

JLX19296G-923-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 | 3~5 |
| 4 | 基本原理 | 5~6 |
| 5 | 技术参数 | 6 |
| 6 | 时序特性 | 6~11 |
| 7 | 指令功能及硬件接口与编程案例 | 12~尾页 |

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 JLX19296G-923 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。

JLX19296G-923 可以显示 192 列*96 行点阵单色图片，或显示 16*16 点阵的汉字 12 个*6 行，或显示 8*16 点阵的英文、数字、符号 24 个*6 行。或显示 5*8 点阵的英文、数字、符号 32 个*12 行。

2. JLX19296G-923 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 ST75256, 功能强大，稳定性好

2.3 功耗低：不带背光 2mW (3.3V* (0.3mA 参考值))，带背光不大于 185mW (3.0V*61mA)；

2.4 显示内容：

(1) 192*96 点阵单色图片，或其它小于 192*96 点阵的单色图片；

(2) 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 12 字*6 行；

(3) 按照 8*16 点阵汉字来计算可显示 24 字*6 行；

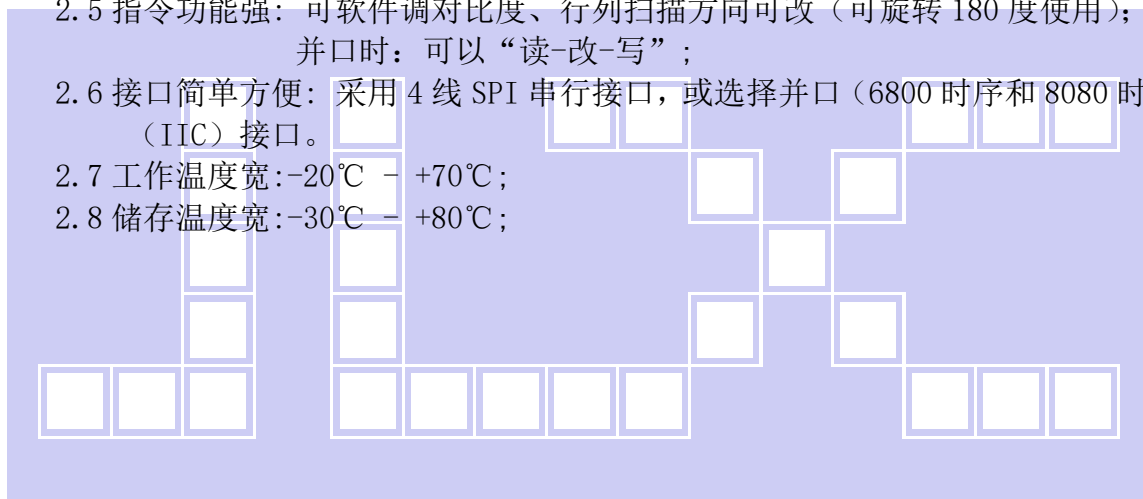
(4) 按照 5*8 点阵汉字来计算可显示 32 字*12 行；

2.5 指令功能强：可软件调对比度、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：采用 4 线 SPI 串行接口，或选择并口（6800 时序和 8080 时序可选），或 I2C（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

3.2 模块的接口引脚功能

3.2.1 并行时接口引脚功能

| 引线号 | 符号 | 名称 | 功能 |
|-------|--------|---------|------------------------------------|
| 1 | VG | 偏压电路 | LCD 偏置驱动电压, VG 与 VSS 之间接一个电容 |
| 2 | V0 | 倍压电路 | V0 与 XV0 之间接一个电容 |
| 3 | XV0 | 倍压电路 | |
| 4 | CA1P | 倍压电路 | |
| 5 | CA1N | 倍压电路 | CA1P 与 CA1N 之间接一个电容 |
| 6 | VDD | 供电电源正极 | 供电电源正极 |
| 7 | VSS | 供电电源负极 | 供电电源负极 |
| 8 | COMSCN | 镜像选择指令 | 默认接 VDD, (镜像 180 度接 VSS) |
| 9 | IF2 | 模式选择 | L: 接低电平 |
| 10 | IF1 | 模式选择 | H: 接高电平 |
| 11 | IF0 | 模式选择 | L: 接低电平 |
| 12 | CS | 片选 | 低电平片选 |
| 13 | 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 "CD") |
| 14 | E (RD) | 使能信号 | 6800 时序: 使能信号 |
| 15 | RW(WR) | 读/写 | 6800 时序: H: 读数据 0: 写数据 |
| 16 | RST | 复位 | 低电平复位, 复位完成后, 回到高电平, 液晶屏开始工作 |
| 17~24 | DO~D7 | I/O | 并行接口时, 数据总线 DB0~DB7 |
| 25 | EXTB | 空脚 | 空脚 |
| 26 | VPP | 空脚 | 空脚 |

表 1: 模块并行接口引脚功能

3.2.2 四线串行时接口引脚功能

| 引线号 | 符号 | 名称 | 功能 |
|-------|------------------|---------|------------------------------------|
| 1 | VG | 偏压电路 | LCD 偏置驱动电压, VG 与 VSS 之间接一个电容 |
| 2 | V0 | 倍压电路 | V0 与 XV0 之间接一个电容 |
| 3 | XV0 | 倍压电路 | |
| 4 | CA1P | 倍压电路 | |
| 5 | CA1N | 倍压电路 | CA1P 与 CA1N 之间接一个电容 |
| 6 | VDD | 供电电源正极 | 供电电源正极 |
| 7 | VSS | 供电电源负极 | 供电电源负极 |
| 8 | COMSCN | 镜像选择指令 | 默认接 VDD, (镜像 180 度接 VSS) |
| 9 | IF2 | 模式选择 | L: 接低电平 |
| 10 | IF1 | 模式选择 | L: 接低电平 |
| 11 | IF0 | 模式选择 | L: 接低电平 |
| 12 | CS | 片选 | 低电平片选 |
| 13 | 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 "CD") |
| 14 | E (RD) | 使能信号 | 串行接口, RD 接高电平 |
| 15 | RW(WR) | 读、写 | 串行接口, RW 接高电平 |
| 16 | RST | 复位 | 低电平复位, 复位完成后, 回到高电平, 液晶屏开始工作 |
| 17 | DO(SCL) | I/O | 串行时钟 (SCL) |
| 18~20 | D1 ~ D3 (SDA) | I/O | 串行数据 (D1、D2、D3 短接一起作为 SDA) |

| | | | |
|-------|-------|-----|-----------------------|
| 21~24 | D4-D7 | I/O | 串行接口, D4-D7 引脚建议接 VDD |
| 25 | EXTB | 空脚 | 空脚 |
| 26 | VPP | 空脚 | 空脚 |

表 2: 4 线 SPI 串行接口引脚功能

3.2.3 I²C 总线时接口引脚功能

| 引线号 | 符号 | 名称 | 功能 |
|-------|------------------|---------|------------------------------|
| 1 | VG | 偏压电路 | LCD 偏置驱动电压, VG 与 VSS 之间接一个电容 |
| 2 | V0 | 倍压电路 | V0 与 XV0 之间接一个电容 |
| 3 | XV0 | 倍压电路 | |
| 4 | CA1P | 倍压电路 | CA1P 与 CA1N 之间接一个电容 |
| 5 | CA1N | 倍压电路 | |
| 6 | VDD | 供电电源正极 | 供电电源正极 |
| 7 | VSS | 供电电源负极 | 供电电源负极 |
| 8 | COMSCN | 镜像选择指令 | 默认接 VDD, (镜像 180 度接 VSS) |
| 9 | IF2 | 模式选择 | L: 接低电平 |
| 10 | IF1 | 模式选择 | L: 接低电平 |
| 11 | IF0 | 模式选择 | H: 接高电平 |
| 12 | CS | 片选 | I2C 接口, 此引脚接 VSS |
| 13 | A0(RS) | 寄存器选择信号 | I2C 接口, 此引脚接高电平 |
| 14 | E(RD) | 使能信号 | I2C 接口, 不用, 此引脚接高电平 |
| 15 | RW(WR) | 读、写 | I2C 接口, 不用, 此引脚接高电平 |
| 16 | RST | 复位 | 低电平复位, 复位完成后, 回到高电平, 液晶屏开始工作 |
| 17 | D0(SCL) | I/O | 串行时钟 (SCL) |
| 18~20 | D1 ~ D3 (SDA) | I/O | 串行数据 (D1、D2、D3 短接一起作为 SDA) |
| 21~22 | D4-D5 | I/O | I2C 接口, D4-D5 引脚接 VDD |
| 23~24 | D6-D7 | I/O | I2C 接口, D6-D7 是从属地址接 VSS |
| 25 | EXTB | 空脚 | 空脚 |
| 26 | VPP | 空脚 | 空脚 |

 表 3: I²C 总线接口引脚功能

4. 基本原理

4.1 液晶屏 (LCD)

在 LCD 上排列着 192×96 点阵, 192 个列信号与驱动 IC 相连, 96 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 工作电路:

图 2 是 JLX19296G-923 图像点阵型模块的电路框图, 它由驱动 IC ST75256 及几个电阻电容组成。
电路框图

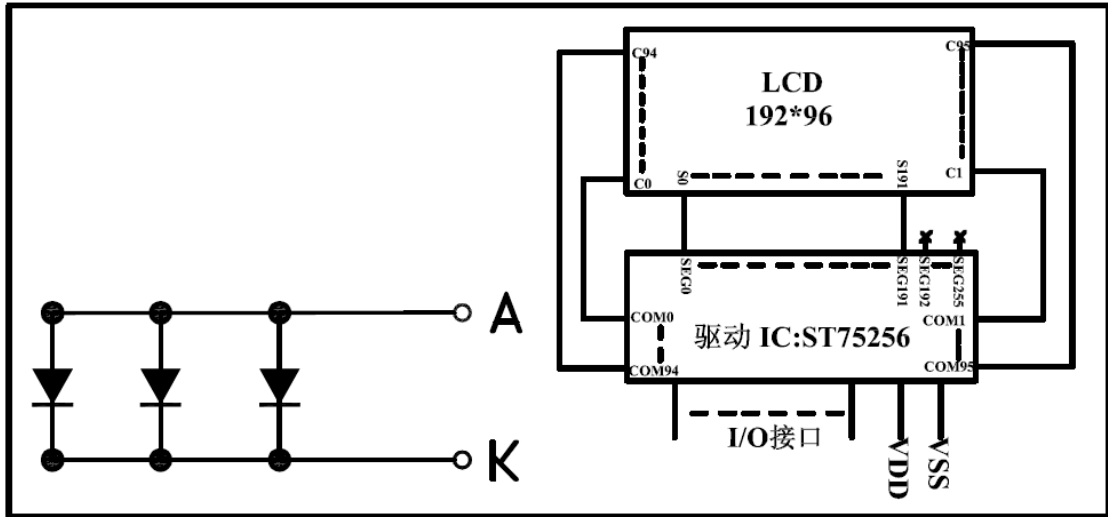


图 2: JLX19296G-923 图像点阵型液晶模块的电路框图

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:

