JLX320240G-905-PC

带字库 IC 的编程说明书

目 录

1．概述

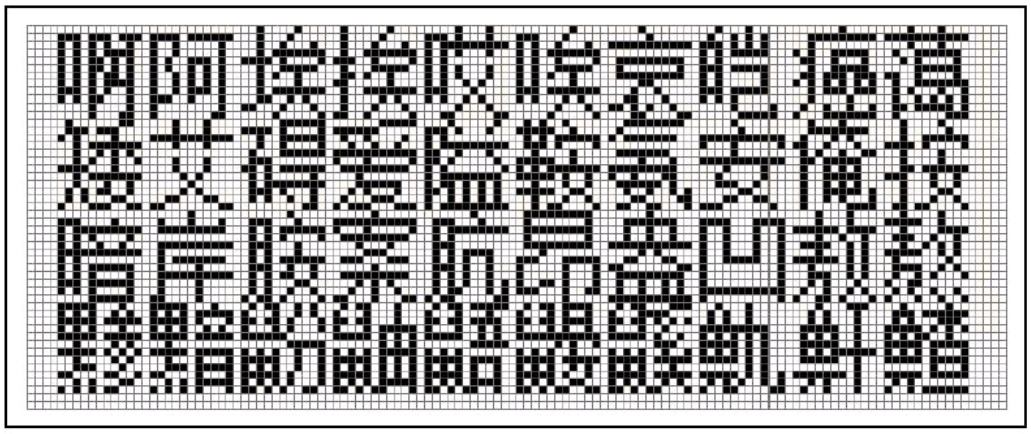
JLX320240G-905-PC 型液晶显示模块既可以当成普通的图像型液晶显示模块使用（即显示普通图像型的单色图片功能），又含有 JLX-GB2312-3207 字库 IC，可以从字库 IC 中读出内置的字库的点阵数据写入到 LCD 驱动 IC 中，以达到显示汉字的目的。

此字库 IC 存储内容如下表所述： 表 1

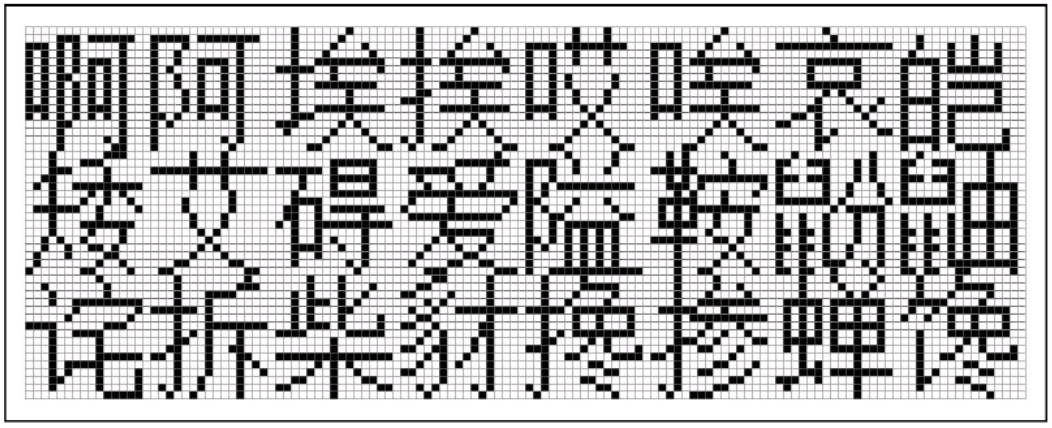
分类	字库	字号	字符数	字体	排列方式	备注
ASCII	ASCII	5x7	96	标准	W-横置横排	
	ASCII	7x8	96	标准	W-横置横排	
	ASCII	6x12	96	标准	W-横置横排	
	ASCII	8x16	96	标准	W-横置横排	
	ASCII	12x24	96	标准	W-横置横排	
	ASCII	16x32	96	标准	W-横置横排	
	ASCII	12点阵不等宽	96	Arial（方头）	W-横置横排	
	ASCII	12点阵不等宽	96	Times new Roman（白正）	W-横置横排	
	ASCII	16点阵不等宽	96	Arial（方头）	W-横置横排	
	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：

