

JLX12864G-542-BN 使用说明书

(焊接式 FPC)

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 | 6~9 |
| 7 | 指令功能及硬件接口与编程案例 | 10~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 JLX12864G-542 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。

JLX12864G-542 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. JLX12864G-542 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: 1~300mW (关掉背光: [0.3mA@3.3V](#), 打开背光不大于 300mW) ;

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。

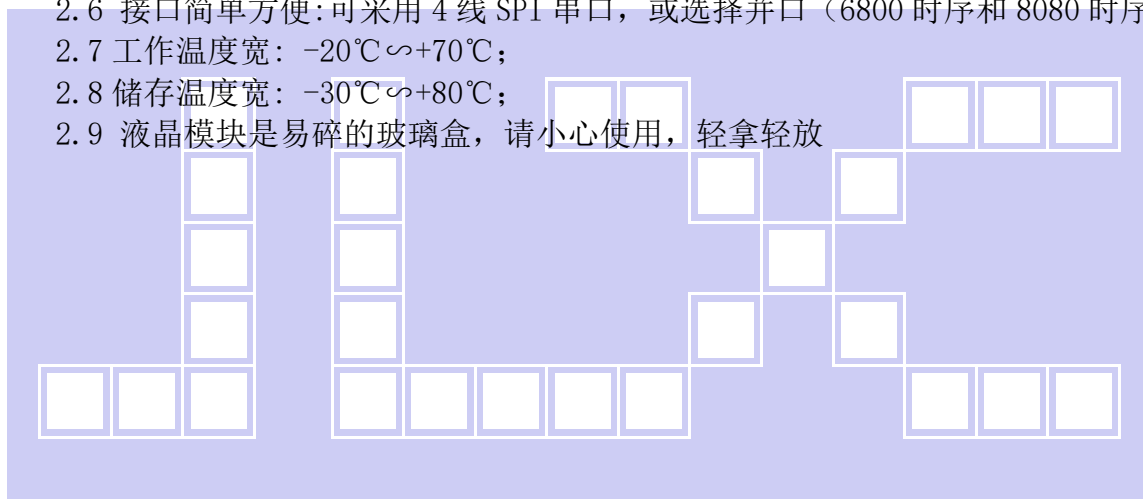
2.5 指令功能强: 可软件调对比度、正显/反显转换、行列扫描方向可改 (可旋转 180 度使用)。并口时: 可以“读-改-写”;

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: $-20^{\circ}\text{C} \sim +70^{\circ}\text{C}$;

2.8 储存温度宽: $-30^{\circ}\text{C} \sim +80^{\circ}\text{C}$;

2.9 液晶模块是易碎的玻璃盒, 请小心使用, 轻拿轻放



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 | CS | 片选 | 低电平片选 |
| 4 | 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 |
| 5 | RS | 寄存器选择信号 | H:数据寄存器 0:指令寄存器 |
| 6 | R/W(/WR) | 6800 时序:读/写 8080 时序: 写 | 并行接口时并且选择 6800 时序时: H:读数据 L:写数据 并行接口时并且选择 8080 时序时: 写数据, 低电平有效. 串行接口时: 接 VDD 或悬空 |
| 7 | E(/RD) | 6800 时序:使能 8080 时序: 读 | 并行接口时并且选择 6800 时序时: 使能信号, 高电平有效. 并行接口时并且选择 8080 时序时: 读数据, 低电平有效. 串行接口时: 接 VDD 或悬空 |
| 8-13 | D0-D5 | I/O | 数据总线 D0~D5 串行接口时: 空脚 |
| 14 | D6 (SCLK) | I/O | 并行接口时: 数据总线 D6 串行接口时: 串行时钟 (SCLK) |
| 15 | D7 (SDA) | I/O | 并行接口时: 数据总线 D7 串行接口时: 串行数据 (SDA) |
| 16 | VDD | 供电电源正极 | 供电电源正极 |
| 17 | VSS | 接地 | 0V |
| 18 | NC | 空脚 | 空脚 |
| 19 | NC | 空脚 | 空脚 |
| 20 | NC | 空脚 | 空脚 |
| 21 | NC | 空脚 | 空脚 |
| 22 | V0 | 倍压电路 | V0 和 XV0 之间串一个 1uf 电容 |
| 23 | XV0 | 倍压电路 | |
| 24 | NC | 空脚 | 空脚 |
| 25 | VG | 偏置电压 | VG 接一个 1uf 电容到地 |
| 26 | NC | 空脚 | 空脚 |
| 27 | NC | 空脚 | 空脚 |
| 28 | NC | 空脚 | 空脚 |
| 29 | NC | 空脚 | 空脚 |
| 30 | NC | 空脚 | 空脚 |
| 31 | NC | 空脚 | 空脚 |
| 32 | C86 | 选择 6800 或 8080 | 并行接口时: H:6800 系统, L:8080 系统。 串行接口时: 接 VDD |
| 33 | P/S | 选串并控制接口 | 接 VDD:选择并行接口, 接 VSS:选择串行接口 |
| 34 | NC | 空脚 | 空脚 |
| 35 | NC | 空脚 | 空脚 |
| 36 | NC | 空脚 | 空脚 |

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)。

4.2 内部电路框图:

电路框图

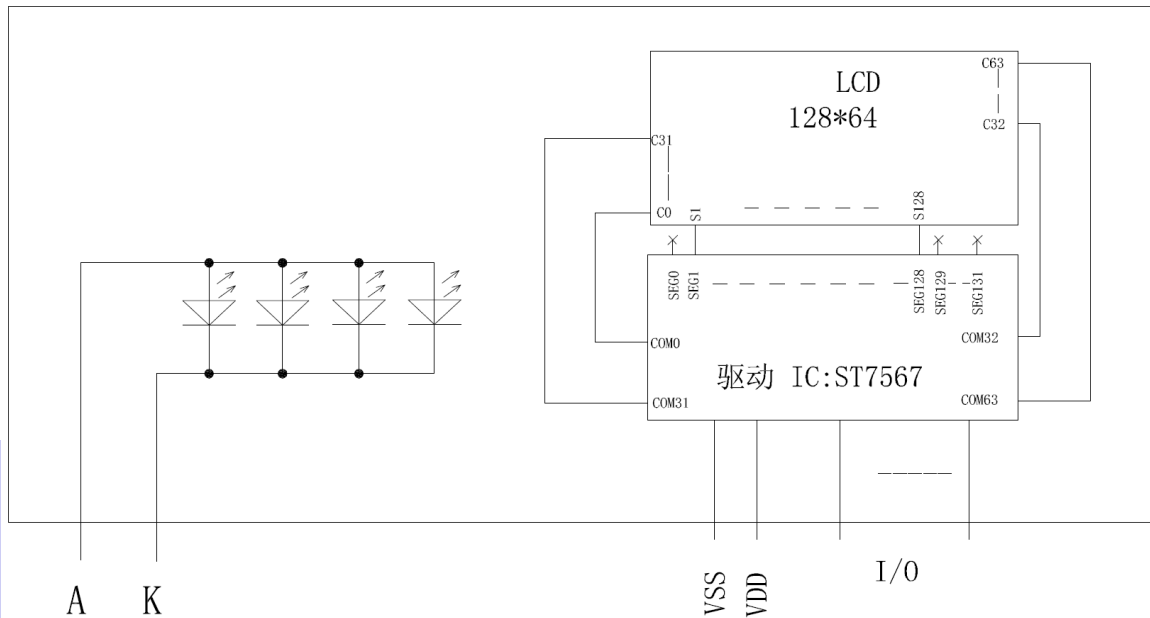


图 2: JLX12864G-542 图像点阵型液晶模块的电路框图

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:

背光灯: 白色。

正常工作电流为: 32~60mA (LED 灯数共 4 颗);

工作电压: 3.0V; (串 20 欧电阻接 3.3V, 串 100 欧接 5.0V)

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 |
| 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 | | 2.4 | 3.3 | 3.6 | V |
| 背光工作电压 | VLED | | 2.9 | 3.0 | 3.1 | V |
| 输入高电平 | V_{IHC} | - | $0.8 \times VDD$ | - | VDD | V |
| 输入低电平 | V_{ILC} | - | VSS | - | $0.2 \times VDD$ | V |
| 输出高电平 | V_{OHC} | $I_{OH} = 0.2\text{mA}$ | $0.8 \times VDD$ | - | VDD | V |
| 输出低电平 | V_{OLC} | $I_{OL} = 1.2\text{mA}$ | VSS | - | $0.2 \times VDD$ | V |
| 模块工作电流 | I_{DD} | VDD = 3.3V | - | | 0.3 | mA |
| 背光工作电流 | I_{LED} | $V_{LED} = 3.0\text{V}$ | 24 | 45 | 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

