

# JLX12848G-940-PN 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           | 5     |
| 5  | 技术参数           | 5~6   |
| 6  | 时序特性           | 6~10  |
| 7  | 指令功能及硬件接口与编程案例 | 11~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 JLX12848G-940-PN 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。

JLX12848G-940-PN 可以显示 128 列\*48 行点阵单色图片，或显示 8 个/行\*3 行 16\*16 点阵的汉字，或显示 16 个/行\*6 行 8\*8 点阵的英文、数字、符号。

## 2. JLX12848G-940-PN 图像型点阵液晶模块的特性

2.1 结构牢：背光带有挡墙，焊接式 FPC。

2.2 IC 采用矽创公司 ST7567, 功能强大，稳定性好

2.3 功耗低:1~100mW（关掉背光：[0.3mA@3.3V](#), 打开背光不大于 100mW）；

2.4 显示内容：

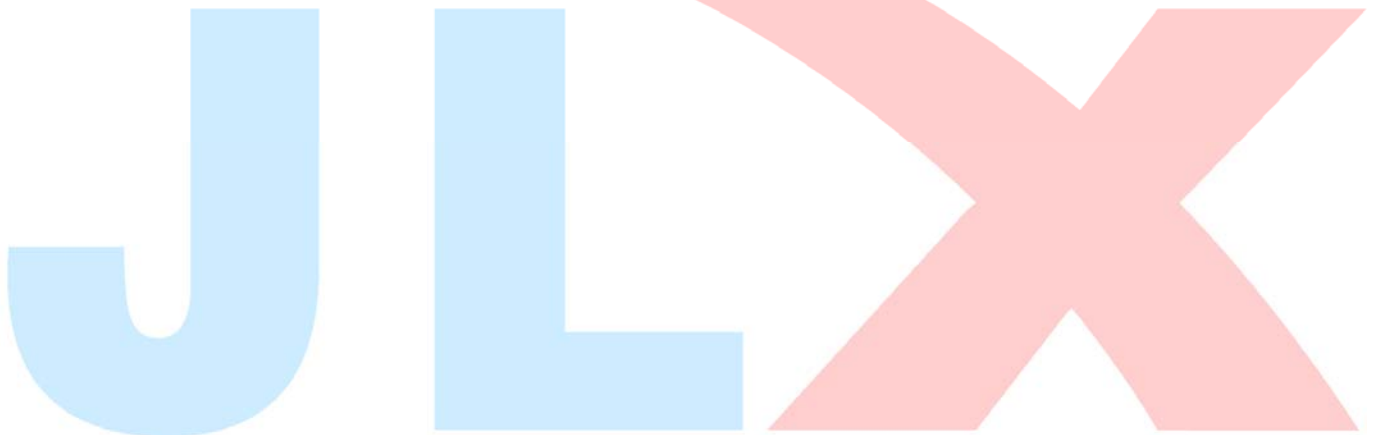
- 128\*48 点阵单色图片；

- 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\*3 行。按照 12\*12 点阵汉字来计算可显示 10 字/行\*3 行。

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）。并口时：可以“读-改-写”；

2.6 接口简单方便:可采用 4 线 SPI 串口，或选择并口（6800 时序和 8080 时序可选）。

2.7 工作温度宽:-20℃ - 70℃，储存温度:-30℃ - 80℃；





模块的接口引脚功能

| 引线号   | 符号      | 名称      | 功能                                 |
|-------|---------|---------|------------------------------------|
| 1     | NC      | 空脚      | 空脚                                 |
| 2     | NC      | 空脚      | 空脚                                 |
| 3     | NC      | 空脚      | 空脚                                 |
| 4     | NC      | 空脚      | 空脚                                 |
| 5     | LEDA    | 背光电源    | 背光电源正极, 同 VDD 电压 (5V 或 3.3V)       |
| 6     | VSS     | 接地      | 0V                                 |
| 7     | VDD     | 电路电源    | 5V 或 3.3V                          |
| 8     | A0 (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") |
| 9     | 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      |
| 10    | CS      | 片选      | 低电平片选                              |
| 11    | D7      | I/O     | 并行时: 数据总线 DB7<br>串行时: 串行数据 (SDA)   |
| 12    | D6      | I/O     | 并行时: 数据总线 DB6<br>串行时: 串行时钟 (SCLK)  |
| 13-18 | D5-D0   | I/O     | 并行时: 数据总线 DB0~DB5<br>串行时: 空        |
| 19    | RD (E)  | 使能信号    | 并行时: 使能信号<br>串行时: 空                |
| 20    | WR      | 读/写     | 并行时: H: 读数据 0: 写数据<br>串行时: 空       |

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 128×48 点阵, 128 个列信号与驱动 IC 相连, 48 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 内部电路框图:

图 2 是 JLX12848G-940-PN 图像点阵型模块的电路框图:

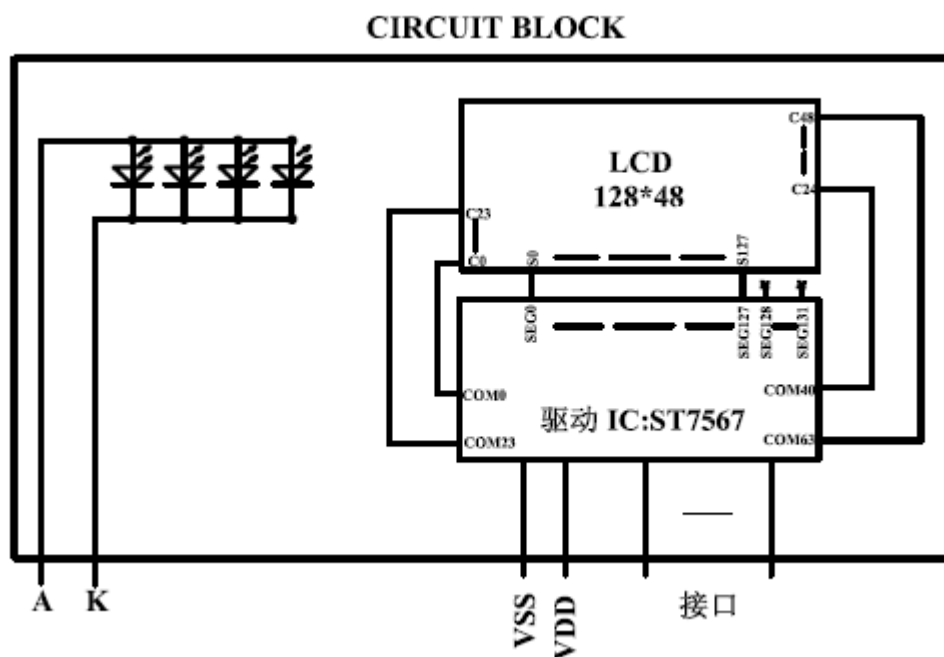


图 2: JLX12848G-940-PN 图像点阵型液晶模块的电路框图

### 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:

背光板可选择绿色、白色。

正常工作电流为: 32~60mA (LED 灯数共 4 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 |    | 3.6  | V  |
| LCD 驱动电压 | V0、VOUT     | -0.3 |    | 13.5 | V  |
| LCD 驱动电压 | V1\V2\V3\V4 | -0.3 |    | V0   | 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              |                          | 2.6     | 3.3  | 3.6     | V  |
| 背光工作电压 | VLED             |                          | 2.9     | 3.0  | 3.1     | 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> = -0.5mA | 0.8xVDD | -    | VDD     | V  |
| 输出低电平  | V <sub>OHC</sub> | I <sub>OL</sub> = -0.5mA | VSS     | -    | 0.2xVDD | V  |
| 模块工作电流 | I <sub>DD</sub>  | VDD = 3.3V               | -       | -    | 0.3     | mA |
| 背光工作电流 | I <sub>LED</sub> | V <sub>LED</sub> =3.0V   | 32      | 60   | 80      | 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 ST7567 (Writing Data from CPU to ST7567)

