JLX384160G-973-PN 使用说明书

目 录

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 JLX384160G-973-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。

JLX384160G-973-PN 可以显示 384 列*160 行点阵单色或 4 灰度级的图片，或显示 12 个/行*5 行 32*32 点阵或显示 16 个/行*6 行 24*24 点阵的汉字，或显示 24 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 48 个*10 行，或显示 5*8 点阵的英文、数字、字符 64 个*20 行。

2. JLX384160G-973-PN 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 ST7586S,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，或选择并行接口。
2.5 工作温度宽:-20℃ - 70℃;
2.6 储存温度宽:-30℃ - 80℃;
2.7 显示内容:

- 可 384*160 点阵单色或 4 灰度级图片;
- 可显示 12 个×5 行 32*32 点阵的汉字;
- 可显示 16 个×6 行 24*24 点阵的汉字;
- 可显示 24 个×10 行 16*16 点阵的汉字;
- 可显示 32 个×13 行 12*12 点阵的汉字;
- 可显示 48 个*10 行 8*16 点阵的英文、数字、字符;
- 可显示 64 个*20 行 5*8 点阵的英文、数字、字符;
- 或显示其他的 ASCII 码等;

3. 外形尺寸及接口引脚功能：

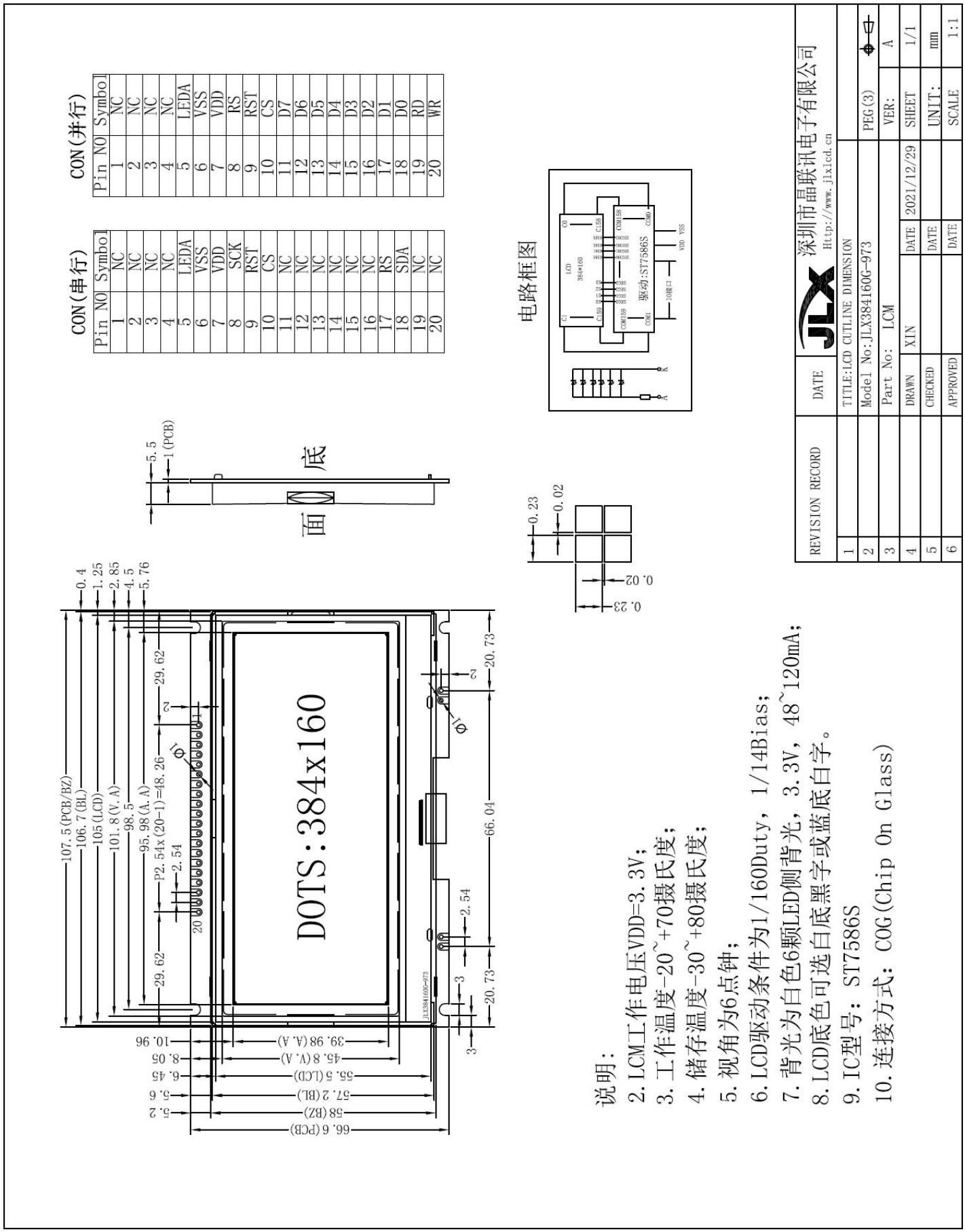


图 1. 液晶模块外形尺寸

模块的接口可选择**并行或串口接口**:

◆当并行时，CON1 功能如下：

引 线 号	符 号	名 称	功 能
1	NC	空脚	当选择带字库 IC 时才使用
2	NC	空脚	当选择带字库 IC 时才使用
3	NC	空脚	当选择带字库 IC 时才使用
4	NC	空脚	当选择带字库 IC 时才使用
5	LEDA	背光电源	背光电源正极，同 VDD 电压（5V 或 3.3V）
6	VSS	接地	0V
7	VDD	电路电源	5V 或3.3V
8	A0	寄存器选择信号	H:数据寄存器 0:指令寄存器
9	RESET	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 DB7-DB0
12	D6		
13	D5		
14	D4		
15	D3		
16	D2		
17	D1		
18	D0		
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

◆当 4 线 SPI 串行时，接口功能如下：

引 线 号	符 号	名 称	功 能
1	NC	空脚	当选择带字库 IC 时才使用
2	NC	空脚	当选择带字库 IC 时才使用
3	NC	空脚	当选择带字库 IC 时才使用
4	NC	空脚	当选择带字库 IC 时才使用
5	LEDA	背光电源	背光电源正极，同 VDD 电压（5V 或 3.3V）
6	VSS	接地	0V
7	VDD	电路电源	5V 或3.3V
8	SCL(A0)	串行时钟	串行时钟
9	RESET	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	片选信号	低电平片选
11	NC(D7)	空脚	
12	NC(D6)	空脚	
13	NC(D5)	空脚	
14	NC(D4)	空脚	
15	NC(D3)	空脚	

