JLX400-042-BN 使用说明书

（IPS 全视角）

(焊接式 FPC)

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 JLX400-042 型 TFT 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。

JLX400-042 可以显示 IPS 全视角 320 列*480 行点阵彩色图片，或显示 20 个/行*30 行 16*16 点阵的汉字，或显示 40 个/行*30 行 8*16 点阵的英文、数字、符号，或显示 40 个/行*60 行 8*8 点阵的英文、数字、符号。

2．JLX400-042 图像型点阵 TFT 模块的特性

2.1 IPS 全视角，结构轻、薄、带背光。

2.2 IC 采用 ST7796S,功能强大，稳定性好

2.3 显示内容:

●320*480 点阵彩色图片;

●可选用 32*32 点阵或其他点阵的图片来自编汉字，按照 32*32 点阵汉字来计算可显示 10 个字/行*15 行。

●可选用 16*16 点阵或其他点阵的图片来自编汉字，按照 16*16 点阵汉字来计算可显示 20 个字/行*30 行。

2.4 指令功能强：例如可以用指令控制显示内容顺时针旋转 90、逆时针旋转 90°或倒立竖放。

2.5 接口方式：并口或串行接口方式。

2.6 工作温度宽:-20℃∽+70℃;

2.7 工作温度宽:-30℃∽+80℃;

2.8 TFT 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

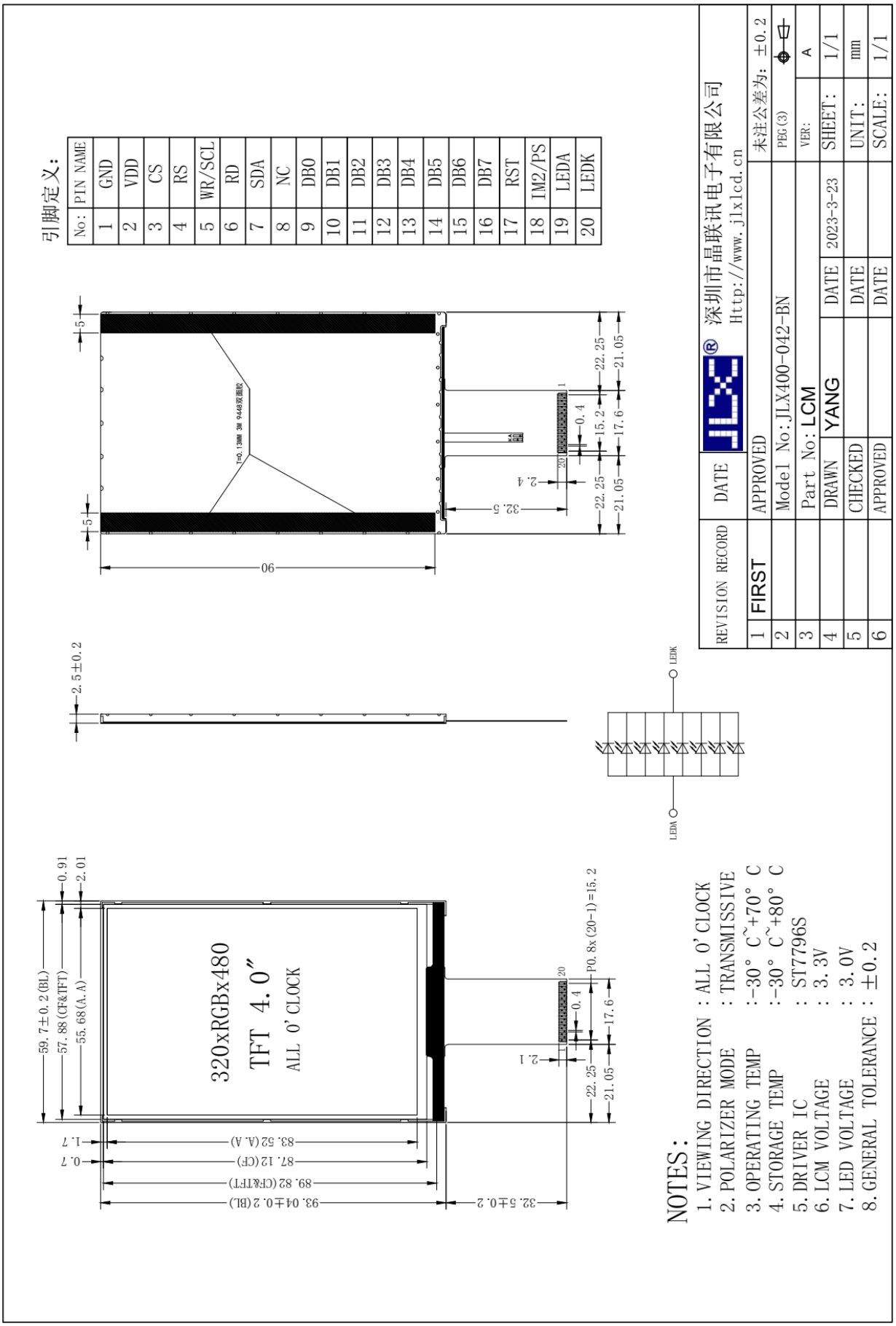


图 1. 带背光的 TFT 模块外形尺寸

模块的接口引脚功能

引 线 号	符 号	名 称	功 能
1	GND	接地	0V
2	VDD	供电电源正极	供电电源正极 3.3V
3	CS	片选	低电平片选
4	RS	寄存器选择信号	H:数据寄存器 0:指令寄存器
5	WR/SCL	写	并口：写功能；串口：串行时钟
6	RD	读	并口：读功能，串口：建议接 VDD 或 VSS
7	SDA	串行数据	串口：SDA 串行数据，并口：空
8	NC	NC	空脚
9	DB0	I/O	并口：数据总线 DB0，串口：建议接 VDD 或 VSS
10	DB1	I/O	并口：数据总线 DB1，串口：建议接 VDD 或 VSS
11	DB2	I/O	并口：数据总线 DB2，串口：建议接 VDD 或 VSS
12	DB3	I/O	并口：数据总线 DB3，串口：建议接 VDD 或 VSS
13	DB4	I/O	并口：数据总线 DB4，串口：建议接 VDD 或 VSS
14	DB5	I/O	并口：数据总线 DB5，串口：建议接 VDD 或 VSS
15	DB6	I/O	并口：数据总线 DB6，串口：建议接 VDD 或 VSS
16	DB7	I/O	并口：数据总线 DB7，串口：建议接 VDD 或 VSS
17	RST	复位	低电平复位，复位完成后，回到高电平，TFT 模块开始工作
18	IM2/PS	接口方式选择	并口:VSS 串口:VDD
19	LEDA	背光电源正极	接 3.0V（接 3.3V 串 5.1 欧电阻）
20	LEDK	背光电源负极	接 GND

表 1：模块的接口引脚功能

4. 基本原理

4.1TFT 屏（LCD）

在 LCD 上排列着 320×480 点阵,320 个列信号与驱动 IC 相连，480 个行信号也与驱动 IC 相连，IC 邦定在 LCD 玻璃上（这种加工工艺叫 COG）.

