD =1.65V~3.3V, Ta = 25°C)

6.3 I²C 接口:

从 CPU 写到 SSD1315 (Writing Data from CPU to SSD1315)

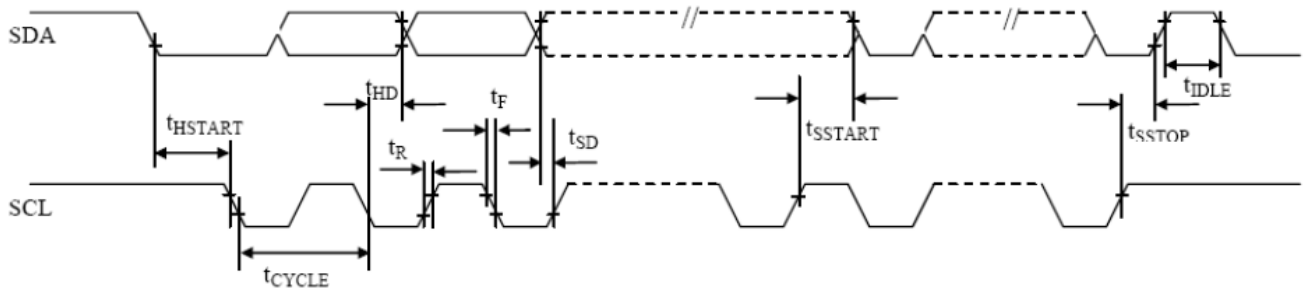


图 4. 从 CPU 写到 SSD1315 (Writing Data from CPU to SSD1315)

6.4 I²C 接口: 时序要求 (AC 参数):

写数据到 SSD1315 的时序要求:

表 4.

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
SPI 串口时钟周期 (SPI Clock Period)	T_{scyc}	引脚: SCK	2.5	—	—	ns
保持 SCK 高电平脉宽 (SCK "H" pulse width)	T_{shw}	引脚: SCK	0.6	—	—	ns
保持 SCK 低电平脉宽 (SCK "L" pulse width)	T_{SLW}	引脚: SCK	0.6	—	—	ns
数据建立时间 (Data setup time)	T_{sds}	引脚: SDA	100	—	—	ns
数据保持时间 (Data hold time)	T_{SDH}	引脚: SDA	300	—	—	ns

* (VDD = 1.65V~3.3V, Ta = 25°C)

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

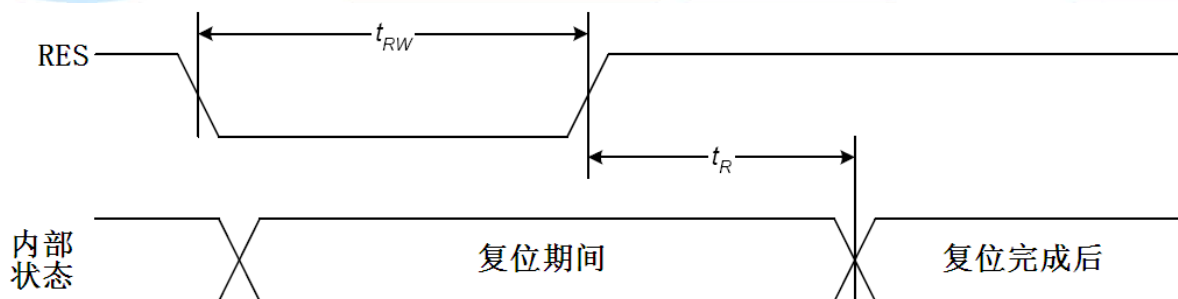


图 7: 电源启动后复位的时序

表 6: 电源启动后复位的时序要求

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
复位时间	t_R		—	—	1.0	us
复位保持低电平的时间	t_{RW}	引脚: RES	1.0	—	—	us

7. 指令功能:

7.1 指令表

指令名称		指令码								说明	
		RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1		DB0
(1)显示开/关 (display on/off)		0	1	0	1	0	1	1	1	0 1	显示开/关: 0XAE :关, 0XAF : 开
(2)显示初始行设置 (Display start line set)		0	0	1	显示初始行地址, 共 6 位						设置显示存储器的显示初始行,可设置值为 0X40~0X7F ,分别代表第 0~63 行, 针对该液晶屏一般设置为 0x40
(3)页地址设置 (Page address set)		0	1	0	1	1	显示页地址, 共 4 位				设置页地址。每 8 行为一个页, 64 行分为 8 个页, 可设置值为: 0XB0~0XB8 分别对应第一页到第九页, 第九页是一个单独的一行图标, 本液晶屏没有这一行图标, 所以设置值为 0XB0~0XB7 分别对应第一页~第八页。
(4)	列地址高4位设置	0	0	0	0	1	列地址的高 4 位			高 4 位与低 4 位共同组成列地址, 指定 128 列中的其中一列。比如液晶模块的第 100 列地址十六进制为 0x64 , 那么此指令由 2 个字节来表达: 0x16, 0x04	
	列地址低4位设置		0	0	0	0	列地址的低 4 位				
(5) 读状态 (Status read)		0	状态				0	0	0	0	串口时: 读驱动IC的当前状态, 串口时不能用此指令。
(6)写显示数据到液晶屏 (Display data write)		1	8 位显示数据								从 CPU 写数据到液晶屏, 每一位对应一个点阵, 1 个字节对应 8 个竖置的点阵
(7)读液晶屏的显示数据 (Display data read)		1	8 位显示数据								串口时: 读已经显示到液晶屏上的点阵数据。 串口时不能用此指令。
(8) 显示列地址增减 (ADC select)			1	0	1	0	0	0	0	0 1	显示列地址增减: 0xA0 : 反转: 列地址从右到左, 0xA1 : 常规: 列地址从左到右
(9)显示正显/反显 (Display normal/reverse)		0	1	0	1	0	0	1	1	0 1	显示正显/反显: 0xA6 : 常规: 正显 0xA7 : 反显
(10)显示全部点阵 (Display all points)		0	1	0	1	0	0	1	0	0 1	显示全部点阵: 0xA4 : 常规 0xA5 : 显示全部点阵
(11) 行扫描顺序选择 (Common output mode select)			1	1	0	0	0	0	0	0 1	行扫描顺序选择: 0XC0 :普通扫描顺序: 从上到下 0XC8 :反转扫描顺序: 从下到上
(12)OLED 振荡频率设置 (Oscillator Frequency)		0	1	1	0	1	0	1	0	1	设置振荡频率: 范围: 0000-1111 , 参考指令: 0Xd5 0X80
(13) 电源控制 (Power control set)		0	1	0	0	0	1	1	0	1	设置升压: 0X8d 0X14

