



JLX256128G-92906-BN 使用说明书

目 录

序号	内 容 标 题	页 码
1	概述	2
2	特点	2
3	外形及接口引脚功能	3-5
4	电路框图	5-6
5	背光参数	6
6	时序特性	6-11
7	指令表及硬件接口、编程案例	12-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 JLX256128G-92906-BN 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。

JLX256128G-92906-BN 可以显示 256 列*128 行点阵单色或 4 灰度级的图片，或显示 8 个/行*4 行 32*32 点阵或显示 10 个/行*5 行 24*24 点阵的汉字，或显示 16 个/行*8 行 16*16 点阵的汉字。

2. JL256128G-92906-BN 图像型点阵液晶模块的特性

2.1 结构牢。

2.2 IC 采用矽创公司 ST75256, 功能强大，稳定性好

2.3 功耗低。

2.4 接口简单方便:可采用 4 线 SPI 串行接口、并行接口，I²C 接口。

2.5 工作温度宽:-20℃ - 70℃;

2.6 储存温度宽:-30℃ - 80℃;

2.7 显示内容:

- 256*128 点阵单色或 4 灰度级图片;
- 或显示 8 个×4 行 32*32 点阵的汉字;
- 或显示 10 个×5 行 24*24 点阵的汉字;
- 或显示 16 个×8 行 16*16 点阵的汉字;
- 或显示其他的 ASCII 码等;



3. 外形尺寸及接口引脚功能:

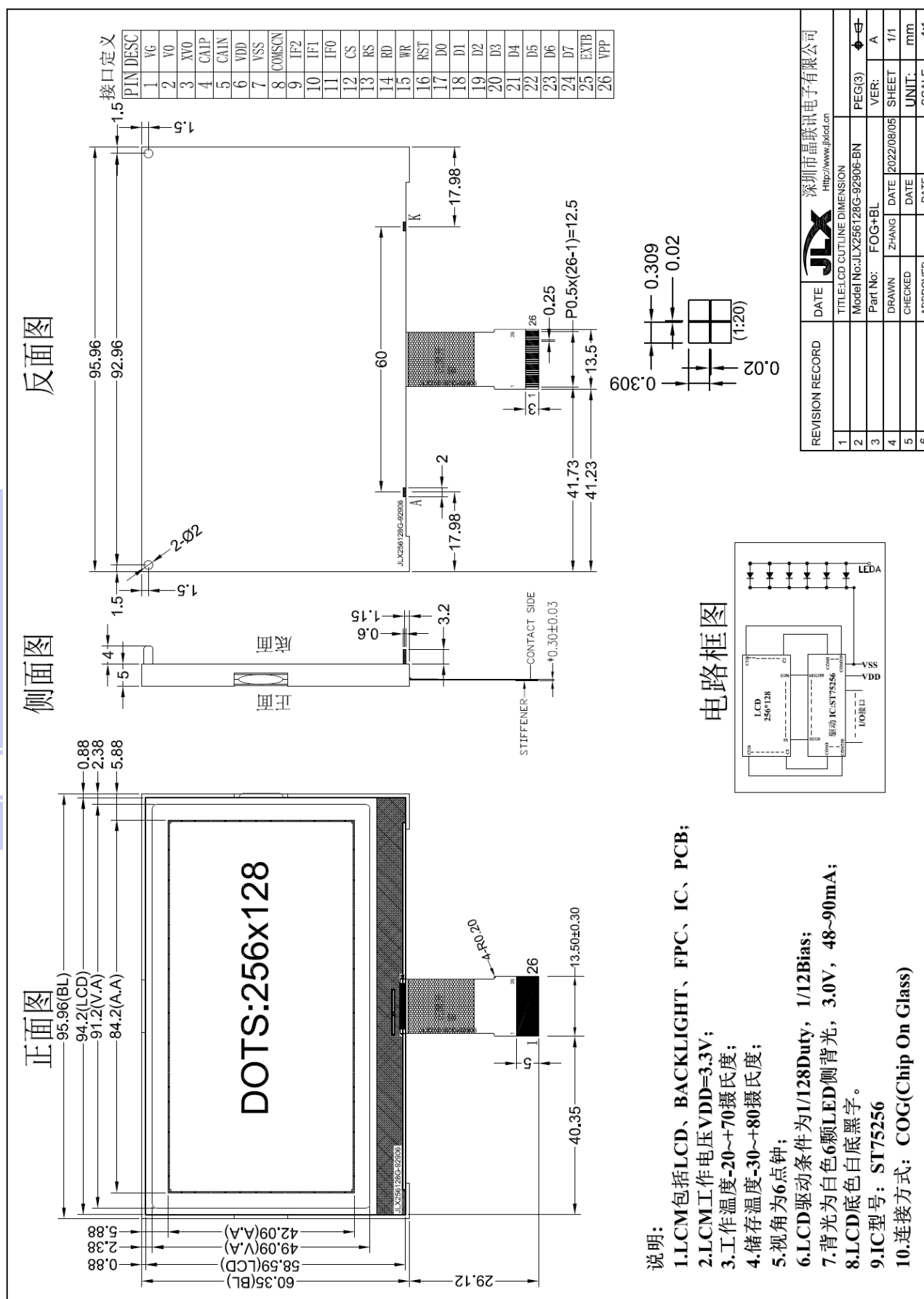


图 1. 液晶模块外形尺寸

3.1 模块的接口引脚功能

3.1.1 并行时接口引脚功能

表 1

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	电源电路	供电电源正极
7	VSS	接地	0V
8	COMSCN	COMSCN	镜像, 默认接 VDD, (接 VSS 旋转 180)
9	IF2	IF2	L: 接低电平
10	IF1	IF1	H: 接高电平
11	IF0	IF0	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	D0~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

3.1.2 四线串行时接口引脚功能

表 2

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	电源电路	供电电源正极
7	VSS	接地	0V
8	COMSCN	COMSCN	镜像, 默认接 VDD, (接 VSS 旋转 180)
9	IF2	IF2	L: 接低电平
10	IF1	IF1	L: 接低电平
11	IF0	IF0	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	D0(SCK)	I/O	串行时钟
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