4.2 背光参数

该型号 TFT 模块带 LED 背光源。它的性能参数如下:
工作温度:-20∽+70℃;
存储温度:-30∽+80℃;
背光板是白色。
正常工作电流为：64∽120mA（LED 灯数共 8 颗,每颗灯是 8～15mA）
工作电压：3.0V（接 3.3V 串 5.1 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 TFT 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
工作温度		-20	+25	+70	℃

储存温度		-30	+25	+80	℃

表 2：最大极限参数

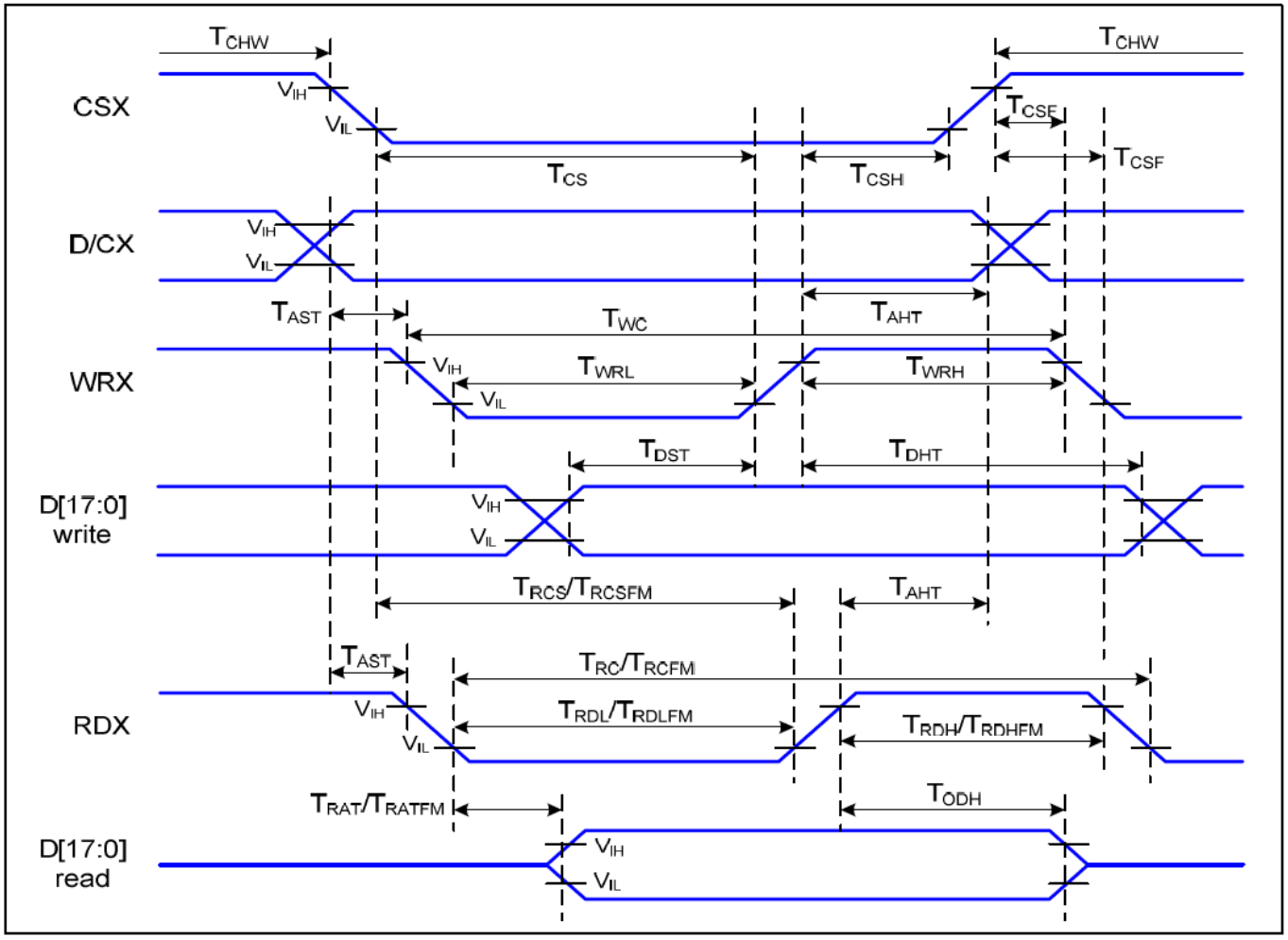
5.2 直流（DC）参数

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	3.6	V
背光工作电压	VLED		2.6	2.8	3.0	V
输入高电平	V_{IHC}	-	0.8xVDD	-	VDD	V
输入低电平	V_{ILC}	-	VSS	-	0.2xVDD	V
输出高电平	V_{OHC}	I_{OH} = -0.5mA	0.8xVDD	-	VDD	V
输出低电平	V_{OHC}	I_{OL} = -0.5mA	VSS	-	0.2xVDD	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	48	90	120	mA

表 3：直流（DC）参数

6. 读写时序特性

6.1 8080 读写时序特性



Parallel Interface Timing Characteristics (8080-Series MCU Interface)

图 2. 8080 时序图

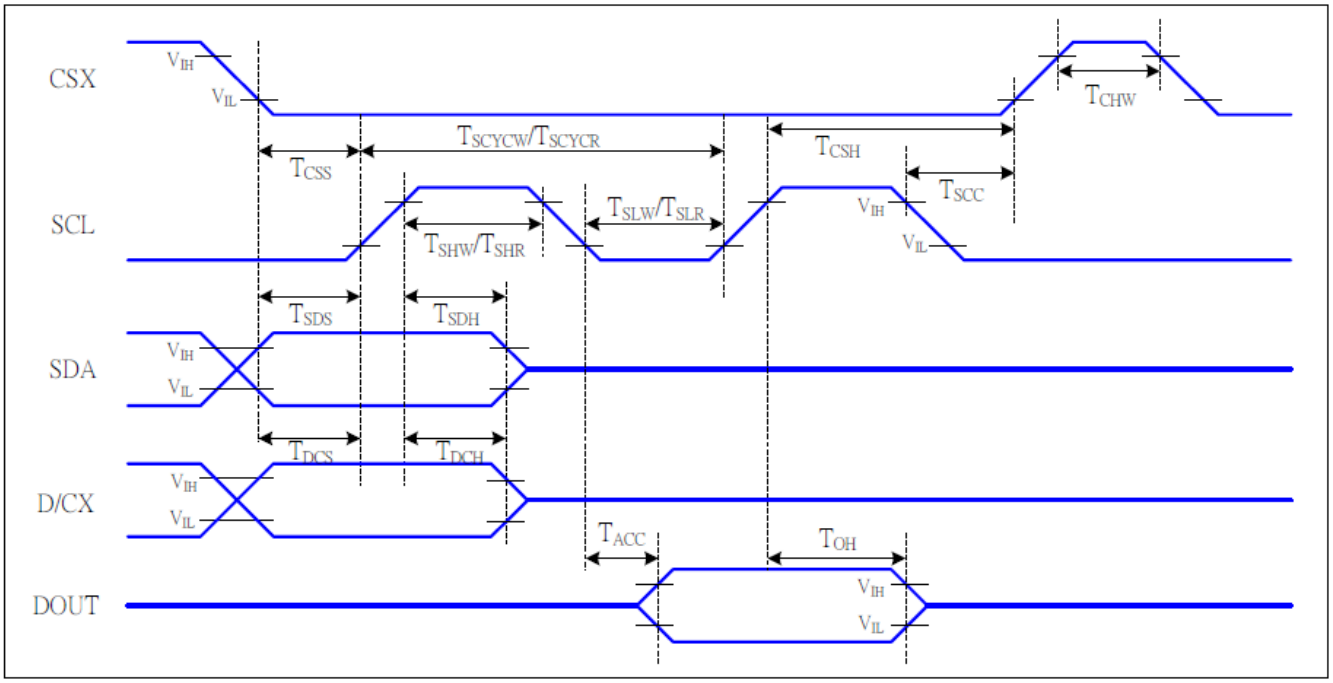
6.2 8080 时序要求（AC 参数）：

表 4.

