



# JLX12864OLED-096X-PN-IIC 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	技术参数	5
6	时序特性	5~6
7	指令功能及硬件接口与编程案例	6~页末

## 1. 概述

晶联讯电子专注于 OLED 屏及液晶模块的研发、制造。所生产 JLX12864OLED-096X-PN-IIC 型液晶模块由于使用方便、显示清晰、视角宽、清晰度高、无需背光广泛应用于各种人机交流面板。

JLX12864OLED-096X-PN-IIC 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-PN-IIC 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；

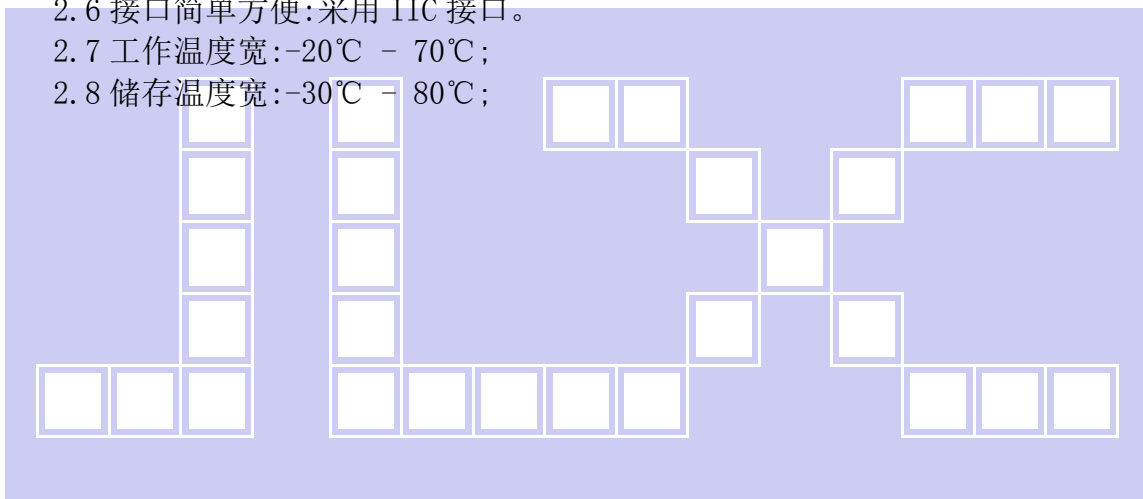
- 可選用 16\*16 点阵或其他点阵的图片来自编汉字，按照 16\*16 点阵汉字来计算可显示 8 字/行\*4 行。

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 接口简单方便:采用 IIC 接口。

2.7 工作温度宽:-20℃ - 70℃；

2.8 储存温度宽:-30℃ - 80℃；





### 模块的接口引脚功能

引线号	符号	名称	功能
1	NC	NC	
2	NC	NC	
3	NC	NC	
4	NC	NC	
5	VSS	接地	0V
6	VDD	电源电路	2.7V—5.0V
7	SCK	I/O	串行时钟
8	SDA	I/O	串行数据