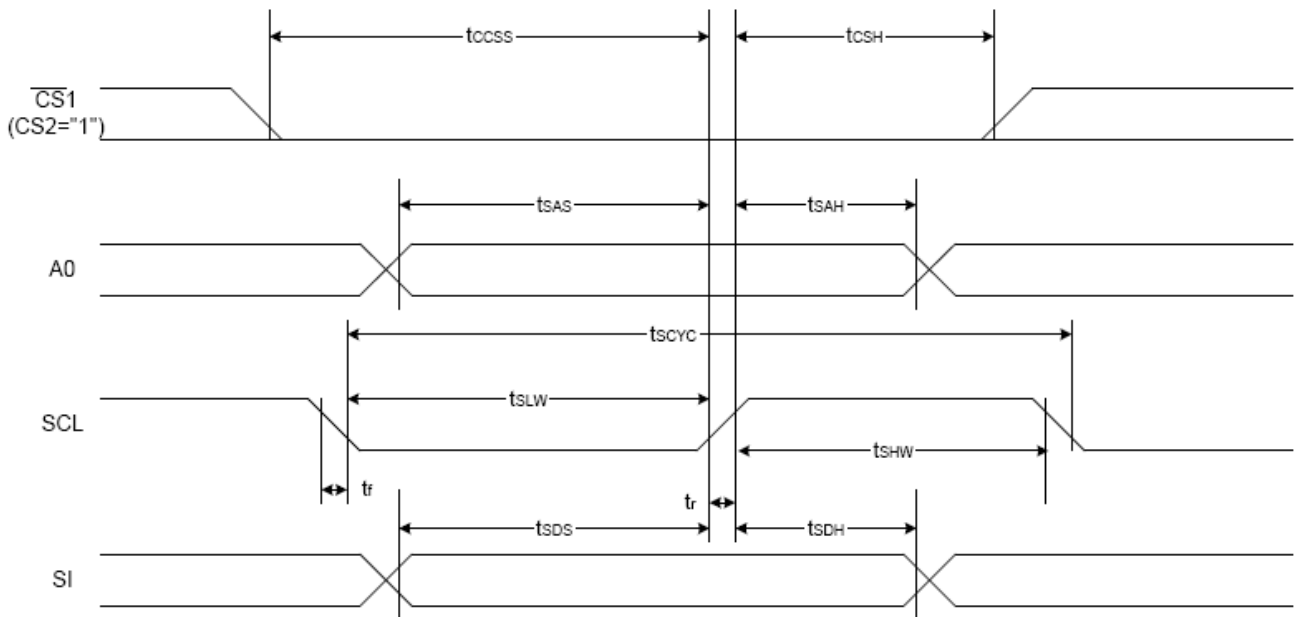


图 3. 从 CPU 写到 ST7567 (Writing Data from CPU to ST7567)

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 (4-line SPI Clock Period) | T_{scyc} | 引脚: SCK | 50 | -- | 25 | ns |
| 保持SCK高电平脉宽 (SCK "H" pulse width) | T_{shw} | 引脚: SCK | 25 | | | ns |
| 保持SCK低电平脉宽 (SCK "L" pulse width) | T_{slw} | 引脚: SCK | 25 | | | ns |
| 地址建立时间 (Address setup time) | T_{sas} | 引脚: RS | 20 | -- | -- | ns |

| | | | | | | |
|-------------------------------|-----------|--------|----|----|----|----|
| 地址保持时间 (Address hold time) | T_{sah} | 引脚: RS | 10 | -- | -- | ns |
| 数据建立时间 (Data setup time) | T_{sds} | 引脚: SI | 20 | -- | -- | ns |
| 数据保持时间 (Data hold time) | T_{SDH} | 引脚: SI | 10 | -- | -- | 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 | 40 | | | ns |

VDD = 3.0V ± 5%, Ta = 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)

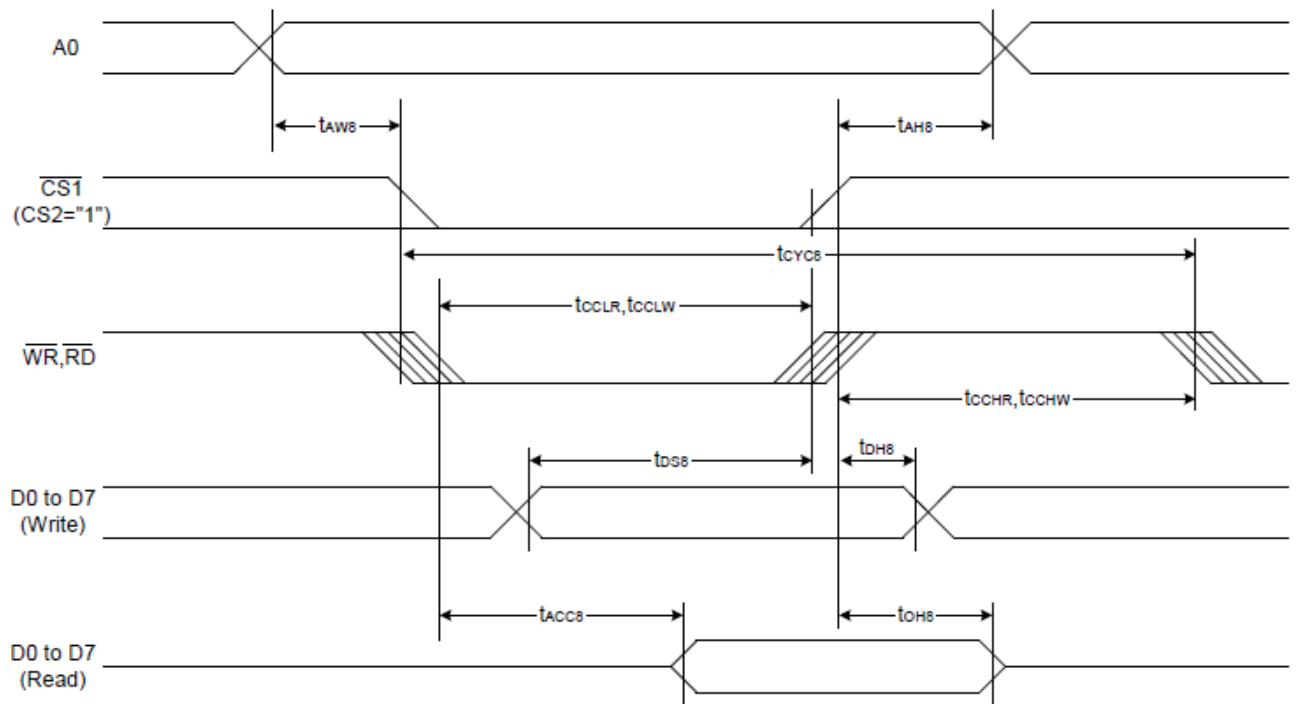


图 4. 从 CPU 写到 ST7567 (Writing Data from CPU to ST7567)