3.1.3 I²C 总线时接口引脚功能

表 3

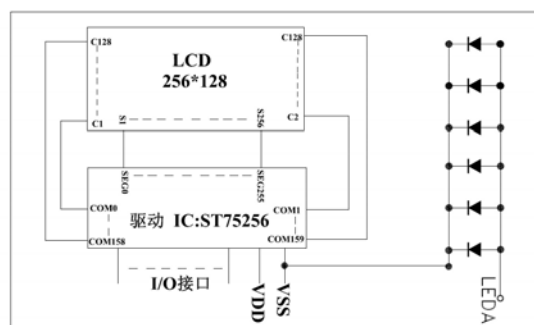
引 线 号	符 号	名 称	功 能
1	VG	偏压电路	LCD 偏置驱动电压，VG 与 VSS 之间接一个电容
2	V0	倍压电路	V0 与 XV0 之间接一个电容
3	XV0	倍压电路	
4	CA1P	倍压电路	CA1P 与 CA1N 之间接一个电容
5	CA1N	倍压电路	
6	VDD	电源电路	供电电源正极
7	VSS	接地	0V
8	COMSCN	COMSCN	镜像，默认接 VDD, (接 VSS 旋转 180)
9	IF2	IF2	L: 接低电平
10	IF1	IF1	L: 接低电平
11	IF0	IF0	H: 接低电平
12	CS	片选	I2C 接口，此引脚接 VSS
13	A0(RS)	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(SCK)	I/O	串行时钟
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. 电路框图

图 2: JLX256128G-92906-BN 图像点阵型液晶模块的电路框图

电路框图



4.1 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： $-20^{\circ}\text{C}\sim+70^{\circ}\text{C}$ ；

背光颜色：白色。

正常工作电流为： $(8\sim 15)\times 6=48\sim 90\text{mA}$ （LED 灯数共 6 颗）；

工作电压：3.0；（接 3.3V 串 20 欧电阻，接 5.0V 串 120 欧电阻）

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	—	4.0	V
LCD 驱动电压	V0 - XV0	-0.3	—	16	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	3.5	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	$I_{OH} = 0.2\text{mA}$	0.8VDD	—	VDD	V
输出低电平	VOO	$I_{OO} = 1.2\text{mA}$	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	48	90	120	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）

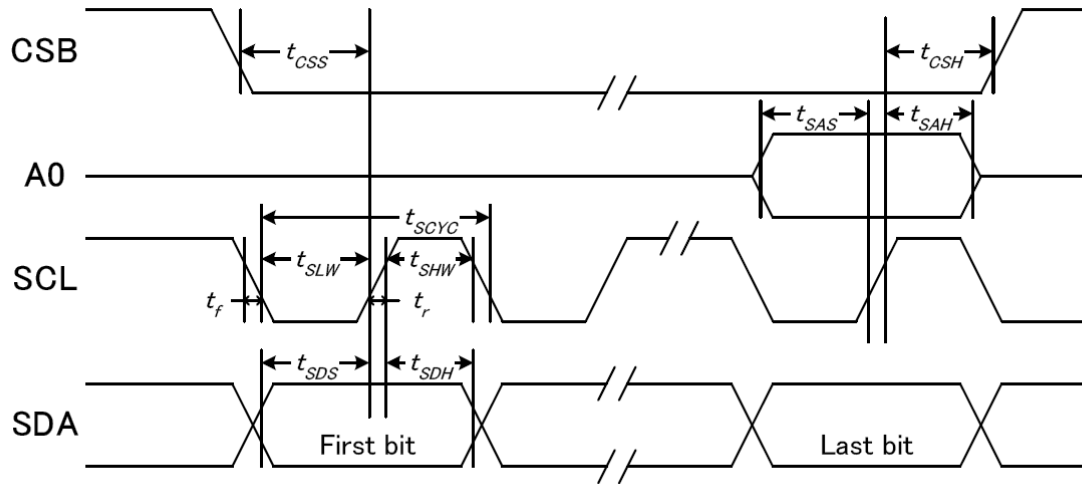


图 3. 从 CPU 写到 ST75256 (Writing Data from CPU to ST75256)

表 6. 写数据到 ST75256 的时序要求

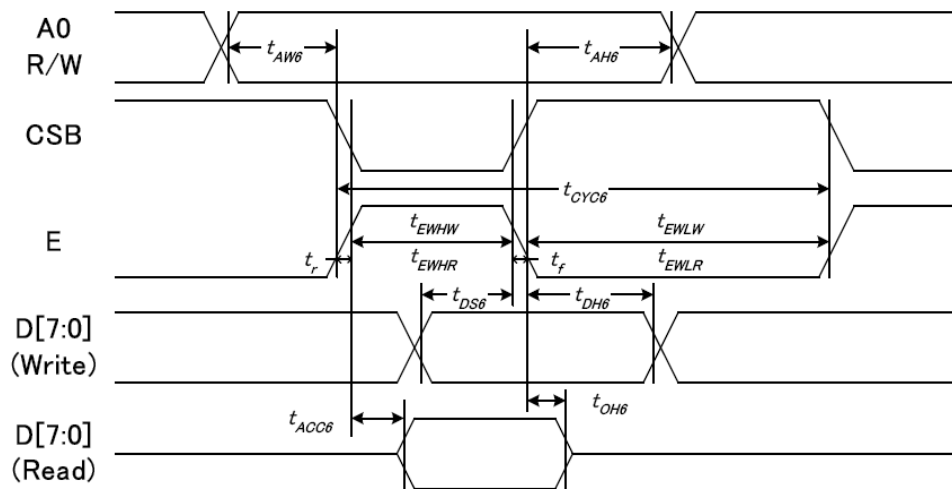
项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
4线 SPI串口时钟周期 (4-line SPI Clock Period)	tSCYC	引脚: SCL	80	--	--	ns
保持SCK高电平脉宽 (SCL "H" pulse width)	tSHW		30	--	--	ns
保持SCLK低电平脉宽 (SCL "L" pulse width)	tSLW		30	--	--	ns
地址建立时间 (Address setup time)	tSAS	引脚: A0	20	--	--	ns
地址保持时间 (Address hold time)	tSAH		20	--	--	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℃

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)



1.

