

# JLX280-020-BN 使用说明书

## (焊接式 FPC)

### 目 录

序号	内 容 标 题	页 码
1	概述	2
2	特点	2
3	外形及接口引脚功能	3~4
4	基本原理	4
5	技术参数	4~5
6	时序特性	5~6
7	指令功能及硬件接口与编程案例	7~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 JLX280-020 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。

JLX280-020 可以显示 320 列\*240 行点阵彩色图片，或显示 20 个/行\*15 行 16\*16 点阵的汉字，或显示 40 个/行\*30 行 8\*8 点阵的英文、数字、符号。

## 2. JLX280-020 图像型点阵 TFT 模块的特性

2.1 结构轻、薄、带背光。

2.2 IC 采用 ST7789V, 功能强大，稳定性好

2.3 显示内容：

- 320\*240 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\*7 行。

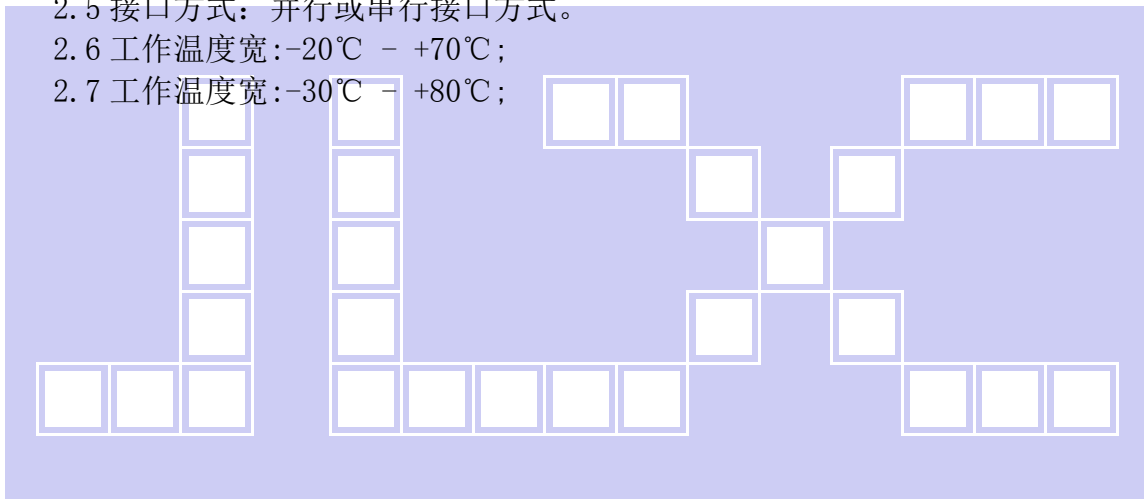
- 可选用 16\*16 点阵或其他点阵的图片来自编汉字，按照 16\*16 点阵汉字来计算可显示 20 个字/行\*15 行。