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

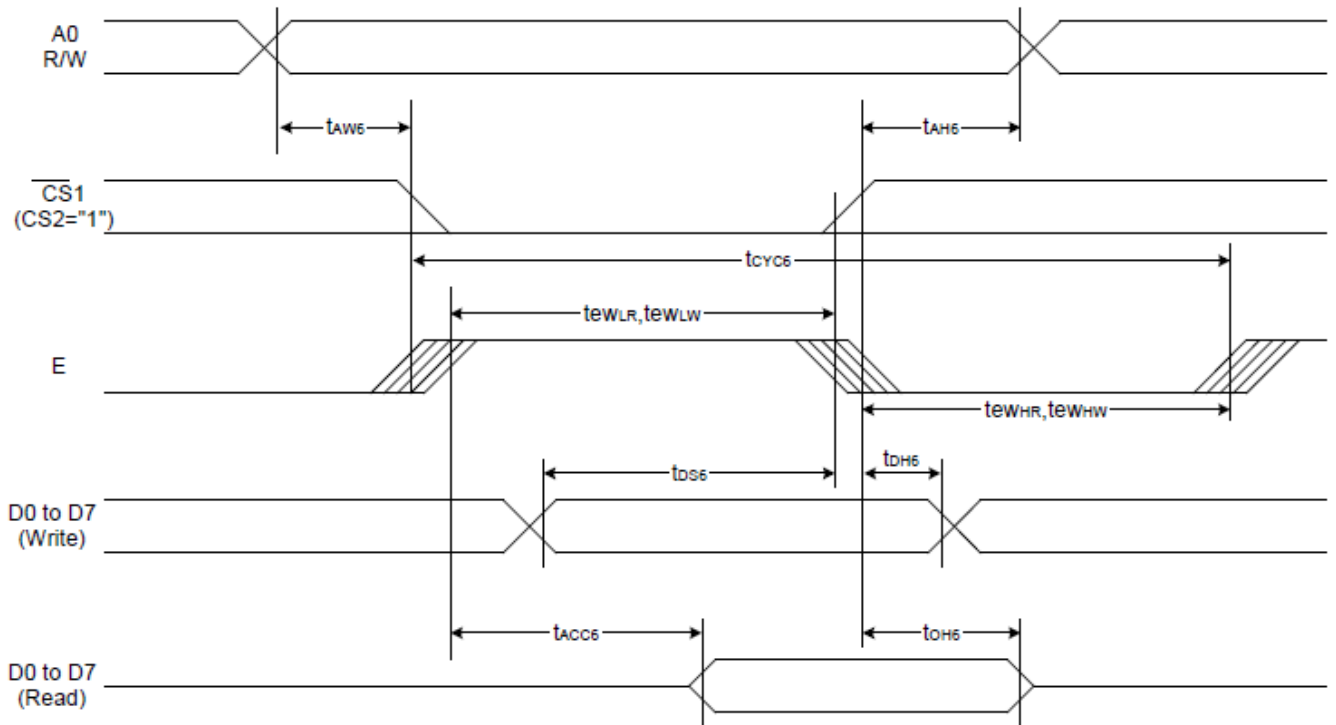


图 5. 从 CPU 写到 ST7567 (Writing Data from CPU to ST7567)

6.4 并行接口: 时序要求 (AC 参数):

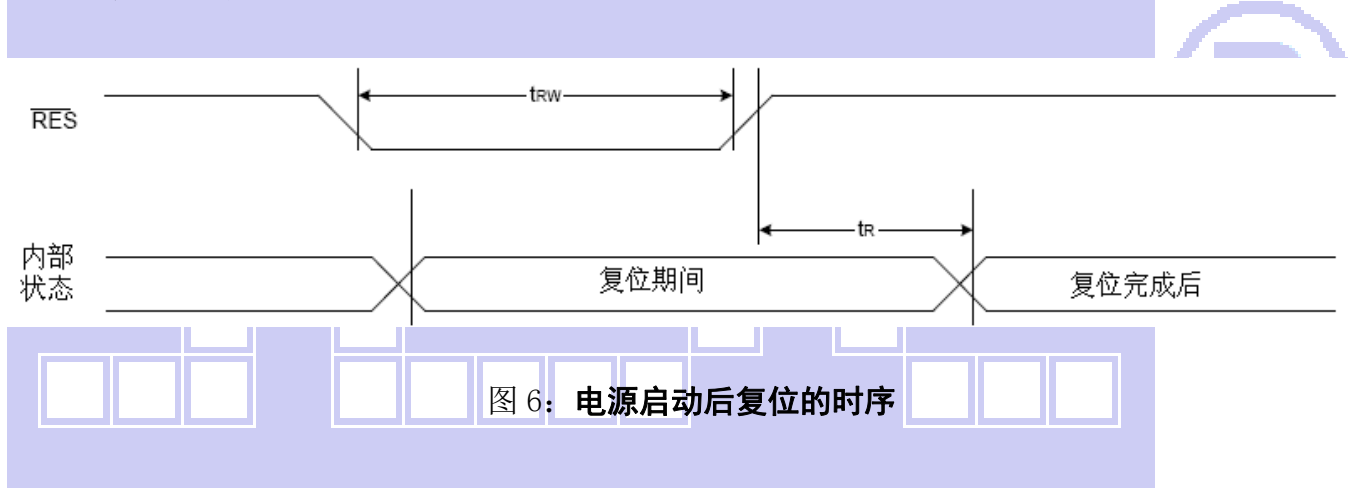
写数据到 ST7567 的时序要求: (8080 系列 MPU)

| 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 | — | — | ns |
| 读时间 | | tACC8 | — | — | 70 | ns |
| 读输出允许时间 | | tOH8 | 5 | — | 50 | ns |

表 5

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 | |
| 读输出允许时间 | | tOH6 | 5 | | 50 | ns |

表 6
6.5 电源启动后复位的时序要求 (RESET CONDITION AFTER POWER UP):

表 7: 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 | | 100 | — | — | ms |
| 复位保持低电平的时间 | trw | 引脚: RESET | 100 | — | — | ms |

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 0x60 | |
| (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 : 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 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) LCD 偏压比设置 (LCD bias set) | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 1 | 设置偏压比: 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 | 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 详情请参考IC资料第43-44页 | |
| (13) 退出上述“读-改-写”指令(End) | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0XEE : 上述“读-改-写”指令结束 详情请参考 IC 资料第 43-44 页 | |
| (14) 软件复位 (Reset) | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0XE2 : 软件复位。 | |

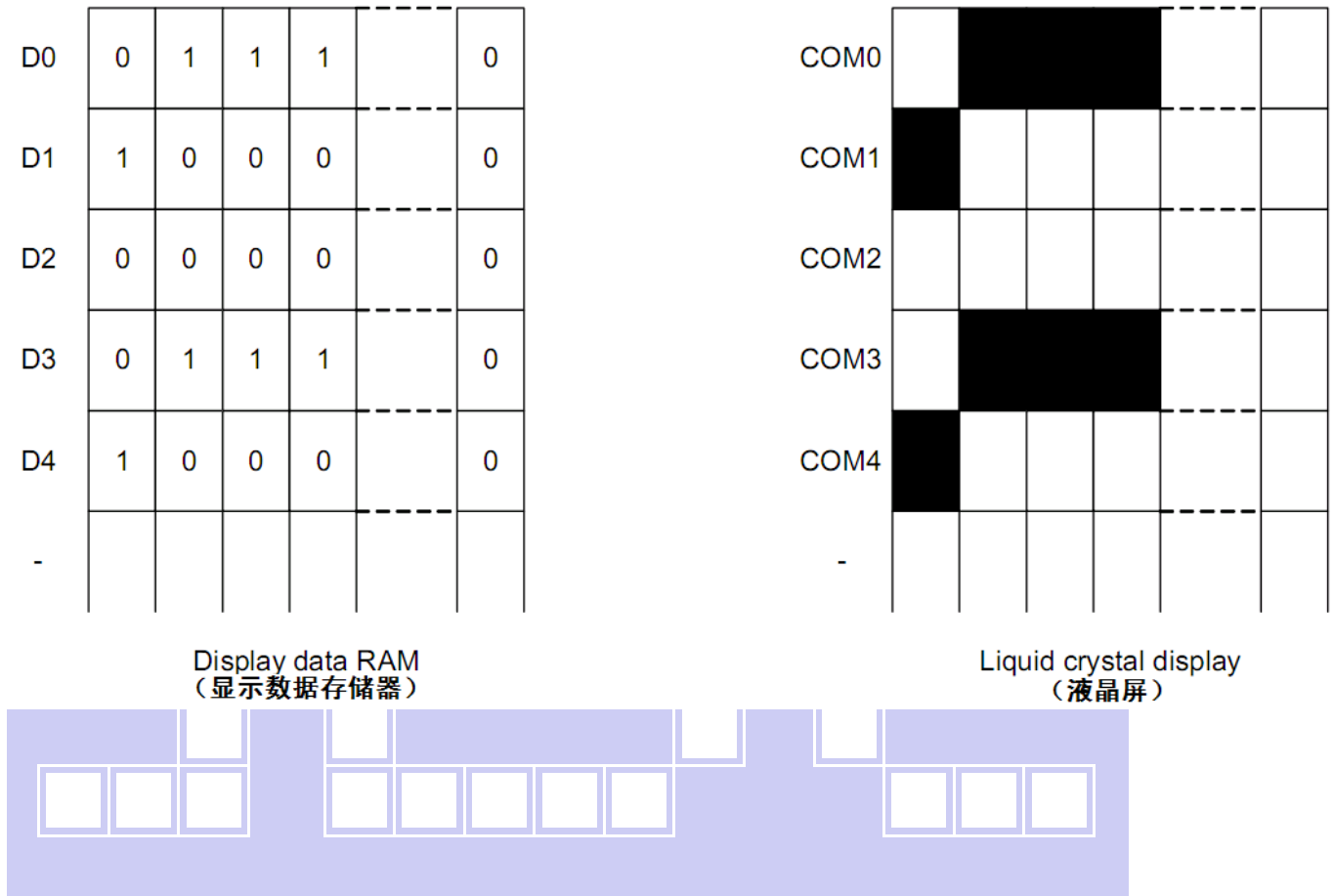
| | | | | | | | | | | |
|---|------------|---|---|---------------------------|---|---|----------------------|---|---|---|
| (15) 行扫描顺序选择 (Common output mode select) | | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 行扫描顺序选择: 0XC0 :普通扫描顺序: 从上到下 0XC8 :反转扫描顺序: 从下到上 |
| (16) 电源控制 (Power control set) | | 0 | 0 | 1 | 0 | 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 | 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 | 静态图标的开关设置: 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 |
| (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! |