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

存储温度: $-30 \sim +80^{\circ}\text{C}$;

背光灯白色。

正常工作电流为: $24 \sim 60\text{mA}$ (LED 灯数共 3 颗);

工作电压: 3.0V ; (注意: 接 3.3V 加 20Ω , 接 5.0V 加 120Ω 限流电阻)

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 | VDD - V0 | -0.3 | — | 12 | V |
| 静电电压 | | — | — | 100 | V |
| 工作温度 | | -20 | — | +70 | $^{\circ}\text{C}$ |
| 储存温度 | | -30 | — | +80 | $^{\circ}\text{C}$ |

表 4: 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 | VIH | — | 0.8VDD | — | VDD | V |
| 输入低电平 | VIO | — | 0 | — | 0.2VDD | V |
| 输出高电平 | VOH | $\text{IOH} = 0.2\text{mA}$ | 0.8VDD | — | VDD | V |
| 输出低电平 | VOO | $\text{IOO} = 1.2\text{mA}$ | 0 | — | 0.2VDD | V |
| 模块工作电流 | IDD | VDD = 3.0V | — | 0.3 | 1.0 | mA |
| 背光工作电流 | ILED | VLED = 3.0V | 24 | 45 | 60 | mA |

表 5: 直流 (DC) 参数

6. 读写时序特性 (AC 参数)

6.1 4 线 SPI 串行接口写时序特性 (AC 参数)

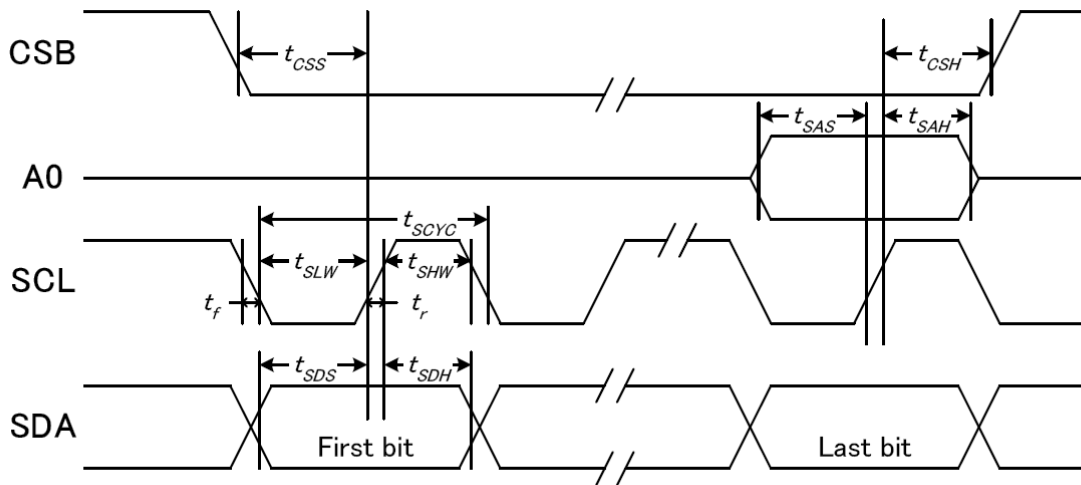


图 3. 从 CPU 写到 ST75256 (Writing Data from CPU to ST75256)

表 6. 写数据到 ST75256 的时序要求

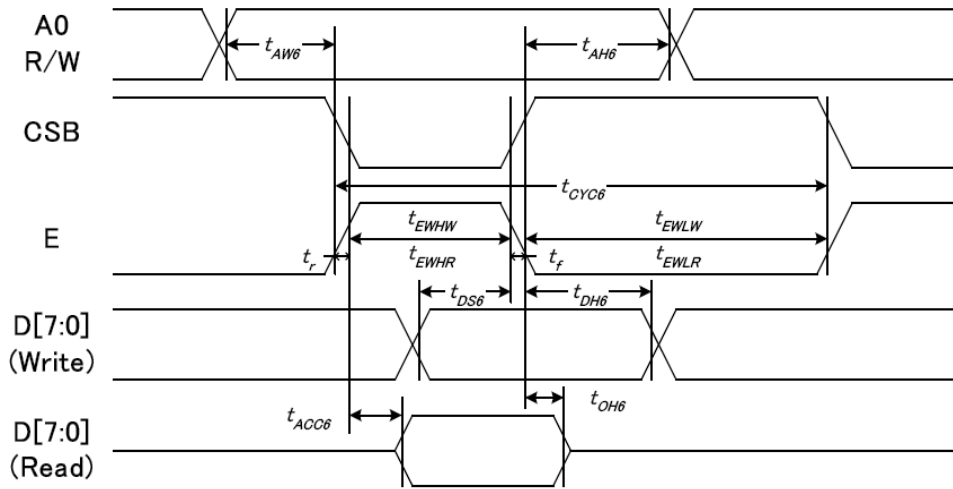
| 项目 | 符号 | 测试条件 | 极限值 | | | 单位 |
|---|-------|---------|-----|------|-----|----|
| | | | MIN | TYPE | MAX | |
| 4线 SPI串口时钟周期 (4-line SPI Clock Period) | tSCYC | | 80 | -- | -- | ns |
| 保持SCK高电平脉宽 (SCL "H" pulse width) | tSHW | 引脚: SCL | 30 | -- | -- | ns |
| 保持SCLK低电平脉宽 (SCL "L" pulse width) | tSLW | | 30 | -- | -- | ns |
| 地址建立时间 (Address setup time) | tSAS | 引脚: A0 | 20 | -- | -- | ns |
| 地址保持时间 (Address hold time) | tSAH | | 20 | -- | -- | ns |
| 数据建立时间 (Data setup time) | tSDS | 引脚: SID | 20 | -- | -- | ns |
| 数据保持时间 (Data hold time) | tSDH | | 20 | -- | -- | ns |
| 片选信号建立时间 (CS-SCL time) | tCSS | 引脚: CSB | 20 | -- | -- | ns |
| 片选信号保持时间 (CS-SCL time) | tCSH | | 20 | -- | -- | ns |

VDD = 1.8~3.3V ± 5%, Ta = -30~85°C

输入信号的上升和下降时间 (TR, TF) 在 15 纳秒或更少的规定。

所有的时间, 用 20%和 80%作为标准规定的测定。

6.2 6800 时序并行接口的时序特性 (AC 参数)



从 CPU 写到 ST75256 (Writing Data from CPU to ST75256)

图 4. 写数据到 ST75256 的时序要求 (6800 系列 MPU)

表 7. 读写数据的时序要求

| 项目 | 符号 | 名称 | 极限值 | | | 单位 |
|-----------|----------|--------------------|-----|------|-----|----|
| | | | MIN | TYPE | MAX | |
| 地址保持时间 | A0 | t _{AH6} | 20 | | — | ns |
| 地址建立时间 | | t _{AW6} | 0 | | — | ns |
| 系统循环时间 | E | t _{CYC6} | 160 | | — | ns |
| 使能“低”脉冲宽度 | | t _{EHLW} | 70 | | — | ns |
| 使能“高”脉冲宽度 | | t _{EHWLW} | 70 | | — | ns |
| 写数据建立时间 | DB[7: 0] | t _{DS6} | 15 | | — | ns |
| 写数据保持时间 | | t _{DH6} | 15 | | — | ns |

VDD = 1.8~3.3V ± 5%, Ta = -30~85°C

输入信号的上升时间和下降时间 (TR, TF) 是在 15 纳秒或更少的规定。当系统循环时间非常快,

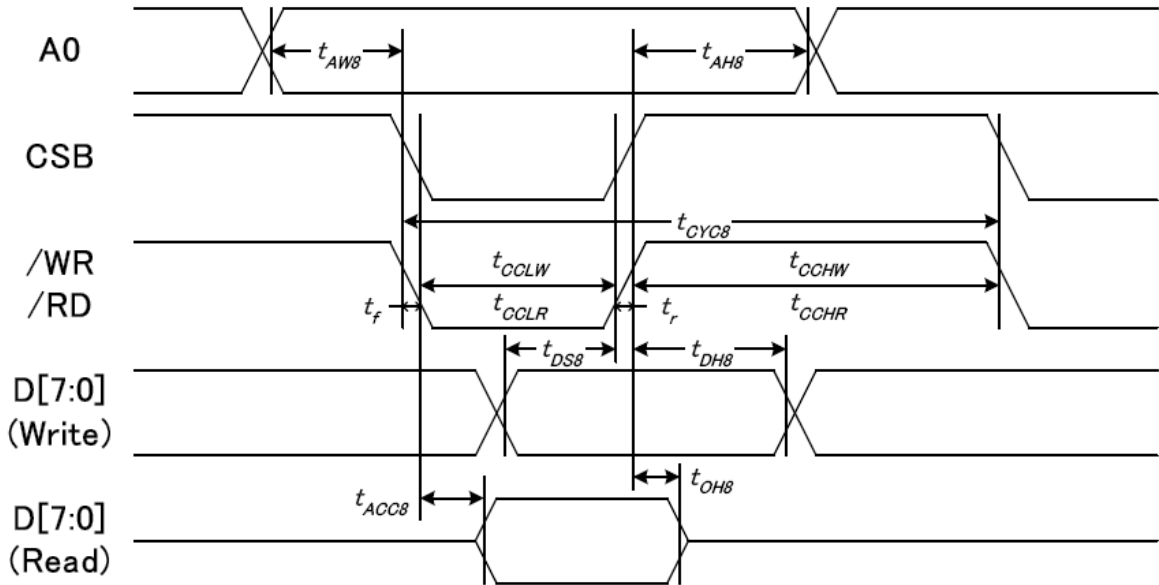
$(TR + TF) \leq (tcyc6 - tewlw - tewhw)$ 指定。

所有的时间, 用 20%和 80%作为参考指定的测定。

tewlw 指定为重叠的 CSB “H” 和 “L”。

R / W 信号总是 “H”

6.3 8080 时序并行接口的时序特性 (AC 参数)



从 CPU 写到 ST75256 (Writing Data from CPU to ST75256)

图 5. 写数据到 ST75256 的时序要求 (8080 系列 MPU)

表 8. 读写数据的时序要求

| 项目 | 符号 | 名称 | 极限值 | | | 单位 |
|-----------|-----|-------------------|-----|------|-----|----|
| | | | MIN | TYPE | MAX | |
| 地址保持时间 | A0 | t _{AH8} | 20 | | — | ns |
| 地址建立时间 | | t _{AW8} | 0 | | — | ns |
| 系统循环时间 | /WR | t _{CYC8} | 160 | | — | ns |
| 使能“低”脉冲宽度 | | t _{CCLW} | 70 | | — | ns |
| 使能“高”脉冲宽度 | | t _{CCHW} | 70 | | — | ns |
| 写数据建立时间 | DB | t _{DS8} | 15 | | — | ns |
| 写数据保持时间 | | t _{DH8} | 15 | | — | ns |