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：并行或串行接口方式。

2.6 工作温度宽：-20℃ - +70℃；

2.7 工作温度宽：-30℃ - +80℃；



### 3. 外形尺寸及接口引脚功能

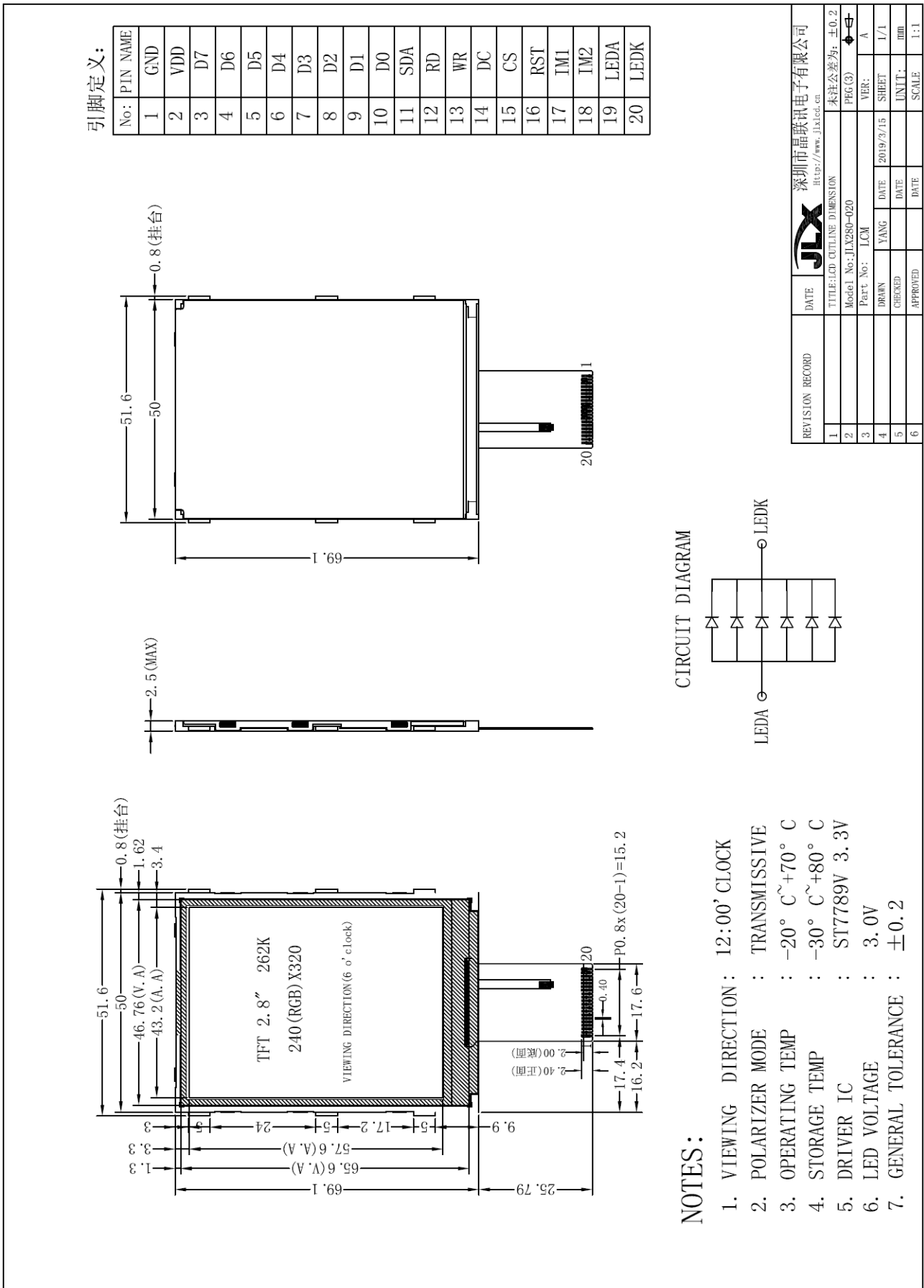


图 1. 带背光的 TFT 模块外形尺寸

## 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	D7	I/O	数据总线 DB7
4	D6	I/O	数据总线 DB6
5	D5	I/O	数据总线 DB5
6	D4	I/O	数据总线 DB4
7	D3	I/O	数据总线 DB3
8	D2	I/O	数据总线 DB2
9	D1	I/O	数据总线 DB1
10	D0	I/O	数据总线 DB0
11	SDA	串行数据	串行数据
12	E(/RD)	读	读功能
13	R/W(/WR)	写	并口:写功能;串口:做为 RS 功能使用
14	DC(RS)	寄存器选择信号	并口: H:数据寄存器 L:指令寄存器;串口:串行时钟 SCK
15	CS	片选	低电平片选
16	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
17	IM1	IM1	IM1=0 选择并口, IM1=1 选择串口
18	IM2	IM2	IM2=0 选择并口, IM2=1 选择串口
19	LEDA	背光电源正极	接 3.0V (接 3.3V 串 10 欧电阻, 接 5.0V 串 51 欧电阻)
20	LEDK	背光电源负极	接 VSS

表 1: 模块的接口引脚功能

## 4. 基本原理

### 4.1 TFT 屏 (LCD)

在 LCD 上排列着 320×240 点阵, 320 个列信号与驱动 IC 相连, 240 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° C;

存储温度: -30~+80° C;

背光板是白色。

正常工作电流为: 32~80mA (LED 灯数共 4 颗, 每颗灯是 10~20 mA)

工作电压: 3.0V (接 3.3V 串 10 欧电阻, 接 5.0V 串 51 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	3.0	3.3	V
工作温度		-20		+70	°C

储存温度		-30		+80	°C
------	--	-----	--	-----	----

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		2.4	-	3.3	V
背光工作电压	VLED		2.9	3.0	3.1	V
输入高电平	V <sub>IHC</sub>	-	0.8xVDD	-	VDD	V
输入低电平	V <sub>ILC</sub>	-	VSS	-	0.2xVDD	V
输出高电平	V <sub>OHC</sub>	I <sub>OH</sub> = -0.5mA	0.8xVDD	-	VDD	V
输出低电平	V <sub>OHC</sub>	I <sub>OL</sub> = -0.5mA	VSS	-	0.2xVDD	V
模块工作电流	I <sub>DD</sub>	VDD = 3.3V	-		0.3	mA
背光工作电流	I <sub>LED</sub>	V <sub>LED</sub> =3.0V	32	60	80	mA

