JLX320160G-969-PL

带字库 IC 的编程说明书

目　　录

序号	内 容 标 题	页码
1	概述	2
2	字型样张：	3～4
3	外形尺寸及接口引脚功能	5～7
4	工作电路框图	7
5	指令	8～13
6	字库排置	14～16
7	点阵数据验证	16
8	附录	17
9	硬件设计及例程：	22～页末

1．概述

JLX320160G-969-PL 型液晶显示模块既可以当成普通的图像型液晶显示模块使用（即显示普通图像型的单色图片功能），又含有 JLX-GB2312-3205 字库 IC，可以从字库 IC 中读出内置的字库的点阵数据写入到 LCD 驱动 IC 中，以达到显示汉字的目的。

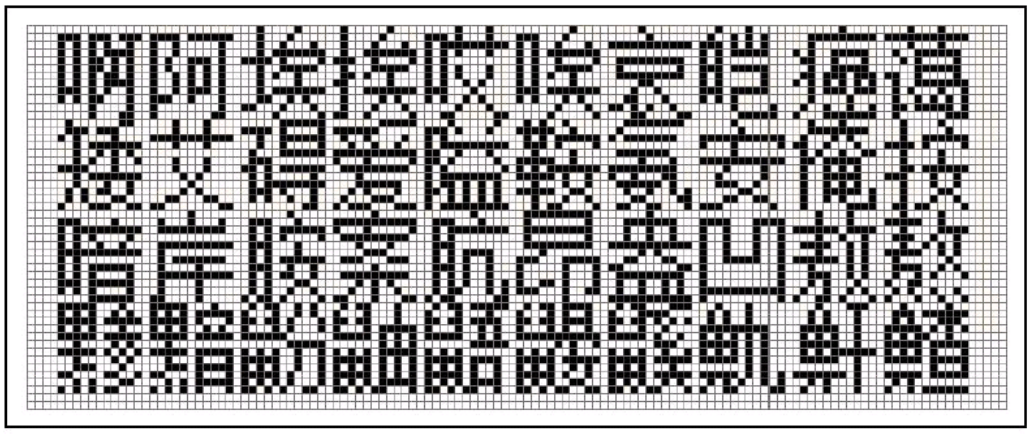
此字库 IC 存储内容如下表所述：

表 1

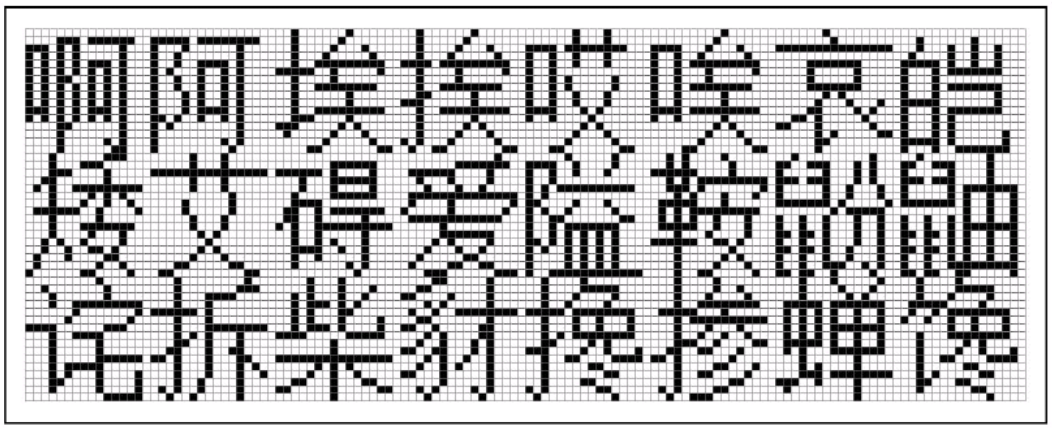
分类	字库	字号	字符数	字体	排列方式	备注
ASCII	ASCII	5x7	96	标准	W-横置横排	
	ASCII	7x8	96	标准	W-横置横排	
	ASCII	6x12	96	标准	W-横置横排	
	ASCII	8x16	96	标准	W-横置横排	
	ASCII	12x24	96	标准	W-横置横排	
	ASCII	16x32	96	标准	W-横置横排	
	ASCII	12点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	12点阵不等宽	96	Times new Roman（白正）	W-横置横排	
	ASCII	16点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	16点阵不等宽	96	Times new Roman（白正）	W-横置横排	
	ASCII	24点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	24点阵不等宽	96	Times new Roman（白正）	W-横置横排	
	ASCII	32点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	32点阵不等宽	96	Times new Roman（白正）	W-横置横排	
汉字字符集	GB2312 汉字	12x12	6763	宋体	W-横置横排	
		16x16	6763	宋体	W-横置横排	
		24x24	6763	宋体	W-横置横排	
		32x32	6763	宋体	W-横置横排	
	GB2312 字符	12x12	846	宋体	W-横置横排	
		16x16	846	宋体	W-横置横排	
		24x24	846	宋体	W-横置横排	
		32x32	846	宋体	W-横置横排	
	国标扩展字符	6x12	126	宋体	W-横置横排	
		8x16	126	宋体	W-横置横排	
		12x24	126	宋体	W-横置横排	
		16x32	126	宋体	W-横置横排	

2．字型样张：

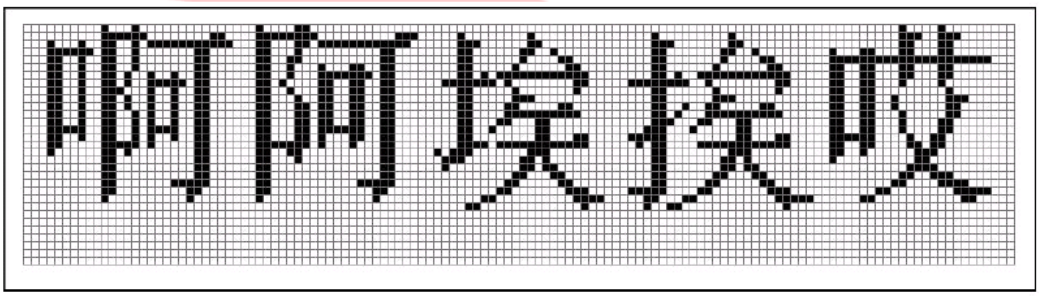
12x12 点阵 GB2312 汉字



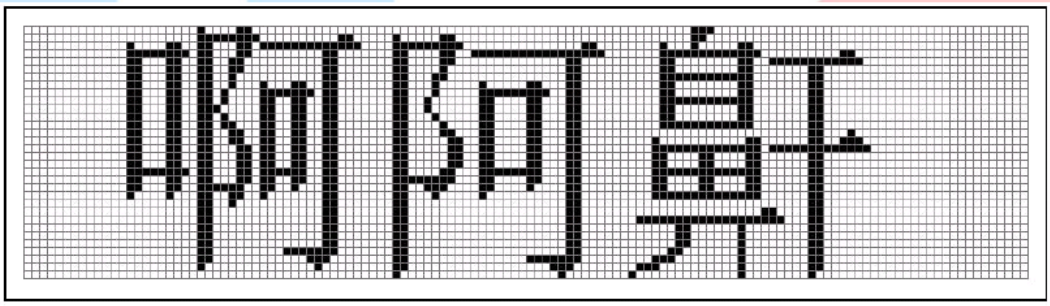
16x16 点阵 GB2312 汉字



24x24 点阵 GB2312 汉字



32x32 点阵 GB2312 汉字



12x24 点阵 ASCII 标准字符

High Bit \ Low Bit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

16x32 点阵 ASCII 标准字符

High Bit \ Low Bit	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	;	:	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