温馨提示: 请详细参考 IC 资料”ST7567_V1.7.PDF”的第 21~28 页。

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*64 点阵的屏分为 8 个“页”, 从第 0“页”到第 7“页”。

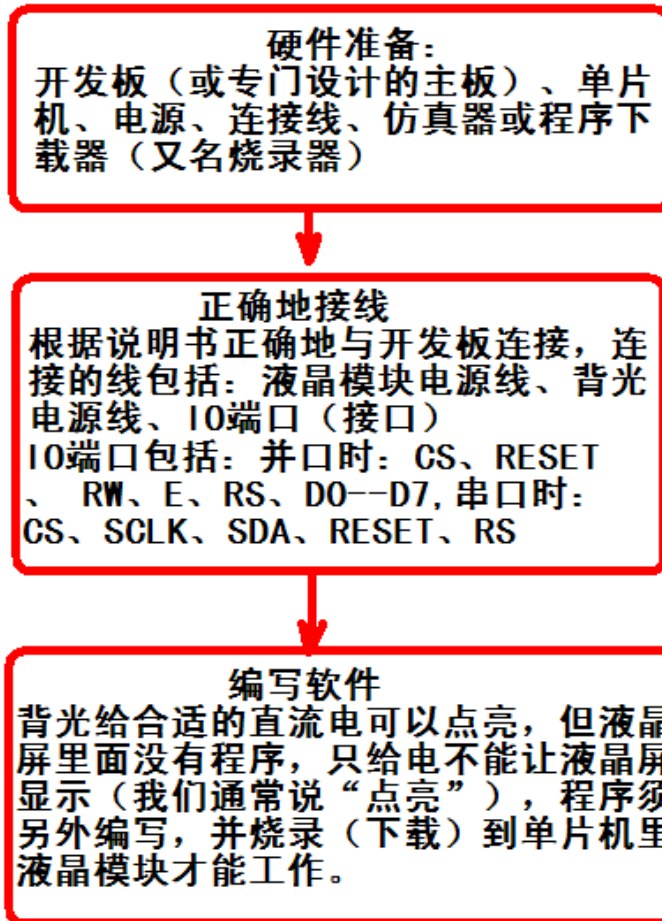
DB7--DB0 的排列方向: 数据是从下向上排列的。最低位 D0 是在最上面, 最高位 D7 是在最下面。每一位 (bit) 数据对应一个点阵, 通常“1”代表点亮该点阵, “0”代表关掉该点阵。 如下图所示:



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



7.5 程序举例:

液晶模块与 CPU (以 51 系列单片机为例) 接口图如下:

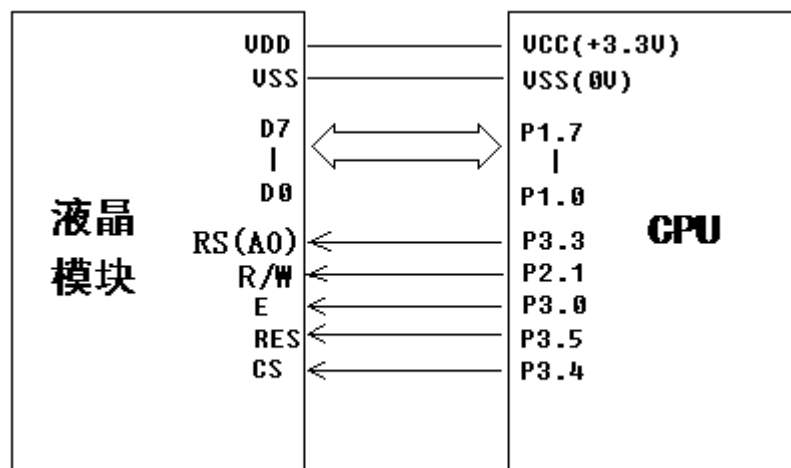
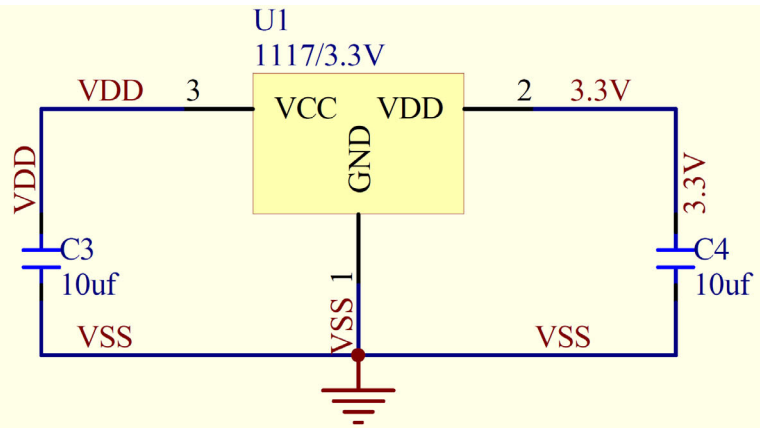


图 7. 并行接口

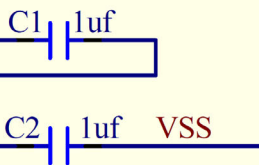
并行电路图

36PIN

| | | |
|------|----|--------|
| NC | 1 | |
| NC | 2 | |
| CS1 | 3 | CS |
| REST | 4 | RST |
| A0 | 5 | A0(RS) |
| WR | 6 | WR |
| RD | 7 | RD(E) |
| DB0 | 8 | D0 |
| DB1 | 9 | D1 |
| DB2 | 10 | D2 |
| DB3 | 11 | D3 |
| DB4 | 12 | D4 |
| DB5 | 13 | D5 |
| DB6 | 14 | D6 |
| DB7 | 15 | D7 |
| VDD | 16 | VDD |
| VSS | 17 | VSS |
| NC | 18 | |
| NC | 19 | |
| NC | 20 | |
| NC | 21 | |
| V0 | 22 | V0 |
| XV0 | 23 | XV0 |
| NC | 24 | |
| VG | 25 | VG |
| NC | 26 | |
| NC | 27 | |
| NC | 28 | |
| NC | 29 | |
| NC | 30 | |
| NC | 31 | |
| C86 | 32 | C86 |
| PS | 33 | VDD |
| NC | 34 | |
| NC | 35 | |
| NC | 36 | |



液晶模块使用电压是2.7V-3.5V
 5.0V供电时，需用1117/3.3将电压转为3.3V再供给LCD使用
 3.3V供电时，可直接供给LCD使用
 所选电容耐压为25V以上



注意： C86接VDD时为： 6800时序
 C86接VSS时为： 8080时序

并口原理图

7.51、程序

点亮液晶模块的编程步骤



以下为并行方式的范例程序:

```

// 液晶模块型号: JLX12864G-542,
// 接口方式: 并行接口, 6800 时序
// 驱动 IC 是: ST7567
// 本程序所带的 8x16 点阵及 5x8 点阵的 ASCII 码字符的数据都是从 JLX-GB2312 型号字库 IC 里读出来的国标的。
// 单片机: ST12C5A60S2 (51 系列单片机), 晶振: 12MHz
// 版权所有: 深圳市晶联讯电子有限公司
// 网址: http://www.jlxlcd.cn;
// 编程: 叶建人 2014 年 3 月 4 日
    
```