(14)	内部设置液晶电压模式	0	1	0	0	0	0	0	0	1	设置内部电阻微调，可以理解为 微调 对比度值，此两个指令需紧接着使用。上面一条指令 0x81 是不改的，下面一条指令可设置范围为： 0x00~0xFF ,数值越大对比度越浓，越小越淡
	设置的电压值		0	0	6 位电压值数据，0~63 共 64 级						
(15)	静态图标显示： 开/关	0	1	0	1	0	1	1	1	0 1	静态图标的开关设置： 0xAE : 关, 0xAF : 开。 此指令在进入及退出睡眠模式时起作用
(16)	省电模式 (Power save)										省电模式，此非一条指令，是由“(10)显示全部点阵”、(19)静态图标显示：开/关等指令合成一个“省电功能”。详细看 IC 规格书 “POWER SAVE”部分
(17)	空指令 (NOP)	0	1	1	1	0	0	0	1	1	空操作

7.4 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

点亮液晶模块的步骤

硬件准备：
开发板（或专门设计的主板）、单片机、电源、连接线、仿真器或程序下载器（又名烧录器）

正确地接线
根据说明书正确地与开发板连接，连接的线包括：液晶模块电源线、背光源电源线、IO端口（接口）
IO端口包括：并口时：CS、RESET、RW、E、RS、D0—D7, 串口时：CS、SCLK、SDA、RESET、RS

编写软件
背光给合适的直流电可以点亮，但液晶屏里面没有程序，只给电不能让液晶屏显示（我们通常说“点亮”），程序须另外编写，并烧录（下载）到单片机里液晶模块才能工作。

7.5 程序举例:

液晶模块与 MPU (以 8051 系列单片机为例) 接口图如下:

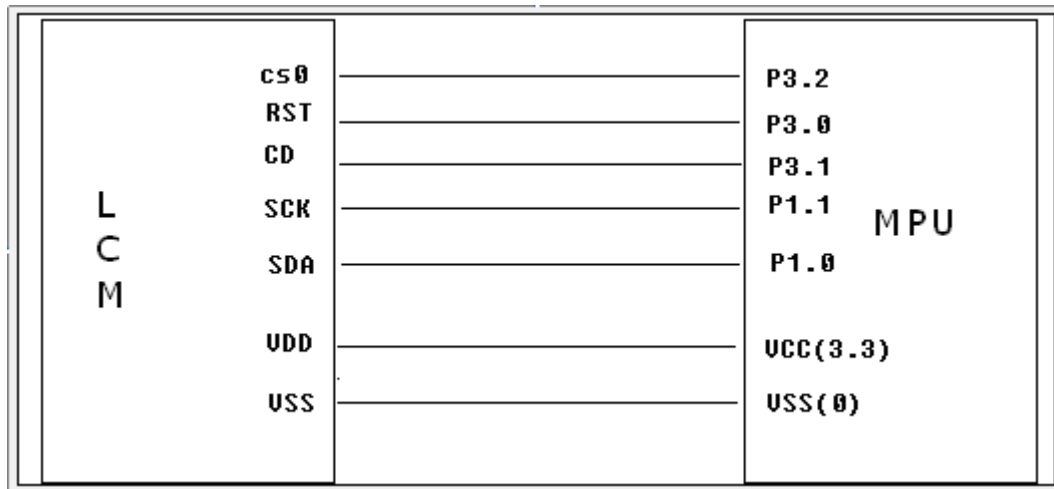
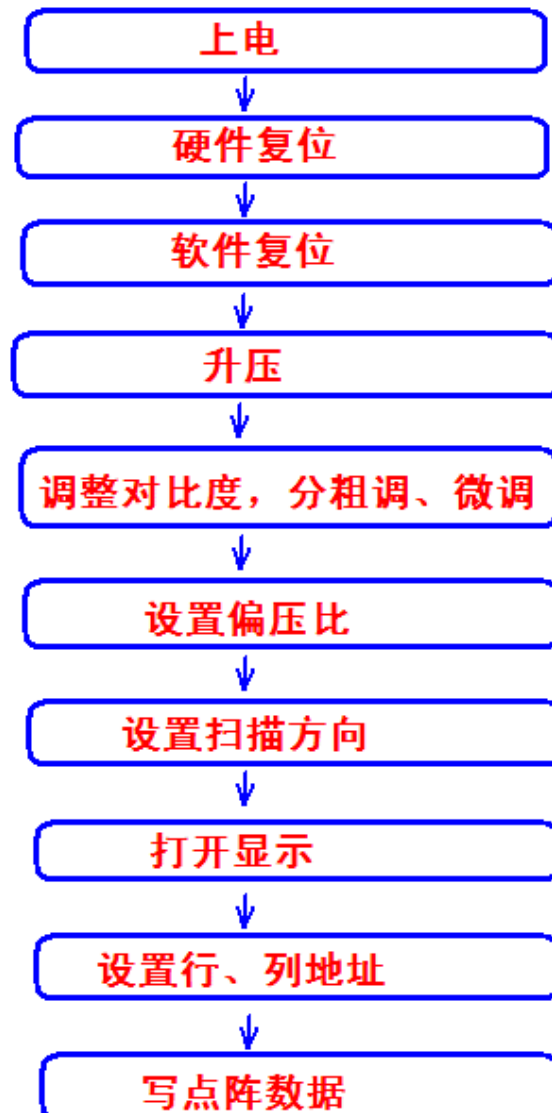
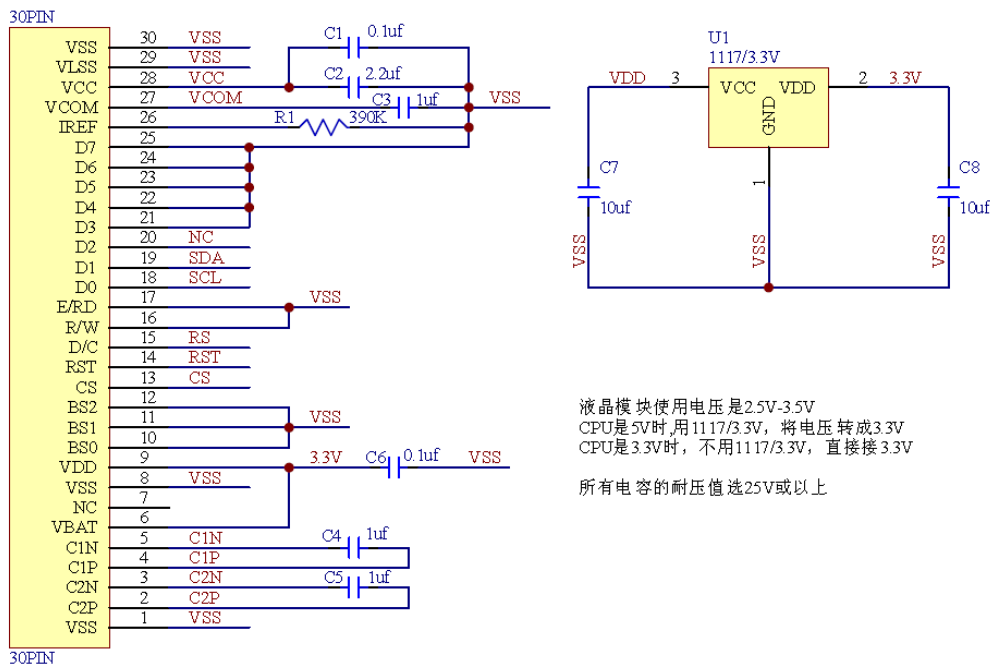


图 8. 串行接口

7.5.1 程序:

点亮液晶模块的编程步骤





液晶模块使用电压是2.5V-3.5V
 CPU是5V时,用1117/3.3V, 将电压转换成3.3V
 CPU是3.3V时, 不用1117/3.3V, 直接接3.3V
 所有电容的耐压值选25V或以上

```
// 液晶演示程序
// 液晶模块型号: JLX12864OLED-096X, 串行接口!
// 驱动 IC 是:SSD1315
// 版权所有: 晶联讯电子; 网址 http://www.jlxlcd.cn;
#include <reg52.h>
#include <intrins.h>
#include <string.h>
#include <stdio.h>
//=====
sbit lcd_sclk =P1^1; //接口定义:lcd_sclk 就是 LCD 的 SCLK //SCLK 接到“D0”脚
sbit lcd_sda =P1^0; //接口定义:lcd_sda 就是 LCD 的 SDA //SDIN 接到“D1”脚
sbit lcd_reset=P3^0; //接口定义:lcd_reset 就是 LCD 的 RESET
sbit lcd_dc =P3^1; //接口定义:lcd_dc 就是 LCD 的 D/C
sbit lcd_cs1=P3^2; //接口定义:lcd_cs1 就是 LCD 的 CS
sbit key=P2^0; //定义一个按键: P2.0 口与 GND 之间接一个按键
//=====
#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long

#include <ASCII_CODE_8X16_5X8_VERTICAL.H>
#include <Chinese_And_Graphic.H>

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

//等待按键：P2.0 口与 GND 之间接一个按键

```
void waitkey()
{
repeat:   if(key==1) goto repeat;
          else delay(2000);
}
```

//写指令到 OLED 显示模块

```
void transfer_command(int data1)
{
    uchar i;
    lcd_cs1=0;
    lcd_dc= 0;
    for(i=0;i<8;i++)
    {
        lcd_sclk = 0;
        if (data1 & 0x80)  lcd_sda = 1;
        else                lcd_sda = 0;
        lcd_sclk = 1;
        data1 <<= 1;
    }
    lcd_cs1=1;
}
```

//写数据到 OLED 显示模块

