

# JLX320160G-976-PN 使用说明书

## 目 录

序号	内 容 标 题	页 码
1	概述	2
2	特点	2
3	外形及接口引脚功能	3~5
4	电路框图及背光参数	5
5	技术参数	5~6
6	时序特性	6~9
7	指令表及硬件接口、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 JLX320160G-976-PN 型液晶模块由于使用方便、显示清晰，广泛应用于各种人机交流面板。

JLX320160G-976-PN 可以显示 320 列\*160 行点阵单色或 4 灰度级的图片，或显示 10 个/行\*5 行 32\*32 点阵或显示 13 个/行\*6 行 24\*24 点阵的汉字，或显示 20 个/行\*10 行 16\*16 点阵的汉字，或显示 8\*16 点阵的英文、数字、字符 40 个\*10 行，或显示 5\*8 点阵的英文、数字、字符 53 个\*20 行。

## 2. JLX320160G-976-PN 图像型点阵液晶模块的特性

2.1 结构牢，焊接式 FPC，带铁框。

2.2 IC 采用矽创公司 ST7586S, 功能强大，稳定性好

2.3 功耗低不带背光 1mW (3.3V\*0.7mA)，带背光不大于 302mW (3.0V\*100mA)；

2.4 接口简单方便:可采用 4 线 SPI 串行接口、并行接口。

2.5 工作温度宽：-20℃ ~ +70℃；

2.6 储存温度宽：-30℃ ~ +80℃；

2.7 显示内容：

- 320\*160 点阵单色或 4 灰度级图片；

- 或显示 10 个×5 行 32\*32 点阵的汉字；

- 或显示 13 个×6 行 24\*24 点阵的汉字；

- 或显示 20 个×10 行 16\*16 点阵的汉字；

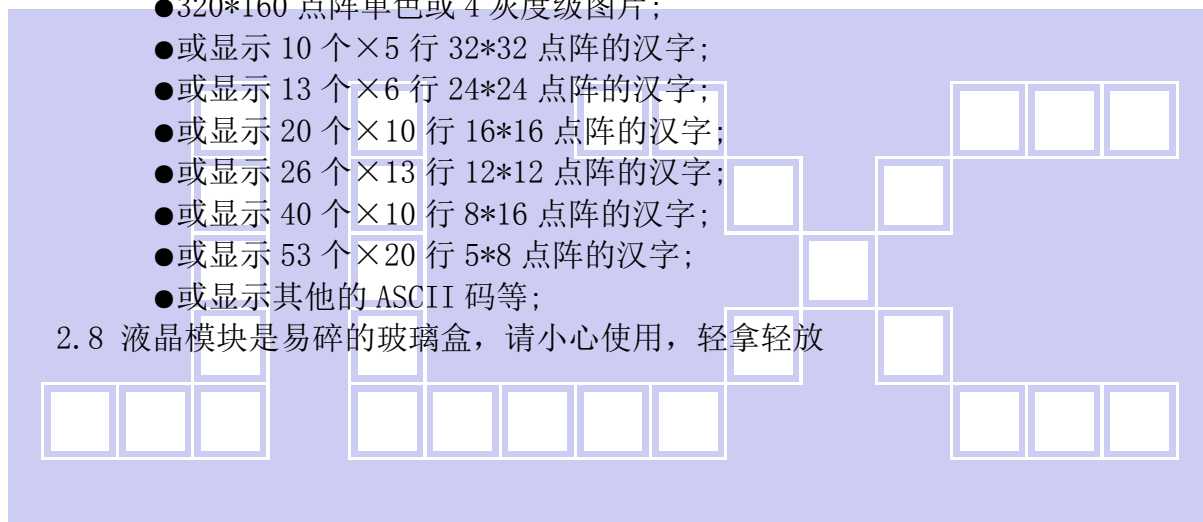
- 或显示 26 个×13 行 12\*12 点阵的汉字；

- 或显示 40 个×10 行 8\*16 点阵的汉字；

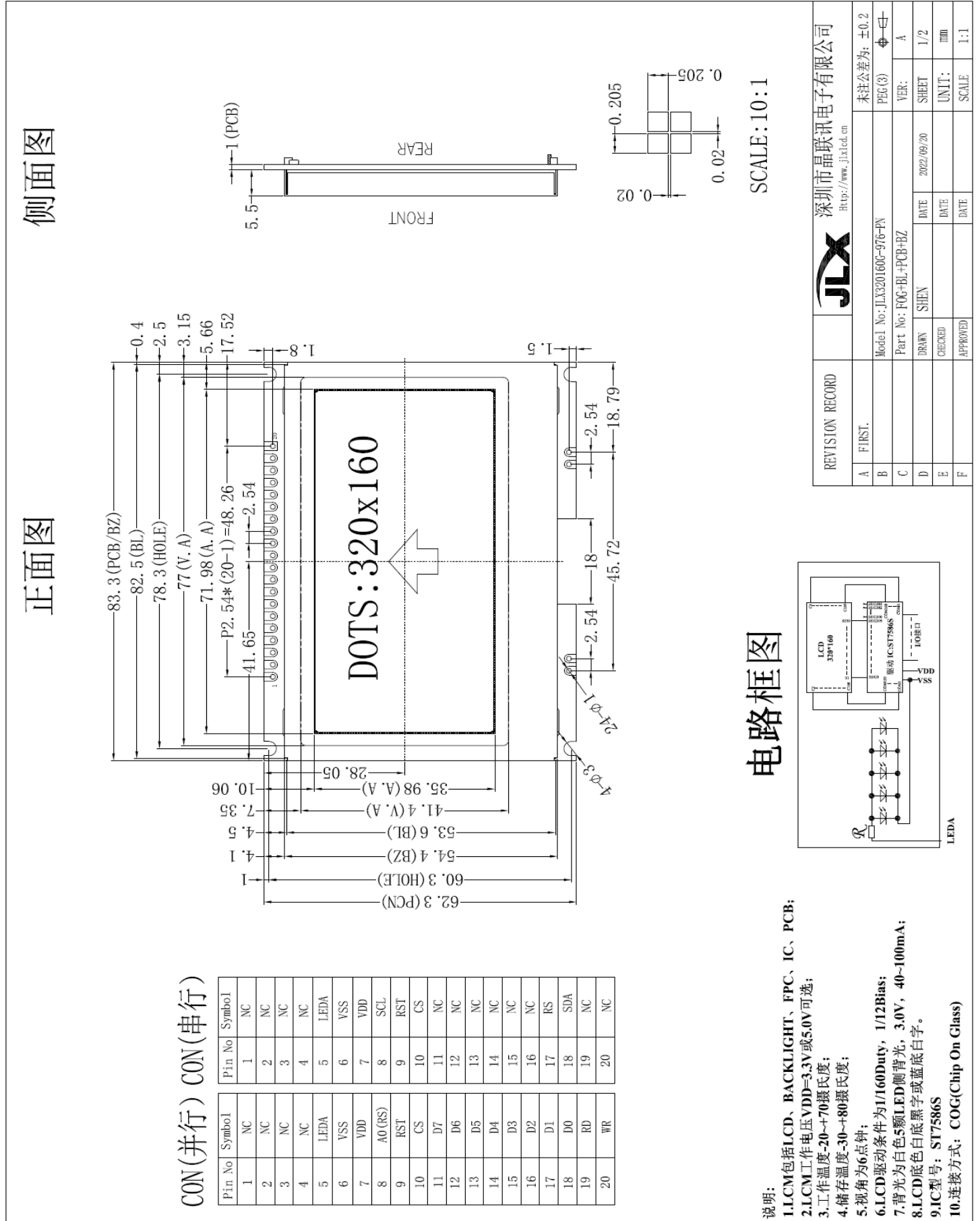
- 或显示 53 个×20 行 5\*8 点阵的汉字；

- 或显示其他的 ASCII 码等；

2.8 液晶模块是易碎的玻璃盒，请小心使用，轻拿轻放



## 3. 外形尺寸及接口引脚功能:



<b>JLX</b>		深圳市晶联讯电子有限公司 Http://www.jlxlcd.cn	
Model No.: JLX320160G-976-PN		未注公差为: ±0.2	
Part No.: FOC+BL+PCB+BZ		PEG (3)	Φ
DRAWN	SHEN	VER:	A
CHECKED		DATE	2022/09/20
APPROVED		DATE	
		SHEET	1/2
		UNIT:	mm
		SCALE	1:1

REVISION RECORD	
A	FIRST.
B	
C	
D	
E	
F	

图 1. 液晶模块外形尺寸

### 3.1 模块的接口引脚功能

#### 3.1.1 并口接口引脚功能说明：

引线号	符号	名称	功能
1	NC	空脚	空脚
2	NC	空脚	空脚
3	NC	空脚	空脚
4	NC	空脚	空脚
5	LEDA	背光电源正极	背光电源正极，常亮可短接到VDD
6	VSS	接地	供电电源负极
7	VDD	供电电源正极	供电电源正极（购买时需选择3.3V或5.0V供电）
8	A0 (RS)	寄存器选择信号	H:数据寄存器 0:指令寄存器
9	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
10	CS	片选	低电平片选
11	D7	I/O	数据总线
12	D6	I/O	数据总线
13	D5	I/O	数据总线
14	D4	I/O	数据总线
15	D3	I/O	数据总线
16	D2	I/O	数据总线
17	D1	I/O	数据总线
18	D0	I/O	数据总线
19	E(RD)	使能信号 (读)	6800 时序时：E:使能信号 8080 时序时：读信号
20	R/W(WR)	读/写 (写)	6800 时序时：RW: H:读信号 L:写信号 8080 时序时：写信号

