

JLX256128G-257-PC 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
3	外形尺寸及接口引脚功能	4~6
4	工作电路框图	6
5	指令	7~8
6	字库的调用方法	9~17
7	硬件设计及例程:	18~末页

1. 概述

JLX257128G-257-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 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：

分类	字库内容	编码体系（字符集）	字符数
汉字及字符	15X16 点 GB2312 标准点阵字库	GB2312	6763+376
	8X16 点国标扩展字符 GB2312	GB2312	126
ASCII 字符	5X7 点 ASCII 字符	ASCII	96
	7X8 点 ASCII 字符	ASCII	96
	8X16 点 ASCII 字符	ASCII	96
	8X16 点 ASCII 粗体字符	ASCII	96
	16 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96
	16 点阵不等宽 ASCII 白正（TimesNewRoman）字符	ASCII	96



2. 字型样张:

15X16 点 GB2312 汉字

啊阿埃挨哎唉哀皑癌蔼矮艾
碍爱隘鞍氨安俺按暗岸胺案
肮昂盎凹敖熬翱袄傲奥懊澳
芭捌扒叭吧芭八疤巴拔跋靶
把耙坝霸罢爸白柏百摆佰败
拜裨斑班搬扳般颁板版扮拌

8x16 点国标扩展字符

!"#\$%&'()*+,-./012345
6789:;<=>?@ABCDEFGHIJK
LMNOPQRSTUVWXYZ[\]^_`a

5x7 点 ASCII 字符

!"#\$%&'()*+,-./0123456789:
=>?@ABCDEFGHIJKLMNPOQRSTU
VWXYZ[\]^`abcdefghijklmnopqr

7x8 点 ASCII 字符

!"#\$%&'()*+,-./01234
56789:;<=>?@ABCDEFGHIJ
KLMNOPQRSTUVWXYZ[\]^`
abcdefghijklmnopqrstu
vwxyz{|}~`@ABCDEFGHIJ

8x16 点 ASCII 字符

!"#\$%&'()*+,-./012345
6789:;<=>?@ABCDEFGHIJK
LMNOPQRSTUVWXYZ[\]^_`a

8x16 点 ASCII 粗体字符

!"#\$%&'()*+,-./012345
6789:;<=>?@ABCDEFGHIJKLM
ijklmnopqrstuvwxyz{|}

16 点阵不等宽 ASCII 方头

!"#\$%&'()*+,-./0123456789:;<=>
ABCDEFGHIJKLMNPOQRSTUVWX
abcdefghijklmnopqrstuvwxyz{

16 点阵不等宽 ASCII 白正

!"#\$%&'()*+,-./0123456789
:;<=>?@ABCDEFGHIJKLM
cdefghijklmnopqrstuvwxyz{|}

3. 外形尺寸及接口引脚功能

3.1 外形图:

