

JLX12832G-908-BN 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	3~4
4	基本原理	4~5
5	技术参数	5
6	时序特性	6~7
7	指令功能及硬件接口与编程案例	7~末页

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 JLX12832G-908 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。

JLX12832G-908 可以显示 128 列*32 行点阵单色图片，或显示 8 个/行*2 行 16*16 点阵的汉字，或显示 16 个/行*8 行 5*8 点阵的英文、数字、符号。

2. JLX12832G-908 图像型点阵液晶模块的特性

2.1 结构轻、薄、带背光、焊接式 FPC。

2.2 IC 采用 UC1604c, 功能强大，稳定性好

2.3 功耗低:当电压为 3.3V 时，功耗低：不带背光 1mW ($3.3V \times 0.3mA$)，带背光不大于 50mW ($3.3V \times 15mA$)；

2.4 显示内容：

- 128*32 点阵单色图片；

- 可选用 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*2 行。

- 按照 12*12 点阵汉字来计算可显示 10 字/行*2 行。

- 可显示 16 个*2 行 8*16 点阵的英文、数字、符号。

- 可显示 21 个*4 行 5*8 点阵的英文、数字、符号。

2.5 指令功能强:可软件调对比度、正显/反显转换、行列扫描方向可改（可旋转 180 度使用）。

2.6 接口简单方便：采用 4 线 SPI 串口

2.7 工作温度宽： -20°C - $+70^{\circ}\text{C}$ ；



3. 外形尺寸及接口引脚功能

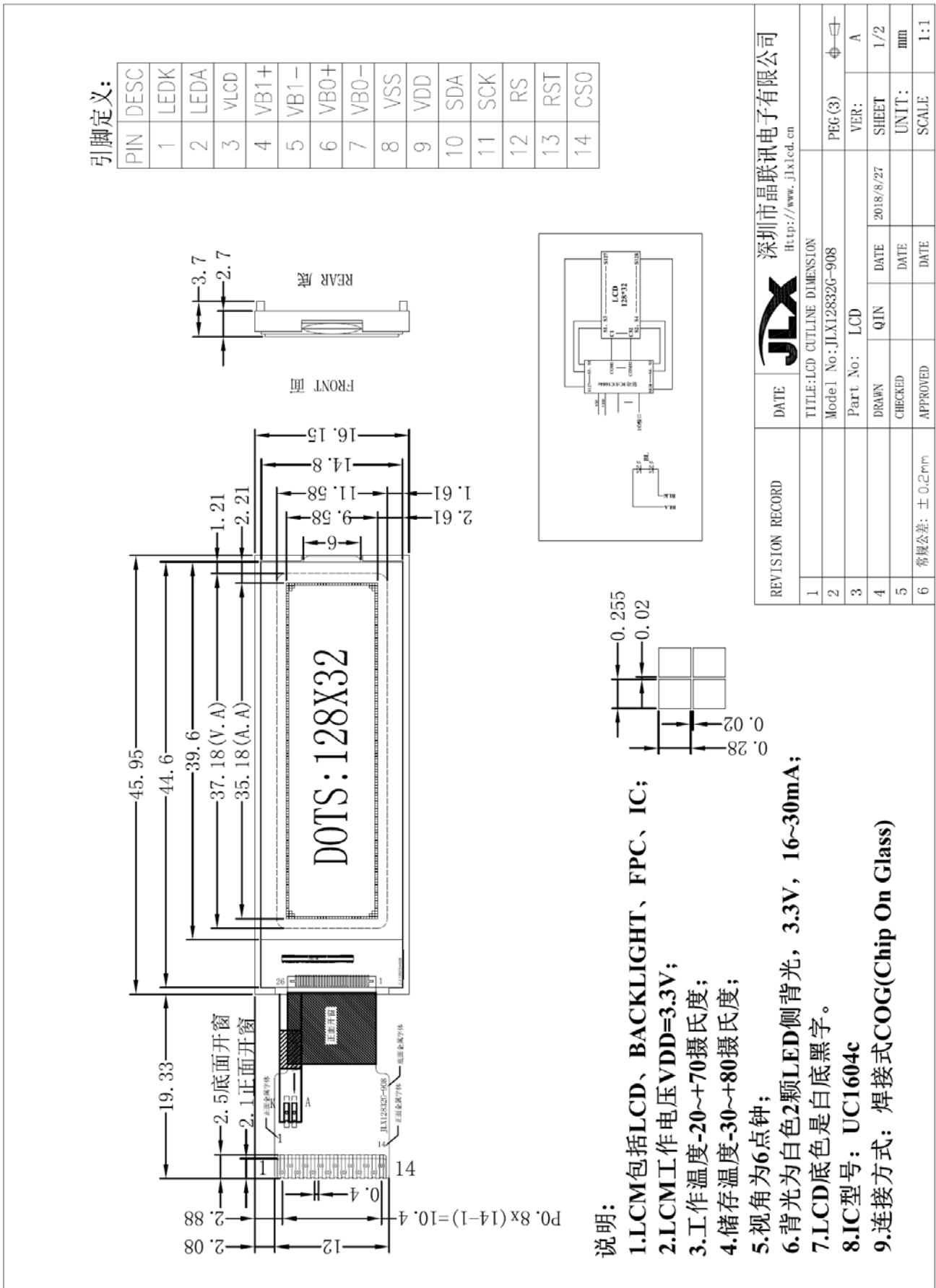


图 1. 外形尺寸

模块的接口引脚功能：

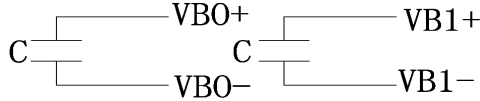
引 线 号	符 号	名 称	功 能	
1	LEDK	背光电源	供电电源负极（VSS）	
2	LEDA	背光电源	供电电源正极（VDD）	
3	VLCD	VLCD	升压输出	
4	VB1+	升压电容	倍压电路	
5	VB1-	升压电容	倍压电路	
6	VB0-	升压电容	倍压电路	
7	VB0+	升压电容	倍压电路	
8	VSS	接地	电源输入地	
9	VDD	供电电源正极	供电电源正极	
10	SDA	I/O	串行接口时：串行数据(SDA)	
11	SCK	I/O	串行接口时：串行时钟(SCLK)	
12	CD(RS)	寄存器选择信号	H:数据寄存器 0:指令寄存器	
13	RST	复位	低电平复位，复位完成后，回到高电平，液晶模块开始工作	
14	CS0 (CS)	片选	低电平片选	

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 128×32 点阵, 128 个列信号与驱动 IC 相连, 32 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）。