项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
地址保持时间	A0	TAHt	10	—	—	ns
地址建立时间		TASt	0	—	—	
芯片选择“高”脉冲宽度	CS	TCHW	0			
芯片选择建立时间（写）		TCS	15			
芯片选择建立时间（读）		TRCS	45			
芯片选择保持时间	WR	TCSH	10			
写周期		TWC	66			
控制脉冲“高”持续时间		TWRH	15			
控制脉冲“低”持续时间		TWRL	15			
芯片选择保持时间	RD	TCSH	10			
读周期		TRC	160			
控制脉冲“高”持续时间		TRDH	90			
控制脉冲“低”持续时间		TRDL	45			
数据建立时间	D7-D0	TDST	10			
数据保持时间		TDHT	10			
读取时间		TRAT		40		
输出禁用时间		TODH	20	80		

VDD=3.3V Ta=25℃

6.3 4_SPI 读写时序特性



4-SPI Interface Timing Characteristics

6.4 4_SPI 时序要求（AC 参数）：

Signal	Symbol	Parameter	MIN	MAX	Unit	Description
CSX	T_{CSS}	Chip select setup time (write)	15		ns	
	T_{CSH}	Chip select hold time (write)	15		ns	
	T_{CSS}	Chip select setup time (read)	60		ns	
	T_{SCC}	Chip select hold time (read)	65		ns	
	T_{CHW}	Chip select “H” pulse width	40		ns	
SCL	T_{SCYCW}	Serial clock cycle (Write)	66		ns	-write command & data ram
	T_{SHW}	SCL “H” pulse width (Write)	15		ns	
	T_{SLW}	SCL “L” pulse width (Write)	15		ns	
	T_{SCYCR}	Serial clock cycle (Read)	150		ns	-read command & data ram
	T_{SHR}	SCL “H” pulse width (Read)	60		ns	
	T_{SLR}	SCL “L” pulse width (Read)	60		ns	
D/CX	T_{DCS}	D/CX setup time	10		ns	
	T_{DCH}	D/CX hold time	10		ns	
SDA (DIN)	T_{SDS}	Data setup time	10		ns	
	T_{SDH}	Data hold time	10		ns	
DOUT	T_{ACC}	Access time	10	50	ns	For maximum CL=30pF
	T_{OH}	Output disable time	15	50	ns	For minimum CL=8pF

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

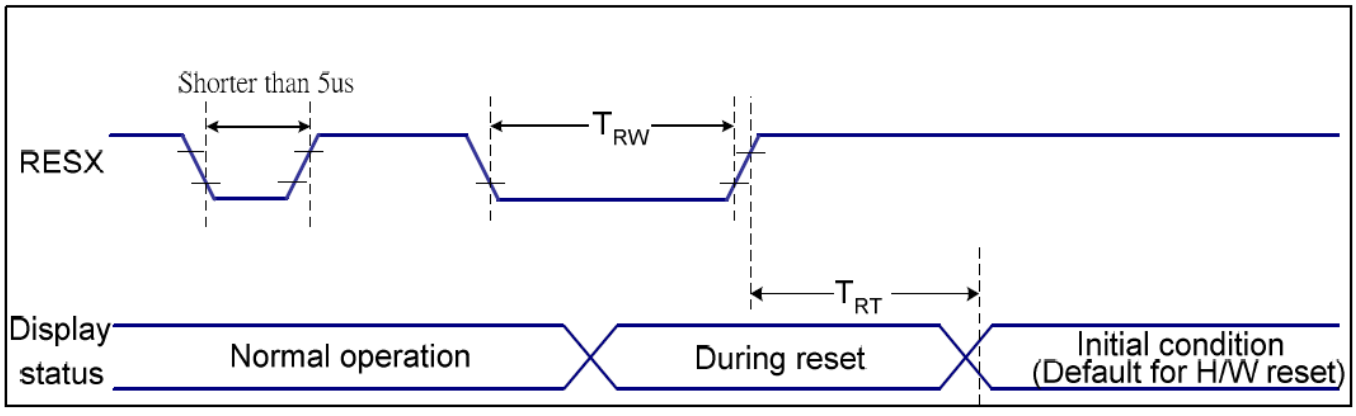


图 3：**电源启动后复位的时序**

表 5：电源启动后复位的时序要求 VDD=3.3V, Ta = 25℃

项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
复位时间	tR		100	--	--	ms
复位保持低电平的时间	tRW	引脚：RES	100	--	--	ms

7．指令功能：

7.1 指令表

COMMAND Table 1														
Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
NOP	0	↑	1	-	0	0	0	0	0	0	0	0	(00h)	No operation
SWRESET	0	↑	1	-	0	0	0	0	0	0	0	1	(01h)	Software reset
RDDID	0	↑	1	-	0	0	0	0	0	1	0	0	(04h)	Read display ID
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10		ID1 read
	1	1	↑	-	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20		ID2 read
	1	1	↑	-	ID37	ID36	ID35	ID34	ID33	ID32	ID31	ID30		ID3 read
Read Number of the Errors on DSI	0	↑	1		0	0	0	0	0	1	0	1	(05h)	Read DSI
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑		D7	D6	D5	D4	D3	D2	D1	D0		
RDDST	0	↑	1	-	0	0	0	0	1	0	0	1	(09h)	Read display status
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	BSTON	MY	MX	MV	ML	RGB	ST25	ST24		-
	1	1	↑	-	ST23	IFPF2	IFPF1	IFPF0	IDMON	PTLON	SLOUT	NORON		-
	1	1	↑	-	Vscroll	ST14	INVON	ALLON	ALLOFF	DISON	TEON	GCS2		-
	1	1	↑	-	GCS1	GCS0	TEM	ST4	ST3	ST2	ST1	ST0		-
RDDPM	0	↑	1	-	0	0	0	0	1	0	1	0	(0Ah)	Read display power
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	BSTON	IDMON	PTLON	SLPOUT	NORON	DISON	0	0		
RDD MADCTL	0	↑	1	-	0	0	0	0	1	0	1	1	(0Bh)	Read display
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	MY	MX	MV	ML	RGB	DISDL	0	0		-
RDD Interface Pixel Format	0	↑	1	-	0	0	0	0	1	1	0	0	(0Ch)	Read display pixel
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	R3	R2	R1	R0	0	D2	D1	D0		-