表 1：并口引脚功能

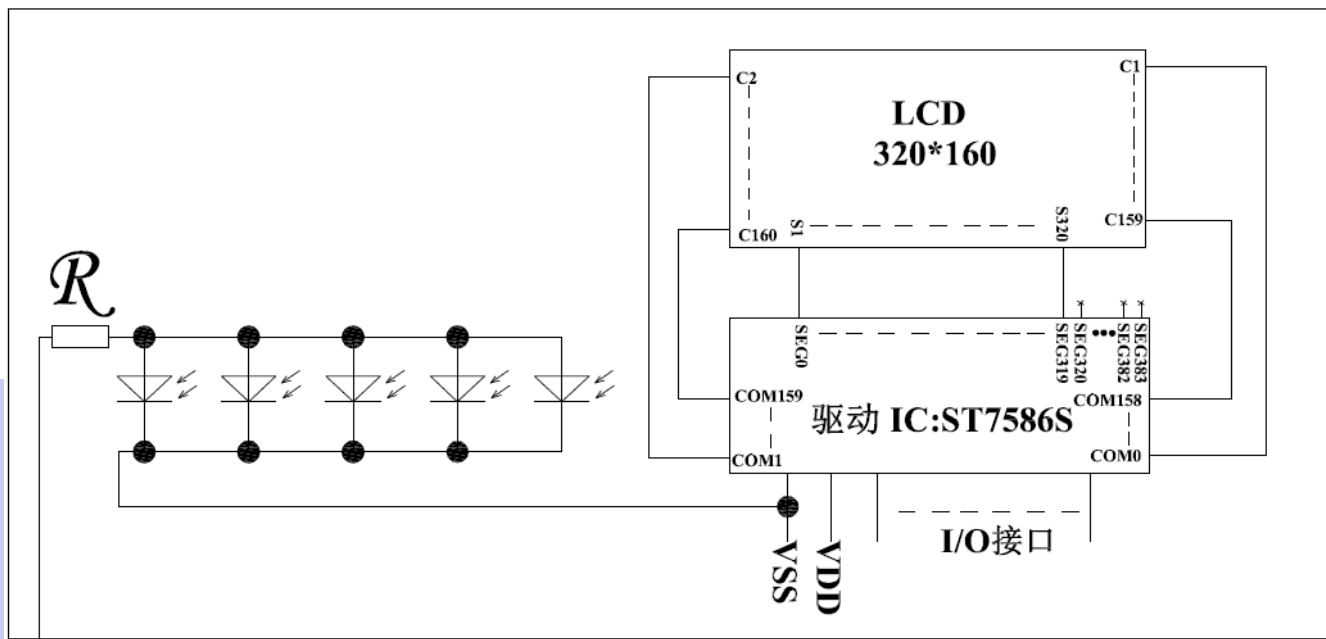
#### 3.1.2 串口接口引脚功能说明：

引线号	符号	名称	功能
1	NC	空脚	空脚
2	NC	空脚	空脚
3	NC	空脚	空脚
4	NC	空脚	空脚
5	LEDA	背光电源正极	背光电源正极，常亮可短接到VDD
6	VSS	接地	供电电源负极
7	VDD	供电电源正极	供电电源正极（购买时需选择3.3V或5.0V供电）
8	A0 (RS)	SCL	串行时钟SCL
9	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
10	CS	片选	低电平片选
11	D7	悬空或接VDD	悬空或接VDD
12	D6	悬空或接VDD	悬空或接VDD
13	D5	悬空或接VDD	悬空或接VDD
14	D4	悬空或接VDD	悬空或接VDD
15	D3	悬空或接VDD	悬空或接VDD
16	D2	悬空或接VDD	悬空或接VDD

17	D1	RS, 寄存器选择信号	RS: H:数据寄存器 0:指令寄存器
18	D0	SDA	串行数据, SDA
19	E(RD)	悬空或接VDD	悬空或接VDD
20	R/W(WR)	悬空或接VDD	悬空或接VDD

表 2: 串口引脚功能

## 4. 电路框图



### LEDA

图 2: JLX320160G-976-PN 图像点阵型液晶模块的电路框图

### 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:  $(8 \sim 15) \times 5 = 40 \sim 100\text{mA}$  (LED 灯数共 5 颗);

工作电压: 3.0V; (PCB 已加限流电阻, 工作电压同 VDD)

## 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 - XV0	-0.3	8.3	19	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 3: 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 (5.0V)	VDD	—	4.5	5.0	5.2	V
工作电压 (3.3V)	VDD	—	2.7	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	IOH = 0.2mA	0.8VDD	—	VDD	V
输出低电平	VOO	IOO = 1.2mA	0	—	0.2VDD	V
模块工作电流	IDD	VDD = 3.0V	—	0.7	1.0	mA
背光工作电流	ILED	VLED=3.0V	40	75	100	mA

表 4: 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)

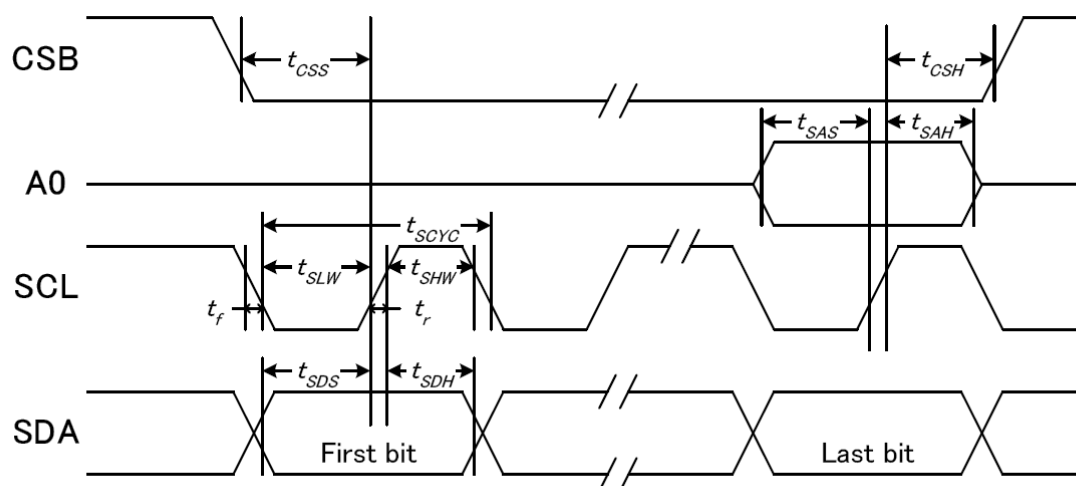


图 3. 从 CPU 写到 ST7586S (Writing Data from CPU to ST7586S)

表 5. 写数据到 ST7586S 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	200	--	--	ns
保持SCK高电平脉宽 (SCL "H" pulse width)	tSHW		140	--	--	ns
保持SCLK低电平脉宽 (SCL "L" pulse width)	tSLW		60	--	--	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	30	---	---	ns
片选信号保持时间 (CS-SCL time)	tCSH		30	---	---	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)

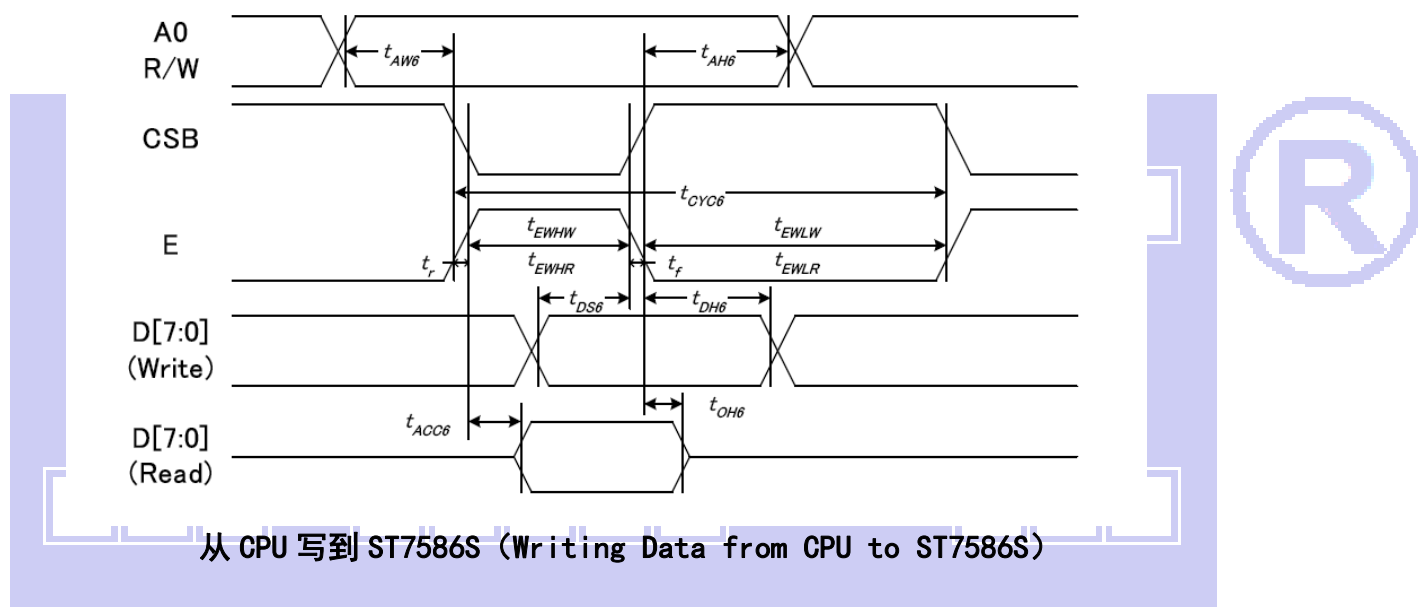


图 4. 写数据到 ST7586S 的时序要求 (6800 系列 MPU)

表 6. 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	0		---	ns
地址建立时间		tAW6	0		---	ns
系统循环时间	E	tCYC6	240		---	ns
使能“低”脉冲宽度		tEHLW	100		---	ns
使能“高”脉冲宽度		tEHWL	100		---	ns
写数据建立时间	DB[7: 0]	tDS6	20		---	ns
写数据保持时间		tDH6	20		---	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,