4.2 工作电图：

图 2 是 JLX12832G-908 图像点阵型模块的电路框图, 它由驱动 UC1604c 及几个电阻电容组成。

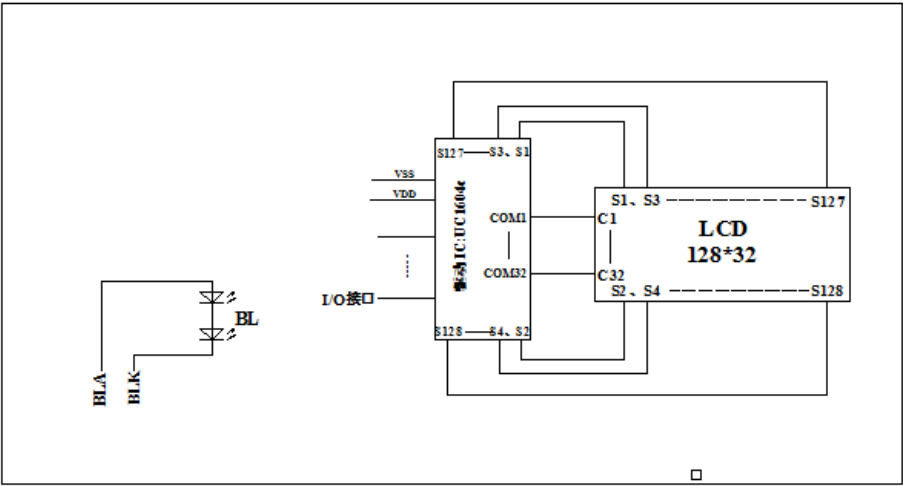


图 2: JLX12832G-908 图像点阵型液晶模块的电路框图

4.2 背光参数

该型号液晶模块带 LED 背光源。它的性能参数如下：

背光板可选择白色。

正常工作电流为：16~40mA（LED 灯数共 2 颗）；

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 - VSS	-0.3	3.3	3.6	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	VDD		2.4	3.3	3.6	V
背光工作电压	VLED		2.9	3.0	3.1	V
输入高电平	VIH	-	0.8xVDD		VDD	V
输入低电平	VI0	-	VSS		0.6	V
输出高电平	VOH	IOH = 0.2mA	0.8xVDD		VDD	V
输出低电平	VO0	I00 = 1.2mA	VSS		0.2xVDD	V
模块工作电流	IDD	VDD = 3.0V	-		0.3	mA
背光工作电流	ILED	VLED=3.0V	16	30	40	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 UC1604c（Writing Data from CPU to UC1604c）

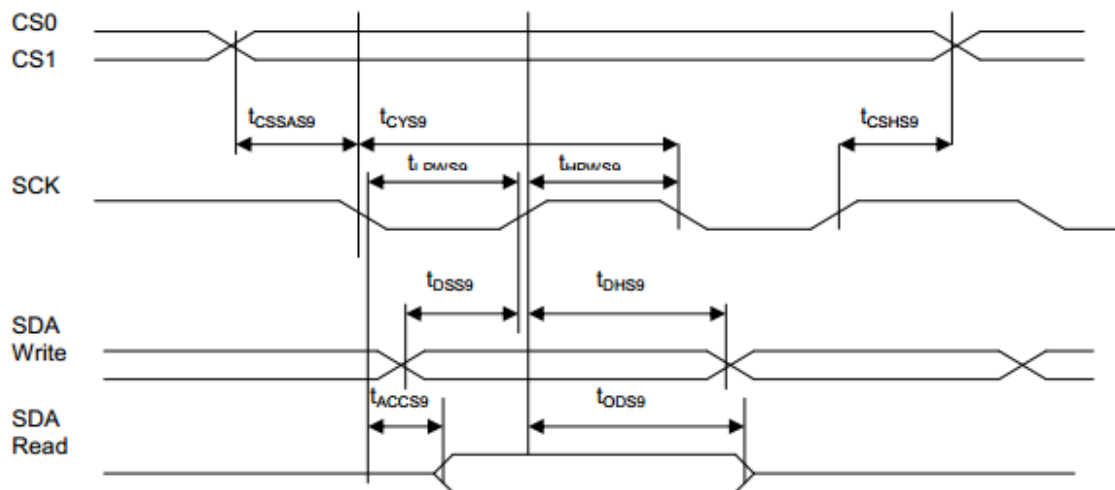


FIGURE 18: Serial Bus Timing Characteristics (for S9)

图 6. 从 CPU 写到 UC1604C (Writing Data from CPU to UC1604C)

6.2 串行接口：时序要求（AC 参数）：

写数据到 UC1604C 的时序要求：（6800 系列 MPU）

Symbol	Signal	Description	Condition	Min.	Max.	Unit
(2.5V ≤ V _{DD} ≤ 3.6V, T _a = -30 to +85°C)						
t _{CSSAS9}	CS1, CS0	Chip select setup time		5	—	nS
t _{CSSH9}		Chip select hold time		5	—	nS
t _{CYS9}	SCK	System cycle time		190 / 70	—	nS
t _{LPWS9}		Low pulse width		80 / 20	—	nS
t _{HPWS9}	SDA (Write)	High pulse width		80 / 20	—	nS
t _{DSS9}		Data setup time		20	—	nS
t _{DHS9}	SDA (Read)	Data hold time		10	—	nS
t _{ACC9}		Read access time	C _L = 100pF	—	80	nS
t _{OD9}		Output disable time		—	30	nS
(1.7V ≤ V _{DD} < 2.5V, T _a = -30 to +85°C)						
t _{CSSAS9}	CS1, CS0	Chip select setup time		10	—	nS
t _{CSSH9}		Chip select hold time		10	—	nS
t _{CYS9}	SCK	System cycle time		230 / 110	—	nS
t _{LPWS9}		Low pulse width		100 / 40	—	nS
t _{HPWS9}	SDA (Write)	High pulse width		100 / 40	—	nS
t _{DSS9}		Data setup time		24	—	nS
t _{DHS9}	SDA (Read)	Data hold time		15	—	nS
t _{ACC9}		Read access time	C _L = 100pF	—	100	nS
t _{OD9}		Output disable time		—	60	nS

