

JLX19296G-770-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

JLX19296G-770-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
VYZ[\]^`abcdefghijklmnopqr

7x8 点 ASCII 字符

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

8x16 点 ASCII 字符

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

8x16 点 ASCII 粗体字符

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

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:

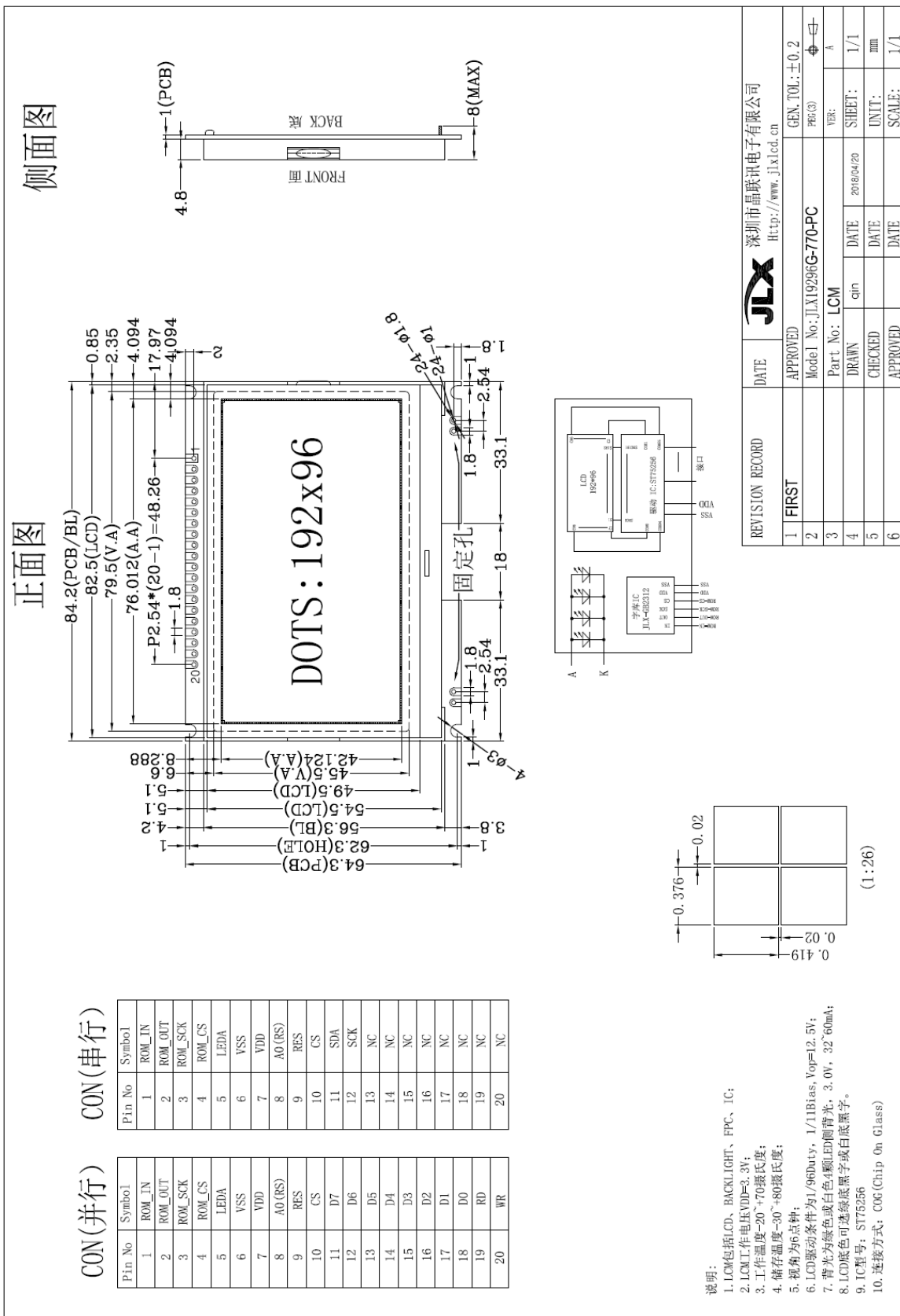


图 1. 外形尺寸

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	串行数据输出
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
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	D7	I/O	数据总线 DB7
12	D6	I/O	数据总线 DB6
13	D5	I/O	数据总线 DB5
14	D4	I/O	数据总线 DB4
15	D3	I/O	数据总线 DB3
16	D2	I/O	数据总线 DB2
17	D1	I/O	数据总线 DB1
18	D0	I/O	数据总线 DB0
19	RD(E)	使能信号	6800 时序: 使能信号
20	WR	读/写	6800 时序: H: 读数据 L: 写数据

表 1: 模块并行接口引脚功能

3.2.2 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
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
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	D7	I/O	串行接口, 此引脚不用, 建议接 VDD
12	D6	I/O	串行接口, 此引脚不用, 建议接 VDD
13	D5	I/O	串行接口, 此引脚不用, 建议接 VDD
14	D4	I/O	串行接口, 此引脚不用, 建议接 VDD
15-17	D3-D1 (SDA)	I/O	串行数据 (D3、D2、D1 连接一起做 SDA)
18	D0 (SCK)	I/O	串行时钟
19	RD (E)	使能信号	串行接口, 此引脚不用, 建议接 VDD
20	WR	读/写	串行接口, 此引脚不用, 建议接 VDD

表 2: 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	串行数据输出	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#
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 (A0)	寄存器选择信号	I2C 接口, 此引脚接 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	IIC 接口, 此引脚接 VSS	
11	D7	I/O	IIC 接口, 从属地址接 VSS	
12	D6	I/O	IIC 接口, 从属地址接 VSS	
13	D5	I/O	IIC 接口, 此引脚不用, 建议接 VDD	
14	D4	I/O	IIC 接口, 此引脚不用, 建议接 VDD	
15-17	D3-D1 (SDA)	I/O	串行数据 (D3、D2、D1 连接一起做 SDA)	
18	D0 (SCK)	I/O	串行时钟	
19	RD (E)	使能信号	IIC 接口, 此引脚不用, 建议接 VDD	
20	WR	读/写	IIC 接口, 此引脚不用, 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 ST75256、字库 IC、背光组成。