VDD = 1.8~3.3V ± 5%, Ta = -30~85°C

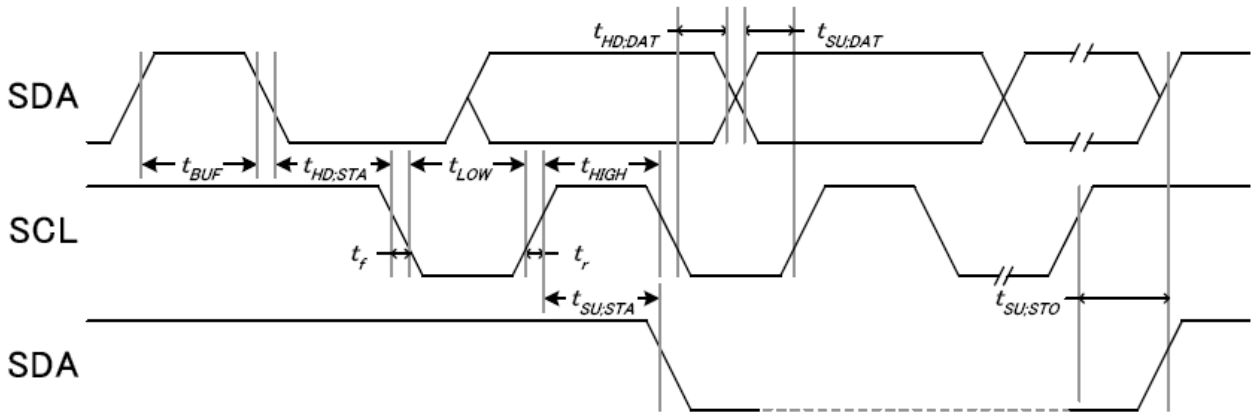
输入信号的上升时间和下降时间 (TR, TF) 是在 15 纳秒或更少的规定。当系统循环时间非常快,

$(TR + TF) \leq (tcyc8 - tcclw - tcchw)$ 指定。

所有的时间, 用 20%和 80%作为参考指定的测定。

tcclw 被指定为“L”之间的重叠 CSB 和 / WR 处于“L”级

6.3 I²C 接口的时序特性 (AC 参数)



从 CPU 写到 ST75256 (Writing Data from CPU to ST75256)

 图 6. 写数据到 ST75256 的时序要求 (I²C 系列 MPU)

表 9. 读写数据的时序要求

| 项目 | 符号 | 名称 | 极限值 | | | 单位 |
|------------------|-----|----------|----------|------|-----|-----|
| | | | MIN | TYPE | MAX | |
| SCL时钟频率 | CSL | FCLK | --- | | 400 | kUZ |
| SCL时钟的低周期 | CSL | TLOW | 1.3 | | --- | us |
| SCL时钟周期 | CSL | THIGH | 0.6 | | --- | us |
| 数据保持时间 | SDA | TSU;Data | 0.1 | | --- | ns |
| 数据建立时间 | SDA | THD;Data | 0 | | 0.9 | us |
| SCL, SDA 的上升时间 | SCL | TR | 20+0.1Cb | | 300 | ns |
| SCL, SDA 下降时间 | SCL | TF | 20+0.1Cb | | 300 | ns |
| 每个总线为代表的电容性负载 | | Cb | --- | | 400 | pF |
| 一个重复起始条件设置时间 | SDA | TSU;SUA | 0.6 | | --- | us |
| 启动条件的保持时间 | SDA | THD;STA | 0.6 | | --- | us |
| 为停止条件建立时间 | | TSU;STO | 0.6 | | --- | us |
| 容许峰值宽度总线 | | TSW | --- | | 50 | ns |
| 开始和停止条件之间的总线空闲时间 | SCL | TBUF | 0.1 | | | us |

所有的时间, 用 20%和 80%作为标准规定的测定。

这是推荐的操作 I C 接口与 VDD1 高于 2.6V。

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

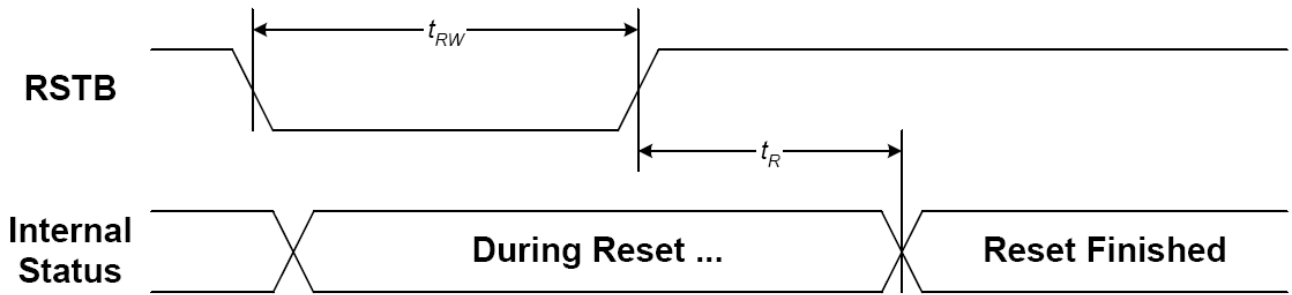
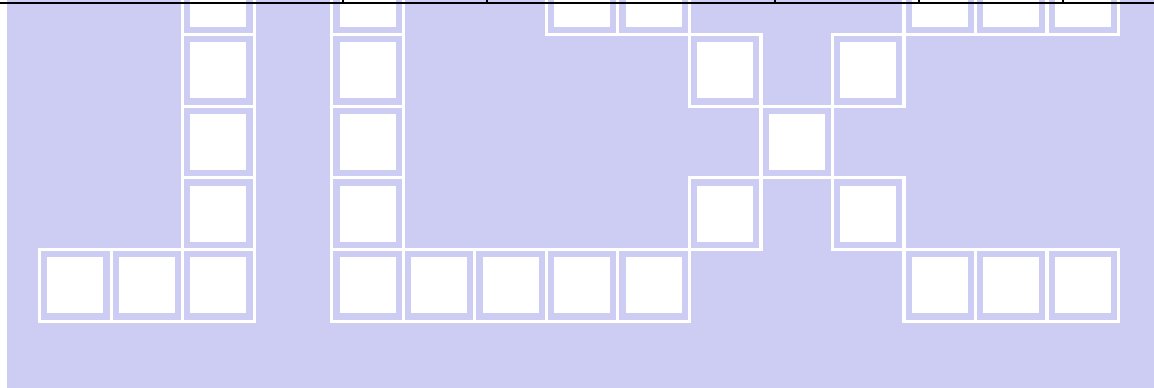


图 7: 电源启动后复位的时序

表 10: 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_{RW} | | 100 | -- | -- | us |
| 复位保持低电平的时间 | T_{RD} | 引脚: RESET, WR | 100 | -- | -- | ms |



7. 指令功能:

7.1 指令表

表 11

| 指令名称 | 指令码 | | | | | | | | | | |
|--|-----|-----|------|------|------|------|------|------|------|------|--|
| | RS | R/W | DB7 | DB6 | DB5 | DB4 | DB3 | DB2 | DB1 | DB0 | |
| (1)扩展指令1 | 0 | 0 | 0 | 0 | 1 | 1 | EXT1 | 0 | 0 | EXT0 | 扩展指令 1、2、3、4 OX30:扩展指令 1 |
| Ext[1:0]=0,0(Extension Command1/扩展指令 1) OX30 扩屏指令 1 一定要调用 OX30 才能用扩展指令 1 | | | | | | | | | | | |
| (2)显示开/关 (display on/off) | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 显示开/关: 1 OXAE:关, OXAF: 开 |
| (3)正显/反显 (Inverse Display) | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 显示正显/反显 1 OXA6:正显, 正常 OXA7: 反显 |
| (4)所有点阵开/关 (All Pixel ON/OFF) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | OX22: 所有点阵关 1 OX23: 所有点阵开 |
| (5) 控制液晶屏显示 (Display Control) | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | OXCA:显示控制 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | CLD | 0 | 0 | OX00:设置 CL 驱动频率: CLD=0 |
| | 1 | 0 | DT7 | DT6 | DT5 | DT4 | DT3 | DT2 | DT1 | DT0 | OX7F:点空比: Duty=128 |
| | 1 | 0 | 0 | 0 | LF4 | F1 | LF3 | LF2 | LF1 | LF0 | OX20:帧周期 |
| (6)省电模式 (Power save) | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | SLP | OX94: SLP=0, 退出睡眠模式 OX95: SLP=1, 进入睡眠模式 |
| (7)页地址设置 (Set Page Address) | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | OX75: 页地址设置 |
| | 1 | 0 | YS7 | YS6 | YS5 | YS4 | YS3 | YS2 | YS1 | YS0 | OX00: 起始页地址 |
| | 1 | 0 | YE7 | YE6 | YE5 | YE4 | YE3 | YE2 | YE1 | YE0 | OX1F: 结束页地址, 每 4 行为 1 页 |
| (8)列地址设置 (Set Column Address) | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | OX15: 列地址设置 |
| | 1 | 0 | XS7 | XS6 | XS5 | XS4 | XS3 | XS2 | XS1 | XS0 | OX00: 起始列地址 |
| | 1 | 0 | XE7 | XE6 | XE5 | XE4 | XE3 | XE2 | XE1 | XE0 | OXFF: 结束列地址 XE=256 |
| (9)行列扫描方向 (Data Scan Direction) | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | OXBC: 行列扫描方向 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | MV | MX | MY | OX00: MX、MY=Normal |
| (10)写数据到液晶屏 (Write Data) | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | OX5C: 写数据 |
| | 1 | 0 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 8 位显示数据 |
| (11)读液晶屏显示数据 (Read Data) | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | OX5D: 读数据 |
| | 1 | 1 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 8 位显示数据 |
| (12)指定区域显示数据 (Partial In) | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | OXA8: 指定显示区域 |
| | 1 | 0 | PTS7 | PTS6 | PTS5 | PTS4 | PTS3 | PTS2 | PTS1 | PTS0 | 起始区域地址: 00h≤PTS≥A1h |
| | 1 | 0 | PTE7 | PTE6 | PTE5 | PTE4 | PTE3 | PTE2 | PTE1 | PTE0 | 结束区域地址: 00h≤PTE≥A1h |
| (13) 退出指定区域显示 (Partial Out) | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | OXA9: 退出指定区域显示 |
| (14)读/改/写 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | OXE0: 进入读/改/写 |
| (15)退出读/改/写 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | OXEE: 退出读/改/写 |
| (16)指定显示滚动区域 (Scroll Area) | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | OXAA: 滚动区域设置 |
| | 1 | 0 | TL7 | TL6 | TL5 | TL4 | TL3 | TL2 | TL1 | TL0 | TL[7:0]:起始区域地址 |
| | 1 | 0 | BL7 | BL6 | BL5 | BL4 | BL3 | BL2 | BL1 | BL0 | BL[7:0]:结束区域地址 |
| | 1 | 0 | NSL7 | NLS6 | NLS5 | NLS4 | NLS3 | NLS2 | NLS1 | NLS0 | NSL[7:0]:指定行数 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | SCM1 | SCM0 | SCM[1:0]:显示模式 |
| (17)显示初始行设置 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | OXAB: 滚动开始初始行设置 |