COMMAND Table 1														
Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
RDDIM	0	↑	1	-	0	0	0	0	1	1	0	1	(0Dh)	Read display image
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	VSSON	0	INVON	0	0	GC2	GC1	GC0		-
RDDSM	0	↑	1	-	0	0	0	0	1	1	1	0	(0Eh)	Read display signal
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	TEON	TEM	HSYN	VSYN	PIXCLK	DATEN	0	DSIER		-
RDDSDR	0	↑	1	-	0	0	0	0	1	1	1	1	(0Fh)	Read display self-diagnostic result
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	D7	D6	0	0	0	0	0	D0		-
SLPIN	0	↑	1	-	0	0	0	1	0	0	0	0	(10h)	Sleep in
SLPOUT	0	↑	1	-	0	0	0	1	0	0	0	1	(11h)	Sleep out
PTLON	0	↑	1	-	0	0	0	1	0	0	1	0	(12h)	Partial mode on
NORON	0	↑	1	-	0	0	0	1	0	0	1	1	(13h)	Partial off (Normal)
INVOFF	0	↑	1	-	0	0	1	0	0	0	0	0	(20h)	Display inversion off
INVON	0	↑	1	-	0	0	1	0	0	0	0	1	(21h)	Display inversion on
DISPOFF	0	↑	1	-	0	0	1	0	1	0	0	0	(28h)	Display off
DISPON	0	↑	1	-	0	0	1	0	1	0	0	1	(29h)	Display on
CASET	0	↑	1	-	0	0	1	0	1	0	1	0	(2Ah)	Column address set
	1	↑	1	-	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8		X address start: 0≦XS≦X
	1	↑	1		XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0		
	1	↑	1		XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8		X address start: S≦XE≦X
	1	↑	1		XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0		

COMMAND Table 1														
Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
RASET	0	↑	1	-	0	0	1	0	1	0	1	1	(2Bh)	Row address set
	1	↑	1	-	YS15	YS14	YS13	YS12	YS11	YS10	YS9	YS8		Y address start: 0≦YS≦Y
	1	↑	1		YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0		
	1	↑	1		YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8		Y address start: S≦YE≦Y
	1	↑	1		YE7	YE6	YE5	YE4	YE3	YE2	YE1	YE0		
RAMWR	0	↑	1	-	0	0	1	0	1	1	0	0	(2Ch)	Memory write
	1	↑	1	-	D7	D6	D5	D4	D3	D2	D1	D0		Write data
RAMRD	0	↑	1	-	0	0	1	0	1	1	1	0	(2Eh)	Memory read
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	D7	D6	D5	D4	D3	D2	D1	D0		Read data
PTLAR	0	↑	1	-	0	0	1	1	0	0	0	0	(30h)	Partial sart/end address set
	1	↑	1	-	PSL15	PSL14	PSL13	PSL12	PSL11	PSL10	PSL9	PSL8		Partial start address: (0, 1,2, ..P)
	1	↑	1	-	PSL7	PSL6	PSL5	PSL4	PSL3	PSL2	PSL1	PSL0		
	1	↑	1	-	PEL15	PEL14	PEL13	PEL12	PEL11	PEL10	PEL9	PEL8		Partial end address (0, 1,2, 3, , P)
	1	↑	1	-	PEL7	PEL6	PEL5	PEL4	PEL3	PEL2	PEL1	PEL0		
VSCRDEF	0	↑	1	-	0	0	1	1	0	0	1	1	(33h)	Vertical scrolling definition
	1	↑	1	-	TFA15	TFA14	TFA13	TFA12	TFA11	TFA10	TFA9	TFA8		
	1	↑	1	-	TFA7	TFA6	TFA5	TFA4	TFA3	TFA2	TFA1	TFA0		
	1	↑	1	-	VSA15	VSA14	VSA13	VSA12	VSA11	VSA10	VSA9	VSA8		
	1	↑	1	-	VSA7	VSA6	VSA5	VSA4	VSA3	VSA2	VSA1	VSA0		
	1	↑	1	-	BFA15	BFA14	BFA13	BFA12	BFA11	BFA10	BFA9	BFA8		
	1	↑	1	-	BFA7	BFA6	BFA5	BFA4	BFA3	BFA2	BFA1	BFA0		
TEOFF	0	↑	1	-	0	0	1	1	0	1	0	0	(34h)	Tearing effect line off
TEON	0	↑	1	-	0	0	1	1	0	1	0	1	(35h)	Tearing effect line on