16	NC(D2)	空脚	
17	A0(D1)	寄存器选择	H:数据寄存器 0:指令寄存器
18	SDA(D0)	串行数据	串行数据
19	NC	空脚	
20	NC	空脚	

4. 电路框图

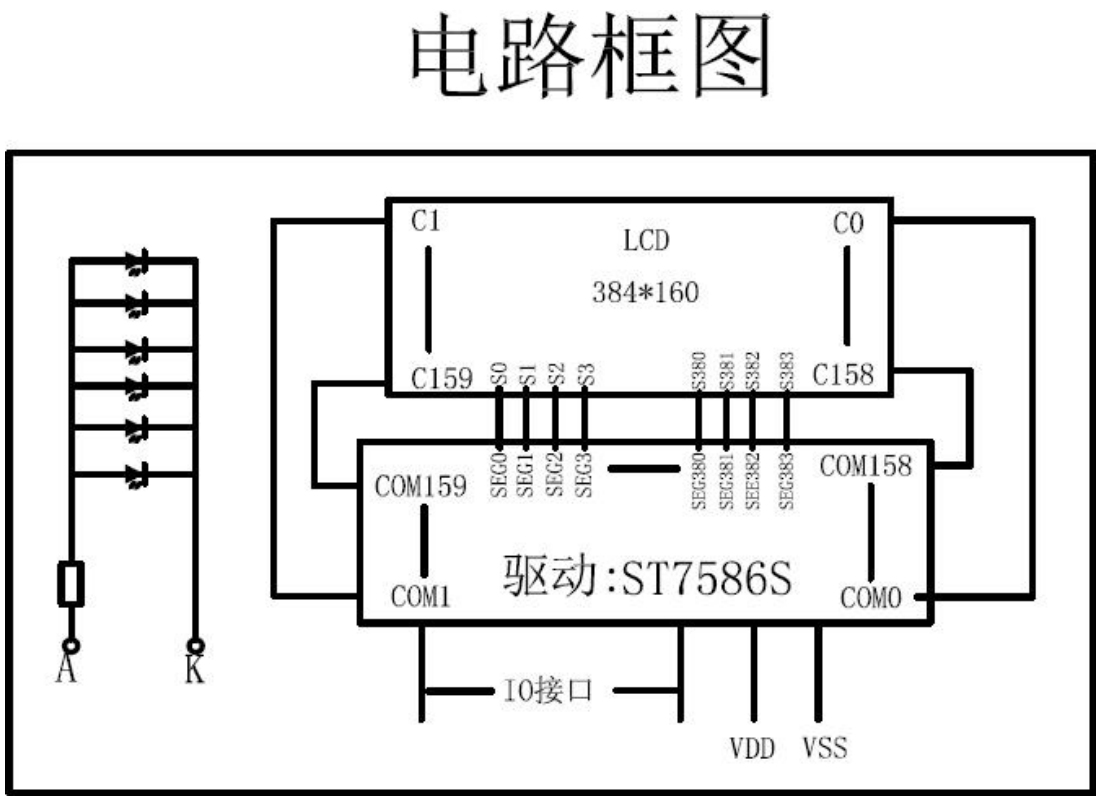


图 2:JLX384160G-973-PN 图像点阵型液晶模块的电路框图

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° C∽+70° C;
背光颜色：白色。
正常工作电流为：(8∽15)×6=48~120mA（LED 灯数共 6 颗）；
工作电压：3.3V 或 5.0V，同 VDD 电压;

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	-0.3	3.3	3.6	V
电路电源	VD1	-0.3	3.3	3.6	V
LCD 驱动电压	V0-XV0	-0.3	15.6	19	V
LCD 驱动电压	VG	-0.3		5.5	V
LCD 驱动电压	VM	-0.3		VDD+0.3	V
工作温度		-20		+70	℃
储存温度		-30		+80	℃

表 2：最大极限参数

5.2 直流（DC）参数

名 称	符 号	测 试 条 件	标 准 值			单位
			MIN	TYPE	MAX	
工作电压	VDD		2.7	3.3	3.4	V
输入高电平	V_{IHC}	-	0.7xVDD	-	VDD	V
输入低电平	V_{ILC}	-	VSS	-	0.3xVDD	V
输出高电平	V_{OHC}	IOH = 0.2mA	0.8xVDD	-	VDD	V
输出低电平	V_{OHC}	IOO = 1.2mA	VSS	-	0.2xVDD	V
背光工作电压	VLED		2.8	3.0	3.1	V
模块工作电流	I_{DD}	VDD = 3.3V	-		0.3	mA
背光工作电流	I_{LED}	V_{LED}=3.0V	56	105	140	mA