The 4-line SPI Interface

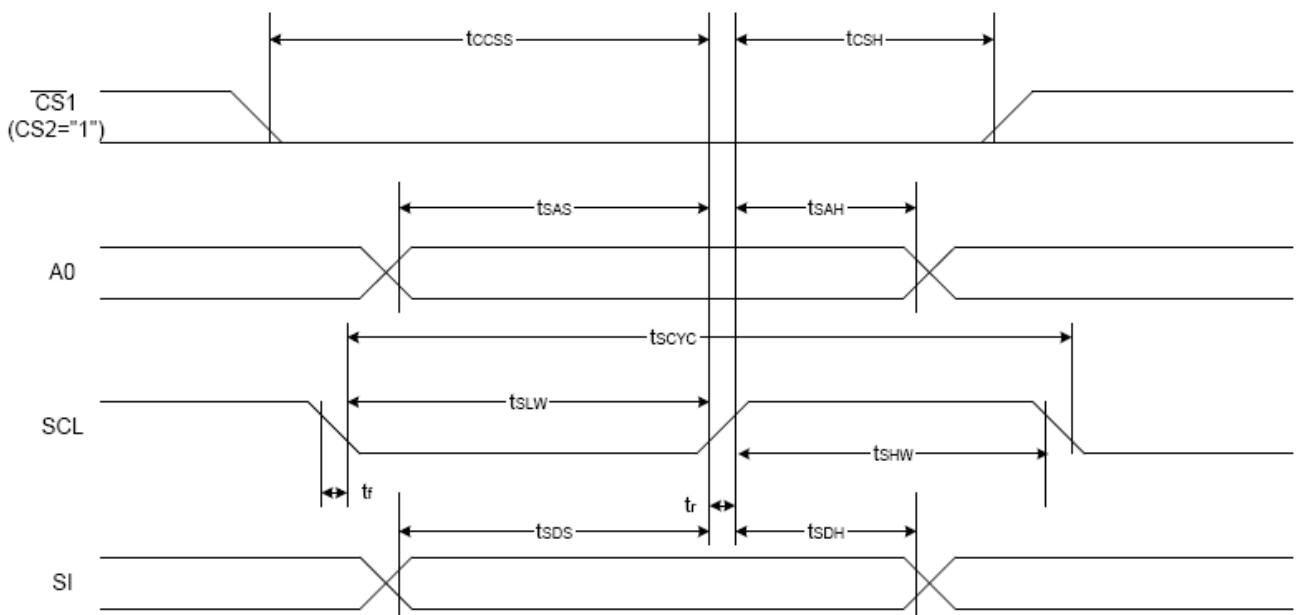


图 4. 从 CPU 写到 ST7567 (Writing Data from CPU to ST7567)

## 6.2 串行接口: 时序要求 (AC 参数):

## 写数据到 ST7567 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<br>(4-line SPI Clock Period) | T <sub>scyc</sub> | 引脚: SCK | 50  | --   | 25  | ns |
| 保持SCK高电平脉宽<br>(SCK "H" pulse width)       | T <sub>shw</sub>  | 引脚: SCK | 25  |      |     | ns |
| 保持SCK低电平脉宽<br>(SCK "L" pulse width)       | T <sub>SLW</sub>  | 引脚: SCK | 25  |      |     | ns |
| 地址建立时间<br>(Address setup time)            | T <sub>SAS</sub>  | 引脚: RS  | 20  | --   | --  | ns |
| 地址保持时间<br>(Address hold time)             | T <sub>sah</sub>  | 引脚: RS  | 10  | --   | --  | ns |
| 数据建立时间<br>(Data setup time)               | T <sub>sds</sub>  | 引脚: SI  | 20  | --   | --  | ns |
| 数据保持时间<br>(Data hold time)                | T <sub>SDH</sub>  | 引脚: SI  | 10  | --   | --  | ns |
| 片选信号建立时间<br>(CS-SCL time)                 | T <sub>css</sub>  | 引脚: CS  | 20  |      |     | ns |
| 片选信号保持时间<br>(CS-SCL time)                 | T <sub>csh</sub>  | 引脚: CS  | 40  |      |     | ns |

VDD = 3.0V ± 5%, T<sub>a</sub> = 25°C

6.3 并行接口:

从 CPU 写到 ST7567 (Writing Data from CPU to ST7567)

System Bus Read/Write Characteristics 1 (For the 8080 Series MPU)