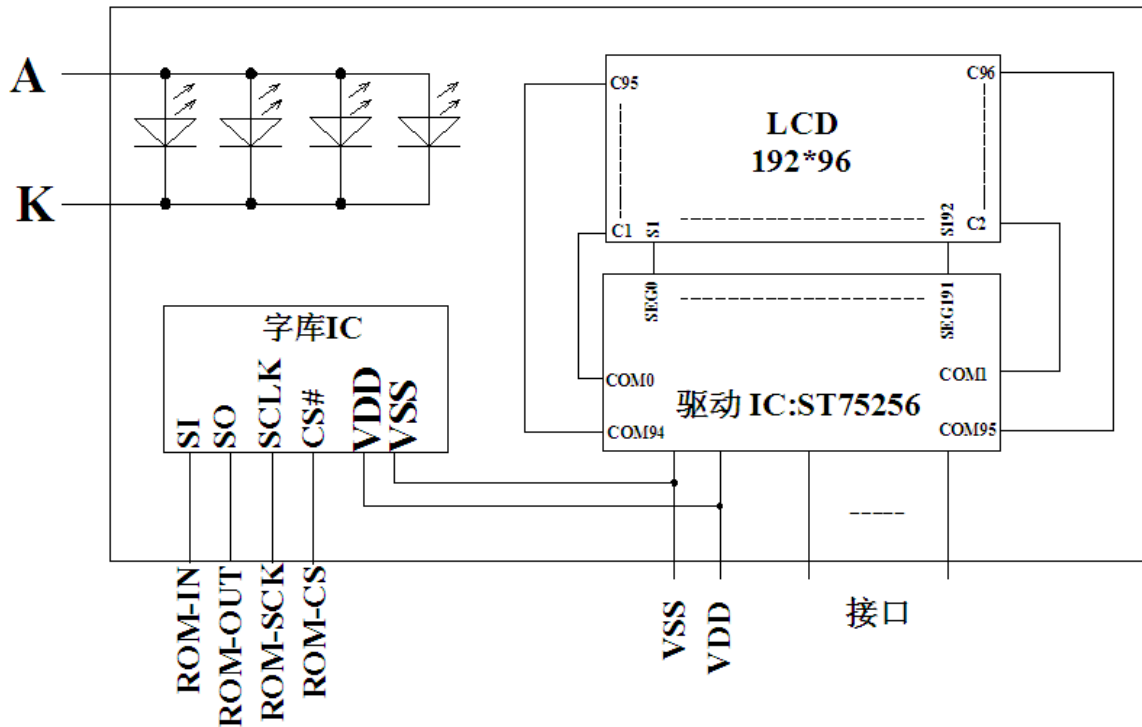


图 2: 电路框图

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):

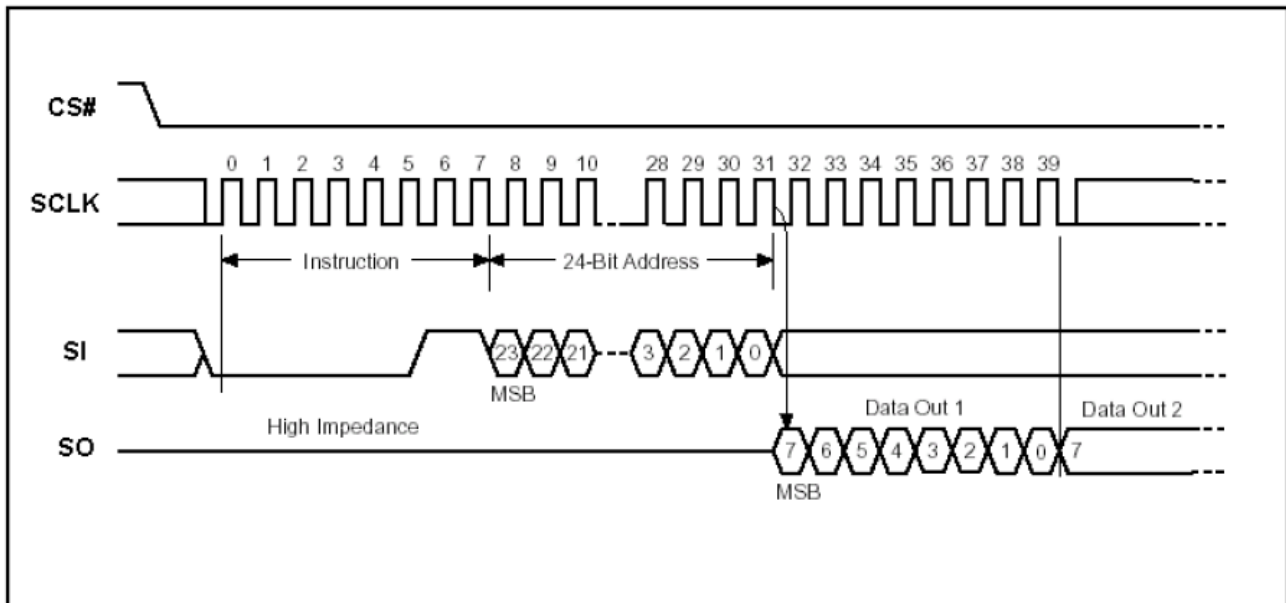
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

■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):