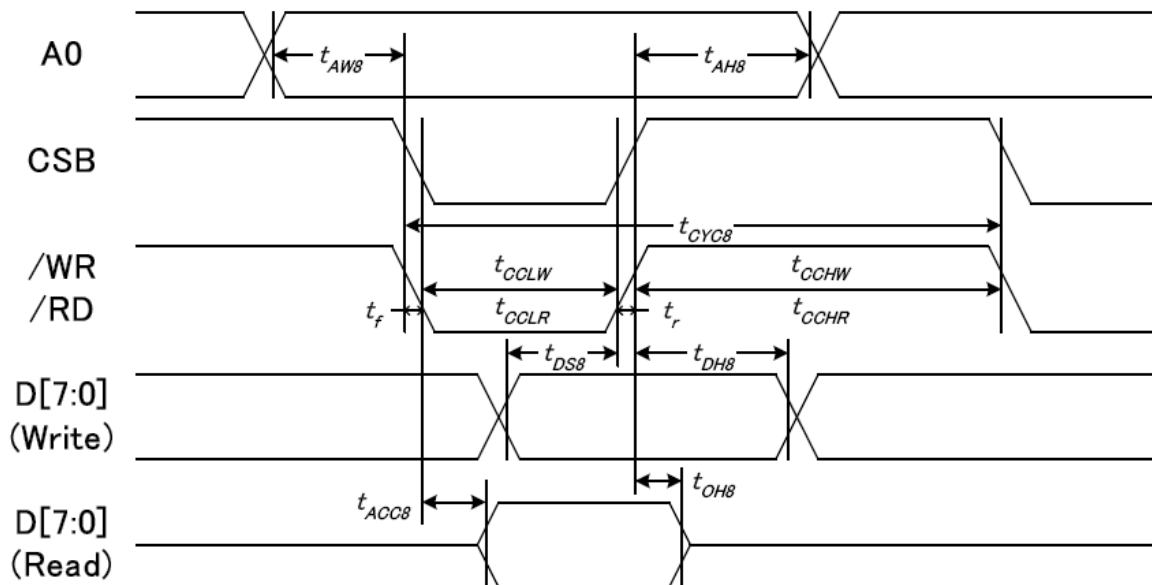
(TR + TF) ≤ (tcyc6 - tewlw - tewhw) 指定。

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 ST7586S (Writing Data from CPU to ST7586S)

图 5. 写数据到 ST7586S 的时序要求 (8080 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	tAH8	0		--	ns
地址建立时间		tAW8	0		--	ns
系统循环时间	/WR	tCYC8	240		--	ns
使能“低”脉冲宽度		tCCLW	100		--	ns
使能“高”脉冲宽度		tCCHW	100		--	ns
写数据建立时间	DB	tDS8	20		--	ns
写数据保持时间		tDH8	20		--	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



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

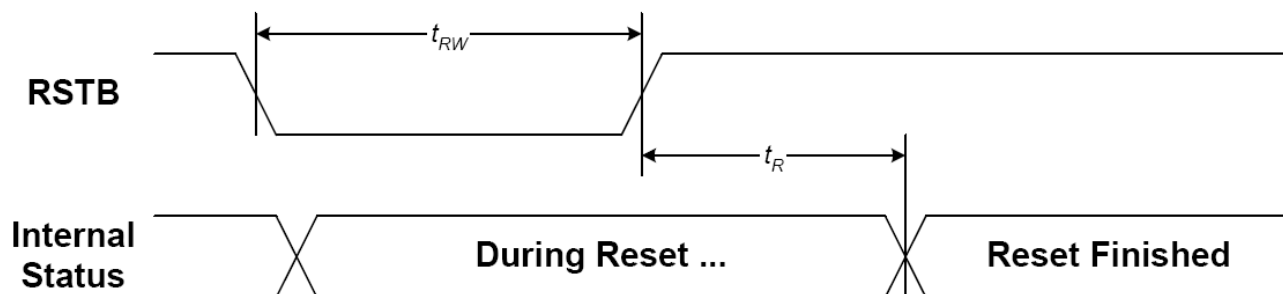


图 6: 电源启动后复位的时序

表 8: 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	
复位保持低电平的时间	$T_{RW}$		100	--	--	ms
复位时间	$T_R$		120	--	--	ms

7. 指令功能:

7.1 指令表

表 9

指令名称	指令码										说明
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(1) 空指令 (NOP)	0	0	0	0	0	0	0	0	0	0	空操作
(2) 软件复位 (Reset)	0	0	0	0	0	0	0	0	0	1	<b>0x01</b> : 软件复位。
(3) 省电/睡眠模式 (Power Save)	0	0	0	0	0	1	0	0	0	0	省电/睡眠模式 <b>0x10</b> : 进入省电模式 <b>0x11</b> : 退出省电模式
(4) 局部模式 (Partial Mode)	0	0	0	0	0	1	0	1	0	0	局部模式开/关 <b>0x12</b> : 开, <b>0x13</b> : 关
(5) 显示正显/反显 (Display normal/reverse)	0	0	0	0	1	0	0	0	0	1	显示正显/反显: <b>0x20</b> : 常规: 正显 <b>0x21</b> : 反显
(6) 全部点阵开/关 (All Pixel ON/OFF)	0	0	0	0	1	0	0	0	1	0	全部点阵开/关 <b>0x22</b> : 全部点阵关 <b>0x23</b> : 全部点阵开
(7) 显示开/关 (Display ON/OFF)	0	0	0	0	1	0	1	0	0	0	显示开/关: <b>0x28</b> : 关, <b>0x29</b> : 开
(8) 列地址设置 (Set Column Address)	0	0	0	0	1	0	1	0	1	0	列地址设置: 起始列地址范围: <b>0x00~0x9f</b> 结束列地址范围: <b>0x00~0x9f</b>
	1	0	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	
	1	0	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	
	1	0	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	
(9) 页地址设置 (Set Row Address)	0	0	0	0	1	0	1	0	1	1	页地址设置: 起始页地址范围: <b>0x00~0x9f</b> 结束页地址范围: <b>0x00~0x9f</b>
	1	0	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	
	1	0	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	
	1	0	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	
(10) 写数据到液晶屏 (Write Display Data)	0	0	0	0	1	0	1	1	0	0	<b>0x2C</b> : 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 Display Data)	0	0	0	0	1	0	1	1	1	0	<b>0x2E</b> : 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 Display Area)	0	0	0	0	1	1	0	0	0	0	<b>0x30</b> : 指定显示区域
	1	0	PTS15	PTS14	PTS13	PTS12	PTS11	PTS10	PTS9	PTS8	起始区域地址: <b>00h≤PTS≤9Fh</b>
	1	0	PTS7	PTS6	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0	结束区域地址: <b>00h≤PTE≤9Fh</b>
	1	0	PTE15	PTE14	PTE13	PTE12	PTE11	PTE10	PTE9	PTE8	显示区域: <b>64h≤Duty≤160h</b>
(13) 指定显示滚动区域 (Scroll Area)	0	0	0	0	1	1	0	0	1	1	<b>0x33</b> : 滚动区域设置
	1	0	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0	起始区域地址: <b>TA=00h~A0h</b>
	1	0	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	滚动区域: <b>SA=00h~A0h</b>
	1	0	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	结束区域地址: <b>BA=00h~A0h</b> TA+SA+BA=160
(14) 控制液晶屏显示	0	0	0	0	1	1	0	1	1	0	<b>0x36</b> : 显示控制

(Display Control)	1	0	MY	MX1	0	0	MX0	0	0	0	MY=0: COM0→COM159 MY=1: COM159→COOM0 MX[1:0]=(0,0): SEG0→SEG383 MX[1:0]=(1,1): SEG383→SEG0
(15) 显示初始行 (Start Line)	0	0	0	0	1	1	0	1	1	1	<b>0x37:</b> 滚动开始初始行设置 S=00h~9Fh
	1	0	S7	S6	S5	S4	S3	S2	S1	S0	
(16) 显示模式 (Display Mode)	0	0	0	0	1	1	1	0	0	0	<b>0xF0:</b> 显示模式设置 <b>0x39:</b> 黑白模式 <b>0x38:</b> 4 灰级度模式
(17) 数据显示接口 (Enable DDRAM Interface)	0	0	0	0	1	1	1	0	1	0	<b>0x3a:</b> 使能数据显示模式 0x02:黑白和 4 灰级模式 0x03:16 灰级模式
	1	0	0	0	0	0	0	0	1	0	
(18) 显示点空比 (Display Duty)	0	0	1	0	1	1	0	0	0	0	<b>0xb0:</b> 点空比范围: DT=03~9f
	1	0	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	
(19) 设置第一行输出 (First Output COM)	0	0	1	0	1	1	0	0	0	1	<b>0xb1:</b> 第一行输出设置范围: FC=00~9f
	1	0	FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0	
(20) FOSC分频 (FOSC Divider)	0	0	1	0	1	1	0	0	1	1	<b>0xb3:</b> 设置 FOSC 分频
	1	0	0	0	0	0	0	0			
(21) 指定区域显示 (Partial Display)	0	0	1	0	1	1	0	1	0	0	<b>0xb4:</b> 指定区域显示
	1	0	1	0	1	0	0	0	0	0	
(22) N行反显 (N-Line Inversion)	0	0	1	0	1	1	0	1	0	1	<b>0xb5:</b> N 行反显
	1	0	M	0	0	NL4	NL3	NL2	NL1	NL0	
(23) 读-改-写 (Read Modify Write)	0	0	1	0	1	1	1	0	0	0	<b>读改写控制:</b> <b>0xB8:</b> 启用读改写 <b>0xB9:</b> 禁用读改写
	1	0								1	
(24) 液晶内部电压设置 (Set Vop)	0	0	1	1	0	0	0	0	0	0	<b>0xc0:</b> 对比度设置 微调对比度范围: 0x00~0xff 粗调对比度范围: 0x01
	1	0	Vop7	Vop6	Vop5	Vop4	Vop3	Vop2	Vop1	Vop0	
	1	0	-	-	-	-	-	-	-	Vop8	
(25) 对比度VOP增加 (Vop Increase)	0	0	1	1	0	0	0	0	0	1	<b>0xc1:</b> Vop 增加一级
(26) 对比度VOP降低 (Vop Decrease)	0	0	1	1	0	0	0	0	1	0	<b>0xc2:</b> Vop 降低一级
(27) LCD 偏压比设置 (BIAS System)	0	0	1	1	0	0	0	0	1	1	<b>0xc3:</b> 设置偏压比:
	1	0	-	-	-	-	-	BS2	BS1	BS0	
(28) 升压倍数 (Booster Level)	0	0	1	1	0	0	0	1	0	0	<b>0xc4:</b> 内建升压倍数设置
	1	0	-	-	-	-	-	BST2	BST1	BST0	
(29) 模拟电路 (Analog Control)	0	0	1	1	0	1	0	0	0	0	<b>0xd0:</b> 模拟电路 <b>0xd1:</b> 使能模拟电路
	1	0	0	0	0	1	1	1	0	1	
(30) 自动读取控制 (Auto Read Control)	0	0	1	1	0	1	0	1	1	1	<b>0xd7:</b> 自动读取控制设置 <b>0x8f:</b> 启用自动读取控制 <b>0x9f:</b> 禁用自动读取控制
	1	0	1	0	0	0	1	1	1	1	
(31) 控制OTP读写	0	0	1	1	1	0	0	0	0	0	<b>0xe0:</b> OTP 读写