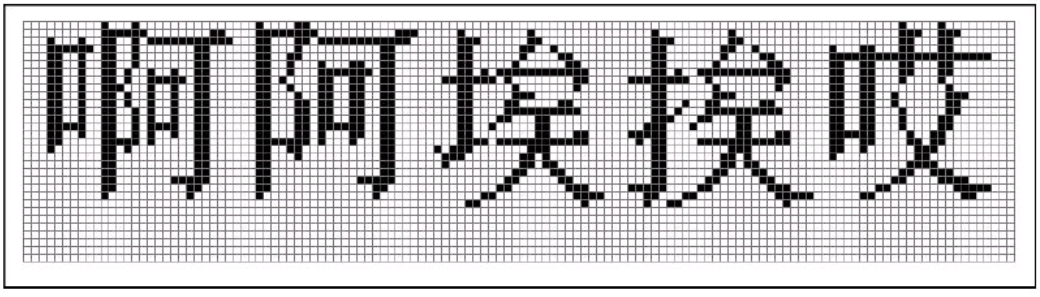
12x12 点阵 GB2312 汉字



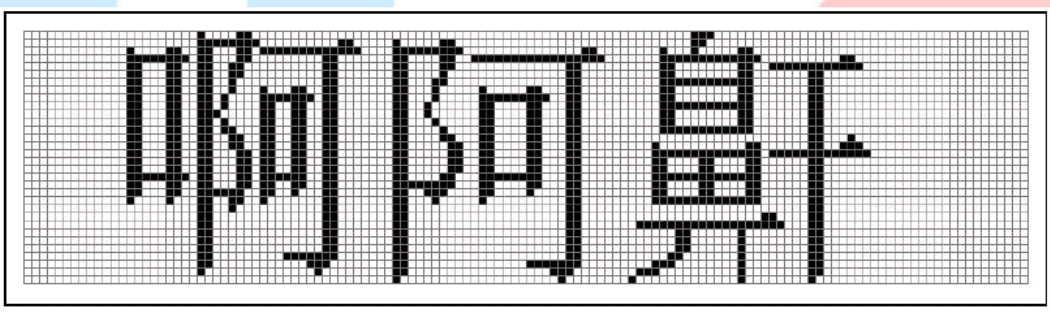
16x16 点阵 GB2312 汉字



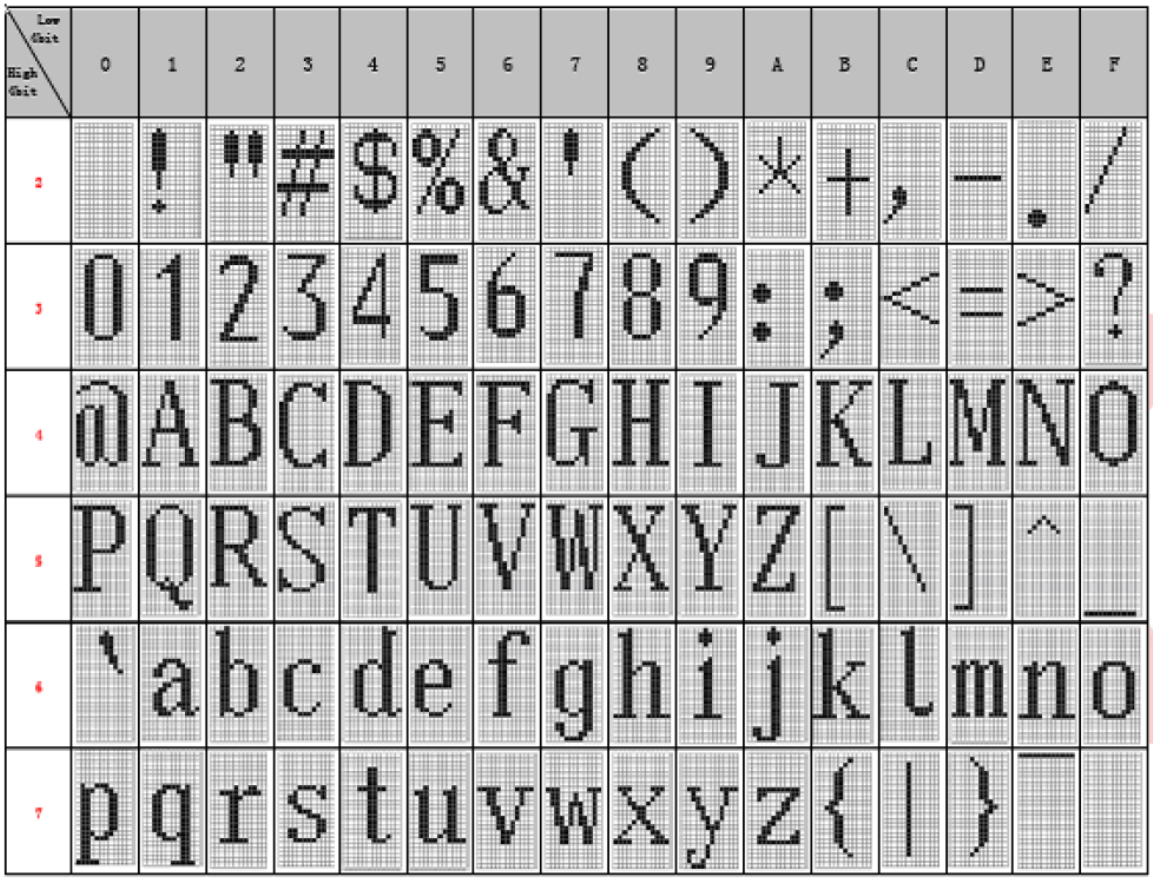
24x24 点阵 GB2312 汉字



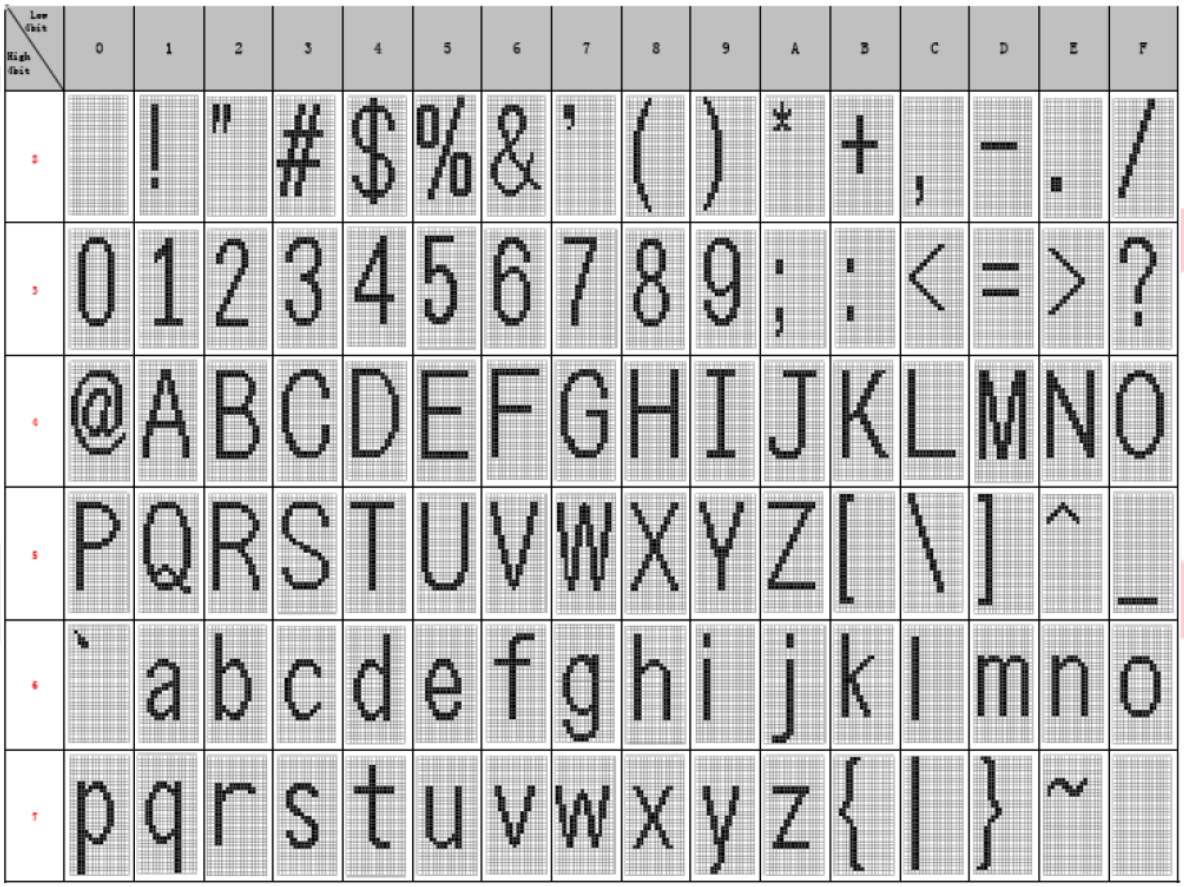
32x32 点阵 GB2312 汉字



12x24 点阵 ASCII 标准字符



16x32 点阵 ASCII 标准字符



3. 外形尺寸及接口引脚功能

3.1 外形图：

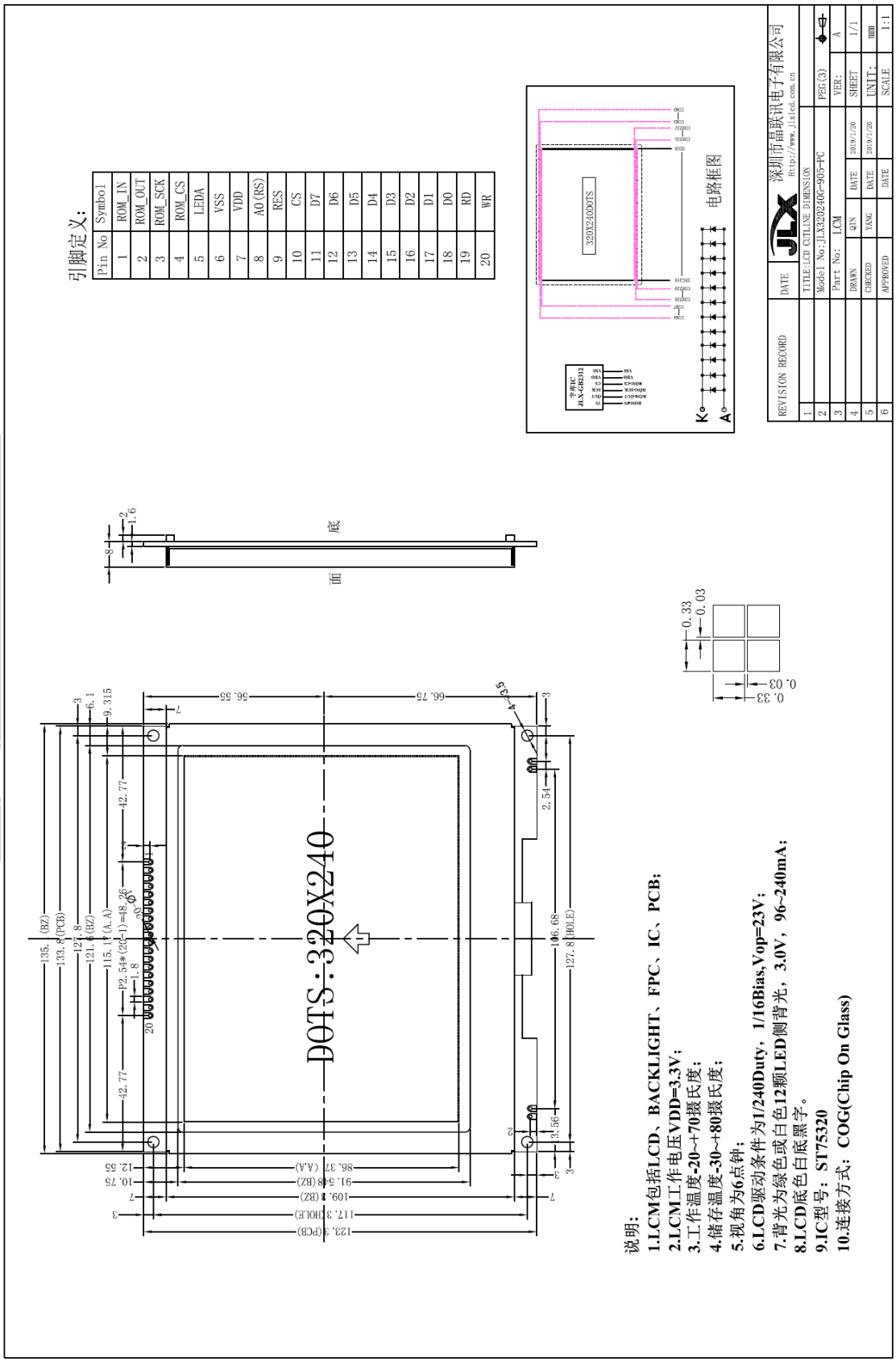


图 1. 外形尺寸

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	串行数据输出	详见字库 IC:JLX-GB2312-3207 说明书：ROM-IN 对应字库 IC 接口 SI，ROM-OUT 对应 SO， ROM-SCK 对应 SCLK，ROM-CS 对应 CS#
2	ROM-OUT	字库 IC 接口 SO	串行数据输入	
3	ROM-SCK	字库 IC 接口 SCLK	串行时钟输入	
4	ROM-CS	字库 IC 接口 CS#	片选输入	
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 可选	
8	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”A0”）	
9	RES	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	
11～18	D7-D0	I/O	数据总线 DB7-DB0	
19	E	使能信号	并行时：使能信号	
20	R/W	读/写	并行时：H:读数据 0:写数据	

表 1：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	串行数据输出	详见字库 IC:JLX-GB2312-3207 说明书：ROM-IN 对应字库 IC 接口 SI，ROM-OUT 对应 SO， ROM-SCK 对应 SCLK，ROM-CS 对应 CS#
2	ROM-OUT	字库 IC 接口 SO	串行数据输入	
3	ROM-SCK	字库 IC 接口 SCLK	串行时钟输入	
4	ROM-CS	字库 IC 接口 CS#	片选输入	
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 可选	
8	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”A0”）	
9	RES	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	
11-14	D7-D4	I/O	串行接口，D7-D4 引脚接 VDD	
15-17	D3-D1(SDA)	I/O	串行数据（D1、D2、D3 接一起作为 SDA）	
18	D0(SCK)	I/O	串行时钟	
19	E	使能信号	串行接口，此引脚接 VDD	
20	R/W	读/写	串行接口，此引脚接 VDD	