Note: t_r (Rising time), t_f (falling time) : ≤ 15nS

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

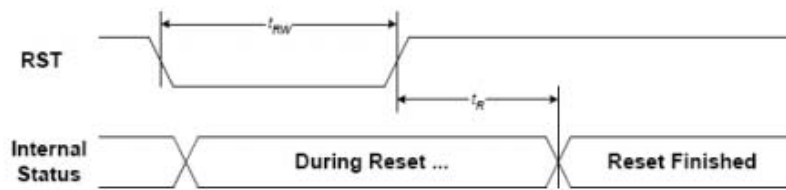


FIGURE 20: Reset Characteristics

 $(1.7V \leq V_{DD} \leq 3.6V, T_a = -30 \text{ to } +85^\circ\text{C})$

Symbol	Signal	Description	Condition	Min.	Max.	Unit
t_{RW}	RST	Reset low pulse width		3	—	μS
t_R	RST, Internal Status	Reset to Internal Status pulse delay		6	—	mS

图 7：电源启动后复位的时序

7. 指令功能：

7.1 指令表

指 令 表

表 8.

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, 32 行分为 4 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 0x63 , 那么此指令由 2 个字节来表达: 0x16, 0x03
	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	并口时: 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								并口时: 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 : 常规: 列地址从左到右, 0xA1 : 反转: 列地址从右到左
(9)显示正显/反显 (Display		0	1	0	1	0	0	1	1	0 1	显示正显/反显: 0xA6 : 常规: 正显

normal/reverse)										0xA7 : 反显	
(10)显示全部点阵 (Display all points)		0	1	0	1	0	0	1	0	0 1 0xA4 : 常规 0xA5 : 显示全部点阵	
(11)LCD 偏压比设置 (LCD bias set)		0	1	0	1	0	0	0	1	0 1 0xA2 : BIAS=1/9 (常用) 0xA3 : BIAS=1/7	
(12) 读-改-写 (Read-modify-write)		0	1	1	1	0	0	0	0	0XE0 : “读-改-写” 开始。 列地址的增加: 写入时: 列地址+1 读出时: 列地址不加 串口时不能用此指令	
(13) 退出上述“读-改-写” 指令(End)		0	1	1	1	0	1	1	1	0XEE :上述“读-改-写” 指令结束	
(14) 软件复位 (Reset)		0	1	1	1	0	0	0	1	0XE2 :软件复位。	
(15) 行扫描顺序选择 (Common output mode select)			1	1	0	0	0	0	0	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	
(16) 电源控制 (Power control set)			0	0	1	0	1	电压操作模式选择, 共 3 位			选择内部电压供应操作模式: D2、D1、D0 位分别对应内部升压是否打开 (1 为打开, 0 为不打开), 电压调整电路是否打开(1 为打开, 0 为不打开), 电压跟随器是否打开(1 为打开, 0 为不打开)。 通常是 0x2C,0x2E,0x2F 三条指令按顺序紧接着写, 表示依次打开内部升压、电压调整电路、电压跟随器。也可以单写 0x2F , 一次性打开三部分电路。
(17) 选择内部电阻比例		0	0	0	1	0	0	内部电压值电阻设置			选择内部电阻比例 (Rb/Ra) :可以理解为 粗调 对比度值。可设置范围为: 0x20~0x27 , 数值越大对比度越浓, 越小越淡
(18)	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: 0x00~0x3F ,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						
(19)静态图标显示: 开/关		0	1	0	1	0	1	1	0	0 1 0xAC : 关, 0xAD : 开。 此指令在进入及退出睡眠模式时起作用	
(20) 升压倍数选择 (Booster ratio set)		0	1	1	1	1	1	0	0	0	选择升压倍数: 00: 2 倍, 3 倍, 4 倍 01: 5 倍 11: 6 倍。本模块外部已设置升压倍数为 4 倍, 不必使用此指令
			0	0	0	0	0	0	2 位数设置 升压倍数		