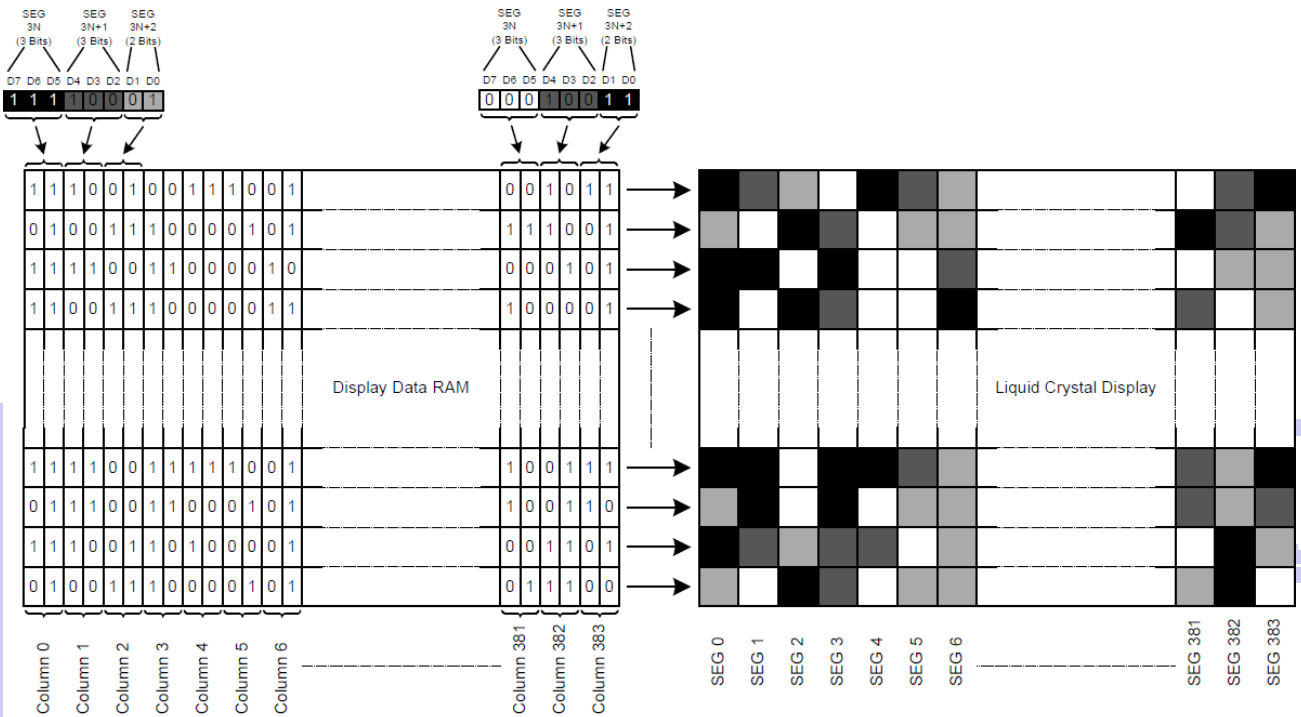
(OTP WR/RD Control)	1	0	0	0	WR RD	0	0	0	0	0	WR/RD=0; 0x00,使能 OTP 读 ER/RD=1; 0x20,使能 OTP 写
(32) 控制OTP输出 (OTP Control Out)	0	0	1	1	1	0	0	0	0	1	0xe1: OTP 输出
(33) 写OTP (OTP Write)	0	0	1	1	1	0	0	0	1	0	0xe2: 写 OTP 程序
(34) 读OTP (OTP Read)	0	0	1	1	1	0	0	0	1	1	0xe3: 读 OTP 程序
(35) OTP控制 (OTP Selection Control)	0	0	1	1	1	0	0	1	0	0	0xe4: OTP 控制设置 0x19: 禁用 OTP 0x59: 启用 OTP
	1	0	0	Ctrl	0	1	1	0	0	1	
(36) OTP 编程设置 (OTP Programming Setting)	0	0	1	1	1	0	0	1	0	1	0xe5: OTP 编程设置 0x0f: 启用 OTP 编程
	1	0	0	0	0	0	1	1	1	1	
(37) 灰度帧速率 Frame Rate (Gray Scale Mode)	0	0	1	1	1	1	0	0	0	0	0xf0: 灰度帧速率设置
	1	0	-	-	-	FRA4	FRA3	FRA2	FRA1	FRA0	
	1	0	-	-	-	FRB4	FRB3	FRB2	FRB1	FRB0	
	1	0	-	-	-	FRC4	FRC3	FRC2	FRC1	FRC0	
(38)单色/黑白帧速率 Frame Rate (Monochrome Mode)	0	0	1	1	1	1	0	0	0	1	0xf1: 黑白帧速率设置
	1	0	-	-	-	FRA4	FRA3	FRA2	FRA1	FRA0	
	1	0	-	-	-	FRB4	FRB3	FRB2	FRB1	FRB0	
	1	0	-	-	-	FRC4	FRC3	FRC2	FRC1	FRC0	
	1	0	-	-	-	FRD4	FRD3	FRD2	FRD1	FRD0	
(39)温度补偿范围 Temperature Range	0	0	1	1	1	1	0	0	1	0	0xf2: 温度补偿范围设置
	1	0	-	TA6	TA5	TA4	TA3	TA2	TA1	TA0	
	1	0	-	TB6	TB5	TB4	TB3	TB2	TB1	TB0	
	1	0	-	TC6	TC5	TC4	TC3	TC2	TC1	TC0	
(40) 温度梯度补偿 Temperature Gradient Compensation	0	0	1	1	1	1	0	1	0	0	0xf4: 温度梯度补偿系数设置
	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		
(41) PWM帧设置 Frame PWM Set	0	0	1	1	1	1	1	0	0	1	0xf9:灰度等级设置
	1	0	-	-	-	P14	P13	P12	P11	P10	
	1	0	-	-	-	P24	P23	P22	P21	P20	
	:	:	:	:	:	:	:	:	:	:	
	0	0	-	-	-	P154	P153	P152	P151	P150	
	0	0	-	-	-	P164	P163	P162	P161	P160	

请详细参考 IC 资料”ST7586S.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 320\*160 点阵的屏分为 20 个“页”，从第 0“页”到第 19“页”。

**DB7--DB0 的排列方向：数据是从左到右排列的。最高位 D7 是在最左边，最低位 D0 是在最右边。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：**

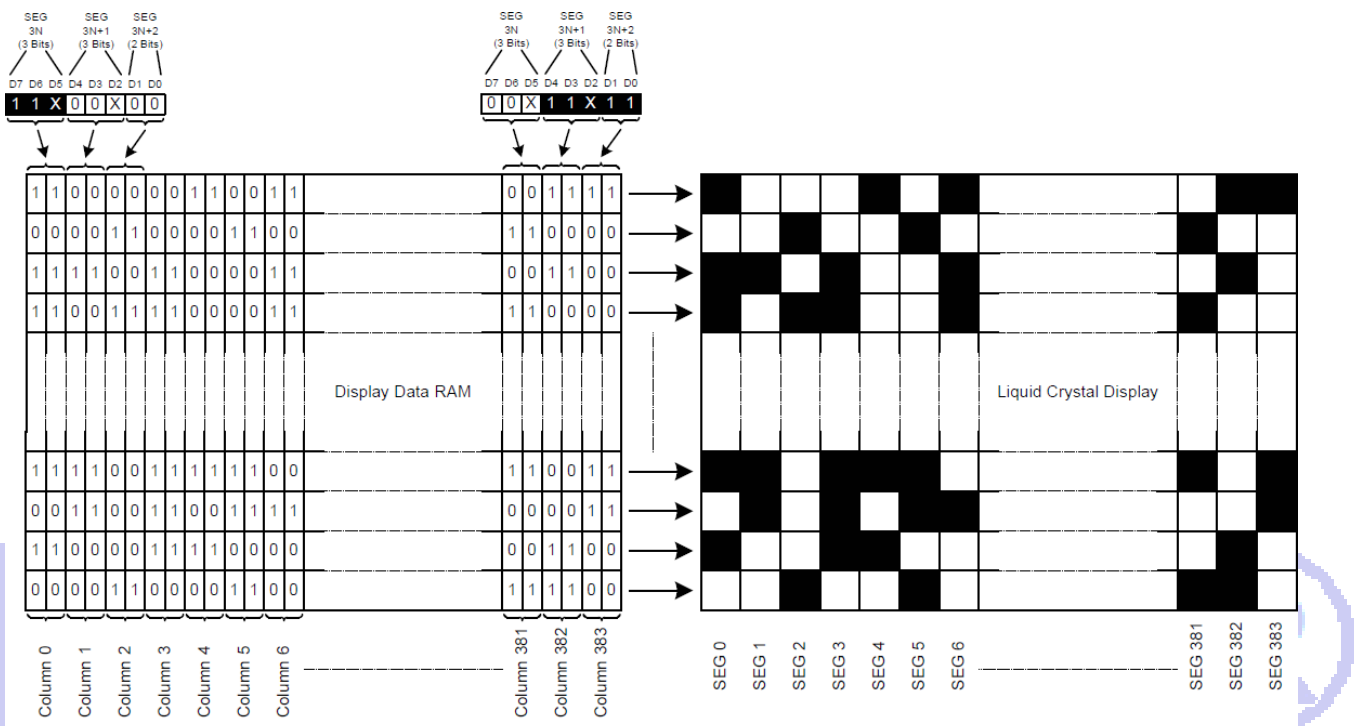


3 Bits Data			DDRAM	LCD
D7 (D4)	D6 (D3)	D5 (D2)		
1	1	1	1	1
0	0	0	0	0
1	0	0	1	0
0	1	0	0	1

2 Bits Data		DDRAM	LCD
D1	D0		
1	1	1	1
0	0	0	0
1	0	1	0
0	1	0	1

Fix LSB to 0 if Gray Mode

Fig. 4 DDRAM Mapping (4-Level Gray Scale Mode)