3. 外形尺寸及接口引脚功能

3.1 外形图：

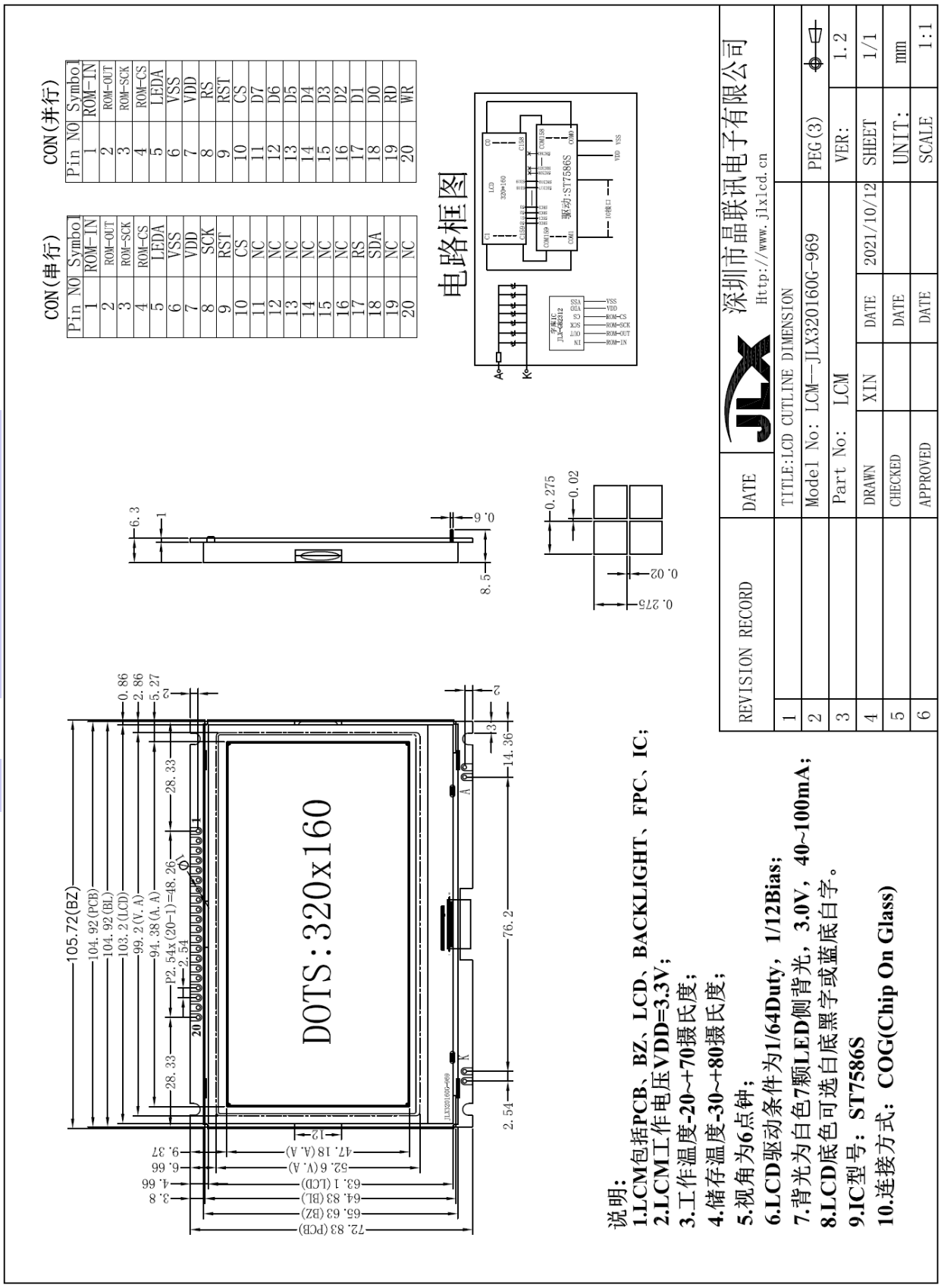


图 1. 外形尺寸

3.2.1 模块的并行接口引脚功能

引 线 号	符 号	名 称	功 能
1	ROM_IN	字库 IC 接口 SI	字库 IC:串行数据输入。不带字库时：空脚
2	ROM_OUT	字库 IC 接口 SO	字库 IC:串行数据输出。不带字库时：空脚
3	ROM_SCK	字库 IC 接口 SCLK	字库 IC:串行时钟。不带字库时：空脚
4	ROM_CS	字库 IC 接口 CS#	字库 IC:片选输入。不带字库时：空脚
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

表 2：模块并行接口引脚功能

3.2.2 串行时接口引脚功能

引 线 号	符 号	名 称	功 能
1	ROM_IN	字库 IC 接口 SI	字库 IC:串行数据输入。不带字库时：空脚
2	ROM_OUT	字库 IC 接口 SO	字库 IC:串行数据输出。不带字库时：空脚
3	ROM_SCK	字库 IC 接口 SCLK	字库 IC:串行时钟。不带字库时：空脚
4	ROM_CS	字库 IC 接口 CS#	字库 IC:片选输入。不带字库时：空脚
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	

表 3：模块串行接口引脚功能

4. 工作电路框图：

见图 2，模块由 LCD 驱动 IC ST7586S、字库 IC、背光组成。

电路框图

电路框图

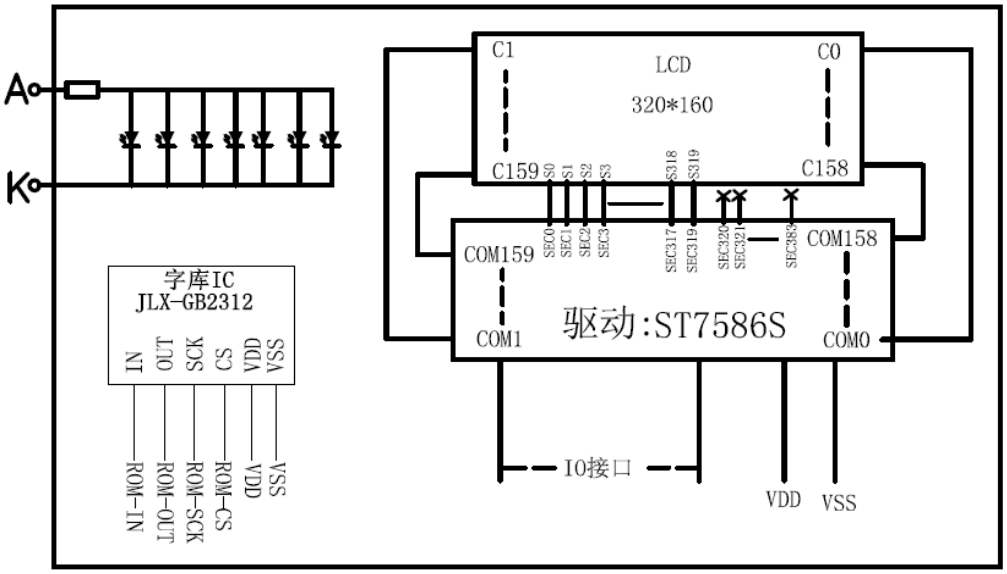


图 2:JLX320160G-969-PL 电路框图

5. 指令：

5.1 字库 IC（JLX-GB2312）指令表

Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
READ	Read Data Bytes	0000 0011	03 h	3	—	1 to ∞
FAST_READ	Read Data Bytes at Higher Speed	0000 1011	0B h	3	1	1 to ∞

所有对本芯片的操作只有2 个，那就是Read Data Bytes (READ “一般读取”)和Read Data Bytes at Higher Speed (FAST_READ “快速读取点阵数据”)。

Read Data Bytes（一般读取）：

Read Data Bytes 需要用指令码来执行每一次操作。READ 指令的时序如下(图)：

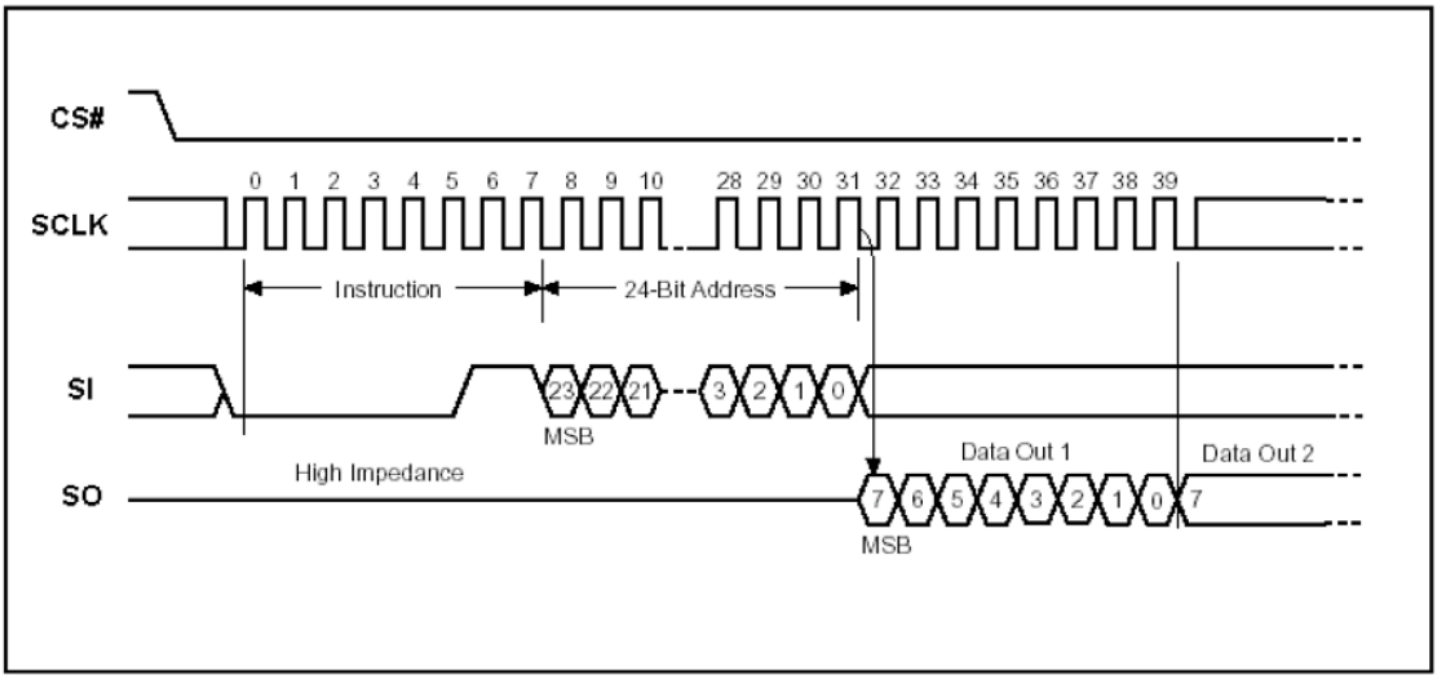
首先把片选信号（CS#）变为低，紧跟着的是1 个字节的命令字（03 h）和3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