(21) 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 规格书第 35 页“POWER SAVE”
(22)空指令 (NOP)	0	1	1	1	0	0	0	1	1	空操作
(23) 测试 (Test)	0	1	1	1	1	*	*	*	*	内部测试用，千万别用！

7.3 点阵与 DD RAM(显示数据存储器)地址的对应关系

请留意页的定义：PAGE, 与平时所讲的“页”并不是一个意思，在此表示 **8 个行就是一个“页”**，一个 128*32 点阵的屏分为 4 个“页”，从第 0“页”到第 3“页”。

DB7--DB0 的排列方向：数据是从下向上排列的。最低位 D0 是在最上面，最高位 D7 是在最下面。每一位 (bit) 数据对应一个点阵，通常“1”代表点亮该点阵，“0”代表关掉该点阵。如下图所示：

D0	0	1	1	1		0
D1	1	0	0	0		0
D2	0	0	0	0		0
D3	0	1	1	1		0
D4	1	0	0	0		0
-						

Display data RAM
(显示数据存储器)

COM0						
COM1						
COM2						
COM3						
COM4						
-						

Liquid crystal display
(液晶屏)

下图摘自 UC1604C IC 资料，可通过“UC1604c_a1.3.pdf”获取最佳效果。

Example for memory mapping: let $MX = 0$, $MY = 0$, $SL = 0$, according to the data shown in the above table:

⇒ Page 0 SEG 1 (D7-D0) : 0001 1111b
⇒ Page 0 SEG 2 (D7-D0) : 1100 1100b

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:

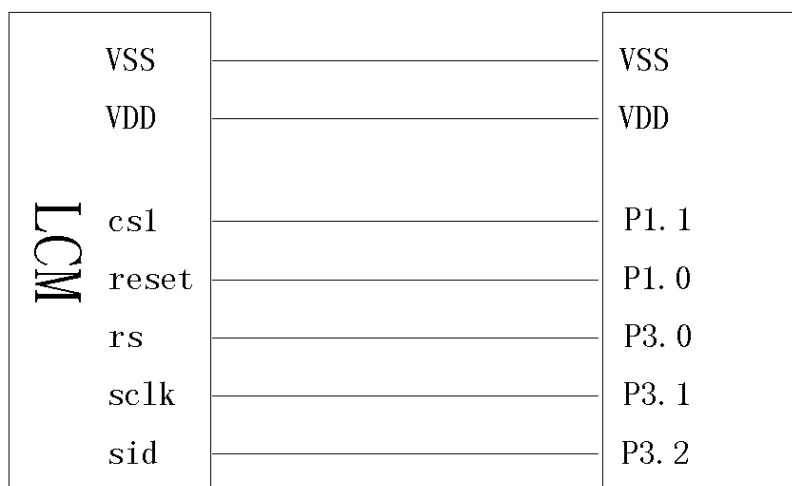
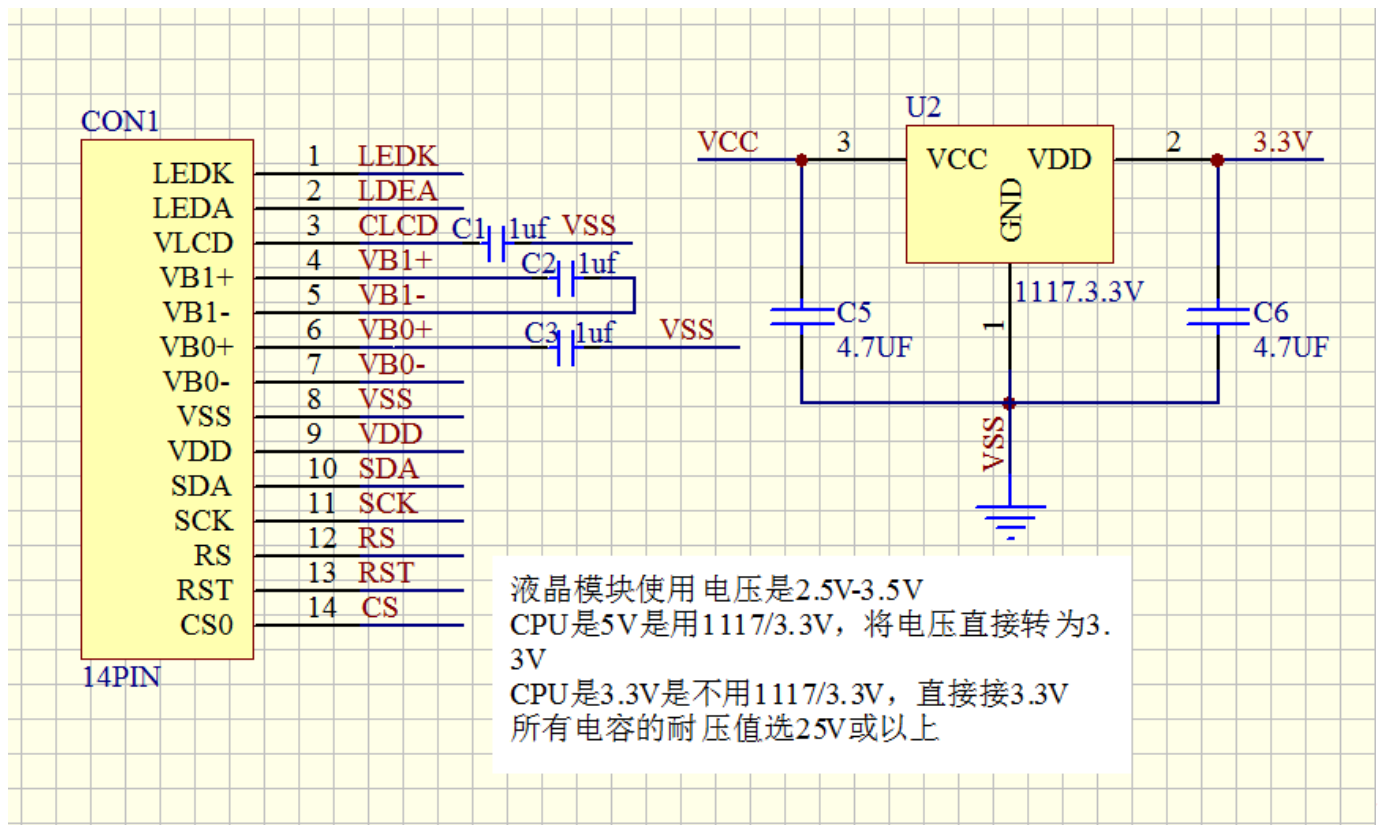