3 Bits Data			DDRAM	LCD
D7 (D4)	D6 (D3)	D5 (D2)		
1	1	X	1	1
0	0	X	0	0

2 Bits Data		DDRAM	LCD
D1	D0		
1	1	1	1
0	0	0	0

Fig. 5 DDRAM Mapping (Monochrome Mode)

下图摘自 ST7586s IC 资料, 可通过“ST7586s.PDF”之第 21 页获取最佳效果。

### LCD Display Function

#### DDRAM Map to LCD Driver Output

The internal relation between DDRAM and LCD driver circuit (SEG/COM output path) with different MX or MY setting is illustrated below.

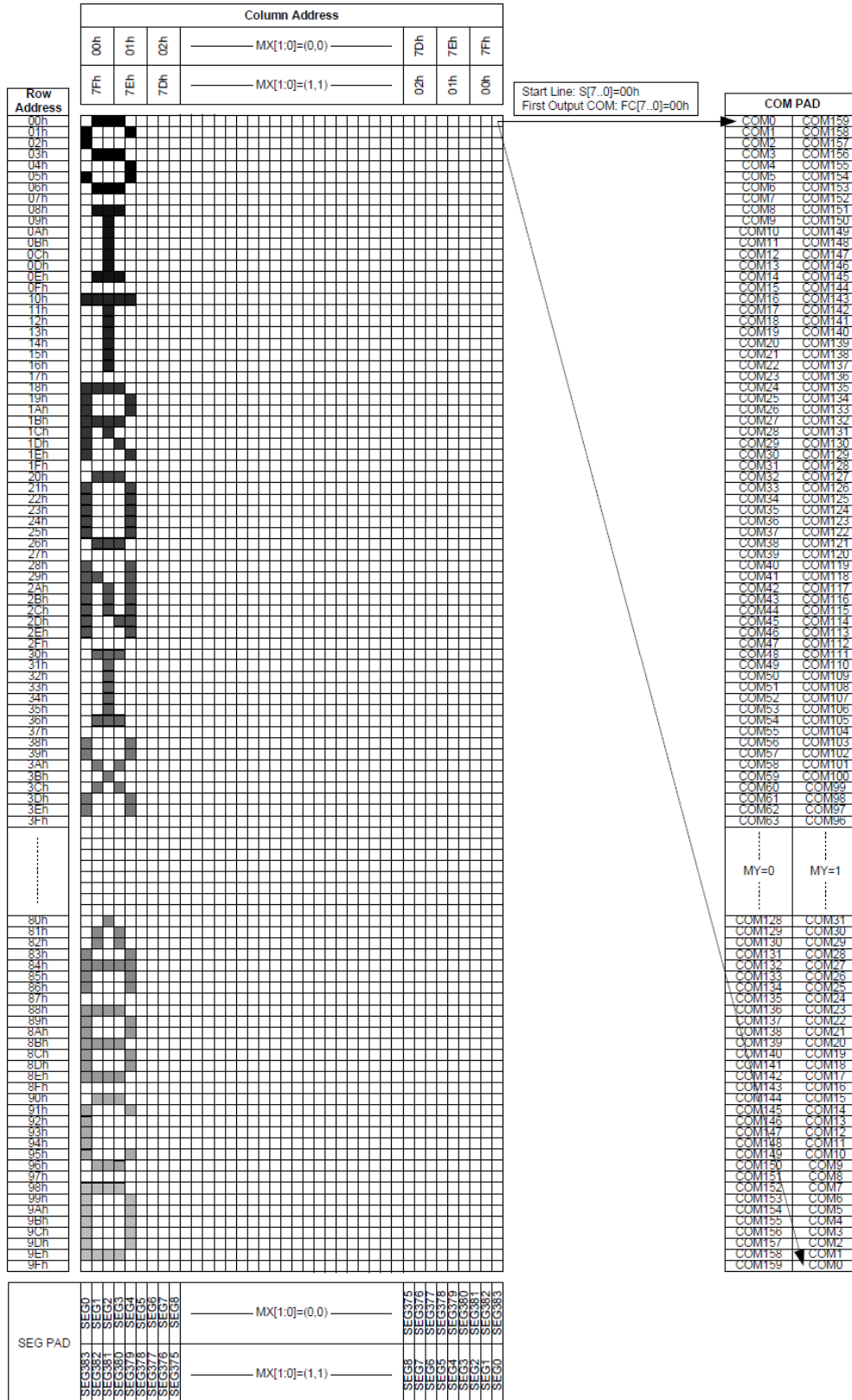
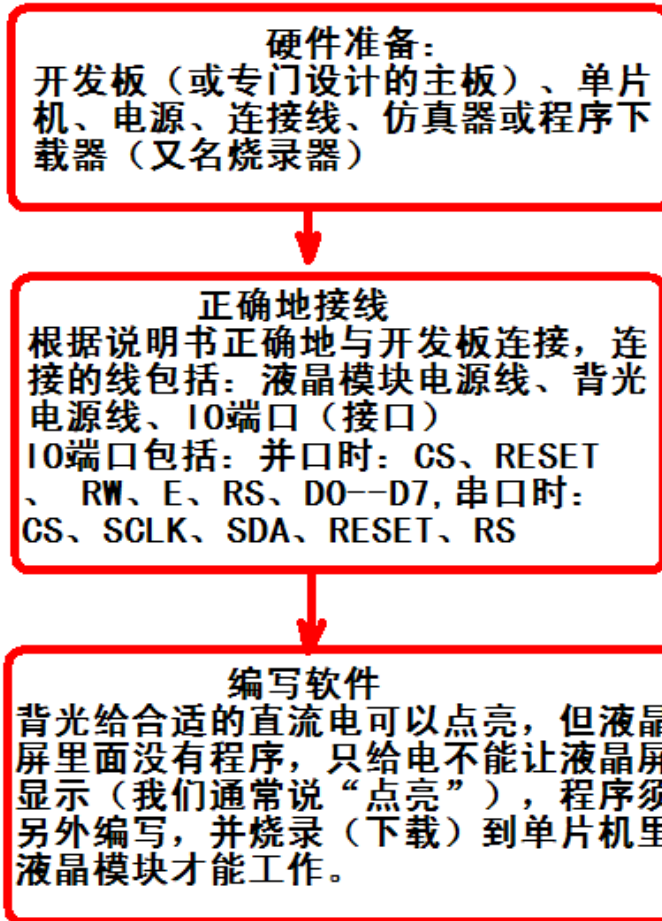


Fig. 6 DDRAM Display Direction

### 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 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:

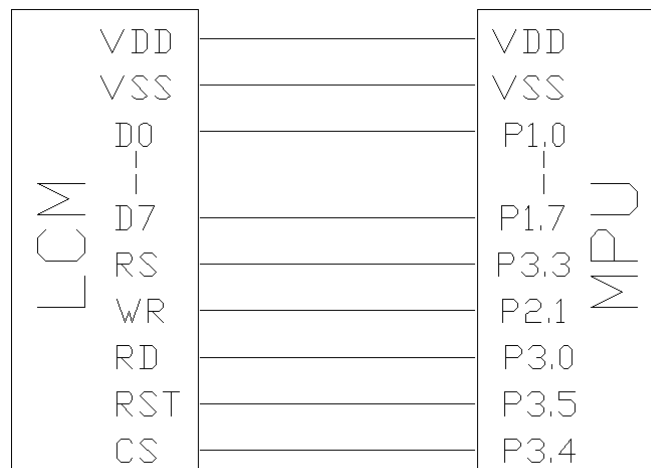


图 7: 并行接口图



```

/* 液晶模块型号: JLX320160G-976
   并行接口: 6800 时序
   驱动 IC 是: ST75256
   版权所有: 晶联讯电子: 网址 http://www.jlxlcd.cn;
*/
#include <STC15F2K60S2.H>
#include <intrins.h>
#include <chinese_code.h>

sbit CS=P3^4;      /*对应 LCD 的 CS 引脚*/
sbit RST=P3^5;    /*对应 LCD 的 RST 引脚*/
sbit RS=P3^3;     /*对应 LCD 的 RS 引脚*/
sbit E=P3^0;      /*对应 LCD 的 E(RD) 引脚*/
sbit WR=P2^1;    /*对应 LCD 的 RW(WR) 引脚。另外 P1.0~1.7 对应 D0~D7*/
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
```

```
/*延时: 1 毫秒的 i 倍*/
```

```
void delay_ms (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=====
```