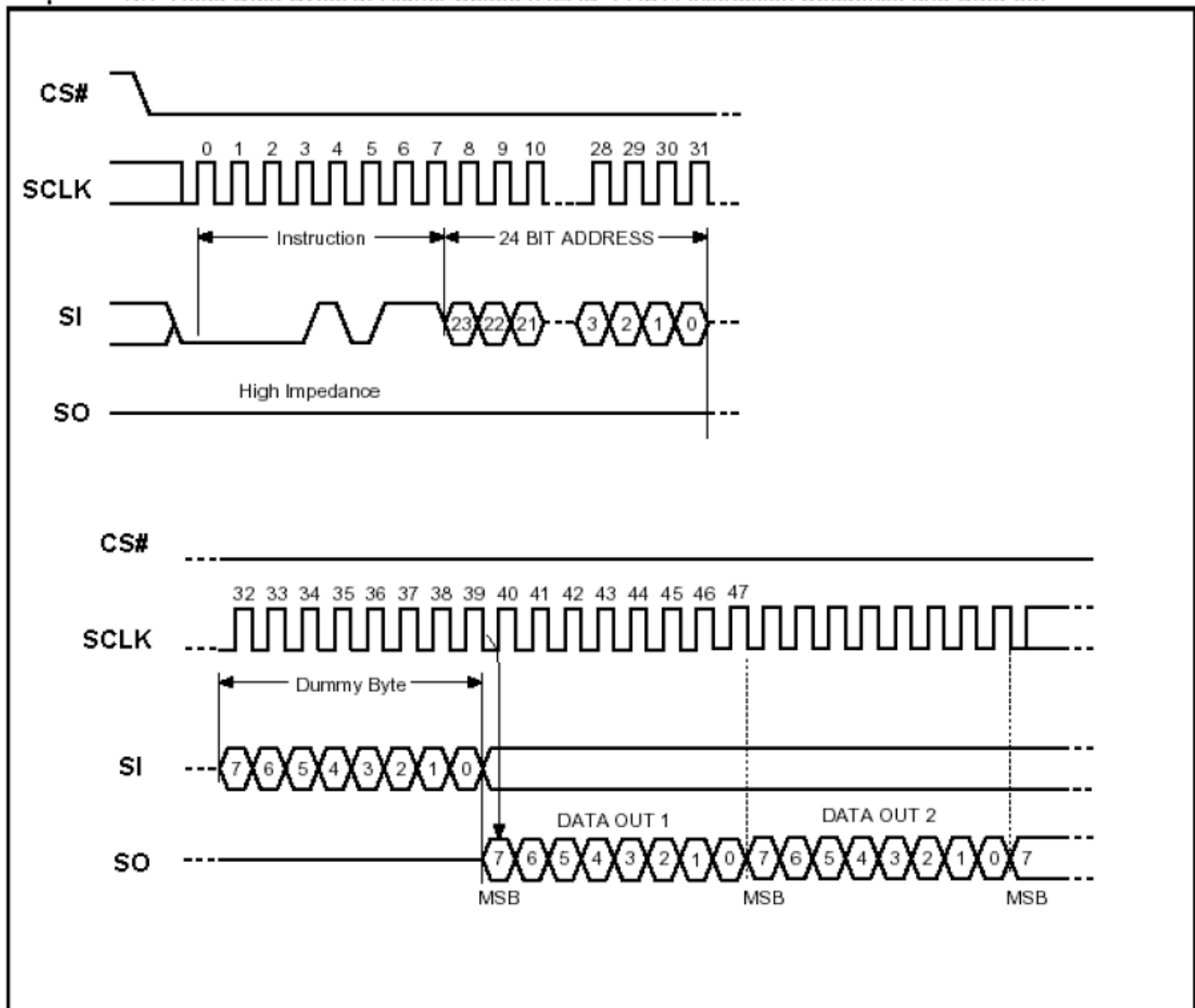
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