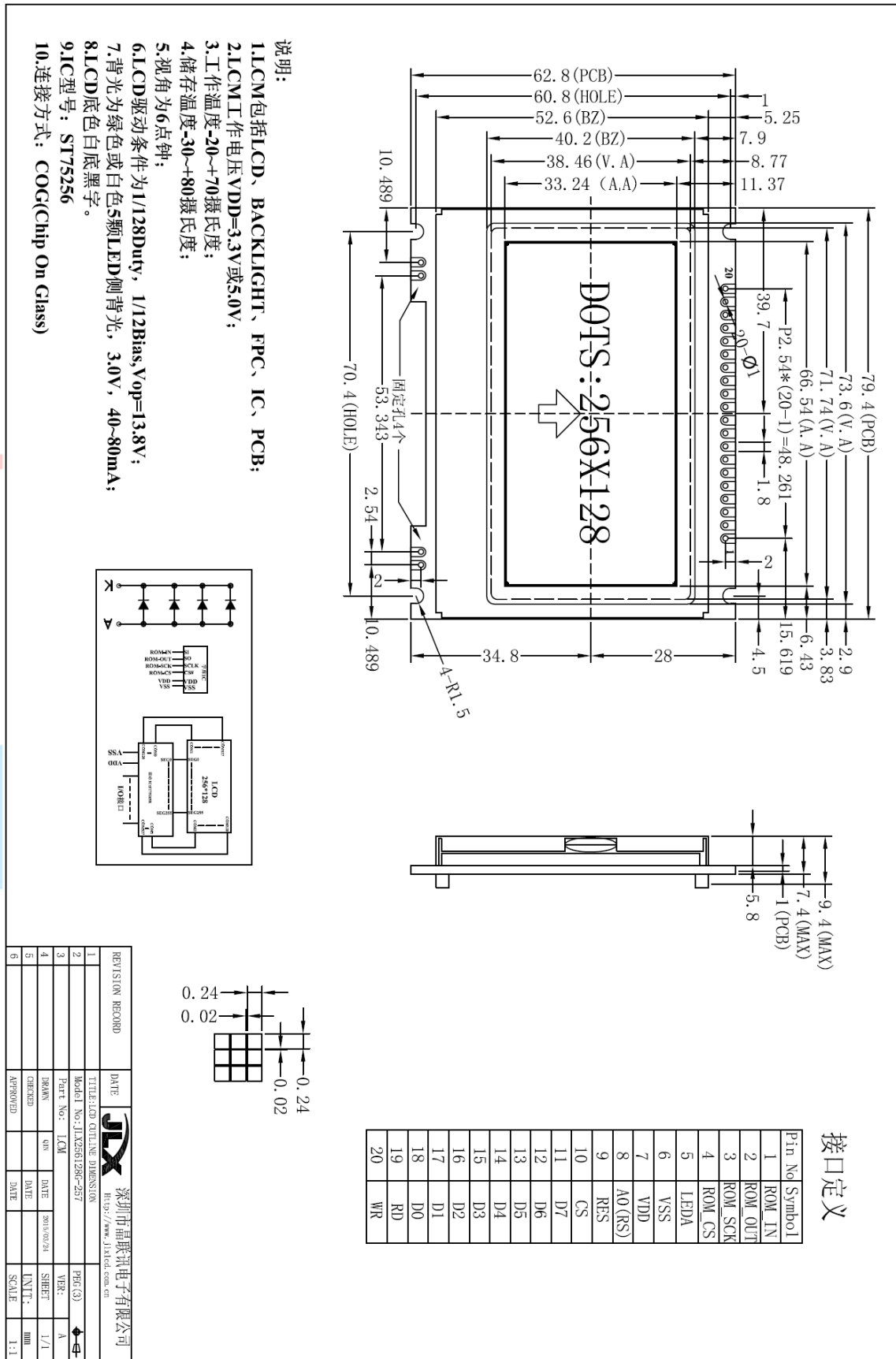


图 1. 外形尺寸

3.2 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	ROM-IN	字库 IC 接口 SI	串行数据输入
2	ROM-OUT	字库 IC 接口 SO	串行数据输出
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
详见字库 IC: JLX-GB2312 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#			
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 (A0)	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 (RD)	使能信号	并行时: 使能信号
20	R/W (WR)	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	串行数据输入
2	ROM-OUT	字库 IC 接口 SO	串行数据输出
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
详见字库 IC: JLX-GB2312 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#			
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 (A0)	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 不用建议接 VDD
15-17	D3-D1 (SDA)	I/O	串行数据
18	D0 (SCK)	I/O	串行时钟
19	E (RD)	使能信号	串行接口, RD 不用建议接 VDD
20	R/W (WR)	读/写	串行接口, WR 不用建议接 VDD

表 2: 模块串行接口引脚功能

3.2.3 I2C 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	串行数据输入
2	ROM-OUT	字库 IC 接口 SO	串行数据输出
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
<p style="color: red;">详见字库 IC: JLX-GB2312 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#</p>			
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 (A0)	寄存器选择信号	I2C 接口接 VDD
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	I2C 接口接 VSS
11-14	D7-D4	I/O	I2C 接口, 不用建议接 VDD
15-17	D3-D1 (SDA)	I/O	I2C 数据
18	D0 (SCK)	I/O	I2C 时钟
19	E (RD)	使能信号	I2C 接口, RD 不用建议接 VDD
20	R/W (WR)	读/写	I2C 接口, WR 不用建议接 VDD

表 2: 模块 I2C 接口引脚功能

4. 工作电路框图:

见图 2, 模块由 LCD 驱动 IC ST7571、字库 IC、背光组成。

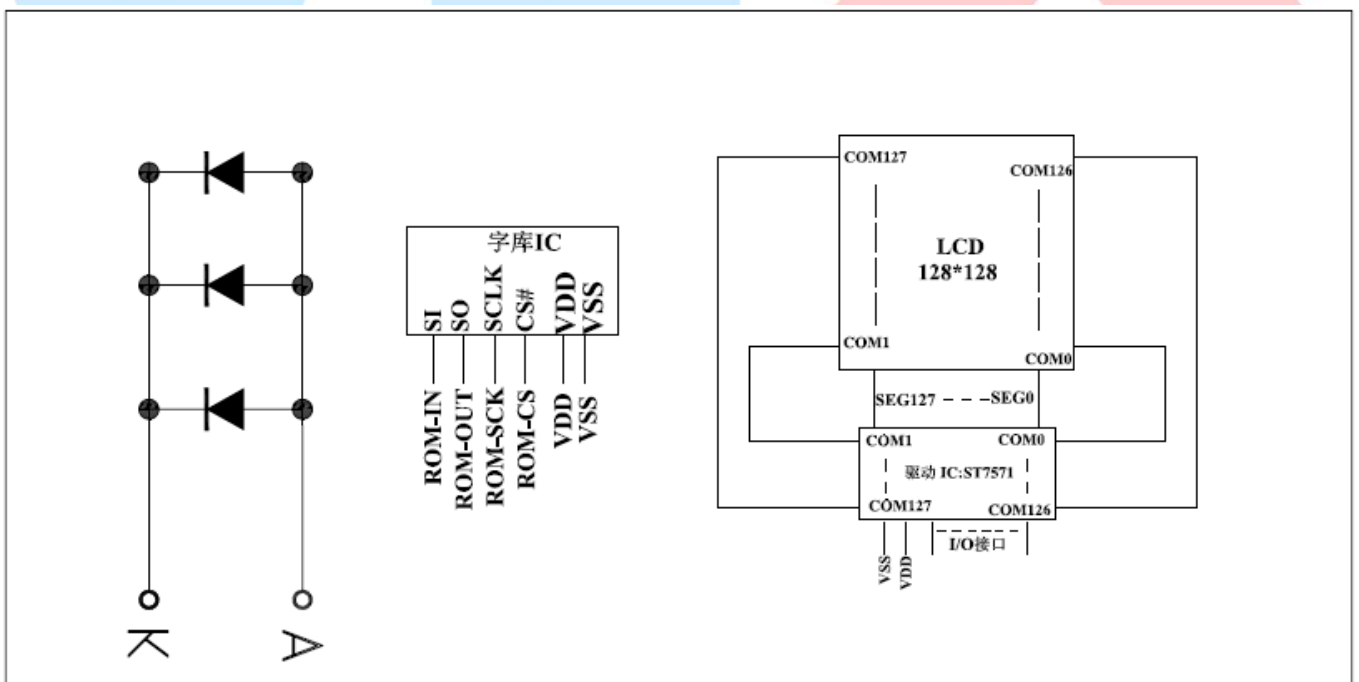


图 2: 电路框图

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):