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

```

sbit lcd_cs1    =P3^4;    //接口定义, CS:片选
sbit lcd_reset =P3^5;    //接口定义, RESET:复位
sbit lcd_rs    =P3^3;    //接口定义, RS:命令/数据寄存器选择。也叫“A0”,或“CD”
sbit lcd_e     =P3^0;    //接口定义, E:读写使能信号
sbit lcd_rw    =P2^1;    //接口定义, R/W:读/写
sbit key       =P2^0;    //按键接口, P2.0 口与 GND 之间接一个按键
    
```

```
//另外 P1.0~1.7 对应 DB0~DB7
```



```

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

uchar code ascii_table_8x16[95][16];
uchar code ascii_table_5x8[95][5];
uchar code cheng1[];
uchar code gong1[];
uchar code zhuang1[];
uchar code tai1[];
uchar code shi1[];
uchar code yong1[];
uchar code bmp_12864_1[];
uchar code bmp_12864_2[];
uchar code bmp_12864_3[];
uchar code bmp_12864_4[];
uchar code bmp_12864_5[];

//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: 当有按键按下时, 单片机会检测到一个低电平
void waitkey()
{
repeat:   if(key==1) goto repeat;
          else      delay(2000);
}

//传一个字节的命令到液晶屏驱动 IC
void transfer_command(int data1)
{
    lcd_cs1=0;    //片选=0, 选中此 IC
    lcd_rs=0;    //RS=0: 表示以下传输的一个字节是命令
    lcd_e=0;     //E:读写使能
    lcd_rw=0;    //RW=0:写
    P1=data1;
    lcd_e=1;
    delay_us(2);
    lcd_cs1=1;   //传完数据后片选=1, 防止意外选中此 IC, 对抗干扰有好处
    lcd_e=0;
}

//传一个字节的数据到液晶屏驱动 IC
void transfer_data(int data1)
{
    lcd_cs1=0;    //片选=0, 选中此 IC
    lcd_rs=1;    //RS=1: 表示以下传输的一个字节是要显示的点阵数据
    lcd_e=0;     //E:读写使能
    lcd_rw=0;    //RW=0:写
    P1=data1;
    lcd_e=1;
    delay_us(2);
    lcd_cs1=1;   //传完数据后片选=1, 防止意外选中此 IC, 对抗干扰有好处
    lcd_e=0;
}
    
```



//LCD 模块初始化

```
void initial_lcd()
```

```
{
    lcd_reset=0;        //低电平复位
    delay(100);         //100 毫秒
    lcd_reset=1;        //复位完毕
    delay(100);         //100 毫秒
    transfer_command(0xe2); //软复位
    delay(5);           //5 毫秒
    transfer_command(0x2c); //升压步骤 1: 打开升压
    delay(5);           //5 毫秒
    transfer_command(0x2e); //升压步骤 2: 打开升压及电压调整器
    delay(5);           //5 毫秒
    transfer_command(0x2f); //升压步骤 3: 打开升压、电压调整器、电压跟随器
    delay(5);           //5 毫秒
    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(0x28); //微调对比度的值, 可设置范围 0x00~0x3f
    transfer_command(0xa2); //1/9 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(0x60); //扫描起始行: 0x40 表示从第 1 行开始, 0x41:第 2 行...0x60:第 33 行, 由于 LCD 的走线设计, 本 LCD 特殊地从 33 行开始扫描。
    transfer_command(0xaf); //打开显示
}
```

//LCD 地址设置, 括号内的参数分别为 (页, 列)

```
void lcd_address(uchar page, uchar column)
```

```
{
    column=column;
    page=page-1; //我们平常说的第 1 页, 其实对 IC 来说是第 0 页, 所以在这里减去 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。
    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++) //液晶驱动 IC 有 65 行 (分为 9 个页), 只引出来 64 行 (8 页), 有 1 行没引出来, 清屏时一并清掉。
    {
        lcd_address(1+i, 1);
        for(j=0; j<132; j++) //液晶驱动 IC 有 132 列, 只引出来 128 列, 有 4 列没引出来, 所以清屏时一并清掉。
        {
            transfer_data(0x00); //全部显示数据是 0, 以此清屏。
        }
    }
}
```

//===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<8; j++)
    {
        lcd_address(j+1, 0);
        for(i=0; i<128; i++)
        {
            transfer_data(data1);
            transfer_data(data2);
        }
    }
}
```

//显示 128x64 点阵图像

```
void display_graphic_128x64(uchar page, uchar column, uchar *dp)
```

```
{
    int i, j;
    for(j=0; j<8; j++)
    {
        lcd_address(page+j, column);
        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 *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 点阵 ASCII 字符串, 括号里的参数是 (页, 列, 数据指针)

```

```

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]); //传完一个字节数据后, 列地址会自动+1
                }
            }
            i++;
            column+=8;
        }
        else

```



```

        i++;
    }
}
//显示 5x8 点阵 ASCII 字符串, 括号里的参数是 (页, 列, 数据指针)
void display_string_5x8(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]); //传完一个字节数据后, 列地址会自动+1
            }
            transfer_data(0x00);
            i++;
            column+=6;
        }
        else
            i++;
    }
}

```

```
void main(void)
```

```

{
    while(1)
    {
        initial_lcd(); //LCD 初始化
        //显示一些出厂电测常用画面
        clear_screen(); //全屏清屏
        test_display(0xff, 0xff); //全显示
        waitkey();

        test_display(0x55, 0xaa); //雪花 1
        waitkey();

        test_display(0xaa, 0x55); //雪花 2
        waitkey();

        //显示一些 128x64 点阵的单个图像
        clear_screen(); //clear all dots
        display_graphic_128x64(1, 1, bmp_12864_1); //显示 128x64 点阵图片
        waitkey();

        clear_screen(); //clear all dots
        display_graphic_128x64(1, 1, bmp_12864_2);
        waitkey();

        clear_screen(); //全屏清屏
        display_graphic_128x64(1, 1, bmp_12864_3);
        waitkey();

        clear_screen(); //全屏清屏
        display_graphic_128x64(1, 1, bmp_12864_4);
        waitkey();

        clear_screen(); //全屏清屏
        display_graphic_128x64(1, 1, bmp_12864_5);
        waitkey();

        //显示一些不同点阵的汉字或字符
        clear_screen(); //全屏清屏
        display_graphic_32x32(1, (1+32*1), cheng1); //在第 1 页, 第 49 列显示单个汉字“成”
        display_graphic_32x32(1, (1+32*2), gong1); //在第 1 页, 第 49 列显示单个汉字“功”
        display_graphic_16x16(6, 1, zhuang1); //在第 5 页, 第 1 列显示单个汉字“状”
        display_graphic_16x16(6, (1+16), tail); //在第 5 页, 第 17 列显示单个汉字“态”
        display_string_8x16(6, (1+16*16), ":"); //在第 5 页, 第 25 列显示单个字符“:”
        display_graphic_16x16(6, (1+16*2+8), shi1); //在第 5 页, 第 41 列显示单个汉字“使”
        display_graphic_16x16(6, (1+16*3+8), yong1); //在第 5 页, 第 49 列显示单个汉字“用”
    }
}

```



```

display_string_8x16(6, 89, "12:45"); //在第 5 页, 第 89 列显示单个数字"0"
waitkey();

//显示一些字符串
clear_screen(); //全屏清屏
display_string_8x16(1, 1, "!\"#$%&'()*+,-./"); //显示 8x16 的 ASCII 码字符串
//-----括号里的参数分别为 (页地址, 列地址, 要显示的字符),
//-----注意: 为了能显示双引号("), 在双引号前面加一个斜杠(\), 斜杠(\)本身并不显示出来
display_string_8x16(3, 1, "0123456789:;<=>?"); //括号里的参数分别为 (页地址, 列地址, 要显示的字符)
display_string_8x16(5, 1, "@ABCDEFGHIJKLMNO");
display_string_8x16(7, 1, "PQRSTUVWXYZ[\\"^_"); //注意: 为了能显示斜杠(\), 在斜杠(\)前面再加一个斜杠(\), 两个斜杠(\)只显示一个出来
waitkey();

//显示一些字符串 //全屏清屏
clear_screen();
display_string_8x16(1, 1, "abcdefghijklmnop");
display_string_8x16(3, 1, "pqrstuvwxyz[ ]~");
display_string_5x8(5, 1, "!\"#$%&'()*+,-./01234"); //显示 5x8 的 ASCII 码字符串
display_string_5x8(6, 1, "56789:;<=>?@ABCDEFGHI");
display_string_5x8(7, 1, "JKLMNOPQRSTUVWXYZ[\"^~");
display_string_5x8(8, 1, "`_abcdefghijklmnopqrs");
waitkey();
}
}

