

# JLX128128G-903-PN 使用说明书

## 目 录

序号	内 容 标 题	页码
1	概述	2
2	字符型模块的特点	2
3	外形及接口引脚功能	3~5
4	基本原理	5~6
5	技术参数	6
6	时序特性	7~11
7	指令功能及硬件接口与编程案例	11~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 JLX128128G-903 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。

JLX128128G-903 可以显示 128×128 点阵单色或 4 灰度级的图片, 或显示 8 个×8 行=64 个的 16\*16 点阵的汉字, 或显示 16 个×8 行=128 个的 8\*16 点阵的英文、数字、符号。或显示 21 个×16 行的 5\*8 点阵的英文、数字、符号。

## 2. JLX128128G-903 图像型点阵液晶模块的特性

1.1 结构牢: 带挡墙背光;

1.2 IC 采用 ST7571, 功能强大, 稳定性好

1.3 功耗低: 1 - 100mW (不开背光 1mW <3.3V@0.3mA>, 开背光不大于 100mW<3.3V@30mA>);

1.4 显示内容:

- 128\*128 点阵单色图片或 4 灰度级的图片,

- 或显示 8 个×8 行=64 个的 16\*16 点阵的汉字。

- 或显示 16 个×8 行=128 个的 8\*16 点阵的英文、数字、符号。

- 或显示 21 个×16 行的 5\*8 点阵的英文、数字、符号。

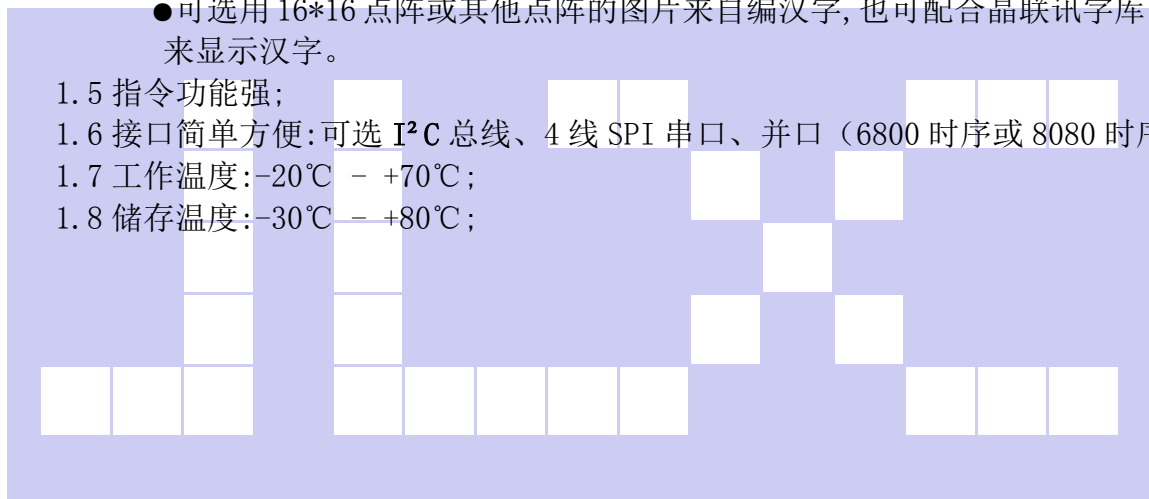
- 可选用 16\*16 点阵或其他点阵的图片来自编汉字, 也可配合晶联讯字库 IC(JLX-GB2312) 来显示汉字。

1.5 指令功能强;

1.6 接口简单方便: 可选 I<sup>2</sup>C 总线、4 线 SPI 串口、并口 (6800 时序或 8080 时序可选择)。

1.7 工作温度: -20℃ - +70℃;

1.8 储存温度: -30℃ - +80℃;



### 3. 外形尺寸及接口引脚功能

#### 3.1 外形尺寸图

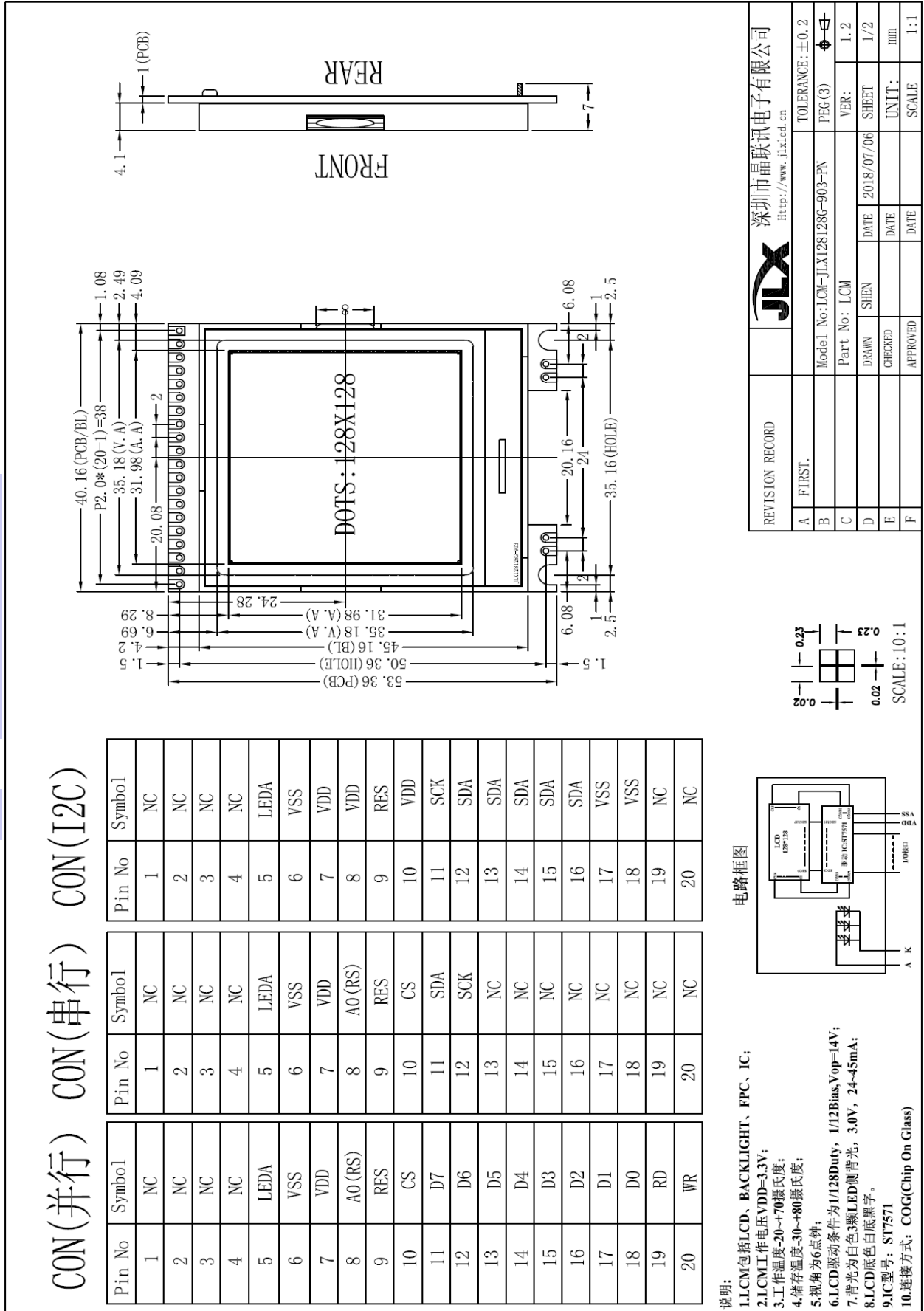


图 1. 外形尺寸

### 3.2 模块的接口引脚功能

#### 3.2.1 并行时接口引脚功能

引线号	符号	名称	功能
1	NC		空脚
2	NC		空脚
3	NC		空脚
4	NC		空脚
5	LEDA	背光电源	供电电源正极
6	VSS	接地	0V
7	VDD	电路电源	供电电源正极
8	A0(RS)	寄存器选择信号	H:数据寄存器 0:指令寄存器 (IC资料上所写为“CD”)
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 DB7
12	D6	I/O	数据总线 DB6
13	D5	I/O	数据总线 DB5
14	D4	I/O	数据总线 DB4
15	D3	I/O	数据总线 DB3
16	D2	I/O	数据总线 DB2
17	D1	I/O	数据总线 DB1
18	D0	I/O	数据总线 DB0
19	RD(E)	使能信号	6800 时序: 使能信号
20	WR	读/写	6800 时序: H:读数据 L:写数据