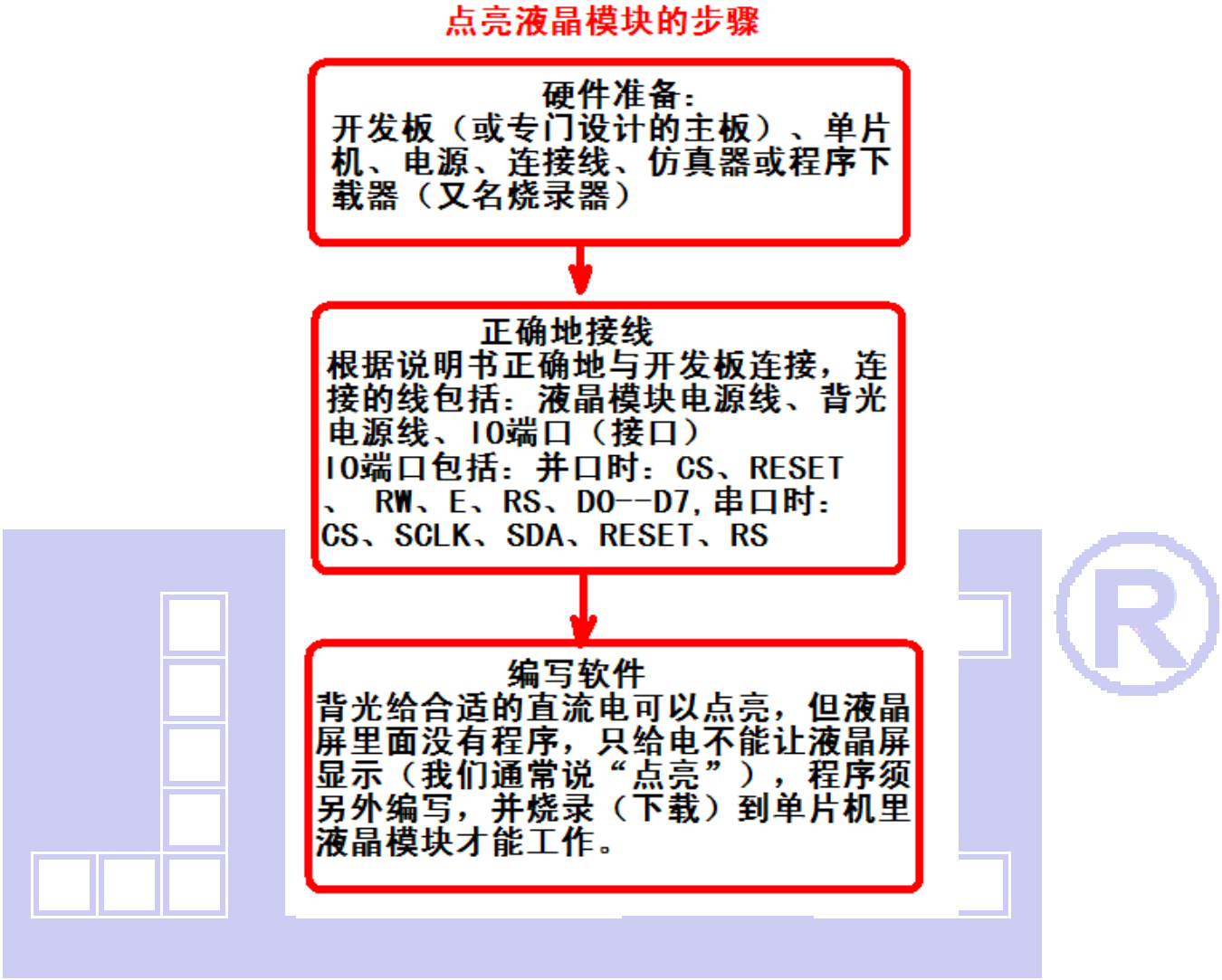
COMMAND Table 1														
Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
	1	↑	1	-	-	-	-	-	-	-	-	TEM		
MADCTL	0	↑	1	-	0	0	1	1	0	1	1	0	(36h)	Memory data access control
	1	↑	1	-	MY	MX	MV	ML	RGB	MH	0	0		-
VSCRSADD	0	↑	1	-	0	0	1	1	0	1	1	1	(37h)	Vertical scrolling start address
	1	↑	1	-	VSP15	VSP14	VSP13	VSP12	VSP11	VSP10	VSP9	VSP8		
	1	↑	1	-	VSP7	VSP6	VSP5	VSP4	VSP3	VSP2	VSP1	VSP0		
IDMOFF	0	↑	1	-	0	0	1	1	1	0	0	0	(38h)	Idle mode off
IDMON	0	↑	1	-	0	0	1	1	1	0	0	1	(39h)	Idle mode on
Interface Pixel Format	0	↑	1	-	0	0	1	1	1	0	1	0	(3Ah)	Interface pixel format
	1	↑	1	-	R3	R2	R1	R0	0	D2	D1	D0		Interface format
RAMWRC	0	↑	1	-	0	0	1	1	1	1	0	0	(3Ch)	Memory write continue
	1	↑	1	-	D17	D16	D15	D14	D13	D12	D11	D10		
	1	↑	1	-	Dx7	Dx6	Dx5	Dx4	Dx3	Dx2	Dx1	Dx0		
	1	↑	1	-	Dn7	Dn6	Dn5	Dn4	Dn3	Dn2	Dn1	Dn0		
RAMRDC	0	↑	1	-	0	0	1	1	1	1	1	0	(3Eh)	Memory read continue
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy Read
	1	1	↑	-	D17	D16	D15	D14	D13	D12	D11	D10		
	1	1	↑	-	Dx7	Dx6	Dx5	Dx4	Dx3	Dx2	Dx1	Dx0		
	1	1	↑	-	Dn7	Dn6	Dn5	Dn4	Dn3	Dn2	Dn1	Dn0		
TESCAN	0	↑	1	-	0	1	0	0	0	1	0	0	(44h)	Set tear scanline
	1	↑	1	-	N15	N14	N13	N12	N11	N10	N9	N8		
	1	↑	1	-	N7	N6	N5	N4	N3	N2	N1	N0		
RDTESCAN	0	↑	1	-	0	1	0	0	0	1	0	1	(45h)	Get scanline
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy Read
	1	1	↑	-	N15	N14	N13	N12	N11	N10	N9	N8		
	1	1	↑	-	N7	N6	N5	N4	N3	N2	N1	N0		

COMMAND Table 1														
Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
WRDISBV	0	↑	1	-	0	1	0	1	0	0	0	1	(51h)	Write display brightness
	1	↑	1	-	DBV7	DBV6	DBV5	DBV4	DBV3	DBV2	DBV1	DBV0		
RDDISBV	0	↑	1	-	0	1	0	1	0	0	1	0	(52h)	Read display brightness value
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	DBV7	DBV6	DBV5	DBV4	DBV3	DBV2	DBV1	DBV0		
WRCTRLD	0	↑	1	-	0	1	0	1	0	0	1	1	(53h)	Write CTRL display
	1	↑	1	-	0	0	BCTRL	0	DD	BL	0	0		
RDCTRLD	0	↑	1	-	0	1	0	1	0	1	0	0	(54h)	Read CTRL value dsiplay
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	0	0	BCTRL	0	DD	BL	0	0		
WRCABC	0	↑	1	-	0	1	0	1	0	1	0	1	(55h)	Write content adaptive brightness control
	1	↑	1	-	CECTRL	0	CE1	CE0	0	0	C1	C0		
RDCABC	0	↑	1	-	0	1	0	1	0	1	1	0	(56h)	Read content adaptive brightness control
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	0	0	0	0	0	0	C1	C0		
WRCABCMB	0	↑	1	-	0	1	0	1	1	1	1	0	(5Eh)	Write CABC minimum brightness
	1	↑	1	-	CMB7	CMB6	CMB5	CMB4	CMB3	CMB2	CMB1	CMB0		
RDCABCMB	0	↑	1	-	0	1	0	1	1	1	1	1	(5Fh)	Read CABC minimum brightness
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	CMB7	CMB6	CMB5	CMB4	CMB3	CMB2	CMB1	CMB0		