| | | | | | | | | | | | |
|--|---|---|-----|-----|------|------|------|------|------|------|--|
| (Set Start Line) | 1 | 0 | SL7 | SL6 | SL5 | SL4 | SL3 | SL2 | SL1 | SL0 | 00h≤SL≥A1h |
| (18)开振荡电路 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0XD1: 开内部振荡电路 |
| (19)关振荡电路 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0XD2: 关内部振荡电路 |
| (20)电源控制 (Power Control) | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0X20: 电源控制 |
| | 1 | 0 | 0 | 0 | 0 | 0 | VB | 0 | VF | VR | 0X0B: VB、VF、VR=1 |
| (21)液晶内部电压设置 (Set Vop) | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0X81: 设置对比度 |
| | 1 | 0 | 0 | 0 | Vop5 | Vop4 | Vop3 | Vop2 | Vop1 | Vop0 | 0X26: 微调对比度, 范围 0X00-0XFF |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | Vop7 | Vop6 | Vop5 | 0X04: 粗调对比度, 范围 0X00-0X07 先微调再粗调, 顺序不能变 |
| (22)液晶内部电压控制 (Vop Control) | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | VOL | 0XD6: VOP 每格增加 0.04V 0XD7: VOP 每格减少 0.04V |
| (23)读寄存器模式 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | REG | 0X7C: 读寄存器值 Vop[5:0] 0X7D: 读寄存器值 Vop[8:6] |
| (24)空操作 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0X25: 空操作 |
| (25)读状态 (并行、IIC) | 0 | 1 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | 读状态字节 |
| (26)读状态 (串行接口) | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 读状态字节 |
| | 0 | 1 | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
| (27)数据格式选择 (Data Format Select) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | D0 | 0 | 0 | 0X08: 数据 D7→D0 0X0C: 数据 D0→D7 |
| (28)显示模式 (Display Mode) | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0XF0: 显示模式设置 |
| | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | DM | 0X10: 黑白模式 0X11: 4 灰级度模式 |
| (29)ICON设置 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | ICON | 0X77: 使能 ICON RAM 0X76: 禁用 ICON RAM |
| (30)设置主/从模式 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | MS | 0X6E: 主模式(使用主模式) 0X6F: 从模式 |
| Ext[1:0]=0,1(Extension Command 2) 0X31 扩屏指令 2 一定要调用 0X31 才能用扩展指令 2 | | | | | | | | | | | |
| (31)灰度设置 Set Gray Level | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0X20: 灰度级设置 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | GL[4:0]: 浅灰度级设置 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | GD[4:0]: 深灰度级设置 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 1 | 0 | 0 | 0 | 0 | GL4 | GL3 | GL2 | GL1 | GL0 | |
| | 1 | 0 | 0 | 0 | 0 | GL4 | GL3 | GL2 | GL1 | GL0 | |
| | 1 | 0 | 0 | 0 | 0 | GL4 | GL3 | GL2 | GL1 | GL0 | |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 1 | 0 | 0 | 0 | 0 | GD4 | GD3 | GD2 | GD1 | GD0 | |
| | 1 | 0 | 0 | 0 | 0 | GD4 | GD3 | GD2 | GD1 | GD0 | |
| | 1 | 0 | 0 | 0 | 0 | GD4 | GD3 | GD2 | GD1 | GD0 | |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |
| (32)LCD偏压比设置 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0X32: 偏压比设置 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | |



| | | | | | | | | | | | |
|--|---|---|------|------|-----------|------|------|------|------|------|---|
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BE1 | BE0 | 0x01: 升压电容频率 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | BS2 | BS1 | BS0 |
| (33)升压倍数 (Booster Level) | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0x51: 内建升压倍数设置 |
| | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | BST | 0x7B: 10 倍 |
| (34)电压驱动选择 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | DS | 0x41: LCD 内部升压 |
| (35)自动读取控制 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | XARD=0: 使能自动读 |
| | 1 | 0 | 1 | 0 | 0 | XARD | 1 | 1 | 1 | 1 | XARD=0: 不使能自动读 |
| (36)控制OTP读写 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0xe0: OTP 读写 |
| | 1 | 0 | 0 | 0 | ER/ RD | 0 | 0 | 0 | 0 | 0 | WR/RD=0; 0x00, 使能 OTP 读 ER/RD=1; 0x20, 使能 OTP 写 |
| (37)控制OTP出 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 控制 OTP 出 |
| (38)写OTP | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 写 OTP |
| (39)读OTP | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 读 OTP |
| (40)OTP选择控制 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0xe4: OTP 选择控制 |
| | 1 | 0 | 1 | Ctrl | 0 | 0 | 1 | 0 | 0 | 1 | Ctrl=1: 0xc9, 不使能 OTP Ctrl=0: 0x89, 使能 OTP |
| (41)OTP程序设置 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | OTP 程序设置 |
| | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | |
| (42) 帧速率 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0xf0: 帧速率设置在不同的温度范 |
| | 1 | 0 | 0 | 0 | 0 | FRA4 | FRA3 | FRA2 | FRA1 | FRA0 | 围 |
| | 1 | 0 | 0 | 0 | 0 | FRB4 | FRB3 | FRB2 | FRB1 | FRB0 | |
| | 1 | 0 | 0 | 0 | 0 | FRC4 | FRC3 | FRC2 | FRC1 | FRC0 | |
| | 1 | 0 | 0 | 0 | 0 | FRD4 | FRD3 | FRD2 | FRD1 | FRD0 | |
| (43) 温度范围 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0xf2: 温度范围设置 |
| | 1 | 0 | 0 | TA6 | TA5 | TA4 | TA3 | TA2 | TA1 | TA0 | |
| | 1 | 0 | 0 | TB6 | TB5 | TB4 | TB3 | TB2 | TB1 | TB0 | |
| | 1 | 0 | 0 | TC6 | TC5 | TC4 | TC3 | TC2 | TC1 | TC0 | |
| (44) 温度梯度补偿 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0xf4: 温度补偿系数设置 |
| | 1 | 0 | MT13 | MT12 | MT11 | MT10 | MT03 | MT02 | MT01 | MT00 | |
| | 1 | 0 | MT33 | MT32 | MT31 | MT30 | MT23 | MT22 | MT21 | MT20 | |
| | 1 | 0 | MT53 | MT52 | MT51 | MT50 | MT43 | MT42 | MT41 | MT40 | |
| | 1 | 0 | MT73 | MT72 | MT71 | MT70 | MT63 | MT62 | MT61 | MT60 | |
| | 1 | 0 | MT93 | MT92 | MT91 | MT90 | MT83 | MT82 | MT81 | MT80 | |
| | 1 | 0 | MTB3 | MTB2 | MTB1 | MTB0 | MTA3 | MTA2 | MTA1 | MTA0 | |
| | 1 | 0 | MTD3 | MTD2 | MTD1 | MTD0 | MTC3 | MTC2 | MTC1 | MTC0 | |
| | 1 | 0 | MTF3 | MTF2 | MTF1 | MTF0 | MTE3 | MTE2 | MTE1 | MTE0 | |
| Ext[1:0]=1,0(Extension Command 3) 0x38 扩屏指令 3 一定要调用 0X38 才能用扩展指令 3 | | | | | | | | | | | |
| (45) ID 设置 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0xd5: ID 设置 |
| | 1 | 0 | ID7 | ID6 | ID5 | ID4 | ID3 | ID2 | ID1 | ID0 | |
| (46) 读 ID | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | RID | RID=1: 0x7f, 使能 |
| Ext[1:0]=1,1(Extension Command 4) 0x39 扩屏指令 4 一定要调用 0X39 才能用扩展指令 4 | | | | | | | | | | | |
| (47) 使能 OTP | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0xd6: 使能 OTP EOTP=1; 不使能 EOTP, 一般不使能 EOTP EOTP=0; 使能 EOTP |

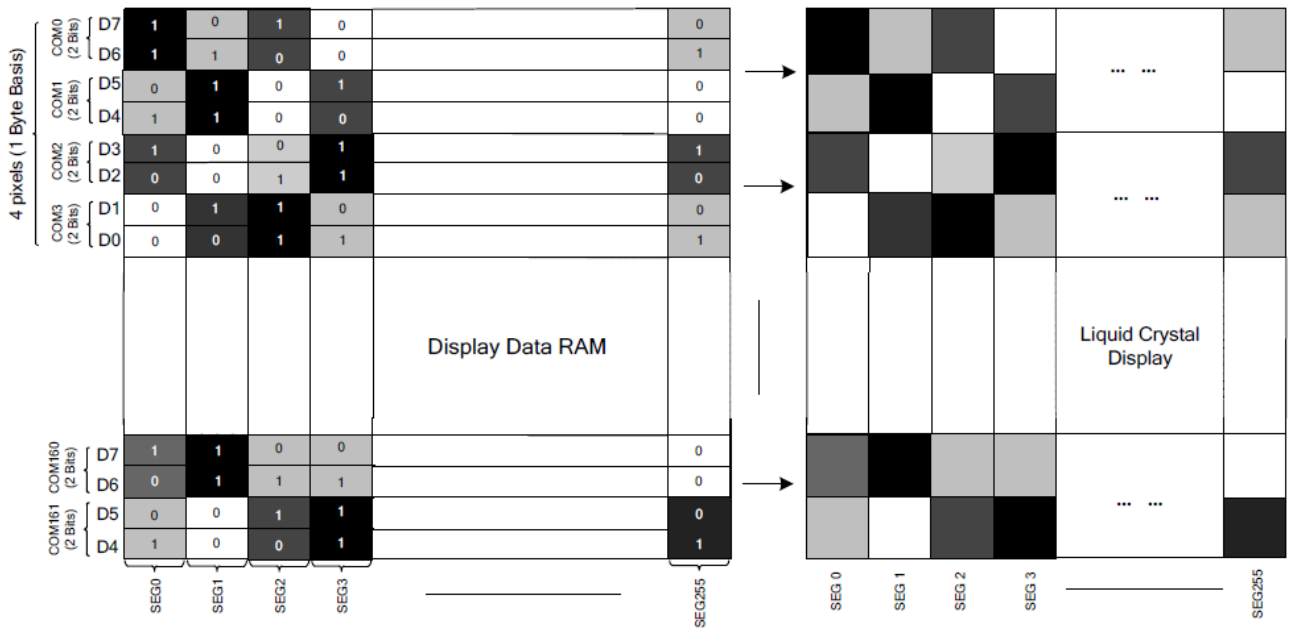
表 8. 指令表

请详细参考 IC 资料”ST75256.PDF”。

7.2 点阵与 DD RAM 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 192*96 点阵的屏分为 12 个“页”, 从第 0 “页” 到第 11 “页”。