```
void transfer_data(int data1)
{
    uchar i;
    lcd_cs1=0;
    lcd_dc= 1;
    for(i=0;i<8;i++)
    {
        lcd_sclk = 0;
        if (data1 & 0x80)  lcd_sda = 1;
        else                lcd_sda = 0;
        lcd_sclk = 1;
        data1 <<= 1;
    }
    lcd_cs1=1;
}
```

//OLED 显示模块初始化

```
void initial_lcd()
{
    lcd_reset=0;        //低电平复位
    delay(500);
}
```

```
lcd_reset=1;          //复位完毕
delay(200);

transfer_command(0xae); //关显示

transfer_command(0xd5); //晶振频率
transfer_command(0x80);

transfer_command(0xa8); //duty 设置
transfer_command(0x3f); //duty=1/64

transfer_command(0xd3); //显示偏移
transfer_command(0x00);

transfer_command(0x40); //起始行

transfer_command(0x8d); //升压允许
transfer_command(0x14);

transfer_command(0x20); //page address mode
transfer_command(0x02);

transfer_command(0xc8); //行扫描顺序: 从上到下
transfer_command(0xa1); //列扫描顺序: 从左到右

transfer_command(0xda); //sequential configuration
transfer_command(0x12);

transfer_command(0x81); //微调对比度, 本指令的 0x81 不要改动, 改下面的值
transfer_command(0xcf); //微调对比度的值, 可设置范围 0x00~0xff

transfer_command(0xd9); //Set Pre-Charge Period
transfer_command(0xf1);

transfer_command(0xdb); //Set VCOMH Deselect Level
transfer_command(0x40);

transfer_command(0xaf); //开显示
}

void lcd_address(uchar page,uchar column)
{
    column=column-1;          //我们平常所说的第 1 列, 在 LCD 驱动 IC 里是第 0 列。所以在这里减去 1.
    page=page-1;
    transfer_command(0xb0+page); //设置页地址。每页是 8 行。一个画面的 64 行被分成 8 个页。我们平常所说的第 1 页, 在 LCD 驱动
    IC 里是第 0 页, 所以在这里减去 1
    transfer_command(((column>>4)&0x0f)+0x10); //设置列地址的高 4 位
}
```

```

        transfer_command(column&0x0f);          //设置列地址的低 4 位
    }

```

//全屏清屏

```

void clear_screen()
{
    unsigned char i, j;
    for(j=0; j<8; j++)
    {
        lcd_address(1+j, 1);
        for(i=0; i<128; i++)
        {
            transfer_data(0x00);
        }
    }
}

```

//full display test

```

void full_display(uchar data1, uchar data2)
{
    int i, j;
    for(i=0; i<8; i++)
    {
        lcd_address(i+1, 1);
        for(j=0; j<64; j++)
        {
            transfer_data(data1);
            transfer_data(data2);
        }
    }
}

```

//测试外框是否缺划（少行、少列）

```

void test_box()
{
    int i, j;

```

//第 1 页:

```

    lcd_address(1, 1);
    transfer_data(0xff);
    for(i=1; i<127; i++)
    {
        transfer_data(0x01);
    }
    transfer_data(0xff);

```

//第 2 页:

```

    lcd_address(2, 1);
    transfer_data(0xff);

```

```
    for(i=1;i<127;i++)
    {
        transfer_data(0x80);
    }
    transfer_data(0xff);

//第 3 页:
    lcd_address(3, 1);
    transfer_data(0xff);
    for(i=1;i<127;i++)
    {
        transfer_data(0x01);
    }
    transfer_data(0xff);

//第 4 页~第 7 页:
    for(j=4;j<=7;j++)
    {
        lcd_address(j, 1);
        transfer_data(0xff);
        for(i=1;i<127;i++)
        {
            transfer_data(0x00);
        }
        transfer_data(0xff);
    }

//第 8 页:
    lcd_address(8, 1);
    transfer_data(0xff);
    for(i=1;i<127;i++)
    {
        transfer_data(0x80);
    }
    transfer_data(0xff);
}

//测试
void test()
{
    full_display(0xff, 0xff);
    waitkey();
    full_display(0x55, 0x55);
    waitkey();
    full_display(0xaa, 0xaa);
    waitkey();
    full_display(0xff, 0x00);
}
```



```

waitkey();
full_display(0x00, 0xff);
waitkey();
full_display(0x55, 0xaa);
waitkey();
full_display(0xaa, 0x55);
waitkey();
test_box();
waitkey();
}

```

//显示 128x64 点阵图像

```
void display_128x64(uchar *dp)
```

```

{
    uint i, j;
    for(j=0; j<8; j++)
    {
        lcd_address(j+1, 1);
        for (i=0; i<128; i++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}

```

//写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1

//显示 128x16 点阵图像

```
void display_128x16(uchar page, uchar column, uchar *dp)
```

```

{
    uint i, j;
    for(j=0; j<2; j++)
    {
        lcd_address(page+j, column);
        for (i=0; i<128; i++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}

```

//写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1

//显示 32x32 点阵图像、汉字、生僻字或 32x32 点阵的其他图标

```
void display_graphic_32x32(uchar page, uchar column, uchar *dp)
```

```

{
    uchar i, j;
    for(j=0; j<4; j++)
    {

```

```

    lcd_address(page+j, column);
    for (i=0;i<32;i++)
    {
        transfer_data(*dp);    //写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1
        dp++;
    }
}

```

//显示 16x16 点阵图像、汉字、生僻字或 16x16 点阵的其他图标