然后该地址的字节数据通过串行数据输出引脚（SO）移位输出，每一位在串行时钟（SCLK）下降沿被移出。

读取字节数据后，则把片选信号（CS#）变为高，结束本次操作。

如果片选信号（CS#）继续保持为低，则下一个地址的字节数据继续通过串行数据输出引脚（SO）移位输出。

图：Read Data Bytes (READ) Instruction Sequence and Data-out sequence:



Read Data Bytes at Higher speed（快速读取）：

Read Data Bytes at Higher Speed 需要用指令码来执行操作。READ_FAST 指令的时序如下(图)：

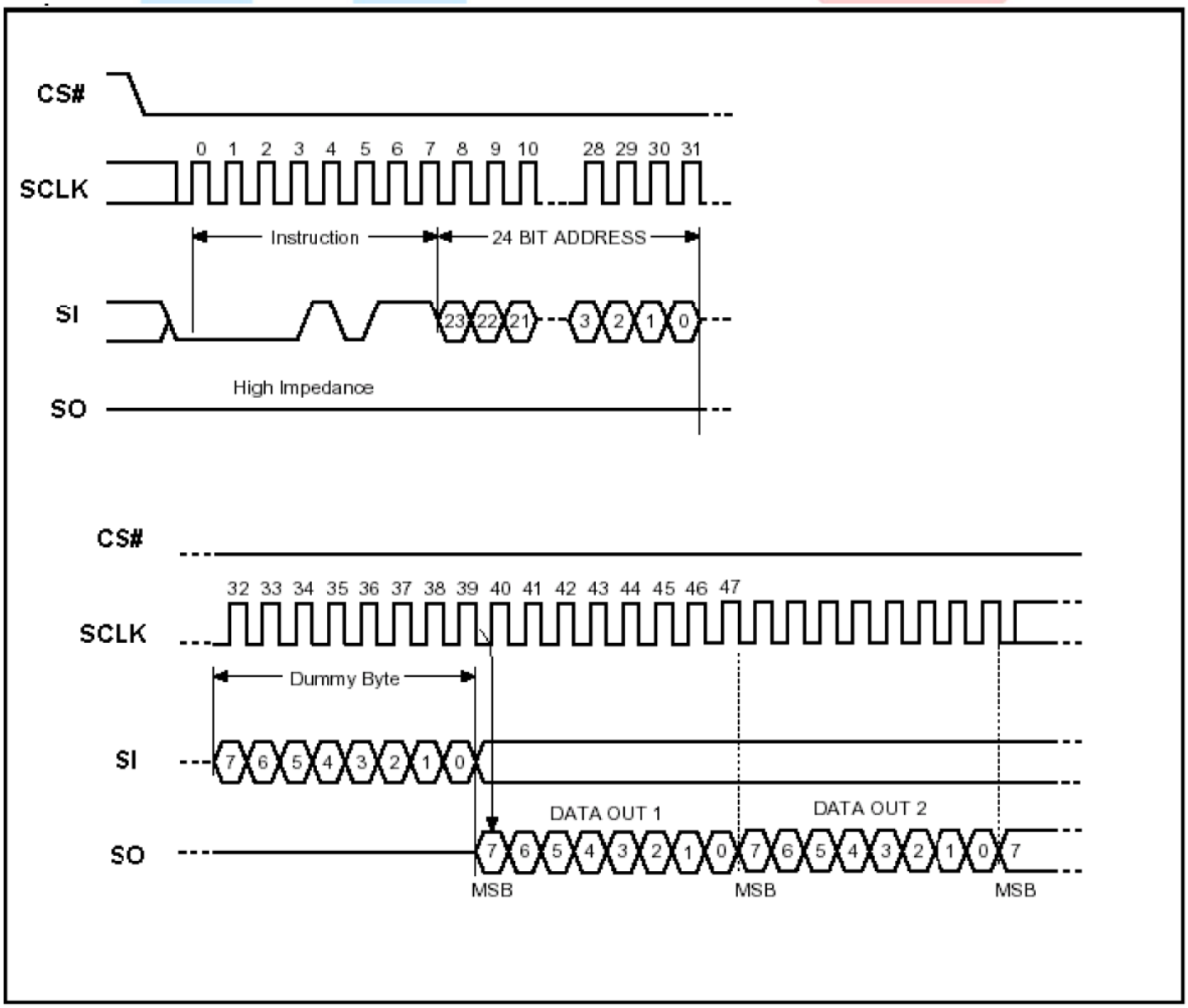
首先把片选信号（CS#）变为低，紧跟着的是1 个字节的命令字（0B h）和3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

然后该地址的字节数据通过串行数据输出引脚（SO）移位输出，每一位在串行时钟（SCLK）下降沿被移出。

如果片选信号（CS#）继续保持为底，则下一个地址的字节数据继续通过串行数据输出引脚（SO）移位输出。例：读取一个15x16 点阵汉字需要32Byte，则连续32 个字节读取后结束一个汉字的点阵数据读取操作。

如果不需要继续读取数据，则把片选信号（CS#）变为高，结束本次操作。

图：Read Data Bytes at Higher Speed (READ_FAST) Instruction Sequence and Data-out sequence:



5.2 LCD 驱动 IC 指令表

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	0	0	1	0	1	1	1	0	0	0	**读改写控制：**
(Read 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	**0xb0：** 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 (Vop Increase)	0	0	1	1	0	0	0	0	0	1	**0xc1:**Vop 增加一级
(26) 对比度VOP降低 (Vop Decrease)	0	0	1	1	0	0	0	0	1	0	**0xc2:**Vop 降低一级
(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	1	0x8f:启用自动读取控制 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	