表 3: 直流 (DC) 参数

## 6. 读写时序特性

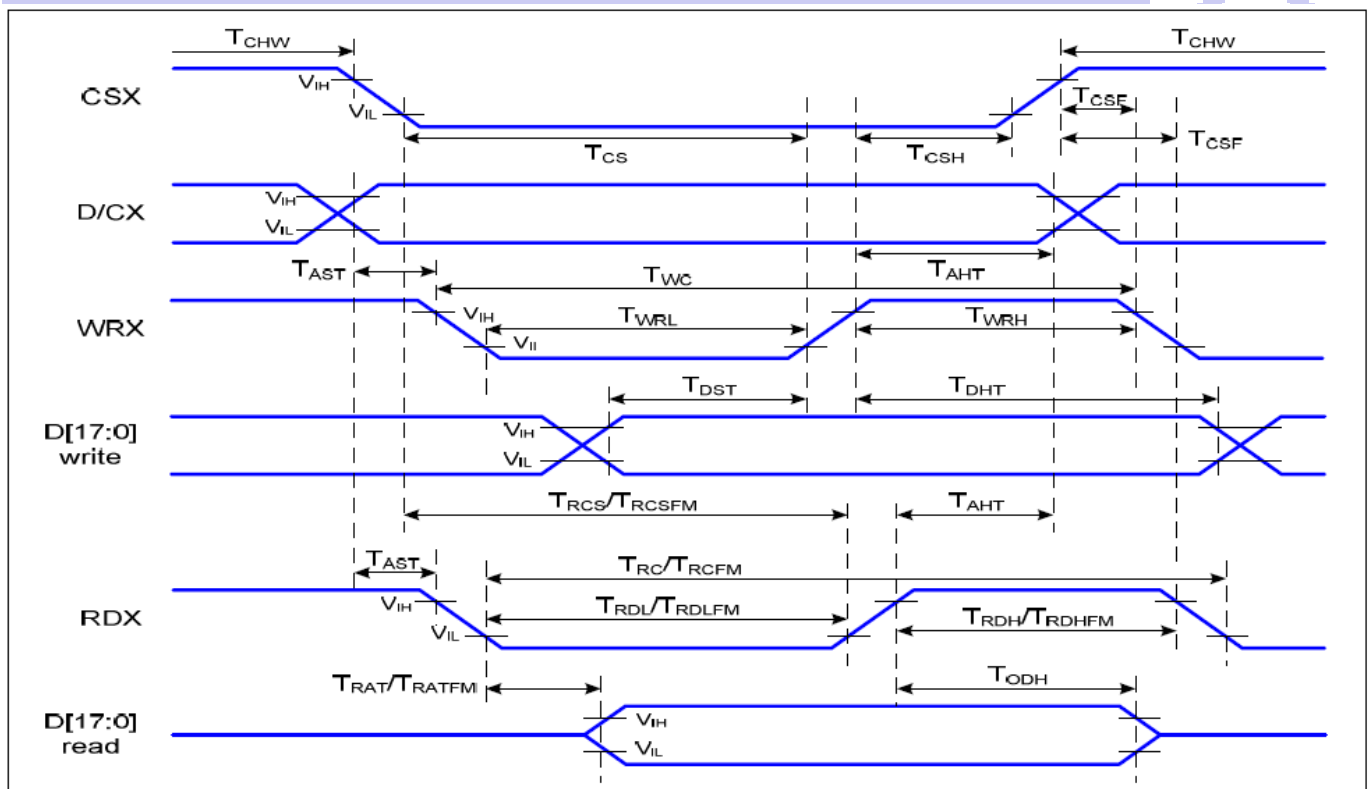


图 2. 8080 时序图

## 6.1 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	T <sub>AHt</sub>	10	—	—	
地址建立时间		T <sub>ASt</sub>	0	—	—	
芯片选择“高”脉冲宽度		T <sub>CHW</sub>	0			

芯片选择建立时间 (写)	CS	$T_{CS}$	15		ns
芯片选择建立时间 (读)		$T_{RCS}$	45		
芯片选择保持时间	WR	$T_{CSH}$	10		
写周期		$T_{WC}$	66		
控制脉冲“高”持续时间		$T_{WRH}$	15		
控制脉冲“低”持续时间		$T_{WRL}$	15		
芯片选择保持时间	RD	$T_{CSH}$	10		
读周期		$T_{RC}$	160		
控制脉冲“高”持续时间		$T_{RDH}$	90		
控制脉冲“低”持续时间		$T_{RDL}$	45		
数据建立时间	D7-D0	$T_{DST}$	10		
数据保持时间		$T_{DHT}$	10		
读取时间		$T_{RAT}$		40	
输出禁用时间		$T_{ODH}$	20	80	