```

void display_graphic_16x16(uchar page, uchar column, uchar *dp)
{
    uchar i, j;
    for(j=0;j<2;j++)
    {
        lcd_address(page+j, column);
        for (i=0;i<16;i++)
        {
            transfer_data(*dp);    //写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1
            dp++;
        }
    }
}

```

//显示 8x16 点阵图像、ASCII, 或 8x16 点阵的自造字符、其他图标

```

void display_graphic_8x16(uchar page, uchar column, uchar *dp)
{
    uchar i, j;
    for(j=0;j<2;j++)
    {
        lcd_address(page+j, column);
        for (i=0;i<8;i++)
        {
            transfer_data(*dp);    //写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1
            dp++;
        }
    }
}

```

//显示 8x16 的点阵的字符串, 括号里的参数分别为 (页, 列, 字符串指针)

```

void display_string_8x16(uint page, uint column, uchar *text)
{
    uint i=0, j, k, n;
    if(column>123)
    {
        column=1;
        page+=2;
    }
}

```

```

}
while(text[i]>0x00)
{
    if((text[i]>=0x20)&&(text[i]<=0x7e))
    {
        j=text[i]-0x20;
        for(n=0;n<2;n++)
        {
            lcd_address(page+n, column);
            for(k=0;k<8;k++)
            {
                transfer_data(ascii_table_8x16[j][k+8*n]); //写数据到LCD,每写完1字节的数据后列地址自动加1
            }
        }
        i++;
        column+=8;
    }
    else
        i++;
}
}

```

//显示 5x8 的点阵的字符串, 括号里的参数分别为 (页, 列, 字符串指针)

```

void display_string_5x8(uint page,uint column,uchar reverse,uchar *text)
{
    uint i=0, j, k, disp_data;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(page, column);
            for(k=0;k<5;k++)
            {
                if(reverse==1)
                {
                    disp_data=~ascii_table_5x8[j][k];
                }
                else
                {
                    disp_data=ascii_table_5x8[j][k];
                }

                transfer_data(disp_data); //写数据到LCD,每写完1字节的数据后列地址自动加1
            }
            if(reverse==1) transfer_data(0xff); //写入一列空白列,使得5x8的字符与字符之间有一列间隔,更美观
            else transfer_data(0x00); //写入一列空白列,使得5x8的字符与字符之间有一列间隔,更美观
        }
    }
}

```

```

        i++;
        column+=6;
        if(column>123)
        {
            column=1;
            page++;
        }
    }
    else
    i++;
}
}

```

//写入一组 16x16 点阵的汉字字符串（字符串表格中需含有此字）

//括号里的参数：（页，列，汉字字符串）

void display_string_16x16(uchar page,uchar column,uchar *text)

```

{
    uchar i, j, k;
    uint address;

    j = 0;
    while(text[j] != '\0')
    {
        i = 0;
        address = 1;
        while(Chinese_text_16x16[i] > 0x7e) // >0x7f 即说明不是 ASCII 码字符
        {
            if(Chinese_text_16x16[i] == text[j])
            {
                if(Chinese_text_16x16[i + 1] == text[j + 1])
                {
                    address = i * 16;
                    break;
                }
            }
            i += 2;
        }

        if(column > 113)
        {
            column = 0;
            page += 2;
        }

        if(address != 1)// 显示汉字
        {

```

```

        for(k=0;k<2;k++)
        {
            lcd_address(page+k, column);
            for(i = 0; i < 16; i++)
            {
                transfer_data(Chinese_code_16x16[address]);
                address++;
            }
        }
        j += 2;
    }
    else //显示空白字符
    {
        for(k=0;k<2;k++)
        {
            lcd_address(page+k, column);
            for(i = 0; i < 16; i++)
            {
                transfer_data(0x00);
            }
        }
        j++;
    }
    column+=16;
}

```

//显示 16x16 点阵的汉字或者 ASCII 码 8x16 点阵的字符混合字符串
 //括号里的参数: (页, 列, 字符串)

```

void disp_string_8x16_16x16(uchar page, uchar column, uchar *text)
{
    uchar temp[3];
    uchar i = 0;

    while(text[i] != '\0')
    {
        if(text[i] > 0x7e)
        {
            temp[0] = text[i];
            temp[1] = text[i + 1];
            temp[2] = '\0'; //汉字为两个字节
            display_string_16x16(page, column, temp); //显示汉字
            column += 16;
        }
    }
}

```

```

        i += 2;
    }
    else
    {
        temp[0] = text[i];
        temp[1] = '\0';          //字母占一个字节
        display_string_8x16(page, column, temp); //显示字母
        column += 8;
        i++;
    }
}
}

```

//显示镜像

```
void display_mirror()
```

```
{
    clear_screen();
    disp_string_8x16_16x16(1, 1, " 左右上下镜像: ");
    delay(7000);

```

```

    clear_screen();
    display_128x64 bmp12864_4;
//    delay(7000);
    waitkey();

```

```

    transfer_command(0xc8);          //上下正常: 0xc8
    transfer_command(0xa0);          //左右镜像: 0xa0
    clear_screen();
    display_128x64 bmp12864_4;
//    delay(7000);
    waitkey();

```