表 2：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 ST75320、字库 IC、背光组成。

电路框图

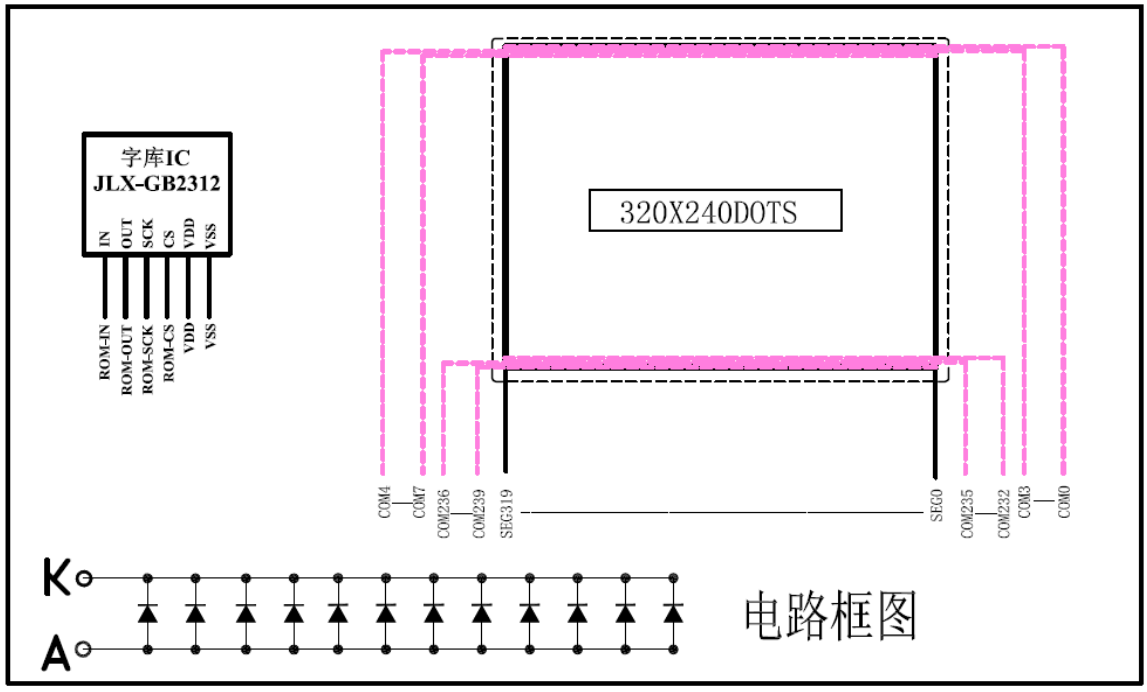


图 2:JLX320240G-905-PC 电路框图

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)：

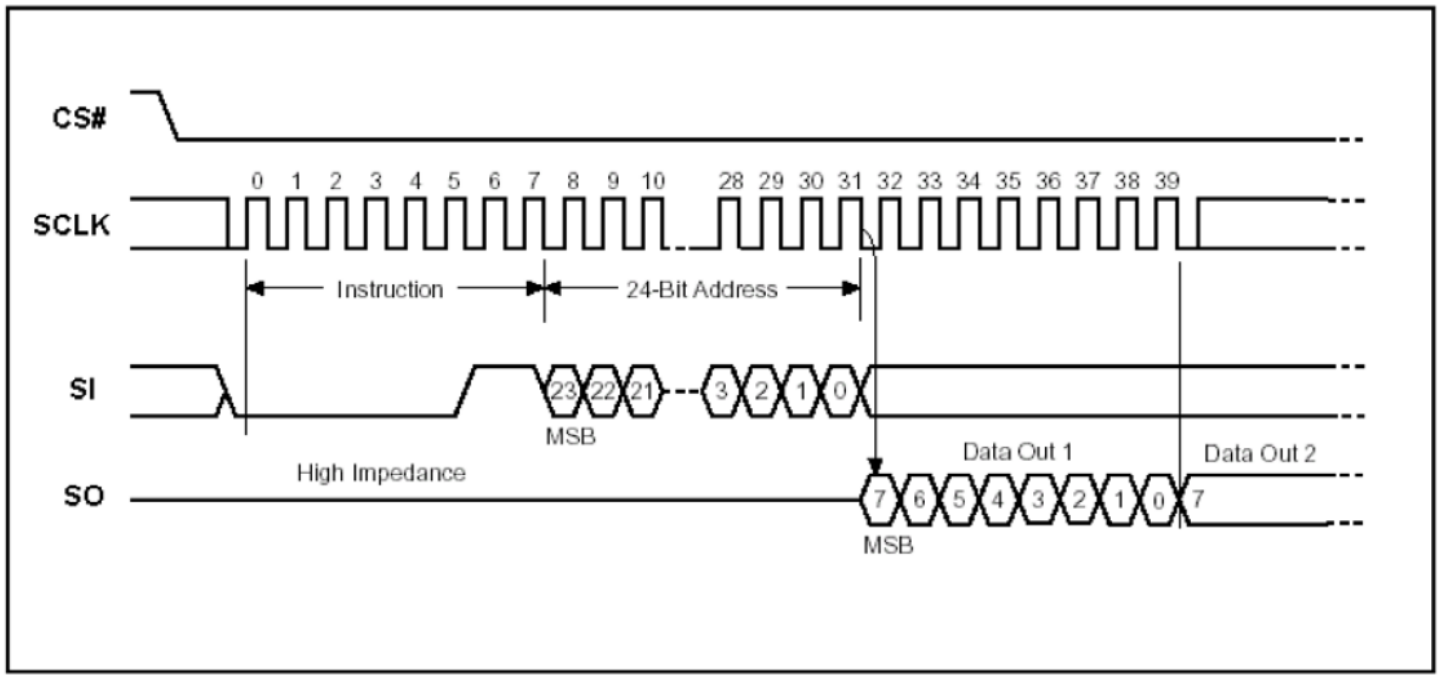
首先把片选信号（CS#）变为低，紧跟着的是1 个字节的命令字（03 h）和3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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)：

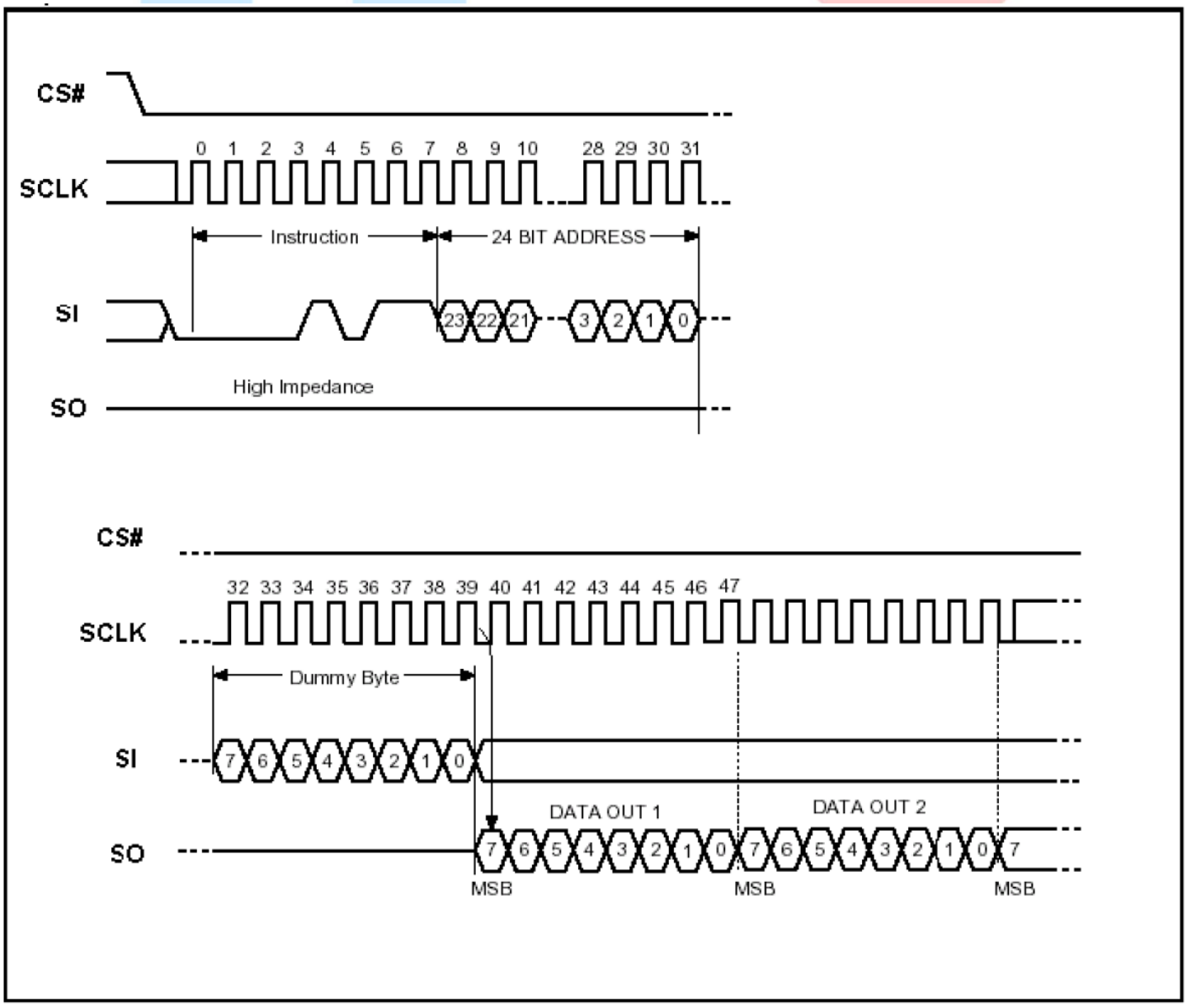
首先把片选信号（CS#）变为低，紧跟着的是1 个字节的命令字（0B h）和3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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										
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(1)显示开/关 （display on/off）	0	0	1	0	1	0	1	1	1	0 1	显示开/关: 0XAE:关，0XAF：开
(2)正显/反显 （Inverse Display）	0	0	1	0	1	0	0	1	1	0 1	显示正显/反显 0XA6:正显，正常 0XA7：反显
(3)所有点阵开/关 (All Pixel ON/OFF)	0	0	1	0	1	0	0	0	1	0 1	0XA2：所有点阵关 0XA3：所有点阵开
(4) COM输出控制 (COM Output Status)	0 1	0 0	1 –	1 –	0 –	0 –	0 –	1 0	0 SCAN	0 MY	设置 COM 扫描模式,0xc4 SCAN =0;正常扫描 SCAN =1；隔行扫描 MY=0；COM0→COM239 MY=0；COM239→COM0
(5) 显示起始行 (Display Start Line)	0 1	0 0	1 –	0 –	0 S5	0 S4	1 S3	0 S2	1 S1	0 S0	设置显示开始行,0x8a
(6)页地址设置 (Set Page Address)	0 1	0 0	1 –	0 –	1 Y5	1 Y4	0 Y3	0 Y2	0 Y1	1 Y0	0Xb1：页地址设置 0X00：起始页地址
(7)列地址设置 (Set Column Address)	0 1 1	0 0 0	0 – X7	0 – X6	0 – X5	1 – X4	0 – X3	0 – X2	1 – X1	1 X8 X0	0X13：列地址设置 0X00：起始列地址 0X00：起始列地址
(8) 写数据到晶液屏 (Display Data Write)	0 1	0 0	0 D7	0 D6	0 D5	1 D4	1 D3	1 D2	0 D1	1 D0	0X1d：写显示数据 DDRAM 8 位显示数据
(9) 读液晶屏显示数据 (Display Data Read)	0 1	0 1	0 D7	0 D6	0 D5	1 D4	1 D3	1 D2	0 D1	0 D0	0X1C：读数据 8 位显示数据
(10)液晶屏扫描方向	0	0	1	0	0	0	0	1	0	DIR	DIR=0;列扫描 DIR=1;行扫描
(11)列扫描方向 (Column Address Direction)	0	0	1	0	1	0	0	0	0	MX	列扫描方向：0xa0 或 0xa1 MX=0：Column0—> Column 319 MX=0：Column 319—> Column0
(12) N行反显 (N-Line Inversion)	0 1	0 0	0 –	0 –	1 NL5	1 NL4	0 NL3	1 NL2	1 NL1	0 NL0	0X36：N 行反显
(13) N行反显开或关	0	0	1	1	1	0	0	1	0	NL	NL=0，0xe4: N 行反显关 NL=1，0xe5: N 行反显开
(14)显示区域 (Display Area)	0 1 1	0 0 0	0 – –	1 – –	1 – SP5	0 – SP4	1 – SP3	1 DTY2 SP2	0 DTY1 SP1	1 DTY0 SP0	0X6C：设置显示区域 DTY[2:0]=0x00～0x07 SP[5:0]=0x00～0x4f
(15)读改写	0	0	1	1	1	0	0	0	0	0	0XE0：进入读改写模式
(16)结束读改写	0	0	1	1	1	0	1	1	1	0	0XEE：结束读改写模式
(17)振荡电路开或关	0	0	1	0	1	0	1	0	1	OSC	内部振荡电路开或关 0XAA：振荡电路关 0XAB：振荡电路开
(18) 工作时钟频率 (Operation Clock Frequency)	0 1 1	0 0 0	0 FRB3 FRD3	1 FRB2 FRD3	0 FRB1 FRD3	1 FRB0 FRD3	1 FRA3 FRD3	1 FRA3 FRC3	1 FRA3 FRC3	1 FRA3 FRC3	0X5F：设置不同温度范围内的帧率
(19)电源控制	0	0	0	0	1	0	0	1	0	1	0X26：设置内建电路开或关