表 3：直流（DC）参数

6. 读写时序特性

6.1 串行接口：

从 CPU 写到 ST7586S（Writing Data from CPU to ST7586S）

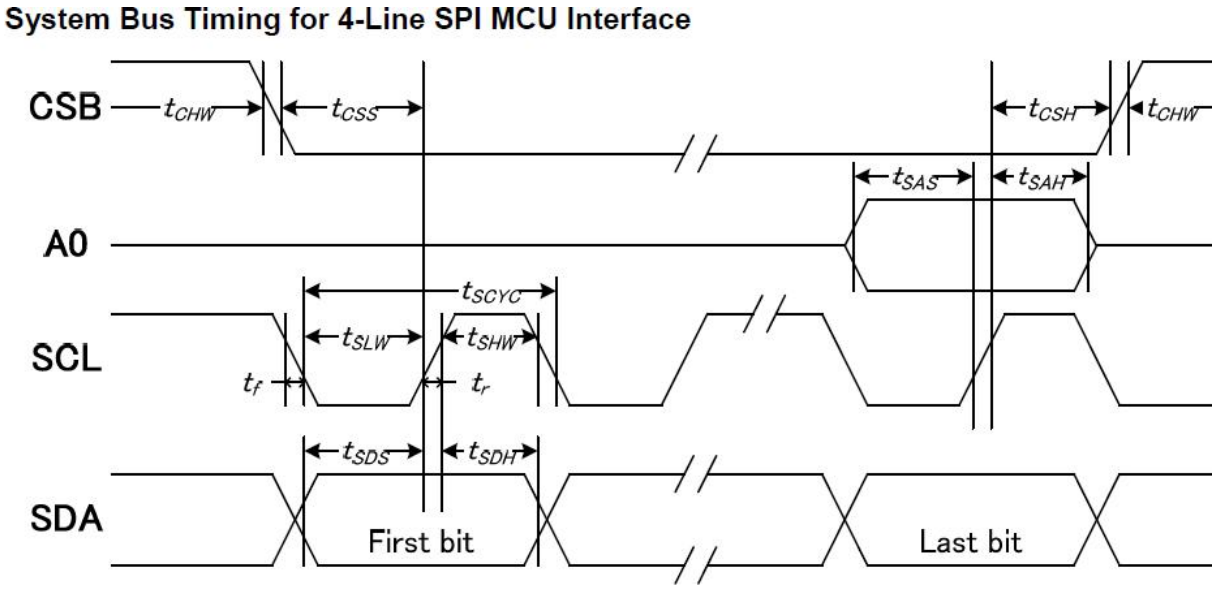


图 4. 从 CPU 写到 ST7586S（Writing Data from CPU to ST7586S）

6.2 串行接口：时序要求（AC 参数）：

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：SCK	100	--	--	ns
保持SCK高电平脉宽 （SCK “H” pulse width）	Tshw		45	--	--	
保持SCK低电平脉宽 （SCK “L” pulse width）	TSLW		45	--	--	
地址建立时间 (Address setup time)	TSAS	引脚：RS	20	--	--	
地址保持时间 (Address hold time)	Tsah		10	--	--	
数据建立时间 (Data setup time)	Tsds	引脚：SDA	20	--	--	
数据保持时间 (Data hold time)	TSDH		20	--	--	
片选信号建立时间 （CS-SCL time）	Tcss	引脚：CS	20	--	--	
片选信号保持时间 （CS-SCL time）	Tcsh		40	--	--	

VDD =3.3V, Ta = 25℃

6.3 并行接口：

从 CPU 写到 ST7586S（Writing Data from CPU to ST7586S）

System Bus Timing for 8080 MCU Interface

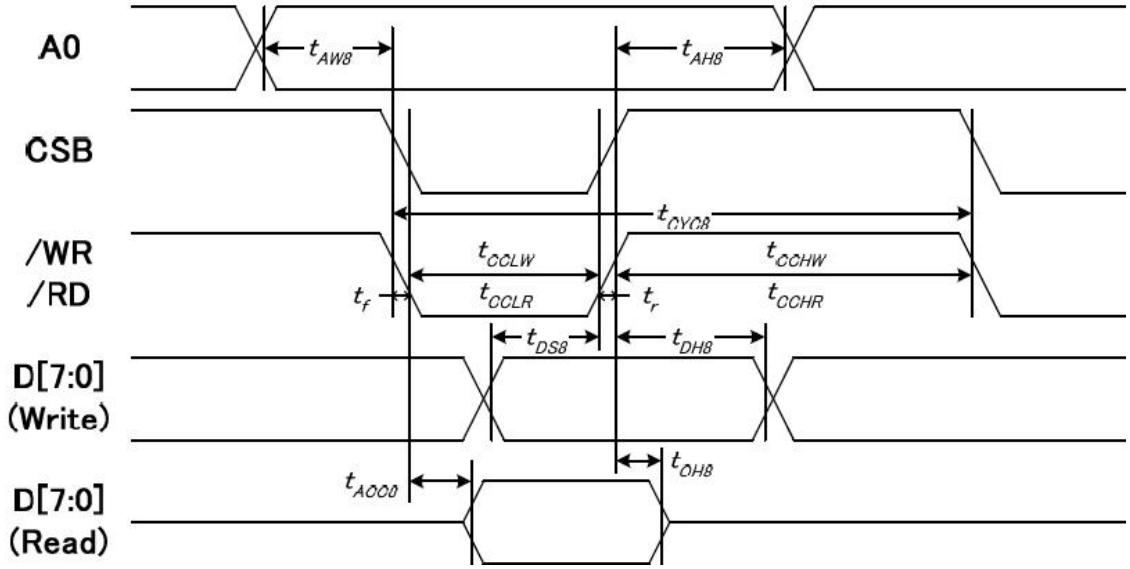


图 5. 从 CPU 写到 ST7586S（Writing Data from CPU to ST7586S）

System Bus Timing for 6800 MCU Interface

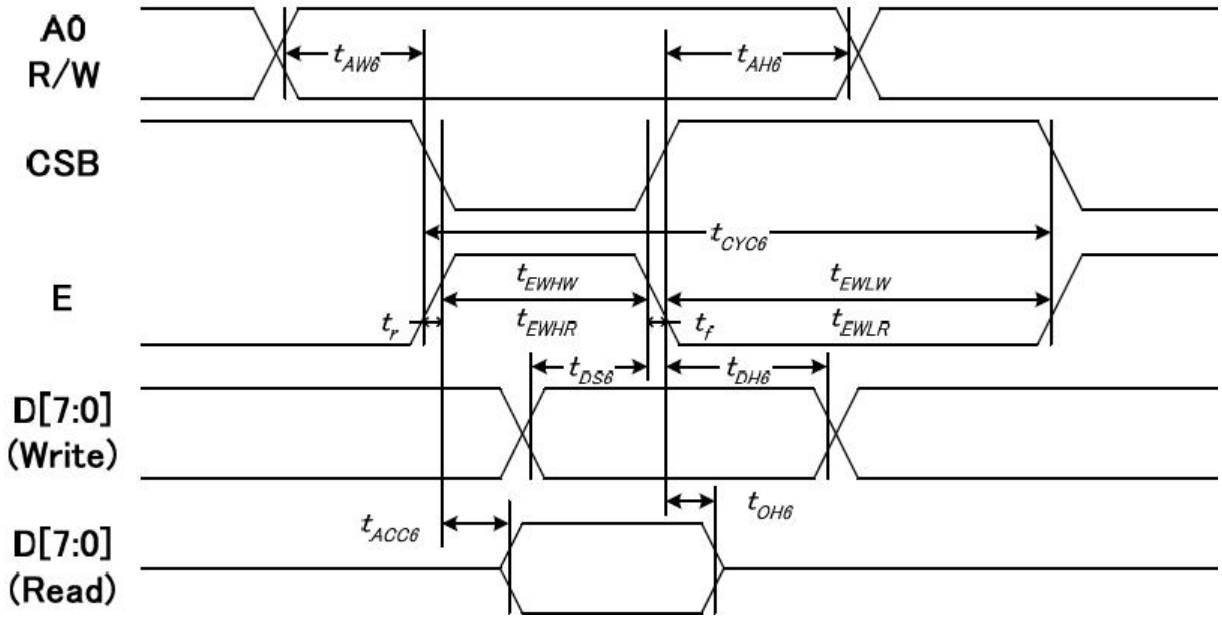


图 6. 从 CPU 写到 ST7586S（Writing Data from CPU to ST7586S）

6.4 并行接口：时序要求（AC 参数）：

写数据到 ST7586S 的时序要求：(8080 系列 MPU)