表 1: 模块并行接口引脚功能

#### 3.2.2 四线串行时接口引脚功能

引线号	符号	名称	功能
1	NC		空脚
2	NC		空脚
3	NC		空脚
4	NC		空脚
5	LEDA	背光电源	供电电源正极
6	VSS	接地	0V
7	VDD	电路电源	供电电源正极
8	A0(RS)	寄存器选择信号	H:数据寄存器 0:指令寄存器 (IC资料上所写为“CD”)
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(SDA)	I/O	串行数据
12	D6(SCK)	I/O	串行时钟
13	D5	I/O	串行接口, 此引脚不用, 建议接 VDD
14	D4	I/O	串行接口, 此引脚不用, 建议接 VDD
15	D3	I/O	串行接口, 此引脚不用, 建议接 VDD
16	D2	I/O	串行接口, 此引脚不用, 建议接 VDD
17	D1	I/O	串行接口, 此引脚不用, 建议接 VDD
18	D0	I/O	串行接口, 此引脚不用, 建议接 VDD

19	RD(E)	使能信号	串行接口, 此引脚不用, 建议接 VDD
20	WR	读/写	串行接口, 此引脚不用, 建议接 VDD

表 2: 4 线 SPI 串行接口引脚功能

 3.2.3 I<sup>2</sup>C 总线时接口引脚功能

引线号	符号	名称	功能
1	NC		空脚
2	NC		空脚
3	NC		空脚
4	NC		空脚
5	LEDA	背光电源	供电电源正极
6	VSS	接地	0V
7	VDD	电路电源	供电电源正极
8	A0 (RS)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器 (IC 资料上所写为“CD”)
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 (SCK)	I/O	串行时钟
12-16	D6-D2 (SDA)	I/O	(D2、D3、D4、D5、D6 连接一起) 串行数据
17	D1	I/O	串行接口, 接 VSS
18	D0	I/O	串行接口, 接 VSS
19	RD(E)	使能信号	串行接口, 此引脚不用, 建议接 VDD
20	WR	读/写	串行接口, 此引脚不用, 建议接 VDD

 表 3: I<sup>2</sup>C 总线接口引脚功能

## 4. 基本原理

## 4.1 液晶屏 (LCD)

在 LCD 上排列着 128×128 点阵, 128 个列信号与驱动 IC 相连, 128 个行信号也与驱动 IC 相连, IC 绑定在 LCD 玻璃上 (这种加工工艺叫 COG)。

## 4.2 工作电图:

图 1 是 JLX128128G-903 图像点阵型模块的电路框图, 它由驱动 IC ST7571 及几个电阻电容组成。

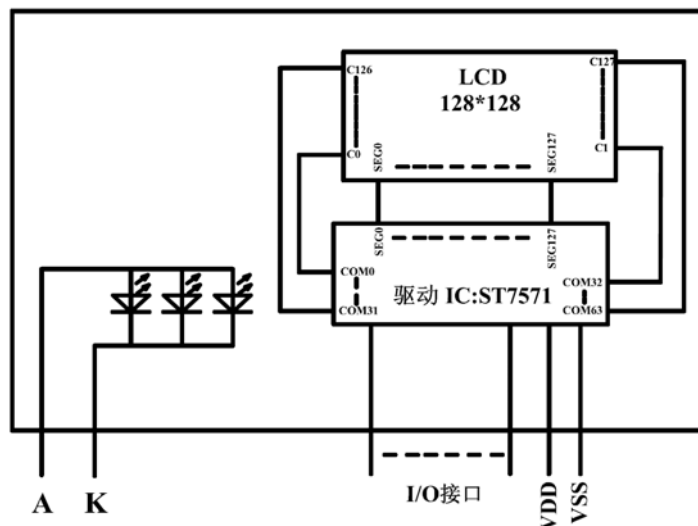


图 2: JLX128128G-903 图像点阵型液晶模块的电路框图

### 4.3 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:  $-20\sim+70^{\circ}\text{C}$ ;

存储温度:  $-30\sim+80^{\circ}\text{C}$ ;

背光板选用白色;

正常工作电流为:  $24\sim45\text{mA}$ ;

工作电压:  $3.0\text{V}$  (PCB 已加限流电阻, 供电同 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 - VSS	-0.3		5.2	V
LCD 驱动电压	VDD - V0	VDD - 13.5		VDD + 0.3	V
静电电压		-	-	100	V
工作温度		-20		+70	$^{\circ}\text{C}$
储存温度		-30		+80	$^{\circ}\text{C}$

表 4: 最大极限参数

### 5.2 直流 (DC) 参数

可以选择 3.3V 供电及 5.0V 供电两种方式:

名称	符号	测试条件	标准值			单位
			MIN	TYPE	MAX	
工作电压	VIN	3.3V 供电	2.7	3.3	3.4	V
输入高电平	VIH	-	2.2		VDD	V
输入低电平	VI0	-	-0.3		0.6	V
输出高电平	VOH	IOH = 0.2mA	2.4		-	V
输出低电平	VO0	I00 = 1.2mA	-		0.4	V
模块工作电流	IDD	VDD = 3.3V	-		0.3	mA
背光工作电流	ILED	VLED=3.0V	24	45	60	mA

表 5: 直流 (DC) 参数

## 6. 读写时序特性 (AC 参数)

### 6.1 4 线 SPI 串行接口写时序特性 (AC 参数)

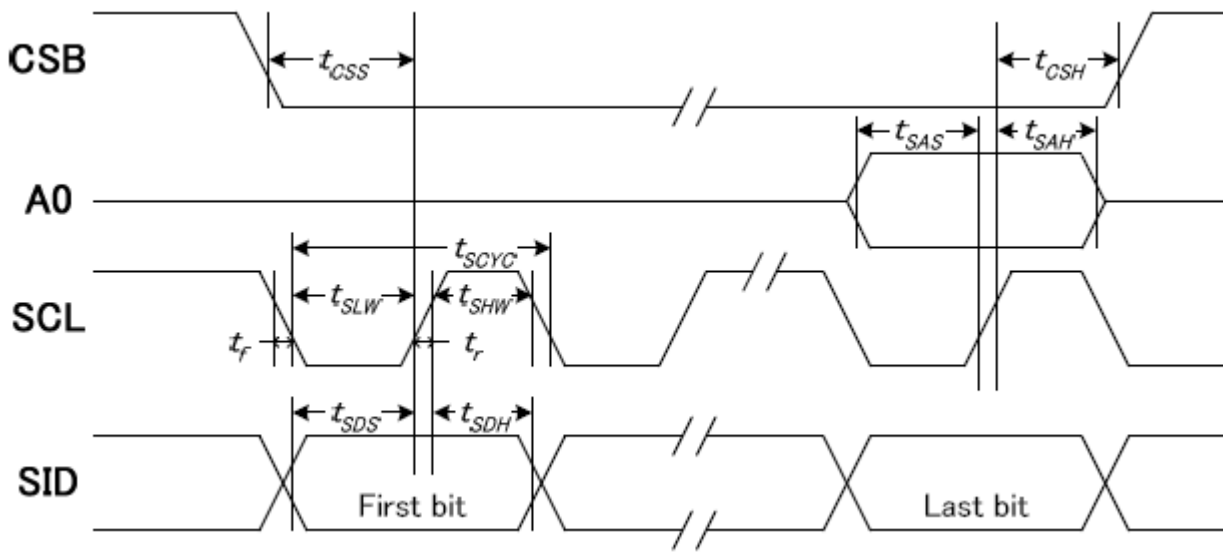


图 3. 从 CPU 写到 ST7571 (Writing Data from CPU to ST7571)