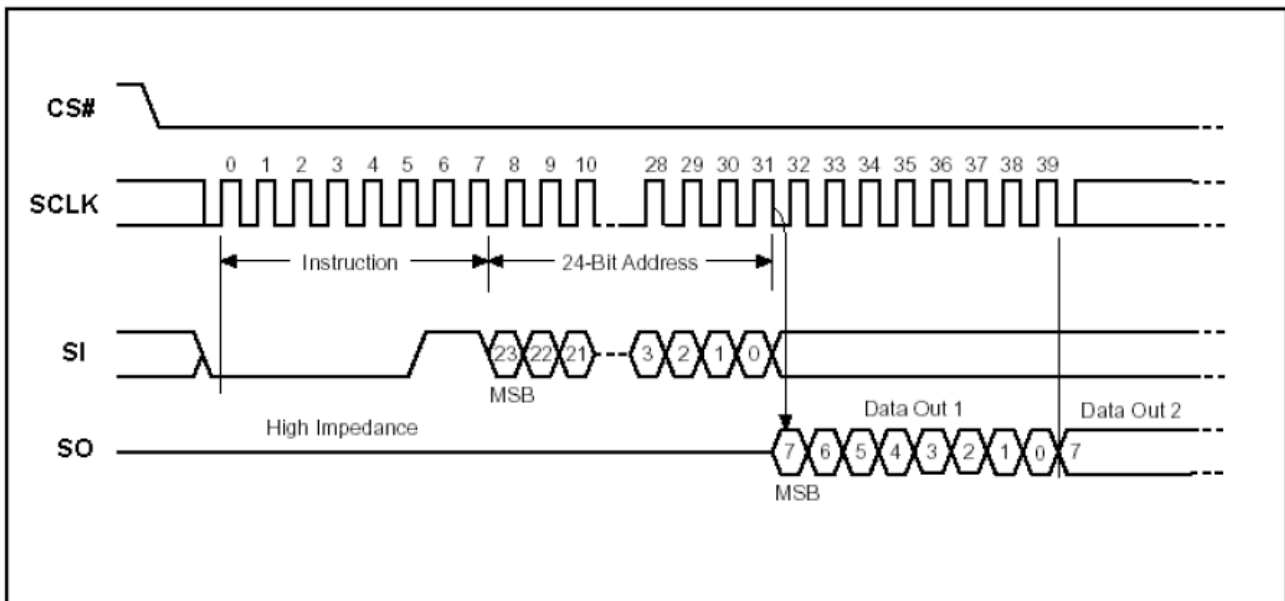
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

■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):

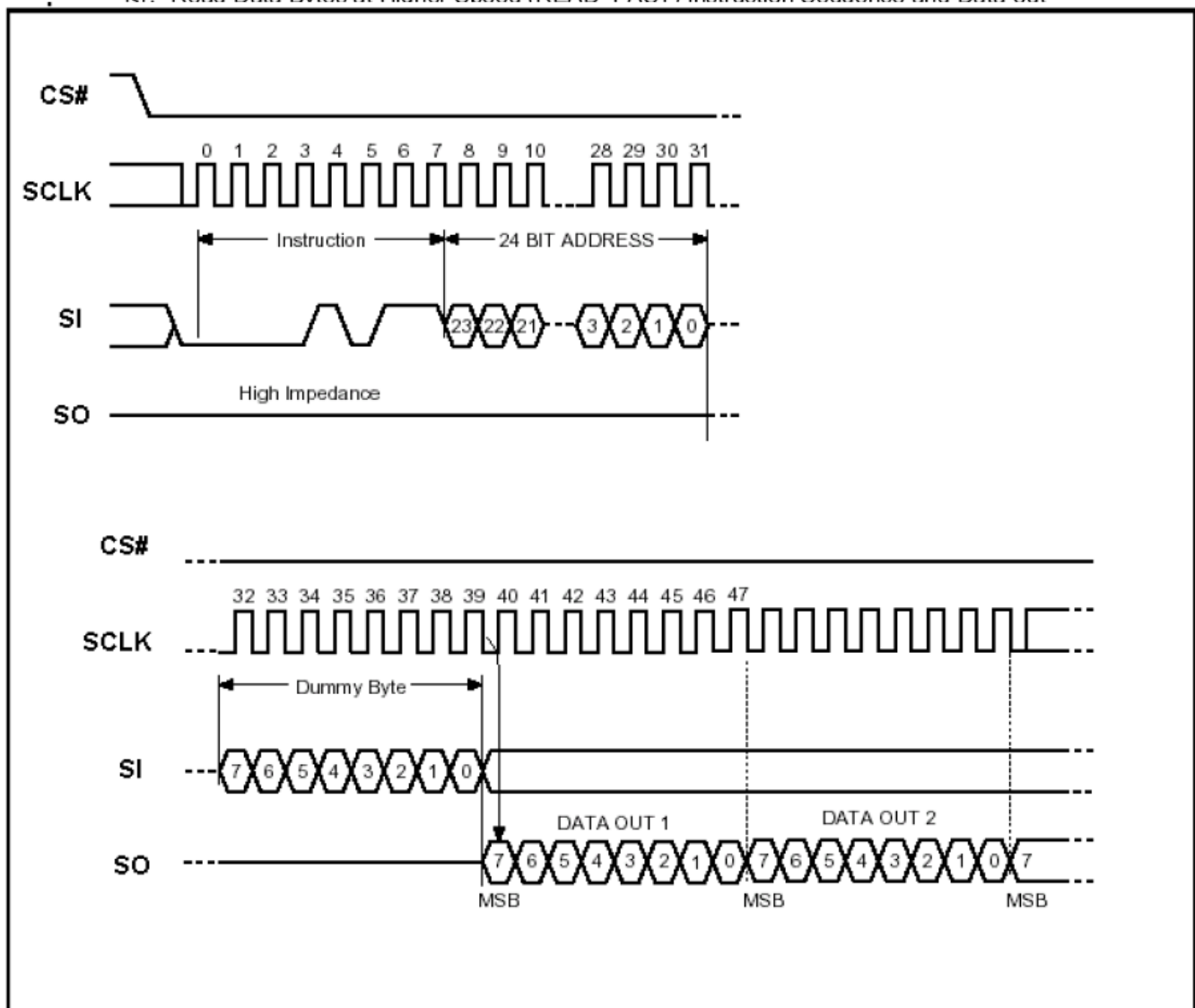
■ 首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