■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“JLX19296G-770”的中文说明书

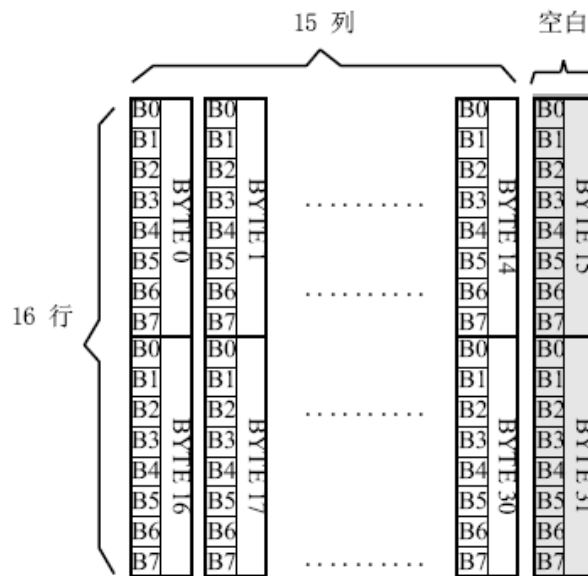
6 字库调用方法

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。

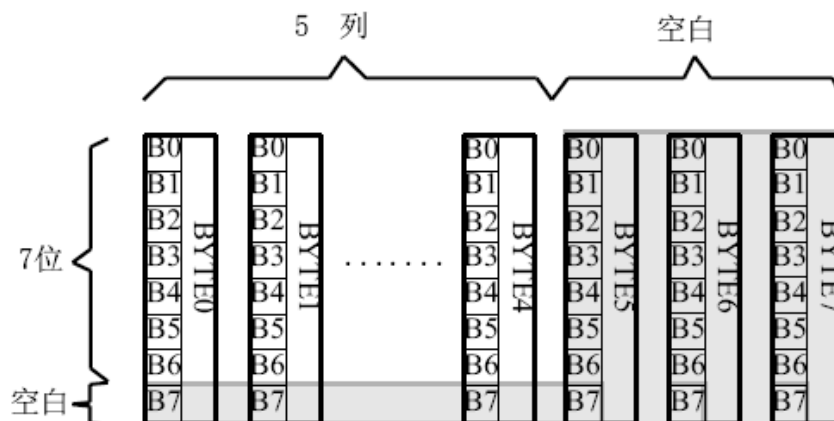
6.1.1 15X16 点汉字排列格式

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：



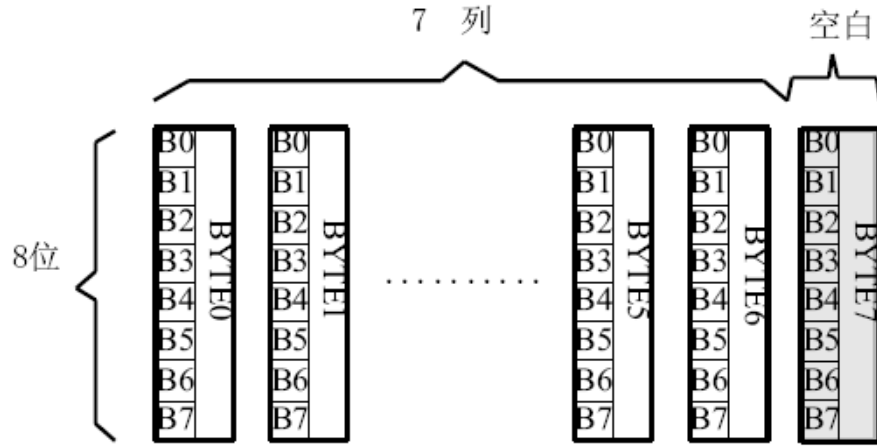
6.1.2 5X7 点 ASCII 字符排列格式

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

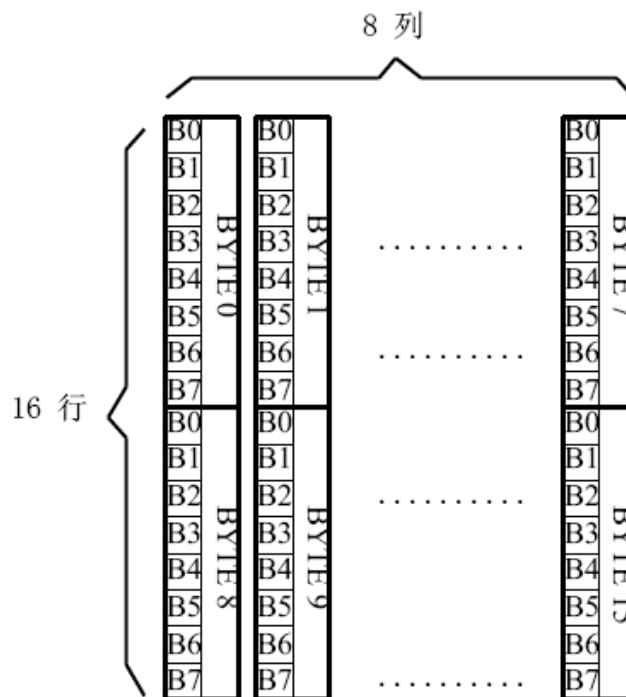
适用于此种排列格式的字有:

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:

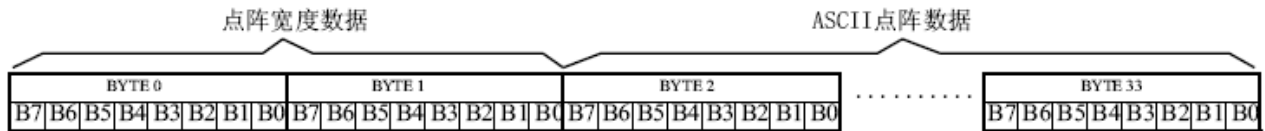


6.1.5 16 点阵不等宽 ASCII 方头 (Arial)、白正 (Times New Roman) 字符排列格式

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

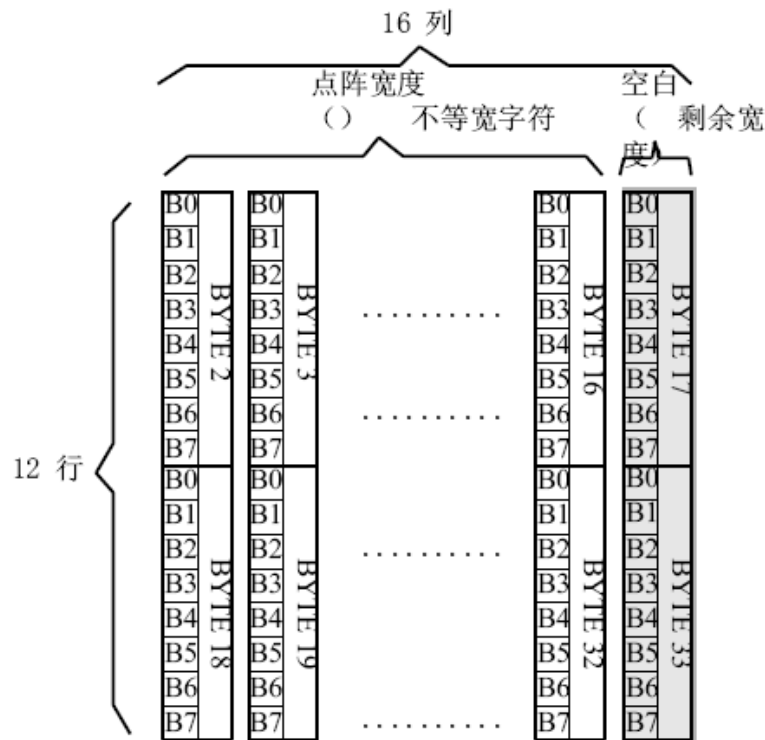
■ 存储格式

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 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	起地址	结地址	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：

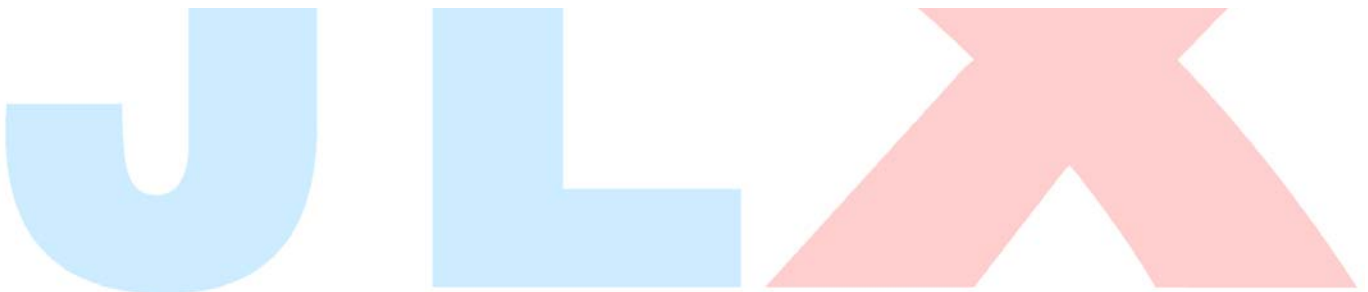
ASCIICode：表示 ASCII 码（8bits）

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	∩	⊙	∫	∫	≡	≈	≈	∞	∞	≠	≠	≠	≠	∞	:	:
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	{		}	~	