图 8. 串行接口

串行电路图



7.5.1 程序

```
// 液晶演示程序 JLX12832G-908， 串行接口！
```

```
// 驱动 IC 是:UC1604c
```

```
#include <reg52.h>
#include <intrins.h>
#include <Ctype.h>
#include <fonit.h>
```

```
sbit cs1=P1^1;
sbit reset=P1^0;
sbit rs=P3^0;
sbit scl=P3^1;
sbit sid=P3^2;
sbit key=P2^0;
```

```
void delay_us(int i);
void delay(int i);
```

```
//写指令到 LCD 模块
```

```
void transfer_command(int data1)
{
    char i;
    cs1=0;
    rs=0;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
    cs1=1;
}
```

//写数据到 LCD 模块

```
void transfer_data(int data1)
{
    char i;
    cs1=0;
    rs=1;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
    cs1=1;
}
```

//延时 1

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

//延时 2

```
void delay_us(int i)
{
}
```

```
int j,k;
for(j=0;j<i;j++)
for(k=0;k<10;k++);
}
```

```
void waitkey()
{
repeat:
    if(key==1)goto repeat;
    else delay(2800);
}
```

//LCD 模块初始化

```
void initial_lcd()
{
    reset=0;          //低电平复位
    delay(800);
    reset=1;          //复位完毕
    delay(800);
    transfer_command(0xe2); //软复位
    delay(500);
    transfer_command(0x2f); //打开内部升压
    delay(500);

    transfer_command(0x81); //微调对比度
    transfer_command(0x3a); //微调对比度的值，可设置范围 0x00~0xFF
    transfer_command(0xe9); //1/7 偏压比 (bias)
    transfer_command(0xc4); //行扫描顺序：从上到下 0xc2
    transfer_command(0xf1);
    transfer_command(0x1f);
    transfer_command(0xaf); //开显示
}
```

```
void lcd_address(uchar page,uchar column)
{
    transfer_command(0xb0+page);          //设置页地址。每页是 8 行。一个画面的 32 行被分成 4 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(i=0;i<8;i++)
{
    lcd_address(i,0);
    for(j=0;j<128;j++)
    {
        transfer_data(0x00);
    }
}
```

```
void display_graphic_128x32(uchar *dp)
{
    uchar i,j;
    for(i=0;i<4;i++)
    {
        lcd_address(i,0);
        for(j=0;j<128;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}
```

//=====display a picture of 128*32 dots=====

```
void full_display(uchar data_left,uchar data_right)
```

```
{
    int i,j;
    for(i=0;i<4;i++)
    {
        lcd_address(i,0);
        for(j=0;j<32;j++)
        {
            transfer_data(data_left);
            transfer_data(data_right);
        }
    }
}
```

//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++;
        }
    }
}
```

```
void display_string_8x16(uint page,uint column,uchar *text)
{
    uint i=0, j, k, n;
    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]); //显示 5x7 的 ASCII 字到 LCD 上, y 为页
地址, x 为列地址, 最后为数据
                }
            }
            i++;
            column+=8;
        }
        else
            i++;
    }
}
```

//显示一串 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）

```
void display_string_5x8(uint page,uint column,uchar reverse,uchar *text)
{
    uchar i=0, j, k, data1;
    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) data1=~ascii_table_5x8[j][k];
            else data1=ascii_table_5x8[j][k];
            transfer_data(data1);
        }
        if(reverse==1) transfer_data(0xff);
        else transfer_data(0x00);
        i++;
        column+=6;
    }
    else
        i++;
}

}

void display_string_5x8_1(uint page,uint column,uchar *text)
{
    uint i=0,j,k;
    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++)
            {
                transfer_data(ascii_table_5x8[j][k]); //显示 5x7 的 ASCII 字到 LCD 上, y 为页地址, x 为
                列地址, 最后为数据
            }
            i++;
            column+=6;
        }
        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)
    {
        if(Chinese_text_16x16[i] == text[j])
        {
            if(Chinese_text_16x16[i+1] == text[j+1])
            {
                address=i*16;
                break;
            }
        }
        i +=2;
    }
    if(column>191)
    {
        column=0;
        page+=2;
    }
    if(address !=1)
    {
        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
    {
        lcd_address(page, column);
        for(k=0;k<2;k++)
        {
            for(i=0;i<16;i++)
            {
                transfer_data(0x00);
            }
        }
        j++;
    }
}
```

```
    }
    column+=16;
}

}

void main(void)
{
    while(1)
    {
        initial_lcd();
        clear_screen();
        display_string_5x8(0,0,1,"      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(2,0,0,"Select>>>>");
        display_string_5x8(2,66,1,"1.Graphic ");
        display_string_5x8(3,66,0,"2.Chinese ");
        waitkey();

        clear_screen(); //clear all dots
        display_graphic_128x32(bmp1);
        waitkey();

        clear_screen(); //clear all dots
        display_string_16x16(1,0,"晶联讯液晶屏模块");
        waitkey();
        clear_screen(); //clear all dots
        display_string_8x16(0,0,"0123456789abcdef");//在第 1 页，第 1 列显示字符串
        display_string_8x16(2,0,"ghijklmnopqrstuv");//在第*页，第*列显示字符串
        waitkey();
        clear_screen();
        display_string_5x8_1(0,0,"0123456789ABCDEFGHIJK");
        display_string_5x8_1(1,0,"LMNOPQRSTUVWXYZabcdef");
        display_string_5x8_1(2,0,"ghijklmnopqrst&*|\\@#_");
        display_string_5x8_1(3,0,"{[(<\" ' abcde012#_-= ' \">>)]}"");
        waitkey();
        full_display(0xff,0xff);
        waitkey();
        full_display(0x55,0xaa);
        waitkey();
        full_display(0xaa,0x55);
        waitkey();
        delay(2000);
        full_display(0xff,0x00);
        waitkey();
```

```
    full_display(0x00, 0xff);  
    waitkey();  
}  
}
```