■ 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。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



5.2 LCD 驱动 IC 指令表详见“JLX128128G-81202”的中文说明书

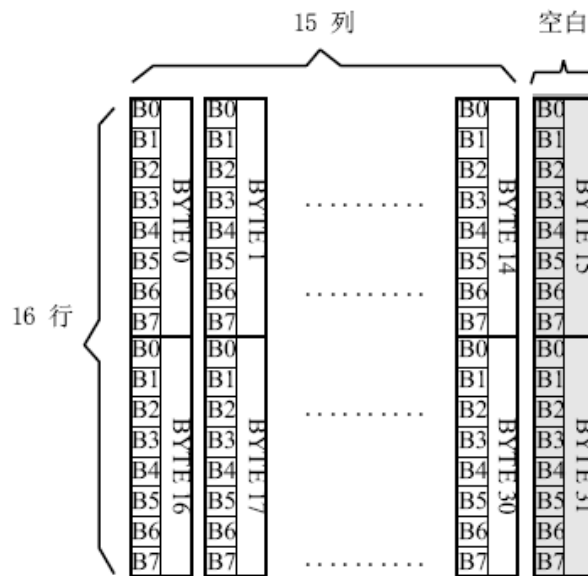
6 字库调用方法

6.1 汉字点阵排列格式

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：即一个字节的低位表示下面的点，高位表示上面的点（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。

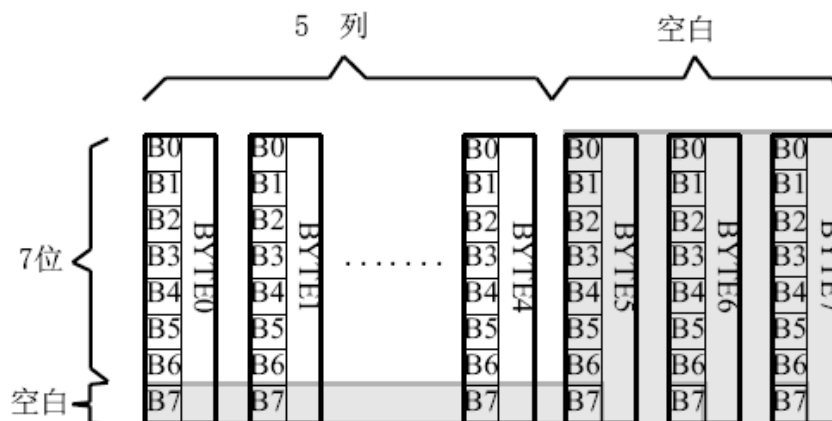
6.1.1 15X16 点汉字排列格式

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：



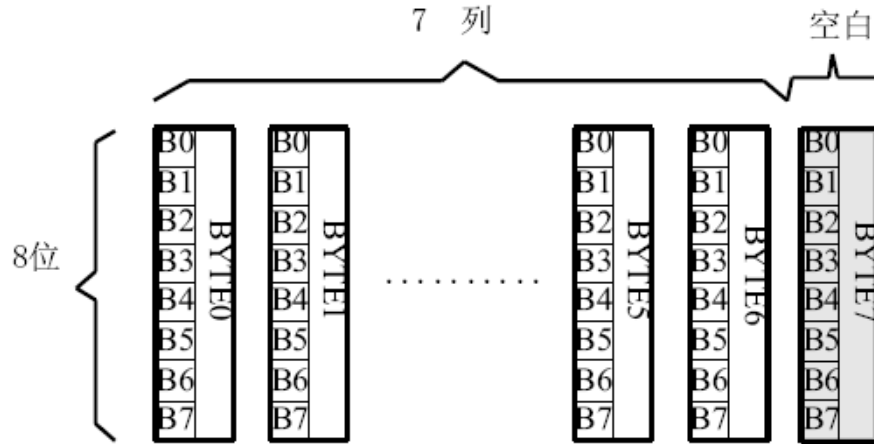
6.1.2 5X7 点 ASCII 字符排列格式

5X7 点 ASCII 的信息需要 8 个字节（BYTE 0 - BYTE7）来表示。该 ASCII 点阵数据是竖置横排的，其具体排列结构如下图：



6.1.3 7X8 点 ASCII 字符排列格式

7X8 点 ASCII 的信息需要 8 个字节 (BYTE 0 - BYTE7) 来表示。该 ASCII 点阵数据是竖置横排的, 其具体排列结构如下图:



6.1.4 8X16 点字符排列格式

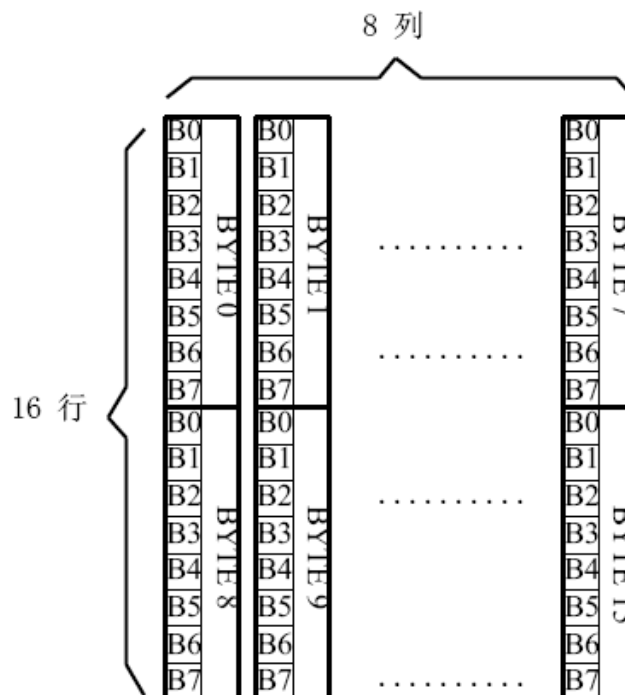
适用于此种排列格式的字体有:

8X16 点 ASCII 字符

8X16 点 ASCII 粗体字符

8X16 点国标扩展字符

8X16 点字符信息需要 16 个字节 (BYTE 0 - BYTE15) 来表示。该点阵数据是竖置横排的, 其具体排列结构如下图:

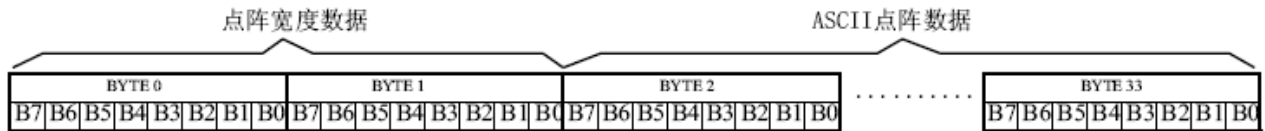


6.1.5 16 点阵不等宽 ASCII 方头 (Arial)、白正 (Times New Roman) 字符排列格式

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

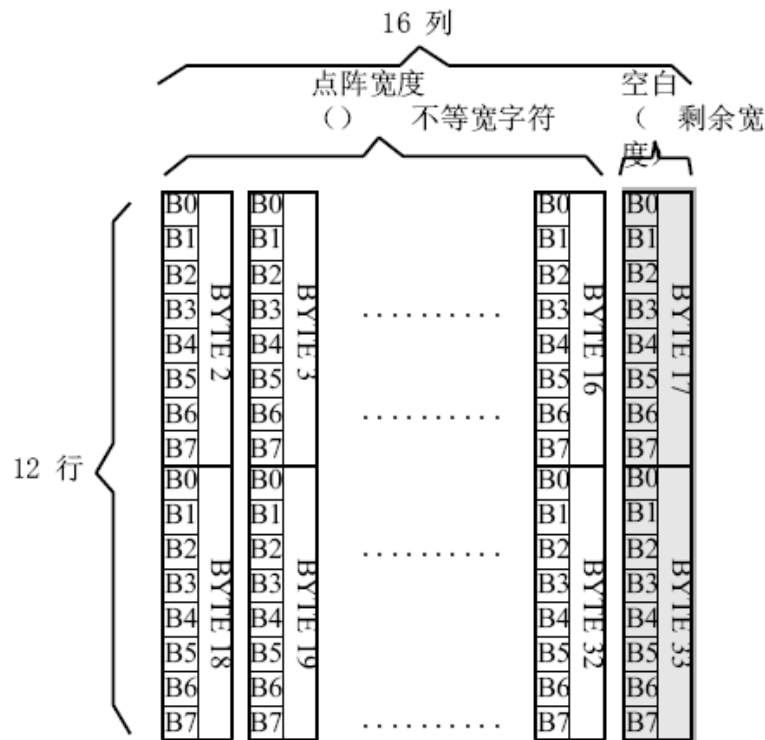
■ 存储格式

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

不等宽字符的点阵存储宽度是以 BYTE 为单位取整的, 根据不同字符宽度会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的实际宽度数据, 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 00 7F 7F 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。

BYTE2-33: 00 F8 F8 18 18 18 18 18 F8 F0 00 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。

6.2 汉字点阵字库地址表

	字库内容	编码体系	码位范围	字符数	起始地址	结束地址	参 考 法
1	15X16 点 GB2312 标准点阵字库	GB2312	A1A1-F7 FE	6763+376	00000	3B7BF	6.3.1.1
2	7X8 点 ASCII 字符	ASCII	20~7F 96		66C0	69BF	6.3.2.2
3	8X16 点国标扩展字符	GB2312	AAA1-A BC0	126	3B7D0	3BFBF	6.3.1.2
4	8X16 点 ASCII 字符	ASCII	20~7F	96	3B7C0	3BFBF	6.3.2.3
5	5X7 点 ASCII 字符 ASCII		20~7F	96	3BFC0	3C2BF	6.3.2.1
6	16 点阵不等宽 ASCII 方头 (Arial) 字符	ASCII	20~7F	96	3C2C0	3CF7F	6.3.2.4
7	8X16 点 ASCII 粗体字符 ASCII		20~7F	96	3CF80	3D57F	6.3.2.5
8	16 点阵不等宽 ASCII 白正 (TimesNewRoman) 字符	ASCII	20~7F	96	3D580	3E23F	6.3.2.6

6.3 字符在芯片中的地址计算方法

用户只要知道字符的内码，就可以计算出该字符点阵在芯片中的地址，然后就可从该地址连续读出点阵信息用于显示。

6.3.1 汉字字符的地址计算

6.3.1.1 15X16 点 GB2312 标准点阵字库

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd: 说明点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0;

if(MSB ==0xA9 && LSB >=0xA1)

Address = (282 + (LSB - 0xA1))*32+BaseAdd;

else if(MSB >=0xA1 && MSB <= 0xA3 && LSB >=0xA1)

Address =((MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;

else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)

Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;