(Power Control)	1	0	–	VOUT	VAD	V3	VPF	VMV3	VMAD	VNF	
(20) 帧率水平	0	0	0	0	1	0	1	0	1	1	0X29：设置帧速率的级别
	1	0	–	–	–	–	–	–	–	DBL	
(21) LCD偏压比设置	0	0	1	0	1	0	0	0	1	0	0XA2: LCD 偏压比设置
BIAS	1	0	–	–	–	–	BS3	BS2	BS1	BS0	
(22)液晶内部电压设置	0	0	1	0	0	0	0	0	0	1	0X81:设置对比度
(Electronic Volume)	1	0	EV7	EV6	EV5	EV4	EV3	EV2	EV1	EV0	微调对比度, 范围 0X00-0XFF
	1	0	–	–	–	–	–	–	EV9	EV8	粗调对比度, 范围 0X00-0X03
(23) 电源放电	0	0	1	1	1	0	1	0	1	0	0Xe9:
(Power Discharge)											
(24)睡眠模式	0	0	1	0	1	0	1	0	0	PD	PD=0, 0XA8：工作模式
(Power Save)											PD=1, 0XA9：睡眠模工
(25) 温度梯度补偿	0	0	0	1	0	0	1	1	1	0	0X4F: 设置温度梯度补偿
(Temperature Gradient	1	0	TM1[3:0]				TM0[3:0]				
Compensation)	1	0	TM3[3:0]				TM2[3:0]				
	1	0	TM5[3:0]				TM4[3:0]				
	1	0	TM7[3:0]				TM6[3:0]				
	1	0	TM9[3:0]				TM8[3:0]				
	1	0	TMB[3:0]				TMA[3:0]				
	1	0	TMD[3:0]				TMC[3:0]				
	1	0	TMF[3:0]				TME[3:0]				
(26)读状态	0	0	1	0	0	0	1	1	1	0	0X80：读 IC 状态
(Read Status)	0	1	D	OSC	AVD	V3	VFP	VMV3	VNAD	VFN	
	0	1	DISV	ITR	MY	PD	TD	NLFR	MLS	–	
(27)温度检测	0	0	0	1	1	0	1	0	0	TD	0X68：不使能
(Temperature Detection)											0X69：使能
(28)LCD驱动模式	0	0	1	1	1	0	0	1	1	1	0XE7：LCD 驱动模式设置
(LCD Driving method)			0	0	0	NLFR	1	0	0	1	
(29)空操作	0	0	1	1	1	0	0	0	1	1	0XE3：空操作
(30) 频率补偿温度范围	0	0	1	1	1	0	1	1	0	0	0XEC: 频率补偿温度范围设置
(Frequency	1	0	–	TA6	TA5	TA4	TA3	TA2	TA1	TA0	
Compensation	1	0	–	TB6	TB5	TB4	TB3	TB2	TB1	TB0	
Tenperature Range)	1	0	–	TC6	TC5	TC4	TC3	TC2	TC1	TC0	
(31) 温度滞后值	0	0	1	1	1	0	1	1	0	1	0XEC：设置温度滞后值
(Tenperature	1	0	–	–	THV5	THV4	THV3	THV2	THV1	THV0	
Hysteresis value)	1	0	0	0	0	0	THF3	THF2	THF1	THF0	
(32) 当前温度数据	0	0	1	1	1	0	1	1	1	1	0XEF: 监控当前温度
(Current Temperature	0	1	T7	T6	T5	T4	T3	T2	T1	T0	
Data)											
(33)读ID	0	0	1	0	0	0	1	1	1	1	0X8F：读 ID 值
	0	1	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	
(34)测试(Test)	0	0	1	1	1	1	1	1	TE	T	此功能不用

表 5. 指令表

请详细参考 IC 资料”ST75320.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*240 点阵的屏分为 30 个“页”，从第 0“页”到第 29“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：

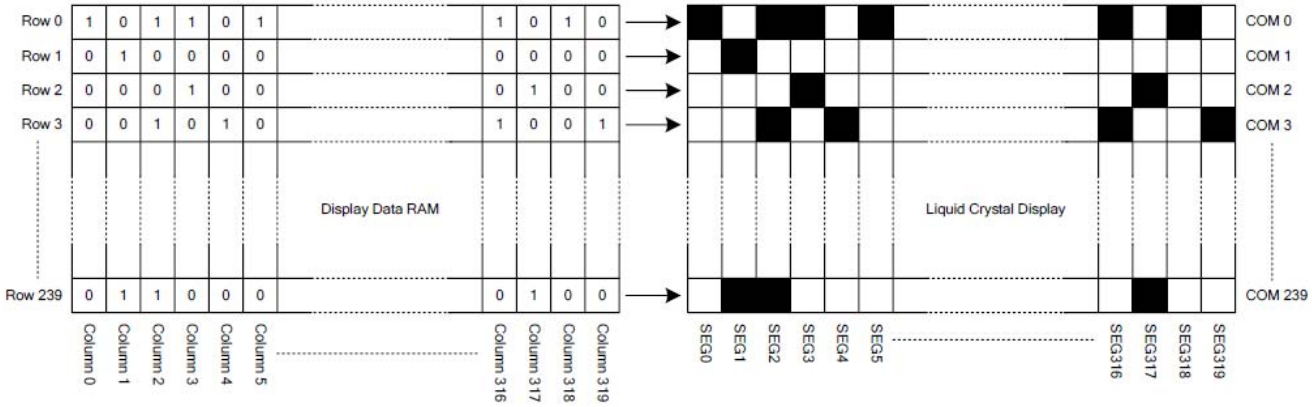


Figure 15 DDRAM Mapping

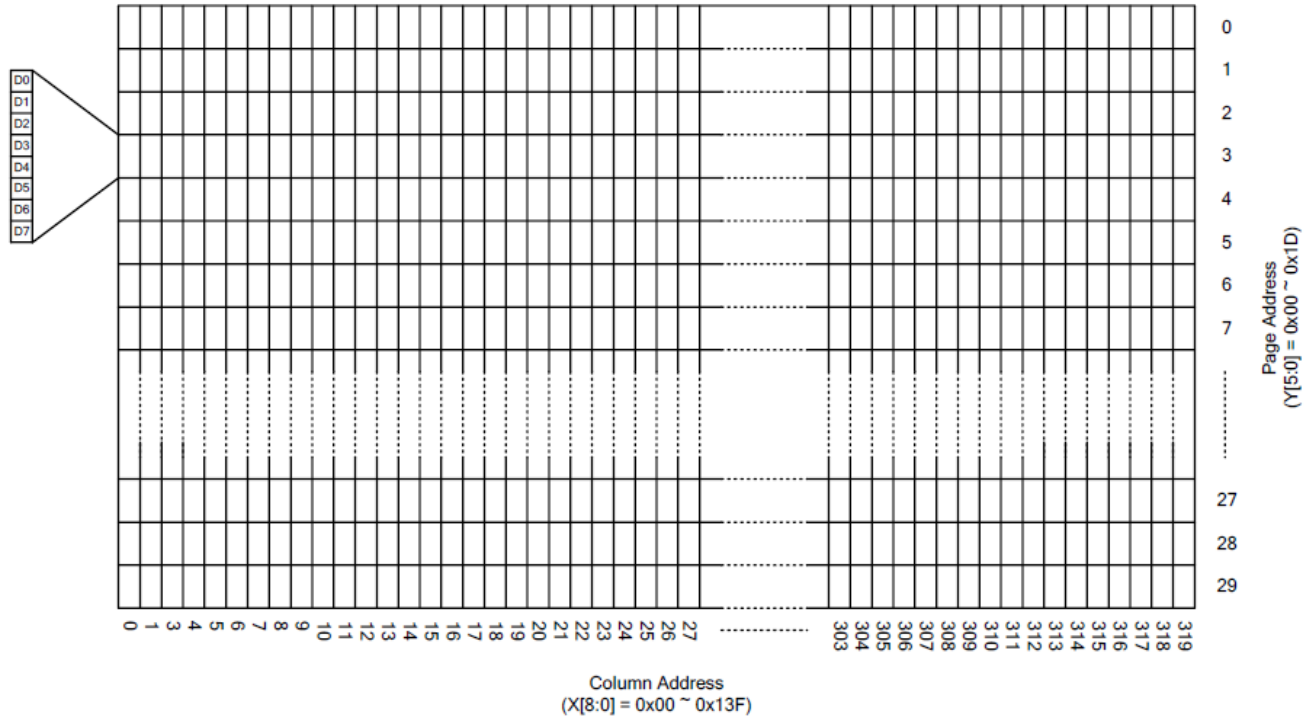


Figure 16 DDRAM Format

下图摘自 ST75320 IC 资料，可通过“ST75320.PDF”之第 29 页获取最佳效果。

The relation between DDRAM and outputs with different MX or MY setting is shown below.