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	tAH6	20		—	ns
地址建立时间		tAW6	0		—	ns
系统循环时间	E	tCYC6	160		—	ns
使能“低”脉冲宽度		tEHLW	70		—	ns
使能“高”脉冲宽度		tEHWLW	70		—	ns
写数据建立时间	DB[7: 0]	tDS6	15		—	ns
写数据保持时间		tDH6	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,

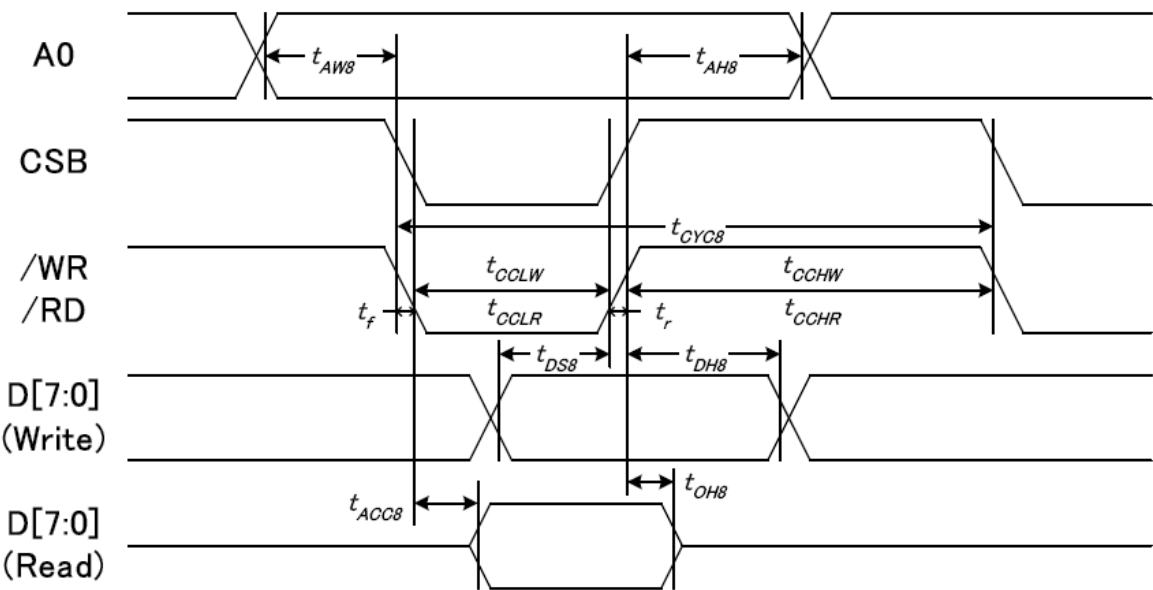
$(TR + TF) \leq (tcyc6 - tewlw - tewhw)$ 指定。

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	tAH8	20		—	ns
地址建立时间		tAW8	0		—	ns
系统循环时间	/WR	tCYC8	160		—	ns
使能“低”脉冲宽度		tCCLW	70		—	ns
使能“高”脉冲宽度		tCCHW	70		—	ns
写数据建立时间	DB	tDS8	15		—	ns
写数据保持时间		tDH8	15		—	ns

VDD =1.8~3.3V±5%, Ta = -30~85℃

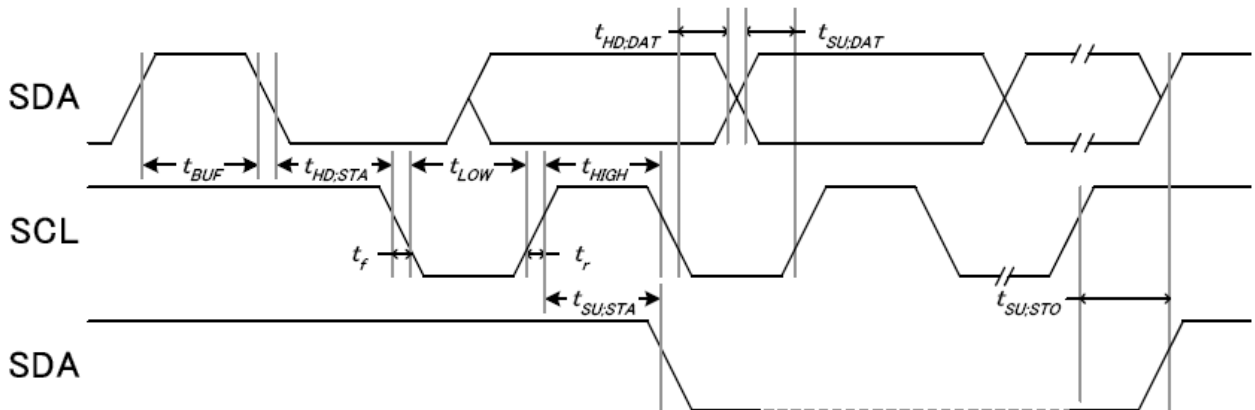
输入信号的上升时间和下降时间（TR，TF）是在 15 纳秒或更少的规定。当系统循环时间非常快，

（TR + TF）≤（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	FSCLK	—		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):