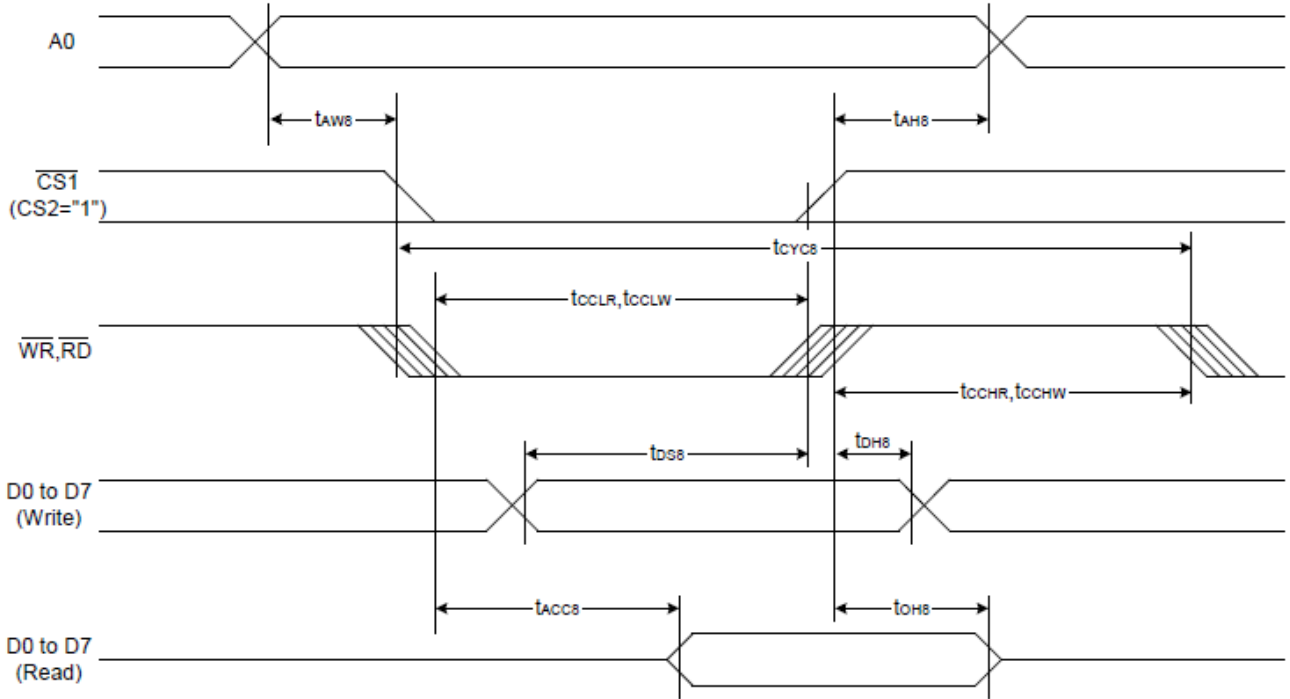


图 5. 从 CPU 写到 ST7567 (Writing Data from CPU to ST7567)

System Bus Read/Write Characteristics 2 (For the 6800 Series MPU)

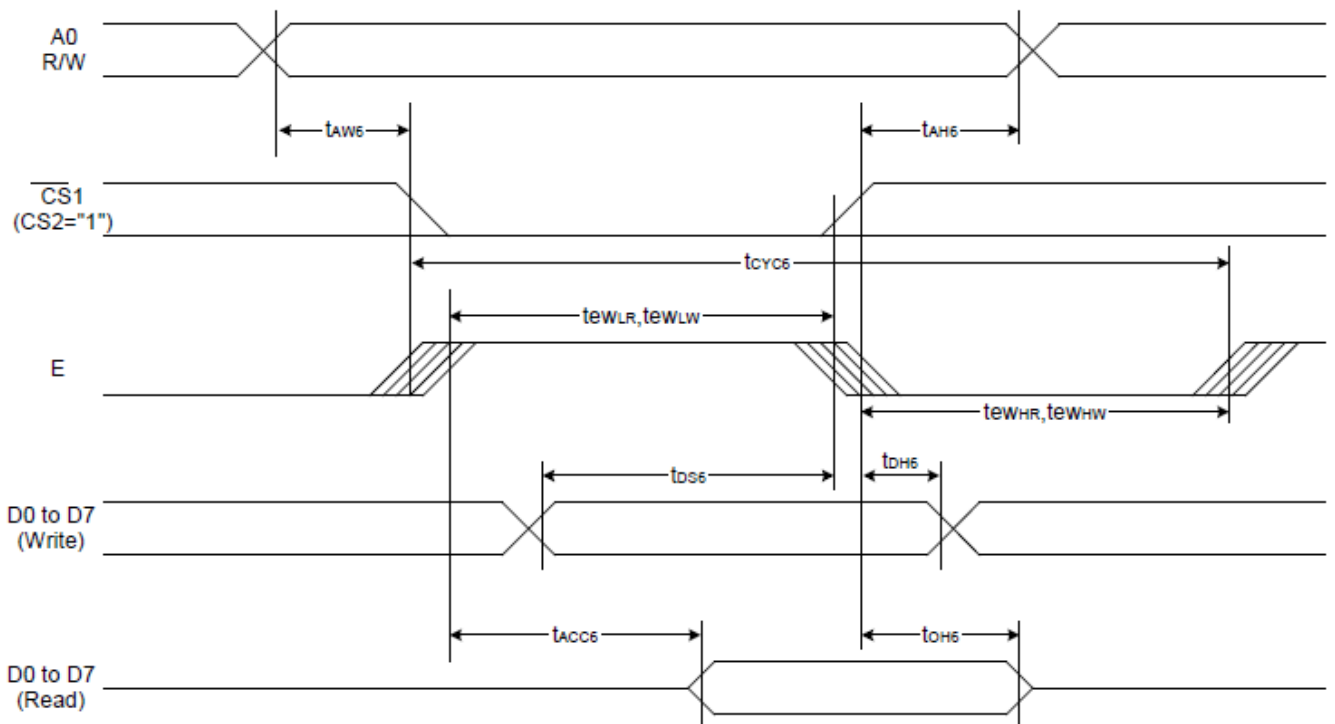


图 6. 从 CPU 写到 ST7567 (Writing Data from CPU to ST7567)



**6.4 并行接口：时序要求（AC 参数）：  
写数据到 ST7567 的时序要求：（8080 系列 MPU）**

| 项目         | 符号    | 测试条件  | 极限值 |      |     | 单位 |
|------------|-------|-------|-----|------|-----|----|
|            |       |       | MIN | TYPE | MAX |    |
| 地址保持时间     | A0    | tAH8  | 0   | --   | --  | ns |
| 地址建立时间     |       | tAW8  | 0   |      | --  | ns |
| 系统循环时间     |       | tCYC8 | 240 |      | --  | ns |
| 使能“低”脉冲（写） | WR    | tCCLW | 80  | --   | --  | ns |
| 使能“高”脉冲（写） |       | tCCHW | 80  | --   | --  | ns |
| 使能“低”脉冲（读） | RD    | tCCLR | 140 | --   | --  | ns |
| 使能“高”脉冲（读） |       | tCCHR | 80  | --   |     | ns |
| 写数据建立时间    | D0-D7 | tDS8  | 40  |      | --  | ns |
| 写数据保持时间    |       | tDH8  | 0   |      | --  |    |
| 读时间        |       | tACC8 | --  |      | 70  |    |
| 读输出来允许时间   |       | tOH8  | 5   |      | 50  | ns |

**写数据到 ST7567 的时序要求：（6800 系列 MPU）**

| 项目         | 符号    | 测试条件  | 极限值 |      |     | 单位 |
|------------|-------|-------|-----|------|-----|----|
|            |       |       | MIN | TYPE | MAX |    |
| 地址保持时间     | A0    | tAH6  | 0   | --   | --  | ns |
| 地址建立时间     |       | tAW6  | 0   |      | --  | ns |
| 系统循环时间     |       | tCYC6 | 240 |      | --  | ns |
| 使能“低”脉冲（写） | WR    | tEWLW | 80  | --   | --  | ns |
| 使能“高”脉冲（写） |       | tEWHW | 80  | --   | --  | ns |
| 使能“低”脉冲（读） | RD    | tEWLR | 80  | --   | --  | ns |
| 使能“高”脉冲（读） |       | tEWHR | 140 | --   |     | ns |
| 写数据建立时间    | D0-D7 | tDS6  | 40  |      | --  | ns |
| 写数据保持时间    |       | tDH6  | 0   |      | --  |    |
| 读时间        |       | tACC6 | --  |      | 70  |    |
| 读输出来允许时间   |       | tOH6  | 5   |      | 50  | ns |