项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
地址建立时间	A0	tAW8	0	—	—	ns
地址保持时间		tAH8	0	—	—	
系统循环时间	WR	tCYC8	240	—	—	
使能“低”脉冲（写）		tCCLW	100	—	—	
使能“高”脉冲（写		tCCHW	100	—	—	
系统循环时间	RD	tCYC8	500			
使能“低”脉冲（读）		tCCLR	220	—	—	
使能“高”脉冲（读）		tCCHR	220	—		
写数据建立时间	D0-D7	tDS8	20		—	
写数据保持时间		tDH8	20		—	
读时间		tACC8	—		100	
读输出来允许时间		tOH8	10		110	

表 5

写数据到 ST7586S 的时序要求：（6800 系列 MPU）

项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
地址保持时间	A0	tAH6	0	--	--	ns
地址建立时间		tAW6	0		--	
系统循环时间（写）	WR	tCYC6	240		--	
使能“低”脉冲（写）		tEWLW	100	--	--	
使能“高”脉冲（写）		tEWHW	100	--	--	
系统循环时间（读）	RD	tCYC6	500	--	--	
使能“低”脉冲（读）		tEWLR	220	--	--	
使能“高”脉冲（读）		tEWHR	220	--	--	
写数据建立时间	D0-D7	tDS6	20		--	
写数据保持时间		tDH6	20		--	
读时间		tACC6	--		110	
读输出来允许时间		tOH6	10		110	

表 6

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

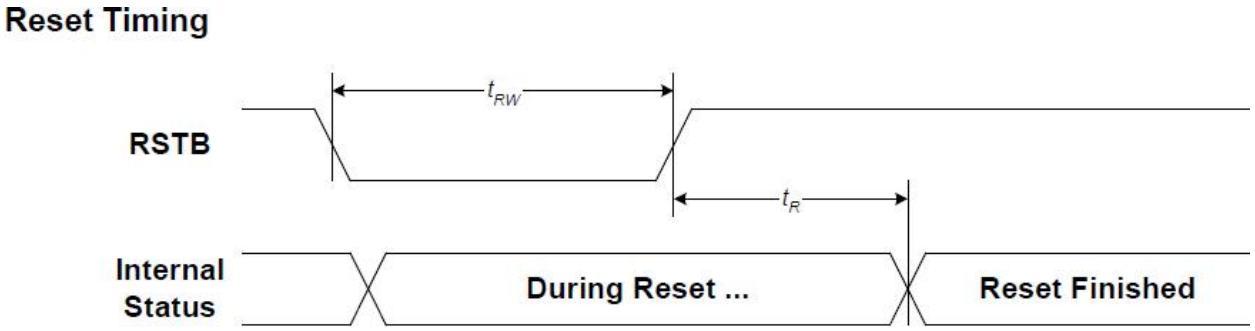


图 7：**电源启动后复位的时序**

项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
复位时间	tR		120	—	1.0	ms
复位保持低电平的时间	tRW	引脚：RESET	10	—	—	us