表 6. 写数据到 ST7571 的时序要求

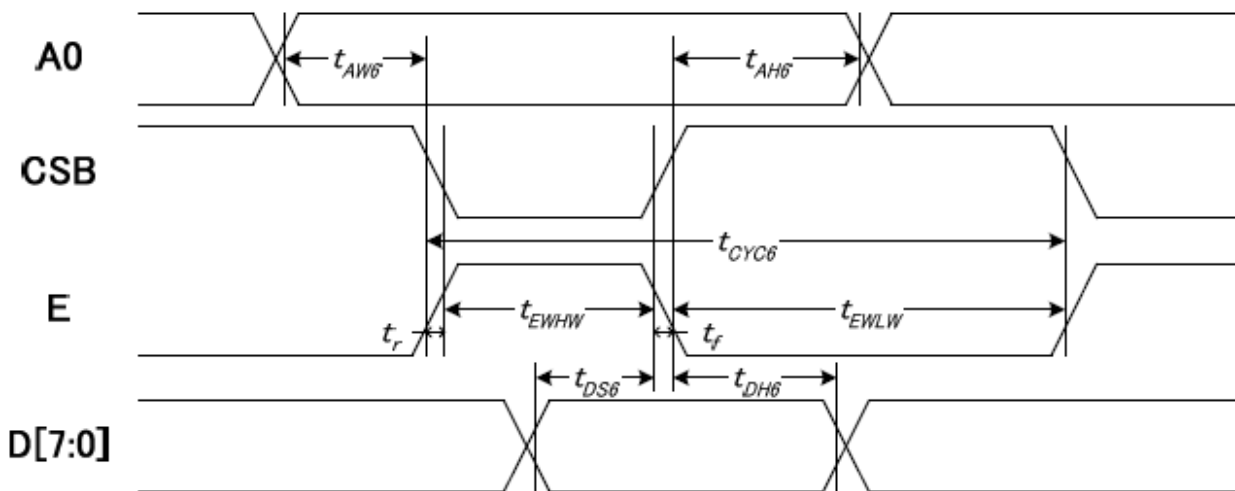
项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
4线 SPI串口时钟周期 (4-line SPI Clock Period)	tSCYC		200	--	--	ns
保持SCK高电平脉宽 (SCL "H" pulse width)	tSHW	引脚: SCL	80	--	--	ns
保持SCLK低电平脉宽 (SCL "L" pulse width)	tSLW		80	--	--	ns
地址建立时间 (Address setup time)	tSAS	引脚: A0	60	--	--	ns
地址保持时间 (Address hold time)	tSAH		30	--	--	ns
数据建立时间 (Data setup time)	tSDS	引脚: SID	60	--	--	ns
数据保持时间 (Data hold time)	tSDH		30	--	--	ns
片选信号建立时间 (CS-SCL time)	tCSS	引脚: CSB	40	--	--	ns
片选信号保持时间 (CS-SCL time)	tCSH		100	--	--	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)



从 CPU 写到 ST7571 (Writing Data from CPU to ST7571)

图 4. 写数据到 ST7571 的时序要求 (6800 系列 MPU)

表 7. 读写数据的时序要求

项目	符号	名称	极限值			单位
			MIN	TYPE	MAX	
地址保持时间	A0	tAH6	0		--	ns
地址建立时间		tAW6	0		--	ns
系统循环时间	E	tCYC6	500		--	ns
使能“低”脉冲宽度		tEHLW	250		--	ns
使能“高”脉冲宽度		tEHW	250		--	ns
写数据建立时间	DB[7: 0]	tDS6	80		--	ns
写数据保持时间		tDH6	30		--	ns

VDD = 1.8~3.3V ± 5%, Ta = -30~85°C

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

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

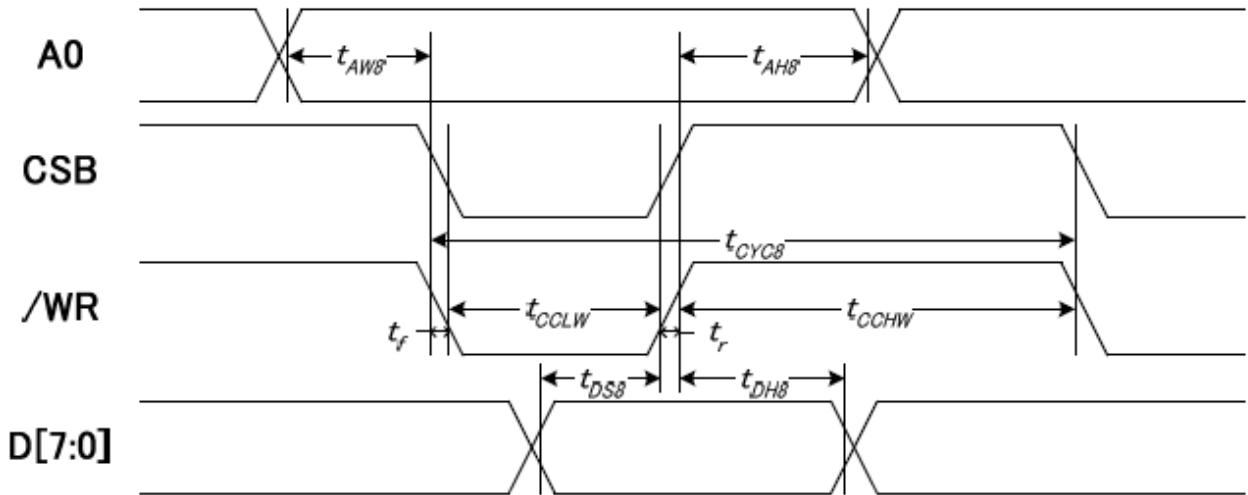
所有的时间, 用 20% 和 80% 作为参考指定的测定。

tewlw 指定为重叠的 CSB “H” 和 “L”。

R / W 信号总是 “H”

## 6.3 8080 时序并行接口的时序特性 (AC 参数)





从 CPU 写到 ST7571 (Writing Data from CPU to ST7571)

图 5. 写数据到 ST7571 的时序要求 (8080 系列 MPU)

表 8. 读写数据的时序要求

项目	符号	名称	极限值			单位
			MIN	TYPE	MAX	
地址保持时间	A0	t <sub>AH8</sub>	0		—	ns
地址建立时间		t <sub>AW8</sub>	0		—	ns
系统循环时间	/WR	t <sub>CYC8</sub>	500		—	ns
使能“低”脉冲宽度		t <sub>CCLW</sub>	250		—	ns
使能“高”脉冲宽度		t <sub>CCHW</sub>	250		—	ns
写数据建立时间	DB	t <sub>DS8</sub>	80		—	ns
写数据保持时间		t <sub>DH8</sub>	30		—	ns

VDD = 1.8~3.3V ± 5%, Ta = -30~85°C

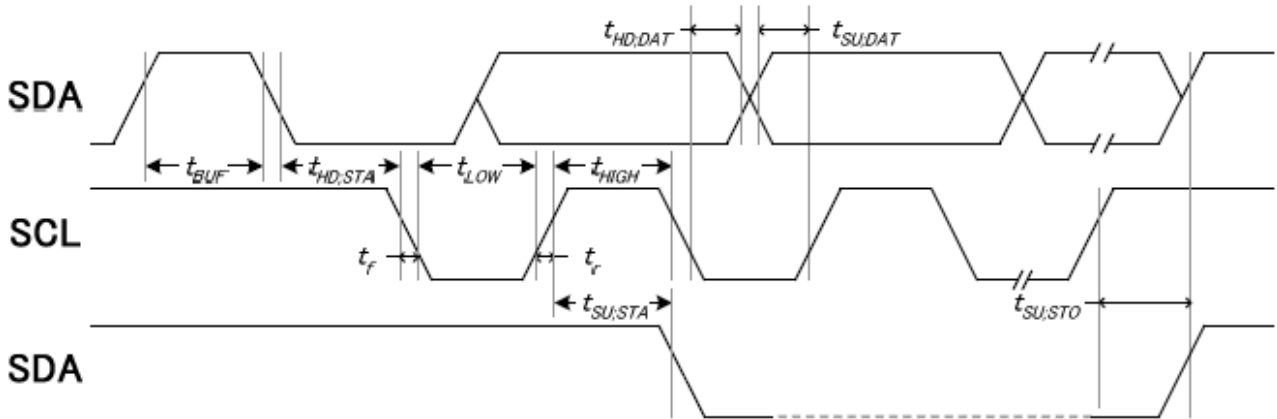
输入信号的上升时间和下降时间 (TR, TF) 是在 15 纳秒或更少的规定。当系统循环时间非常快,