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

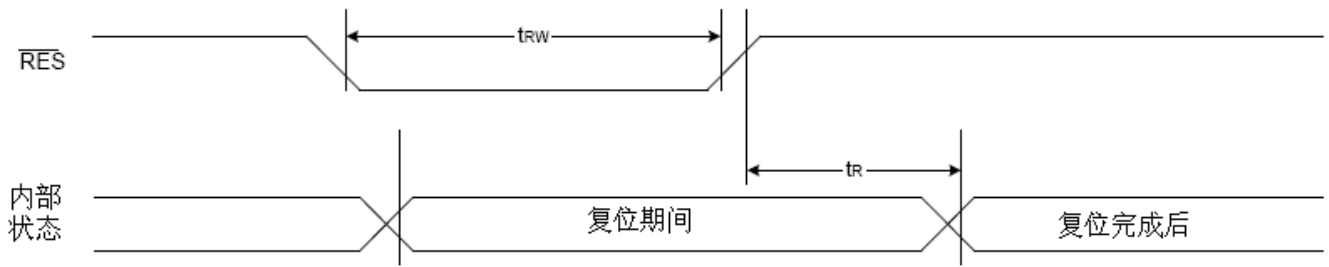
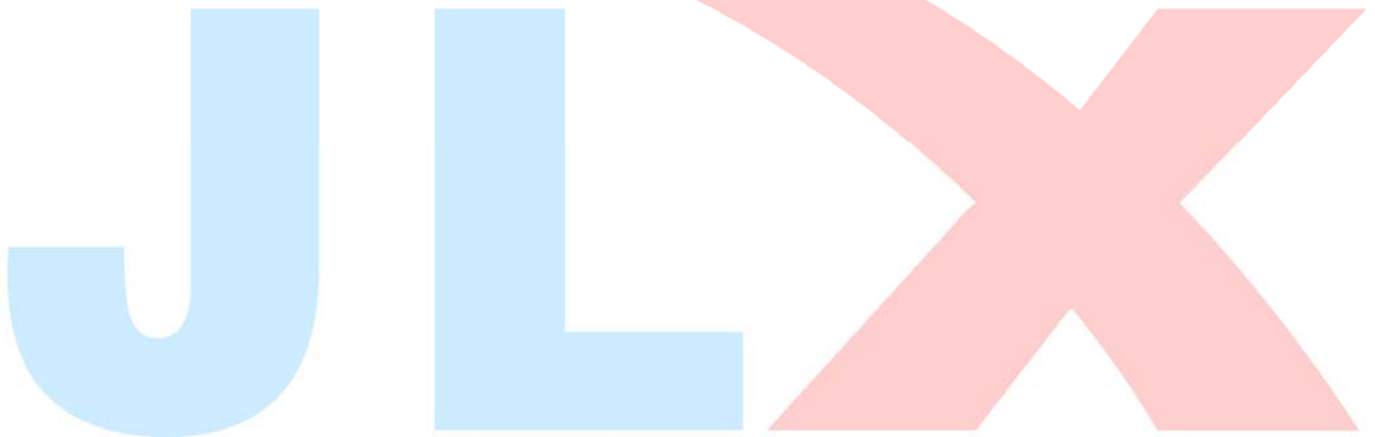


图 7：电源启动后复位的时序

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       | tr  |        | --  | --   | 3.0 | us |
| 复位保持低电平的时间 | trw | 引脚：RES | 3.0 | --   | --  | us |



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<br>(display on/off)           | 0   | 1              | 0   | 1                     | 0   | 1                   | 1   | 1   | 0<br>1 | 显示开/关:<br><b>0XAE</b> :关, <b>0XAF</b> : 开   |  |
| (2)显示初始行设置<br>(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<br>(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<br>列地址低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。比如液晶模块的第 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>                           |  |
|  |     | 0              | 0   | 0                     | 0   | <b>列地址的低 4 位</b>    |     |     |        |   |  |
| (5) 读状态<br>(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<br>(Display data write)   | 1   | <b>8 位显示数据</b> |     |                       |     |                     |     |     |        |   | 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<br>(Display data read)    | 1   | <b>8 位显示数据</b> |     |                       |     |                     |     |     |        |   | 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<br>(ADC select)            |     | 1              | 0   | 1                     | 0   | 0                   | 0   | 0   | 0<br>1 | 显示列地址增减:<br><b>0xA0</b> : 常规: 列地址从左到右,<br><b>0xA1</b> : 反转: 列地址从右到左   |  |
| (9)显示正显/反显<br>(Display normal/reverse) | 0   | 1              | 0   | 1                     | 0   | 0                   | 1   | 1   | 0<br>1 | 显示正显/反显:<br><b>0xA6</b> : 常规: 正显<br><b>0xA7</b> : 反显  |  |
| (10)显示全部点阵<br>(Display all points)     | 0   | 1              | 0   | 1                     | 0   | 0                   | 1   | 0   | 0<br>1 | 显示全部点阵:<br><b>0xA4</b> : 常规<br><b>0xA5</b> : 显示全部点阵   |  |
| (11)LCD 偏压比设置<br>(LCD bias set)        | 0   | 1              | 0   | 1                     | 0   | 0                   | 0   | 1   | 0<br>1 | 设置偏压比:<br><b>0XA2</b> : BIAS=1/9 (常用)<br><b>0XA3</b> : BIAS=1/7   |  |
| (12) 读-改-写<br>(Read-modify-write)      | 0   | 1              | 1   | 1                     | 0   | 0                   | 0   | 0   | 0      | <b>0XE0</b> : “读-改-写” 开始。<br>列地址的增加:<br>写入时: 列地址+1<br>读出时: 列地址不加<br><a href="#">详情请参考IC资料第43-44页</a>                                |  |
| (13) 退出上述“读-改-写”指令(End)                | 0   | 1              | 1   | 1                     | 0   | 1                   | 1   | 1   | 0      | <b>0XEE</b> :上述“读-改-写”指令结束<br><a href="#">详情请参考 IC 资料第 43-44 页</a>  |  |
| (14) 软件复位 (Reset)                      | 0   | 1              | 1   | 1                     | 0   | 0                   | 0   | 1   | 0      | <b>0XE2</b> :软件复位。  |  |

|   |            |   |   |   |                         |   |   |                     |   |   |  |
|---|------------|---|---|---|-------------------------|---|---|---------------------|---|---|--|
| (15) 行扫描顺序选择<br>(Common output mode select) |            |   | 1 | 1 | 0                       | 0 | 0 | 0                   | 0 | 0 | 行扫描顺序选择：<br><b>0XC0</b> :普通扫描顺序：从上到下<br><b>0XC8</b> :反转扫描顺序：从下到上   |
| (16) 电源控制<br>(Power control set)            |            |   | 0 | 0 | 1                       | 0 | 1 | <b>电压操作模式选择，共3位</b> |   |   | 选择内部电压供应操作模式：<br>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)。<br>通常是 <b>0x2C,0x2E,0x2F</b> 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 <b>0x2F</b> ，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 | <b>内部电压值电阻设置</b>    |   |   | 选择内部电阻比例 (Rb/Ra) :可以理解为 <b>粗调</b> 对比度值。可设置范围为： <b>0x20~0x27</b> ，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 <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>0x00~0x3F</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> |   |   |                     |   |   |  |
| (19)静态图标显示：<br>开/关                          |            | 0 | 1 | 0 | 1                       | 0 | 1 | 1                   | 0 | 0 | 静态图标的开关设置：<br><b>0xAC</b> : 关, <b>0xAD</b> : 开。<br>此指令在进入及退出睡眠模式时起作用   |
| (20) 升压倍数选择<br>(Booster ratio set)          |            | 0 | 1 | 1 | 1                       | 1 | 1 | 0                   | 0 | 0 | 选择升压倍数：<br><b>00</b> : 2 倍, 3 倍, 4 倍<br><b>01</b> : 5 倍<br><b>11</b> : 6 倍。本模块外部已设置升压倍数为 4 倍，不必使用此指令   |
| (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 47 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！  |