表 7：电源启动后复位的时序要求

7. 指令功能：

7.1 指令表

指 令 表 **表 8.**

指令名称	指 令 码										说 明
	RS	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	**0X01** :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
										1	**0x10**：进入省电模式
											0x11：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**
										1	**0x12:开，0x13:关**
(5) 显示正显/反显 (Display normal/reverse)	0	0	0	0	1	0	0	0	0	0	显示正显/反显:
										1	**0x20**：常规：正显
											0x21：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
										1	0x22：全部点阵关
											0x23：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：
										1	0x28：关， 0x29：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:**
	1	0	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	**起始列地址范围：0x00～0x9f**
	1	0	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	**结束列地址范围：0x00～0x9f**
	1	0	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	
	1	0	XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0	
(9) 页地址设置 (Set Row Address)	0	0	0	0	1	0	1	0	1	1	**页地址设置:**
	1	0	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	**起始页地址范围：0x00～0x9f**
	1	0	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	**结束页地址范围：0x00～0x9f**
	1	0	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	
	1	0	XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0	
(10)写数据到晶液屏 (Write Display Data)	0	0	0	0	1	0	1	1	0	0	**0X2C：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	**0X2E：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	**0X30：指定显示区域**
	1	0	PTS15	PTS14	PTS13	PTS12	PTS11	PTS10	PTS9	PTS8	**起始区域地址：00h≤PTS≤9Fh**
	1	0	PTS7	PTS6	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0	**结束区域地址：00h≤PTE≤9Fh**
	1	0	PTE15	PTE14	PTE13	PTE12	PTE11	PTE10	PTE9	PTE8	**显示区域：64h≤Duty≤160h**
	1	0	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	
(13) 指定显示滚动区域 (Scroll Area)	0	0	0	0	1	1	0	0	1	1	**0X33：滚动区域设置**
	1	0	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0	起始区域地址：TA=00h~A0h
	1	0	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	滚动区域: SA=00h~A0h
	1	0	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	结束区域地址：BA=00h~A0h
											TA+SA+BA=160
(14) 控制液晶屏显示	0	0	0	0	1	1	0	1	1	0	**0X36: 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	0	0	0	0	1	1	0	1	1	1	**0X37:滚动开始初始行设置**
(Start Line)	1	0	S7	S6	S5	S4	S3	S2	S1	S0	S=00h~9Fh
(16) 显示模式 (Display Mode)	0	0	0	0	1	1	1	0	0	0 1	**0XF0**：显示模式设置 **0X39**：黑白模式 **0X38**：4 灰级度模式
(17)数据显示接口	0	0	0	0	1	1	1	0	1	0	**0x3a:**使能数据显示模式
(Enable DDRAM	1	0	0	0	0	0	0	0	1	0	0x02:黑白和 4 灰级模式
Interface)										1	0x03:16 灰级模式
(18)显示点空比	0	0	1	0	1	1	0	0	0	0	0xb0:点空比范围：DT=03~9f
(Display Duty)	1	0	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	
(19)设置第一行输出	0	0	1	0	1	1	0	0	0	1	0xb1: 第一行输出设置范围：
(First Output COM)	1	0	FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0	FC=00~9f
(20) FOSC分频	0	0	1	0	1	1	0	0	1	1	0xb3:设置 FOSC 分频
(FOSC Divider)	1	0	0	0	0	0	0	0			
(21)指定区域显示	0	0	1	0	1	1	0	1	0	0	0xb4: 指定区域显示
(Partial Display)	1	0	1	0	1	0	0	0	0	0	
(22) N行反显	0	0	1	0	1	1	0	1	0	1	0xb5:N 行反显
(N-Line Inversion)	1	0	M	0	0	NL4	NL3	NL2	NL1	NL0	
(23) 读-改-写(Read	0	0	1	0	1	1	1	0	0	0	**读改写控制：**
Modify Write)										1	**0XB8**：启用读改写
											0XB9：禁用读改写
(24)液晶内部电压设置	0	0	1	1	0	0	0	0	0	0	**0xC0**：对比度设置
（Set Vop）	1	0	Vop7	Vop6	Vop5	Vop4	Vop3	Vop2	Vop1	Vop0	微调对比度范围：0x00~0xff
	1	0	-	-	-	-	-	-	-	Vop8	粗调对比度范围：0x01
(25)对比度VOP增加	0	0	1	1	0	0	0	0	0	1	0xc1:Vop 增加一级
(Vop Increase)											
(26) 对比度VOP降低	0	0	1	1	0	0	0	0	1	0	0xc2:Vop 降低一级
(Vop Decrease)											
(27) LCD 偏压比设置	0	0	1	1	0	0	0	0	1	1	0xc3:设置偏压比：
(BIAS System)	1	0	-	-	-	-	-	BS2	BS1	BS0	
(28) 升压倍数	0	0	1	1	0	0	0	1	0	0	0xc4: 内建升压倍数设置
(Booster Level)	1	0	-	-	-	-	-	BST2	BST1	BST0	
(29) 模拟电路	0	0	1	1	0	1	0	0	0	0	**0xd0:**模拟电路
(Analog Control)	1	0	0	0	0	1	1	1	0	1	0x1d:使能模拟电路
(30)自动读取控制	0	0	1	1	0	1	0	1	1	1	**0xd7:** 自动读取控制设置
(Auto Read Control)	1	0	1	0	0	0	1	1	1	1	0x8f:启用自动读取控制
						1					0x9f:禁用动读取控制
(31) 控制OTP读写	0	0	1	1	1	0	0	0	0	0	**0xe0：OTP 读写**

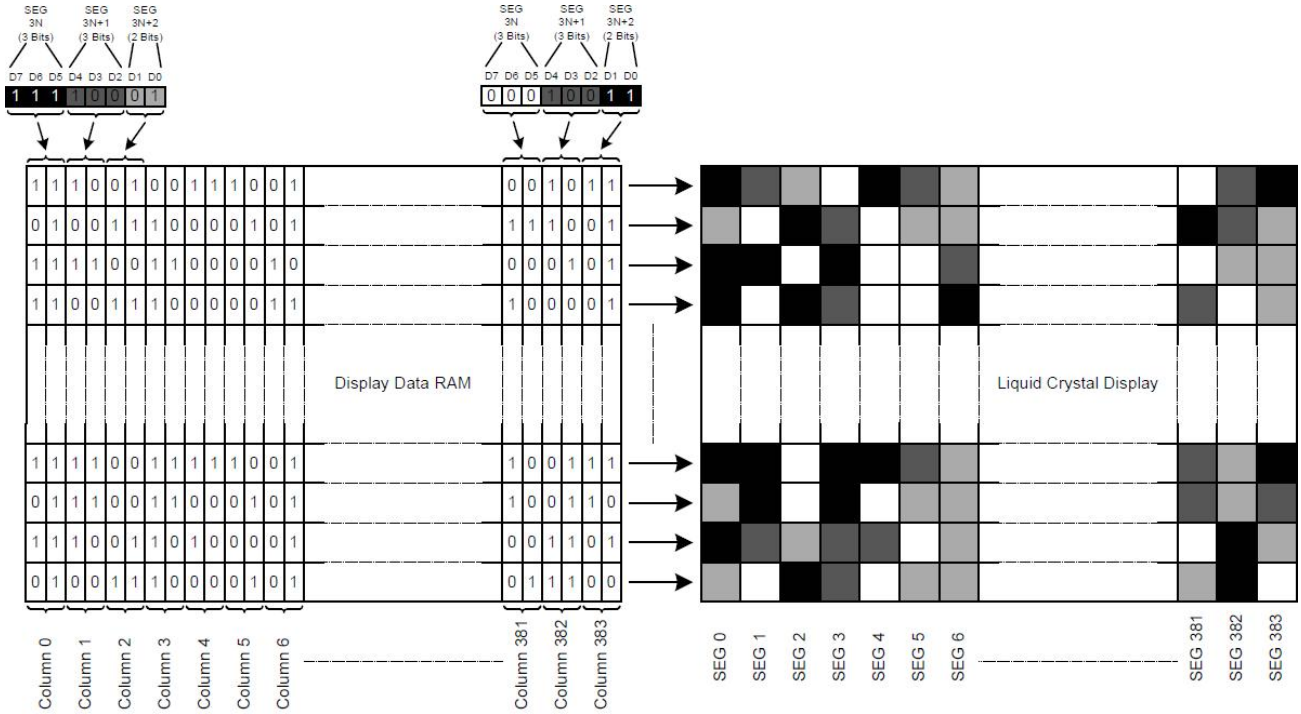
(OTP WR/RD Control)	1	0	0	0	WR RD	0	0	0	0	0	WR/RD=0；0x00,使能 OTP 读 ER/RD=1；0x20,使能 OTP 写
(32) 控制OTP输出 (OTP Control Out)	0	0	1	1	1	0	0	0	0	1	0xe1：OTP 输出
(33) 写OTP (OTP Write)	0	0	1	1	1	0	0	0	1	0	0xe2: 写 OTP 程序
(34) 读OTP (OTP Read)	0	0	1	1	1	0	0	0	1	1	0xe3: 读 OTP 程序
(35)OTP控制 (OTP Selection Control)	0	0	1	1	1	0	0	1	0	0	**0xe4**：OTP 控制设置
	1	0	0	Ctrl	0	1	1	0	0	1	0x19:禁用OTP 0x59:启用OTP
(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
	1	0	0	0	0	0	1	1	1	1	**0x0f**:启用OTP 编程
(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	
	1	0	–	–	–	FRD4	FRD3	FRD2	FRD1	FRD0	
(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 ST7586S 的 IC 资料.