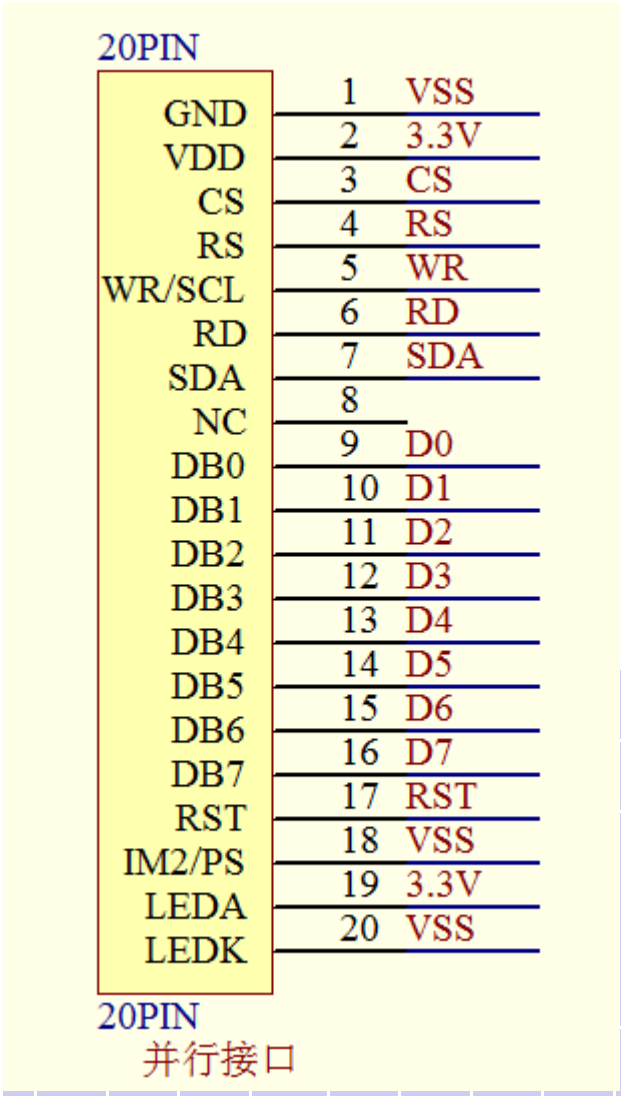
COMMAND Table 1														
Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
RDFCHKSUM	0	↑	1		1	0	1	0	1	0	1	0	(Aah)	Read First Checksum
	1	1	↑		-	-	-	-	-	-	-	-		Dummy read
	1	1	↑		FCS7	FCS6	FCS5	FCS4	FCS3	FCS2	FCS1	FCS0		
RDCCHKSUM	0	↑	1		1	0	1	0	1	0	1	0	(Afh)	Read Continue Checksum
	1	1	↑		-	-	-	-	-	-	-	-		Dummy read
	1	1	↑		CCS7	CCS6	CCS5	CCS4	CCS3	CCS2	CCS1	CCS0		
RDID1	0	↑	1	-	1	1	0	1	1	0	1	0	(Dah)	Read ID1
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	ID17	ID16	ID15	ID14	ID13	ID12	ID11	ID10		Read parameter
RDID2	0	↑	1	-	1	1	0	1	1	0	1	1	(DBh)	Read ID2
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20		Read parameter
RDID3	0	↑	1	-	1	1	0	1	1	1	0	0	(DCh)	Read ID3
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	ID37	ID36	ID35	ID34	ID33	ID32	ID31	ID30		Read parameter

7.2 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



7.3 并行原理图



7.4 程序

TFT 模块与 MPU(以 8051 系列单片机为例)并口接口图如下:

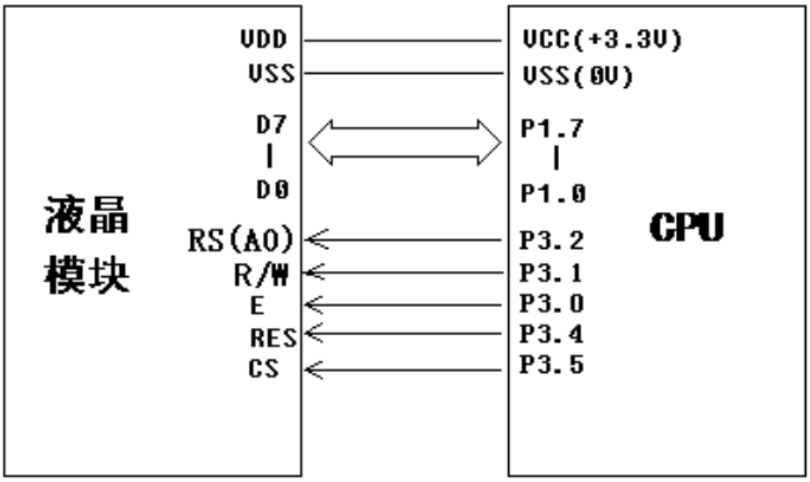


图 4. 并行接口

//型号:JLX400-042-BN、横屏;

//驱动 IC:ST7796S;

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

//液晶屏 IC 所需要的信号线的接口定义
sbit RS=P3^2;
sbit R/W=P3^1;
sbit E=P3^0;
sbit CS=P3^5;
sbit RES=P3^4;
sbit IM2=P3^6;
sbit key=P2^0;          //P2.0 口与 GND 之间接一个按键

void transfer_command(int com1)
{
    CS = 0;
    RS = 0;
    E = 1;
    P1=com1;
    R/W = 0;
    delay_us(2);
    R/W = 1;
    CS = 1;
}

void transfer_data(int data1)
{
   CS = 0;
    RS = 1;
    E = 1;
    P1=data1;
    R/W = 0;
    R/W = 1;
    CS = 1;
}

//连写 2 个字节（即 16 位）数据到 LCD 模块
void transfer_data_16(uint data_16bit)
{
    transfer_data(data_16bit>>8);
    transfer_data(data_16bit);
}

void delay(long i)
{
    int j,k;
```

```
    for(j=0;j<i;j++)
    for(k=0;k<110;k++);
}

void delay_us(long i)
{
    int j,k;
    for(j=0;j<i;j++);
    for(k=0;k<1;k++);
}

void Switch()
{
 repeat:
    if (key==1) goto repeat;
    else delay(1000);
}

void lcd_initial()
{
    RES=1;
    delay(200); //延时 200ms
    RES=0;
    delay(200); //延时 200ms
    RES=1;
    delay(200); //延时 200ms

//************* Start Initial Sequence **********//
    transfer_command(0x11);    //Sleep Out
    delay(200);
    transfer_command(0xf0);   //
    transfer_data(0xc3);      //enable command 2 part 1
    transfer_command(0xf0);
    transfer_data(0x96);      //enable command 2 part 2
    transfer_command(0x36);  //内存数据访问控制
    transfer_data(0x28);     //0x28 横屏显示,0x48 竖屏显示

    transfer_command(0x3a);  //16bit pixel
    transfer_data(0x55);

    transfer_command(0xb4);
    transfer_data(0x01);

    transfer_command(0xb7);
    transfer_data(0xc6);
```

```
transfer_command(0xe8);
transfer_data(0x40);
transfer_data(0x8a);
transfer_data(0x00);
transfer_data(0x00);
transfer_data(0x29);
transfer_data(0x19);
transfer_data(0xa5);
transfer_data(0x33);

transfer_command(0xc1);
transfer_data(0x06);

transfer_command(0xc2);
transfer_data(0xa7);

transfer_command(0xc5);
transfer_data(0x18);

transfer_command(0xe0);  //Positive Voltage Gamma Control
transfer_data(0xf0);
transfer_data(0x09);
transfer_data(0x0b);
transfer_data(0x06);
transfer_data(0x04);
transfer_data(0x15);
transfer_data(0x2f);
transfer_data(0x54);
transfer_data(0x42);
transfer_data(0x3c);
transfer_data(0x17);
transfer_data(0x14);
transfer_data(0x18);
transfer_data(0x1b);

transfer_command(0xe1);  //Negative Voltage Gamma Coltrol
transfer_data(0xf0);
transfer_data(0x09);
transfer_data(0x0b);
transfer_data(0x06);
transfer_data(0x04);
transfer_data(0x03);
transfer_data(0x2d);
transfer_data(0x43);
transfer_data(0x42);
transfer_data(0x3b);
```