表 1: 模块的接口引脚功能

## 4. 基本原理

### 4.1 OLED 屏 (OLED)

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

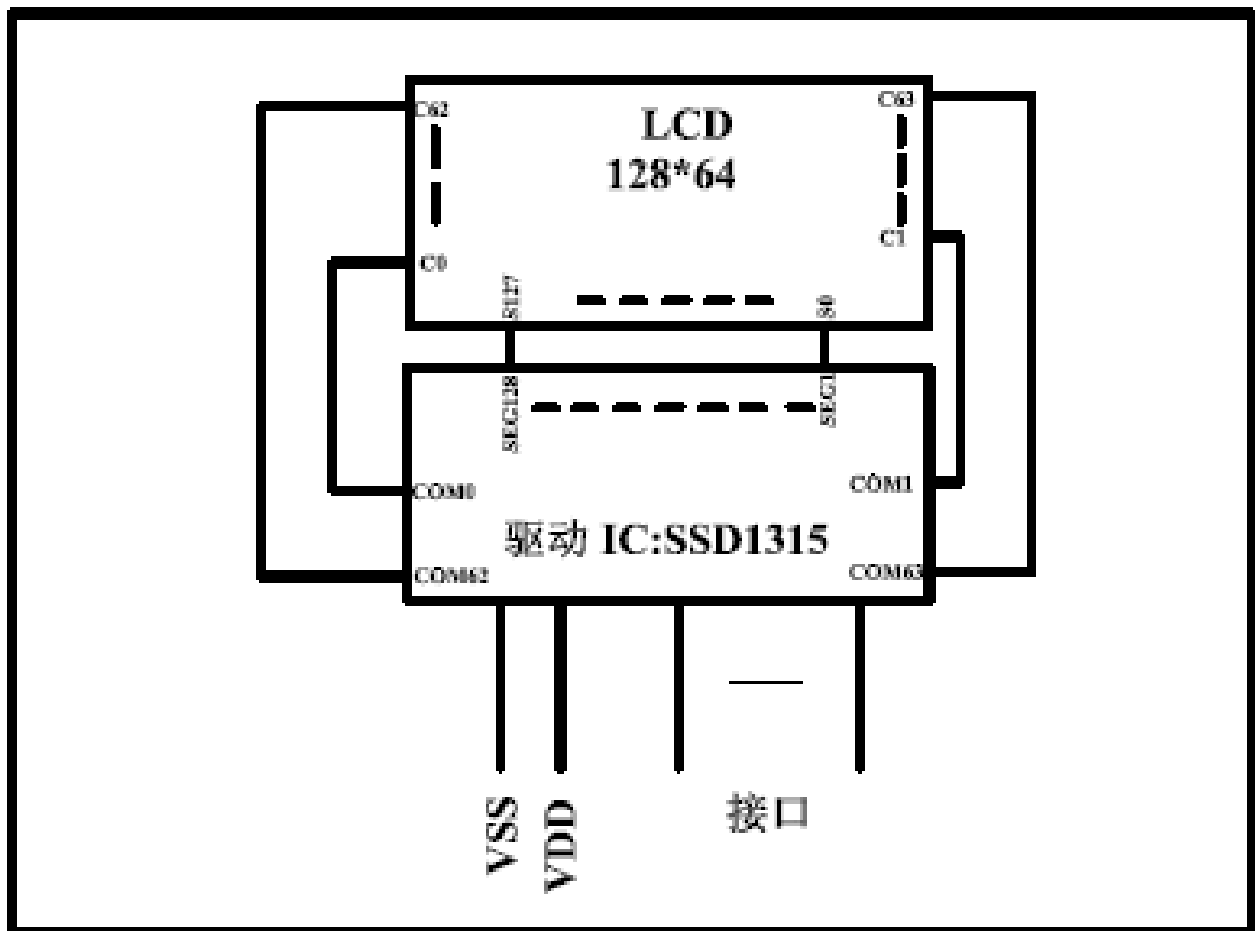


图 2. JLX12864OLED-096X-PN-IIC

## 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	4	V
驱动电压	VCC	0	—	18	V
DC 电压	VBAT	-0.3	—	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
输入高电平	V <sub>IHC</sub>		0.8xVDD	—	VDD	V
输入低电平	V <sub>ILC</sub>		VSS	—	0.2xVDD	V
输出高电平	V <sub>OHC</sub>	I <sub>OH</sub> = 0.2mA	0.8xVDD	—	VDD	V
输出低电平	V <sub>OHC</sub>	I <sub>OO</sub> = 1.2mA	VSS	—	0.2xVDD	V
模块工作电流	I <sub>DD</sub>	VDD = 3.3V	—		0.3	mA