请详细参考 IC 资料”ST7567.PDF”

### 7.3 点阵与 DD RAM(显示数据存储器)地址的对应关系

请注意页的定义：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\*48 点阵的屏分为 6 个“页”，从第 0“页”到第 5“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：

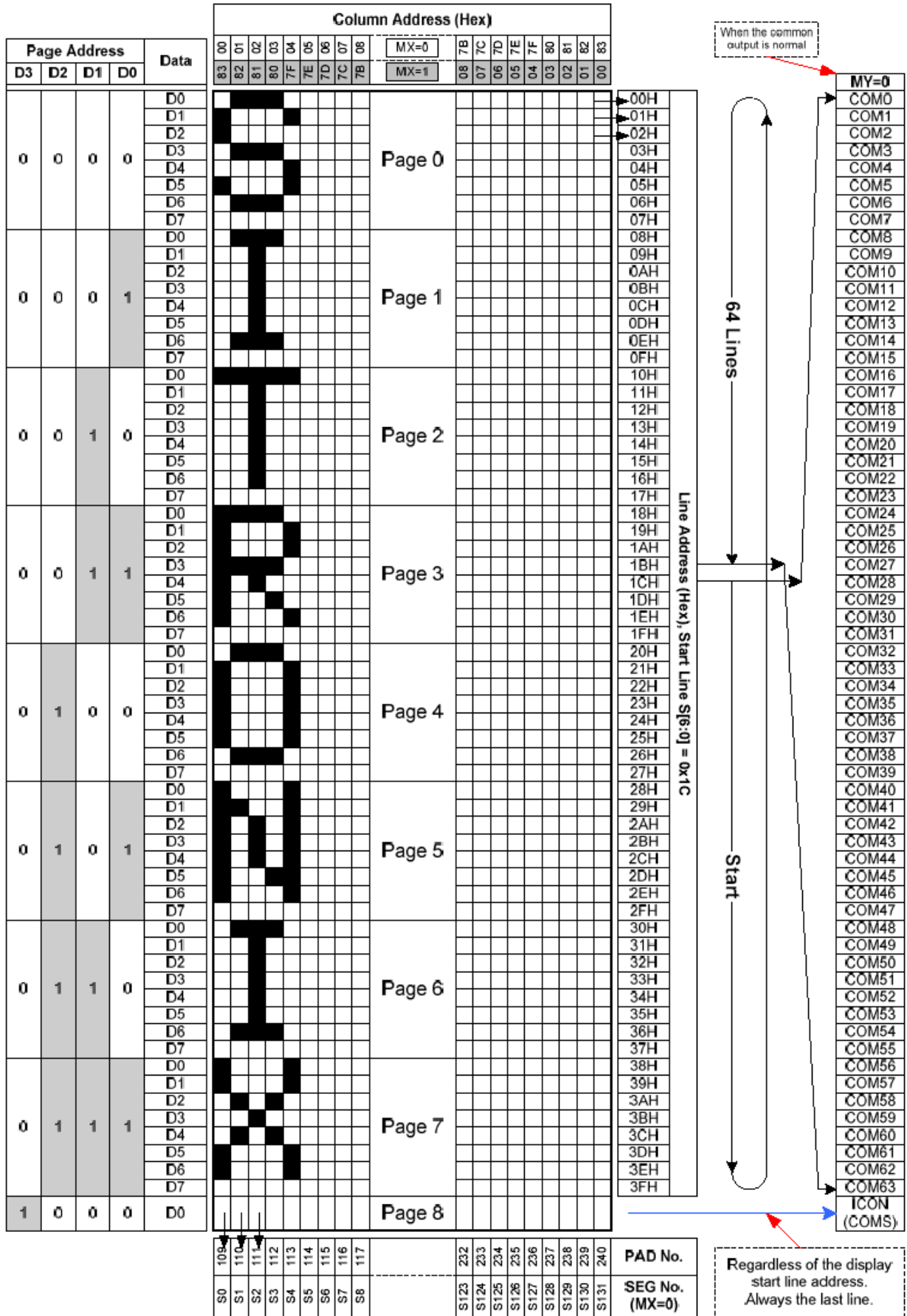
|    |   |   |   |   |  |   |
|----|---|---|---|---|--|---|
| D0 | 0 | 1 | 1 | 1 |  | 0 |
| D1 | 1 | 0 | 0 | 0 |  | 0 |
| D2 | 0 | 0 | 0 | 0 |  | 0 |
| D3 | 0 | 1 | 1 | 1 |  | 0 |
| D4 | 1 | 0 | 0 | 0 |  | 0 |
| -  |   |   |   |   |  |   |

Display data RAM  
(显示数据存储器的)

|      |  |  |  |  |  |  |
|------|--|--|--|--|--|--|
| COM0 |  |  |  |  |  |  |
| COM1 |  |  |  |  |  |  |
| COM2 |  |  |  |  |  |  |
| COM3 |  |  |  |  |  |  |
| COM4 |  |  |  |  |  |  |
| -    |  |  |  |  |  |  |

Liquid crystal display  
(液晶屏)

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。下图摘自 ST7567 IC 资料，可通过“ST7567.PDF”获取最佳效果。