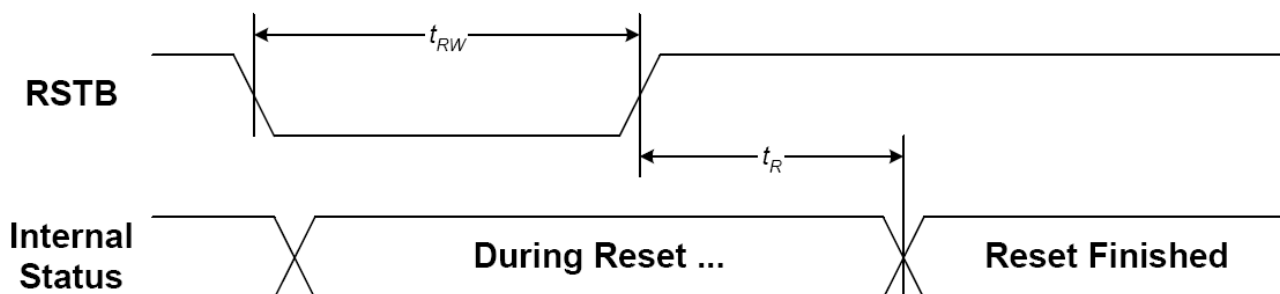


图 7: 电源启动后复位的时序

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}		---	---	1	us
复位保持低电平的时间	T_{RD}	引脚: RESET, WR	1	---	---	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 0X30: 扩展指令 1
Ext[1:0]=0,0(Extension Command1/扩展指令 1) 0X30 扩屏指令 1 一定要调用 0X30 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: 0XAE: 关, 0XAF: 开
(3) 正显/反显 (Inverse Display)	0	0	1	0	1	0	0	1	1	0	显示正显/反显 0XA6: 正显, 正常 0XA7: 反显
(4) 所有点阵开/关 (All Pixel ON/OFF)	0	0	0	0	1	0	0	0	1	0	0X22: 所有点阵关 0X23: 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	0XCA: 显示控制
	1	0	0	0	0	0	0	CLD	0	0	0X00: 设置 CL 驱动频率: CLD=0
	1	0	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	0X7F: 点空比: Duty=128
	1	0	0	0	LF4	F1	LF3	LF2	LF1	LF0	0X20: 帧周期
(6) 省电模式 (Power save)	0	0	1	0	0	1	0	1	0	SLP	0X94: SLP=0, 退出睡眠模式 0X95: 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	0X75: 页地址设置
	1	0	YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0	0X00: 起始页地址
	1	0	YE7	YE6	YE5	YE4	YE3	YE2	YE2	YE0	0X1F: 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	0X15: 列地址设置
	1	0	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	0X00: 起始列地址
	1	0	XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0	0XFF: 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	0XBC: 行列扫描方向
	1	0	0	0	0	0	0	MV	MX	MY	0X00: MX、MY=Normal
(10) 写数据到液晶屏 (Write Data)	0	0	0	1	0	1	1	1	0	0	0X5C: 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	0X5D: 读数据
	1	1	D7	D6	D5	D4	D3	D2	D1	D0	8 位显示数据
	1	0	PTS7	PTS6	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0	0XA8: 指定显示区域 起始区域地址: 00h≤PTS≥A1h
(12) 指定区域显示数据 (Partial In)	1	0	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	结束区域地址: 00h≤PTE≥A1h
	1	0	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	0XA9: 退出指定区域显示
(13) 退出指定区域显示 (Partial Out)	0	0	1	0	1	0	1	0	0	1	0XA9: 退出指定区域显示
(14) 读/改/写	0	0	1	1	1	0	0	0	0	0	0XE0: 进入读/改/写
(15) 退出读/改/写	0	0	1	1	1	0	1	1	1	0	0XEE: 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	0XAA: 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	NSL5	NSL4	NSL3	NSL2	NSL1	NSL0	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	0XAB: 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
	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]
(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	读状态字节
(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
	0	0	1	1	1	1	0	0	0	0	0XF0: 显示模式设置 0X10: 黑白模式 0X11: 4 灰级度模式
(28)显示模式 (Display Mode)	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	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	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	
	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	BS2	BS1	BS0	0x02: 偏压比, BIAS=1/12
(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围
	1	0	0	0	0	FRA4	FRA3	FRA2	FRA1	FRA0	
	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

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 256*128 点阵的屏分为 16 个“页”, 从第 0 “页”到第 15 “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:

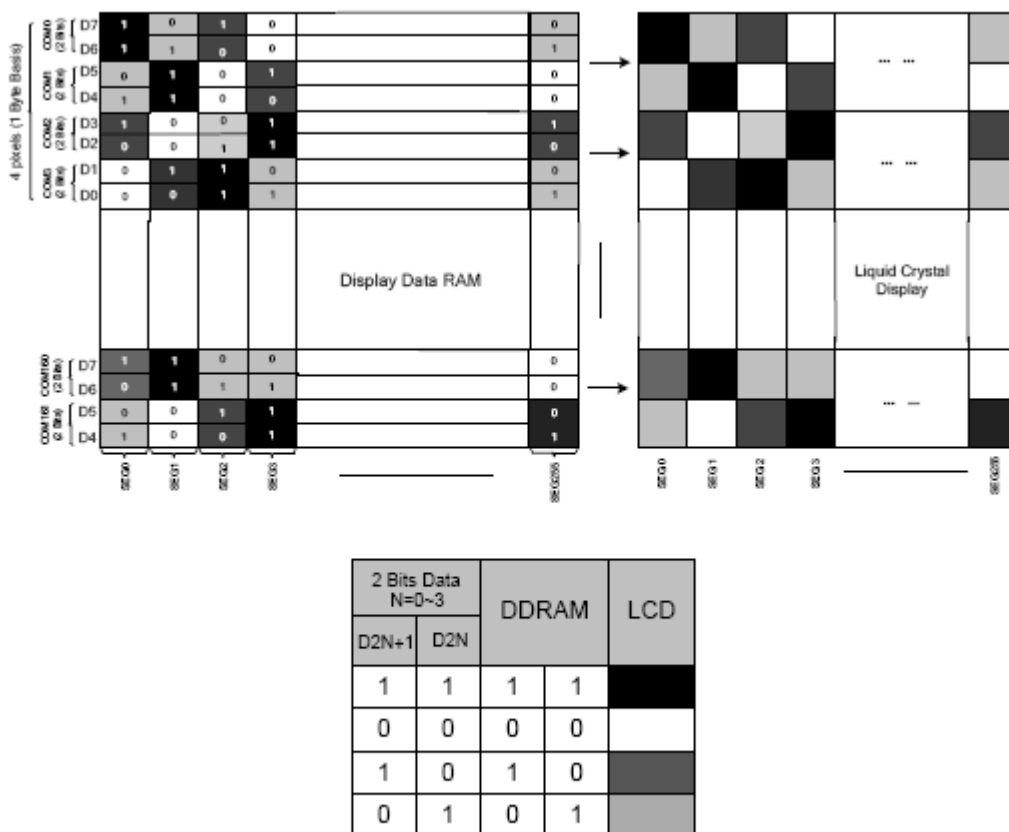
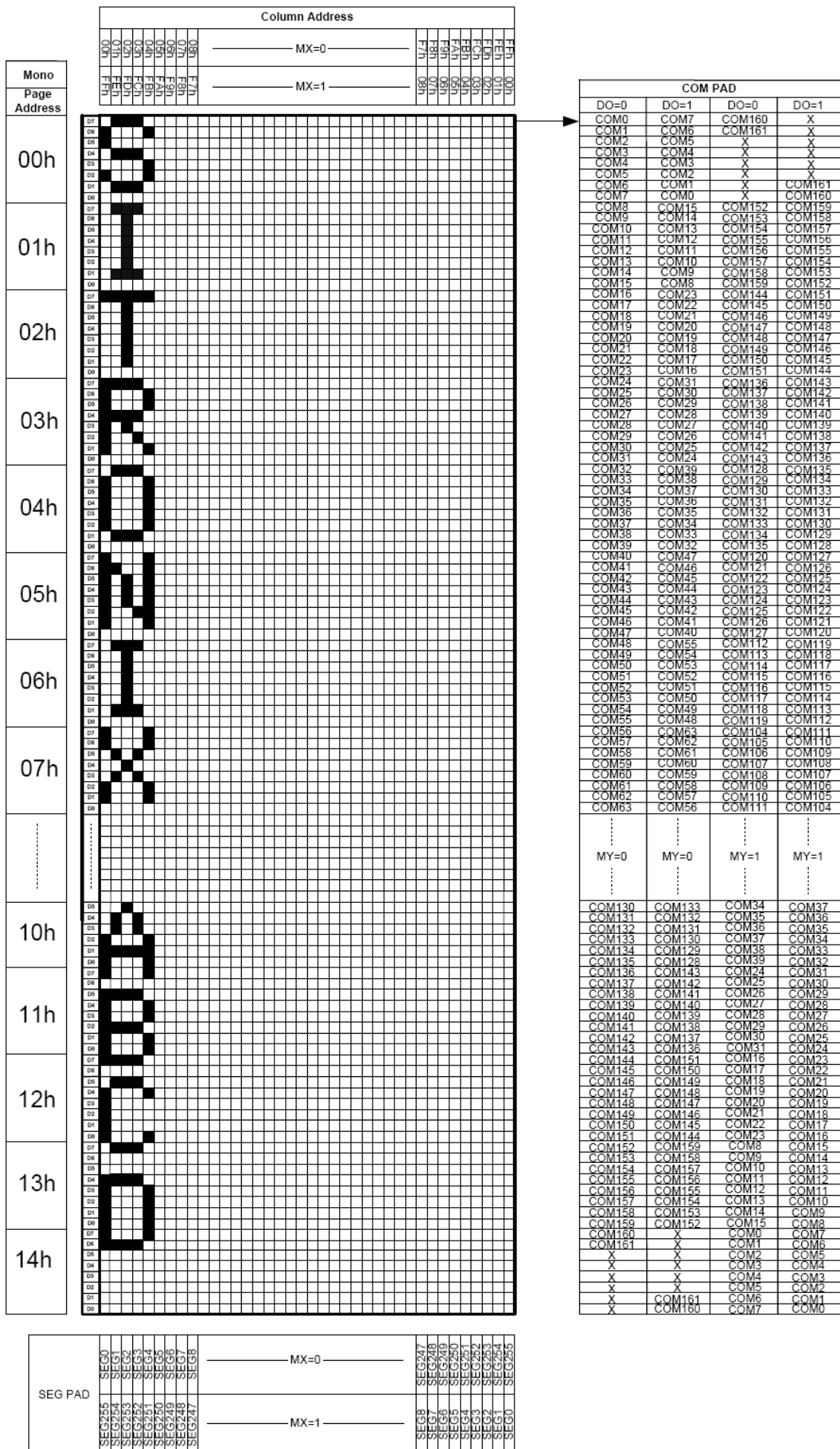


Figure 21 DDRAM Mapping (4-Level Gray Scale Mode)

下图摘自 ST75256 IC 资料, 可通过 “ST75256.PDF” 之第 37 页获取最佳效果。



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

硬件准备:
开发板 (或专门设计的主板)、单片机、电源、连接线、仿真器或程序下载器 (又名烧录器)

正确地接线
根据说明书正确地与开发板连接, 连接的线包括: 液晶模块电源线、背光源电源线、IO端口 (接口)
IO端口包括: 并口时: CS、RESET、RW、E、RS、D0--D7, 串口时: CS、SCLK、SDA、RESET、RS

编写软件
背光给合适的直流电可以点亮, 但液晶屏里面没有程序, 只给电不能让液晶屏显示 (我们通常说“点亮”), 程序须另外编写, 并烧录 (下载) 到单片机里液晶模块才能工作。

7.4 接口方式及程序:

7.4.1 液晶模块与 MPU (以 8051 系列单片机为例) 接口图如下:

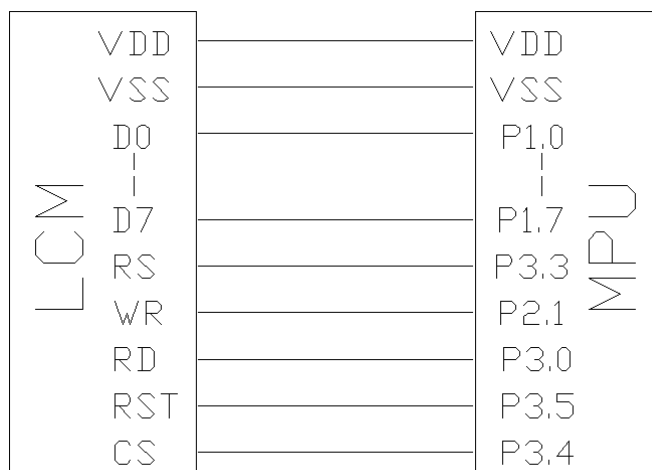


图 8: 并行接口图



/* 液晶模块型号: JLX256128G-92906

并行接口

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>

sbit cs1=P3⁵; /*接口定义*/

sbit reset=P3²; /*接口定义*/

sbit rs=P3⁴; /*接口定义*/

sbit rd=P3⁰; /*接口定义*/

sbit wr=P3¹; /*接口定义。另外 P1.0~1.7 对应 DB0~DB7*/

sbit key=P2⁰; /*按键接口, P2.0 口与 GND 之间接一个按键*/

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

```
/*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(2000);
}
```

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

```
void transfer_command_lcd(int data1)
{
    cs1=0;
    rs=0;
    rd=0;
    delay_us(1);
    wr=0;
    P1=data1;
    rd=1;
    delay_us(1);
    cs1=1;
    rd=0;
```



```

}

//-----transfer data to LCM-----
void transfer_data_lcd(int data1)
{
    cs1=0;
    rs=1;
    rd=0;
    delay_us(1);
    wr=0;
    P1=data1;
    rd=1;
    delay_us(1);
    cs1=1;
    rd=0;
}

//-----对比度设置值,粗调 0x04,微调 0x39-----//
void initial_lcd()
{
    reset=0;
    delay(100);
    reset=1;
    delay(100);
    transfer_command_lcd(0x30); //EXT=0
    transfer_command_lcd(0x94); //Sleep out
    transfer_command_lcd(0x31); //EXT=1
    transfer_command_lcd(0xD7); //Autoread disable
    transfer_data_lcd(0X9F); //
    transfer_command_lcd(0x32); //Analog SET
    transfer_data_lcd(0x00); //OSC Frequency adjustment
    transfer_data_lcd(0x01); //Frequency on booster capacitors->6KHz
    transfer_data_lcd(0x02); //Bias=1/12

    transfer_command_lcd(0x31); //Analog SET
    transfer_command_lcd(0xf2); //温度补偿
    transfer_data_lcd(0x1e); //OSC Frequency adjustment
    transfer_data_lcd(0x28); //Frequency on booster capacitors->6KHz
    transfer_data_lcd(0x32); //

    transfer_command_lcd(0x20); // Gray Level
    transfer_data_lcd(0x01);
    transfer_data_lcd(0x03);
    transfer_data_lcd(0x05);
    transfer_data_lcd(0x07);
    transfer_data_lcd(0x09);

```



```

transfer_data_lcd(0x0b);
transfer_data_lcd(0x0d);
transfer_data_lcd(0x10);
transfer_data_lcd(0x11);
transfer_data_lcd(0x13);
transfer_data_lcd(0x15);
transfer_data_lcd(0x17);
transfer_data_lcd(0x19);
transfer_data_lcd(0x1b);
transfer_data_lcd(0x1d);
transfer_data_lcd(0x1f);

transfer_command_lcd(0x30); //EXT=0
transfer_command_lcd(0x75); //Page Address setting
transfer_data_lcd(0X00); // XS=0
transfer_data_lcd(0X14); // XE=159
transfer_command_lcd(0x15); //Column Address setting
transfer_data_lcd(0X00); // XS=0
transfer_data_lcd(0Xff); // XE=256
transfer_command_lcd(0xBC); //Data scan direction
transfer_data_lcd(0x00); //MX.MY=Normal

transfer_command_lcd(0xCA); //Display Control
transfer_data_lcd(0X00); //
transfer_data_lcd(0X7F); //Duty=128
transfer_data_lcd(0X20); //Nline=off

transfer_command_lcd(0xF0); //Display Mode
transfer_data_lcd(0X10); //10=Monochrome Mode, 11=4Gray
transfer_command_lcd(0x81); //EV control
transfer_data_lcd(0x39); //VPR[5-0]
transfer_data_lcd(0x04); //VPR[8-6]
transfer_command_lcd(0x20); //Power control
transfer_data_lcd(0x0B); //D0=regulator ; D1=follower ; D3=booste, on:1
off:0
delay_us(100);
transfer_command_lcd(0xAF); //Display on
}

```

/*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_lcd(0x15); //Set Column Address
```

```
transfer_data_lcd(x);
```

```
transfer_data_lcd(x+x_total-1);
```

```
transfer_command_lcd(0x75); //Set Page Address
```

```
transfer_data_lcd(y);
```

```
transfer_data_lcd(y+y_total-1);
```

```
transfer_command_lcd(0x30);
```

```
transfer_command_lcd(0x5c);
```

```
}
```

```
/*清屏*/
```

```
void clear_screen()
```

```
{
```

```
int i, j;
```

```
lcd_address(0, 0, 256, 17);
```

```
for(i=0; i<17; i++)
```

```
{
```

```
for(j=0; j<256; j++)
```

```
{
```

```
transfer_data_lcd(0x00);
```

```
}
```

```
}
```

```
}
```

```
void test(int data1, int data2)
```

```
{
```

```
int i, j;
```

```
lcd_address(1, 1, 256, 16);
```

```
for(i=0; i<16; i++)
```

```
{
```

```
for(j=0; j<256; j++)
```

```
{
```

```
transfer_data_lcd(data1);
```

```
transfer_data_lcd(data2);
```

```
}
```

```
}
```

```
}
```



//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(column > 255)
            {
                column=0;
                page+=2;
            }
            if(address != 1)
            {
                lcd_address(column, page, 16, 2);
                for(k=0; k<2; k++)
                {
                    for(i=0; i<16; i++)
                    {
                        transfer_data_lcd(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_lcd(0x00);
        }
    }
    j++;
}
column+=16;
}
}