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 384*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：

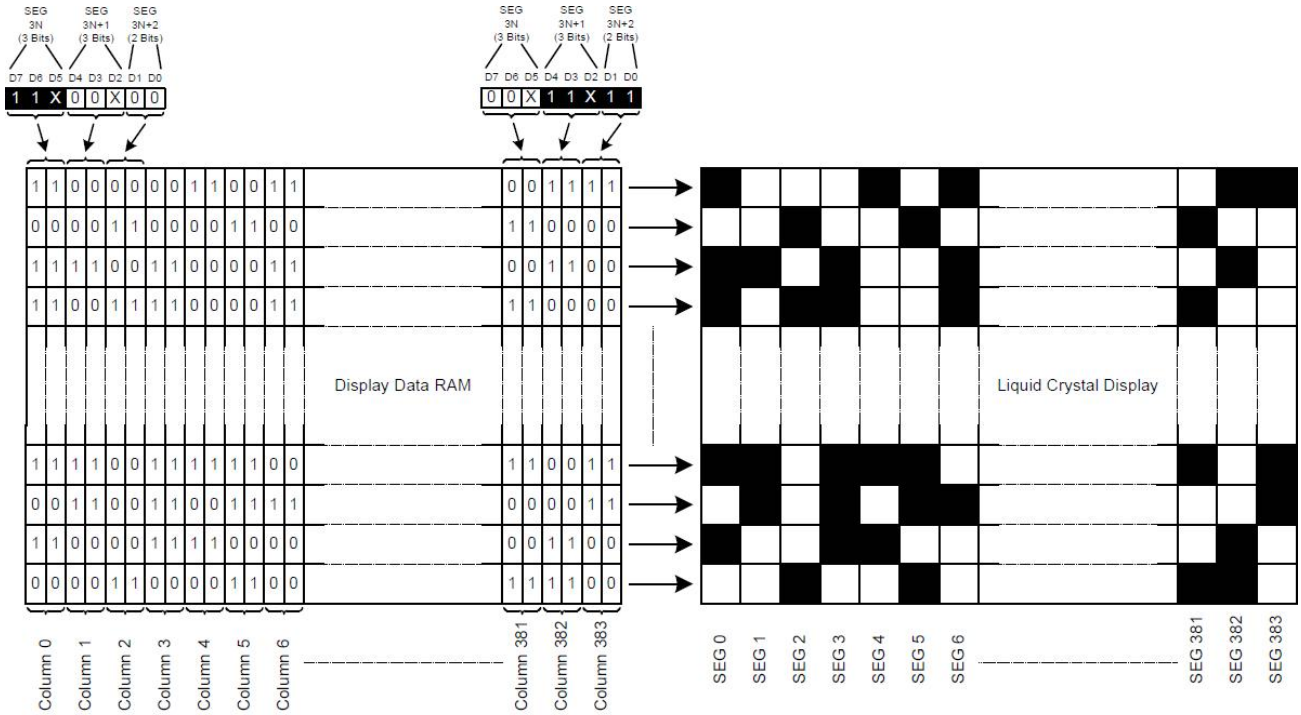


3 Bits Data			DDRAM		LCD
D7 (D4)	D6 (D3)	D5 (D2)			
1	1	1	1	1	
0	0	0	0	0	
1	0	0	1	0	
0	1	0	0	1	

Fix LSB to 0 if Gray Mode

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

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.