DB7—DB0 的排列方向: 数据是从上向下排列的。最高位 D7 是在最上面, 最低位 D0 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:



| 2 Bits Data N=0~3 | | DDRAM | | LCD |
|----------------------|-----|-------|---|------------|
| D2N+1 | D2N | | | |
| 1 | 1 | 1 | 1 | Black |
| 0 | 0 | 0 | 0 | White |
| 1 | 0 | 1 | 0 | Dark Gray |
| 0 | 1 | 0 | 1 | Light Gray |

Figure 21 DDRAM Mapping (4-Level Gray Scale Mode)

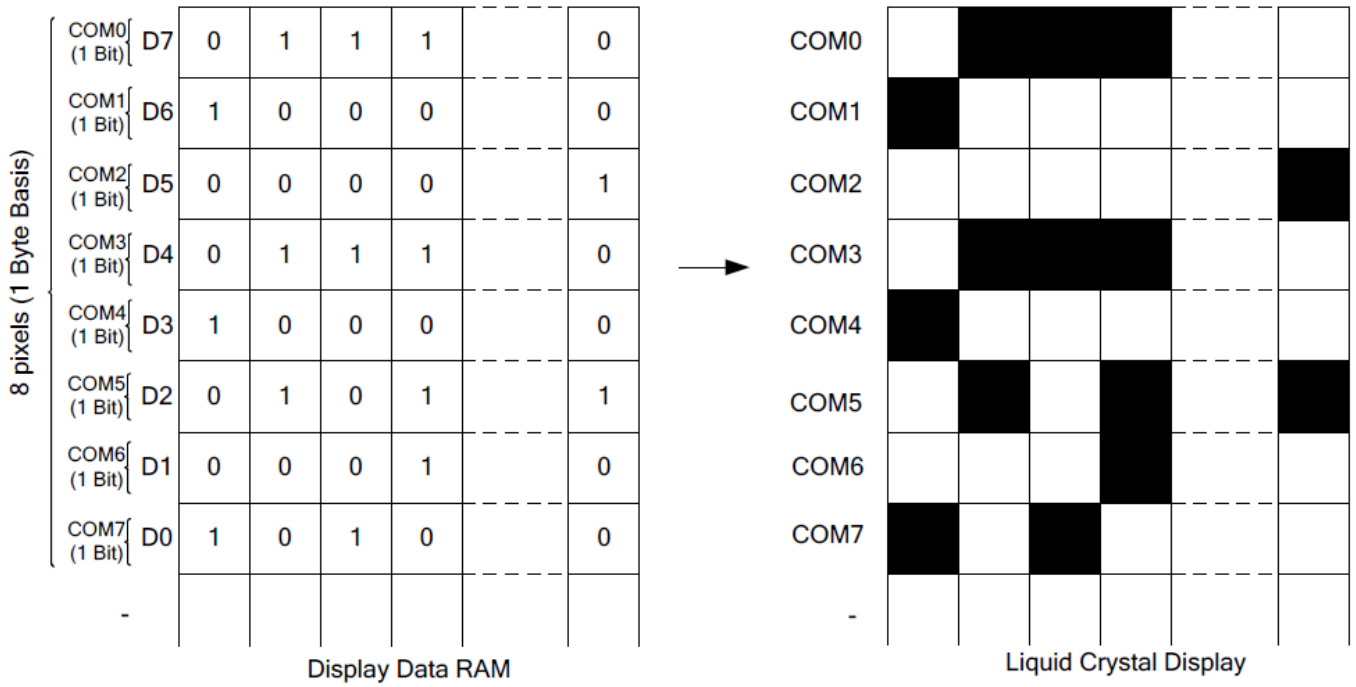
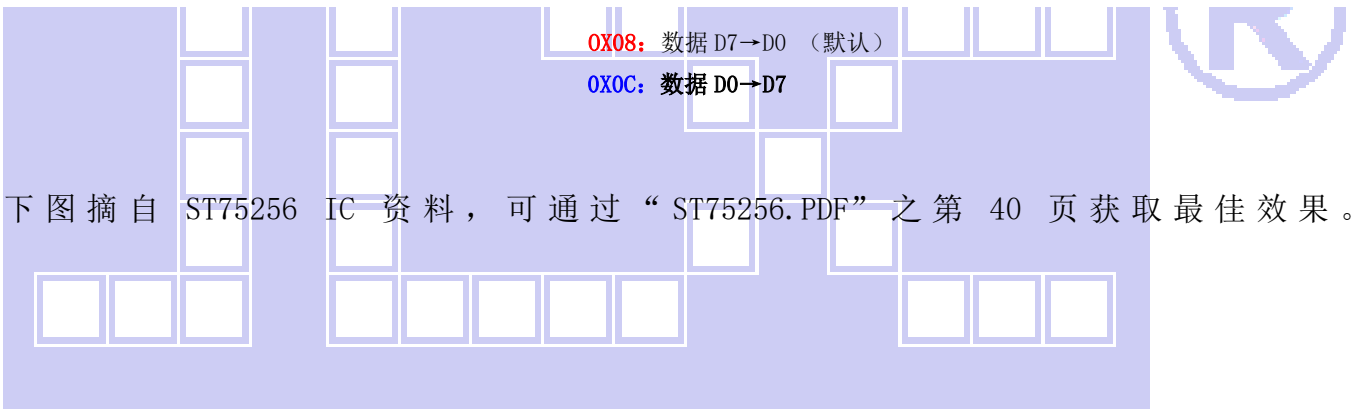


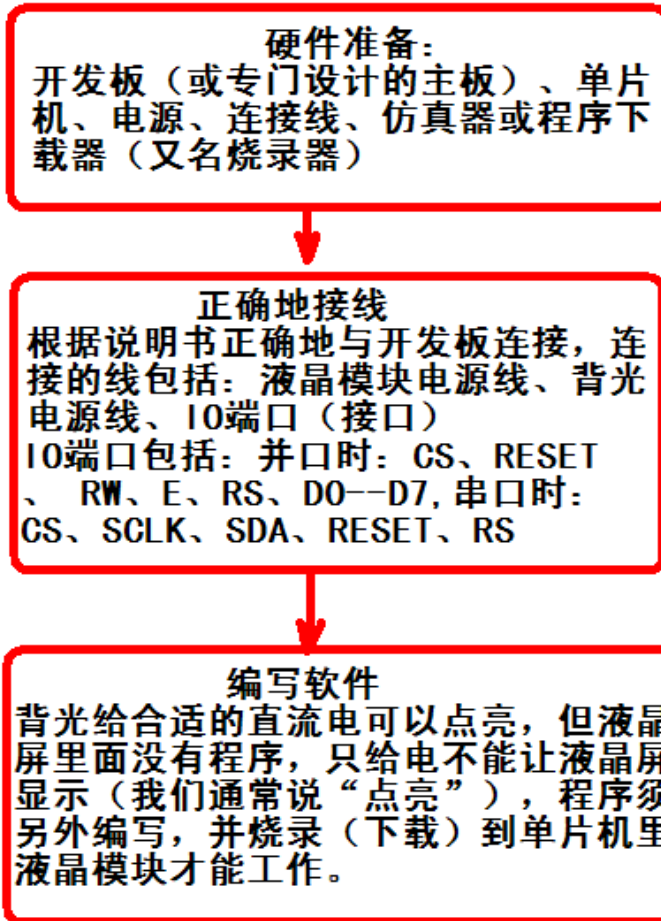
Figure 22 DDRAM Mapping (Monochrome Mode)



7.3 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



7.4 程序举例:

7.4.1 串行接口

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:

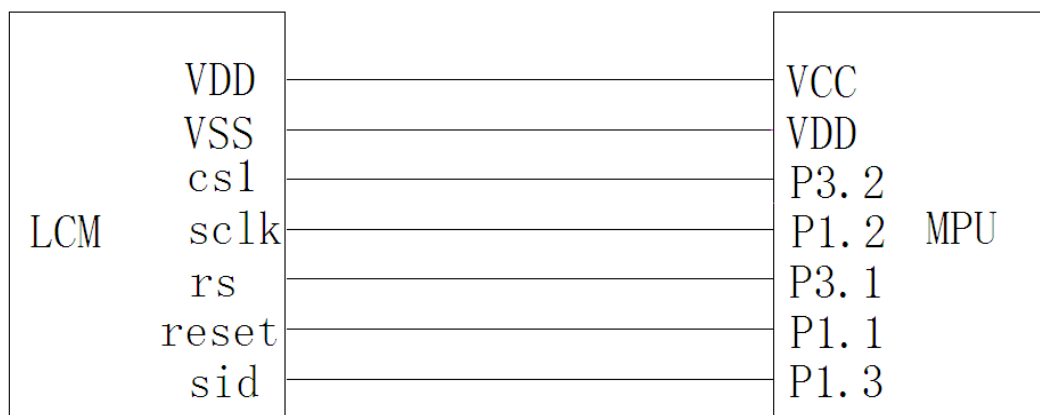
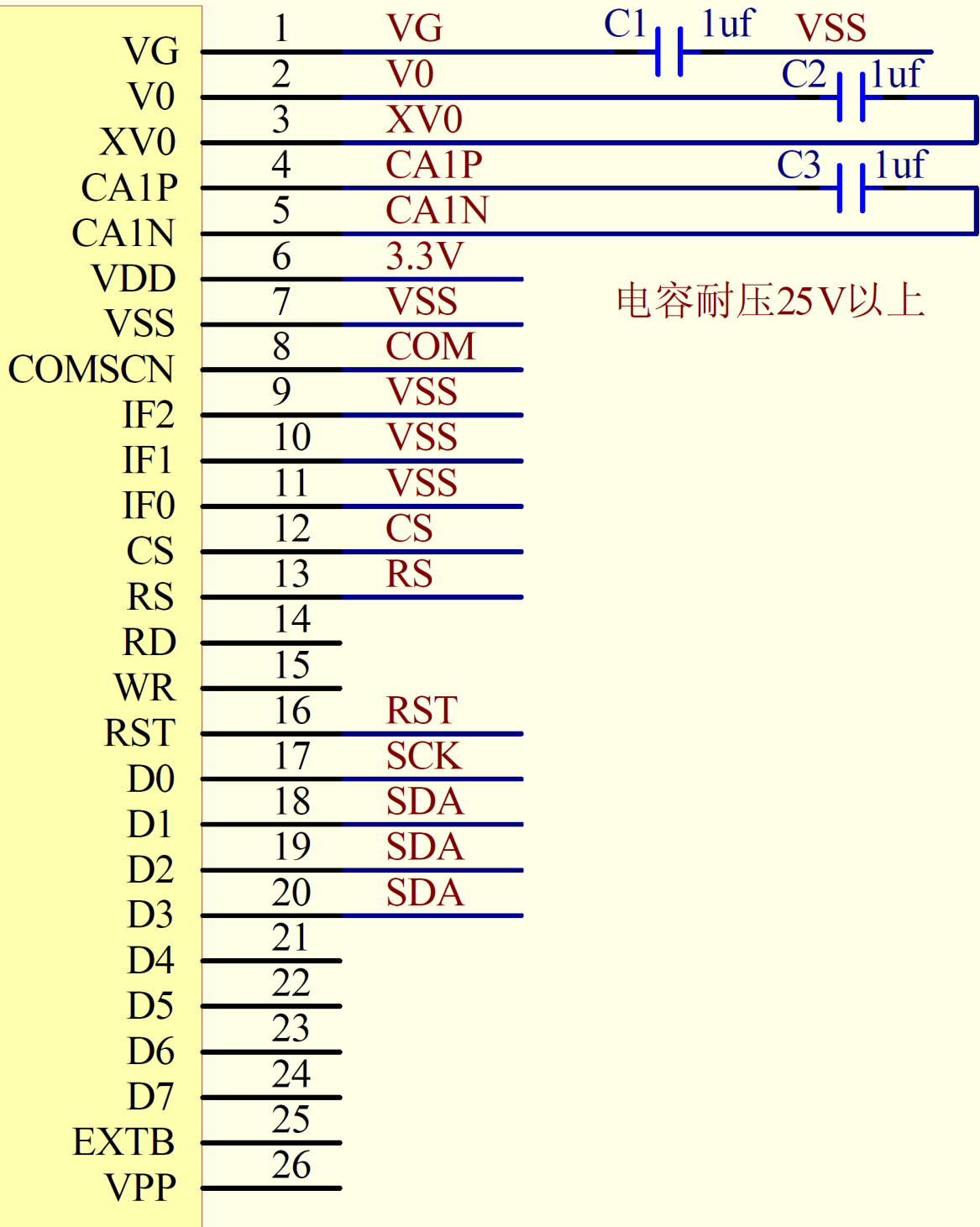
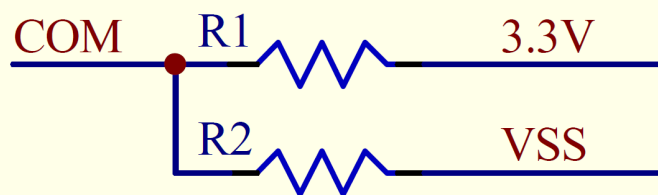