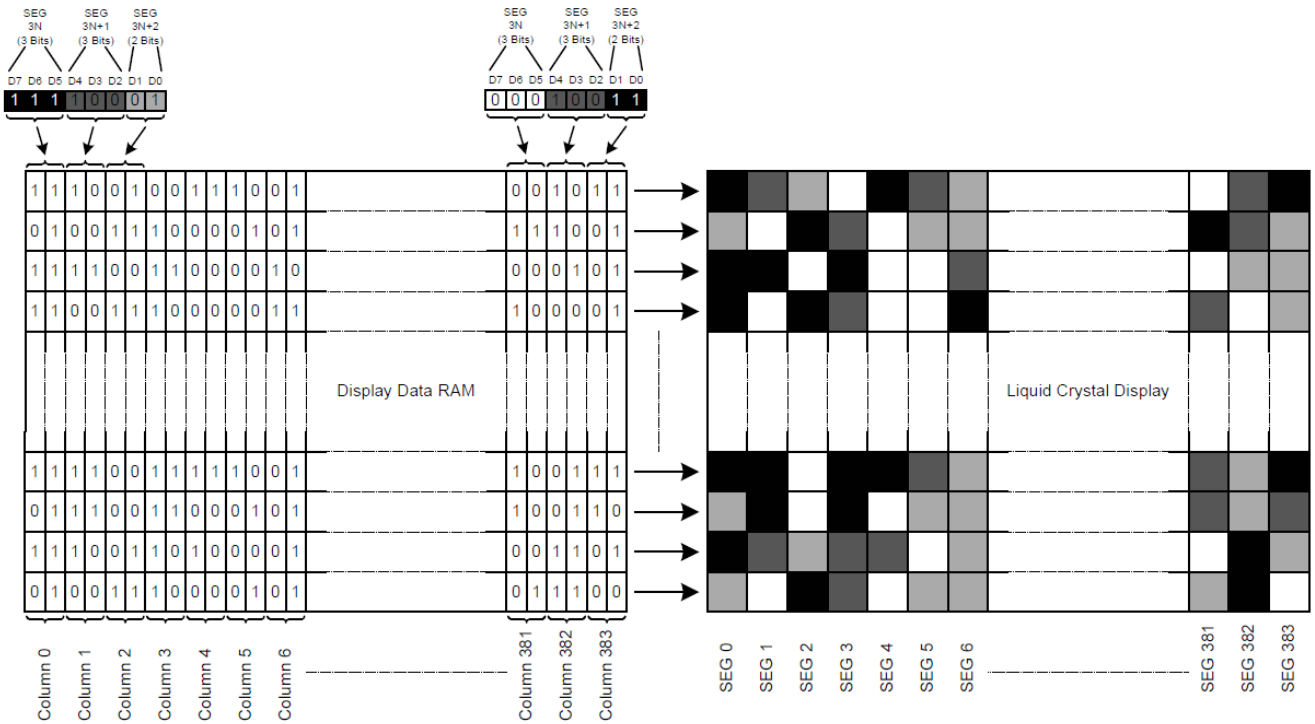
表 5. 指令表

请详细参考 IC 资料”ST7586S.PDF”。

5.3 点阵与 DD RAM 地址的对应关系

请留意页的定义：PAGE,与平时所讲的“页”并不是一个意思，在此表示 8 个行就是一个“页”，一个 320*160 点阵的屏分为 20 个“页”，从第 0“页”到第 19“页”。

DB7--DB0 的排列方向：数据是从下向上排列的。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：

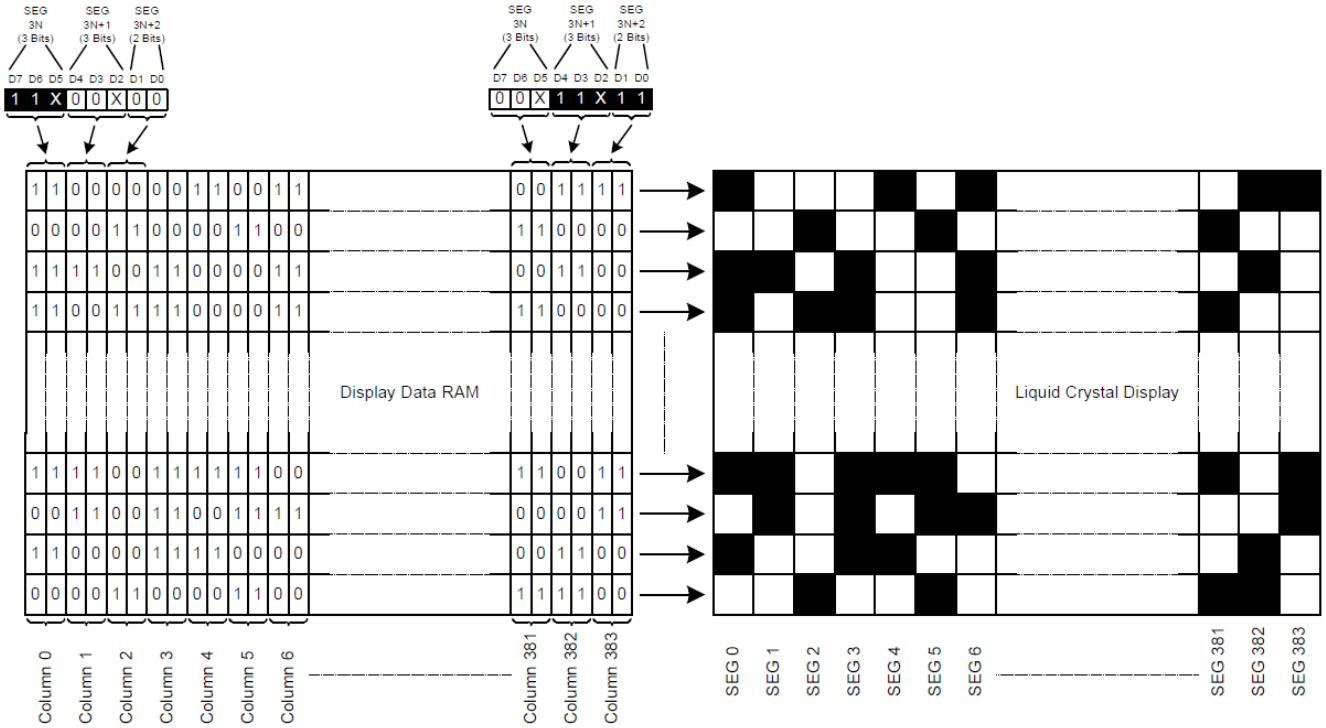


3 Bits Data			DDRAM		LCD
D7 (D4)	D6 (D3)	D5 (D2)			
1	1	1	1	1	
0	0	0	0	0	
1	0	0	1	0	
0	1	0	0	1	

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.