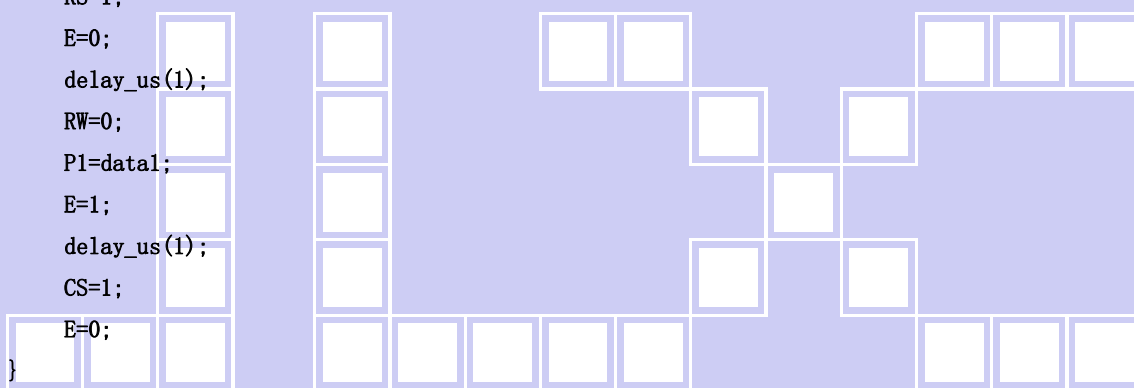


```
void transfer_command_lcd(int data1)
{
    CS=0;
    RS=0;
    E=0;
    delay_us(1);
    RW=0;
    P1=data1;
    E=1;
    delay_us(1);
    CS=1;
    E=0;
}
```

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

```
void transfer_data_lcd(int data1)
{
```

```
    CS=0;
    RS=1;
    E=0;
    delay_us(1);
    RW=0;
    P1=data1;
    E=1;
    delay_us(1);
    CS=1;
    E=0;
}
```




```
void initial_lcd()
```

```
{
    RST=1;
    RST=0;
    delay_ms(100);
    RST=1;
    delay_ms(100);

    transfer_command_lcd(0x11);    //退出睡眠模式

    transfer_command_lcd(0xC0);    // 设置 VOP
    transfer_data_lcd(0x44);       // 设置 VOP 的值的低 8 位 (总共 9 位)
    transfer_data_lcd(0x01);       // 设置 VOP 的值的第 9 位, 也是最高一位
    transfer_command_lcd(0xC3);    // 设置 BIAS
    transfer_data_lcd(0x02);       // 00: BIAS = 1/14  02 = 1/12
    transfer_command_lcd(0xC4);    // 设置升压倍数
    transfer_data_lcd(0x07);       // 07: 8 倍压
}
```

```

transfer_command_lcd(0xD0);    // 允许模拟电路
transfer_data_lcd(0x1D);      // 允许模拟电路

transfer_command_lcd(0xB5);    // N-Line = 13
transfer_data_lcd(0x00);      // 8d

transfer_command_lcd(0x38);    // 0x38: 设置为灰度模式; 0x39: 设置为黑白模式。
transfer_command_lcd(0x3A);    // 允许 DDRAM 接口: 单色模式、4 灰度级、16 灰度级;
transfer_data_lcd(0x02);      // 0x03:16 灰度级; 0x02:4 灰度级或单色模式。

transfer_command_lcd(0x36);    // 扫描顺序设置
transfer_data_lcd(0x00);      // 扫描顺序设置:MX=1,MY=1: 从左到右, 从上到下的扫描顺序
transfer_command_lcd(0xB0);    // Duty 设置
transfer_data_lcd(0x9f);      // Duty 设置:1/160
transfer_command_lcd(0x20);    // 反显设置: OFF
    
```

```

transfer_command_lcd(0xf1);    //温度补偿, 温度变化改变帧频
transfer_data_lcd(0x15);
transfer_data_lcd(0x15);
transfer_data_lcd(0x15);
transfer_data_lcd(0x15);

transfer_command_lcd(0xb1);    // 扫描起始行设置
transfer_data_lcd(0x00);      // 扫描起始行设置: 从 COM0 开始

transfer_command_lcd(0x29);    // 打开显示: 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)
{
    int x_end,y_end;

    x_end=x+(x_total-1)/3;
    y_end=y+y_total-1;

    transfer_command_lcd(0x2A);
    transfer_data_lcd((x>>8)&0x00ff);
    transfer_data_lcd(x&0x00ff);
    transfer_data_lcd(x_end>>8&0x00ff);
    transfer_data_lcd(x_end&0x00ff);
    transfer_command_lcd(0x2B);
    transfer_data_lcd((y>>8)&0x00ff);
    transfer_data_lcd(y&0x00ff);
    }
    
```

```
transfer_data_lcd(y_end>>8&0x00ff);
transfer_data_lcd(y_end&0x00ff);
}
```

//传送同一个地址的 3 个点阵的黑白的数据：比如 SEG0、SEG1、SEG2（这 3 个点阵是同一个列地址，无法分开）  
 //送数据时左起第 1 列的数据是“D7 D6 D5 D4 D3 D2 D1 D0”中的高 3 位——D7 D6 D5，第 2 列是中 3 位——D4 D3 D2，第 3 列是低两位——D1 D0。

```
void transfer_mono_data_3pixel(uchar mono_data)
{
```

```
    uchar gray_data=0;
```

```
    if(mono_data&0x80)
```

```
    {
```

```
        gray_data=0xe0; //二进制 11100000，就是给 D7、D6、D5 赋值
```

```
    }
```

```
    else
```

```
    {
```

```
        gray_data=0;
```

```
    }
```

```
    mono_data<<=1;
```

```
    if(mono_data&0x80)
```

```
    {
```

```
        gray_data+=0x1c; //二进制 00011100，就是给 D4、D3、D2 赋值
```

```
    }
```

```
    else;
```

```
    mono_data<<=1;
```

```
    if(mono_data&0x80)
```

```
    {
```

```
        gray_data+=0x03; //二进制 00000011，就是给 D1、D0 赋值
```

```
    }
```

```
    else;
```

```
    transfer_data_lcd(gray_data); //display 3 dots (seg_N, seg_N+1, seg_N+2)
```

```
}
```

//显示 6 个点阵

```
void transfer_mono_data_6pixel(uchar dat1)
```

```
{
```

```
    transfer_mono_data_3pixel(dat1);
```

```
    transfer_mono_data_3pixel(dat1<<3);
```

```
}
```

//显示 8 个点阵

```
void transfer_mono_data_8pixel(uchar dat1)
```

```
{
```

transfer\_mono\_data\_3pixel(dat1); //传送 dat1 的 D7\D6\D5 这 3 位，对应 3 个点阵(第 1、2、3 个)会显示出来；列地址是自动+1 的

```
transfer_mono_data_3pixel(dat1<<3); //传送 dat1 的 D4\D3\D2 这 3 位, 对应 3 个点阵(第 4、5、6 个) 会显示出来; 列地址是自动+1 的
```

```
transfer_mono_data_3pixel(dat1<<6); //传送 dat1 的 D1\D0 这 2 位, 对应 3 个点阵(第 7、8、9 个) 会显示出来
```

//这个液晶驱动 IC 的每个列地址管 3 个点阵, 无法分开, 所以第 7、8 个点阵会连累到第 9 个点阵, 结果是每次显示 9 个点阵, 只不过第 9 个点阵会补 “0”

```
//如果第 9 个点阵本来有显示内容, 就会被无情地清掉
```

```
}
```

```
//显示 9 个点阵
```

```
void transfer_mono_data_9pixel(uchar dat1, uchar dat2)
```

```
{
```

```
transfer_mono_data_6pixel(dat1); //先显示 6 个点阵
```

```
transfer_mono_data_3pixel((dat1<<6)|(dat2>>2)); //显示 dat1 的 D1、D0 和 dat2 的 D7 位, 对应 3 个点阵(第 7、7、9 个) 会显示出来; 列地址是自动+1 的
```

```
}
```

```
//显示 12 个点阵
```

```
void transfer_mono_data_12pixel(uchar dat1, uchar dat2)
```

```
{
```

```
transfer_mono_data_9pixel(dat1, dat2); //先显示 9 个点阵
```

```
transfer_mono_data_3pixel(dat2<<1); //传送 dat2 的 D6\D5\D4 这 3 位, 对应第 10、11、12 个个点阵会显示出来;
```

列地址是自动+1 的

```
}
```

```
//显示 15 个点阵
```

```
void transfer_mono_data_15pixel(uchar dat1, uchar dat2)
```

```
{
```

```
transfer_mono_data_12pixel(dat1, dat2); //先显示 12 个点阵
```

```
transfer_mono_data_3pixel(dat2<<4); //传送 dat2 的 D3\D2\D1 这 3 位, 对应第 13、14、15 个点阵会显示出来; 列地址是自动+1 的
```

地址是自动+1 的

```
}
```

```
//显示 16 个点阵
```

```
void transfer_mono_data_16pixel(uchar dat1, uchar dat2)
```

```
{
```

```
transfer_mono_data_15pixel(dat1, dat2); //先显示 15 个点阵
```

```
transfer_mono_data_3pixel(dat2<<7); //显示第 16 个点阵, 对应 dat2 的 D0 位。
```

//这个液晶驱动 IC 的每个列地址管 3 个点阵, 无法分开, 所以第 16 个点阵会连累到第 17、18 个点阵, 结果是每次显示 18 个点阵, 只不过第 17、18 个点阵会补 “0”

```
//如果第 17、18 个点阵本来有显示内容, 就会被无情地清掉
```

```
}
```

```
//显示 18 个点阵
```

```
void transfer_mono_data_18pixel(uchar dat1, uchar dat2, uchar dat3)
```

```
{
```

```
transfer_mono_data_15pixel(dat1, dat2); //先显示 15 个点阵
```

```
transfer_mono_data_3pixel((dat2<<7)|(dat3>>1)); //传送 dat2 的 D0 和 dat3 的 D7、D6 这 3 位，对应第 16、17、18
个点阵会显示出来；列地址是自动+1 的
}
```

//显示 21 个点阵

```
void transfer_mono_data_21pixel(uchar dat1, uchar dat2, uchar dat3)
{
    transfer_mono_data_18pixel(dat1, dat2, dat3); //先显示 18 个点阵
    transfer_mono_data_3pixel(dat3<<2); //传送 dat3 的 D5、D4、D3 这 3 位，对应第 19、20、21 个点阵会显示出来；
列地址是自动+1 的
}
```

//显示 24 个点阵。方法一：

```
void transfer_mono_data_24pixel(uchar dat1, uchar dat2, uchar dat3)
{
    transfer_mono_data_21pixel(dat1, dat2, dat3); //先显示 21 个点阵
    transfer_mono_data_3pixel(dat3<<5); //传送 dat3 的 D2、D1、D0 这 3 位，对应第 22、23、24 个点阵会显示出来；
列地址是自动+1 的
}
```