图 8. 串行接口

26PIN



26PIN



显示方向按图纸,COM接3.3V
显示方向旋转180度,COM接VSS



```
/* 液晶模块型号: JLX19296G-923-PN-S  
   串行接口  
   驱动 IC 是: ST75256  
   版权所有: 晶联讯电子: 网址 http://www.jlxlcd.cn;  
*/  
#include <reg52.H>  
#include <intrins.h>  
#include <chinese_code.h> //购买后, 源文件可向销售人员索要
```

```
sbit key=P2^0;  
sbit cs1=P3^2; //对应 LCD 的 CS 引脚  
sbit reset=P1^1; //对应 LCD 的 RST 引脚  
sbit rs=P3^1; //对应 LCD 的 RS 引脚  
sbit sclk=P1^2; //对应 LCD 的 SCK 引脚  
sbit sid=P1^3; //对应 LCD 的 SDA 引脚
```

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

```
/*延时: 1 毫秒的 i 倍*/  
void delay(int i)  
{  
    int j,k;  
    for(j=0;j<i;j++)  
        for(k=0;k<110;k++);  
}
```

```
/*延时: 1us 的 i 倍*/  
void delay_us(int i)  
{  
    int j,k;  
    for(j=0;j<i;j++)  
        for(k=0;k<1;k++);  
}
```

```
/*等待一个按键, 我的主板是用 P2.0 与 GND 之间接一个按键*/  
void waitkey()  
{  
    repeat:  
        if (key==1) goto repeat;  
        else delay(3000);  
}
```

//=====transfer command to LCM=====

```
void transfer_command(int data1)
{
```

```
    char i;
    csl=0;
    rs=0;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        delay_us(1);
        data1=data1<<=1;
    }
```

```
}
```

//-----transfer data to LCM-----

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

```
{
    char i;
    csl=0;
    rs=1;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
```

```
//    delay_us(1);
        data1=data1<<=1;
```

```
    }
```

```
}
```

```
void initial_lcd()
```

```
{
```

```
    reset=0;
    delay(100);
    reset=1;
    delay(100);
```

```
    transfer_command(0x30); //EXT1=0, EXT0=0, 表示选择了“扩展指令表1”
    transfer_command(0x94); //退出睡眠
    transfer_command(0x31); //EXT=1
    transfer_command(0xD7); //Autoread disable
```



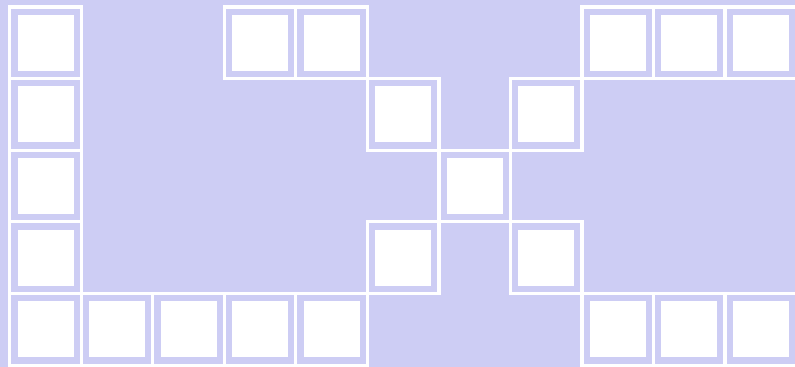
```
transfer_data(0X9F); //

transfer_command(0x32); //LCD 偏压比设置
transfer_data(0x00); //振荡频率的调整
transfer_data(0x01); //升压电容器的频率->6KHz
transfer_data(0x03); //Bias=1/11

transfer_command(0x31); //EXT=1
transfer_command(0xf0); //此指令比较重要, 不加此指令升压会慢 0.5s
transfer_data(0x0f);
transfer_data(0x0f);
transfer_data(0x0f);
transfer_data(0x0f);
```

```
transfer_command(0x20); //灰度设置
```

```
transfer_data(0x01);
transfer_data(0x03);
transfer_data(0x05);
transfer_data(0x07);
transfer_data(0x09);
transfer_data(0x0b);
transfer_data(0x0d);
transfer_data(0x10);
transfer_data(0x11);
transfer_data(0x13);
transfer_data(0x15);
transfer_data(0x17);
transfer_data(0x19);
transfer_data(0x1b);
transfer_data(0x1d);
transfer_data(0x1f);
```



```
transfer_command(0x30); // EXT1=0, EXT0=0, 表示选择了“扩展指令表 1”
transfer_command(0x75); // 页地址设置
transfer_data(0X00); // 起始页地址: YS=0X00
transfer_data(0X60); // 结束页地址: YE=0x1F 每 4 行为一页, 第 0~3 行为第 0 页, 第 124~127 行为第 31 页
(31=0x1f)
transfer_command(0x15); // 列地址设置
transfer_data(0X00); // 起始列地址: XS=0
transfer_data(0Xc0); // 结束列地址: XE=256 (0xff)

transfer_command(0xBC); //数据扫描方向
transfer_data(0x00); //MX.MY=Normal
transfer_data(0xA6);

transfer_command(0x0c); //数据格式选择, 0x0C 是低位在前 D0~D7, 0x08 是高位在前 D7~D0
```

```

transfer_command(0xCA); //显示控制
transfer_data(0X00); //设置 CL 驱动频率: CLD=0
transfer_data(0X5F); //占空比: Duty=96
transfer_data(0X20); //N 行反显: Nline=off

transfer_command(0xF0); //显示模式
transfer_data(0X10); //如果设为 0x11: 表示选择 4 灰度级模式, 如果设为 0x10: 表示选择黑白模式

transfer_command(0x81); //设置对比度, “0x81” 不可改动, 紧跟着的 2 个数据是可改的, 但“先微调后粗调”这个顺序
别乱了
transfer_data(0x30); //对比度微调, 可调范围 0x00~0x3f, 共 64 级
transfer_data(0x03); //对比度粗调, 可调范围 0x00~0x07, 共 8 级
transfer_command(0x20); //Power control
transfer_data(0x0B); //D0=regulator ; D1=follower ; D3=booste, on:1 off:0
delay_us(100);
transfer_command(0xAF); //打开显示

```

```

}

/*写 LCD 行列地址: X 为起始的列地址, Y 为起始的行地址, x_total, y_total 分别为列地址及行地址的起点到终点的差值 */
void lcd_address(int x, int y, x_total, y_total)
{
    x=x-1;
    y=y-1;
    transfer_command(0x15); //Set Column Address
    transfer_data(x);
    transfer_data(x+x_total-1);

    transfer_command(0x75); //Set Page Address
    transfer_data(y);
    transfer_data(y+y_total-1);
    transfer_command(0x30);
    transfer_command(0x5c);
}

/*清屏*/
void clear_screen()
{
    int i, j;
    lcd_address(1, 1, 256, 17);
    for(i=0; i<17; i++)

```

```

{
    for(j=0;j<256;j++)
    {
        transfer_data(0x00);
    }
}
}

```

//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）

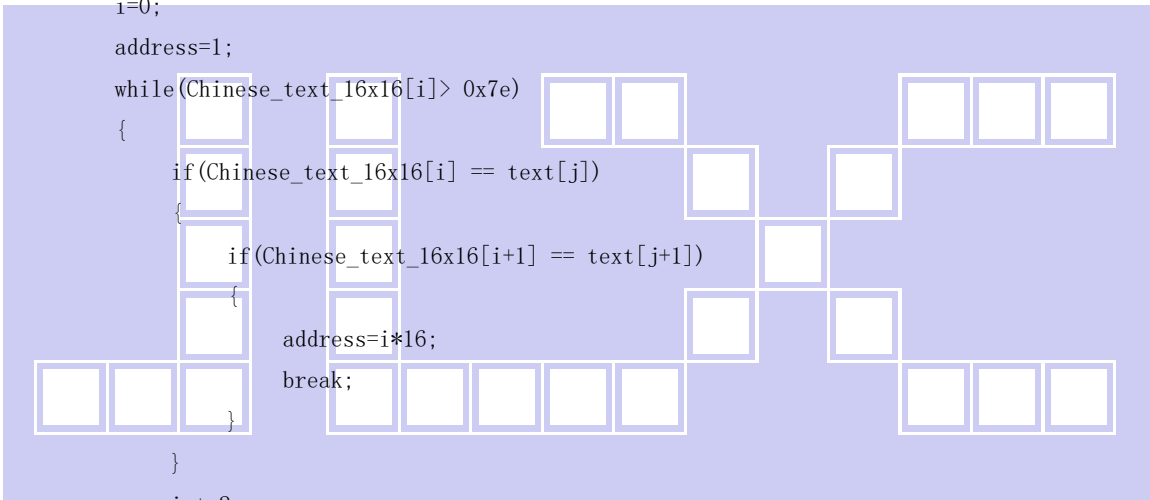
void display_string_16x16(uchar column, uchar page,uchar *text)

```

{
    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(address !=1)
        {
            lcd_address(column, page, 16, 2);
            for(k=0;k<2;k++)
            {
                for(i=0;i<16;i++)
                {
                    transfer_data(Chinese_code_16x16[address]);
                    address++;
                }
            }
            j +=2;
        }
        else