```

/*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_lcd(*dp);
            dp++;
        }
    }
}

```

/*显示 256*128 点阵的图像*/

```
void disp_256x128(int x,int y,char *dp)
```

```

{
    int i,j;
    lcd_address(x,y,256,16);
    for(i=0;i<16;i++)
    {
        for(j=0;j<256;j++)
        {
            transfer_data_lcd(*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_256x128(1, 1, bmp1); //显示一幅 256*128 点阵的黑白图。
        waitkey();
        disp_256x128(1, 1, bmp2); //显示一幅 256*128 点阵的黑白图。
        waitkey();
        disp_256x128(1, 1, bmp3); //显示一幅 256*128 点阵的黑白图。
        waitkey();
        disp_256x128(1, 1, bmp4); //显示一幅 256*128 点阵的黑白图。
        waitkey();
        disp_256x128(1, 1, bmp5); //显示一幅 256*128 点阵的黑白图。
        waitkey();
        disp_256x128(1, 1, bmp6); //显示一幅 256*128 点阵的黑白图。
        waitkey();
        disp_256x128(1, 1, bmp7); //显示一幅 256*128 点阵的黑白图。
        waitkey();
        disp_256x128(1, 1, bmp8); //显示一幅 256*128 点阵的黑白图。
        waitkey();

        clear_screen();
        display_string_16x16(33, 4, "深圳市晶联讯电子有限公司");
        disp_32x32((32*0+48), 8, jing2);
        disp_32x32((32*1+48), 8, lian2);
        disp_32x32((32*2+48), 8, xun2);
        disp_32x32((32*3+48), 8, dian2);
        disp_32x32((32*4+48), 8, zi2);
        waitkey();

        test(0xff, 0xff);
        waitkey();
        // transfer_command_lcd(0x95); //进入睡眠模式
        // waitkey();
        // transfer_command_lcd(0x94); //退出睡眠模式
        // waitkey();
    }
}

```

7.5 程序举例:

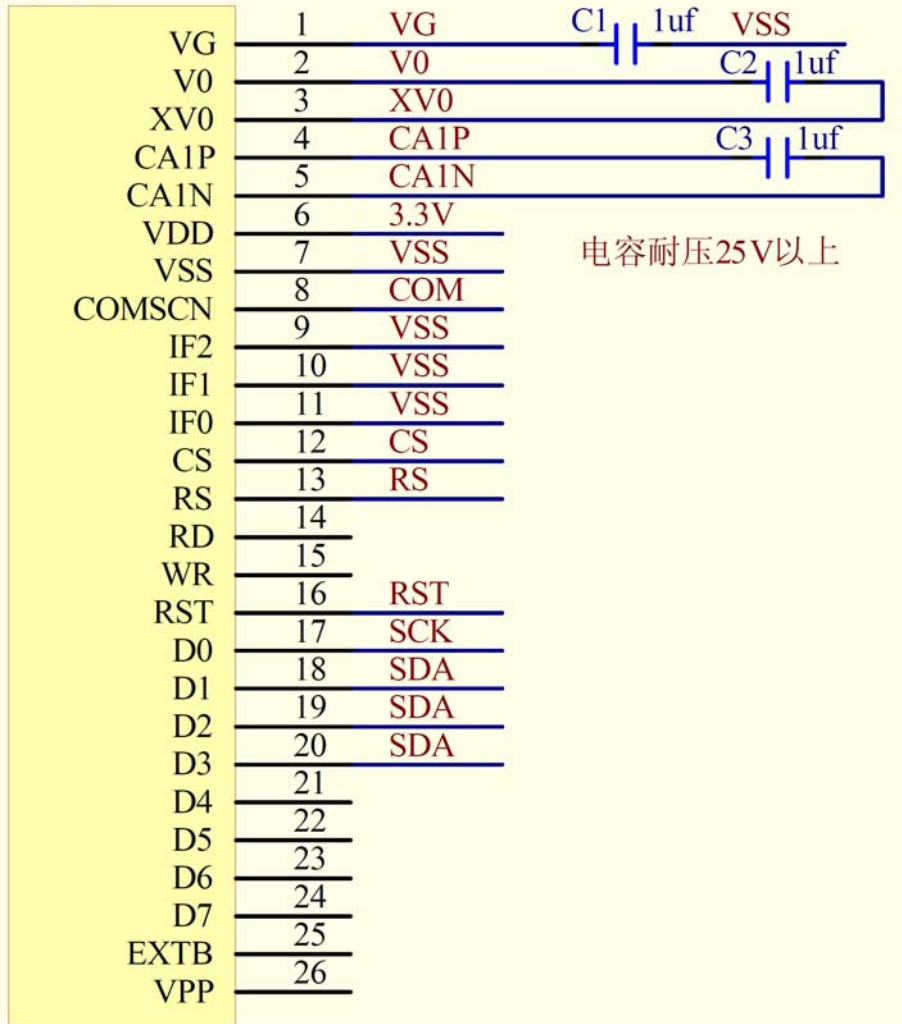
7.5.1 串行接口

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:

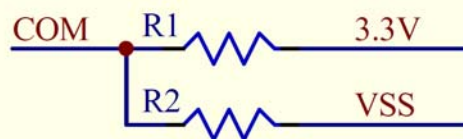


图 9. 串行接口

26PIN



26PIN



显示方向按图纸,COM接3.3V
显示方向旋转180度,COM接VSS

并程序与串行只是接口定义、写数据和命令不一样，其它都一样

串程序:

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

sbit cs1=P3^4;    /*接口定义*/
sbit reset=P3^5;  /*接口定义*/
sbit rs=P3^3;     /*接口定义*/
sbit sclk=P1^0;   //对应 LCD 的 SCK(D0)
sbit sid=P1^1;    //对应 LCD 的 SDA(D1)

sbit key=P2^0;    /*按键接口, P2.0 口与 GND 之间接一个按键*/ #define uchar unsigned char
#define uint unsigned int

/*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(2000);
}

//=====transfer command to LCM=====
void transfer_command_lcd(int data1)
{
    char i;
    cs1=0;
    rs=0;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
}

//-----transfer data to LCM-----
void transfer_data_lcd(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;  
}  
}
```