$(TR + TF) \leq (tcyc8 - tcclw - tcchw)$  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

### 6.3 I<sup>2</sup>C 接口的时序特性 (AC 参数)



从 CPU 写到 ST7571 (Writing Data from CPU to ST7571)

 图 6. 写数据到 ST7571 的时序要求 (I<sup>2</sup>C 系列 MPU)

表 9. 读写数据的时序要求

项目	符号	名称	极限值			单位
			MIN	TYPE	MAX	
SCL时钟频率	CSL	FSCLK	---		400	kUZ
SCL时钟的低周期	CSL	TLOW	1.3		---	us
SCL时钟周期	CSL	THIGH	0.6		---	us
数据保持时间	SDA	TSU;Data	100		---	ns
数据建立时间	SDA	THD;Data	0		0.9	us
SCL, SDA 的上升时间	SCL	TR	20+0.1Cb		300	ns
SCL, SDA 下降时间	SCL	TF	20+0.1Cb		300	ns
每个总线为代表的电容性负载		Cb	---		400	pF
一个重复起始条件设置时间	SDA	TSU;SUA	0.6		---	us
启动条件的保持时间	SDA	THD;STA	0.6		---	us
为停止条件建立时间		TSU;STO	0.6		---	us
容许峰值宽度总线		TSW	---		50	ns
开始和停止条件之间的总线空闲时间	SCL	TBUF	1.3			us

VDD = 1.8~3.3V ± 5%, Ta = -30~85°C

注:

所有的时间, 用 20%和 80%作为标准规定的测定。

这是推荐的操作 I C 接口与 VDD1 高于 2.6V。

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

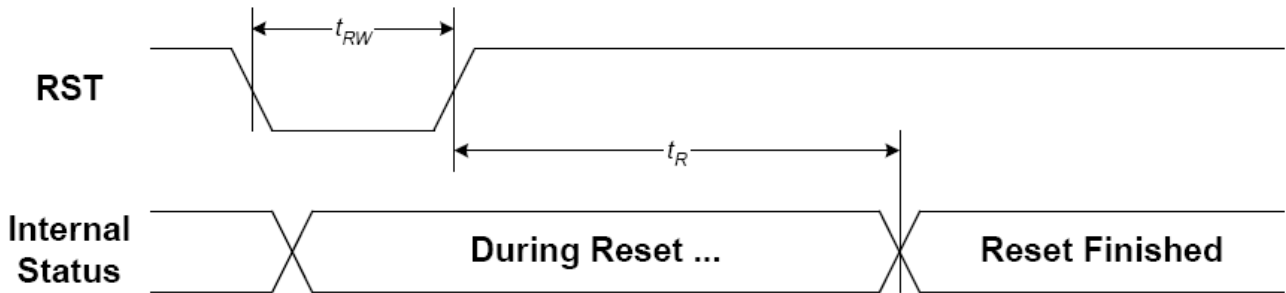


图 7: 电源启动后复位的时序

表 10: 电源启动后复位的时序要求

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
复位时间	tR		120	--	--	ms
复位保持低电平的时间	tRW	引脚: RESET	2.0	--	--	us

## 7. 指令功能:

### 7.1 指令表

表 11

指令名称	指令码										说明
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(1) 设置模式 (Set Mode)	0	0	0	0	1	1	1	0	0	0	2字节的指令 FR [ 3 : 0 ]: 设置帧频 BE [1: 0 ]: 设置增压效率 <b>模式: 0X38, FR、BE 0Xb8</b>
(2) 写显示数据 (Write Display Data)	1	0	写数据							将数据写入 DDRAM	
(3) 设置图标 (Set Icon)	0	0	1	0	1	0	0	0	1	ION	ION = 0: 禁止图标功能 ION = 1: 启用图标功能 并设置页地址= 16
(4) 设置页地址 (Set Page Address)	0	0	1	0	1	1	<b>显示页地址, 共 4 位</b>			设置页地址。每 8 行为一个页, 128 行分为 16 个页, 可设置值为: 0XB0~0XBF 分别对应第一页到第 16 页,	
(5) 列地址高 4 位设置 (Set Column Address (MSB))	0	0	0	0	0	1	<b>列地址的高 4 位</b>			高 4 位与低 4 位共同组成列地址, 指定 128 列中的其中一列。比如液晶模块的第 100 列地址十六进制为 0x64, 那么此指令由 2 个字节来表达: 0x16, 0x04	
	0	0	0	0	0	0	<b>列地址的低 4 位</b>				

Column Address (LSB)											
(6)显示开/关 (Display ON/OFF)		0	0	1	0	1	0	1	1	1	0
											1
		显示开/关: <b>0xAE</b> :关, <b>0xAF</b> : 开									
(7)设置起始行 (Set Display Start Line)		0	0	0	1	0	0	0	0	--	--
		0	0	--	<b>显示初始行地址, 共 7 位</b>						
		设置显示存储器的显示初始行,可设置值为 <b>0x40~0xBF</b> ,分别代表第 <b>0~63</b> 行,针对该液晶屏一般设置为 <b>0x40</b>									
(8)设置COM0 (Set COM0)		0	0	0	1	0	0	0	1	--	--
		0	0	--	C6	C5	C4	C3	C2	C1	C0
		2字节的指令。 指定COM引脚为COM0 <b>0x44</b>									
(9)设置显示Duty (Set Display Duty)		0	0	0	1	0	0	1	0	--	--
		0	0	L7	L6	L5	L4	L3	L2	L1	L0
		2字节的指令。 显示设置Duty									
(10) Set N-line Inversion		0	0	0	1	0	0	1	0	--	--
		0	0	--	--	--	N4	N3	N2	N1	N0
		2-byte instruction. Set N-line inversion counter									
(11) Release N-line Inversion		0	0	1	1	1	0	0	1	0	0
		Exit N-line inversion mode									
(12)正显/反显 (Reverse Display)		0	0	1	0	1	0	0	1	1	0
											1
		显示正显/反显: <b>0xA6</b> : 常规: 正显 <b>0xA7</b> : 反显									
(13)显示全部点阵 (Entire Display ON)		0	0	1	0	1	0	0	1	0	0
											1
		显示全部点阵: <b>0xA4</b> : 常规 <b>0xA5</b> : 显示全部点阵									
(14)电源控制 (Power Control)		0	0	0	0	1	0	1	<b>电压操作模式选择, 共 3 位</b>		
		选择内部电压供应操作模式: D2、D1、D0位分别对应内部升压是否打开(1为打开,0为不打开),电压调整电路是否打开(1为打开,0为不打开),电压跟随器是否打开(1为打开,0为不打开)。  通常是 <b>0x2C,0x2E,0x2F</b> 三条指令按顺序紧接着写,表示依次打开内部升压、电压调整电路、电压跟随器。也可以单写 <b>0x2F</b> ,一次性打开三部分电路。									
(15)选择内部电阻比例 (Select Regulator Register)		0	0	0	0	1	0	0	<b>内部电压值电阻设置</b>		
		选择内部电阻比例(Rb/Ra):可以理解为 <b>粗调</b> 对比度值。可设置范围为: <b>0x20~0x27</b> , 数值越大对比度越浓,越小越淡									
(16) Set Contrast	内部设置液晶电压模式	0	0	1	0	0	0	0	0	0	1
	设置的电压值	0	0	--	--	<b>6位电压值数据, 0~63 共 64 级</b>					
		设置内部电阻微调,可以理解为 <b>微调</b> 对比度值,此两个指令需紧接着使用。上面一条指令 <b>0x81</b> 是不改的,下面一条指令可设置范围为: <b>0x00~0x3F</b> ,数值越大对比度越浓,越小越淡									
(17) LCD 偏压比设置 (Select LCD)		0	0	0	1	0	1	0	B2	B1	B0
		设置偏压比: 此液晶: <b>0x54 1/9bias</b>									