//显示 27 个点阵

```
void transfer_mono_data_27pixel(uchar dat1, uchar dat2, uchar dat3, uchar dat4)
{
    transfer_mono_data_24pixel(dat1, dat2, dat3); //先显示 24 个点阵
    transfer_mono_data_3pixel(dat4); //传送 dat4 的 D7、D6、D5 这 3 位，对应第 25、26、27 个点阵会显示出来；
列地址是自动+1 的
}
```

//显示 30 个点阵

```
void transfer_mono_data_30pixel(uchar dat1, uchar dat2, uchar dat3, uchar dat4)
{
    transfer_mono_data_24pixel(dat1, dat2, dat3); //先显示 24 个点阵
    transfer_mono_data_6pixel(dat4); //再显示 6 个点阵，24+6=30
}
```

//显示 32 个点阵

```
void transfer_mono_data_32pixel(uchar dat1, uchar dat2, uchar dat3, uchar dat4)
{
    transfer_mono_data_24pixel(dat1, dat2, dat3); //先显示 24 个点阵
    transfer_mono_data_8pixel(dat4); //再显示 8 个点阵，24+8=32
    //这个液晶驱动 IC 的每个列地址管 3 个点阵，无法分开，所以第 31、32 个点阵会连累到第 33 个点阵，结果是每次显示 33
个点阵，只不过第 33 个点阵会补“0”
    //如果第 33 个点阵本来有显示内容，就会被无情地清掉
}
```

//显示 33 个点阵

```
void transfer_mono_data_33pixel(uchar dat1, uchar dat2, uchar dat3, uchar dat4, uchar dat5)
{
```

```

transfer_mono_data_24pixel(dat1, dat2, dat3);          //先显示 24 个点阵
transfer_mono_data_9pixel(dat4, dat5);              //再显示 9 个点阵
}

//显示 48 个点阵
void transfer_mono_data_48pixel(uchar dat1, uchar dat2, uchar dat3, uchar dat4, uchar dat5, uchar dat6)
{
    transfer_mono_data_24pixel(dat1, dat2, dat3);          //先显示 24 个点阵
    transfer_mono_data_24pixel(dat4, dat5, dat6);          //再显示 24 个点阵
}

//传送同一个地址的 3 个点阵的 4 灰度级的数据: 比如 SEG0、SEG1、SEG2, 这 3 个点阵是同一个列地址, 无法分开
//送灰度数据(gray_data)时, SEG0 对应高 3 位 (D7、D6、D5), SEG1 对应中 3 位 (D4、D3、D2), SEG2 对应低两位 (D1、D0)。
void transfer_gray_data_3pixel(uchar dat1)
{
    uchar gray_data;
    gray_data=dat1&0xc0;;    //给 gray_data 的 D7、D6 赋值(=dat1 的 D7、D6)
    if((dat1&0xc0)==0xc0)
    {
        gray_data|=0x20;    //给 gray_data 的 D5 赋值, 当 dat1 的 D7、D6 都是 1 的时候, gray_data 的 D5=1, 当 dat1 的 D7\6
        不都是 1 的时候, gray_data 的 D5=0
    }
    gray_data|=((dat1>>1)&0x18);    //给 gray_data 的 D4、D3 赋值 (=dat1 的 D5、D4)
    if((dat1&0x30)==0x30)
    {
        gray_data|=0x04;    //给 gray_data 的 D2 赋值, 当 dat1 的 D5、D4 都是 1 的时候, gray_data 的 D2=1, 当 dat1 的 D7、
        D6 不都是 1 的时候, gray_data 的 D2=0
    }
    gray_data|=((dat1>>2)&0x03);    //给 gray_data 的 D1、D0 赋值(=dat1 的 D3、D2)
    transfer_data_lcd(gray_data);    //传送 1 个字节灰度数据给液晶驱动 IC, 对应的 3 个点阵会显示
    (seg_N, seg_N+1, seg_N+2)
}

//传送同一个地址的 12 个点阵的 4 灰度的数据: 比如 SEG0、SEG1、SEG2.....SEG9、SEG10、SEG11 (这 12 个点阵是 4 个列地址)
//每 2 位数据对应一个点阵, 12 个点阵用: 2*12=24 位, 即 3 个字节: dat1、dat2、dat3
void transfer_gray_data_12pixel(uchar dat1, uchar dat2, uchar dat3)
{
    transfer_gray_data_3pixel(dat1);                //显示 3 个点阵(seg_N, seg_N+1, SEG_N+2)
    transfer_gray_data_3pixel((dat1<<6)|(dat2>>2));    //显示 3 个点阵(seg_N+3, seg_N+4, SEG_N+5)
    transfer_gray_data_3pixel((dat2<<4)|(dat3>>4));    //显示 3 个点阵(seg_N+6, seg_N+7, SEG_N+8)
    transfer_gray_data_3pixel(dat3<<2);                //显示 3 个点阵(seg_N+9, seg_N+10, SEG_N+11)
}

/*清屏*/
void clear_screen()
{

```



```

int i, j;
lcd_address(0, 0, 384, 160);
transfer_command_lcd(0x2c);
for(i=0;i<160;i++)
{
    for(j=0;j<24;j++)
    {
        transfer_mono_data_18pixel(0x00, 0x00, 0x00); //每个字节显示 8 个点阵，显示 8*3=24 个点阵
    }
}
}

```

/\*显示 8\*16 点阵 ASCII 码字符或等同于 8\*16 点阵的图像\*/

```
void disp_8x16(int x, int y, uchar *dp)
```

```

{
    int i, j;
    uchar dat1;
    lcd_address(x, y, 8, 16);
    transfer_command_lcd(0x2c);
    for(i=0;i<16;i++)
    {
        for(j=0;j<16;j++)
        {
            dat1=*dp;dp++;
            transfer_mono_data_8pixel(dat1);
        }
    }
}

```

//括号里的参数分别为（列，行，数据指针）

```
void display_string_8x16(int x, int y, uchar *text)
```

```

{
    uint i=0, j, n, dat1;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x, y, 8, 16);
            transfer_command_lcd(0x2c);
            for(n=0;n<16;n++)
            {
                dat1=ascii_table_8x16[j][n];
                transfer_mono_data_8pixel(dat1);
            }
            i++;
        }
    }
}

```





```

        x+=3;
    }
    else
    i++;
}
}

```

//括号里的参数分别为(列, 行, 数据指针)

```
void display_string_12x24(int x,int y,uchar *text)
```

```

{
    uint i=0, j, n, dat1, dat2;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x, y, 12, 24);
            transfer_command_lcd(0x2c);
            for(n=0;n<24;n++)
            {
                dat1=ascii_table_12x24[j][2*n];
                dat2=ascii_table_12x24[j][2*n+1];
                transfer_mono_data_12pixel(dat1, dat2);
            }
            i++;
            x+=4;
        }
        else
            i++;
    }
}

```



//显示 12\*12 点阵的图像

```
void disp_12x12(int x,int y,uchar *dp)
```

```

{
    int i, j;
    uchar dat1, dat2;
    lcd_address(x, y, 12, 12);
    transfer_command_lcd(0x2C);
    for(i=0;i<12;i++)
    {
        for(j=0;j<1;j++)//循环 1 次, 每次显示 12 个点阵
        {
            dat1=*dp;dp++;
            dat2=*dp;dp++;
            transfer_mono_data_12pixel(dat1, dat2); //每个字节显示 8 个点阵, 显示 8*2=16 个点阵
        }
    }
}

```

```

    }
}

//显示 16*16 点阵的图像
void disp_16x16(int x, int y, uchar *dp)
{
    int i, j;
    uchar dat1, dat2;
    lcd_address(x, y, 16, 16);
    transfer_command_lcd(0x2C);
    for(i=0; i<16; i++)
    {
        for(j=0; j<16; j++)//循环 1 次, 每次显示 18 个点阵
        {
            dat1=*dp; dp++;
            dat2=*dp; dp++;
            transfer_mono_data_16pixel(dat1, dat2); //每个字节显示 8 个点阵, 显示 8*2=16 个点阵
        }
    }
}

//显示 18*18 点阵的图像
void disp_18x18(int x, int y, uchar *dp)
{
    int i, j;
    uchar dat1, dat2, dat3;
    lcd_address(x, y, 18, 18);
    transfer_command_lcd(0x2C);
    for(i=0; i<18; i++)
    {
        for(j=0; j<18; j++)//循环 1 次, 每次显示 18 个点阵
        {
            dat1=*dp; dp++;
            dat2=*dp; dp++;
            dat3=*dp; dp++;
            transfer_mono_data_18pixel(dat1, dat2, dat3); //每个字节显示 8 个点阵, 显示 8*2=16 个点阵
        }
    }
}

//显示 21*21 点阵的图像
void disp_21x21(int x, int y, uchar *dp)
{
    int i, j;
    uchar dat1, dat2, dat3;

```



```

lcd_address(x, y, 21, 21);
transfer_command_lcd(0x2C);
for(i=0;i<21;i++)
{
    for(j=0;j<1;j++)//循环 1 次, 每次显示 18 个点阵
    {
        dat1=*dp;dp++;
        dat2=*dp;dp++;
        dat3=*dp;dp++;
        transfer_mono_data_21pixel(dat1, dat2, dat3);    //每个字节显示 8 个点阵, 显示 8*2=16 个点阵
    }
}
}

```

//显示 24\*24 点阵的图像

```

void disp_24x24(int x, int y, uchar *dp)
{

```

```

    int i, j;
    uchar dat1, dat2, dat3;
    lcd_address(x, y, 24, 24);
    transfer_command_lcd(0x2C);
    for(i=0;i<24;i++)
    {
        for(j=0;j<1;j++)//循环 1 次, 每次显示 24 个点阵
        {
            dat1=*dp;dp++;
            dat2=*dp;dp++;
            dat3=*dp;dp++;
            transfer_mono_data_24pixel(dat1, dat2, dat3);    //每个字节显示 8 个点阵, 显示 8*3=24 个点阵
        }
    }
}