```




```

{
    lcd_address(column, page, 16, 2);
    for(k=0;k<2;k++)
    {
        for(i=0;i<16;i++)
        {
            transfer_data(0x00);
        }
    }
    j++;
}
column+=16;
}
}

```

//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(uchar column,uchar page,uchar reverse, uchar *text)
```

```

{
    uchar data1;
    uint i=0, j, k, n;

    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(column, page, 9, 2);
            for(n=0;n<2;n++)
            {
                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);
                }
                if(reverse==0) transfer_data(0x00);
                else transfer_data(0xff);
            }
            i++;
            column+=8;
        }
        else
            i++;
    }
}

```



```
}

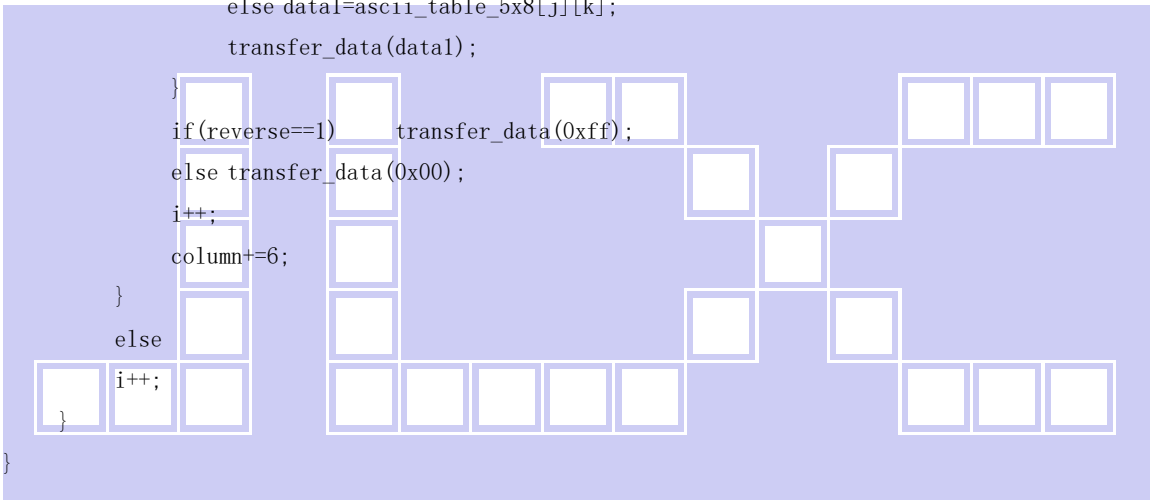
```

```
//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 column,uint page,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(column,page,7,1);
            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++;
        }
    }
}
```



```
/*显示 32*32 点阵的汉字或等同于 32*32 点阵的图像*/
```

```
void disp_32x32(int x,int y,uchar *dp)
```

```
{
    int i,j;
    lcd_address(x,y,32,4);
    for(i=0;i<4;i++)
    {
        for(j=0;j<32;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}
```

```

/*显示 64*48 点阵的汉字或图像*/
void disp_64x48(int x,int y,char *dp)
{
    int i,j;
    lcd_address(x,y,55,6);
    for(i=0;i<6;i++)
    {
        for(j=0;j<55;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}

```

```

/*显示 192*96 点阵的图像*/
void disp_192x96(int x,int y,char *dp)
{
    int i,j;
    lcd_address(x,y,192,12);
    for(i=0;i<12;i++)
    {
        for(j=0;j<192;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}

```

//-----

```

void main ()
{
    initial_lcd(); //对液晶模块进行初始化设置
    while(1)
    {
        clear_screen(); //清屏
        disp_192x96(1,1,bmp1); //显示一幅 192*96 点阵的黑白图。
        waitkey();
        clear_screen();
        disp_64x48(5,1,bmp5);
        disp_64x48(69,1,bmp6);
        disp_64x48(133,1,bmp7);
    }
}

```



```

disp_64x48(5, 7, bmp8);
disp_64x48(69, 7, bmp9);
disp_64x48(133, 7, bmp10);
waitkey();
clear_screen(); //清屏
disp_32x32(16, 1, jing2);
disp_32x32((32*1+16), 1, lian2);
disp_32x32((32*2+16), 1, xun2);
disp_32x32((32*3+16), 1, dian2);
disp_32x32((32*4+16), 1, zi2);
display_string_16x16(1, 5, "深圳市晶联讯电子有限公司");
display_string_8x16(1, 7, 1, "JLX19296G-923");
display_string_5x8(1, 9, 0, "JLX19296G-923");
waitkey();
}
}

```

7.5、并行接口

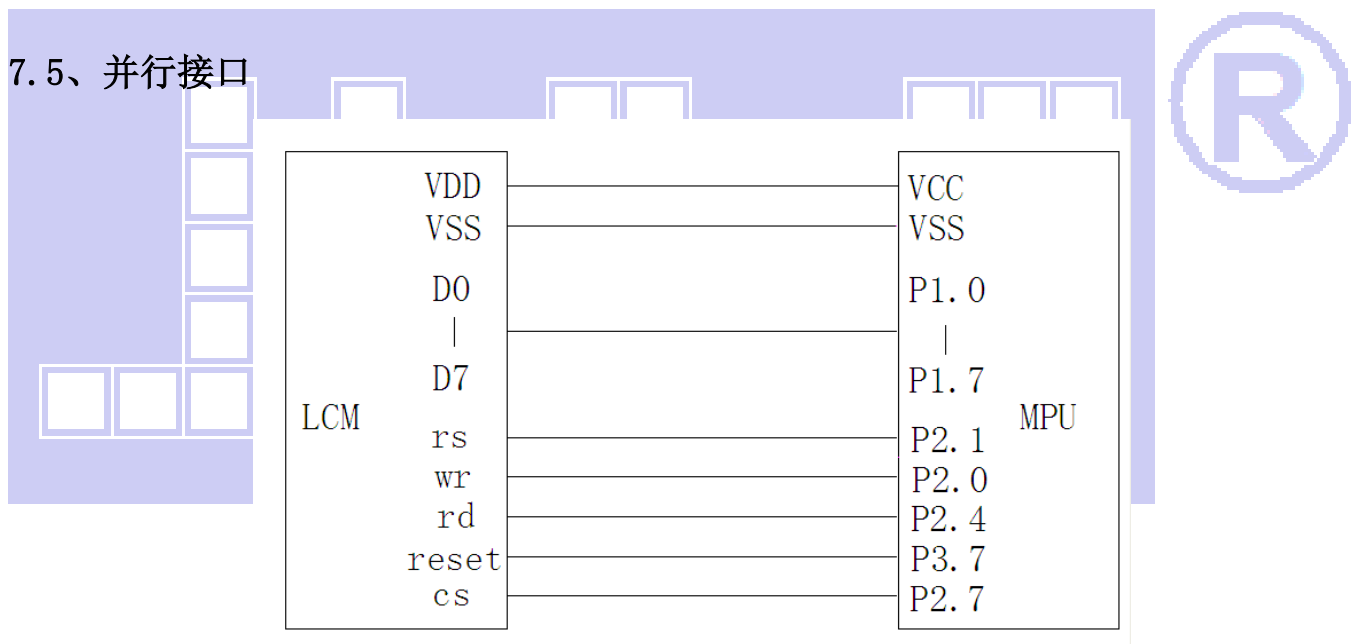
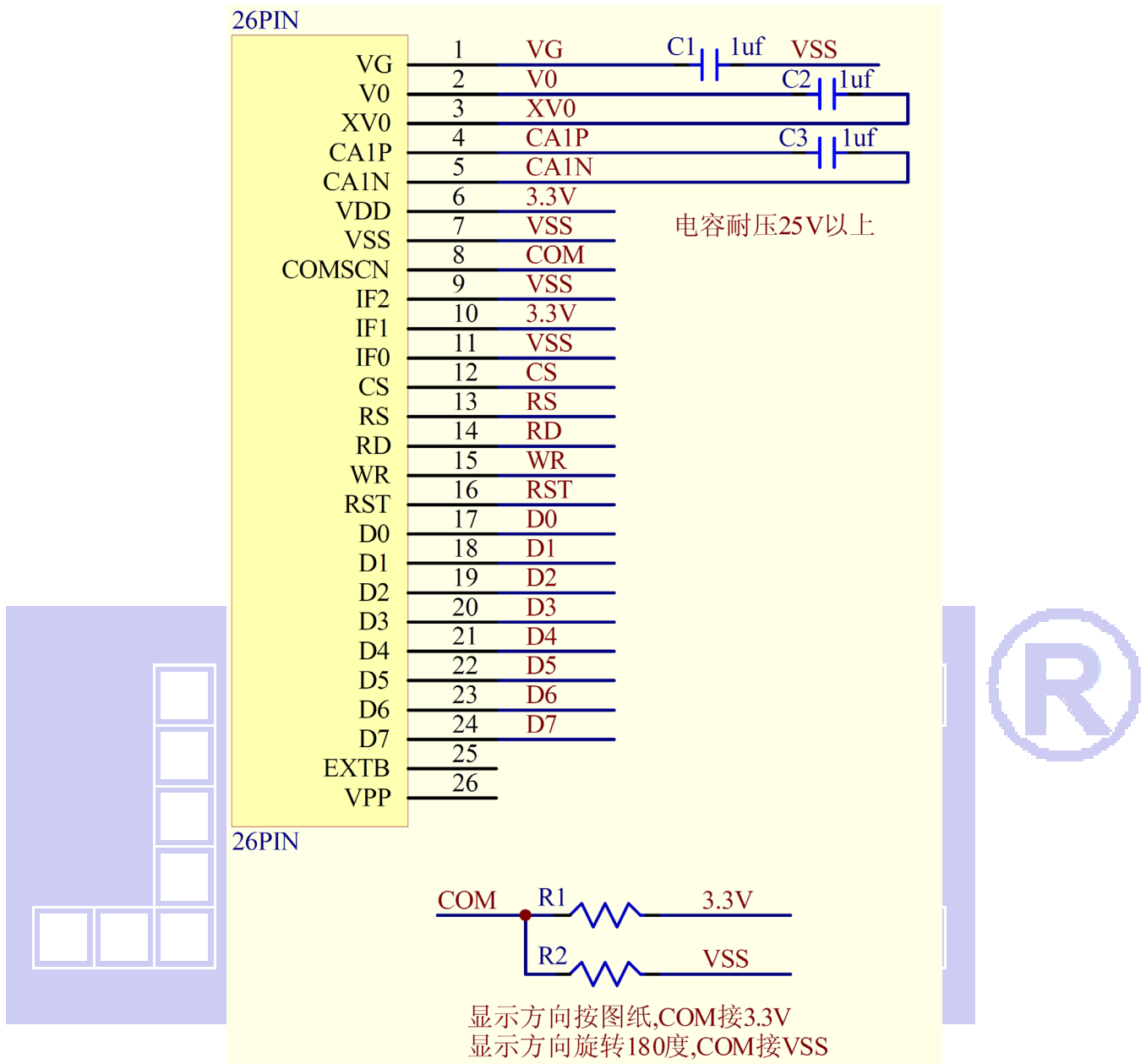


图 9. 并行接口



7.5.1、以下为并行接口方式范例程序

与串行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

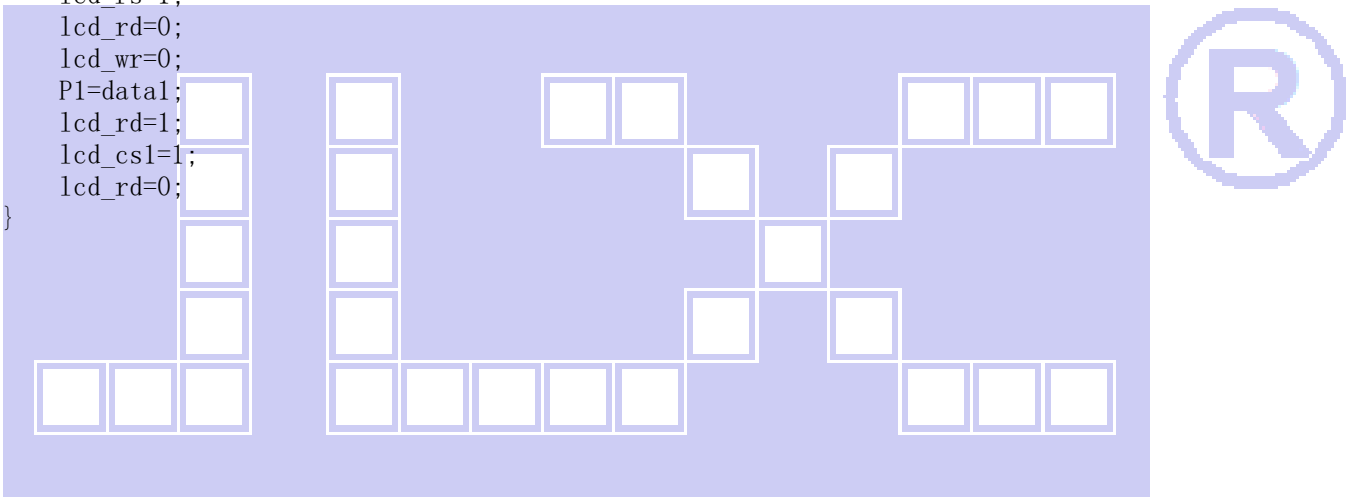
并程序序：