表 3: 直流 (DC) 参数

## 6. 读写时序特性

### 6.1 IIC 接口:

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

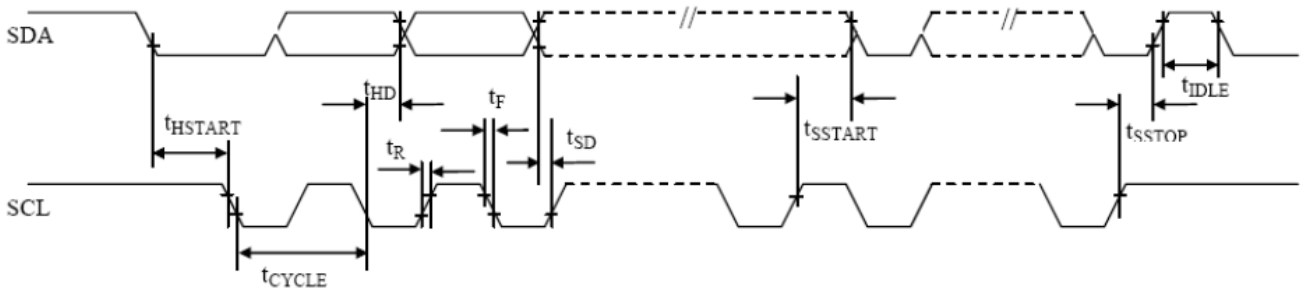


图 3. 从 CPU 写到 SSD1315 (Writing Data from CPU to SSD1315)

### 6.2 IIC 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 <sub>scyc</sub>	引脚: SCK	2.5	—	—	ns
保持 SCK 高电平脉宽 (SCK "H" pulse width)	T <sub>shw</sub>	引脚: SCK	0.6	—	—	ns
保持 SCK 低电平脉宽 (SCK "L" pulse width)	T <sub>slw</sub>	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,  $T_a = 25^\circ\text{C}$ )

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

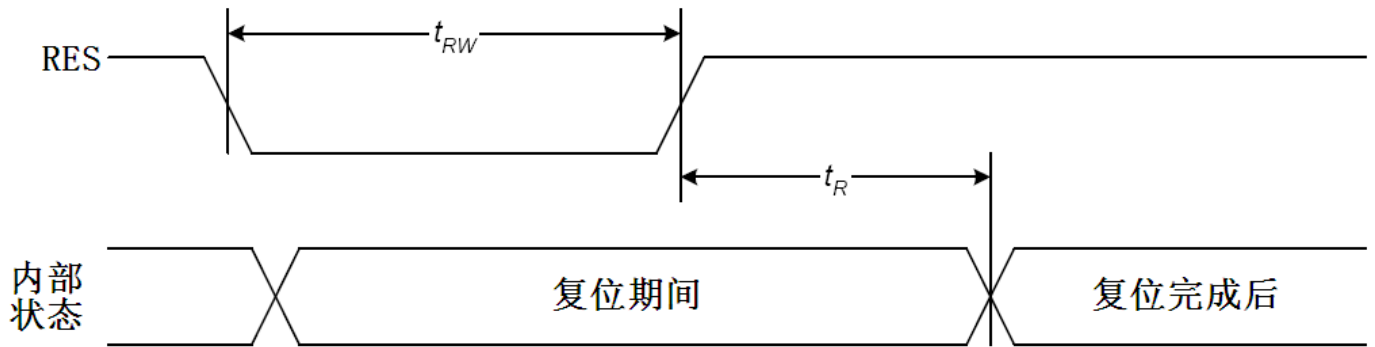


图 4: 电源启动后复位的时序

表 5: 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: <b>0XAE</b> : 关, <b>0XAF</b> : 开	
(2) 显示初始行设置 (Display start line set)	0	0	1	<b>显示初始行地址, 共 6 位</b>							设置显示存储器的显示初始行, 可设置值为 <b>0X40~0X7F</b> , 分别代表第 <b>0~63</b> 行, 针对该液晶屏一般设置为 <b>0x40</b>
(3) 页地址设置 (Page address set)	0	1	0	1	1	<b>显示页地址, 共 4 位</b>				设置页地址。每 8 行为一个页, 64 行分为 8 个页, 可设置值为: <b>0XB0~0XB8</b> 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 <b>0XB0~0XB7</b> 分别对应第一页~第八页。	
(4)	列地址高4位设置	0	0	0	0	1	<b>列地址的高 4 位</b>				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 OLED 模块的第 100 列地址十六进制为 <b>0x64</b> , 那么此指令由 2 个字节来表达: <b>0x16, 0x04</b>
	列地址低4位设置		0	0	0	0	<b>列地址的低 4 位</b>				
(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, <b>串口时不能用此指令。</b>	