VDD=3.3V  $T_a=25^\circ\text{C}$

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

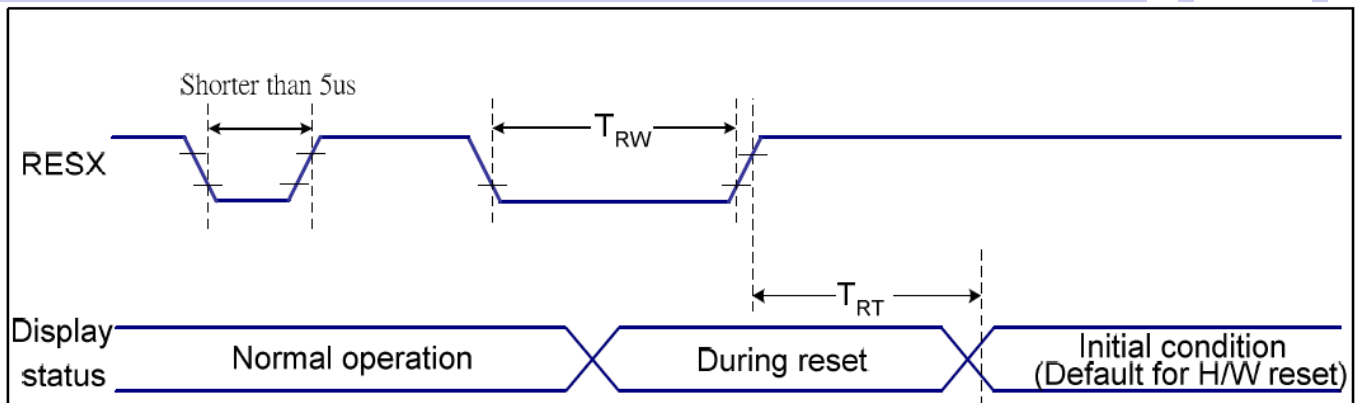


图 3: 电源启动后复位的时序

表 5: 电源启动后复位的时序要求

VDD=3.3V,  $T_a = 25^\circ\text{C}$

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
复位时间	$t_R$		--	--	120	ms
复位保持低电平的时间	$t_{RW}$	引脚: RES	10	--	--	us

## 7. 指令功能:

### 7.1 指令表

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
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	MH	ST24		-
	1	1	↑	-	ST23	IFPF2	IFPF1	IFPF0	IDMON	PTLON	SLOUT	NORON		-
	1	1	↑	-	ST15	ST14	INVON	ST12	ST11	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		
RDMADCTL	0	↑	1	-	0	0	0	0	1	0	1	1	(0Bh)	Read display
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	MY	MX	MV	ML	RGB	MH	0	0		-
RDCOLMOD	0	↑	1	-	0	0	0	0	1	1	0	0	(0Ch)	Read display pixel
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	0	D6	D5	D4	0	D2	D1	D0		-
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

Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
	1	1	↑	-	TEON	TEM	0	0	0	0	0	0		-
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	0		-
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
GAMSET	0	↑	1	-	0	0	1	0	0	0	0	1	(26h)	Display inversion on
	1	↑	1	-	0	0	0	0	GC3	GC2	GC1	GC0		
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:
	1	↑	1		XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0		$0 \leq XS \leq X$
	1	↑	1		XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8		X address start:
1	↑	1		XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0		$S \leq XE \leq X$	
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:
	1	↑	1		YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0		$0 \leq YS \leq Y$
	1	↑	1		YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8		Y address start:
1	↑	1		YE7	YE6	YE5	YE4	YE3	YE2	YE1	YE0		$S \leq YE \leq Y$	
RAMWR	0	↑	1	-	0	0	1	0	1	1	0	0	(2Ch)	Memory write
	1	↑	1	D1[17:8]	D1[7]	D1[6]	D1[5]	D1[4]	D1[3]	D1[2]	D1[1]	D1[0]		Write data
	1	↑	1	Dx[17:8]	Dx[7]	Dx[6]	Dx[5]	Dx[4]	Dx[3]	Dx[2]	Dx[1]	Dx[0]		
1	↑	1	Dn[17:8]	Dn[7]	Dn[6]	Dn[5]	Dn[4]	Dn[3]	Dn[2]	Dn[1]	Dn[0]			
RAMRD	0	↑	1	-	0	0	1	0	1	1	1	0	(2Eh)	Memory read



Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	D1[17:8]	D1[7]	D1[6]	D1[5]	D1[4]	D1[3]	D1[2]	D1[1]	D1[0]		Read data
	1	1	↑	Dx[17:8]	Dx[7]	Dx[6]	Dx[5]	Dx[4]	Dx[3]	Dx[2]	Dx[1]	Dx[0]		
	1	1	↑	Dn[17:8]	Dn[7]	Dn[6]	Dn[5]	Dn[4]	Dn[3]	Dn[2]	Dn[1]	Dn[0]		
PTLAR	0	↑	1	-	0	0	1	1	0	0	0	0	(30h)	Partial start/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)
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
	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	0	0	0		-
VSCRADD	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

Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
COLMOD	0	↑	1	-	0	0	1	1	1	0	1	0	(3Ah)	Interface pixel format
	1	↑	1	-	0	D6	D5	D4	0	D2	D1	D0		Interface format
RAMWRC	0	↑	1	-	0	0	1	1	1	1	0	0	(3Ch)	Memory write continue
	1	↑	1	D1[17:8]	D1[7]	D1[6]	D1[5]	D1[4]	D1[3]	D1[2]	D1[1]	D1[0]		Write data
	1	↑	1	Dx[17:8]	Dx[7]	Dx[6]	Dx[5]	Dx[4]	Dx[3]	Dx[2]	Dx[1]	Dx[0]		
	1	↑	1	Dn[17:8]	Dn[7]	Dn[6]	Dn[5]	Dn[4]	Dn[3]	Dn[2]	Dn[1]	Dn[0]		
RAMRDC	0	↑	1	-	0	0	1	1	1	1	1	0	(3Eh)	Memory read continue
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy Read
	1	1	↑	D1[17:8]	D1[7]	D1[6]	D1[5]	D1[4]	D1[3]	D1[2]	D1[1]	D1[0]		
	1	1	↑	Dx[17:8]	Dx[7]	Dx[6]	Dx[5]	Dx[4]	Dx[3]	Dx[2]	Dx[1]	Dx[0]		
	1	1	↑	Dn[17:8]	Dn[7]	Dn[6]	Dn[5]	Dn[4]	Dn[3]	Dn[2]	Dn[1]	Dn[0]		
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	↑	-	-	-	-	-	-	-	N9	N8		
	1	1	↑	-	N7	N6	N5	N4	N3	N2	N1	N0		
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 display
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	0	0	BCTRL	0	DD	BL	0	0		

Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
WRCACE	0	↑	1	-	0	1	0	1	0	1	0	1	(55h)	Write content adaptive brightness control and Color enhancemnet
	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	CECTRL	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		
RDABCSDR	0	↑	1	-	0	1	1	0	1	0	0	0	(68h)	Read Automatic Brightness Control Self-Diagnostic Result
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑	-	D7	D6	0	0	0	0	0	0		-
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

Instruction	D/CX	WRX	RDX	D17-8	D7	D6	D5	D4	D3	D2	D1	D0	Hex	Function
	1	1	↑	-	-	-	-	-	-	-	-	-		Dummy read
	1	1	↑		ID37	ID36	ID35	ID34	ID33	ID32	ID31	ID30		Read parameter

## 7.2 初始化方法

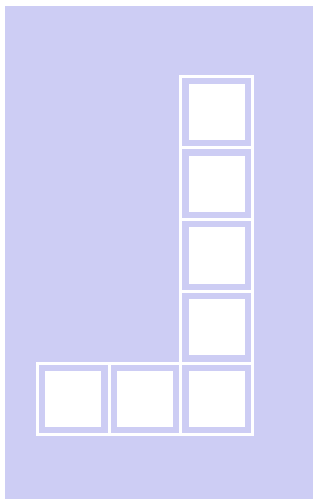
用户所编的显示程序, 开始必须进行初始化, 否则模块无法正常显示, 过程请参考程序

### 点亮液晶模块的步骤

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

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

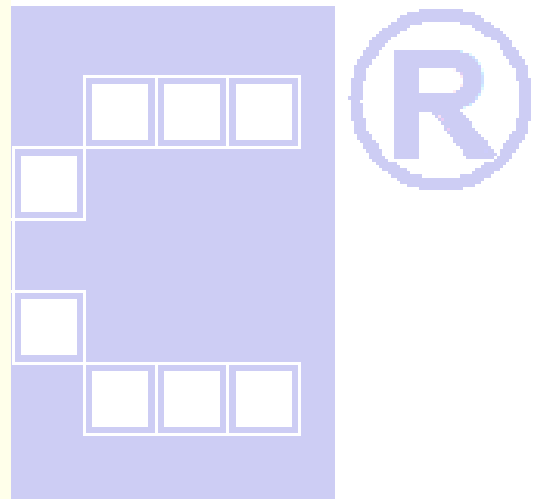
**编写软件**  
背光给合适的直流电可以点亮, 但液晶屏里面没有程序, 只给电不能让液晶屏显示 (我们通常说“点亮”), 程序须另外编写, 并烧录 (下载) 到单片机里液晶模块才能工作。



## 7.3 原理图

## 20PIN

GND	1	VSS
VDD	2	VDD
D7	3	D7
D6	4	D6
D5	5	D5
D4	6	D4
D3	7	D3
D2	8	D2
D1	9	D1
D0	10	D0
SDA	11	
RD	12	RD
WR	13	WR
DC	14	RS
CS	15	CS
RST	16	RST
IM1	17	VSS
IM2	18	VSS
LEDA	19	3.0V
LEDK	20	VSS