```
#include <reg52.H>
#include <intrins.h>
#include <chinese_code.h>

sbit lcd_rs=P2^1; /*接口定义:lcd_rs 就是 LCD 的 rs*/
sbit lcd_rd=P2^4; /*接口定义:lcd_e 就是 LCD 的 rd*/
sbit lcd_wr=P2^0; /*接口定义:lcd_rw 就是 LCD 的 wr*/
sbit lcd_reset=P3^7; /*接口定义:lcd_reset 就是 LCD 的 reset*/
sbit lcd_cs1=P2^7; /*接口定义:lcd_cs1 就是 LCD 的 cs1*/
sbit key = P2^0; //按键
```

```
//写指令到LCD 模块
void transfer_command_lcd(int data1)
{
    lcd_cs1=0;
    lcd_rs=0;
    lcd_rd=0;
    lcd_wr=0;
    P1=data1;
    lcd_rd=1;
    lcd_cs1=1;
    lcd_rd=0;
}
```

```
//写数据到LCD 模块
void transfer_data_lcd(int data1)
{
    lcd_cs1=0;
    lcd_rs=1;
    lcd_rd=0;
    lcd_wr=0;
    P1=data1;
    lcd_rd=1;
    lcd_cs1=1;
    lcd_rd=0;
}
```



7.6、IIC 接口

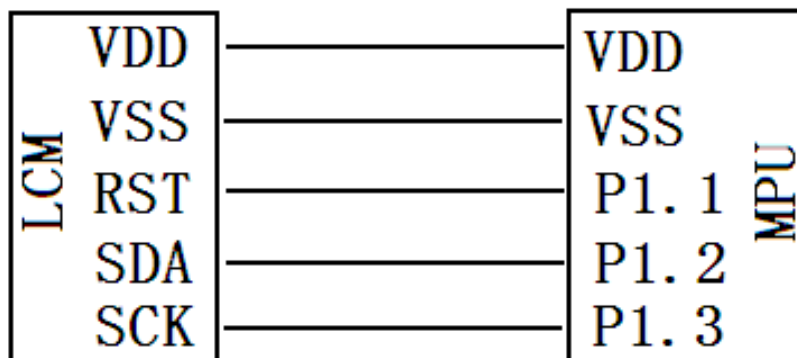
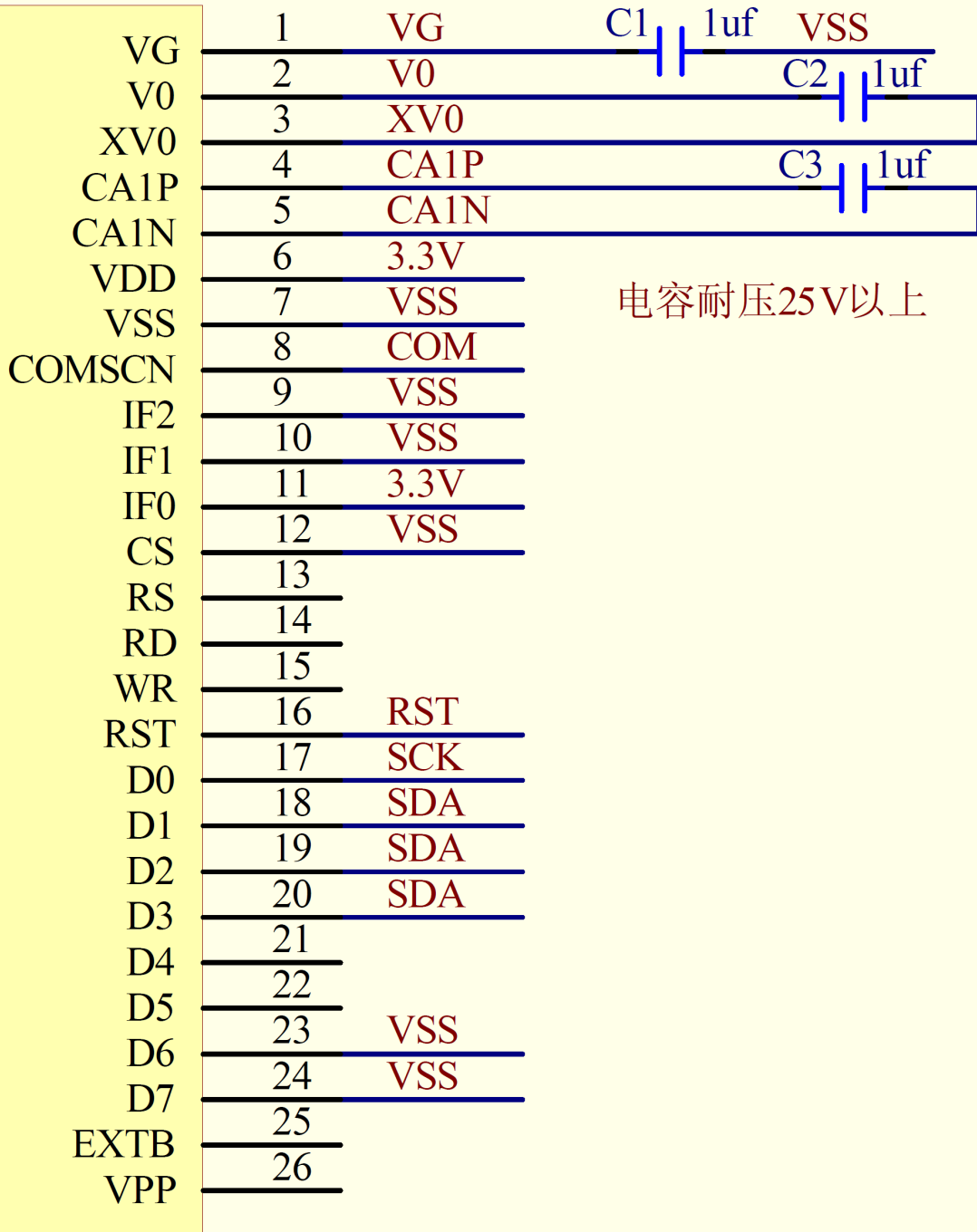
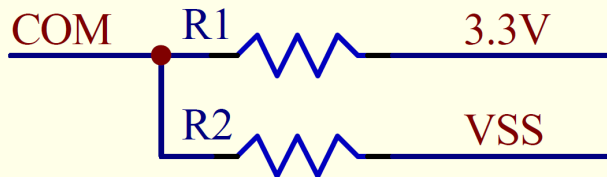


图 10. IIC

26PIN



26PIN



显示方向按图纸,COM接3.3V
显示方向旋转180度,COM接VSS

7.5.1、以下为 I2C 接口方式范例程序

与串行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

```

/* 液晶模块型号：JLX19296G-923
   IIC 接口
   驱动 IC 是：ST75256
   版权所有：晶联讯电子：网址 http://www.jlxlcd.cn;
*/

```

```

#include <reg52.H>
#include <intrins.h>
#include <chinese_code.h>

```

```

sbit reset=P1^1;
sbit scl=P1^3;
sbit sda=P1^2;
sbit key=P2^0;

```

```

void transfer(int data1)
{

```

```

    int i;
    for(i=0;i<8;i++)
    {

```

```

        scl=0;
        if(data1&0x80) sda=1;
        else sda=0;
        scl=1;
        scl=0;
        data1=data1<<1;
    }

```

```

        sda=0;
        scl=1;
        scl=0;
    }

```

```

void start_flag()
{

```

```

    scl=1;    /*START FLAG*/
    sda=1;    /*START FLAG*/
    sda=0;    /*START FLAG*/
}

```

```

void stop_flag()
{

```

```

    scl=1;    /*STOP FLAG*/
    sda=0;    /*STOP FLAG*/
    sda=1;    /*STOP FLAG*/
}

```

//写命令到液晶显示模块

```

void transfer_command(uchar com)
{

```

```

    start_flag();
    transfer(0x78);
    transfer(0x80);
    transfer(com);
    stop_flag();
}

```

//写数据到液晶显示模块

```

void transfer_data(uchar dat)

```



```
{  
  start_flag();  
  transfer(0x78);  
  transfer(0xC0);  
  transfer(dat);  
  stop_flag();  
}
```

-END-