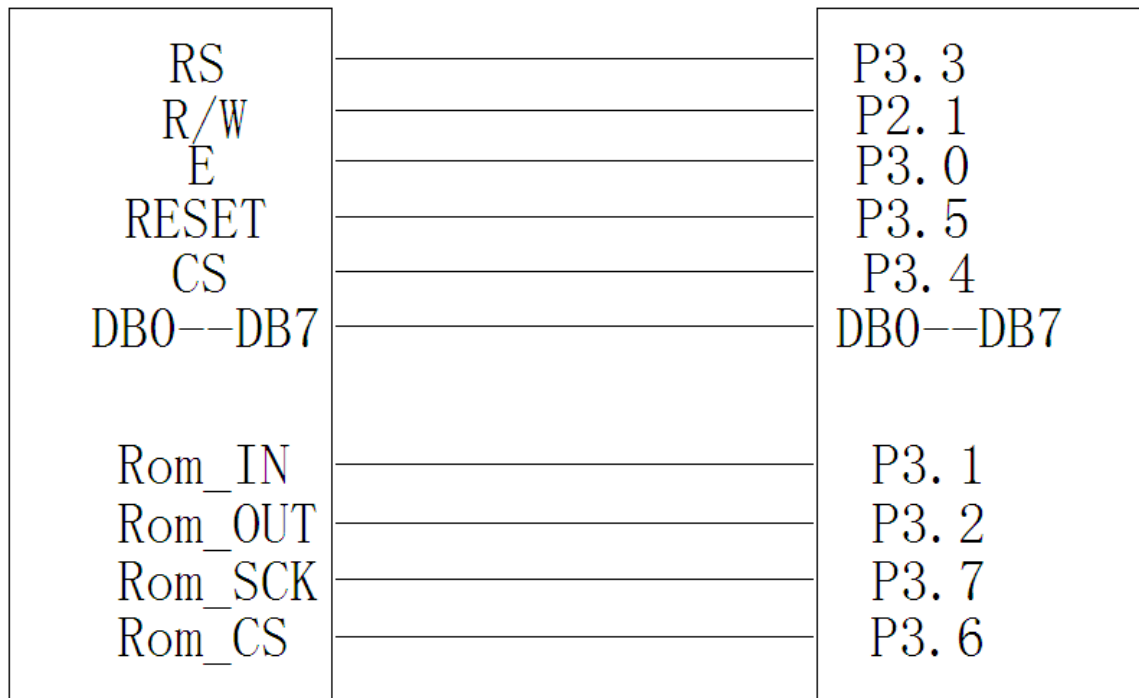
AB 0 1 2 3 4 5 6 7 8 9 A B C D E F

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	ň	ñ	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:

```

/* 液晶模块型号: JLX19296G-770-PC-P
   并行接口
   驱动 IC 是:ST75256
   版权所有: 晶联讯电子: 网址 http://www.jlxlcd.cn;
*/
#include <reg52.H>
#include <intrins.h>
#include <chinese_code.h>

sbit lcd_cs1=P3^4; /*3.4 接口定义*/
sbit lcd_reset=P3^5; /*3.3 接口定义*/
sbit lcd_rs=P3^3; /*接口定义*/
sbit lcd_rd=P3^0; /*接口定义*/
    
```

```
sbit lcd_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 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#*/
```

```
#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long
```

```
/*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);
}
```

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

```
void transfer_command(int data1)
{
    lcd_cs1=0;
    lcd_rs=0;
    lcd_rd=0;
```

```
lcd_wr=0;
P1=data1;
lcd_rd=1;
delay_us(1);
lcd_csl=1;
lcd_rd=0;
}

//-----transfer data to LCM-----
void transfer_data(int data1)
{
lcd_csl=0;
lcd_rs=1;
lcd_rd=0;
lcd_wr=0;
P1=data1;
lcd_rd=1;
delay_us(1);
lcd_csl=1;
lcd_rd=0;
}

void initial_lcd()
{
lcd_reset=0;
delay(100);
lcd_reset=1;
delay(100);

transfer_command(0x30); //EXT=0
transfer_command(0x94); //Sleep out
transfer_command(0x31); //EXT=1
transfer_command(0xD7); //Autoread disable
transfer_data(0X9F); //

transfer_command(0x31); //EXT=1
transfer_command(0xf0); //此指令比较重要, 不加之指令升压会慢 0.5s
transfer_data(0x0f);
transfer_data(0x0f);
transfer_data(0x0f);
transfer_data(0x0f);

transfer_command(0x32); //Analog SET
transfer_data(0x00); //OSC Frequency adjustment
transfer_data(0x01); //Frequency on booster capacitors->6KHz
```

```
transfer_data(0x03); //Bias=1/11

transfer_command(0x20); // Gray Level
transfer_data(0x01);
transfer_data(0x03);
transfer_data(0x05);
transfer_data(0x07);
transfer_data(0x09);
transfer_data(0x0b);
transfer_data(0x0d);
transfer_data(0x10);
transfer_data(0x11);
transfer_data(0x13);
transfer_data(0x15);
transfer_data(0x17);
transfer_data(0x19);
transfer_data(0x1b);
transfer_data(0x1d);
transfer_data(0x1f);

transfer_command(0x30); //EXT=0
transfer_command(0x75); //Page Address setting
transfer_data(0x00); // XS=0
transfer_data(0x60); // XE=159 0x28
transfer_command(0x15); //Column Address setting
transfer_data(0x00); // XS=0
transfer_data(0xc0); // XE=256

transfer_command(0xbc); //Data scan direction
transfer_data(0x00); //MX.MY=Normal
transfer_data(0xA6);

transfer_command(0xcA); //Display Control
transfer_data(0x00); //
transfer_data(0x5F); //Duty=160
transfer_data(0x20); //Nline=off

transfer_command(0xF0); //Display Mode
transfer_data(0x10); //10=Monochrome Mode, 11=4Gray

transfer_command(0x81); //EV control
transfer_data(0x2a); //VPR[5-0]
transfer_data(0x03); //VPR[8-6]
transfer_command(0x20); //Power control
transfer_data(0x0B); //D0=regulator ; D1=follower ; D3=booste, on:1 off:0
delay_us(100);
```

```
transfer_command(0xAF); //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)
{
    x=x-1;
    y=y-1;

    transfer_command(0x15); //Set Column Address
    transfer_data(x);
    transfer_data(x+x_total-1);

    transfer_command(0x75); //Set Page Address
    transfer_data(y);
    transfer_data(y+y_total-1);
    transfer_command(0x30);
    transfer_command(0x5c);
}

/*清屏*/
void clear_screen()
{
    int i,j;
    lcd_address(0,0,256,17);
    for(i=0;i<17;i++)
    {
        for(j=0;j<256;j++)
        {
            transfer_data(0x00);
        }
    }
}

//void test(int x,int y)
//{
//    int i,j;
//    lcd_address(x,y,256,16);
//    for(i=0;i<16;i++)
//    {
//        for(j=0;j<256;j++)
//        {
```

```
//          transfer_data(0xff);
//      }
//  }
//}

//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 column, uchar page, 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>255)
        {
            column=0;
            page+=2;
        }
        if(address !=1)
        {
            lcd_address(column, page, 16, 2);
            for(k=0;k<2;k++)
            {
                for(i=0;i<16;i++)
                {
                    transfer_data(Chinese_code_16x16[address]);
                    address++;
                }
            }
            j +=2;
        }
    }
}
```