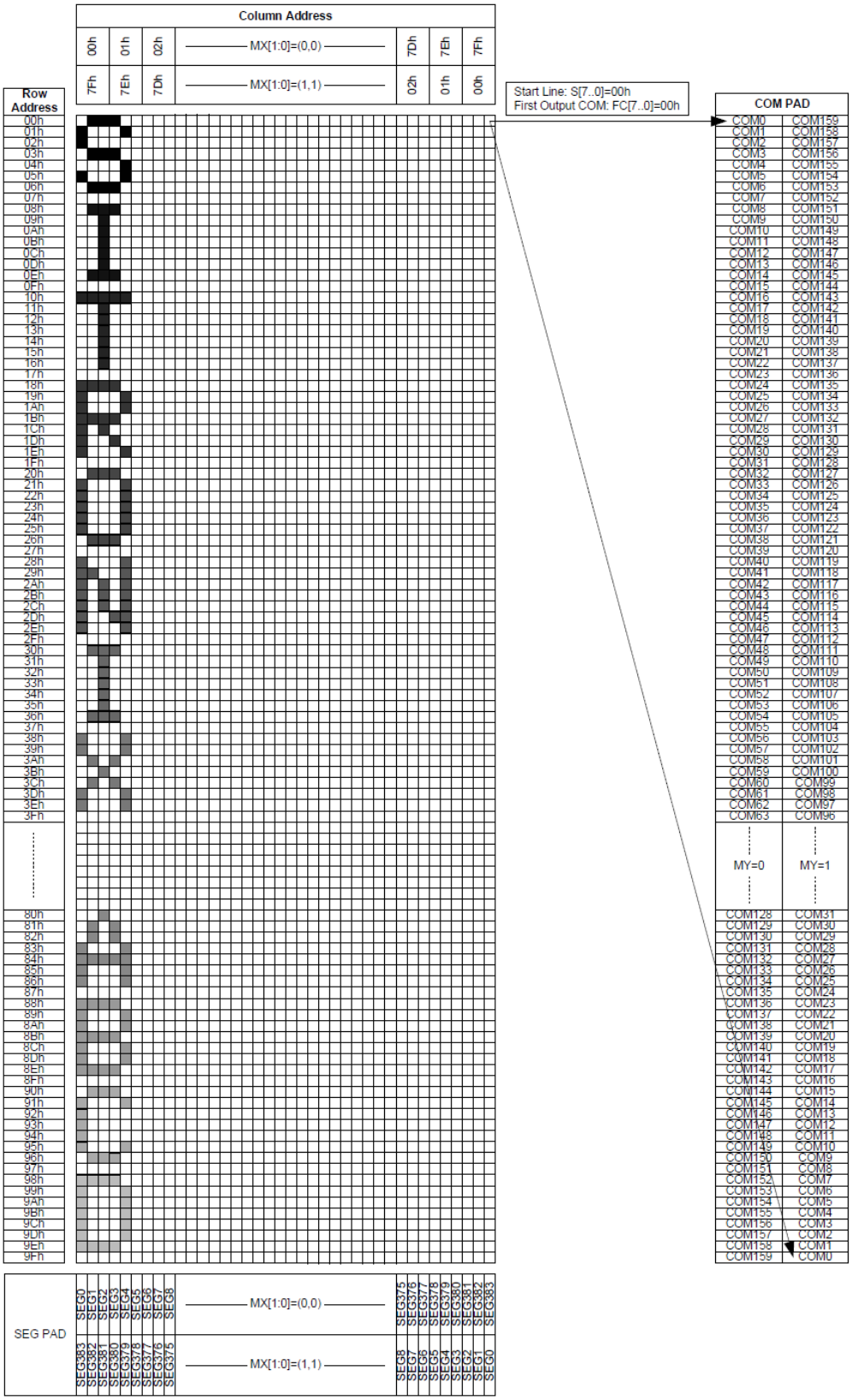


Fig. 6　　DDRAM Display Direction

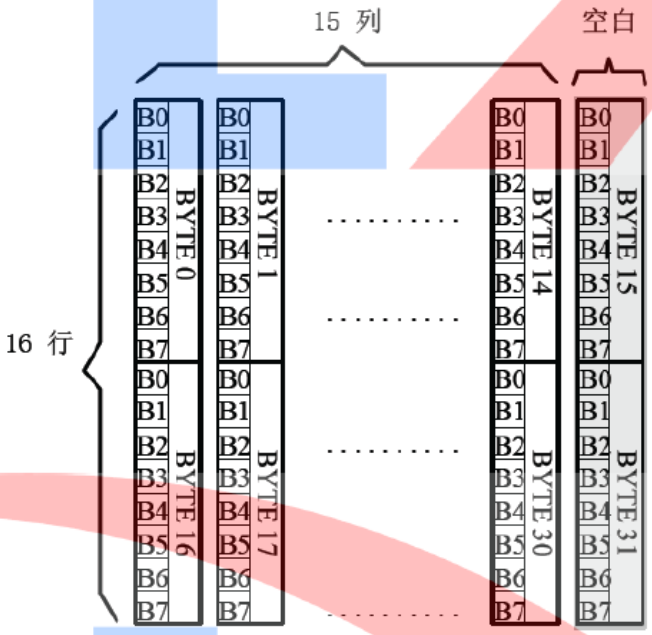
6 字库排置（竖置横排）

6.1 点阵排列格式

每个汉字在芯片中是以汉字点阵字模的形式存储的，每个点用一个二进制位表示，存 1 的点，当显示时可以在屏幕上显示亮点，存 0 的点，则在屏幕上不显示。点阵排列格式为竖置横排：即一个字节的高位表示下面的点，低位表示上面的点（如果用户按 16bit 总线宽度读取点阵数据，请注意高低字节的顺序），排满一行后再排下一行。这样把点阵信息用来直接在显示器上按上述规则显示，则将出现对应的汉字。

6.2 15X16 点阵汉字排列格式

15X16 点汉字的信息需要 32 个字节（BYTE 0 - BYTE 31）来表示。该 15X16 点汉字的点阵数据是竖置横排的，其具体排列结构如下图：

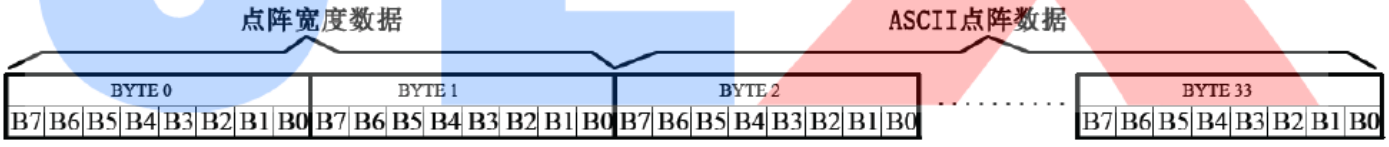


6.3 16 点阵不等宽 ASCII 方头（Arial）字符排列格式

16 点阵不等宽字符的信息需要 34 个字节（BYTE 0 - BYTE33）来表示。

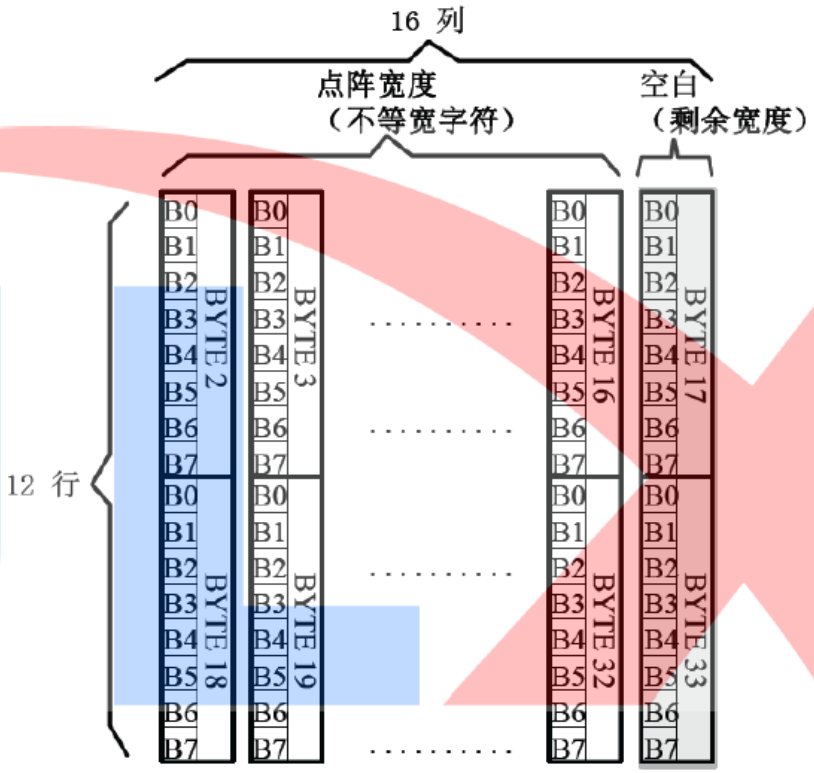
存储格式：

由于字符是不等宽的，因此在存储格式中 BYTE0~ BYTE1 存放点阵宽度数据，BYTE2-33 存放竖置横排点阵数据。具体格式见下图：



存储结构：

点阵存储宽度固定为 16，根据不同字符，其实际点阵宽度会小于 16，并会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的宽度数据，可以对还原下一个字的显示或排版留作参考。



例如：ASCII 方头字符 B

0-33BYTE 的点阵数据是:　00 0C 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 63 63 63 63 67 3E 1C 00 00 00 00 00

其中：

BYTE0~ BYTE1:　　00 0C　　为 ASCII 方头字符 B 的点阵宽度数据，即：12 位宽度。字符后面有 4 位空白区，可以在排版下一个字时考虑到这一点，将下一个字的起始位置前移。（见下图）

BYTE2-33:　00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 63 63 63 63 67 3E 1C 00 00 00 00 00　　为 ASCII 方头字符 B 的点阵数据。

7 点阵数据验证（客户参考用）

客户将芯片内“A”的数据调出与以下进行对比。若一致，表示 SPI 驱动正常工作；若不一致，请重新编写驱动。

排置：Y（竖置横排）点阵大小 8X16
字母"A"
点阵数据：00 E0 9C 82 9C E0 00 00 0F 00 00 00 00 00 0F 00

排置：W（横置横排）点阵大小 8X16
字母"A"
点阵数据：00 10 28 28 28 44 44 7C 82 82 82 82 00 00 00 00

8 附录

8.1 GB23121 区字符（846 字符）

GB2312 标准点阵字符 1 区对应码位的 A1A1~A9EF 共计 846 个字符；

GB2312　1 区

A1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A			、	。	·	ˉ	ˇ	¨	〃	々	—	～	‖	…	‘	’
B	“	”	（	）	〈	〉	《	》	「	」	『	』	〖	〗	【	】
C	±	×	÷	∶	∧	∨	∑	∏	∪	∩	∈	∷	√	⊥	∥	∠
D	⌒	⊙	∫	∮	≡	≌	≈	∽	∝	≠	≮	≯	≤	≥	∞	∵
E	∴	♂	♀	°	′	″	℃	＄	¤	￠	￡	‰	§	№	☆	★
F	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	〓	

A2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ⅰ	ⅱ	ⅲ	ⅳ	ⅴ	ⅵ	ⅶ	ⅷ	ⅸ	ⅹ					
B		1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
C	16.	17.	18.	19.	20.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
D	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	①	②	③	④	⑤	⑥	⑦
E	⑧	⑨	⑩	€		㈠	㈡	㈢	㈣	㈤	㈥	㈦	㈧	㈨	㈩	
F		Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ	Ⅺ	Ⅻ			

A3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		！	″	＃	￥	％	＆	′	（	）	＊	＋	，	－	．	／
B	０	１	２	３	４	５	６	７	８	９	：	；	＜	＝	＞	？
C	＠	Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ	Ｇ	Ｈ	Ｉ	Ｊ	Ｋ	Ｌ	Ｍ	Ｎ	Ｏ
D	Ｐ	Ｑ	Ｒ	Ｓ	Ｔ	Ｕ	Ｖ	Ｗ	Ｘ	Ｙ	Ｚ	［	＼	］	＾	＿
E	｀	ａ	ｂ	ｃ	ｄ	ｅ	ｆ	ｇ	ｈ	ｉ	ｊ	ｋ	ｌ	ｍ	ｎ	ｏ
F	ｐ	ｑ	ｒ	ｓ	ｔ	ｕ	ｖ	ｗ	ｘ	ｙ	ｚ	｛	｜	｝	￣	

GB2312　1 区

A4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		あ	あ	い	い	う	う	え	え	お	お	か	が	き	ぎ	く
B	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
C	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
D	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
E	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	わ	わ
F	ゐ	ゑ	を	ん												