uchar code ascii_table_8x16[95][16]={
//粗体 8x16 点阵的 ASCII 码的点阵数据, 从"JLX-GB2312"型号的库 IC 中读出来的国标的。
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //-(即"空格") ASCII 码: 0X20
0x00, 0x00, 0x38, 0xFC, 0xFC, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x00, 0x00, 0x00, 0x00, //!- ASCII 码: 0X21
0x00, 0x0E, 0x1E, 0x00, 0x00, 0x1E, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //"- ASCII 码: 0X22
0x20, 0xF8, 0xF8, 0x20, 0xF8, 0xF8, 0x20, 0x00, 0x02, 0x0F, 0x0F, 0x02, 0x0F, 0x0F, 0x02, 0x00, //#- ASCII 码: 0X23
0x38, 0x7C, 0x44, 0x47, 0x47, 0xCC, 0x98, 0x00, 0x06, 0x0C, 0x08, 0x38, 0x38, 0x0F, 0x07, 0x00, //$$- ASCII 码: 0X24
0x30, 0x30, 0x00, 0x80, 0xC0, 0x60, 0x30, 0x00, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x0C, 0x0C, 0x00, //%- ASCII 码: 0X25
0x80, 0xD8, 0x7C, 0xE4, 0xBC, 0xD8, 0x40, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, //&- ASCII 码: 0X26
0x00, 0x10, 0x1E, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //'- ASCII 码: 0X27
0x00, 0x00, 0xF0, 0xF8, 0x0C, 0x04, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x00, 0x00, //(- ASCII 码: 0X28
0x00, 0x00, 0x04, 0x0C, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0C, 0x07, 0x03, 0x00, 0x00, //)- ASCII 码: 0X29
0x80, 0xA0, 0xE0, 0xC0, 0xC0, 0xE0, 0xA0, 0x80, 0x00, 0x02, 0x03, 0x01, 0x01, 0x03, 0x02, 0x00, //*- ASCII 码: 0X2A
0x00, 0x80, 0x80, 0xE0, 0xE0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00, 0x00, 0x00, //+- ASCII 码: 0X2B
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x1E, 0x0E, 0x00, 0x00, 0x00, //,- ASCII 码: 0X2C
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //-- ASCII 码: 0X2D
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0C, 0x00, 0x00, 0x00, //.- ASCII 码: 0X2E
0x00, 0x00, 0x00, 0x80, 0xC0, 0x60, 0x30, 0x00, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, ///- ASCII 码: 0X2F
0xF8, 0xF8, 0x0C, 0xC4, 0x0C, 0xF8, 0xF0, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x0C, 0x07, 0x03, 0x00, //0- ASCII 码: 0X30
0x00, 0x10, 0x18, 0xFC, 0xFC, 0x00, 0x00, 0x00, 0x08, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x00, //1- ASCII 码: 0X31
0x08, 0x0C, 0x84, 0xC4, 0x64, 0x3C, 0x18, 0x00, 0x0E, 0x0F, 0x09, 0x08, 0x08, 0x0C, 0x0C, 0x00, //2- ASCII 码: 0X32
0x08, 0x0C, 0x44, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x04, 0x0C, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //3- ASCII 码: 0X33
0xC0, 0xE0, 0xB0, 0x98, 0xFC, 0xFC, 0x80, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x00, //-4- ASCII 码: 0X34
0x7C, 0x7C, 0x44, 0x44, 0x44, 0xC4, 0x84, 0x00, 0x04, 0x0C, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //-5- ASCII 码: 0X35
0xF0, 0xF8, 0x4C, 0x44, 0x44, 0xC0, 0x80, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //-6- ASCII 码: 0X36
0x0C, 0x0C, 0x04, 0x84, 0xC4, 0x7C, 0x3C, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, //-7- ASCII 码: 0X37
0xB8, 0xFC, 0x44, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //-8- ASCII 码: 0X38
0x38, 0x7C, 0x44, 0x44, 0x44, 0xFC, 0xF8, 0x00, 0x00, 0x08, 0x08, 0x08, 0x0C, 0x07, 0x03, 0x00, //-9- ASCII 码: 0X39
0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x06, 0x00, 0x00, 0x00, 0x00, //:- ASCII 码: 0X3A
0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0E, 0x06, 0x00, 0x00, 0x00, //;- ASCII 码: 0X3B
0x00, 0x80, 0xC0, 0x60, 0x30, 0x18, 0x08, 0x00, 0x00, 0x00, 0x01, 0x03, 0x06, 0x0C, 0x08, 0x00, //<- ASCII 码: 0X3C
0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, //=- ASCII 码: 0X3D
0x00, 0x08, 0x18, 0x30, 0x60, 0xC0, 0x80, 0x00, 0x00, 0x08, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x00, //>- ASCII 码: 0X3E
0x18, 0x1C, 0x04, 0xC4, 0xE4, 0x3C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x0D, 0x00, 0x00, 0x00, //?- ASCII 码: 0X3F
0xF0, 0xF0, 0x08, 0xC8, 0xC8, 0xF8, 0xF0, 0x00, 0x07, 0x0F, 0x08, 0x0B, 0x0B, 0x0B, 0x01, 0x00, //@- ASCII 码: 0X40
0xE0, 0xF0, 0x98, 0x8C, 0x98, 0xF0, 0xE0, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, //-A- ASCII 码: 0X41
0x04, 0xFC, 0xFC, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0F, 0x07, 0x00, //-B- ASCII 码: 0X42
0xF0, 0xF8, 0x0C, 0x04, 0x04, 0x0C, 0x18, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x08, 0x0C, 0x06, 0x00, //-C- ASCII 码: 0X43
0x04, 0xFC, 0xFC, 0x04, 0x0C, 0xF8, 0xF0, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x0C, 0x07, 0x03, 0x00, //-D- ASCII 码: 0X44
0x04, 0xFC, 0xFC, 0x44, 0xE4, 0x0C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0C, 0x0C, 0x0E, 0x00, //-E- ASCII 码: 0X45
0x04, 0xFC, 0xFC, 0x44, 0xE4, 0x0C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, //-F- ASCII 码: 0X46
0xF0, 0xF8, 0x0C, 0x84, 0x84, 0x8C, 0x98, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x08, 0x07, 0x0F, 0x00, //-G- ASCII 码: 0X47

```





```

0xFC, 0xFC, 0x40, 0x40, 0x40, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, //H- ASCII 码: 0X48
0x00, 0x00, 0x04, 0xFC, 0xFC, 0x04, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, //I-
0x00, 0x00, 0x00, 0x04, 0xFC, 0xFC, 0x04, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x0F, 0x07, 0x00, 0x00, //J-
0x04, 0xFC, 0xFC, 0x00, 0xE0, 0x3C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x01, 0x0F, 0x0E, 0x00, //K-
0x04, 0xFC, 0xFC, 0x04, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0C, 0x0E, 0x00, //L-
0xFC, 0xFC, 0x38, 0x70, 0x38, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, //M-
0xFC, 0xFC, 0x38, 0x70, 0xE0, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, //N-
0xF8, 0xFC, 0x04, 0x04, 0x04, 0xFC, 0xF8, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //O-
0x04, 0xFC, 0xFC, 0x44, 0x44, 0x7C, 0x38, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, //P-
0xF8, 0xFC, 0x04, 0x04, 0x04, 0xFC, 0xF8, 0x00, 0x07, 0x0F, 0x08, 0x0E, 0x3C, 0x3F, 0x27, 0x00, //Q-

```

```

0x04, 0xFC, 0xFC, 0x44, 0xC4, 0xFC, 0x38, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x00, 0x0F, 0x0F, 0x00, //R-
0x18, 0x3C, 0x64, 0x44, 0xC4, 0x9C, 0x18, 0x00, 0x06, 0x0E, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //S-
0x00, 0x1C, 0x0C, 0xFC, 0xFC, 0x0C, 0x1C, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, //T-
0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //U-
0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x01, 0x03, 0x06, 0x0C, 0x06, 0x03, 0x01, 0x00, //V-
0xFC, 0xFC, 0x00, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x07, 0x0F, 0x0E, 0x03, 0x0E, 0x0F, 0x07, 0x00, //W-
0x0C, 0x3C, 0xF0, 0xE0, 0xF0, 0x3C, 0x0C, 0x00, 0x0C, 0x0F, 0x03, 0x01, 0x03, 0x0F, 0x0C, 0x00, //X-
0x00, 0x0C, 0x7C, 0xC0, 0xC0, 0x7C, 0x3C, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, //Y-
0x1C, 0x0C, 0x84, 0xC4, 0x64, 0x3C, 0x1C, 0x00, 0x0E, 0x0F, 0x09, 0x08, 0x08, 0x0C, 0x0E, 0x00, //Z-
0x00, 0x00, 0xFC, 0xFC, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x08, 0x08, 0x00, 0x00, //[-

```

```

0x38, 0x70, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0E, 0x00, //\ -
0x00, 0x00, 0x04, 0x04, 0xFC, 0xFC, 0x00, 0x00, 0x00, 0x08, 0x08, 0x0F, 0x0F, 0x00, 0x00, //] -
0x08, 0x0C, 0x06, 0x03, 0x06, 0x0C, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //^ -
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, //_ -
0x00, 0x00, 0x03, 0x07, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //` -
0x00, 0xA0, 0xA0, 0xA0, 0xE0, 0xC0, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, //a- ASCII 码: 0X61
0x04, 0xFC, 0x00, 0x20, 0x60, 0xC0, 0x80, 0x00, 0x00, 0x0F, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //b-
0xC0, 0xE0, 0x20, 0x20, 0x20, 0x60, 0x40, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0C, 0x04, 0x00, //c-
0x80, 0xC0, 0x60, 0x24, 0xFC, 0xFC, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, //d-
0xC0, 0xE0, 0xA0, 0xA0, 0xA0, 0xE0, 0xC0, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0C, 0x04, 0x00, //e-

```