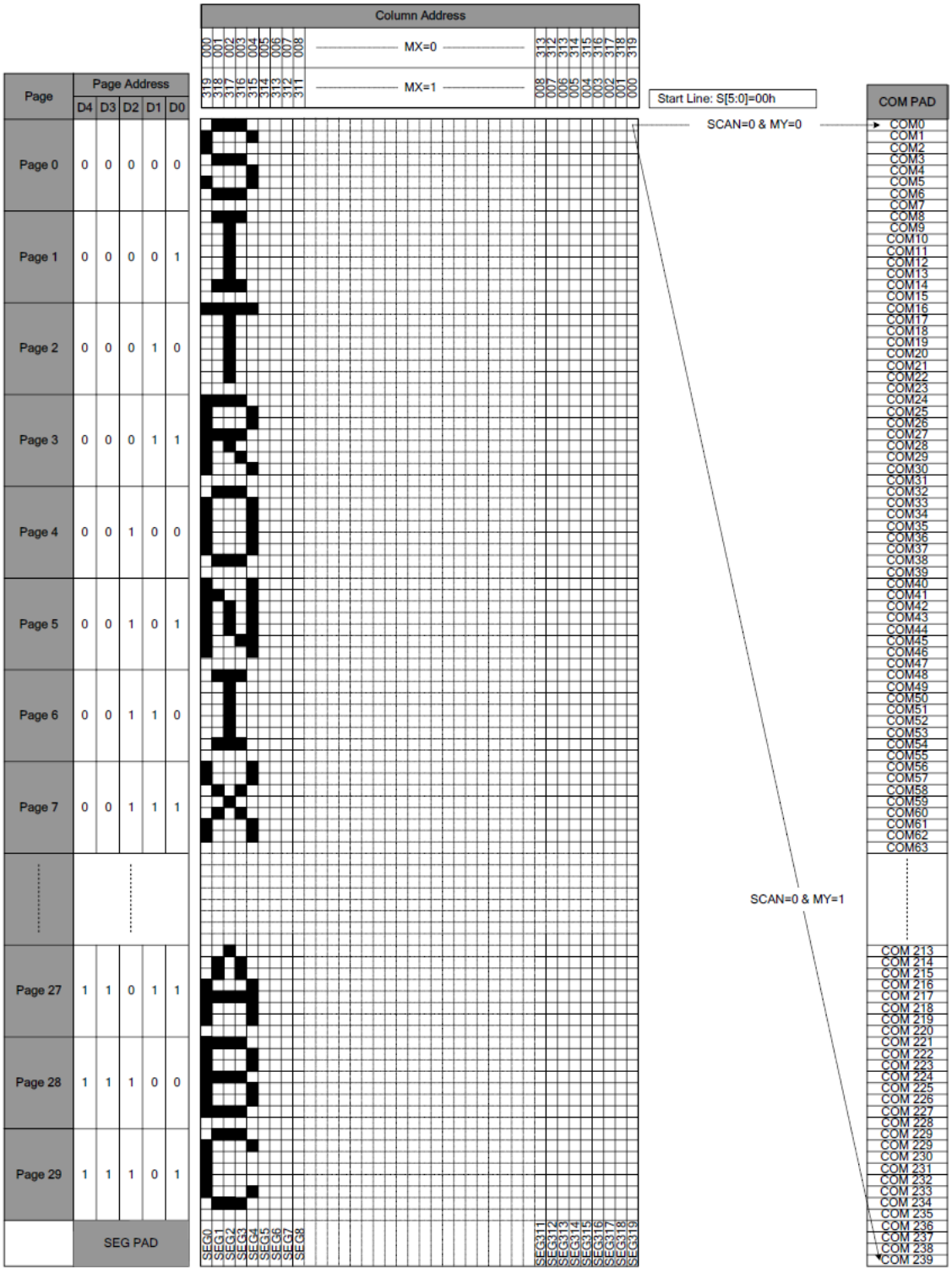


Figure 17 DDRAM Display Direction (Normal Scan)

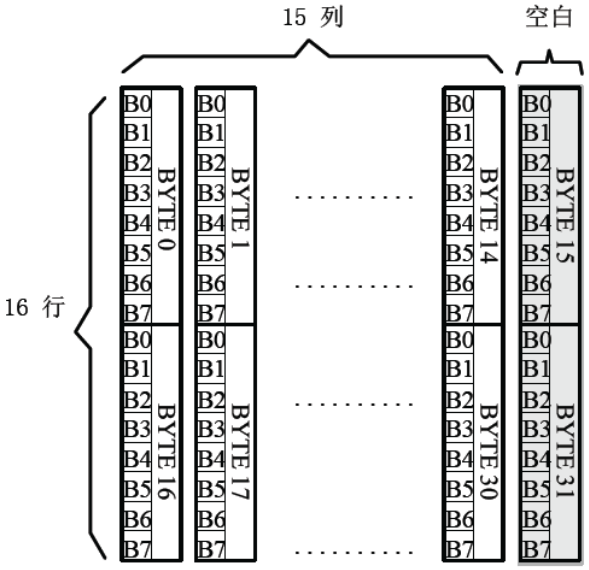
6 字库排置（竖置横排）

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：

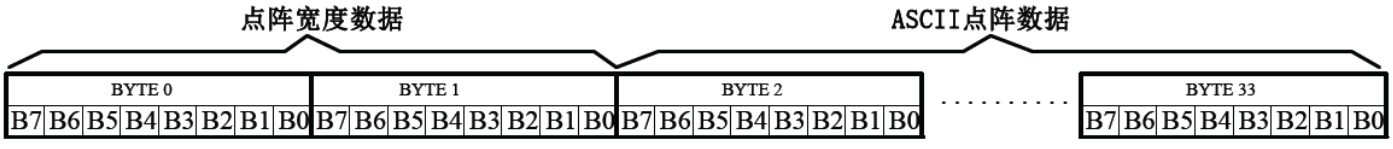


6.3 16 点阵不等宽 ASCII 方头（Arial）字符排列格式

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

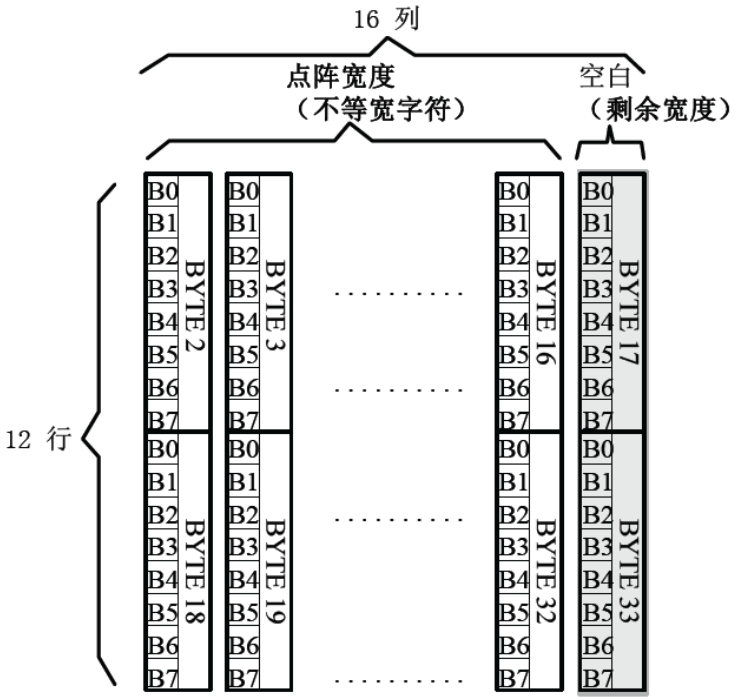
■ 存储格式

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			、	。	·	‒	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	ゎ	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	ヮ	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	ˆ	ˇ	ˉ	ˇ	ˆ	ˇ	ˊ	ˋ	ˍ	˩	˨	˧	˦	˥	▁	▔
F	ˆ	˜	‖	⋮	‖	⁀										

GB2312 1区

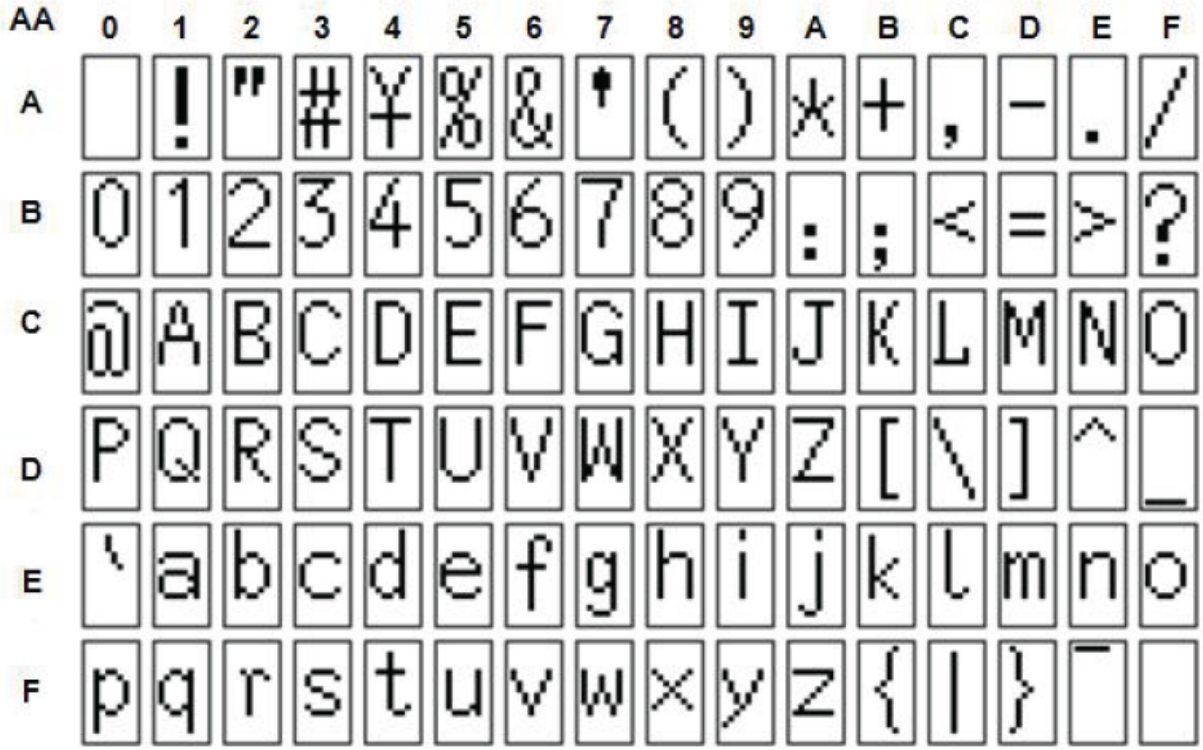
A7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
B	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
C	Ю	Я														
D		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
E	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
F	ю	я														