A5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
B	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
C	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
D	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
E	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ワ	ワ
F	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									

A6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
B	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
C		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
D	π	ρ	σ	τ	υ	φ	χ	ψ	ω	，	。	、	：	；	！	？
E	︵	︶	︹	︺	︿	﹀	︽	︾	︻	︼	︷	︸	︹	︺	︻	︼
F	︿	﹀	｜	⋮	｜	﹁										

GB2312　1 区

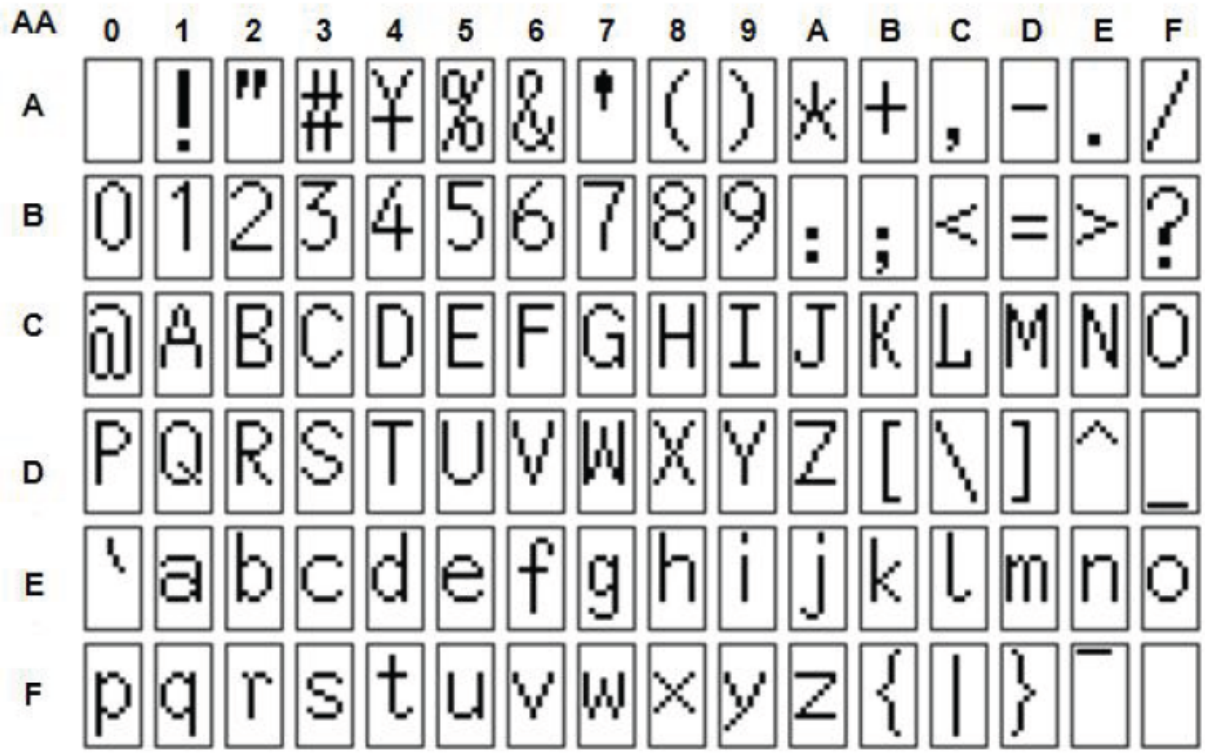
A7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
B	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
C	Ю	Я														
D		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
E	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
F	ю	я														

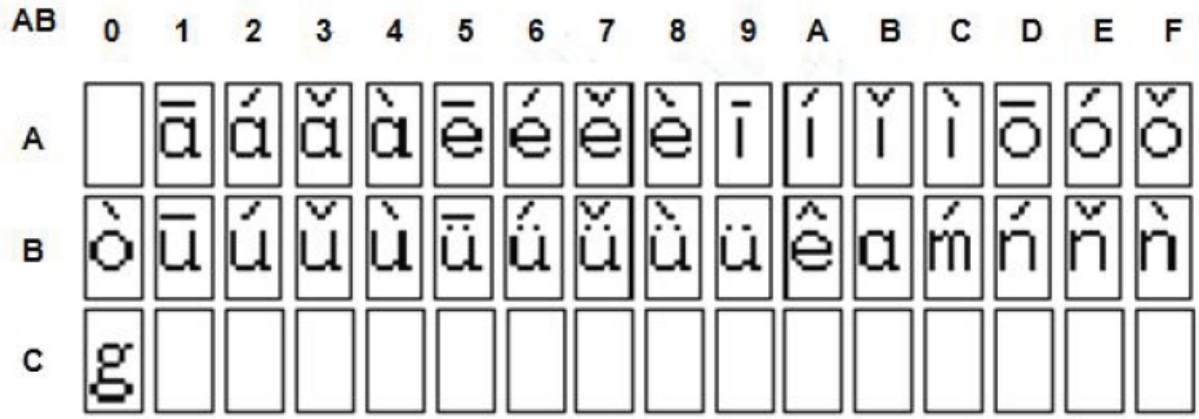
A8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ǖ	ǘ	ǚ	ǜ	ü	ê	ɑ	ḿ	ń	ň	ǹ
C	ɡ					ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ
D	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄚ	ㄛ	ㄜ	ㄝ	ㄞ	ㄟ
E	ㄠ	ㄡ	ㄢ	ㄣ	ㄤ	ㄥ	ㄦ	ㄧ	ㄨ	ㄩ						
F																

A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A					─	━	│	┃	┄	┅	┆	┇	┈	┉	┊	┋
B	┌	┌	┌	┌	┐	┐	┐	┐	└	└	└	└	┘	┘	┘	┘
C	├	├	├	├	├	├	├	├	┤	┤	┤	┤	┤	┤	┤	┤
D	┬	┬	┬	┬	┬	┬	┬	┬	┴	┴	┴	┴	┴	┴	┴	┴
E	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼	┼
F																