```

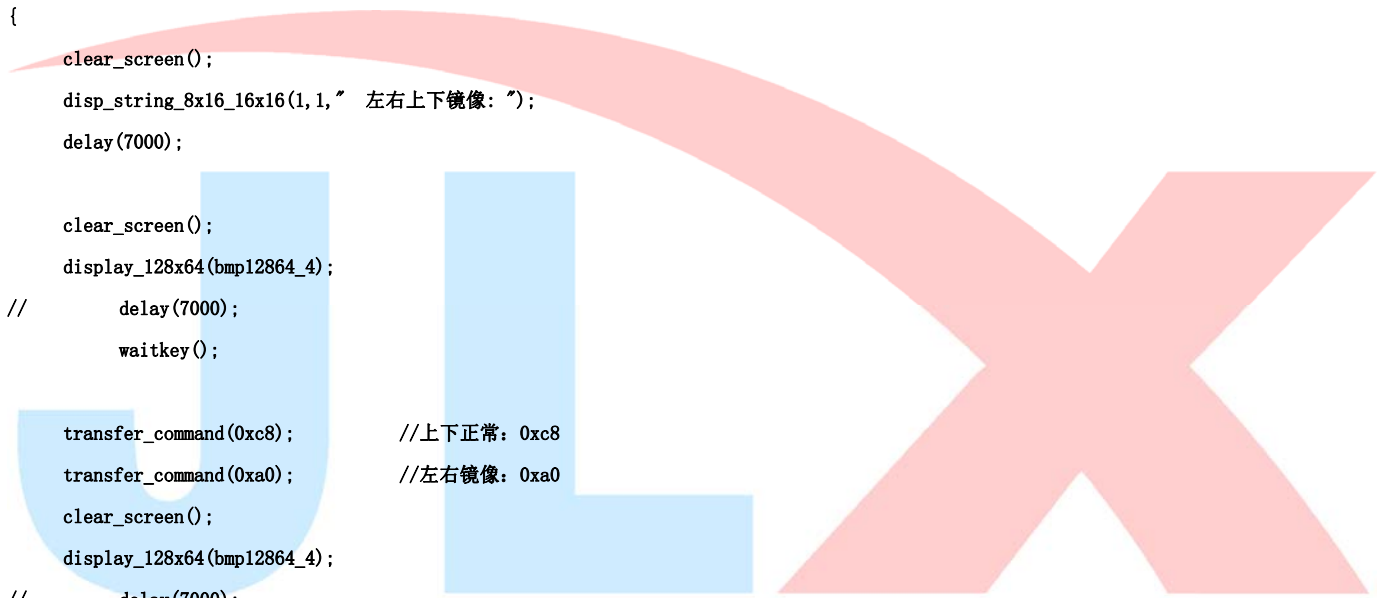
    transfer_command(0xc0);          //上下镜像: 0xc0
    transfer_command(0xa1);          //左右正常: 0xa1
    clear_screen();
    display_128x64 bmp12864_4;
//    delay(7000);
    waitkey();

```

```

    transfer_command(0xc0);          //上下镜像: 0xc0
    transfer_command(0xa0);          //左右镜像: 0xa0
    clear_screen();
    display_128x64 bmp12864_4;
//    delay(7000);
    waitkey();

```



```

transfer_command(0xc8);          //上下正常: 0xc8
transfer_command(0xa1);          //左右正常: 0xa0

}

//对比度调节
void contrast_control()
{
    clear_screen();
    disp_string_8x16_16x16(1, 1, "软件调节亮度:");

    display_string_8x16(4, 52, "0xcf");
    display_128x16(7, 1, bmp12816_1);
    display_graphic_16x16(7, 1+16*4, bmp16x16_1);
//    delay(7000);
    waitkey();

    transfer_command(0x81);
    transfer_command(0xdf);
    display_string_8x16(4, 52, "0xdf");
    display_128x16(7, 1, bmp12816_1);
    display_graphic_16x16(7, 1+16*5, bmp16x16_1);
//    delay(7000);
    waitkey();
    transfer_command(0x81);
    transfer_command(0xef);
    display_string_8x16(4, 52, "0xef");
    display_128x16(7, 1, bmp12816_1);
    display_graphic_16x16(7, 1+16*6, bmp16x16_1);
//    delay(7000);
    waitkey();

    transfer_command(0x81);
    transfer_command(0xff);
    display_string_8x16(4, 52, "0xff");
    display_128x16(7, 1, bmp12816_1);
    display_graphic_16x16(7, 1+16*7, bmp16x16_1);
//    delay(7000);
    waitkey();

    transfer_command(0x81);
    transfer_command(0x00);
    display_string_8x16(4, 52, "0x00");
    display_128x16(7, 1, bmp12816_1);
    display_graphic_16x16(7, 1+16*0, bmp16x16_1);
//    delay(7000);
}

```

```

waitkey();

transfer_command(0x81);
transfer_command(0x5f);
display_string_8x16(4, 52, "0x5f");
display_128x16(7, 1, bmp12816_1);
display_graphic_16x16(7, 1+16*1, bmp16x16_1);
//
delay(7000);
waitkey();

transfer_command(0x81);
transfer_command(0xcf);
display_string_8x16(4, 52, "0xcf");
display_128x16(7, 1, bmp12816_1);
display_graphic_16x16(7, 1+16*4, bmp16x16_1);
//
delay(7000);
waitkey();
}

void main(void)
{
    while(1)
    {
        initial_lcd(); //初始化
        clear_screen(); //清屏