```

0x40, 0xF8, 0xFC, 0x44, 0x0C, 0x18, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, //f-
0xC0, 0xE0, 0x20, 0x20, 0xC0, 0xE0, 0x20, 0x00, 0x27, 0x6F, 0x48, 0x48, 0x7F, 0x3F, 0x00, 0x00, //g-
0x04, 0xFC, 0xFC, 0x40, 0x20, 0xE0, 0xC0, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x00, 0x0F, 0x0F, 0x00, //h-
0x00, 0x00, 0x20, 0xEC, 0xEC, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, //i-
0x00, 0x00, 0x00, 0x00, 0x20, 0xEC, 0xEC, 0x00, 0x00, 0x30, 0x70, 0x40, 0x40, 0x7F, 0x3F, 0x00, //j-
0x04, 0xFC, 0xFC, 0x80, 0xC0, 0x60, 0x20, 0x00, 0x08, 0x0F, 0x0F, 0x01, 0x03, 0x0E, 0x0C, 0x00, //k-
0x00, 0x00, 0x00, 0x04, 0xFC, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, //l-
0xE0, 0xE0, 0x60, 0xC0, 0x60, 0xE0, 0xC0, 0x00, 0x0F, 0x0F, 0x00, 0x07, 0x00, 0x0F, 0x0F, 0x00, //m-
0x20, 0xE0, 0xC0, 0x20, 0x20, 0xE0, 0xC0, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x0F, 0x0F, 0x00, //n-
0xC0, 0xE0, 0x20, 0x20, 0x20, 0xE0, 0xC0, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, //o-

```

```

0x20, 0xE0, 0xC0, 0x20, 0x20, 0xE0, 0xC0, 0x00, 0x40, 0x7F, 0x7F, 0x48, 0x08, 0x0F, 0x07, 0x00, //p-
0xC0, 0xE0, 0x20, 0x20, 0xC0, 0xE0, 0x20, 0x00, 0x07, 0x0F, 0x08, 0x48, 0x7F, 0x7F, 0x40, 0x00, //q-
0x20, 0xE0, 0xC0, 0x60, 0x20, 0xE0, 0xC0, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, //r-
0x40, 0xE0, 0xA0, 0x20, 0x20, 0x60, 0x40, 0x00, 0x04, 0x0C, 0x09, 0x09, 0x0B, 0x0E, 0x04, 0x00, //s-
0x20, 0x20, 0xF8, 0xFC, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x0C, 0x04, 0x00, //t-
0xE0, 0xE0, 0x00, 0x00, 0xE0, 0xE0, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, //u-
0x00, 0xE0, 0xE0, 0x00, 0x00, 0xE0, 0xE0, 0x00, 0x00, 0x03, 0x07, 0x0C, 0x0C, 0x07, 0x03, 0x00, //v-
0xE0, 0xE0, 0x00, 0x80, 0x00, 0xE0, 0xE0, 0x00, 0x07, 0x0F, 0x0C, 0x07, 0x0C, 0x0F, 0x07, 0x00, //w-
0x20, 0x60, 0xC0, 0x80, 0xC0, 0x60, 0x20, 0x00, 0x08, 0x0C, 0x07, 0x03, 0x07, 0x0C, 0x08, 0x00, //x-
0xE0, 0xE0, 0x00, 0x00, 0x00, 0xE0, 0xE0, 0x00, 0x47, 0x4F, 0x48, 0x48, 0x68, 0x3F, 0x1F, 0x00, //y-

```

```

0x60, 0x60, 0x20, 0xA0, 0xE0, 0x60, 0x20, 0x00, 0x0C, 0x0E, 0x0B, 0x09, 0x08, 0x0C, 0x0C, 0x00, //z- //
0x00, 0x40, 0x40, 0xF8, 0xBC, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x00, //{ -
0x00, 0x00, 0x00, 0xBC, 0xBC, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, //{ | -
0x00, 0x04, 0x04, 0xBC, 0xF8, 0x40, 0x40, 0x00, 0x00, 0x08, 0x08, 0x0F, 0x07, 0x00, 0x00, 0x00, //{ } -
0x08, 0x0C, 0x04, 0x0C, 0x08, 0x0C, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //~ - ASCII 码: 0X7E
};

```

```

uchar code ascii_table_5x8[95][5]={
//5x8 点阵的 ASCII 码的点阵数据, 从"JLX-GB2312"型号的字库 IC 中读出来的国标的。
0x00, 0x00, 0x00, 0x00, 0x00, // - //空格
0x00, 0x00, 0x4f, 0x00, 0x00, // ! -
0x00, 0x07, 0x00, 0x07, 0x00, // " -
0x14, 0x7f, 0x14, 0x7f, 0x14, // # -
0x24, 0x2a, 0x7f, 0x2a, 0x12, // $ -
0x23, 0x13, 0x08, 0x64, 0x62, // % -

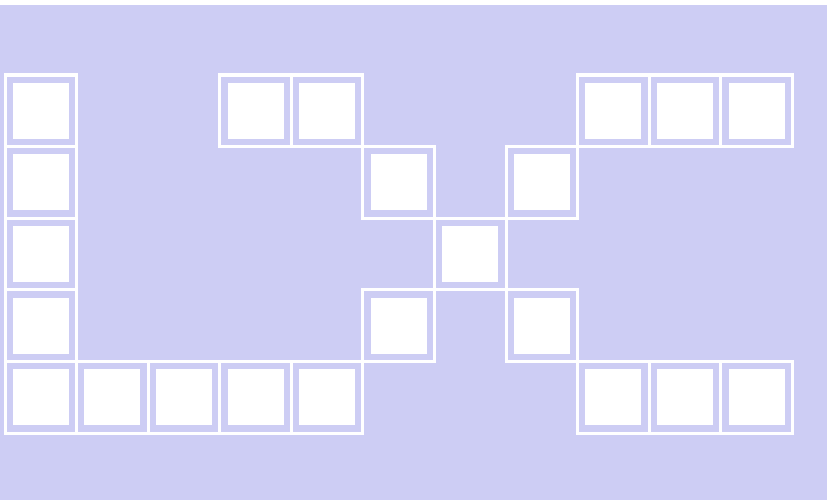
```



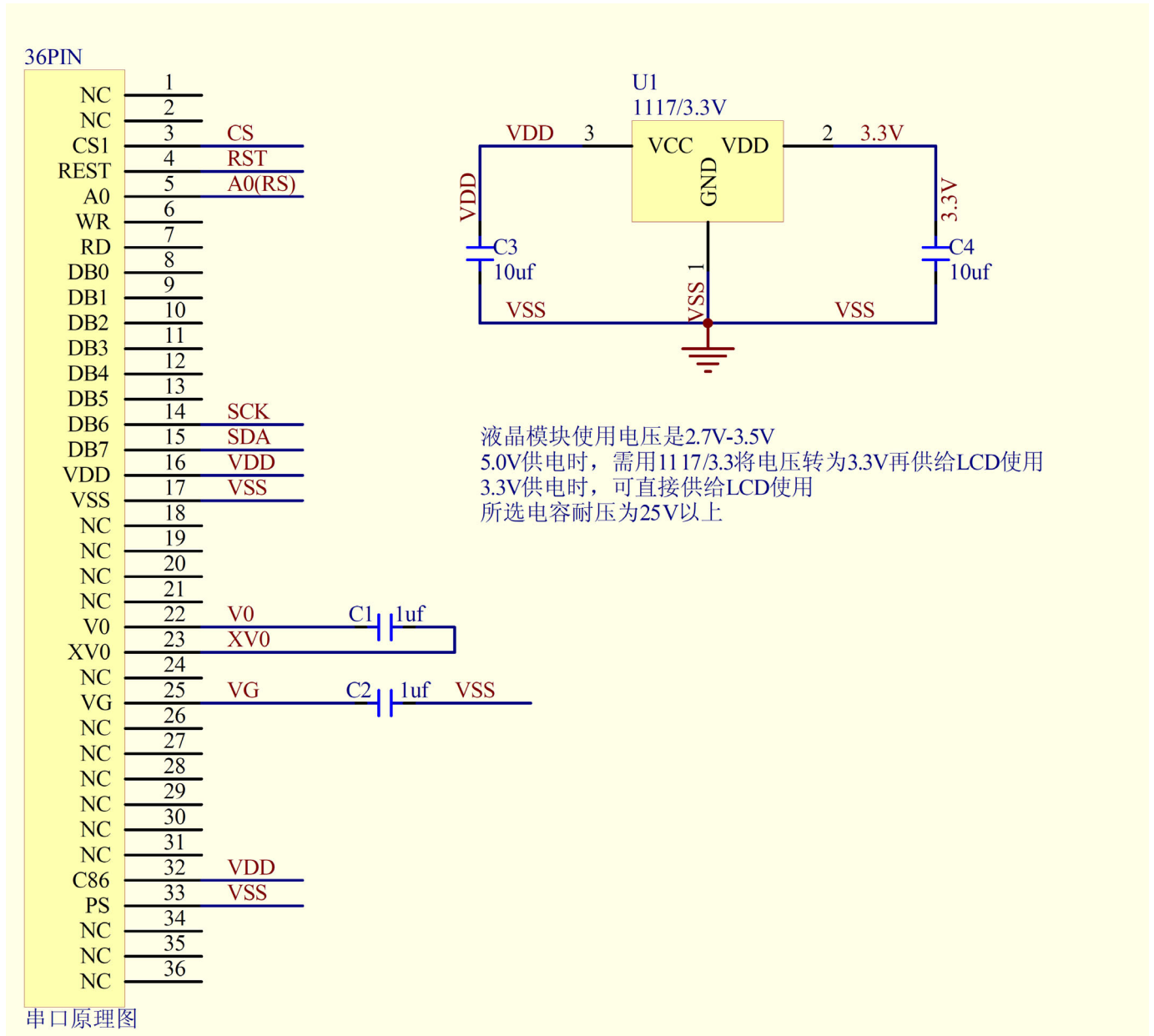
```