bias)											
(18)设定行扫描方向(Set COM Scan Direction)	0	0	1	1	0	0	1	0	0	0	行扫描顺序选择: <b>0XC0</b> :普通扫描顺序: 从上到下 <b>0XC8</b> :反转扫描顺序: 从下到上
(19)设定列扫描方向(Set SEG Scan Direction)	0	0	1	0	1	0	0	0	0	1	列扫描顺序选择: <b>0xA0</b> : 常规: 列地址从左到右, <b>0xA1</b> : 反转: 列地址从右到左
(20)开振荡器电路(Oscillator ON)	0	0	1	0	1	0	1	0	1	1	<b>0x8B</b> : 开启内部震荡电路
(21)设置睡眠模式 (Release Power-Save Mode)	0	0	1	0	1	0	1	0	0	1	<b>0xA8</b> : 正常模式 <b>0xA9</b> : 睡眠模式
(22)退出省电模式 (Release Power-Save Mode)	0	0	1	1	1	0	0	0	0	1	<b>0XE1</b> : 退出睡眠模式
(23)RESET	0	0	1	1	1	0	0	0	1	0	<b>0XE2</b> : 软件复位
(24)设置显示数据长度 (Set Display Data Length)	-	-	DL7	DL6	DL5	DL4	DL3	DL2	DL1	DL0	2 字节指令, 设定数据仅用在 3-SPI
(25) 扩展指令1 (Extension Command Set1)	0	0	1	1	1	1	1	1	0	1	<b>0XFD</b>
(26) 扩展指令2 (Extension Command Set2)	0	0	1	1	0	1	0	0	0	1	<b>0XD1</b>
(27) 扩展指令3 (Extension Command Set3)	0	0	0	1	1	1	1	0	1	1	<b>0X7B</b>
<b>扩展指令 1</b>											
(1)增加Vop偏移 (Increase Vop offset)	0	0	0	1	0	1	0	0	0	1	<b>0X51</b>
(2)降低Vop偏移 (Decrease Vop offset)	0	0	0	1	0	1	0	0	1	0	<b>0X52</b>
(3)返回正常模式 (Return normal mode)	0	0	0	0	0	0	0	0	0	0	<b>0X00</b>
<b>扩展指令 2</b>											
(1)禁用自动读 (Disable autoread)	0	0	1	0	1	0	1	0	1	0	<b>0XAA</b>
(2) 进入EEPROM模式 (Enter EEPROM mode)	0	0	0	0	0	1	0	0	1	1	<b>0X13</b>

(3)启用阅读模式 (Enable read mode)	0	0	0	0	1	0	0	0	0	0	0X20
(4) 设置读取脉冲 (Set read pulse)	0	0	0	1	1	1	0	0	0	1	0X71
(5) 退出EEPROM模式 (Exit EEPROM mode)	0	0	1	0	0	0	0	0	1	1	0X83
(6) 启用擦除模式 (Enable erase mode)	0	0	0	1	0	0	1	0	1	0	0X4A
(7) 设置擦除脉冲 (Set erase pulse)	0	0	0	1	0	1	0	1	0	1	0X55
(8) 启用写入模式 (Enable write mode)	0	0	0	0	1	1	0	1	0	1	0X35
(9) 设置写入脉冲 (Set write pulse)	0	0	0	1	1	0	1	0	1	0	0X6A
(10) 返回正常模式 (Return normal mode)	0	0	0	0	0	0	0	0	0	0	0X00
扩展指令 3											
(1)设置显示模式 (Set Color Mode)	0	0	0	0	0	1	0	0	0	1	显示模式选择 黑白模式: 0X11, 四灰阶模式: 0X10
(2)返回正常模式	0	0	0	0	0	0	0	0	0	0	0X00

表 11. 指令表

请详细参考 IC 资料”ST7571.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 128\*128 点阵的屏分为 16 个“页”, 从第 0“页”到第 15“页”。

DB7--DB0 的排列方向: 数据是从上向下排列的。最高位 D7 是在最上面, 最低位 D0 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:

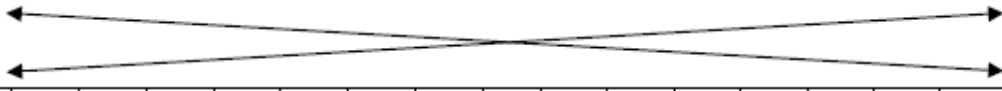
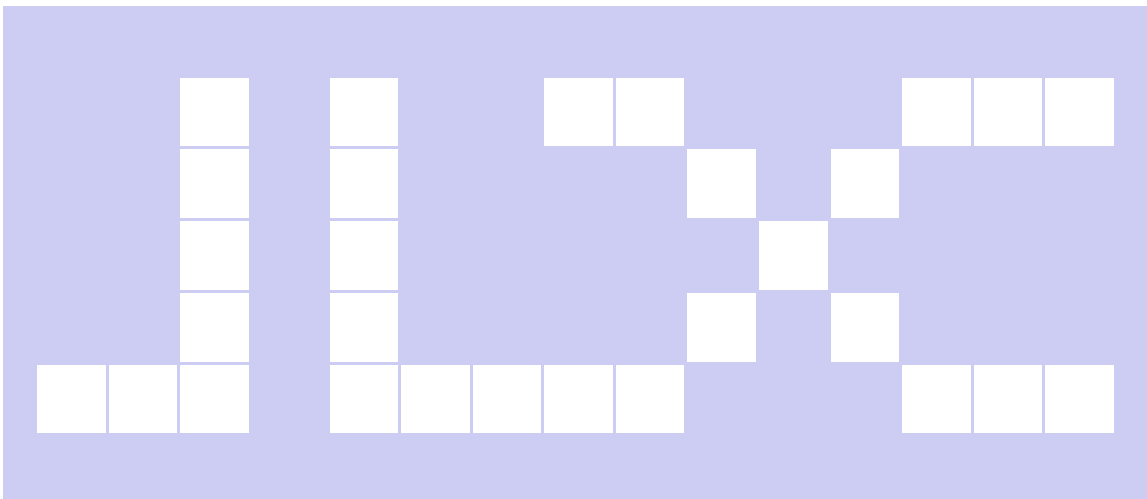
SEG Output	SEG 0		SEG 1		SEG 2		SEG 3		...	SEG 124		SEG 125		SEG 126		SEG 127	
Column Address X[7:1]	00H		01H		02H		03H		...	7CH		7DH		7EH		7FH	
Internal column address X[7:0]	00	01	02	03	04	05	06	07	...	F8	F9	FA	FB	FC	FD	FE	FF
Display Data (MX=0)	1	1	1	0	0	1	0	0	...	1	1	1	0	0	1	0	0
LCD panel display	[Shaded]		[Shaded]		[Shaded]		[Shaded]		...	[Shaded]		[Shaded]		[Shaded]		[Shaded]	
																	
Display data (MX=1)	0	0	0	1	1	0	1	1	...	0	0	0	1	1	0	1	1
LCD panel display	[Shaded]		[Shaded]		[Shaded]		[Shaded]		...	[Shaded]		[Shaded]		[Shaded]		[Shaded]	