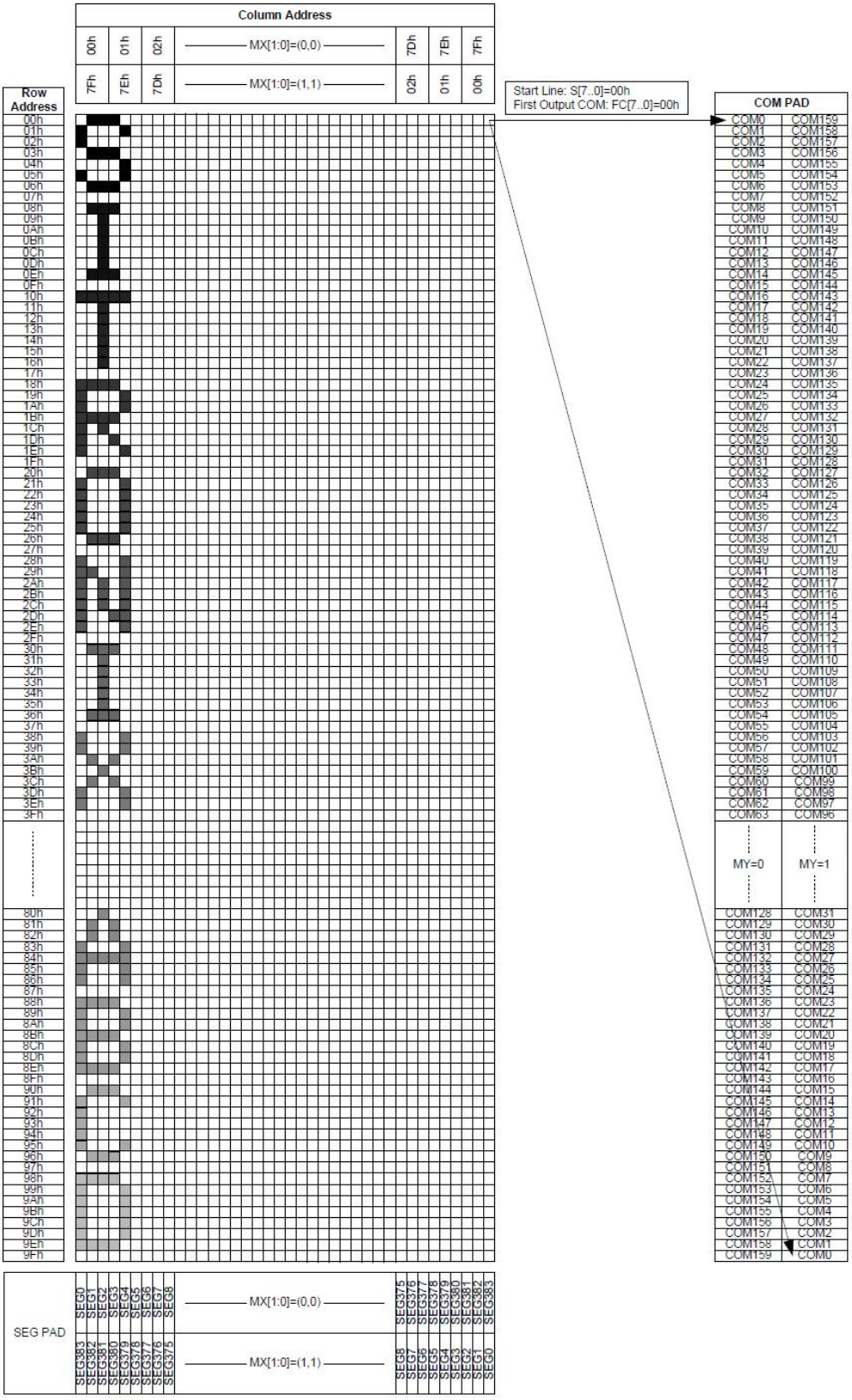
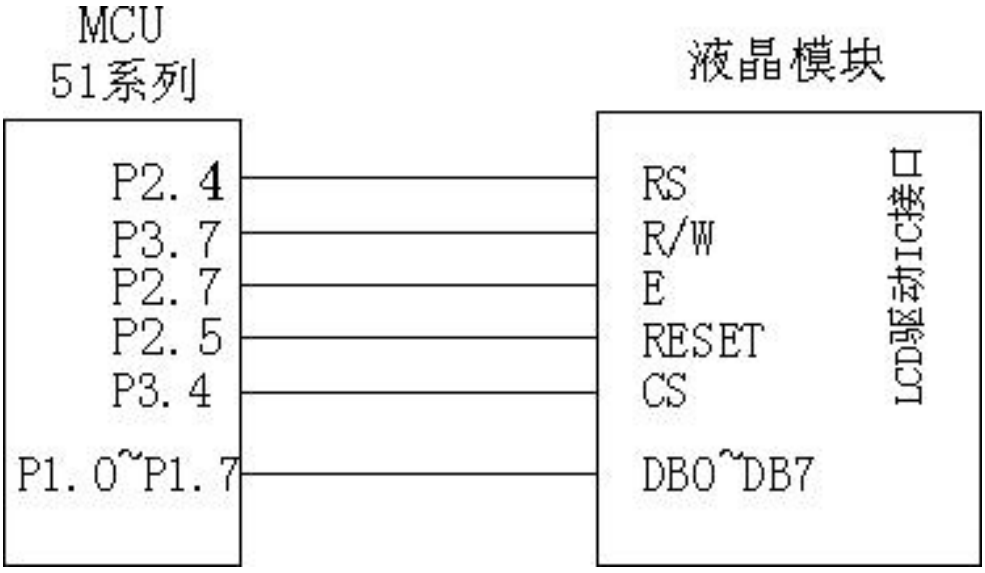


Fig. 6　DDRAM Display Direction

7.3 初始化方法

7.3.1 液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:



并行接口图

7.3.2 并行程序:

```
/*     液晶模块型号：JLX384160G-973-BN-P，
 并行接口,6800 时序，
       驱动IC 是:ST7586S(or compatible),
       版权所有：晶联讯电子：网址 http://www.jlxlcd.cn;
*/
#include <reg51.H>
#include <intrins.h>
#include <Ctype.h>

#include    <ASCII_TABLE_5X8_8X16_12x24_16x32_horizontal.h>
sbit lcd_cs1 =  P3^4;
sbit lcd_reset= P3^5;
sbit  lcd_rs  =  P3^3;
sbit  lcd_rw  =  P2^1;
sbit lcd_e    =  P3^0;
/*另外：DB0~DB7 与 P1.0~P1.7 相连*/

sbit key  =  P2^0;      //按键：我的主板上是P2.0 口与GND 之间接一个按键


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

uchar code bmp320160_1[];

//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);
}

//写指令到LCD 模块
void transfer_command_lcd(int data1)
{

 lcd_cs1=0;
 lcd_rs=0;
 lcd_e=0;
 lcd_rw=0;
 P1=data1;
 lcd_e=1;
 delay_us(1);
 lcd_cs1=1;
 lcd_e=0;
}

//写数据到LCD 模块
void transfer_data_lcd(int data1)
{
 lcd_cs1=0;
 lcd_rs=1;
 lcd_e=0;
 lcd_rw=0;
 P1=data1;
 lcd_e=1;
//     delay_us(1);
 lcd_cs1=1;
 lcd_e=0;
}

/*LCD 模块初始化*/
void initial_lcd()
{
 lcd_reset=1;
 lcd_reset=0;                          //硬件复位
 delay(10);
 lcd_reset=1;                          //硬件复位完成后置高
 delay(10);

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

```
 transfer_command_lcd(0xC0);     // 设置VOP
 transfer_data_lcd(0x2c);        // 设置 VOP 的值的低 8 位（总共 9 位）,每调一级是 0.03667V
 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(0x39);     // 39: 设置为黑白模式
//     transfer_command_lcd(0x3A);     // 允许 DDRAM 接口
//     transfer_data_lcd(0x02);         // 允许 DDRAM 接口
 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
}


//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 24 个点阵。方法二：
/*
void transfer_mono_data_24pixel(uchar dat1,uchar dat2,uchar dat3)   //每个字节显示 8 个点阵，显示 8*3=24 个点阵
{
 transfer_mono_data_3pixel(dat1);                  //传送 dat1 的 D7\D6\D5 这 3 位，对应第 1、2、3 个点阵会显示出来，列地址是自动+1 的
 transfer_mono_data_3pixel(dat1<<3);                    //传送 dat1 的 D4\D3\D2 这 3 位，对应第 4、5、6 个点阵会显示出来，列地址是自动+1 的

 transfer_mono_data_3pixel((dat1<<6)|(dat2>>2));   //传送dat1 的D1\D0 和dat2 的D7 位，对应第 7、8、9 个点阵会显示出来，列地址是自动+1
```