```
    transfer_data(0x16);
    transfer_data(0x14);
    transfer_data(0x17);
    transfer_data(0x1b);

    transfer_command(0xf0);
    transfer_data(0x3c);

    transfer_command(0xf0);
    transfer_data(0x69);
    delay(120);                 //延时 120ms
     transfer_command(0x21);    //IPS 需要加这条指令
    transfer_command(0x29);     //Display ON
    transfer_command(0x2c);     // 写数据开始
}

//定义窗口坐标：开始坐标（XS,YS)以及窗口大小（x_total,y_total）
void lcd_address(int XS,int YS,int x_total,int y_total)
{
    int XE,YE;
    XE=XS+x_total-1;
    YE=YS+y_total-1;
    transfer_command(0x2a);     // 设置 X 开始及结束的地址
    transfer_data_16(XS);  // X 开始地址(16 位)
    transfer_data_16(XE);  // X 结束地址(16 位)

    transfer_command(0x2b);     // 设置 Y 开始及结束的地址
    transfer_data_16(YS);  // Y 开始地址(16 位)
    transfer_data_16(YE);  // Y 结束地址(16 位)

    transfer_command(0x2c);     // 写数据开始
}

void mono_transfer_data_16(int mono_data,int font_color,int back_color)
{
    int i;
    for(i=0;i<8;i++)
    {
        if(mono_data&0x80)
        {
            transfer_data_16(font_color);    //当数据是 1 时，显示字体颜色
        }
        else
        {
            transfer_data_16(back_color);    //当数据是 0 时，显示底色
        }
```

```
            mono_data<<=1;
        }
    }

//显示 8x16 点阵的字符串
/*
void disp_string_8x16(int x,int y,char *text,int font_color,int back_color)
{
    int i=0,j,k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x,y,8,16);
            for(k=0;k<16;k++)
            {
                mono_transfer_data_16(ascii_table_8x16[j*16+k],font_color,back_color);
//?a??"ascii_table_8x16[]"?a??êy×é?ú"ASCII_TABLE_5X8_8X16_horizontal.h"à?
            }
            x+=8;
            i++;
        }
        else
        i++;
    }
}
*/

//显示 16x32 点阵的字符串
void disp_string_16x32(int x,int y,char *text,int font_color,int back_color)
{
    int i=0,j,k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x,y,16,32);
            for(k=0;k<64;k++)
            {
                mono_transfer_data_16(ascii_table_16x32[j*64+k],font_color,back_color);
//?a??"ascii_table_8x16[]"?a??êy×é?ú"ASCII_TABLE_5X8_8X16_horizontal.h"à?
            }
            x+=16;
```

```
            i++;
        }
        else
        i++;
    }
}

void display_string_16x16(int x,int y,uchar *text,int font_color,int back_color)
{
    uchar i,j,k;
    uint address;
    j = 0;
    while(text[j] != '\0')                              //'\0' 字符串结束标志
    {
        i = 0;
        address = 1;
        while(Chinese_horizontal_text_16x16[i] > 0x7e)  // >0x7f 即说明不是 ASCII 码字符
        {
            if(Chinese_horizontal_text_16x16[i] == text[j])
            {
                if(Chinese_horizontal_text_16x16[i + 1] == text[j + 1])
                {
                    address = i * 16;
                    break;
                }
            }
            i += 2;
        }
        if(y > 480)
        {
            y=0;
            x+=16;
        }

        if(address != 1)// 显示汉字
        {
            lcd_address(x,y,16,16);
            for(i=0;i<2;i++)
            {
                for(k = 0; k <16; k++)
                {
                mono_transfer_data_16(Chinese_horizontal_code_16x16[address],font_color,back_color);
                address++;
                }
            }
            j+=2;
```

```
        }
        else              //显示空白字符
        {
             lcd_address(x,y,16,16);
             for(i = 0; i <2; i++)
             {
             for(k = 0; k < 16; k++)
             {
             mono_transfer_data_16(0x00,font_color,back_color);
             }
             }
             j+=2;
        }
        x=x+16;
    }
}

void display_string_32x32(int x,int y,uchar *text,int font_color,int back_color)
{
    uchar i,j,k;
    uint address;
    j = 0;
    while(text[j] != '\0')                          //'\0' 字符串结束标志
    {
        i = 0;
        address = 1;
        while(Chinese_horizontal_text_32x32[i] > 0x7e)  // >0x7f 即说明不是 ASCII 码字符
        {
             if(Chinese_horizontal_text_32x32[i] == text[j])
             {
                  if(Chinese_horizontal_text_32x32[i + 1] == text[j + 1])
                  {
                       address = i * 64;
                       break;
                  }
             }
             i += 2;
        }

        if(address != 1)// 显示汉字
        {
             lcd_address(x,y,32,32);
             for(i=0;i<4;i++)
             {
                  for(k = 0; k <32; k++)
                  {
```

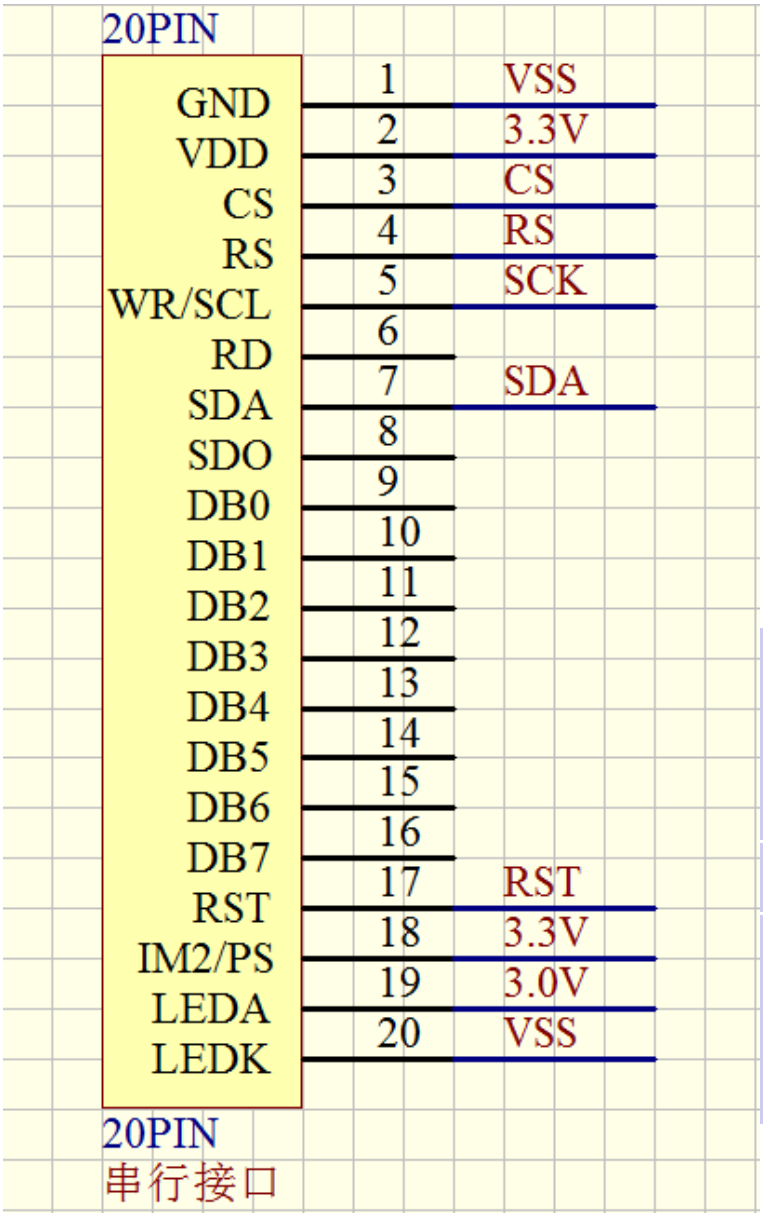
```
                    mono_transfer_data_16(Chinese_horizontal_code_32x32[address],font_color,back_color);
                    address++;
                }
            }
            j+=2;
        }
        else              //显示空白字符
        {
            lcd_address(x,y,32,32);
            for(i = 0; i <4; i++)
            {
            for(k = 0; k < 32; k++)
            {
                mono_transfer_data_16(0x00,font_color,back_color);
            }
            }
            j+=2;
        }
        x=x+32;
    }
}