## 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、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:

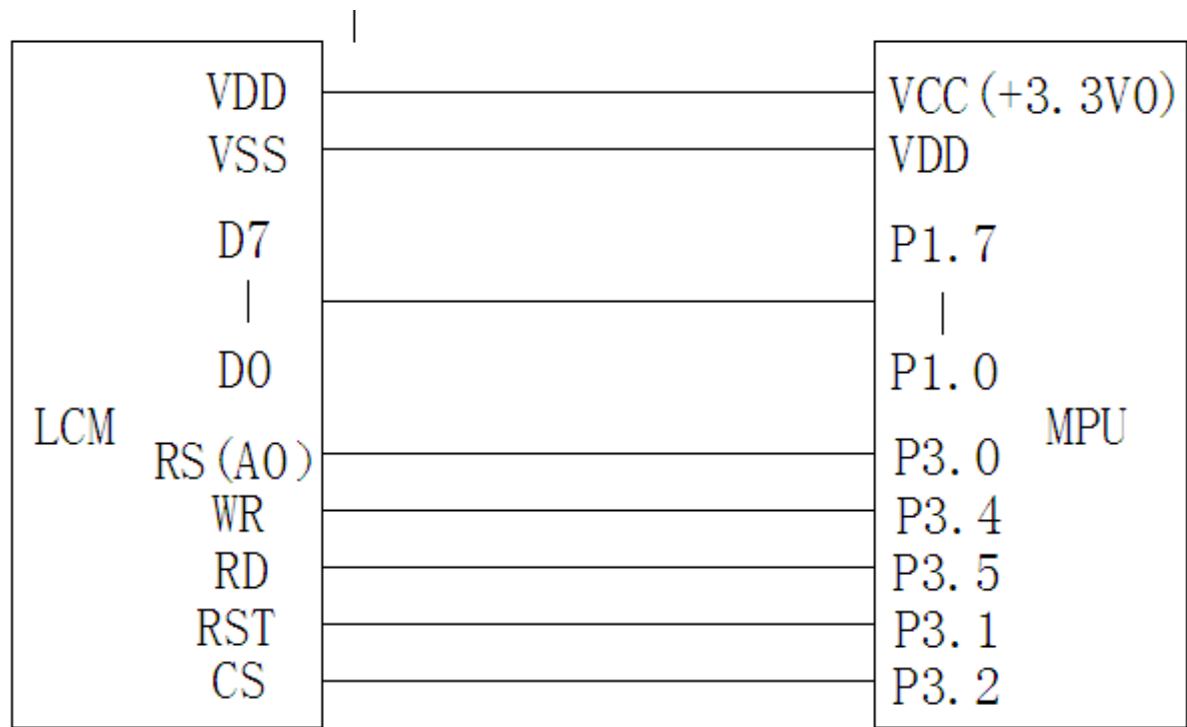
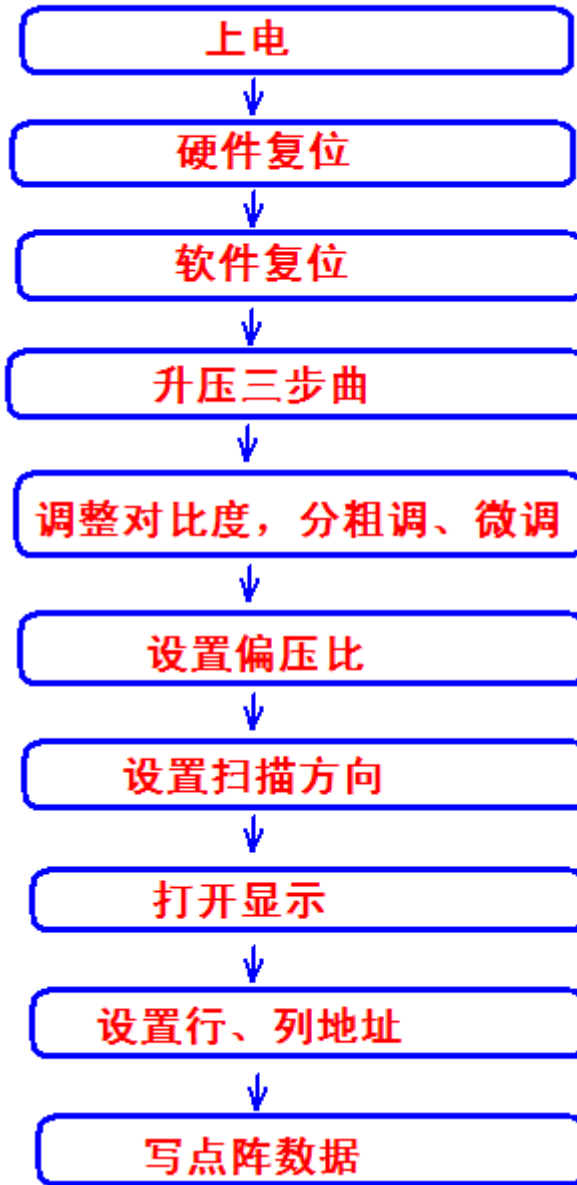


图 8. 并行接口



7.51、程序

**点亮液晶模块的编程步骤**



**以下为并行方式的例程序**

```

/* Test program for JLX12848G-940-PN-P, 并行接口
   驱动 IC 是:ST7567(or compatible)
   晶联讯电子: 网址 http://www.jlxlcd.cn; http://www.jlxlcd.com.cn
*/
#include <reg51.h>
#include <intrins.h>
#include <Ctype.h>
#include <math.h>
#include <chinese.h>

sbit rw=P3^4; //接口定义:lcd_rw 就是 LCD 的 wr,有些图纸上写“WR”
sbit e=P3^5; //接口定义:lcd_e 就是 LCD 的 rd
sbit rs=P3^0; //接口定义:lcd_rs 就是 LCD 的 rs,有些图纸上写“A0”\“DC”\“DC0”,都是它
sbit cs1=P3^2; //接口定义:lcd_cs1 就是 LCD 的 cs1
sbit reset=P3^1; //接口定义:lcd_reset 就是 LCD 的 reset
sbit key=P2^0; //P2.0 口与 GND 之间接一个按键
  
```

```
/*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++);
}

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

//写指令到 LCD 模块
void transfer_command(int data1)
{
    cs1=0;
    rs=0;
    rw=0;
    P1=data1;
    e=1;
    delay_us(1);
    e=0;
    cs1=1;
    e=0;
}

//写数据到 LCD 模块
void transfer_data(int data1)
{
    cs1=0;
    rs=1;
    rw=0;
    P1=data1;
    e=1;
    delay_us(1);
    e=0;
    cs1=1;
    e=0;
}

/*LCD 模块初始化*/
//=====微调对比度设置值 0x18=====
void initial_lcd()
{
    reset=0;        /*低电平复位*/
    delay(200);
    reset=1;        /*复位完毕*/
    delay(50);
    transfer_command(0xe2); /*软复位*/
    delay(5);
    transfer_command(0x2c); /*升压步聚 1*/
    delay(5);
    transfer_command(0x2e); /*升压步聚 2*/
}
```

```

delay(5);
transfer_command(0x2f); /*升压步聚 3*/
delay(5);

transfer_command(0x24); /*粗调对比度, 可设置范围 0x20~0x27*/
transfer_command(0x81); /*微调对比度*/
transfer_command(0x18); /*微调对比度的值, 可设置范围 0x00~0x3f*/
transfer_command(0xa2); /*1/8 偏压比 (bias) */

transfer_command(0xc8); /*行扫描顺序: 从上到下*/
transfer_command(0xa0); /*列扫描顺序: 从左到右*/

transfer_command(0x40); /*起始行: 第一行开始*/
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 48 行被分成 6 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
}

/*
void lcd_address(uchar page,uchar column)
{
    column=column+3; //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 48 行被分成 6 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<9;i++)
    {
        lcd_address(1+i,1);
        for(j=0;j<132;j++)
        {
            transfer_data(0x00);
        }
    }
}

//===显示测试画面: 例如全显示, 隔行显示, 隔列显示, 雪花显示=====
void test_display(uchar data1,uchar data2)
{
    int i,j;
    for(j=0;j<6;j++)
    {
        lcd_address(j+1,0);
        for(i=0;i<128;i++)
        {
            transfer_data(data1);
            transfer_data(data2);
        }
    }
}