Fig. 12 The Relationship between the Column Address and The Segment Outputs

下图摘自 ST7571 IC 资料，可通过“ST7571.PDF”之第 29 页获取最佳效果。



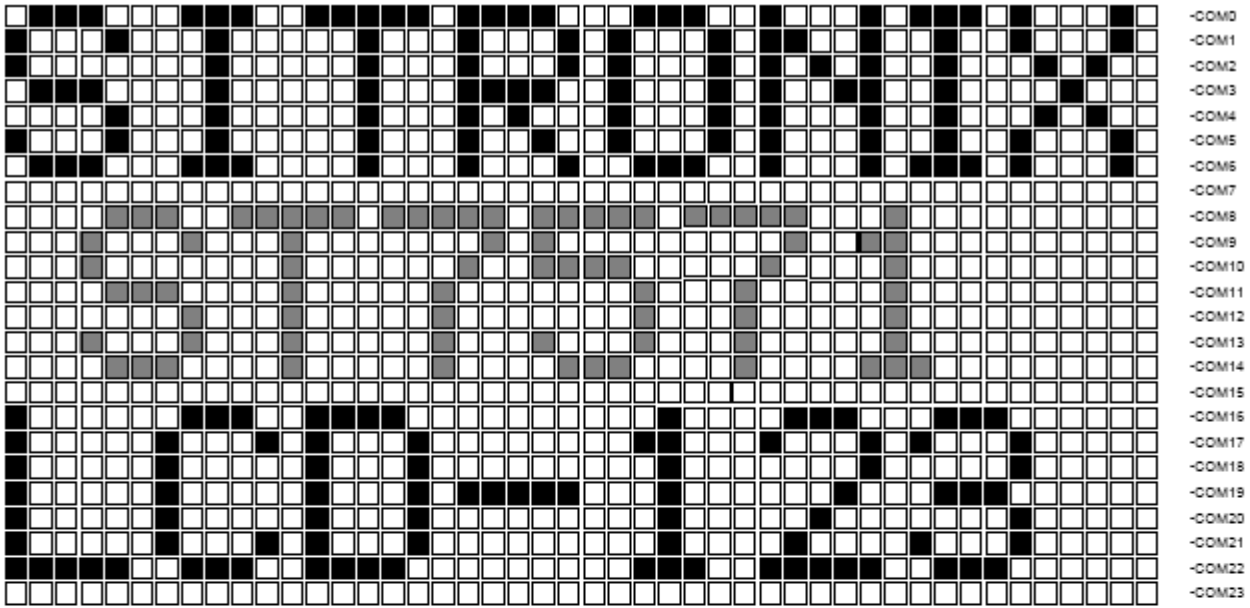


Fig. 15 Reference Example for Partial Display

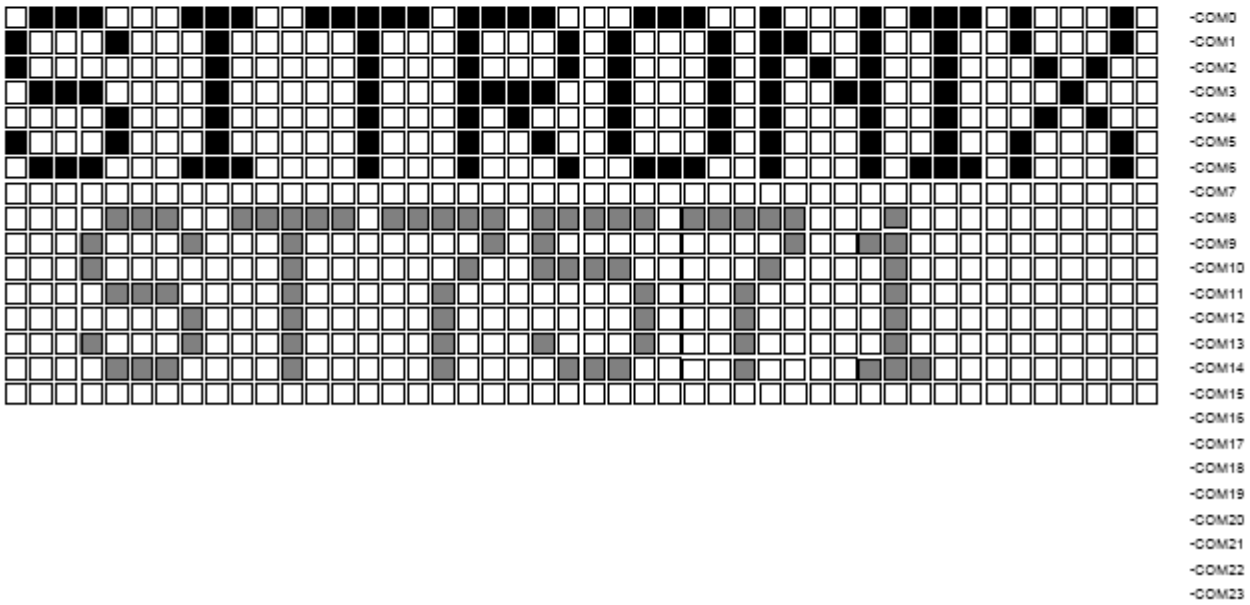


Fig. 16 Partial Display (Partial Display Duty=16, initial COM0=0)



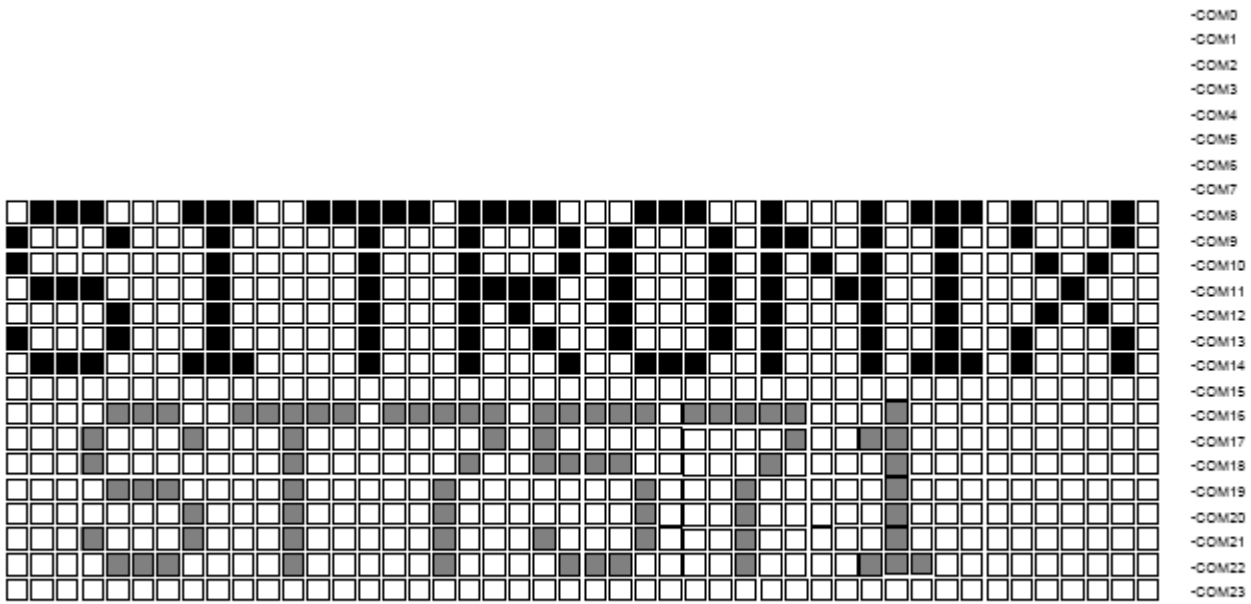


Fig. 17 Moving Display (Partial Display Duty=16, initial COM0=8)

### 7.3 初始化方法

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

#### 点亮液晶模块的步骤

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

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

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

## 7.4 程序举例:

### 7.4.1 并行接口

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:

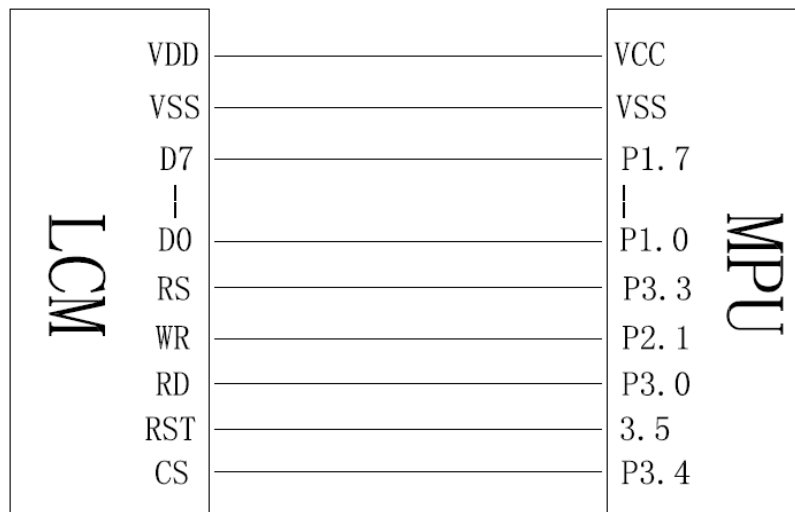


图 8. 并行接口

### 7.5.2 以下是并行接口例程序

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

sbit cs1=P3^4; /*3.4 接口定义*/
sbit reset=P3^5; /*3.3 接口定义*/
sbit rs=P3^3; /*接口定义*/
sbit rd=P3^0; /*接口定义*/
sbit wr=P2^1; /*接口定义。另外 P1.0~1.7 对应 DB0~DB7*/
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
#define ulong unsigned long
```

```
void delay_us(int i);
```

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

```
void transfer_command(int data1)
```

```
{
    cs1=0;
    rs=0;
    rd=0;
    delay_us(1);
```

```

    wr=0;
    P1=data1;
    rd=1;
    delay_us(1);
    cs1=1;
    rd=0;
}

```

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

```

void transfer_data(int data1)

```

```

{
    cs1=0;
    rs=1;
    rd=0;
    delay_us(1);
    wr=0;
    P1=data1;
    rd=1;
    delay_us(1);
    cs1=1;
    rd=0;
}

```

```

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

```

```

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

```

//等待一个按键

```

void waitkey()
{
    repeat:
        if (key==1) goto repeat;
        else;
            delay(1500);
}

```



```

void initial_lcd()
{
    reset=0;
    delay(500);
    reset=1;
    delay(100);

    transfer_command(0x2c);
    delay(200);
    transfer_command(0x2e);
    delay(200);
    transfer_command(0x2f);
    delay(10);

    transfer_command(0xae); //显示关
    transfer_command(0x38); //模式设置
    transfer_command(0xb8); //85HZ
    transfer_command(0xc8); //行扫描顺序
    transfer_command(0xa0); //列扫描顺序

    transfer_command(0x44); //Set initial COM0 register
    transfer_command(0x00);
    transfer_command(0x40); //Set initial display line register
    transfer_command(0x00);

    transfer_command(0xab);
    transfer_command(0x67);
    transfer_command(0x27); //粗调对比度, 可设置范围 0x20~0x27
    transfer_command(0x81); //微调对比度
    transfer_command(0x2a); //微调对比度的值, 可设置范围 0x00~0x3f

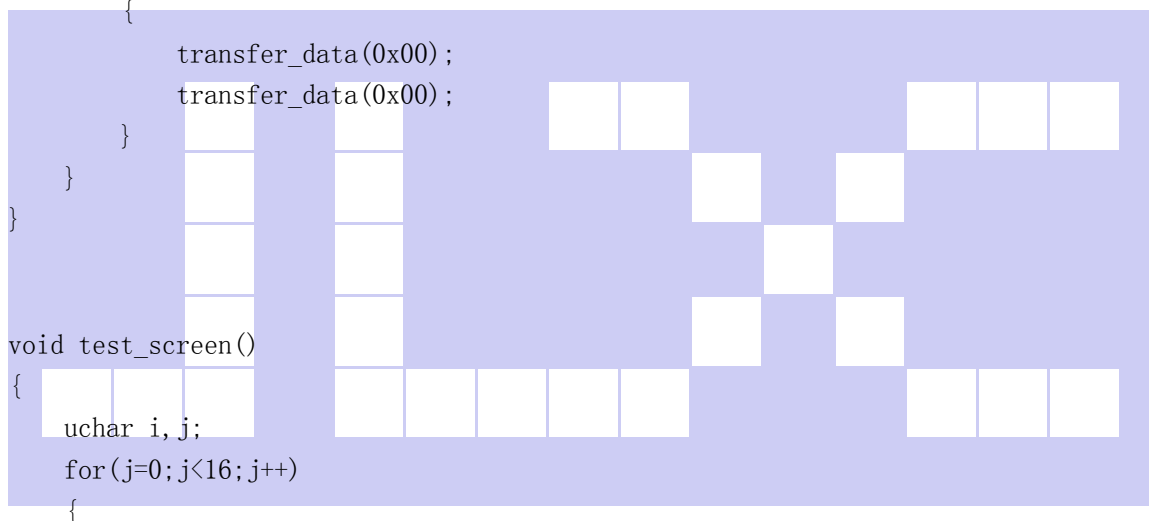
    transfer_command(0x56); //1/11 bias
    transfer_command(0xf3);
    transfer_command(0x04);
    transfer_command(0x93);

    // transfer_command(0x7b); //Extension Command Set3
    // transfer_command(0x11); //Gray mode
    // transfer_command(0x10); //Gray mode
    // transfer_command(0x00);
    transfer_command(0xaf); //显示开
}
    
```



```
void lcd_address(uchar page,uchar column)
{
    cs1=0;
    column=column;
    page=page-1;
    transfer_command(0xb0+page);
    transfer_command(((column>>4)&0x0f)+0x10);
    transfer_command(column&0x0f);
}
```

```
void clear_screen()
{
    uchar i,j;
    for(j=0;j<16;j++)
    {
        lcd_address(j+1,0);
        for(i=0;i<128;i++)
        {
```



```
void test_screen()
{
    uchar i,j;
    for(j=0;j<16;j++)
    {
        lcd_address(j+1,0);
        for(i=0;i<256;i++)
        {
            transfer_data(0xaa);
            waitkey();
        }
    }
}
```

//显示 8x16 的点阵的字符串，括号里的参数分别为（页，列，字符串指针）

```
void display_string_8x16(uchar page,uchar column,uchar *text)
{
    uint i=0,j,k,n;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
```

```

    {
        j=text[i]-0x20;
        for(n=0;n<2;n++)
        {
            lcd_address(page+n, column);
            for(k=0;k<8;k++)
            {
                transfer_data(ascii_table_8x16[j][k+8*n]);
                transfer_data(ascii_table_8x16[j][k+8*n]);
            }
        }
        i++;
        column+=8;
    }
    else
        i++;

```

```

if(column>127)

```

```

{
    column=0;
    page+=2;
}

```

//写入一组 16x16 点阵的汉字字符串（字符串表格中需含有此字）

//括号里的参数：（页，列，汉字字符串）

```

void display_string_16x16(uchar page,uchar column,uchar *text)

```

```

{
    uchar i, j, k;
    uint address;
    j = 0;
    while(text[j] != '\0')
    {
        i=0;
        address=1;
        while(Chinese_text_16x16[i]> 0x7e )
        {
            if(Chinese_text_16x16[i] == text[j])
            {
                if(Chinese_text_16x16[i+1] == text[j+1])
                {
                    address = i*16;
                    break;
                }
            }
        }
    }
}

```



```

        i +=2;
    }
    if(column>127)
    {
        column =0;
        page +=2;
    }
    if(address !=1)
    {
        for(k=0;k<2;k++)
        {
            lcd_address(page+k, column);
            for(i=0;i<16;i++)
            {
                transfer_data(Chinese_code_16x16[address]);
                transfer_data(Chinese_code_16x16[address]);
                address++;
            }
        }
        j +=2;
    }
    else
    {
        for(k=0;k<2;k++)
        {
            lcd_address(page+k, column);
            for(i=0;i<16;i++)
            {
                transfer_data(0x00);
                transfer_data(0x00);
            }
        }
        j++;
    }
    column +=16;
}
}

```



//显示 16x16 点阵的汉字或者 ASCII 码 8x16 点阵的字符混合字符串

//括号里的参数: (页, 列, 字符串)