6.3.1.2 8X16 点国标扩展字符

说明：

BaseAdd：说明本套字库在字库芯片中的起始字节地址。

FontCode：表示字符内码（16bits）

ByteAddress：表示字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x3b7d0

if (FontCode >= 0xAAA1) and (FontCode <= 0xAAFE) then

ByteAddress = (FontCode - 0xAAA1) * 16 + BaseAdd

Else if (FontCode >= 0xABA1) and (FontCode <= 0xABC0) then

ByteAddress = (FontCode - 0xABA1 + 95) * 16 + BaseAdd

6.3.2 ASCII 字符的地址计算

6.3.2.1 5X7 点 ASCII 字符

参数说明：

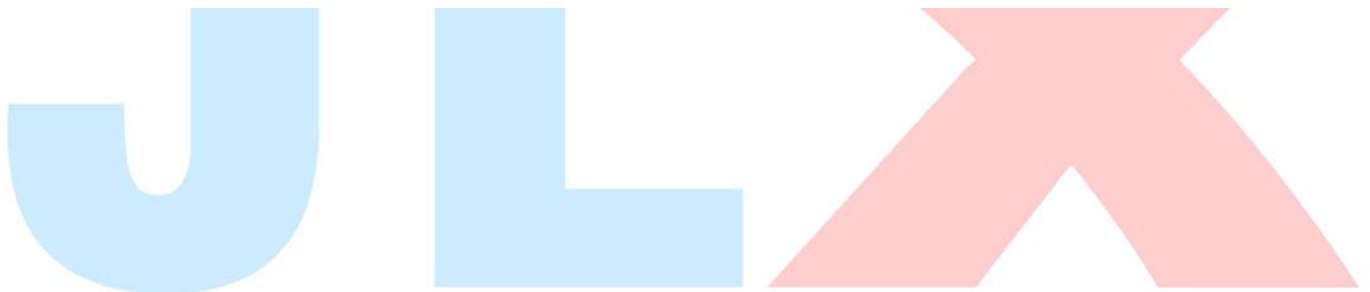
ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x3bfc0



if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode -0x20) * 8+BaseAdd

6.3.2.2 7X8 点 ASCII 字符

参数说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x66c0

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode -0x20) * 8+BaseAdd

6.3.2.3 8X16 点 ASCII 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3b7c0

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode -0x20) * 16+BaseAdd



6.3.2.4 16 点阵不等宽 ASCII 方头 (Arial) 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3c2c0

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then  
    Address = (ASCIICode -0x20 ) * 34 + BaseAdd
```

6.3.2.5 8X16 点 ASCII 粗体字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3cf80

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then  
    Address = (ASCIICode -0x20 ) * 16+BaseAdd
```

6.3.2.6 16 点阵不等宽 ASCII 白正 (Times New Roman) 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3d580

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then  
    Address = (ASCIICode -0x20 ) * 34 + BaseAdd
```

6.4 附录

6.4.1 GB2312 1 区 (376 字符)

GB2312 标准点阵字符 1 区对应码位的 A1A1~A9EF 共计 376 个字符:

GB2312 1 区

A1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A			、	。	·	-	∨	∩	〃	々	一	~		...	‘	’
B	“	”	{	}	<	>	《	》	「	」	『	』	【	】	【	】
C	±	×	÷	:	∧	∨	Σ	Π	U	∩	€	::	√	⊥	//	∠
D	∩	⊙	∫	φ	≡	≈	≈	∞	≠	≠	≠	≠	≠	≠	∞	::
E	∴	↑	♀	°	'	”	℃	\$	⊗	⊙	£	%	§	No	☆	★
F	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	=	

A2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A																
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	€		(一)	(二)	(三)	(四)	(五)	(六)	(七)	(八)	(九)	(十)	
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	{		}	~	

A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A					—	—			---	---	!	!	---	---	!	!
B	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌
C	└	└	└	└	└	└	└	└	└	└	└	└	└	└	└	└
D	┘	┘	┘	┘	┘	┘	┘	┘	┘	┘	┘	┘	┘	┘	┘	┘
E	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F																

6.4.2 8×16点国标扩展字符

内码组成为 AAA1~ABC0 共计 126 个字符

AA 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	{		}	~	

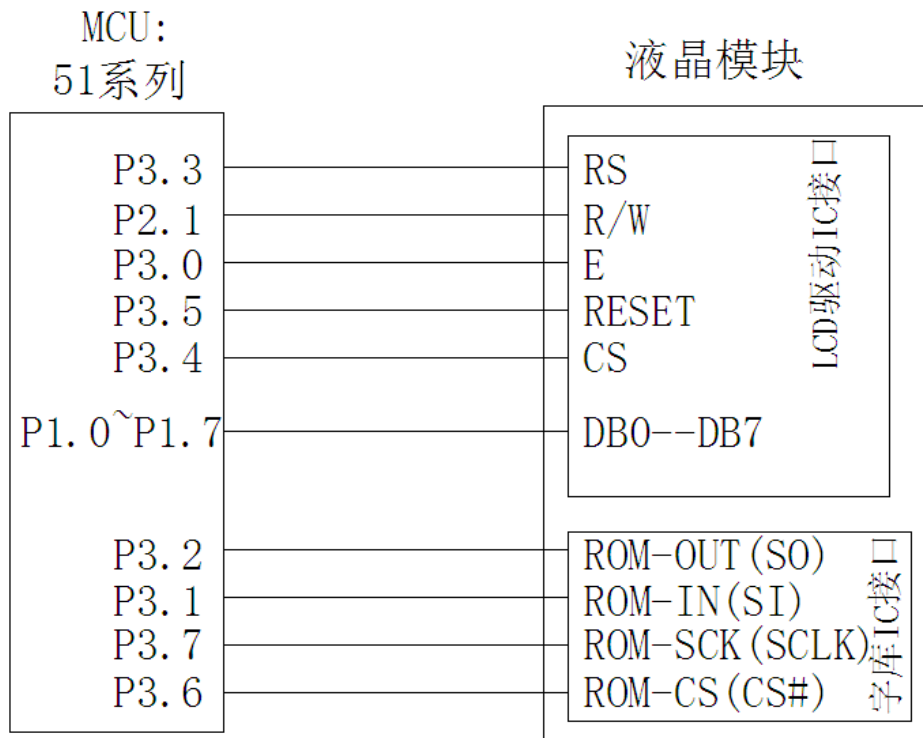
AB 0 1 2 3 4 5 6 7 8 9 A B C D E F

A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ǘ	ú	ǚ	ù	ü	ê	á	ń	ñ	ň	ň
C	g															

7. 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:

7.1.1 硬件接口: 下图为并行方式的硬件接口:



7.1.2 例程: 以下为并行方式显示汉字及 ASCII 字符的例程:

```
#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*/

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 S0*/
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#*/

#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);
}

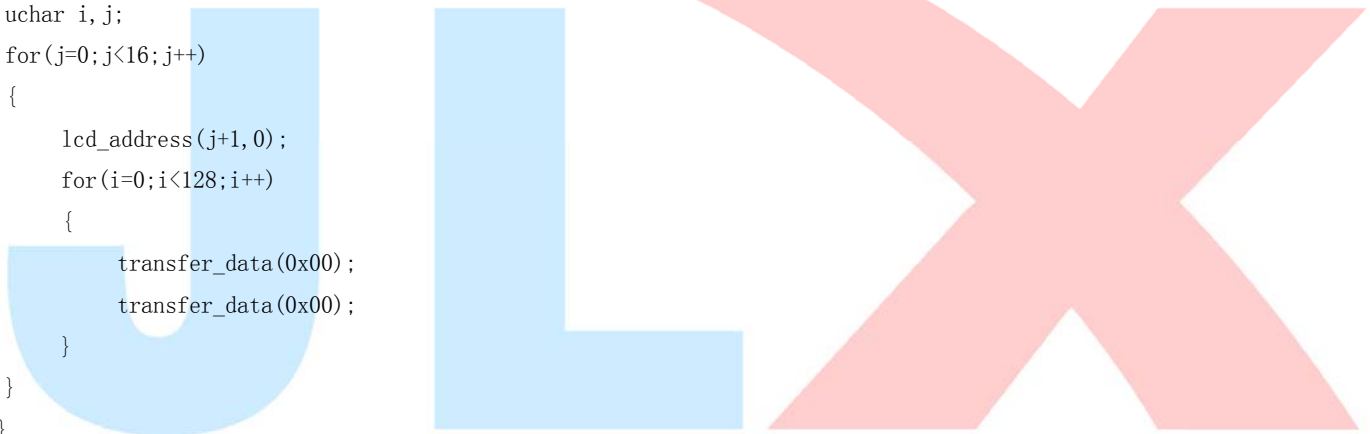
void initial_lcd()
{
reset=0;
delay(500);
reset=1;
delay(100);
transfer_command(0x2c);
delay(200);
transfer_command(0x2e);
delay(200);
transfer_command(0x2f);
delay(10);
transfer_command(0xae); //显示关
transfer_command(0x38); //模式设置
transfer_command(0xb8); //85HZ
transfer_command(0xc8); //行扫描顺序
transfer_command(0xa0); //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(0x26); //0x24 粗调对比度，可设置范围 0x20~0x27
transfer_command(0x81); //微调对比度
transfer_command(0x36); //36 微调对比度的值，可设置范围 0x00~0x3f
```

```
transfer_command(0x54); //0x54 1/9 bias
transfer_command(0xf3);
transfer_command(0x04);
transfer_command(0x93);
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++)
        {
            transfer_data(0x00);
            transfer_data(0x00);
        }
    }
}
```

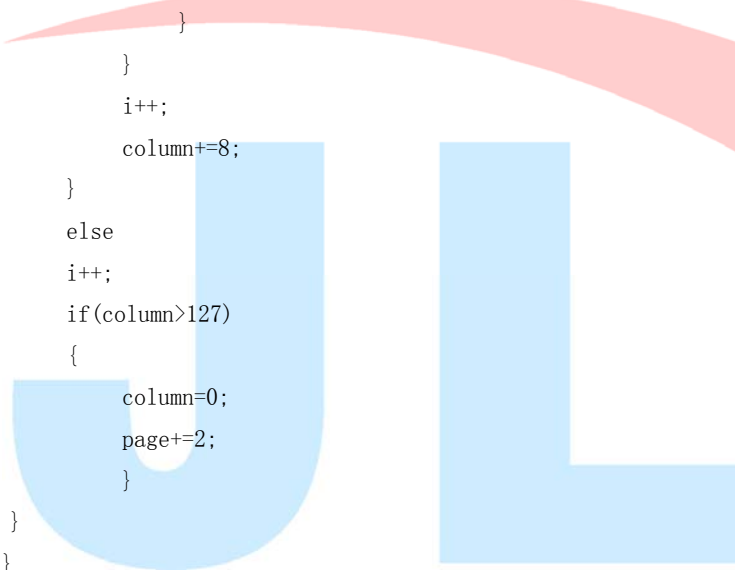
```
void test_screen()
{
    uchar i, j;
    for(j=0; j<16; j++)
    {
        lcd_address(j+1, 0);
        for(i=0; i<256; i++)
        {
            transfer_data(0xaa);
            waitkey();
        }
    }
}
```



//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'; //汉字为两个字节
            display_string_16x16(page, column, temp); //显示汉字
            column +=16;
            i +=2;
            if(column>127)
            {
                column =0;
                page +=2;
            }
        }
        else
        {
            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++;
        }
    }
}