(6)写显示数据到 OLED 屏 (Display data write)	1	<b>8 位显示数据</b>								从 CPU 写数据到 OLED 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OLED屏的显示数据 (Display data read)	1	<b>8 位显示数据</b>								并口时：读已经显示到 OLED 屏上的点阵数据。 <b>串口时不能用此指令。</b>
(8) 显示列地址增减 (ADC select)		1	0	1	0	0	0	0	0	显示列地址增减： <b>1 0xA0</b> : 反转：列地址从右到左， <b>0xA1</b> : 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	显示正显/反显： <b>1 0xA6</b> : 常规：正显 <b>0xA7</b> : 反显
(10)显示全部点阵 (Display all points)	0	1	0	1	0	0	1	0	0	显示全部点阵： <b>1 0xA4</b> : 常规 <b>0xA5</b> : 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	行扫描顺序选择： <b>0XC0</b> :普通扫描顺序：从上到下 <b>0XC8</b> :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： <b>0000-1111</b> ， 参考指令： <b>0Xd5</b> <b>0X80</b>
(13) 电源控制 (Power control set)	0	1	0	0	0	1	1	0	1	设置升压： <b>0X8d</b> <b>0X14</b>
(14)	内部设置OLED电压模式	0	1	0	0	0	0	0	1	设置内部电阻微调，可以理解为 <b>微调</b> 对比度值，此两个指令需紧接着使用。上面一条指令 <b>0x81</b> 是不改的，下面一条指令可设置范围为： <b>0x00~0xFF</b> ，数值越大对比度越浓，越小越淡
	设置的电压值		0	0	<b>6 位电压值数据，0~63 共 64 级</b>					
(15)静态图标显示： 开/关	0	1	0	1	0	1	1	1	0	静态图标的开关设置： <b>1 0xAE</b> : 关， <b>0xAF</b> : 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、背光电源线、I/O 端口 (接口)  
I/O 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:

OLED 模块与 MPU (以 8051 系列单片机为例) 接口图如下:

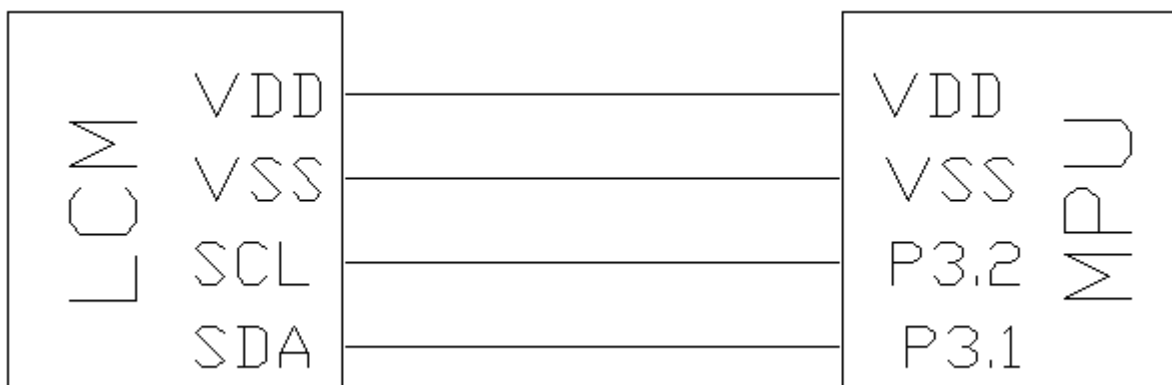


图 5.IIC 接口



### 7.5.1 程序:

#### 点亮液晶模块的编程步骤



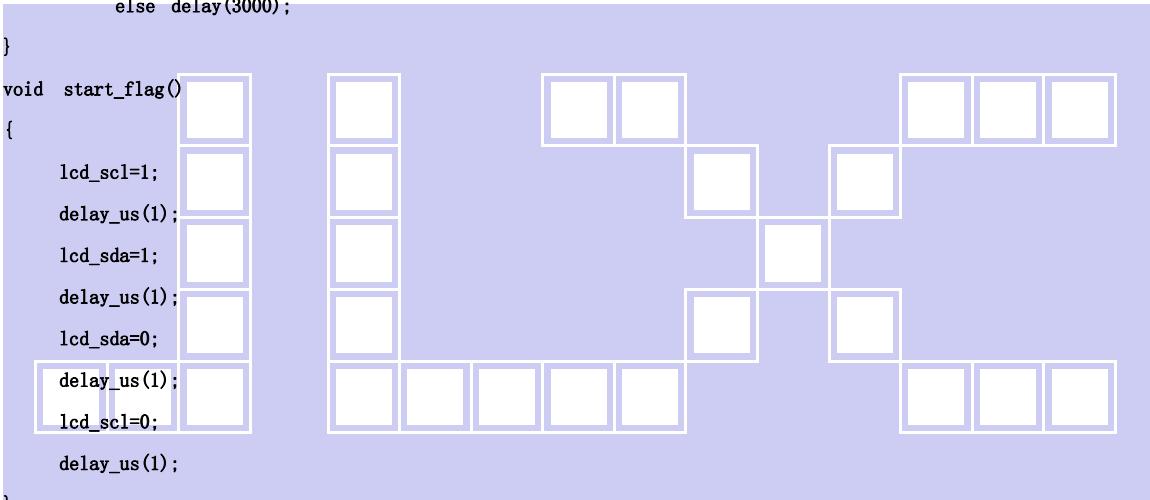
```

// OLED 演示程序
// OLED 模块型号: JLX12864OLED-096X-PN-IIC, IIC 接口!
// 驱动 IC 是:SSD1315
// 资料(源程序、驱动手册、使用说明书等)销售统一发
#include <reg52.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(3000);
}
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;
    lcd_scl=1;
    lcd_scl=0;
    data1<<=1;
//    delay_us(1);
    }
    lcd_sda=0;
    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();
}

```

```

}

```

//OLED 显示模块初始化

```

void initial_lcd()
{

```

```

    delay(2000);

    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);
        }
    }
}

//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); //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 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)

```

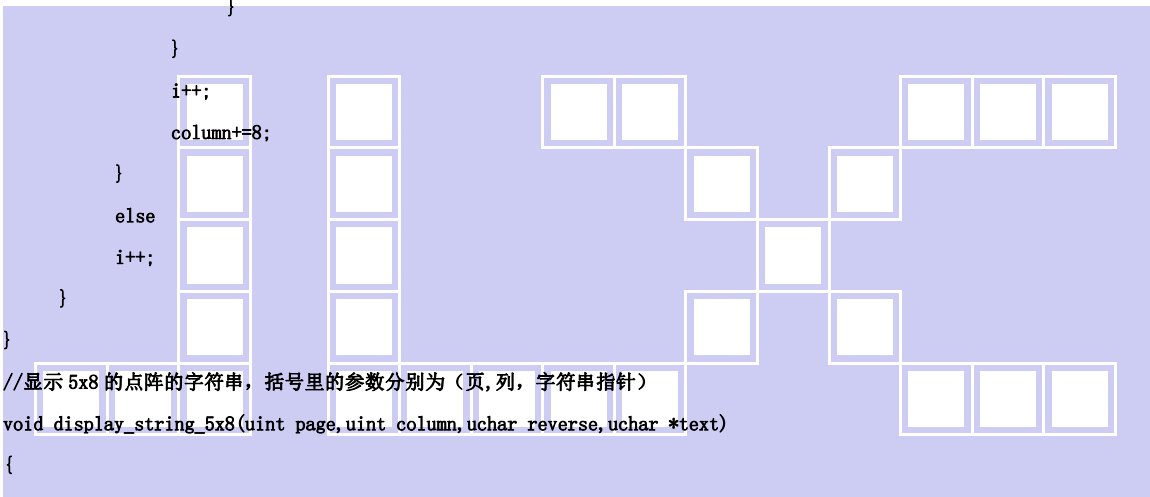
void display_string_8x16(uint page, uint column, uchar *text)
{

```

```

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
            }
        }
    }
}

```



```

//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++;
    }
}
}
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,字符串指针)
        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();
        clear_screen();                //clear all dots
        display_128x64 (bmp1);
        waitkey();
        clear_screen();                //clear all dots
        display_128x64 (bmp2);
        waitkey();
    }
}
}

```





**-END-**