```
void display_string_8x16_16x16(uchar page,uchar column,uchar *text)
```

```
{
    uchar temp[3];
    uchar i=0;
    while(text[i] !='\0')
    {
```

```

if(text[i]>0x7e)
{
    temp[0]=text[i];
    temp[1]=text[i+1];
    temp[2]='\0';
    display_string_16x16(page, column, temp); //显示汉字
    column +=16;
    i +=2;
    if(column>127)
    {
        column =0;
        page +=2;
    }
}
else
{

```

//汉字为两个字节

```

    temp[0]=text[i];
    temp[1]='\0';

```

//字母占一个字节

```

    display_string_8x16(page, column, temp); //显示字母

```

```

    column +=8;

```

```

    i++;

```

```

    if(column>127)
    {

```

```

        column =0;

```

```

        page +=2;
    }
}
}

```



```

void display_32x32(uchar page, uchar column, uchar *dp)
{

```

```

    int i, j;

```

```

    for(j=0; j<4; j++)
    {

```

```

        lcd_address(page+j, column);

```

```

        for(i=0; i<32; i++)
        {

```

```

            transfer_data(*dp);

```

```

            transfer_data(*dp);

```

```

            dp++;
        }
    }
}

```



```

void display_graphic(uchar *dp)
{
    int i, j;
    for(j=0; j<16; j++)
    {
        lcd_address(j+1, 0);
        for(i=0; i<128; i++)
        {
            transfer_data(*dp);
            transfer_data(*dp);
            dp++;
        }
    }
}
    
```

```

void main(void)
    
```

```

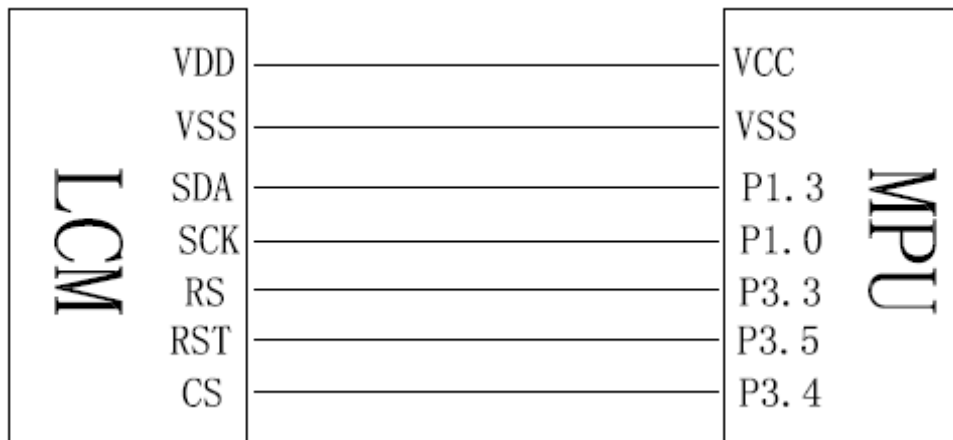
{
    initial_lcd();
    while(1)
    {
        clear_screen();
        display_graphic(bmp2);
        waitkey();
        clear_screen();
        display_32x32(1, 16, jing32);
        display_32x32(1, 48, lian32);
        display_32x32(1, 80, xun32);
        display_string_16x16(5, 1, "深圳市晶联讯电子有限公司是集研发、生产、销售于一体的从事液晶显示屏及液晶显示模块的高科技公司。");
        waitkey();
        clear_screen();

        display_string_8x16(1, 1, "0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!#$%&()*+,-./:;<=>?@[\\]^_`{|}~0123456789ABCDEFGHIJKLMNopqrstuvwxyz");
        waitkey();
        clear_screen();
        display_string_8x16_16x16(1, 1, "深圳市晶联讯电子 JLX128128G-903 128x128 点阵 视区:43.5x45.1mm 带 16x16 点阵中文字库, 或 8x16 或 5x7 点阵 ASCII 码, 四灰度级显示功能。");
        waitkey();
    }
}
    
```



### 7.5.4 串行接口

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:



### 7.5.5 以下是串行接口例程序

与并程序相比, 只需改变接口顺序及传送数据和命令子程序即可

//传送指令

```
sbit rs=P3^3;    /*接口定义:LCD 的 rs*/
sbit sclk=P1^6; /*接口定义:LCD 的 sclk*/
sbit sid=P1^7;  /*接口定义:LCD 的 sid*/
sbit reset=P3^5; /*接口定义:LCD 的 reset*/
sbit cs1=P3^4; /*接口定义:LCD 的 cs1*/
sbit key=P2^0; //P2.0 口与 GND 之间接一个按键
```

/\*写指令到 LCD 模块\*/

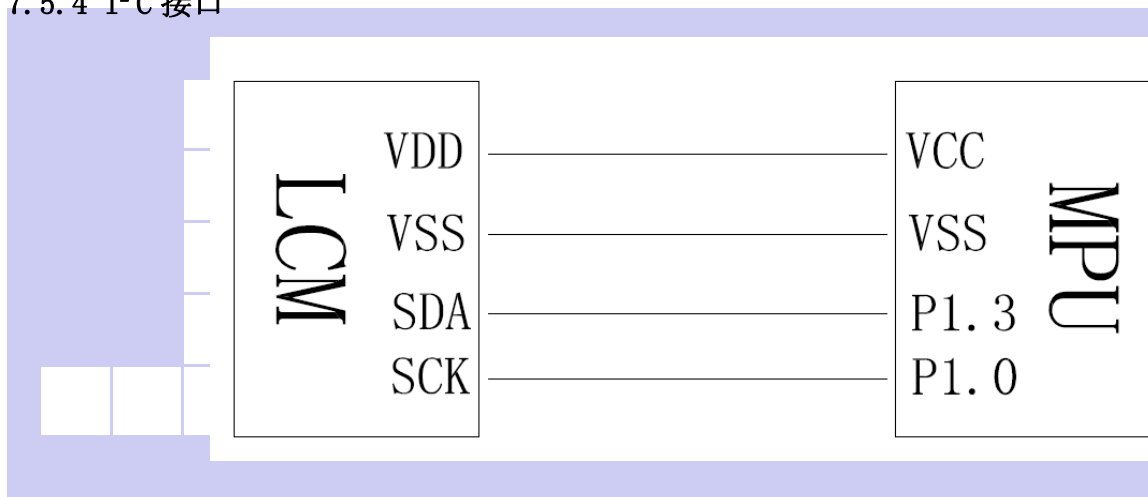
```
void transfer_command(int data1)
{
    char i;
    cs1=0;
    rs=0;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        delay_us(1);
        data1<<=1;
    }
    cs1=1;
}
```



/\*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<<=1;
    }
    cs1=1;
}
```

#### 7.5.4 I<sup>2</sup>C 接口



#### 7.5.5 以下是 I<sup>2</sup>C 接口例程序

#### 7.5.5 以下是 I<sup>2</sup>C 接口例程序

与并行程序相比，只需改变接口顺序及传送数据和命令子程序即可

//传送指令

```
sbit scl=P1^0;
sbit sda=P1^3
```

```
void transfer(int data1)
{
    int i;
    for(i=0;i<8;i++)
    {
```

```

        scl=0;
        if(data1&0x80) sda=1;
        else sda=0;
        scl=1;
        scl=0;
        data1=data1<<1;
    }

    sda=0;
    scl=1;
    scl=0;
}

```

```
void start_flag()
```

```

{
    scl=1;    /*START FLAG*/
    sda=1;    /*START FLAG*/
    sda=0;    /*START FLAG*/
}

```

```
void stop_flag()
```

```

{
    scl=1;    /*STOP FLAG*/
    sda=0;    /*STOP FLAG*/
    sda=1;    /*STOP FLAG*/
}

```

```
//写命令到液晶显示模块
```

```
void transfer_command(uchar com)
```

```

{
    start_flag();
    transfer(0x78);
    transfer(0x00);
    transfer(com);
    stop_flag();
}

```

```
//写数据到液晶显示模块
```

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

```

{
    start_flag();
    transfer(0x78);
    transfer(0x40);
    transfer(dat);
    stop_flag();
}

```