## 20PIN

## 7.4 程序

TFT 模块与 MPU (以 8051 系列单片机为例) 接口图如下:

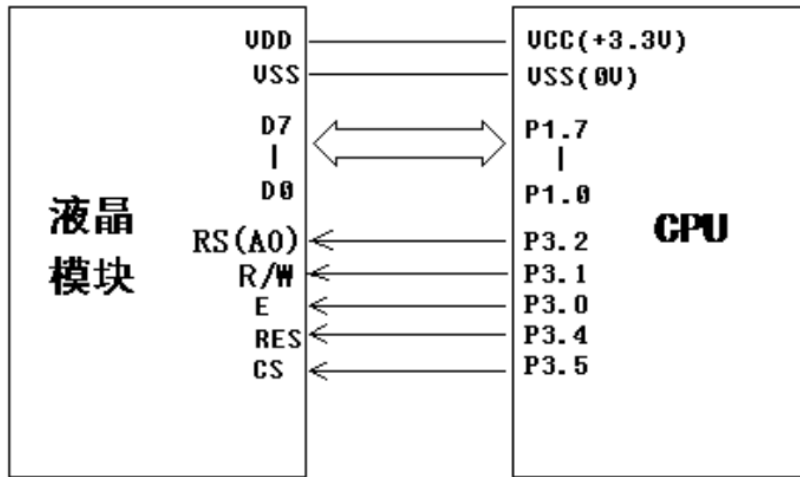


图 4. 并行接口

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

//液晶屏 IC 所需要的信号线的接口定义

```
sbit DC0=P3^2; //RS(A0)
sbit WR0=P3^1; //R/W
sbit RD0=P3^0; //E
sbit CS0=P3^5; //CS
sbit reset=P3^4; //RES
sbit key=P2^0; //P2.0 口与 GND 之间接一个按键
```

```
void transfer_command(int com1)
{
    CS0 = 0;
    DC0 = 0;
    RD0 = 1;
    P1=com1;
    WR0 = 0;
    delay_us(1);
    WR0 = 1;
    CS0 = 1;
}
```

```
void transfer_data(int data1)
{
    CS0 = 0;
    DC0 = 1;
    RD0 = 1;
    P1=data1;
    WR0 = 0;
    WR0 = 1;
    CS0 = 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);
        if (key) goto repeat;
        else ;
}
```

```
void lcd_initial()
```

```
{
    reset=0;
    delay(200);
    reset=1;
    delay(200);
    //***** Start Initial Sequence *****//

    //-----display and color format setting-----//
    transfer_command(0x36);    //行扫描顺序及 RGB，列扫描顺序，横放/竖放
    transfer_data(0x00);
    transfer_data(0x48);

    transfer_command(0xB6);    //显示功能设置：列/行 显示顺序
    transfer_data(0x0A);
}
```

```

transfer_data(0x82);          //改变 SOURCE 线的方向: 0xa2: 左到右, 0x82: 右到左

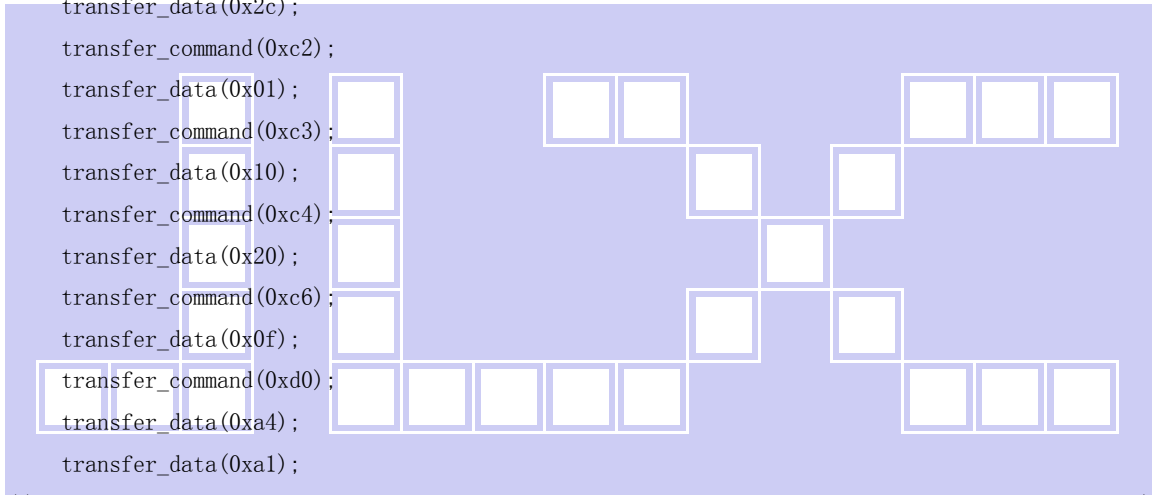
transfer_command(0x3a);      //256K 16bit/pixel
transfer_data(0x05);

//-----ST7789V Frame rate setting-----//
transfer_command(0xb2);
transfer_data(0x0c);
transfer_data(0x0c);
transfer_data(0x00);
transfer_data(0x33);
transfer_data(0x33);
transfer_command(0xb7);
transfer_data(0x35);

//-----ST7789V Power setting-----//
transfer_command(0xbb);
transfer_data(0x28);
transfer_command(0xc0);
transfer_data(0x2c);
transfer_command(0xc2);
transfer_data(0x01);
transfer_command(0xc3);
transfer_data(0x10);
transfer_command(0xc4);
transfer_data(0x20);
transfer_command(0xc6);
transfer_data(0x0f);
transfer_command(0xd0);
transfer_data(0xa4);
transfer_data(0xa1);

//-----ST7789V gamma setting-----//
transfer_command(0xe0);
transfer_data(0xd0);
transfer_data(0x00);
transfer_data(0x02);
transfer_data(0x07);
transfer_data(0x0a);
transfer_data(0x28);
transfer_data(0x32);
transfer_data(0x44);
transfer_data(0x42);
transfer_data(0x06);
transfer_data(0x0e);
transfer_data(0x12);
transfer_data(0x14);
transfer_data(0x17);