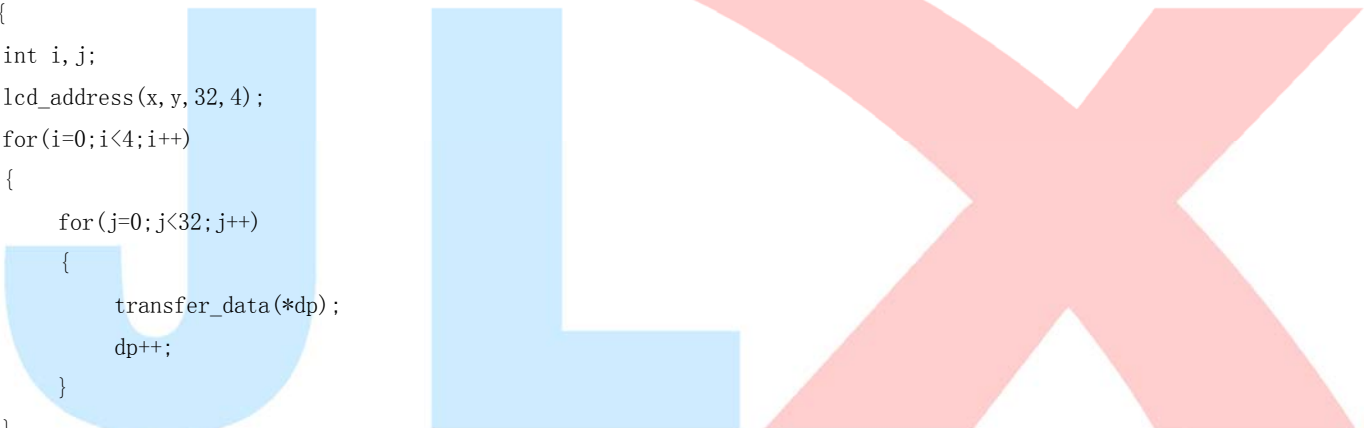


```
    }
    else
    {
        lcd_address(column, page, 16, 2);
        for(k=0;k<2;k++)
        {
            for(i=0;i<16;i++)
            {
                transfer_data(0x00);
            }
            j++;
        }
        column+=16;
    }
}
```

/*显示 32*32 点阵的汉字或等同于 32*32 点阵的图像*/

```
void disp_32x32(int x,int y,uchar *dp)
```

```
{
    int i, j;
    lcd_address(x, y, 32, 4);
    for(i=0;i<4;i++)
    {
        for(j=0;j<32;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}
```