7.6、IIC 接口

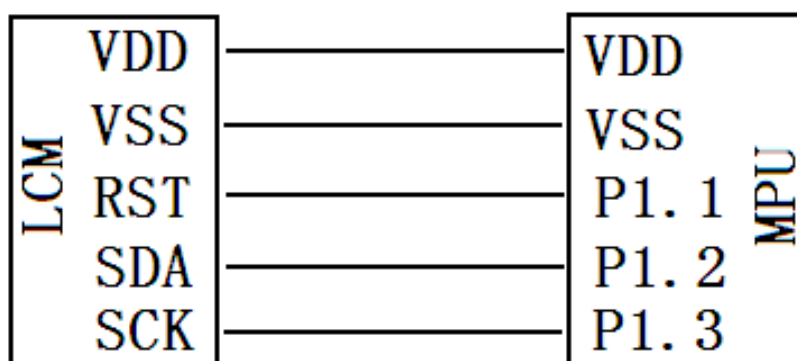
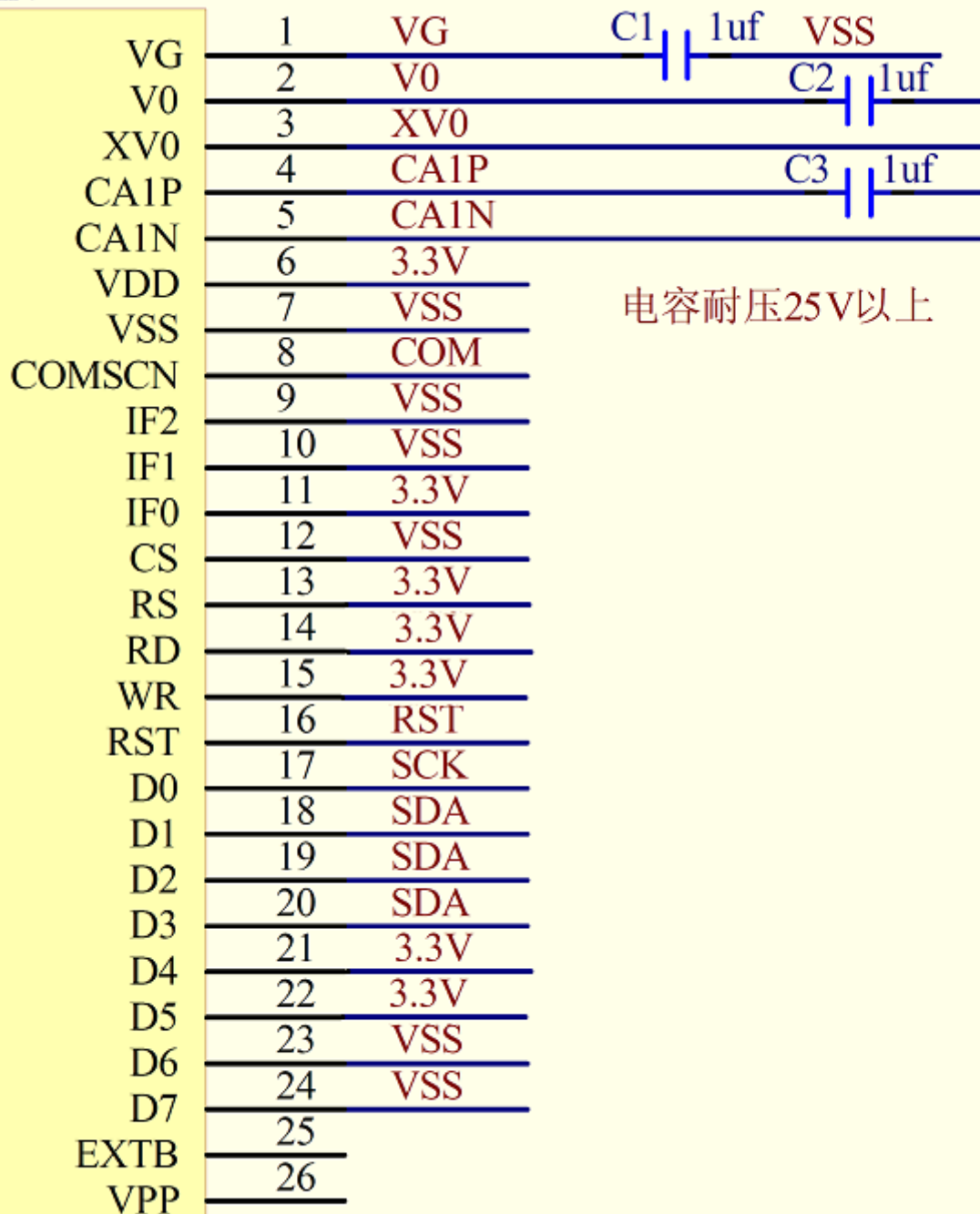


图 10. IIC

26PIN



26PIN

7.6.1、以下为 IIC 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：

```
// 液晶演示程序 JLX256128G-92906，IIC 接口！
// 驱动 IC 是：ST75256

sbit reset=P1^1; //对应 LCD 的 RST 引脚
sbit scl=P1^3; //对应 LCD 的 D0 引脚
sbit sda=P1^2; //对应 LCD 的 D1（D1、D2、D3 短接）引脚

sbit key=P2^0; //我司测试主板按键接口的 SDA(D1)
```

```
#define DataBus P1
#define uchar unsigned char
#define uint unsigned int
```

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

```
//延时2
void delay_us(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<10;k++);
}
```

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

```
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(0x7c);  
    transfer(com);  
    stop_flag();  
}
```

//写数据到液晶显示模块

```
void transfer_data(uchar dat)
```

```
{  
    start_flag();  
    transfer(0x7e);  
    transfer(dat);  
    stop_flag();  
}
```

-END-