```





```

transfer_command(0xe1);
transfer_data(0xd0);
transfer_data(0x00);
transfer_data(0x02);
transfer_data(0x07);
transfer_data(0x0a);
transfer_data(0x28);
transfer_data(0x31);
transfer_data(0x54);
transfer_data(0x47);
transfer_data(0x0e);
transfer_data(0x1c);
transfer_data(0x17);
transfer_data(0x1b);
transfer_data(0x1e);

transfer_command(0x11);    //退出睡眠
delay(200);
transfer_command(0x29);    //打开显示
}
//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;
}
}

```

//全屏显示一种颜色

```
void display_color(int color_data)
```

```

{
    int i, j;
    lcd_address(0, 0, 240, 320);
    for(i=0; i<240; i++)
    {

```

```

        for(j=0; j<320; j++)
        {
            transfer_data_16(color_data);
        }
    }
}

```

```
void display_black(void)
```

```

{
    int i, j, k;
    transfer_command(0x2c); // 写数据开始
    for(i=0; i<240; 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<238; j++)
        {
            transfer_data_16(0x0000);
        }
        for(k=0; k<1; k++)
        {
            transfer_data_16(0xffff);
        }
    }
}

```



```

    }
    for(i=0;i<320;i++)
    {
        transfer_data_16(0xffff);
    }
}

//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??ÿ×é?ú"ASCII_TABLE_5X8_8X16_horizontal.h"à?
            }
            x+=8;
            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 > 320)
{
    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;
}
}

```

//显示 32x32 点阵的单色的图像

```

void disp_32x32(int x,int y,char *dp,int font_color,int back_color)
{
    int i,j;
    lcd_address(x,y,32,32);

```

```

for(i=0;i<32;i++)
{
    for(j=0;j<4;j++)
    {
        mono_transfer_data_16(*dp, font_color, back_color);
        dp++;
    }
}

```

//显示一幅彩图

```

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

```

```

    uchar i,j,k=0;

```

```

    lcd_address(x,y,120,160);

```

```

    for(i=0;i<120;i++)
    {

```

```

        for(j=0;j<160;j++)
        {

```

```

            transfer_data(*dp);

```

```

            dp++;

```

```

            transfer_data(*dp);

```

```

            dp++;
        }
    }
}

```

```

void main(void)
{

```

```

    lcd_initial();

```

```

    while(1)
    {

```

```

        display_color(blue);

```

```

        disp_32x32(40+32*0,8,jing_32x32,white,blue);

```

```

        disp_32x32(40+32*1,8,lian_32x32,white,blue);

```

```

        disp_32x32(40+32*2,8,xun_32x32,white,blue);

```

```

        disp_32x32(40+32*3,8,dian_32x32,white,blue);

```

```

        disp_32x32(40+32*4,8,zi_32x32,white,blue);

```

```

        display_string_16x16(24,56,"深圳市晶联讯电子有限公司",white,blue);

```

```

        display_string_16x16(48,88,"型号",white,blue);

```

```

        disp_string_8x16(80,88,":JLX280-020-BN",white,blue);

```

```

        display_string_16x16(48,120,"视窗",white,blue);

```

```

        disp_string_8x16(80,120,":36.7x48.9mm",white,blue);

```

```

        display_string_16x16(48,152,"驱动",white,blue);