/*显示 48*48 点阵的汉字或图像*/

```
void disp_48x48(int x,int y,char *dp)
```

```
{
    int i, j;
    lcd_address(x, y, 48, 6);
    for(i=0;i<6;i++)
    {
        for(j=0;j<48;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}
```

```
}

/*显示 64*48 点阵的汉字或图像*/
void disp_64x48(int x,int y,char *dp)
{
    int i,j;
    lcd_address(x,y,55,6);
    for(i=0;i<6;i++)
    {
        for(j=0;j<55;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}
```

```
/*显示 196*96 点阵的图像*/
void disp_192x96(int x,int y,char *dp)
{
    int i,j;
    lcd_address(x,y,192,12);
    for(i=0;i<12;i++)
    {
        for(j=0;j<192;j++)
        {
            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>>=1;
if( Rom_OUT )
ret_data+=0x80;
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 column,uchar page)
{
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
lcd_address(column, page, 16, 2);
for(j=0;j<2;j++)
{
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
}
}
}
```

```
}
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 column, uchar page)
{
    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
    lcd_address(column, page, 8, 2);
    for(j=0; j<2; j++)
    {
        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
        }
    }
    Rom_CS=1;
}

//*****
ulong fontaddr=0;
void display_GB2312_string(uchar column, uchar page, 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);
        }
    }
}
```

```
get_and_write_16x16(fontaddr, column, page); //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<=15&&temp1<=256)&&column>248)
{
    //自动换行, 当遇到奇数个字母或符号就提前 8 个点
    //设成符>256 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>15)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);

    get_and_write_16x16(fontaddr, column, page); //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 ((temp1<=15&&temp2<=256)&&column>248)
    {
        //自动换行, 当遇到奇数个字母或符号就提前 8 个点
        //设成符>128 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>15)column=1;
    }
}

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, column, page); //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;

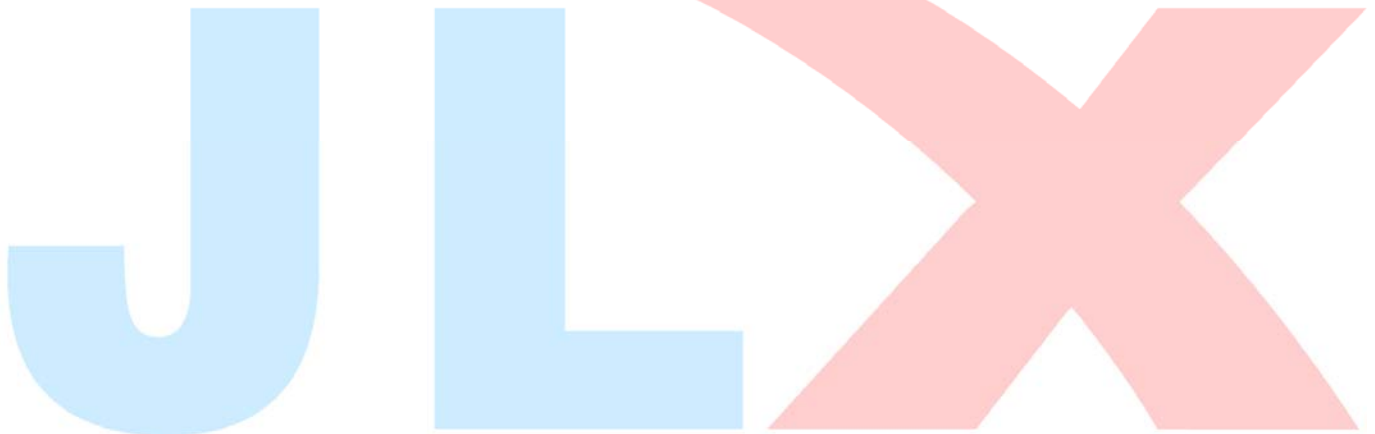
if ((temp1<=15&&temp2<=256)&&column>248)
{
    //自动换行, 当遇到奇数个字母或符号就提前 8 个点
    //设成符>128 时当有奇数个字符时就会显半个汉字, 因为一个字符只占 8 个点 (一个字节)
    column=1;
    page+=2;
    if (page>15)column=1;
}

}
else
    i++;
}
}
//-----
void main ()
{
    initial_lcd(); //对液晶模块进行初始化设置
    while(1)
    {
        clear_screen();
        display_GB2312_string(1, 1, "JLX19296G-770-PC 带中文字");
        display_GB2312_string(1, 3, "库, 可以显示 6 行每行 12 个 16");
        display_GB2312_string(1, 5, "x16 简体汉字库, 或 8x16 点阵");
        display_GB2312_string(1, 7, "ASCII, 或 7x8 点阵 ASCII, 有");
        display_GB2312_string(1, 9, "并行/串行/IIC 等接口方式");
        display_GB2312_string(1, 11, "并且有 4 灰度级功能。");
        waitkey();
        clear_screen(); //清屏
        disp_192x96(1, 1, bmp1); //显示一幅 192*96 点阵的黑白图。
        waitkey();
        clear_screen(); //清屏
        disp_192x96(1, 1, bmp2); //显示一幅 192*96 点阵的黑白图。
        waitkey();

        clear_screen(); //清屏
        disp_192x96(1, 1, bmp3); //显示一幅 192*96 点阵的黑白图。
        waitkey();
        clear_screen(); //清屏
        disp_192x96(1, 1, bmp4); //显示一幅 192*96 点阵的黑白图。
        waitkey();
        clear_screen();
    }
}

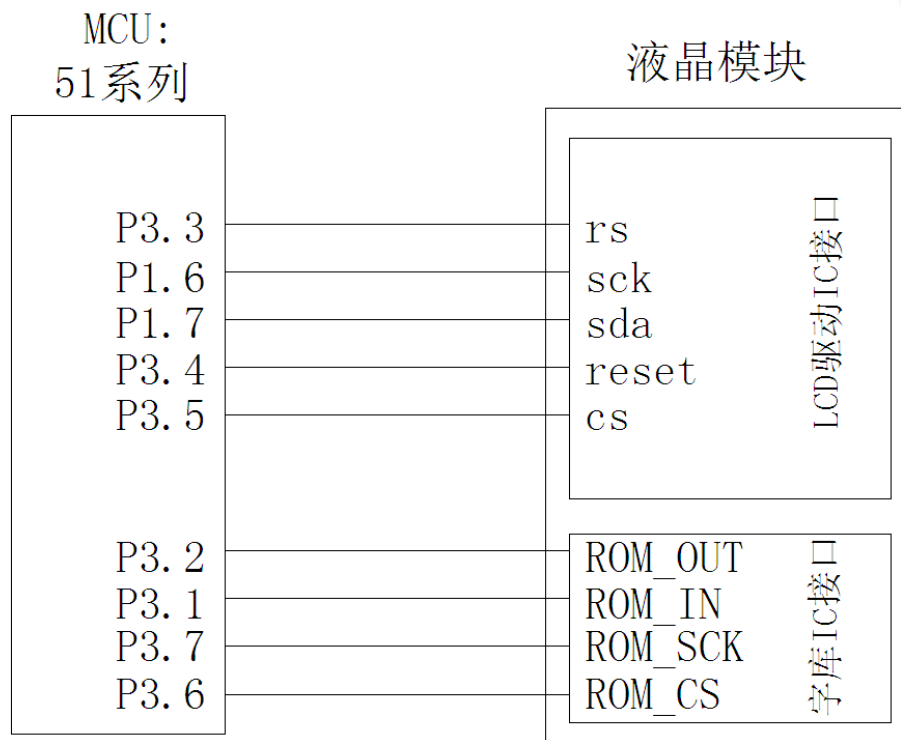
```

```
disp_64x48(5, 1, bmp5);
disp_64x48(69, 1, bmp6);
disp_64x48(133, 1, bmp7);
disp_64x48(5, 7, bmp8);
disp_64x48(69, 7, bmp9);
disp_64x48(133, 7, bmp10);
waitkey();
clear_screen(); //清屏
disp_32x32(16, 1, jing2);
disp_32x32((32*1+16), 1, lian2);
disp_32x32((32*2+16), 1, xun2);
disp_32x32((32*3+16), 1, dian2);
disp_32x32((32*4+16), 1, zi2);
waitkey();
display_string_16x16(1, 5, "深圳市晶联讯电子有限公司");
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. 连接方式

7.2.2、以下为并行接口方式范例程序

与串行方式相比较，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：

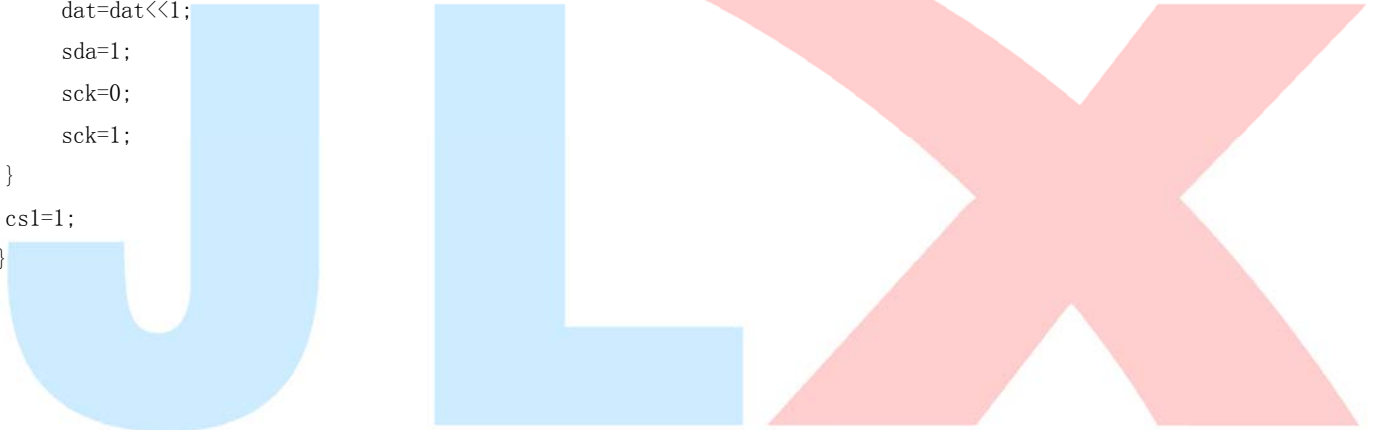
```
#include <reg51.H>
#include <intrins.h>
#include <Ctype.h>