void display_black(void)
{
    int i,j,k;
    transfer_command(0x2c);          // 写数据开始
    for(i=0;i<480;i++)
    {
        transfer_data_16(0xffff);
    }
    for(i=0;i<318;i++)
    {
        for(k=0;k<1;k++)
        {
            transfer_data_16(0xffff);
        }
        for(j=0;j<478;j++)
        {
            transfer_data_16(0x0000);
        }
        for(k=0;k<1;k++)
        {
            transfer_data_16(0xffff);
        }
    }
    for(i=0;i<480;i++)
```

```
    {
        transfer_data_16(0xffff);
    }
}

//显示一幅彩图
void display_image(int x,int y,uchar *dp)
{
    uchar i,j,k=0;
    lcd_address(x,y,160,160);
    for(i=0;i<160;i++)
    {
        for(j=0;j<160;j++)
        {
            transfer_data(*dp);              //传一个像素的图片数据的高位
            dp++;
            transfer_data(*dp);              //传一个像素的图片数据的低位
            dp++;
        }
    }
}

//全屏显示 RGB 颜色
void display_RGB(void)
{
    int i,j;
    lcd_address(0,0,480,320);
    for(i=0;i<160;i++)
    {
        for(j=0;j<320;j++)
        {
            transfer_data_16(red);
        }
    }

    for(i=0;i<160;i++)
    {
        for(j=0;j<320;j++)
        {
            transfer_data_16(green);
        }
    }

    for(i=0;i<160;i++)
    {
        for(j=0;j<320;j++)
```

```
        {
            transfer_data_16(blue);
        }
    }
}

/*********************************************************************
函数名：Gray16
功能：显示 16 灰阶
*********************************************************************/
void display_Gray16(void)        //color: WHITE, RED, GREEN, BLUE
{
    uint dataa;
    uint i, j, k;

    for(i = 0; i < 320; i++)
    {
        for(j = 0; j < 16; j++)
        {
            dataa = ((2 * j) << 11) | ((4 * j) << 5) | (2 * j);
            for(k = 0; k < 480 / 16; k++)
            {
                transfer_data_16(dataa);
            }
        }
    }
}

//全屏显示一种颜色
void display_color(uint color_data)
{
    int i,j;
    lcd_address(0,0,480,320);
    for(i=0;i<480;i++)
    {
        for(j=0;j<320;j++)
        {
            transfer_data_16(color_data);
        }
    }
}

void main(void)
{
    IM2=0;
```

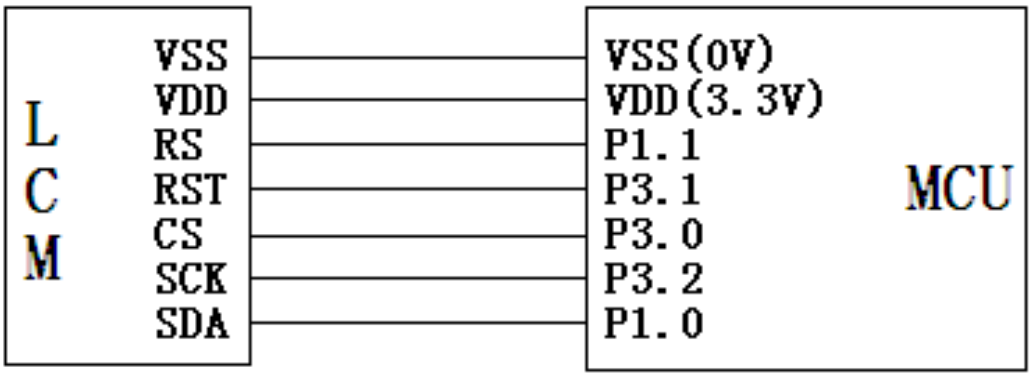
```
    lcd_initial();
    while(1)
    {
        display_color(0x001f);
        display_string_32x32(48,16,"深圳市晶联讯电子有限公司",white,blue);
        display_string_32x32(120,64,"型号",white,blue);
        display_string_32x32(120,112,"视窗",white,blue);
        display_string_32x32(120,160,"驱动",white,blue);
        disp_string_16x32(184,64,":JLX400-042",white,blue);
        disp_string_16x32(184,112,"48.9x73.4mm",white,blue);
        disp_string_16x32(184,160,"IC:ST7789S",white,blue);

        display_string_16x16(40,208,"经营宗旨：制造高品质产品及提供良好服务",white,blue);
        display_string_16x16(40,240,"质量方针：客户至上，质量保证，持续改进，服务到位",white,blue);
        display_string_16x16(40,272,"经营目标：做最好的液晶模块厂家，做客户信得过的企业",white,blue);
        Switch();
        display_RGB();
        Switch();
        display_Gray16();
        Switch();
        display_color(0xf800);
        Switch();
        display_color(0x07e0);
        Switch();
        display_color(0x001f);
        Switch();
        display_black();
        Switch();
        display_color(0xffff);
        Switch();
    }
}
```

7.5 串行原理图



7.6 程序

TFT 模块与 MPU(以 8051 系列单片机为例)串行接口图如下:



与并行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

串行程序：

```
//型号:JLX400-042-BN、横屏；
//驱动 IC:ST7796S；
#include <STC15F2K60S2.H>
#include <chinese_code.h>
//液晶屏 IC 所需要的信号线的接口定义
sbit RS=P1^1;
sbit SCK=P3^2;
sbit SDA=P1^0;
sbit CS=P3^0;
sbit RST=P3^1;
sbit IM2=P3^6;
sbit key=P2^0;          //P2.0 口与 GND 之间接一个按键
void transfer_command(int com1)
{
    char i;
    CS=0;
    RS=0;
    for(i=0;i<8;i++)
    {
        SCK=0;
        if(com1&0x80) SDA=1;
        else SDA=0;
        SCK=1;
        com1=com1<<=1;
    }
}
void transfer_data(int data1)
{
  char i;
    CS=0;
    RS=1;
    for(i=0;i<8;i++)
    {
        SCK=0;
        if(data1&0x80) SDA=1;
        else SDA=0;
        SCK=1;
        data1=data1<<=1;
    }
}
```

-END-