//演示 32x32 点阵的汉字, 16x16 点阵的汉字, 8x16 点阵的字符, 5x8 点阵的字符
        display_string_5x8(1, 1, 0, "{(5x8dot ASCII char)}"); //显示字符串, 括号里的参数分别为 (PAGE, 列, 字符串指针)
        display_string_5x8(2, 1, 0, "{[(<~!@#%&*_+=?>)]}");
        disp_string_8x16_16x16(3, 1, "标准 16x16dot 汉字"); //显示 16x16 点阵汉字串或 8x16 点阵的字符串, 括号里的参数分别为 (页, 列, 字符串指针)

        display_graphic_32x32 (5, 1+32*0, jing1); //显示单个 32x32 点阵的汉字, 括号里的参数分别为 (PAGE, 列, 字符指针)

        display_graphic_32x32 (5, 1+32*1, lian1);
        display_graphic_32x32 (5, 1+32*2, xun1);
        disp_string_8x16_16x16(5, 1+32*3, "JLX:");
        disp_string_8x16_16x16(7, 1+32*3, "OLED");
        waitkey();

//演示显示一页纯英文的 5x8 点阵的菜单界面
        clear_screen(); //clear all dots
        display_string_5x8(1, 1, 1, "012345678901234567890");
        display_string_5x8(1, 1, 1, " MENU "); //显示 5x8 点阵的字符串, 括号里的参数分别为 (页, 列, 是否反显, 数据指针)
        display_string_5x8(3, 1, 0, "Select>>>>");
        display_string_5x8(3, 64, 1, "1. Graphic ");
        display_string_5x8(4, 64, 0, "2. Chinese ");
        display_string_5x8(5, 64, 0, "3. Movie ");
    }
}

```



```

display_string_5x8(6,64,0,"4. Contrast");
display_string_5x8(7,64,0,"5. Mirror ");
display_string_5x8(8,1,1,"PRE USER DEL NEW");
display_string_5x8(8,19,0," ");
display_string_5x8(8,65,0," ");
display_string_5x8(8,97,0," ");
waitkey();

//演示对比度调节
contrast_control();
waitkey();

//演示镜像设置
display_mirror();
waitkey();
test();
}
}

```

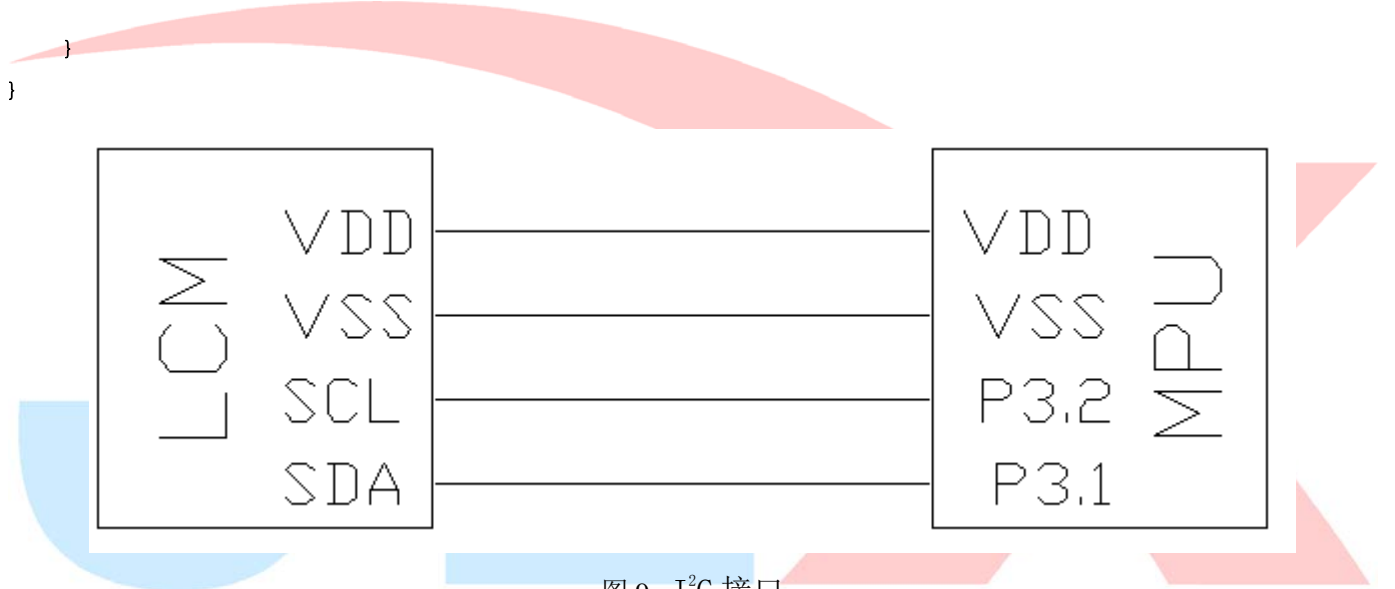
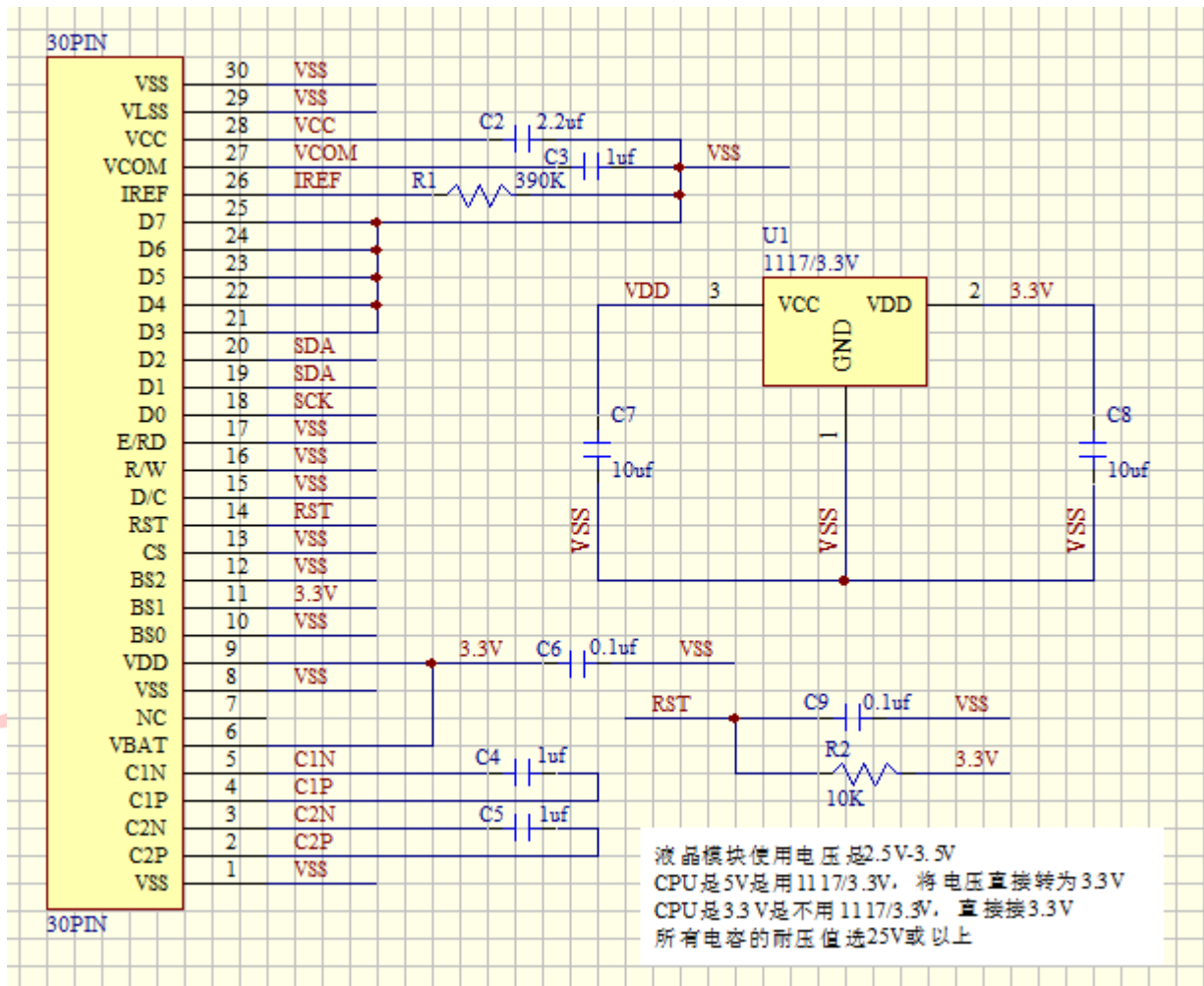


图9. I²C 接口



与串行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