8.2 8x16 点国际扩展字符（126 字符）

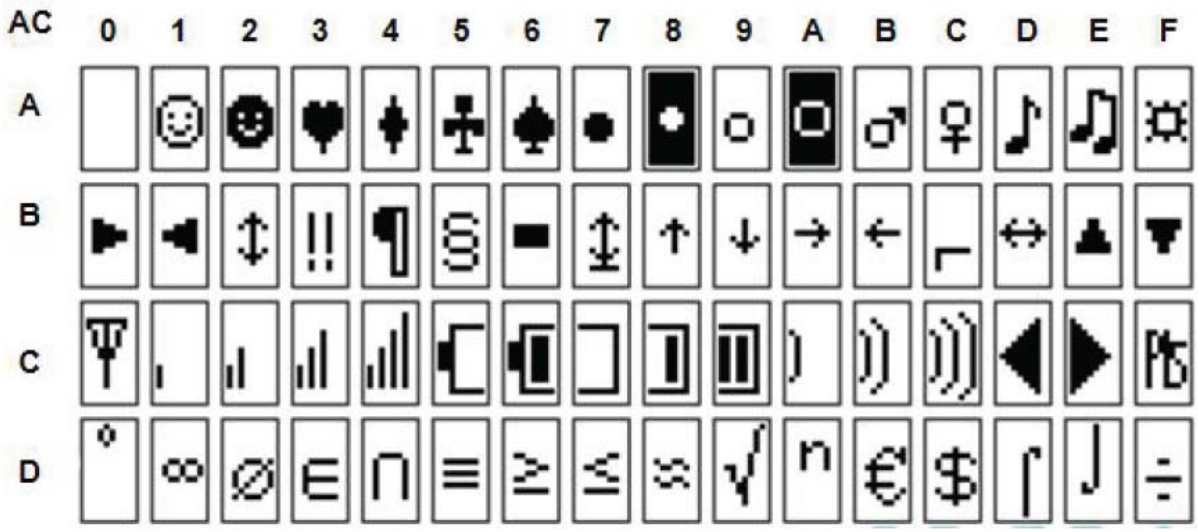
内码组成为 AAA1~ABC0 共计 126 个字符





8.3 8x16 点特殊字符（64 字符）

内码组成为 ACA1~ACDF 共计 64 个字符



9. 硬件设计及例程：

7.1 当 LCD 驱动 IC 采用并行接口方式时的硬件设计及例程：

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）

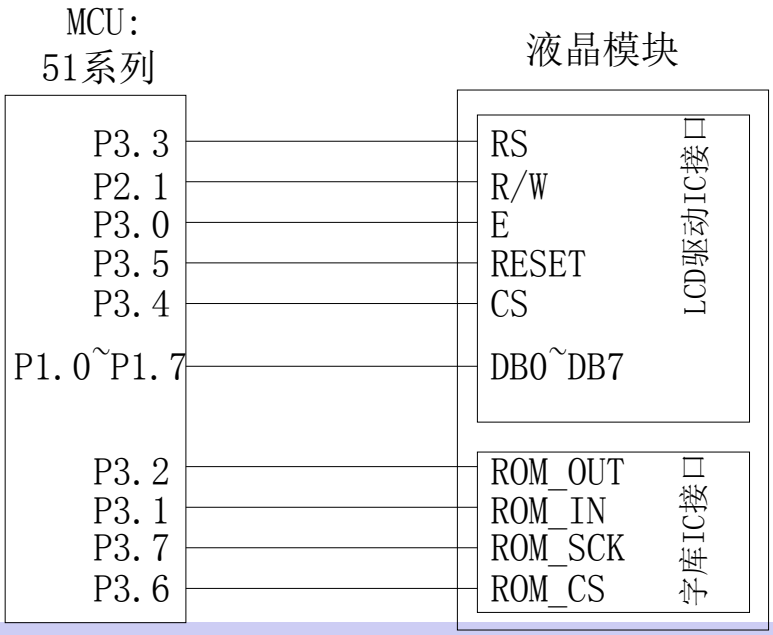
↓

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。

9.2 硬件接口： 下图为并行方式的硬件接口：



9.2 并行接口

```
/*    液晶模块型号：JLX320160G-969,
      并行接口,6800 时序,
      驱动 IC 是:ST7586S(or compatible),
      字库 IC:JLX-GB2312-3205
      可以调用字库 ROM 里的汉字：32*32 点阵，16*32 点阵，16*16 点阵，8*16 点阵，12*12 点阵，5*7 点阵等
      编写：叶建人
      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;

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#

/*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(0x4a);        // 设置 VOP 的值的低 8 位（总共 9 位）
    transfer_data_lcd(0x01);        // 设置 VOP 的值的第 9 位，也是最高一位
    transfer_command_lcd(0xC3);     // 设置 BIAS
    transfer_data_lcd(0x02);        // 00：BIAS = 1/14  02 = 1/12
    transfer_command_lcd(0xC4);     // 设置升压倍数
    transfer_data_lcd(0x07);         // 07：8 倍压

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

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


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

//  transfer_command_lcd(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);
            }
      }
}
/*显示 16*32 点阵 ASCII 码字符或等同于 8*16 点阵的图像*/
void disp_16x32(int x,int y,uchar *dp)
{
      int i,j;
      uchar dat1,dat2;

      lcd_address(x,y,16,132);

      transfer_command_lcd(0x2C);

      for(i=0;i<32;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
            }
      }
```

```
}
//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 320*160 点阵的图像
void disp_320x160(uchar *dp)
{
    int i,j;
    uchar dat1,dat2,dat3,dat4;

    lcd_address(0,0,320,160);

    transfer_command_lcd(0x2C);

    for(i=0;i<160;i++)
    {
        for(j=0;j<13;j++)//循环 13 次，每次显示 24 个点阵，合计 312 个点阵
          {
              dat1=*dp;dp++;
              dat2=*dp;dp++;
              dat3=*dp;dp++;
              transfer_mono_data_24pixel(dat1,dat2,dat3);  //每个字节显示 8 个点阵，显示 8*3=24 个点阵
          }
          dat4=*dp;dp++;
          transfer_mono_data_8pixel(dat4);//每个字节显示 8 个点阵,312+8=320 个点阵
    }
}
//显示 320*160 点阵的 4 灰度级图像
void disp_4gray_320x160(uchar *dp)
{
    uchar i,j;
    uchar dat1,dat2,dat3;
    lcd_address(0,0,320,160);  //
    transfer_command_lcd(0x2C);
    for(i=0;i<160;i++)
    {
        for(j=0;j<26;j++)//循环 26 次，每次显示 12 个点阵，合计 26*12=312 个点阵
          {
```

```
                dat1=*dp;dp++;
                dat2=*dp;dp++;
                dat3=*dp;dp++;
                transfer_gray_data_12pixel(dat1,dat2,dat3);  //每个字节显示 4 个点阵，共显示 4*3=12 个点阵
            }
            dat1=*dp;dp++;
            dat2=*dp;dp++;
            transfer_gray_data_3pixel(dat1);                 //显示 3 个点阵
            transfer_gray_data_3pixel((dat1<<6)&(dat2>>4));  //显示 3 个点阵
            transfer_gray_data_3pixel(dat2<<4);                  //显示 3 个点阵
        }
}
/****送指令到晶联讯字库 IC***/
void send_command_to_ROM( uchar datu )
{
        uchar i;
        for(i=0;i<8;i++ )
        {
            if(datu&0x80)
                Rom_IN = 1;
            else
                Rom_IN = 0;
                datu = datu<<1;
                Rom_SCK=0;
              delay_us(1);
                Rom_SCK=1;
//            delay_us(1);
        }
}

/****从晶联讯字库 IC 中取汉字或字符数据（1 个字节）***/
static uchar get_data_from_ROM( )
{
        uchar i;
        uchar ret_data=0;
        Rom_SCK=1;
        for(i=0;i<8;i++)
        {
            Rom_OUT=1;
            Rom_SCK=0;
            ret_data=ret_data<<1;
            if( Rom_OUT )
                ret_data=ret_data+1;
            else
                ret_data=ret_data+0;
            Rom_SCK=1;
        }
        return(ret_data);
}
/*从相关地址（addrHigh：地址高字节,addrMid：地址中字节,addrLow：地址低字节）中连续读出 DataLen 个字节的数据到 pBuff 的地址*/
/*连续读取*/
void get_n_bytes_data_from_ROM(uchar addrHigh,uchar addrMid,uchar addrLow,uchar *pBuff,uchar DataLen)
{
        uchar i;
        Rom_CS = 0;
        lcd_cs1=1;
        Rom_SCK=0;
        send_command_to_ROM(0x03);
        send_command_to_ROM(addrHigh);
        send_command_to_ROM(addrMid);
```

```
    send_command_to_ROM(addrLow);
    for(i = 0; i < DataLen; i++ )
        *(pBuff+i) =get_data_from_ROM();
    Rom_CS=1;
}
/***********/
ulong  fontaddr;
void disp_GB2312_32x32_string(uchar x,uchar y,uchar *text)
{
    uchar i= 0,j;
    uchar addrHigh,addrMid,addrLow ;
    uchar fontbuf[128];
    while((text[i]>0x00))
    {
        if( ((text[i]>=0xb0) && (text[i]<=0xf7) ) && (text[i+1]>=0xa1) )
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0Xedf00);

            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;

            Rom_CS = 0;
            lcd_cs1=1;
            send_command_to_ROM(0x03);
            send_command_to_ROM(addrHigh);
            send_command_to_ROM(addrMid);
            send_command_to_ROM(addrLow);

            for(j = 0; j < 128; j++ )
            {
                fontbuf[j] =get_data_from_ROM();
            }
            Rom_CS = 1;
            disp_32x32(x,y,fontbuf);
            i+=2;
            x+=11;
        }
        else if(( (text[i]>=0xa1) && (text[i]<=0xab) ) && (text[i+1]>=0xa1) )
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0Xedf00);

            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;

            Rom_CS = 0;
            lcd_cs1=1;
            Rom_SCK=0;

            send_command_to_ROM(0x03);
            send_command_to_ROM(addrHigh);
            send_command_to_ROM(addrMid);
            send_command_to_ROM(addrLow);
```