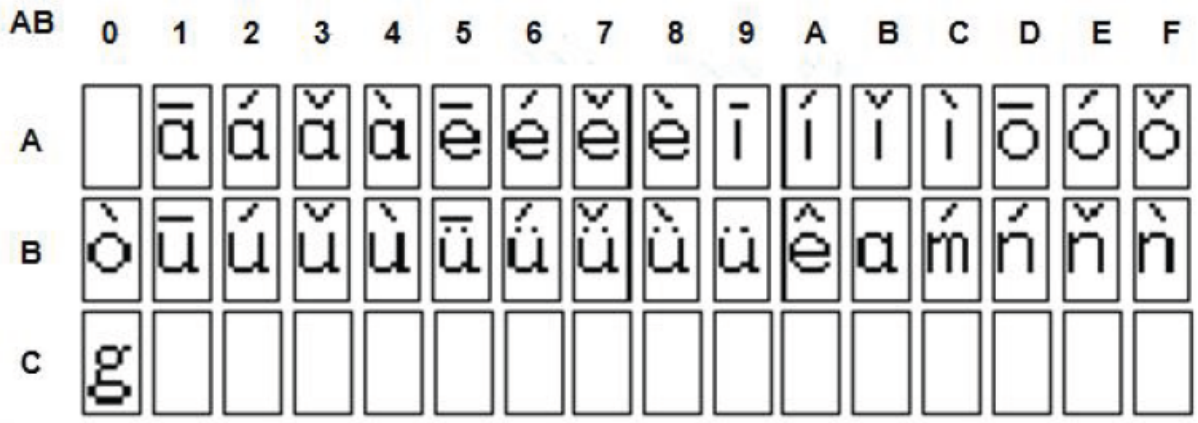
A8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ǖ	ǘ	ǚ	ǜ	ü	ê	ɑ	ḿ	ń	ň	ǹ
C	ɡ					ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ
D	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄚ	ㄛ	ㄜ	ㄝ	ㄞ	ㄟ
E	ㄠ	ㄡ	ㄢ	ㄣ	ㄤ	ㄥ	ㄦ	ㄧ	ㄨ	ㄩ						
F																

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）

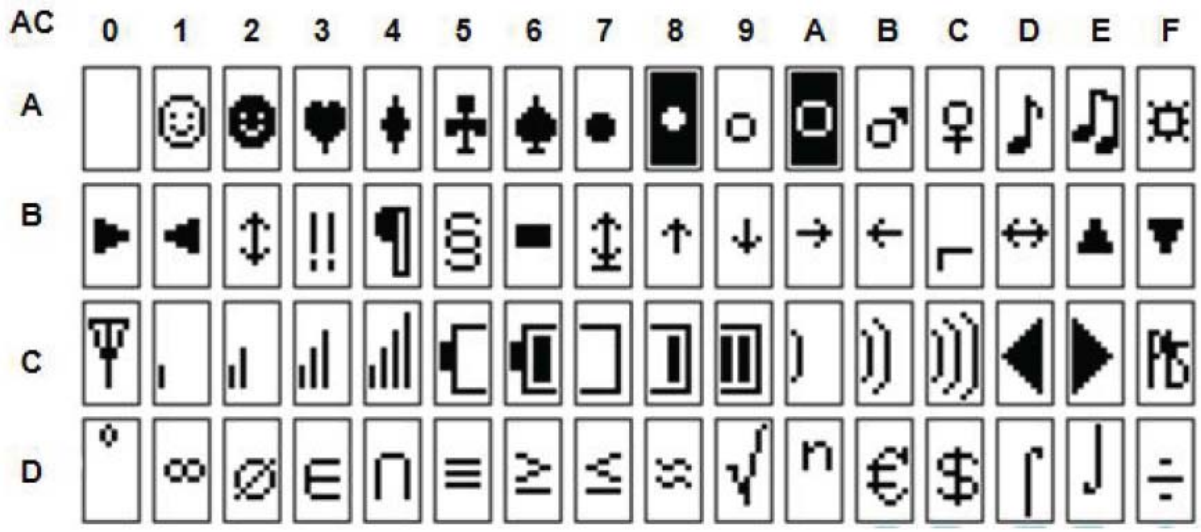
内码组成为 AAA1~ABC0 共计 126 个字符





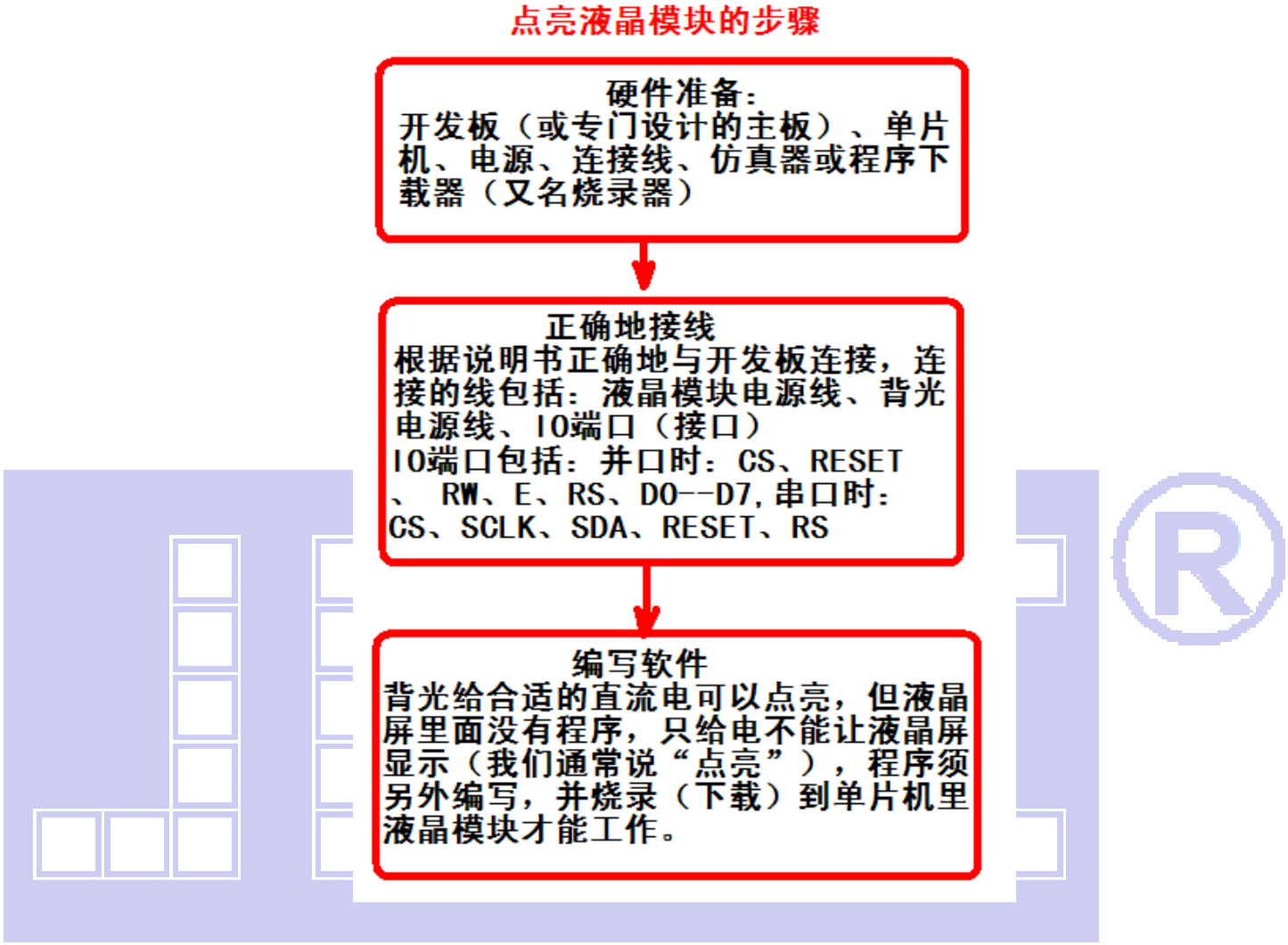
8.3 8x16 点特殊字符（64 字符）

内码组成为 ACA1~ACDF 共计 64 个字符

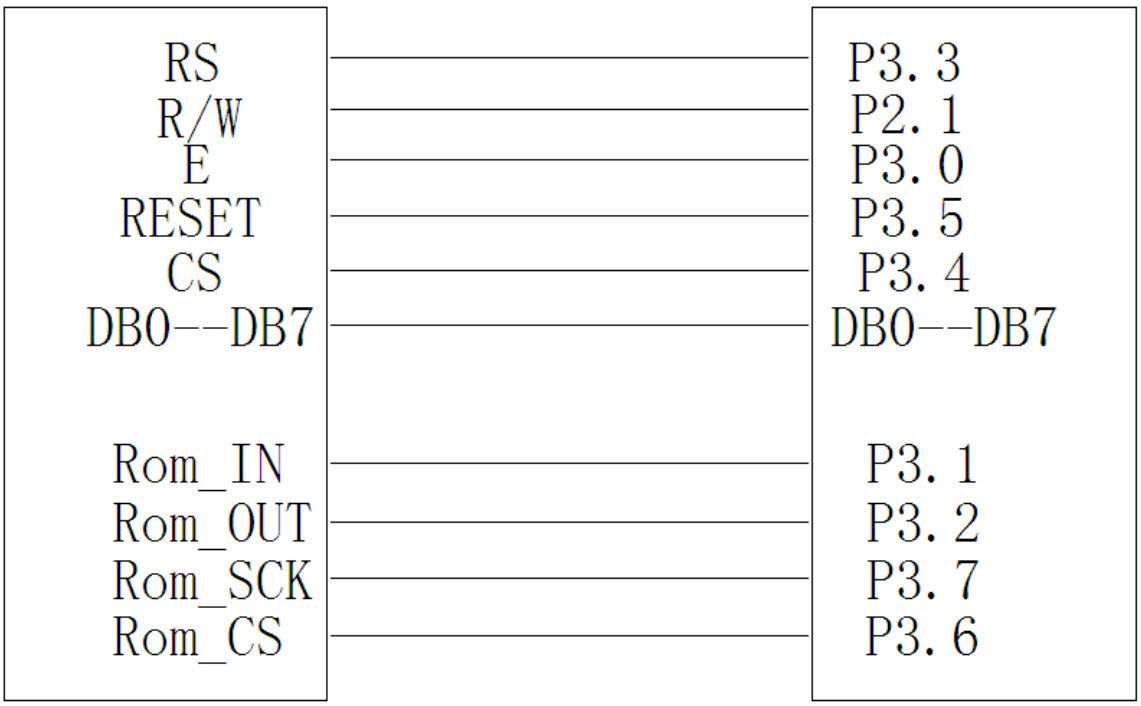


9. 硬件设计及例程：

7.1 当 LCD 驱动 IC 采用并行接口方式时的硬件设计及例程：



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：JLX320240G-905-PC-P,带字库并行接口
      驱动 IC 是:ST75320
      版权所有：晶联讯电子：网址  http://www.jlxlcd.cn;
*/
#include<reg51.h>
#include<intrins.h>
#include<chinese.h>