```

```

    }
}

void display_graphic(uchar *dp)
{
    uchar i, j;
    for(j=0; j<6; j++)
    {
        lcd_address(j+1, 1);
        for(i=0; i<128; i++)
        {
            transfer_data(*dp);
            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<31; 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 reverse, uchar *dp)
{
    uchar i, j;
    for(j=0; j<2; j++)
    {
        lcd_address(page+j, column);
        for (i=0; i<16; i++)
        {
            if (reverse==1)
            {
                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*/
            }
            else
                transfer_data(*dp);
            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++;
        }
    }
}

```

```
//显示一串 8x16 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_8x16(uint page,uint column,uchar reverse,uchar *text)
{
```

```
    uint i=0, j, k, n, data1;
    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++)
                {
                    if(reverse==1) data1=~ascii_table_8x16[j][k+8*n];
                    else data1=ascii_table_8x16[j][k+8*n];
                    transfer_data(data1);
                }
            }
            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++;
    }
}
```

```
/**从液晶屏驱动 IC 中读取数据（1 个字节）**/
```

```
uchar read_data ()
{
    uchar ret_data=0;
    P1=0xff;
    rw=1;
    rs=1;
    cs1=0;
    e=0;
```

```

    e=1;
    delay_us(1);
    ret_data=P1;
    cs1=1;
    return(ret_data);
}

```

//===开始 “读取-修改-写入” 模式===

```

void Start_Read_Modify_Write()
{
    transfer_command(0xe0);
}

```

//===结束 “读取-修改-写入” 模式===

```

void End_Read_Modify_Write()
{
    transfer_command(0xee);
}

```

//从液晶屏驱动 IC 中读取数据, 含一次空读和一次正式读取

```

uchar read_data_twice()
{

```

```

    uchar ret_data=0;
    P1=0xff;
    cs1=0;
    rs=1;
    rw=1;

```

```

//一次空读-----
    e=0;
    e=1;
    delay_us(1);

```

```

//---一次正式读-----
    e=0;
    e=1;
    delay_us(1);

```

```

//
    ret_data=P1;
    cs1=1;
    e=0;
    return(ret_data);
}

```

//画点函数。括号里的参数分别为：坐标(column, row), row 为行, 共 48 行, 最小值为 1, 最大值为 48; column 为列, 共 128 列, 最小值为 1, 最大值为 128。

```

void draw_point(uint column, uint row)
{

```

```

    uchar i, dat, page;
    row--;

```

```

    column--; //我们平常的“第 1 行”在液晶屏里叫“第 0 行”，所以减 1，同理，列也如此

```

```

    page=row/8;

```

```

    page++;

```

```

    lcd_address(page, column); //设置地址

```

```

    Start_Read_Modify_Write(); //开始 “读取-修改-写入” 模式

```

```

    i=row%8; //行位置分成了第几页和这一页内的第几位, i=y%8 即是这个第几位

```

```

    dat=1;

```

```

    dat<<=i; //用移位的方法, 根据这个“第几位”算出这个字节的数值来, 这个就是我们要画的“点”的

```

一个字节数据

```

    dat|=read_data_twice(); //要画点的数据和刚读到的原来液晶屏的位置上的数据进行“或”运算, 以免覆盖原来的数据。

```

```

    transfer_data(dat);
}

```

```

End_Read_Modify_Write(); //结束 “读取-修改-写入” 模式
}

// “画直线” 函数的注意事项:
//直线的斜率  $k=(y_2-y_1)/(x_2-x_1)$ ,  $k \setminus x_1 \setminus y_1 \setminus x_2 \setminus y_2$  必须是浮点型的数据, 除此之外, 由于终点坐标可能小于起点坐标, 所以斜率还有可能是负数的。
//由于  $k=(y_2-y_1)/(x_2-x_1)$ , 所以在  $x_2=x_1$  时, 导致除数为 0, 数学上这是行不通的, 所以另外想办法。同理  $k_y=(x_2-x_1)/(y_2-y_1)$  当  $y_2=y_1$  时也是行不通的。
//另外 int 是 16 位的整数, uchar 却只有 8 位。
//
void draw_line(float x1,float y1,float x2, float y2)
{
    int i;float k,k_y,x,z;
    if( (y2-y1)==0&&(x2-x1)!=0 ) //如果 y2-y1=0 且 x2-x1 不等于 0, 则画一条横线。
    {
        if(x2<x1) //如果 x2<x1, 则两个坐标互换。画横线时, 从左到右与从右到左是一样的结果。
        {
            z=x2; x2=x1;x1=z;
        }
        for(i=0;i<=(x2-x1);i++)
        {
            draw_point((x1+i),y1);
        }
    }
    else if( ((x2-x1)==0)&&((y2-y1)!=0 ) //如果 x2-x1=0 且 y2-y1 不等于 0, 则画一条竖线。
    {
        if(y2<y1) //如果 y2<y1, 则两个坐标互换。画竖线时, 从上到下与从下到上是一样的结果。
        {
            z=y2; y2=y1;y1=z;
        }
        for(i=0;i<=(y2-y1);i++)
        {
            draw_point(x1,(y1+i));
        }
    }
    else if((x2-x1)==0&&(y2-y1)==0) //如果 (x2=x1)且(y2=y1), 画一个点即可
    {
        draw_point(x1,y2);
    }
    else //否则, 画斜线
    {
        if(x2<x1) //如果 x2<x1, 则两个坐标互换。(从起点画到终点)与(从终点画到起点)结果是一样的。
        {
            z=y2; y2=y1;y1=z;
            z=x2; x2=x1;x1=z;
        }
        else; //

        if(fabs(y2-y1)<=fabs(x2-x1)) //如果(y2-y1)的绝对值小于等于(x2-x1)的绝对值, 就启动方案一: x 逐点扫描, y 按斜率计算, 然后画点 (x+i, y)
        {
            k=(y2-y1)/(x2-x1); //k 是斜率
            for(i=0;i<=(x2-x1);i++)
            {
                draw_point((x1+i),(y1+k*i));
            }
        }
        else //如果(y2-y1)的绝对值大于(x2-x1)的绝对值, 就启动方案二: y 逐点扫描, x 按斜率计算, 然后画点 (x, y+i) 或 (x, y-i)
        {
            k_y=fabs((x2-x1)/(y2-y1)); //k_y 是反斜率 (即 x 除以 y)。fabs 是浮点型数据的绝对值
            for(i=0;i<=fabs(y2-y1);i++)