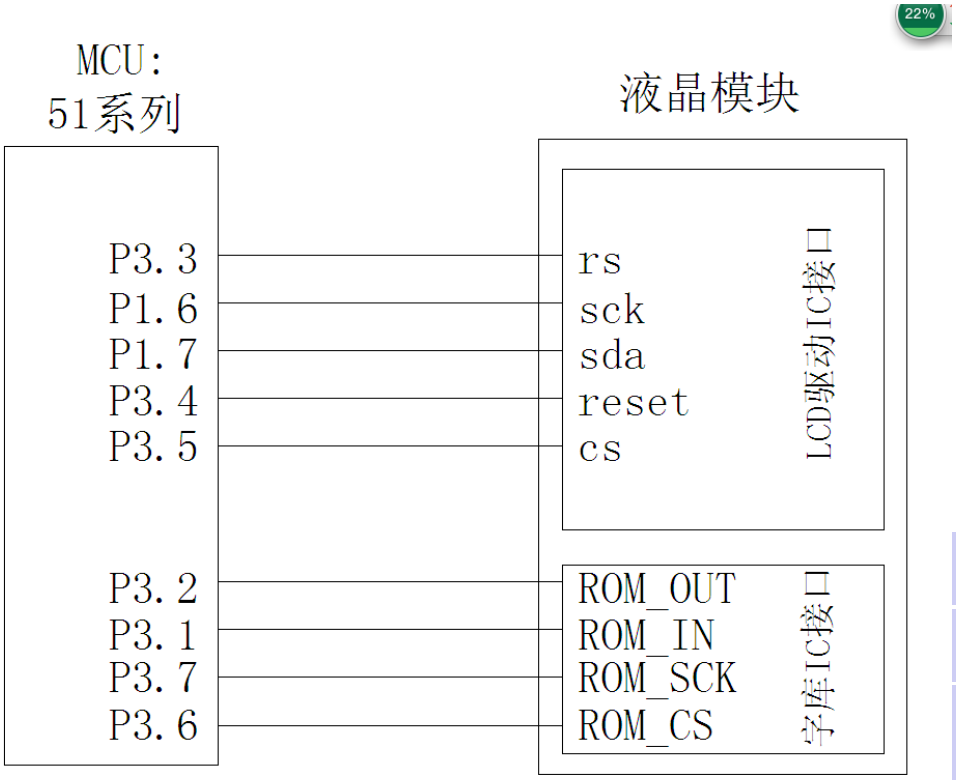
```
            for(j = 0; j < 128; j++ )
            {
                fontbuf[j] =get_data_from_ROM();
            }
            Rom_CS = 1;

            disp_32x32(x,y+11,fontbuf);
            i+=2;
            x+=11;
        }

        else if( (text[i]>=0x20) && (text[i]<=0x7e) )
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (ulong)(fontaddr*64);
            fontaddr = (ulong)(fontaddr+0x1e5a50);

            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;

            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,64);

            disp_16x32(x,y,fontbuf);
            i+=1;
            x+=6;
        }

        else
            i++;
    }
}

/*=====从字库读数据，显示 16*16 点阵的汉字或 8*16 点阵的数字========*/
void disp_GB2312_16x16_string(uchar x,uchar y,uchar *text)
{
    uchar i= 0;
    uchar addrHigh,addrMid,addrLow ;
    uchar fontbuf[64];
    ulong  fontaddr;
    while((text[i]>0x00))
    {

        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);

            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;
            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,32 );
            disp_16x16(x,y,fontbuf);
            i+=2;
            x+=5;
```

```
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xab))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);

            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;
            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,32 );
            disp_16x16(x,y,fontbuf);
            i+=2;
            x+=5;
        }
        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            uchar fontbuf[16];
            fontaddr = (text[i]- 0x20);
            fontaddr = (ulong)(fontaddr*16);
            fontaddr = (ulong)(fontaddr+0x1dd780);
            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;

            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,16 );

            disp_8x16(x,y,fontbuf);
            i+=1;
            x+=3;
        }
        else
            i++;
    }
}

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

    while(1)
    {
        initial_lcd();
        clear_screen();//清屏
        disp_GB2312_32x32_string(0,32*0,"  深圳市晶联讯电子");
        disp_GB2312_32x32_string(0,32*1,"  JLX320160G-969");
        disp_GB2312_16x16_string(0,64*1,"自带 GB2312 国标汉字库,16X16、32X32 汉字库");
        disp_GB2312_16x16_string(0,80*1,"的 ASCII 码⑴①○●◎◇◆ 1.2.‖…ABCDEFGHI");
        disp_GB2312_16x16_string(0,96*1,"鑫森淼焱晶磊众品鬑鬣麽麾麽麂麇麈麋麒麈麝磊");
        disp_GB2312_16x16_string(0,112*1,"abcdefghijklmnABCDEFGHIJKLMNOPQRSTU");
        disp_GB2312_16x16_string(0,128*1,"麟黛黜黝黠黟黢黩黜黯黯鼢黼鼯鼾鼹麝麟齄淼");
        disp_GB2312_16x16_string(0,144*1,"△▽○●◆□☆♀♂※▼▲√×★■@#【】!?&±/");
        waitkey();
        clear_screen();//清屏
        disp_GB2312_32x32_string(0,32*0,"  盼春----宁馨儿");
        disp_GB2312_32x32_string(0,32*1,"湖边寒鸟无心问柳梢,");
        disp_GB2312_32x32_string(0,32*2,"村外桃娘商略结花苞.");
```

```
        disp_GB2312_32x32_string(0,32*3,"白雪消融跳出几痕绿,");
        disp_GB2312_32x32_string(0,32*4,"谁家少女来寻荠菜挑.");
        waitkey();
        clear_screen();//清屏
        disp_320x160(bmp2);        //显示一个 320x160 点阵的图片
        waitkey();

        clear_screen();//清屏
        disp_320x160(bmp1);        //显示一个 320x160 点阵的图片
        waitkey();
        clear_screen();//清屏
        disp_320x160(bmp3);        //显示一个 320x160 点阵的图片
        waitkey();

//      clear_screen();//清屏
//      disp_4gray_320x160(bmp_4gray_1);        //显示一个 320x160 点阵的 4 灰度级的图片
//      waitkey();
        clear_screen();//清屏
        disp_4gray_320x160(bmp_4gray_2);        //显示一个 320x160 点阵的 4 灰度级的图片
        waitkey();

//      clear_screen();//清屏
//      disp_24x24(0,0,jing_24);   //在(0,0)位置显示一个 24x24 点阵的汉字或图片,三个参数分别是(x,y,24x24 点阵的指针)
//      disp_24x24(7,0,lian_24);   //在(7,0)位置显示一个 24x24 点阵的汉字或图片,三个参数分别是(x,y,24x24 点阵的指针)
//      disp_24x24(14,0,xun_24);   //在(14,0)位置显示一个 24x24 点阵的汉字或图片,三个参数分别是(x,y,24x24 点阵的指针)
//      disp_16x16(40,0,jing_16);  //在(40,0)位置显示一个 16x16 点阵的汉字或图片,三个参数分别是(x,y,16x16 点阵的指针)
//      disp_16x16(45,0,lian_16);  //在(45,0)位置显示一个 16x16 点阵的汉字或图片,三个参数分别是(x,y,16x16 点阵的指针)
//      disp_16x16(50,0,xun_16);   //在(50,0)位置显示一个 16x16 点阵的汉字或图片,三个参数分别是(x,y,16x16 点阵的指针)
//      disp_32x32(60,0,jing_32);  //在(60,0)位置显示一个 32x32 点阵的汉字或图片,三个参数分别是(x,y,32x32 点阵的指针)
//      disp_32x32(70,0,lian_32);  //在(70,0)位置显示一个 32x32 点阵的汉字或图片,三个参数分别是(x,y,32x32 点阵的指针)
//      disp_32x32(80,0,xun_32);   //在(80,0)位置显示一个 32x32 点阵的汉字或图片,三个参数分别是(x,y,32x32 点阵的指针)
//      disp_12x12(92,0,jing_12);  //在(92,0)位置显示一个 12x12 点阵的汉字或图片,三个参数分别是(x,y,12x12 点阵的指针)
//      disp_12x12(96,0,lian_12);  //在(96,0)位置显示一个 12x12 点阵的汉字或图片,三个参数分别是(x,y,12x12 点阵的指针)
//      disp_12x12(100,0,xun_12);  //在(100,0)位置显示一个 12x12 点阵的汉字或图片,三个参数分别是(x,y,12x12 点阵的指针)
//      disp_18x18(8,32,jing_18);
//      disp_21x21(15,32,jing_21);
//      disp_27x27(22,32,jing_27);
//      disp_30x30(30,32,jing_30);
//      disp_33x33(40,32,jing_33);  //
//      disp_48x48(52,32,jing_48);
//      disp_8x16(0,32,A_1);
//      display_string_8x16(0,80,"ABCDEFG!@#$%^&123");
//      display_string_12x24(0,96,"ABCDEFG!@#$%^&123");
//      waitkey();
```

9.3 当 LCD 驱动 IC 采用串行接口方式时的硬件设计及例程：

9.3.1 硬件接口：下图为串行方式的硬件接口



9.3.2、以下为串行接口方式范例程序

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：

```
#include <reg51.H>
#include <intrins.h>
#include <Ctype.h>

 sbit cs1=P3^5;          /*3.4 接口定义*/
 sbit reset=P3^4;        /*3.3 接口定义*/
 sbit rs=P3^0;           /*接口定义*/
 sbit sck=P1^6;          /*接口定义*/
 sbit sda=P1^7;          /*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*/
 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#*/
```

```
//传送指令
void transfer_command(unsigned char cmd)
{
 int k;
 cs1=0;
 rs=0;
 for (k=0;k<8;k++)
 {
      cmd=cmd<<1;
      sck=0;
      sda=0;
      sck=1;
 }
 cs1=1;
}
//传送数据
void transfer_data(unsigned char dat)
{
 unsigned char k;
 cs1=0;
 rs=1;
 for(k=0;k<8;k++)
 {
      dat=dat<<1;
      sda=1;
      sck=0;
      sck=1;
 }
 cs1=1;
 }
```

-END-