//=====================================================
sbit RES=P3^5;
sbit CS1=P3^4;
sbit RS=P3^3;
sbit E=P3^0;
sbit RW=P2^1;
sbit key=P2^0;

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_CS=P3^6;     /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#*/
sbit Rom_SCK=P3^7;    /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
```

```
//=====================================================
void transfer_command_lcd(int data1)
{
  RS=0;
  E=0;
  RW=0;
  CS1=0;
  E=1;
  P1=data1;
  delay_us(1);
  E=0;
  CS1=1;
}



//=====================================================
void transfer_data_lcd(int data1)
{
  RS=1;
  E=0;
  RW=0;
  CS1=0;
  E=1;
  P1=data1;
  delay_us(1);
  E=0;
  CS1=1;
}

/*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(3000);
}

/*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++);
}



//=====================================================
void initial_lcd()
{
  RES=1;
  RES=0;
  delay(10);
  RES=1;
  delay(20);

  transfer_command_lcd(0xAE); //Display OFF
  transfer_command_lcd(0xEA); //Power Discharge Control
  transfer_data_lcd(0x00);    //Discharge OFF
  transfer_command_lcd(0xA8); //sleep out

  transfer_command_lcd(0xAB); //OSC ON
  transfer_command_lcd(0x69); //Temperature Detection ON
  transfer_command_lcd(0x4E); //TC Setting
  transfer_data_lcd(0xff);    //0mV/° C, should be adjusted by customer
  transfer_data_lcd(0x44);    //0mV/° C, should be adjusted by customer
  transfer_data_lcd(0x12);    //0mV/° C, should be adjusted by customer
  transfer_data_lcd(0x11);    //0mV/° C, should be adjusted by customer
  transfer_data_lcd(0x11);    //0mV/° C, should be adjusted by customer
  transfer_data_lcd(0x11);    //0mV/° C, should be adjusted by customer
  transfer_data_lcd(0x22);    //0mV/° C, should be adjusted by customer
  transfer_data_lcd(0x23);    //0mV/° C, should be adjusted by customer

  transfer_command_lcd(0x39); //TC Flag
  transfer_data_lcd(0x00);    //
  transfer_data_lcd(0x00);    //

  transfer_command_lcd(0x2B); //Frame Rate Level
  transfer_data_lcd(0x00);    //

  transfer_command_lcd(0x5F); //Set Frame Frequency
  transfer_data_lcd(0x66);    //fFR=80Hz in all temperature range
  transfer_data_lcd(0x66);    //should be adjusted by customer
```

```
  transfer_command_lcd(0xEC); //FR Compensation Temp. Range
  transfer_data_lcd(0x19);    //TA = -15 degree, should be adjusted by customer
  transfer_data_lcd(0x64);    //TB = 60 degree, should be adjusted by customer
  transfer_data_lcd(0x6e);    //TC = 70 degree, should be adjusted by customer

  transfer_command_lcd(0xED); //Temp. Hysteresis Value (thermal sensitivity)
  transfer_data_lcd(0x04);    //Vop threshold: +2° C
  transfer_data_lcd(0x04);    //fFR threshold: +4° C

  transfer_command_lcd(0xA6); //Display Inverse OFF
  transfer_command_lcd(0xA4); //Disable Display All Pixel ON

  transfer_command_lcd(0xC4); //COM Output Status
  transfer_data_lcd(0x02);    //Interlace mode, MY=0

  transfer_command_lcd(0xA1); //Column Address Direction: MX=0

  transfer_command_lcd(0x6D); //Display Area
  transfer_data_lcd(0x07);    //Duty = 1/240 duty
  transfer_data_lcd(0x00);    //Start Group = 1

  transfer_command_lcd(0x84); //Display Data Input Direction: Column

  transfer_command_lcd(0x36); //Set N-Line
  transfer_data_lcd(0x1e);    //N-Line

  transfer_command_lcd(0xE4); //N-Line On

  transfer_command_lcd(0xE7); //LCD Drive Method
  transfer_data_lcd(0x19);    //NLFR=1//

//--------正常 VOP-----------------//
  transfer_command_lcd(0x81); //Set EV=64h
  transfer_data_lcd(0x46);    //VOP
//    transfer_data_lcd(0x3c);    //VOP
  transfer_data_lcd(0x01);    //

  transfer_command_lcd(0xA2); //BIAS
  transfer_data_lcd(0x0a);    //1/16 BIAS

  transfer_command_lcd(0x25); //Power Control
  transfer_data_lcd(0x20);    //AVDD ON
  delay(10);

  transfer_command_lcd(0x25); //Power Control
  transfer_data_lcd(0x60);    //AVDD, MV3 & NAVDD ON
```

```
  delay(10);

  transfer_command_lcd(0x25); //Power Control
  transfer_data_lcd(0x70);    //AVDD, MV3, NAVDD & V3 ON
  delay(10);

  transfer_command_lcd(0x25); //Power Control
  transfer_data_lcd(0x78);    //AVDD, MV3, NAVDD, V3 & VPF ON
  delay(10);

  transfer_command_lcd(0x25); //Power Control
  transfer_data_lcd(0x7c);    //AVDD, MV3, NAVDD, V3, VPF & VNF ON
  delay(10);

  transfer_command_lcd(0x25); //Power Control
  transfer_data_lcd(0x7e);    //VOUT, AVDD, MV3, NAVDD, V3, VPF & VNF ON
  delay(10);

  transfer_command_lcd(0x25); //Power Control
  transfer_data_lcd(0x7f);    //VOUT, AVDD, MV3, NAVDD, V3, VPF & VNF ON
  delay(10);

  clear_screen();
  transfer_command_lcd(0xaf); //Display ON
}


void lcd_address(int y,int x)
{
  transfer_command_lcd(0xb1);
  transfer_data_lcd(y);
  transfer_command_lcd(0x13);
  transfer_data_lcd((x>>8)&0x0001);
  transfer_data_lcd(x);   //
  transfer_command_lcd(0x1d);
}


//=======================================================
void clear_screen()
{
  int i,j;
  for(i=0;i<30;i++)
  {
      for(j=0;j<320;j++) //
      {
```

```
            lcd_address(i,j);
            transfer_data_lcd(0x00);
        }
    }
}


void display_string_8x16(uint page,uint 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_lcd(ascii_table_8x16[j][k+8*n]);/*显示 5x7 的 ASCII 字到 LCD 上，y 为页地址，x 为列地址，最后为数据*/
              }
          }
          i++;
          column+=8;
      }
      else
      i++;
  }
}

//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>320)
        {
            column=0;
            page+=2;
        }
        if(address !=1)
        {
            for(k=0;k<2;k++)
            {
                for(i=0;i<16;i++)
                {
                    lcd_address(page+k,column+i);
                    transfer_data_lcd(Chinese_code_16x16[address]);
                    address++;
                }
            }
            j +=2;
        }
        else
        {

            for(k=0;k<2;k++)
            {
                for(i=0;i<16;i++)
                {
                    lcd_address(page+k,column+i);
                    transfer_data_lcd(0x00);
                }
            }
            j++;
        }
        column+=16;
    }
}
```

```
//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_32x32(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_32x32[i]> 0x7e)
      {
           if(Chinese_text_32x32[i] == text[j])
           {
                if(Chinese_text_32x32[i+1] == text[j+1])
                {
                     address=i*64;
                     break;
                }
           }
           i +=2;
      }
      if(address !=1)
      {
           for(k=0;k<4;k++)
           {
                for(i=0;i<32;i++)
                {
                     lcd_address(page+k,column+i);
                     transfer_data_lcd(Chinese_code_32x32[address]);
                     address++;
                }
           }
           j +=2;
      }
      else
      {

           for(k=0;k<4;k++)
           {
                for(i=0;i<32;i++)
                {
                     lcd_address(page+k,column+i);
                     transfer_data_lcd(0x00);
```

```
                }
            }
            j++;
        }
        column+=32;
    }
}

void test(uchar data1,uchar data2)
{
  int i,j;
  for(i=0;i<30;i++)
  {
        for(j=0;j<320;j++)
        {
            lcd_address(i,j);
            transfer_data_lcd(data1);
            transfer_data_lcd(data2);
        }
  }
}


//==========================================================
void showpic(char *k)
{
  int i,j;
  for(i=0;i<30;i++)
  {
        for(j=0;j<320;j++) //
        {
            lcd_address(i,j);
            transfer_data_lcd(*k++);
        }
  }
}

/****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<<=1;
       if( Rom_OUT )
           ret_data+=1;
       else
           ret_data=ret_data+0;
       Rom_SCK=1;
       delay_us(10);
  }
  return(ret_data);
}

//从指定地址读出数据写到液晶屏指定（page,column)座标中
void get_and_write_8x16(ulong fontaddr,uint page,uint column)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                          //地址的低 8 位,共 24 位
  for(j=0;j<2;j++)
  {
      for(i=0; i<8; i++ )
    {
            lcd_address(page+j,column+i);
            disp_data=get_data_from_ROM();
```