```
的

    transfer_mono_data_3pixel(dat2<<1);                    //传送dat2 的D6\D5\D4 这 3 位，对应第 10、11、12 个点阵会显示出来；列地址是自动+1 的

    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

    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

    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

    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\D6 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<1;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<1;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<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_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 384*160 点阵的图像void
disp_384x160(uchar *dp)
{
 int i,j;
 uchar dat1,dat2,dat3;

 lcd_address(0,0,384,160);

 transfer_command_lcd(0x2C);

 for(i=0;i<160;i++)
 {
     for(j=0;j<16;j++)//循环 16 次，每次显示 24 个点阵，合计 384 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
      }
 }
}

//===显示测试画面：例如全显示，隔行显示，隔列显示，雪花显示=====
void test_display(uchar dat1,uchar dat2,uchar dat3)
{
 int i,j;

 lcd_address(0,0,384,160);

 transfer_command_lcd(0x2C);

 for(i=0;i<160;i++)
 {
    for(j=0;j<16;j++)//循环 16 次，每次显示 24 个点阵，合计 384 个点阵
      {

            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 384*160 点阵的 4 灰度级图像
void disp_4gray_384x160(uchar *dp)
{
 uchar i,j;
 uchar dat1,dat2,dat3;
 lcd_address(0,0,384,160); //
 transfer_command_lcd(0x2C);
 for(i=0;i<160;i++)
 {
    for(j=0;j<32;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
      }

 }
}


//-----------------------------
void main ()
{

 while(1)
 {
      initial_lcd();
      clear_screen();//清屏
      disp_384x160(bmp1);        //显示一个 320x160 点阵的图片
      waitkey();
//          clear_screen();//清屏
//          disp_384x160(bmp2);        //显示一个 320x160 点阵的图片
//          waitkey();
      clear_screen();//清屏
```

```
        disp_384x160(bmp4);         //显示一个 320x160 点阵的图片
        waitkey();
        clear_screen();//清屏
        disp_384x160(bmp3);         //显示一个 320x160 点阵的图片
        waitkey();
        test_display(0xff,0xff,0xff);
        waitkey();
        clear_screen();//清屏
        disp_4gray_384x160(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,"ABCDEFG!@#$%^&123");
        display_string_12x24(0,96,"ABCDEFG!@#$%^&123");
        waitkey();

    }
}
```

串行接口程序

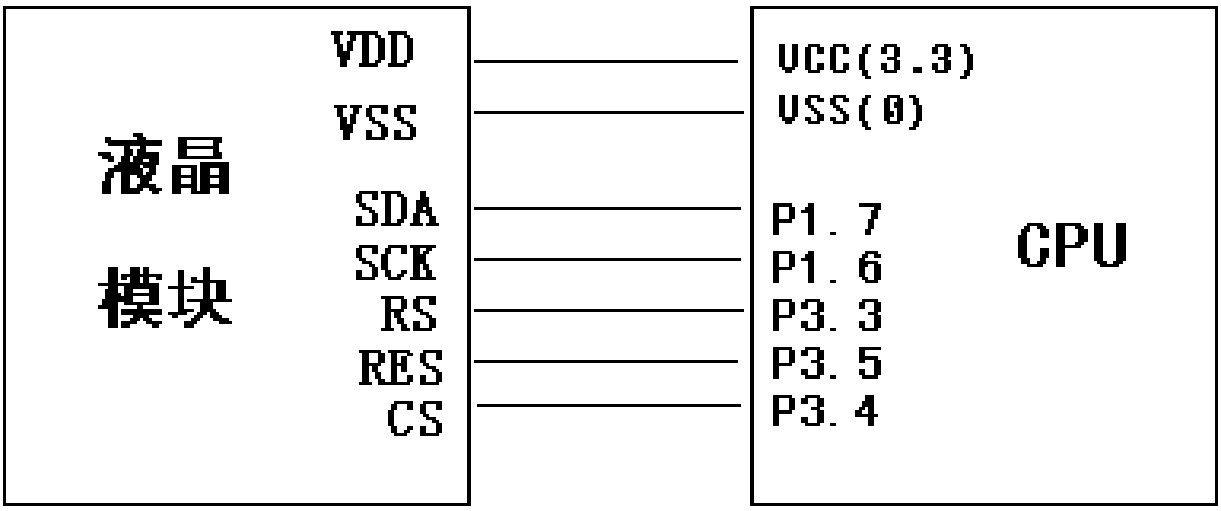


图 9. 串行接口

```
/*      液晶模块型号：JLX384160G-973，
  4 线串行接口：接口有点特别，请注意：
  “SCLK”接在 PCB 印刷为"RS"那一个脚，
  “RS”接在印刷为“D1”那一个脚，
  “SDA”接在印刷为“D0” 那一个脚。
        驱动 IC 是:ST7586S
        版权所有：晶联讯电子：网址 http://www.jlxlcd.cn;
*/
#include <reg51.H>
#include <intrins.h>
#include <Ctype.h>

#include <ASCII_TABLE_5X8_8X16_12x24_16x32_horizontal.h>
sbit key  =  P2^0;   //按键：我的主板上是 P2.0 口与 GND 之间接一个按键
sbit lcd_cs1=P3^4;          /*对应 LCD 的 CS 引脚*/
sbit lcd_reset=P3^5; /*对应 LCD 的 RST 引脚*/
sbit lcd_rs=P1^1;             /*对应 LCD 的 RS 引脚*/
sbit lcd_sclk=P3^3; //对应 LCD 的 SCK(D0)
sbit lcd_sid=P1^0;    //对应 LCD 的 SDA(D1)

sbit Rom_IN   =  P3^1;  //字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI
sbit Rom_OUT  =  P3^2;  //字库 IC 接口定义:Rom_OUT 就是字库 IC 的 SO
sbit Rom_SCK  =  P3^7;  //字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK
sbit Rom_CS   =  P3^6;  //字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#

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

uchar code bmp320160_1[];

//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);
}

////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=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=data1<<=1;
  }
  lcd_cs1=1;
}
```

-END-