sbit csl=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 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)
{
```

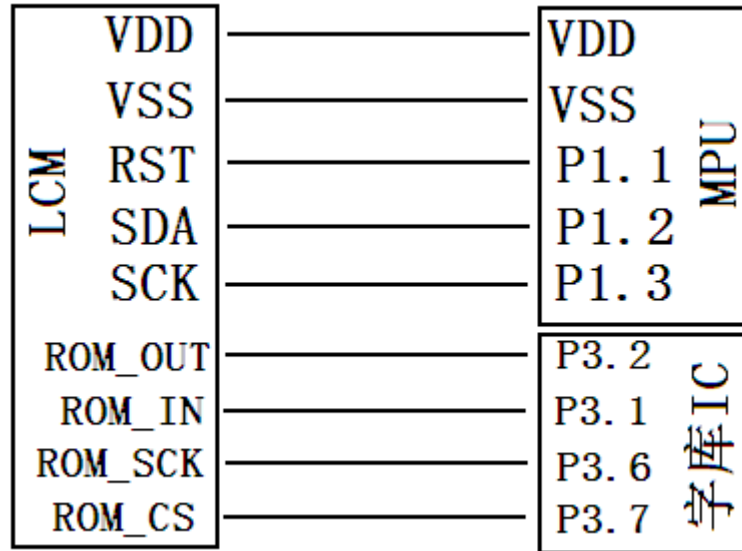


```
int k;
cs1=0;
rs=0;
for (k=0;k<8;k++)
{
    cmd=cmd<<1;
    sck=0;
    sda=0;
    sck=1;
}
cs1=1;
}
//传送数据
void transfer_data(unsigned char dat)
{
    unsigned char k;
    cs1=0;
    rs=1;
    for(k=0;k<8;k++)
    {
        dat=dat<<1;
        sda=1;
        sck=0;
        sck=1;
    }
    cs1=1;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



7.3 当 LCD 驱动 IC 采用 IIC 接口方式时的硬件设计及例程:

7.3.1 硬件接口: 下图为 IIC 方式的硬件接口:



7.3.2、以下为 IIC 接口方式范例程序

与串行方式相比较，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：

```

/* 液晶模块型号: JLX79296G-770
   并行接口
   驱动 IC 是:ST75256
   版权所有: 晶联讯电子; 网址 http://www.jlxlcd.cn;
*/
#include <reg52.H>
#include <intrins.h>
#include <chinese_code.h>

sbit reset=P1^1;
sbit scl=P1^3;
sbit sda=P1^2;
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=P3^6; /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
sbit Rom_CS=P3^7; /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS*/
sbit key=P2^0;

#define uchar unsigned char
#define uint unsigned int
    
```

```
void transfer(int data1)
{
    int i;
    for(i=0;i<8;i++)
    {
        scl=0;
        if(data1&0x80) sda=1;
        else sda=0;
        scl=1;
        scl=0;
        data1=data1<<1;
    }
    sda=0;
    scl=1;
    scl=0;
}
```

```
void start_flag()
{
    scl=1;    /*START FLAG*/
    sda=1;    /*START FLAG*/
    sda=0;    /*START FLAG*/
}
```

```
void stop_flag()
{
    scl=1;    /*STOP FLAG*/
    sda=0;    /*STOP FLAG*/
    sda=1;    /*STOP FLAG*/
}
```

//写命令到液晶显示模块

```
void transfer_command(uchar com)
{
    start_flag();
    transfer(0x78);
    transfer(0x80);
    transfer(com);
    stop_flag();
}
```

//写数据到液晶显示模块

```
void transfer_data(uchar dat)
{
    start_flag();
    transfer(0x78);
}
```

```
transfer(0xC0);  
transfer(dat);  
stop_flag();  
}
```