```
// 液晶演示程序
//      液晶模块型号：JLX12864OLED096X-IIC, IIC 接口！
//      驱动 IC 是:SSD1315
//      版权所有：晶联讯电子；网址 http://www.jlxlcd.cn;

#include <reg52.h>
#include <intrins.h>
#include <string.h>
#include <stdio.h>

//=====
sbit lcd_scl  =P3^2; //接口定义:lcd_sclk 就是 LCD 的 SCL
sbit lcd_sda  =P3^1; //接口定义:lcd_sda 就是 LCD 的 SDA

sbit key=P2^0; //定义一个按键：P2.0 口与 GND 之间接一个按键

#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long
```

```
#include <ASCII_CODE_8X16_5X8_VERTICAL.H>
```

```
#include <Chinese_And_Graphic.H>
```

```
//延时
```

```
void delay(int i)
```

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

```
//短延时
```

```
void delay_us(int i)
```

```
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<1;k++);
}
```

```
//等待按键：P2.0 口与 GND 之间接一个按键
```

```
void waitkey()
```

```
{
repeat:   if(key==1) goto repeat;
          else delay(1500);
}
```

```
void start_flag()
```

```
{
    lcd_scl=1;
    delay_us(1);
    lcd_sda=1;
    delay_us(1);
    lcd_sda=0;
    delay_us(1);
    lcd_scl=0;
    delay_us(1);
}
```

```
void stop_flag()
```

```
{
    lcd_scl=0;
    delay_us(1);
    lcd_sda=0;
    delay_us(1);
    lcd_sda=1;
    delay_us(1);
    lcd_scl=1;
}
```

```
    delay_us(1);
}

//传 8 位指令或数据到 OLED 显示模块
void transfer(uchar data1)
{
    unsigned char j;
    for(j=0;j<8;j++)
    {
        lcd_scl=0;
        if(data1&0x80)    lcd_sda=1;
        else             lcd_sda=0;
        data1<<=1;
        lcd_scl=1;
        delay_us(1);
    }
    lcd_scl=0;
    lcd_scl=1;
}

//写指令到 OLED 显示模块
void transfer_command(uchar com)
{
    start_flag();
    transfer(0x78);
    transfer(0x00);
    transfer(com);
    stop_flag();
}

//写数据到 OLED 显示模块
void transfer_data(uchar dat)
{
    start_flag();
    transfer(0x78);
    transfer(0x40);
    transfer(dat);
    stop_flag();
}
```