```

```

        {
            x=x1+k_y*i;
            if((y2-y1)>0)
            {
                draw_point(x, (y1+i));
            }
            else
            {
                draw_point(x, (y1-i));
            }
        }
    }
}

void sleep()
{
    transfer_command(0xac); /*静态图标关闭*/
    transfer_command(0x00); /*静态图标寄存器设置：关闭。此指令与上述指令一起完成静态图标关闭*/
    transfer_command(0xae); /*显示：关*/
    transfer_command(0xa5); /*全屏显示：开*/
}

void wake()
{
    transfer_command(0xa4); /*全屏显示：关。进入正常模式*/
    transfer_command(0xad); /*静态图标开启*/
    transfer_command(0x03); /*静态图标寄存器设置：开。此指令与上述指令一起完成静态图标开启*/
    transfer_command(0xaf); /*显示：开*/
}

void main(void)
{
    initial_lcd();
    while(1)
    {
        clear_screen();
        draw_point(4, 2); //画点
        draw_point(10, 2);
        draw_point(16, 2);
        draw_point(22, 2);
        draw_point(28, 2);
        draw_point(34, 2);
        draw_point(40, 2);
        draw_point(46, 2);
        draw_point(52, 2);
        draw_point(58, 2);
        draw_point(64, 2);
        draw_point(70, 2);
        draw_point(76, 2);
        draw_point(82, 2);
        draw_point(88, 2);
        draw_point(94, 2);
        draw_point(100, 2);
        draw_point(106, 2);
        draw_point(112, 2);
        draw_point(118, 2);
        draw_point(124, 2);

        draw_line(56, 6, 72, 6); //画线
        draw_line(48, 9, 80, 9);
        draw_line(40, 12, 88, 12);
        draw_line(32, 15, 96, 15);
        draw_line(24, 18, 104, 18);

        display_graphic_16x16(4, 17, 1, hua);
    }
}

```



```

display_graphic_16x16(4, 33, 1, dian);
draw_line(16, 23, 16, 42);           //画矩形，画 4 条线实现
draw_line(16, 23, 51, 23);
draw_line(51, 23, 51, 42);
draw_line(16, 42, 51, 42);

display_graphic_16x16(4, 80, 1, hua);
display_graphic_16x16(4, 96, 1, xian);
draw_line(79, 23, 79, 42);           //画矩形，画 4 条线实现
draw_line(79, 23, 114, 23);
draw_line(114, 23, 114, 42);
draw_line(79, 42, 114, 42);
waitkey();

clear_screen();
display_graphic(bmp2);
waitkey();

clear_screen();
display_graphic(bmp1);
waitkey();

clear_screen();                               //clear all dots
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 ");
waitkey();

clear_screen();
display_graphic_32x32(1, 33, cheng1);           //在第 1 页，第 49 列显示单个汉字“成”*/
display_graphic_32x32(1, 65, gong);             //在第 1 页，第 49 列显示单个汉字“成”*/
display_graphic_16x16(5, 1, 1, zhuang1);        //在第 5 页，第 1 列显示单个汉字“状”
display_graphic_16x16(5, (1+16), 1, tail);     //在第 5 页，第 17 列显示单个汉字“态”
display_string_8x16(5, 33, 0, ":");            //在第 1 页，第 1 列显示字符串
display_graphic_16x16(5, 41, 0, shi1);         //在第 5 页，第 41 列显示单个汉字“使”
display_graphic_16x16(5, (1+16*3+8), 0, yong1); //在第 5 页，第 49 列显示单个汉字“用”
display_string_8x16(5, 89, 0, "00:00");        //显示 8x16 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）
waitkey();

clear_screen();                               //clear all dots
display_string_8x16(1, 1, 0, "0123456789abcdef"); //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_8x16(3, 1, 0, "~!@#%&^*()_+="); //同上
display_string_5x8(5, 1, 1, " ! # $ % & ' ( ) * + , - . / 0 1 2 3 4");
display_string_5x8(6, 1, 0, "56789:;<=>?@ABCDEFGHI");

waitkey();
test_display(0xff, 0xff);
waitkey();
test_display(0xaa, 0x55);
waitkey();
test_display(0x55, 0xaa);
waitkey();
}
}

```

串行接口：

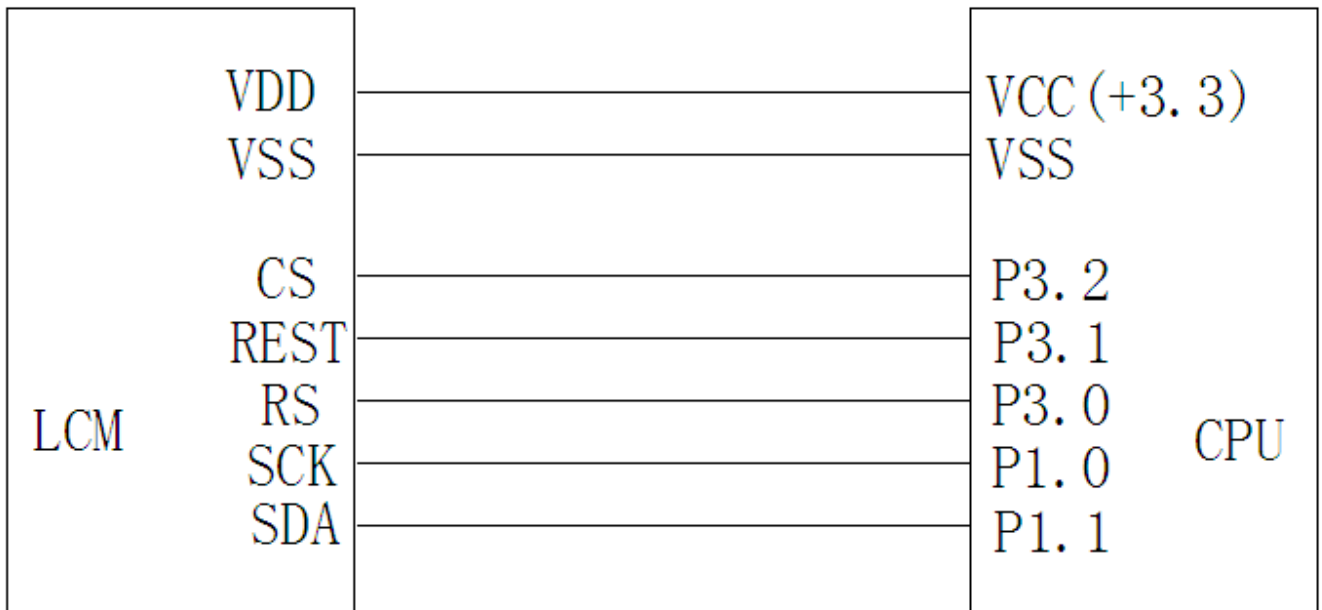


图 9. 串行接口

7.52、以下为串行方式例程序  
与并行方式相比较，只需改变传送数据和命令子程序即可

```
#include <reg51.h>
#include <intrins.h>
#include <ctype.h>
#include <math.h>
#include <chinese.h>
```

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

```
/*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;
//        delay(1);
        sclk=1;
//        delay(1);
        data1=data1<<=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;
    }
}
```