```
            transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
        }
    }
    Rom_CS=1;
}

void get_and_write_12x24(ulong fontaddr,uint page,uint column)
{
    uchar i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                          //地址的低 8 位,共 24 位
    for(j=0;j<3;j++)
    {
        for(i=0; i<16; i++ )
        {
            lcd_address(page+j,column+i);
            disp_data=get_data_from_ROM();
            transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
        }
    }
    Rom_CS=1;
}

void get_and_write_16x32(ulong fontaddr,uint page,uint column)
{
    uchar i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                  //地址的低 8 位,共 24 位
    for(j=0;j<4;j++)
    {
        for(i=0; i<16; i++ )
        {
            lcd_address(page+j,column+i);
            disp_data=get_data_from_ROM();
            transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
        }
    }
    Rom_CS=1;
```

```
}


//从指定地址读出数据写到液晶屏指定（page,column)座标中
void get_and_write_16x16(ulong fontaddr,uint page,uint column)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
  for(j=0;j<2;j++)
  {
      for(i=0; i<16; i++ )
    {
          lcd_address(page+j,column+i);
          disp_data=get_data_from_ROM();
          transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
      }
  }
  Rom_CS=1;
}

//从指定地址读出数据写到液晶屏指定（page,column)座标中
void get_and_write_24x24(ulong fontaddr,uint page,uint column)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
  for(j=0;j<3;j++)
  {
      for(i=0; i<24; i++ )
      {
          lcd_address(page+j,column+i);
          disp_data=get_data_from_ROM();
          transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
      }
  }
  Rom_CS=1;
}
```

```
//从指定地址读出数据写到液晶屏指定（page,column)座标中
void get_and_write_32x32(ulong fontaddr,uint page,uint column)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
  for(j=0;j<4;j++)
  {
      for(i=0; i<32; i++ )
    {
          lcd_address(page+j,column+i);
          disp_data=get_data_from_ROM();
          transfer_data_lcd(disp_data);   //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
      }
  }
  Rom_CS=1;
}

//*************************************************************
ulong  fontaddr=0;
void display_GB2312_16x16_string(uint page,uint column,uchar *text)
{
  uchar i= 0,   temp1,temp2;
  temp1=column;
  temp2=page;
  while((text[i]>0x00))
  {
      if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
      {
          //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
          //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
          //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
          fontaddr = (text[i]- 0xb0)*94;
          fontaddr += (text[i+1]-0xa1)+846;
          fontaddr = (ulong)(fontaddr*32);
          fontaddr = (ulong)(fontaddr+0x2c9d0);

          get_and_write_16x16(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
          i+=2;
          column+=16;

          if ((temp2<=29&&temp1<=320)&&column>312)
          {
```

```
            //自动换行，当遇到奇数个字母或符号就提前 8 个点
            //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
            column=1;
            page+=2;
            if (page>29)column=1;
        }
    }

    else if(((text[i]>=0xa1) &&(text[i]<=0xa3))&&(text[i+1]>=0xa1))
    {
        //国标简体（GB2312）15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算：
        //Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0
        //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
        fontaddr = (text[i]- 0xa1)*94;
        fontaddr += (text[i+1]-0xa1);
        fontaddr = (ulong)(fontaddr*32);
        fontaddr = (ulong)(fontaddr+0x2c9d0);

        get_and_write_16x16(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
        i+=2;
        column+=16;

        if ((temp2<=29&&temp1<=320)&&column>312)
        {
            //自动换行，当遇到奇数个字母或符号就提前 8 个点
            //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
            column=1;
            page+=2;
            if (page>29)column=1;
        }
    }

    else if((text[i]>=0x20) &&(text[i]<=0x7e))
    {
        fontaddr = (text[i]- 0x20);
        fontaddr = (unsigned long)(fontaddr*16);
        fontaddr = (unsigned long)(fontaddr+0x1dd780);

        get_and_write_8x16(fontaddr,page,column); //从指定地址读出数据写到液晶屏指定（page,column)座标中
        i+=1;
        column+=8;

        if ((temp1<=29&&temp2<=320)&&column>312)
        {
            //自动换行，当遇到奇数个字母或符号就提前 8 个点
            //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
```

```
                column=1;
                page+=2;
                if (page>29)column=1;
            }
        }

        else
            i++;
    }
}


//*********************************************************************
void display_GB2312_24x24_string(uint page,uint column,uchar *text)
{
  uchar i= 0,temp1,temp2;
  temp1=column;
  temp2=page;

  while((text[i]>0x00))
  {
      if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
      {
          //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
          //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
          //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
          fontaddr = (text[i]- 0xb0)*94;
          fontaddr += (text[i+1]-0xa1)+846;
          fontaddr = (ulong)(fontaddr*72);
          fontaddr = (ulong)(fontaddr+0X068190);

          get_and_write_24x24(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
          i+=2;
          column+=24;

          if ((temp2<=29&&temp1<=320)&&column>312)
          {
              //自动换行，当遇到奇数个字母或符号就提前 8 个点
              //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
              column=1;
              page+=2;
              if (page>29)column=1;
          }
      }

      else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
```

```
        {
            //国标简体（GB2312）15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*72);
            fontaddr = (ulong)(fontaddr+0X068190);

            get_and_write_24x24(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=24;

            if ((temp2<=29&&temp1<=320)&&column>312)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
                page+=2;
                if (page>29)column=1;
            }
        }

        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long)(fontaddr*48);
            fontaddr = (unsigned long)(fontaddr+0x1dff00);
            get_and_write_12x24(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=1;
            column+=12;

            if ((temp2<=29&&temp1<=320)&&column>312)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
                page+=3;
                if (page>29)column=1;
            }
        }
        else
            i++;
    }
}
```

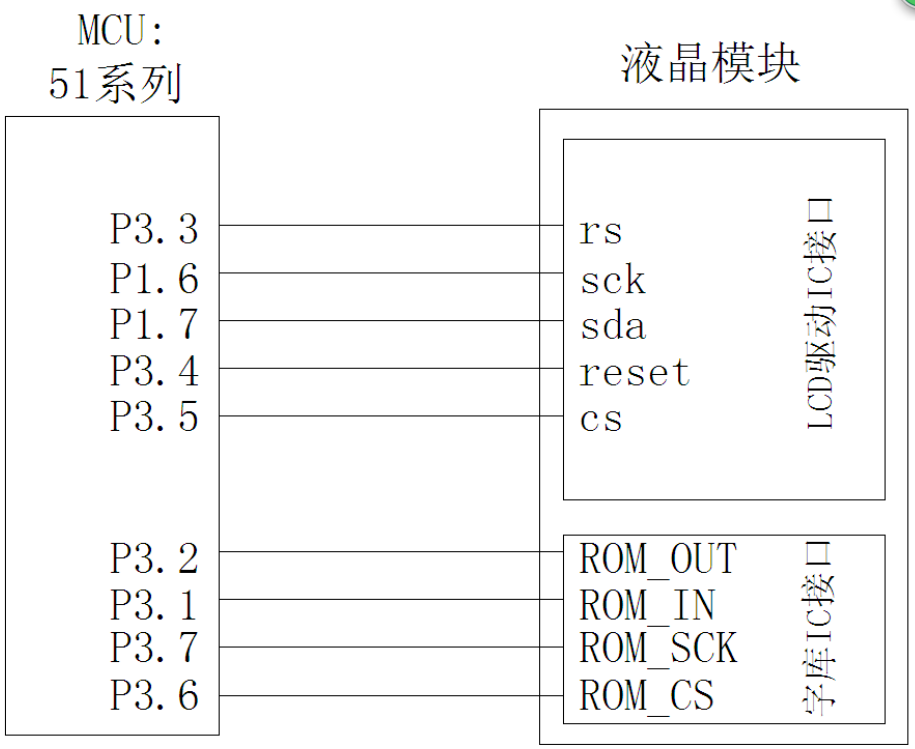
```
//*************************************************************
void display_GB2312_32x32_string(uint page,uint column,uchar *text)
{
  uchar i= 0,temp1,temp2;
  temp1=column;
  temp2=page;

  while((text[i]>0x00))
  {
      if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
      {
          //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
          //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
          //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
          fontaddr = (text[i]- 0xb0)*94;
          fontaddr += (text[i+1]-0xa1)+846;
          fontaddr = (ulong)(fontaddr*128);
          fontaddr = (ulong)(fontaddr+0xedf00);

          get_and_write_32x32(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
          i+=2;
          column+=32;

          if ((temp2<=29&&temp1<=320)&&column>312)
          {
              //自动换行，当遇到奇数个字母或符号就提前 8 个点
              //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
              column=1;
              page+=4;
              if (page>29)column=1;
          }
      }

      else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
      {
          //国标简体（GB2312）15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算：
          //Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0
          //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
          fontaddr = (text[i]- 0xa1)*94;
          fontaddr += (text[i+1]-0xa1);
          fontaddr = (ulong)(fontaddr*128);
          fontaddr = (ulong)(fontaddr+0xedf00);

          get_and_write_32x32(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
          i+=2;
```

```
            column+=32;

            if ((temp2<=29&&temp1<=320)&&column>312)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
                page+=4;
                if (page>29)column=1;
            }
        }

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

            get_and_write_16x32(fontaddr,page,column);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=1;
            column+=16;

            if ((temp2<=29&&temp1<=320)&&column>312)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>320 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
                page+=4;
                if (page>29)column=1;
            }
        }

        else
            i++;
    }
}


//=====================================================
void main()
{
  initial_lcd();          //对液晶模块进行初始化设置;
  while(1)
  {
      clear_screen();
```

```
        display_GB2312_24x24_string(0,15,"深圳市晶联讯电子有限公司");
        display_GB2312_32x32_string(3,23,"JLX320240G-905-PC");
        display_GB2312_16x16_string(7,0,"1.点阵:320x240、320 列、30 页、每页 8 行;");
        display_GB2312_16x16_string(10,0,"2.带 16x16 简体汉字库及 8x16 点阵 ASCII 字符;");
        display_GB2312_24x24_string(13,0,"3.带 24x24 简体汉字库及 12x24   点阵 ASCII 字符;");
        display_GB2312_32x32_string(20,0,"4.带 32x32 简体汉字库   16x32 点阵 ASCII 字符");
        waitkey();
        clear_screen();
        display_string_8x16(0,31,"JLX320240G-905");
        display_string_16x16(2,31,"深圳市晶联讯电子有限公司");
        display_string_32x32(4,31,"深圳市晶联讯电子");
        waitkey();
        showpic(bmp1);
        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-