```

//显示 27\*27 点阵的图像

```

void disp_27x27(int x, int y, uchar *dp)
{

```

```

    int i, j;
    uchar dat1, dat2, dat3, dat4;
    lcd_address(x, y, 27, 27);
    transfer_command_lcd(0x2C);
    for(i=0;i<27;i++)
    {
        for(j=0;j<1;j++)//循环 1 次, 每次显示 24 个点阵
        {
            dat1=*dp;dp++;
            dat2=*dp;dp++;

```



```

        dat3=*dp;dp++;
        dat4=*dp;dp++;
        transfer_mono_data_27pixel(dat1, dat2, dat3, dat4); //每个字节显示 8 个点阵, 显示 8*3=24 个点阵
    }
}
}

```

//显示 30\*30 点阵的图像

```

void disp_30x30(int x, int y, uchar *dp)
{

```

```

    int i, j;
    uchar dat1, dat2, dat3, dat4;
    lcd_address(x, y, 30, 30);
    transfer_command_lcd(0x2C);
    for(i=0; i<30; i++)

```

```

    {
        for(j=0; j<1; j++)//循环 1 次, 每次显示 30 个点阵

```

```

        {
            dat1=*dp;dp++;
            dat2=*dp;dp++;
            dat3=*dp;dp++;
            dat4=*dp;dp++;
            transfer_mono_data_30pixel(dat1, dat2, dat3, dat4); //每个字节显示 8 个点阵, 显示 8*3=24 个点阵
        }
    }
}

```

//显示 32\*32 点阵的图像

```

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

```

```

    int i, j;
    uchar dat1, dat2, dat3, dat4;
    lcd_address(x, y, 32, 32);
    transfer_command_lcd(0x2C);
    for(i=0; i<32; i++)

```

```

    {
        for(j=0; j<1; j++)//循环 1 次, 每次显示 32 个点阵

```

```

        {
            dat1=*dp;dp++;
            dat2=*dp;dp++;
            dat3=*dp;dp++;
            dat4=*dp;dp++;
            transfer_mono_data_32pixel(dat1, dat2, dat3, dat4); //每个字节显示 8 个点阵, 显示 8*4=32 个点阵
        }
    }
}

```



//显示 33\*33 点阵的图像

```
void disp_33x33(int x, int y, uchar *dp)
```

```
{
```

```
    int i, j;
```

```
    uchar dat1, dat2, dat3, dat4, dat5;
```

```
    lcd_address(x, y, 33, 33);
```

```
    transfer_command_lcd(0x2C);
```

```
    for(i=0; i<33; i++)
```

```
    {
```

```
        for(j=0; j<1; j++)//循环 1 次, 每次显示 24 个点阵
```

```
        {
```

```
            dat1=*dp; dp++;
```

```
            dat2=*dp; dp++;
```

```
            dat3=*dp; dp++;
```

```
            dat4=*dp; dp++;
```

```
            dat5=*dp; dp++;
```

```
            transfer_mono_data_33pixel(dat1, dat2, dat3, dat4, dat5); //每个字节显示 8 个点阵, 显示 8*3=24 个点阵
```

```
        }
```

```
    }
```

```
}
```

//显示 48\*48 点阵的图像

```
void disp_48x48(int x, int y, uchar *dp)
```

```
{
```

```
    int i, j;
```

```
    uchar dat1, dat2, dat3, dat4, dat5, dat6;
```

```
    lcd_address(x, y, 48, 48);
```

```
    transfer_command_lcd(0x2C);
```

```
    for(i=0; i<48; i++)
```

```
    {
```

```
        for(j=0; j<1; j++)//循环 1 次, 每次显示 24 个点阵
```

```
        {
```

```
            dat1=*dp; dp++;
```

```
            dat2=*dp; dp++;
```

```
            dat3=*dp; dp++;
```

```
            dat4=*dp; dp++;
```

```
            dat5=*dp; dp++;
```

```
            dat6=*dp; dp++;
```

```
            transfer_mono_data_48pixel(dat1, dat2, dat3, dat4, dat5, dat6); //每个字节显示 8 个点阵, 显示 8*3=24 个点阵
```

```
        }
```

```
    }
```

```
}
```

//显示 320\*160 点阵的图像



```

void disp_320x160(uchar *dp)
{
    int i, j;
    uchar dat1, dat2, dat3, dat4;
    lcd_address(0, 0, 320, 160);
    transfer_command_lcd(0x2C);
    for(i=0; i<160; i++)
    {
        for(j=0; j<13; j++)//循环 13 次，每次显示 24 个点阵，合计 312 个点阵
        {
            dat1=*dp; dp++;
            dat2=*dp; dp++;
            dat3=*dp; dp++;
            transfer_mono_data_24pixel(dat1, dat2, dat3); //每个字节显示 8 个点阵，显示 8*3=24 个点阵
        }
        dat4=*dp; dp++;
        transfer_mono_data_8pixel(dat4); //每个字节显示 8 个点阵，312+8=320 个点阵
    }
}
    
```

//显示 320\*160 点阵的 4 灰度级图像  
 void disp\_4gray\_320x160(uchar \*dp)

```

{
    uchar i, j;
    uchar dat1, dat2, dat3;
    lcd_address(0, 0, 320, 160); //
    transfer_command_lcd(0x2C);
    for(i=0; i<160; i++)
    {
        for(j=0; j<26; j++)//循环 26 次，每次显示 12 个点阵，合计 26*12=312 个点阵
        {
            dat1=*dp; dp++;
            dat2=*dp; dp++;
            dat3=*dp; dp++;
            transfer_gray_data_12pixel(dat1, dat2, dat3); //每个字节显示 4 个点阵，共显示 4*3=12 个点阵
        }
        dat1=*dp; dp++;
        dat2=*dp; dp++;
        transfer_gray_data_3pixel(dat1); //显示 3 个点阵
        transfer_gray_data_3pixel((dat1<<6)&(dat2>>4)); //显示 3 个点阵
        transfer_gray_data_3pixel(dat2<<4); //显示 3 个点阵
    }
}
    
```

//-----

void main ()



```

{
P1M1=0x00;
    P1M0=0x00; //配置为准双向
    P2M1=0x00;
    P2M0=0x00; //配置为准双向
    P3M1=0x00;
    P3M0=0x00; //配置为准双向
    while(1)
    {
        initial_lcd();
        clear_screen();//清屏
        disp_320x160 bmp1); //显示一个 320x160 点阵的图片
        waitkey();
        clear_screen();//清屏
        disp_4gray_320x160 bmp_4gray_2); //显示一个 320x160 点阵的 4 灰度级的图片
        waitkey();
        clear_screen();//清屏
        disp_24x24(0, 0, jing_24); //在 (0, 0) 位置显示一个 24x24 点阵的汉字或图片, 三个参数分别是 (x, y, 24x24 点阵的指针)
        disp_24x24(7, 0, lian_24); //在 (7, 0) 位置显示一个 24x24 点阵的汉字或图片, 三个参数分别是 (x, y, 24x24 点阵的指针)
        disp_24x24(14, 0, xun_24); //在 (14, 0) 位置显示一个 24x24 点阵的汉字或图片, 三个参数分别是 (x, y, 24x24 点阵的指针)
        disp_16x16(40, 0, jing_16); //在 (40, 0) 位置显示一个 16x16 点阵的汉字或图片, 三个参数分别是 (x, y, 16x16 点阵的指针)
        disp_16x16(45, 0, lian_16); //在 (45, 0) 位置显示一个 16x16 点阵的汉字或图片, 三个参数分别是 (x, y, 16x16 点阵的指针)
        disp_16x16(50, 0, xun_16); //在 (50, 0) 位置显示一个 16x16 点阵的汉字或图片, 三个参数分别是 (x, y, 16x16 点阵的指针)
        disp_32x32(60, 0, jing_32); //在 (60, 0) 位置显示一个 32x32 点阵的汉字或图片, 三个参数分别是 (x, y, 32x32 点阵的指针)
        disp_32x32(70, 0, lian_32); //在 (70, 0) 位置显示一个 32x32 点阵的汉字或图片, 三个参数分别是 (x, y, 32x32 点阵的指针)
        disp_32x32(80, 0, xun_32); //在 (80, 0) 位置显示一个 32x32 点阵的汉字或图片, 三个参数分别是 (x, y, 32x32 点阵的指针)
        disp_12x12(92, 0, jing_12); //在 (92, 0) 位置显示一个 12x12 点阵的汉字或图片, 三个参数分别是 (x, y, 12x12 点阵的指针)
        disp_12x12(96, 0, lian_12); //在 (96, 0) 位置显示一个 12x12 点阵的汉字或图片, 三个参数分别是 (x, y, 12x12 点阵的指针)
        disp_12x12(100, 0, xun_12); //在 (100, 0) 位置显示一个 12x12 点阵的汉字或图片, 三个参数分别是 (x, y, 12x12 点阵的指针)
        disp_18x18(8, 32, jing_18);
        disp_21x21(15, 32, jing_21);
        disp_27x27(22, 32, jing_27);
        disp_30x30(30, 32, jing_30);
        disp_33x33(40, 32, jing_33); //

```

```

disp_48x48(52, 32, jing_48);
disp_8x16(0, 32, A_1);
display_string_8x16(0, 80, "ABCDEFGH!@#$$%^&123");
display_string_12x24(0, 96, "ABCDEFGH!@#$$%^&123");
waitkey();
}
}
    
```

## 7.5 串行程序举例:

### 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 8. 串行接口

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

#### 串行接口程序

```

sbit lcd_cs1 = P1^1; //CS
sbit lcd_reset = P1^0; //RST
sbit lcd_sclk = P3^2; //串行时钟
sbit lcd_rs = P3^0; //RS
sbit lcd_sid = P3^1; //串行数据
sbit key = P2^0; //按键
//写指令到 LCD 模块
void transfer_command_lcd(int data1)
{
    char i;
    lcd_cs1=0;
    lcd_rs=0;
    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;  
}  
  
//写数据到LCD模块  
void transfer_data_lcd(int data1)  
{  
    char i;  
    lcd_cs1=0;  
    lcd_rs=1;  
    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;  
}
```

**-END-**