0x36, 0x49, 0x55, 0x22, 0x50, //&-
0x00, 0x05, 0x07, 0x00, 0x00, //-' -
0x00, 0x1c, 0x22, 0x41, 0x00, //-( -
0x00, 0x41, 0x22, 0x1c, 0x00, //)- -
0x14, 0x08, 0x3e, 0x08, 0x14, //-* -
0x08, 0x08, 0x3e, 0x08, 0x08, //+- -
0x00, 0x50, 0x30, 0x00, 0x00, //-, -
0x08, 0x08, 0x08, 0x08, 0x08, //--- -
0x00, 0x60, 0x60, 0x00, 0x00, //-. -
0x20, 0x10, 0x08, 0x04, 0x02, //-/ -
0x3e, 0x51, 0x49, 0x45, 0x3e, //-0 -
0x00, 0x42, 0x7f, 0x40, 0x00, //-1 -
0x42, 0x61, 0x51, 0x49, 0x46, //-2 -
0x21, 0x41, 0x45, 0x4b, 0x31, //-3 -
0x18, 0x14, 0x12, 0x7f, 0x10, //-4 -
0x27, 0x45, 0x45, 0x45, 0x39, //-5 -
0x3c, 0x4a, 0x49, 0x49, 0x30, //-6 -
0x01, 0x71, 0x09, 0x05, 0x03, //-7 -
0x36, 0x49, 0x49, 0x49, 0x36, //-8 -
0x06, 0x49, 0x49, 0x29, 0x1e, //-9 -
0x00, 0x36, 0x36, 0x00, 0x00, //-:- -
0x00, 0x56, 0x36, 0x00, 0x00, //;- -
0x08, 0x14, 0x22, 0x41, 0x00, //<- -
0x14, 0x14, 0x14, 0x14, 0x14, //== -
0x00, 0x41, 0x22, 0x14, 0x08, //>- -
0x02, 0x01, 0x51, 0x09, 0x06, //?- -
0x32, 0x49, 0x79, 0x41, 0x3e, //@ -
0x7e, 0x11, 0x11, 0x11, 0x7e, //-A -
0x7f, 0x49, 0x49, 0x49, 0x36, //-B -
0x3e, 0x41, 0x41, 0x41, 0x22, //-C -
0x7f, 0x41, 0x41, 0x22, 0x1c, //-D -
0x7f, 0x49, 0x49, 0x49, 0x41, //-E -
0x7f, 0x09, 0x09, 0x09, 0x01, //-F -
0x3e, 0x41, 0x49, 0x49, 0x7a, //-G -
0x7f, 0x08, 0x08, 0x08, 0x7f, //-H -
0x00, 0x41, 0x7f, 0x41, 0x00, //-I -
0x20, 0x40, 0x41, 0x3f, 0x01, //-J -
0x7f, 0x08, 0x14, 0x22, 0x41, //-K -
0x7f, 0x40, 0x40, 0x40, 0x40, //-L -
0x7f, 0x02, 0x0c, 0x02, 0x7f, //-M -
0x7f, 0x04, 0x08, 0x10, 0x7f, //-N -
0x3e, 0x41, 0x41, 0x41, 0x3e, //-O -
0x7f, 0x09, 0x09, 0x09, 0x06, //-P -
0x3e, 0x41, 0x51, 0x21, 0x5e, //-Q -
0x7f, 0x09, 0x19, 0x29, 0x46, //-R -
0x46, 0x49, 0x49, 0x49, 0x31, //-S -
0x01, 0x01, 0x7f, 0x01, 0x01, //-T -
0x3f, 0x40, 0x40, 0x40, 0x3f, //-U -
0x1f, 0x20, 0x40, 0x20, 0x1f, //-V -
0x3f, 0x40, 0x38, 0x40, 0x3f, //-W -
0x63, 0x14, 0x08, 0x14, 0x63, //-X -
0x07, 0x08, 0x70, 0x08, 0x07, //-Y -
0x61, 0x51, 0x49, 0x45, 0x43, //-Z -
0x00, 0x7f, 0x41, 0x41, 0x00, //-[ -
0x02, 0x04, 0x08, 0x10, 0x20, //-\ -
0x00, 0x41, 0x41, 0x7f, 0x00, //-] -
0x04, 0x02, 0x01, 0x02, 0x04, //^- -
0x40, 0x40, 0x40, 0x40, 0x40, //~ - -
0x01, 0x02, 0x04, 0x00, 0x00, //` - -
0x20, 0x54, 0x54, 0x54, 0x78, //-a -
0x7f, 0x48, 0x48, 0x48, 0x30, //-b -
0x38, 0x44, 0x44, 0x44, 0x44, //-c -
0x30, 0x48, 0x48, 0x48, 0x7f, //-d -
0x38, 0x54, 0x54, 0x54, 0x58, //-e -
0x00, 0x08, 0x7e, 0x09, 0x02, //-f -
0x48, 0x54, 0x54, 0x54, 0x3c, //-g -
0x7f, 0x08, 0x08, 0x08, 0x70, //-h -
0x00, 0x00, 0x7a, 0x00, 0x00, //-i -
0x20, 0x40, 0x40, 0x3d, 0x00, //-j -
0x7f, 0x20, 0x28, 0x44, 0x00, //-k -
0x00, 0x41, 0x7f, 0x40, 0x00, //-l -
0x7c, 0x04, 0x38, 0x04, 0x7c, //-m -
0x7c, 0x08, 0x04, 0x04, 0x78, //-n -

```



串行电路图



7.5.2、以下为串行接口方式范例程序

与并行方式相比较，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：

```
//-----
sbit lcd_cs1    =P3^4;        //接口定义，CS:片选
sbit lcd_reset  =P3^5;        //接口定义，RESET:复位
sbit lcd_rs     =P3^3;        //接口定义，RS:命令/数据寄存器选择。也叫“A0”，或“CD”
sbit lcd_sid    =P1^7;        //接口定义，SID 即 SDA:串行数据
sbit lcd_sclk   =P1^6;        //接口定义，sclk: 串行时钟
```

```
//-----
//写命令到LCD 模块
void transfer_command(int data1)
```

```

{
    char i;
    lcd_cs1=0;          //cs1=0,片选清零才可以传送命令或数据
    lcd_rs=0;          //rs=0: 表示以下发送 1 个字节的命令
    for(i=0;i<8;i++)
    {
        lcd_sclk=0;
        if(data1&0x80) lcd_sid=1;
        else lcd_sid=0;
        lcd_sclk=1;
        data1<<=1;
    }
    lcd_cs1=1;          //cs1=1,当不用传数据给液晶屏时片选尽量置高,以免接收到一些干扰信号
}
    
```

//写数据到LCD 模块

```
void transfer_data(int data1)
```

```

{
    char i;
    lcd_cs1=0;          //cs1=0,片选清零才可以传送命令或数据
    lcd_rs=1;          //rs=0: 表示以下发送 1 个字节的命令
    for(i=0;i<8;i++)
    {
        lcd_sclk=0;
        if(data1&0x80) lcd_sid=1;
        else lcd_sid=0;
        lcd_sclk=1;
        data1<<=1;
    }
    lcd_cs1=1;          //cs1=1,当不用传数据给液晶屏时片选尽量置高,以免接收到一些干扰信号
}
    
```