```

```

/****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;
        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;
    delay_us(1);
}
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_16x16(ulong fontaddr, uchar page, uchar 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++)
    {
        lcd_address(page+j, column);
        for(i=0; i<16; i++)
        {
            disp_data=get_data_from_ROM();
            transfer_data(disp_data); //写数据到 LCD, 每写完 1 字节的数据后列地址自动加 1
            transfer_data(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_8x16(ulong fontaddr, uchar page, uchar 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++)
{
    lcd_address(page+j, column);
    for(i=0; i<8; i++)
    {
        disp_data=get_data_from_ROM();
        transfer_data(disp_data); //写数据到 LCD, 每写完 1 字节的数据后列地址自动加 1
        transfer_data(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_string(uchar page, uchar column, uchar *text)
{
    uchar i= 0;
    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);
            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;
        }

        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);
            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;
    }
    else if((text[i]>=0x20) &&(text[i]<=0x7e))
    {
        fontaddr = (text[i]- 0x20);
        fontaddr = (unsigned long) (fontaddr*16);
        fontaddr = (unsigned long) (fontaddr+0x3cf80);

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

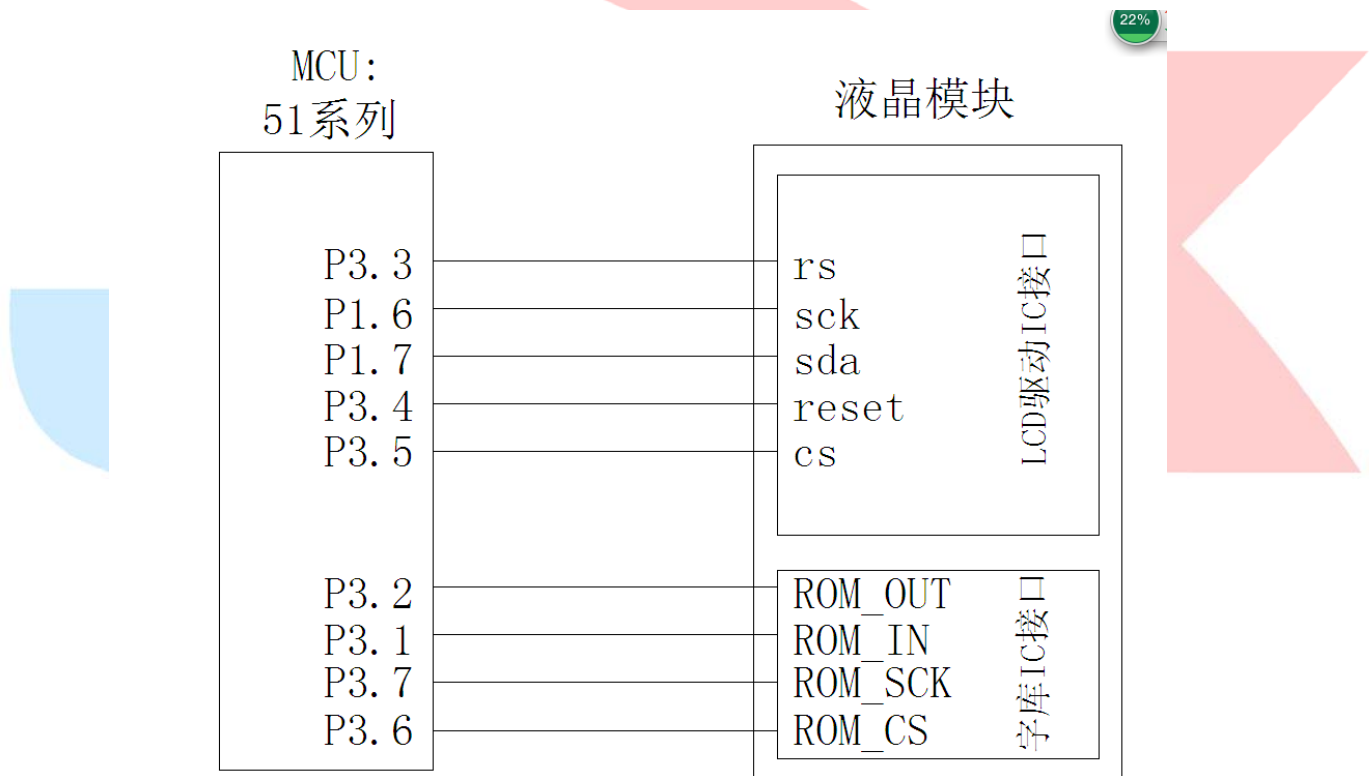
```
void main(void)
{
    initial_lcd();
    while(1)
    {
        clear_screen();
        display_GB2312_string(1, 24, "晶联讯电子");
        display_GB2312_string(3, 1, "JLX128128G-81202");
        display_GB2312_string(5, 24, "128x128 点阵");
        display_GB2312_string(7, 1, "视区:43.5x45.1mm");
        display_GB2312_string(9, 1, "带 16x16 中文字库,");
        display_GB2312_string(11, 1, "或 8x16 或 5x7 点阵");
        display_GB2312_string(13, 1, "能显示 16x16 字体,");
        display_GB2312_string(15, 1, "整屏 8 行每行 8 个字");
        waitkey();
        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!#$%&()*+,-./:;<=>?@[\\]^_`{|}~0123456789ABCDEFGHIJKLMNPOQRSTUVWXYZ");
    waitkey();
    clear_screen();
    display_string_8x16_16x16(1, 1, "深圳市晶联讯电子 JLX128128G-81202 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();
}
}
    
```

7.2 当 LCD 驱动 IC 采用串行接口方式时的硬件设计及例程:

7.2.1 硬件接口: 下图为串行方式的硬件接口:



1. ...

7.2.2 以下为串行方式的例程:

与串行方式相比较, 只需改变传送数据、指令的子程序改一下即可:

```

#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=P3^3;     /*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 S0*/
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#*/
```

//写指令到 LCD 模块

```
void transfer_command_lcd(int data1)
```

```
{
char i;
lcd_cs1=0;
lcd_rs=0;
for(i=0;i<8;i++)
{
    lcd_sclk=0;
    if(data1&0x80) lcd_sid=1;
    else lcd_sid=0;
    lcd_sclk=1;
    data1<<=1;
}
lcd_cs1=1;
}
```

//写数据到 LCD 模块

```
void transfer_data_lcd(int data1)
```

```
{
char i;
lcd_cs1=0;
lcd_rs=1;
for(i=0;i<8;i++)
{
    lcd_sclk=0;
    if(data1&0x80) lcd_sid=1;
    else lcd_sid=0;
    lcd_sclk=1;
    data1<<=1;
}
lcd_cs1=1;
```

}