```

```

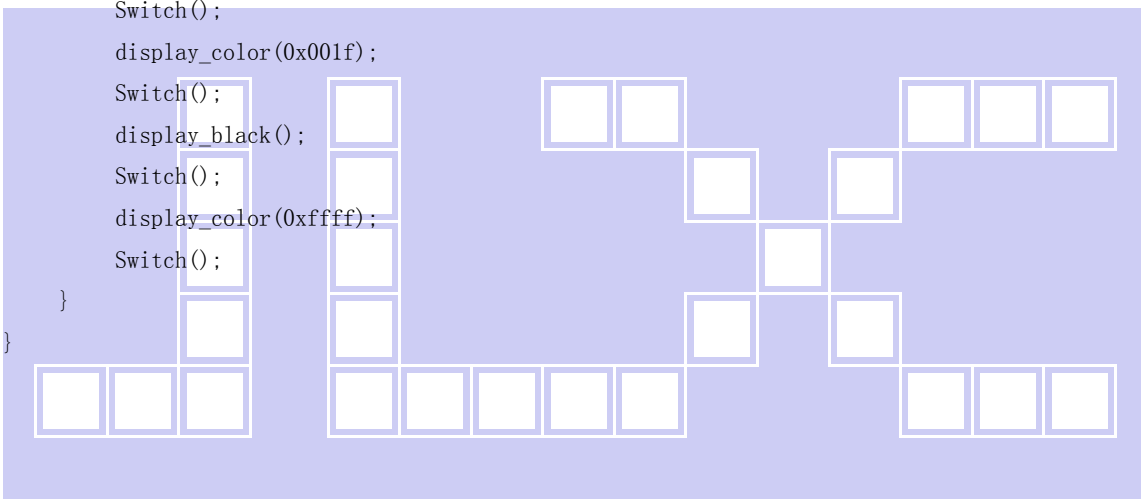
        disp_string_8x16(80,152,"IC:ST7789V",white,blue);
    }
}

```

```
display_string_16x16(0, 184, "经营宗旨：制造高品质产品及服务", white, blue);
display_string_16x16(0, 216, "质量方针：客户至上，质量保证", white, blue);
display_string_16x16(79, 236, "持续改进，服务到位", white, blue);
display_string_16x16(0, 270, "经营目标：做最好的液晶模块厂家", white, blue);
display_string_16x16(79, 292, "做客户信得过的企业", white, blue);
Switch();
```

```
display_image(0, 0, pic1);
display_image(120, 0, pic1);
display_image(0, 160, pic1);
display_image(120, 160, pic1);
Switch();
```

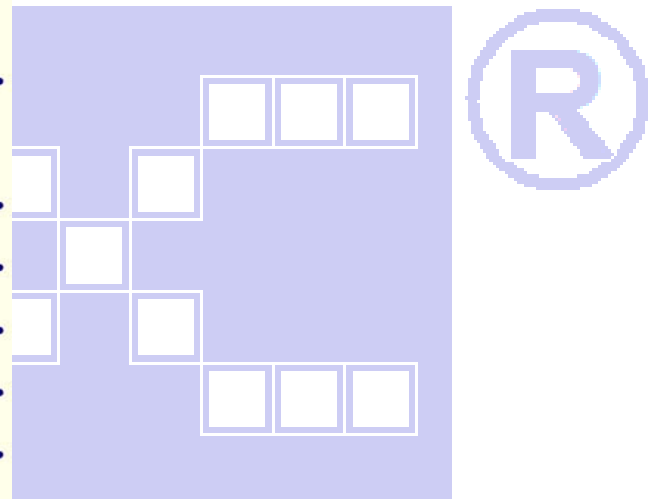
```
display_color(0xf800);
Switch();
display_color(0x07e0);
Switch();
```



## 7.5 串口原理图:

## 20PIN

GND	1	VSS
VDD	2	3.3V
D7	3	
D6	4	
D5	5	
D4	6	
D3	7	
D2	8	
D1	9	
D0	10	
SDA	11	SDA
RD	12	
WR	13	RS
DC	14	SCK
CS	15	CS
RST	16	RST
IM1	17	3.3V
IM2	18	3.3V
LEDA	19	3.0V
LEDK	20	VSS



## 20PIN

## 7.6 程序举例:

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

串程序与并行, 只是接口定义、写数据和命令不一样, 其它都一样

```

#include <STC15F2K60S2.H>
#include <chinese_code.h>

//液晶屏 IC 所需要的信号线的接口定义
sbit cs1=P3^2; //CS
sbit reset=P1^1; //RST
sbit rs=P3^1; //RS
sbit sclk=P1^2; //SCK
sbit sid=P1^3; //SDA
sbit key=P2^0; //P2.0 口与 GND 之间接一个按键

/*写指令到 LCD 模块*/
void transfer_command(int data1)
{
    char i;
    cs1=0;
    rs=0;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
    cs1=1;
}

/*写数据到 LCD 模块*/
    
```





```

void transfer_data(int data1)
{
    char i;
    cs1=0;
    rs=1;
    for (i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
    cs1=1;
}

```

