

JLX19296G-923-PC

带字库 IC 的编程说明书

目 录

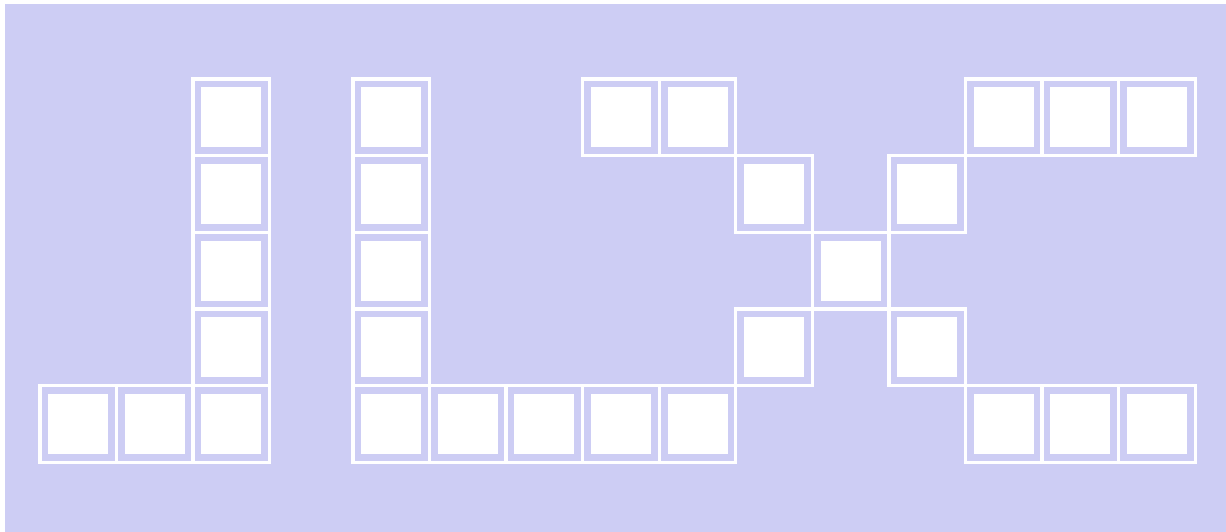
序号	内 容 标 题	页 码
1	概述	2
2	字型样张:	3
3	外形尺寸及接口引脚功能	4~6
4	工作电路框图	6
5	指令	6~8
6	字库的调用方法	9~17
7	硬件设计及例程:	18~页末

1. 概述

JLX19296G-923-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[\]^_`abcdefghijklmnopqrstuvwxyz

7x8 点 ASCII 字符

!"#\$%&'()*+,-./01234
56789:;<=>?@ABCDEFGHIJ
KLMNOPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz
6789:;<=>?@ABCDEFGHIJ

8x16 点 ASCII 字符

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

8x16 点 ASCII 粗体字符

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

16 点阵不等宽 ASCII 方头

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

16 点阵不等宽 ASCII 白正

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



3.2 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 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	
6	VSS	接地	供电电源负极	
7	VDD	电路电源	供电电源正极 (注意: 购买时须选择 3.3V 或者是 5V 供电)	
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 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	
6	VSS	接地	供电电源负极	
7	VDD	电路电源	供电电源正极 (注意: 购买时须选择 3.3V 或者是 5V 供电)	
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	NC	空脚	空脚	
15-17	SDA	I/O	串行数据 (D1、D2、D3 短接一起作为 SDA)	
18	SCK	I/O	串行时钟	
19~20	空	空	空	

表 2: 模块串行接口引脚功能

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):

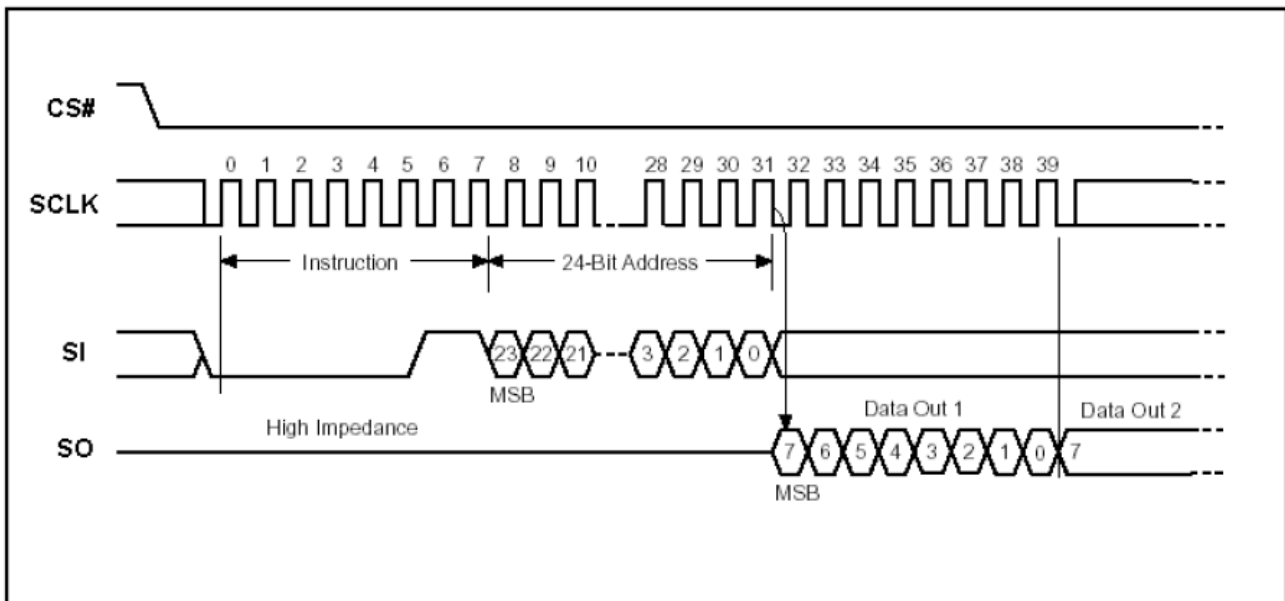
■ 首先把片选信号 (CS#) 变为低，紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入，每一位在串行时钟 (SCLK) 上升沿被锁存。

■ 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):

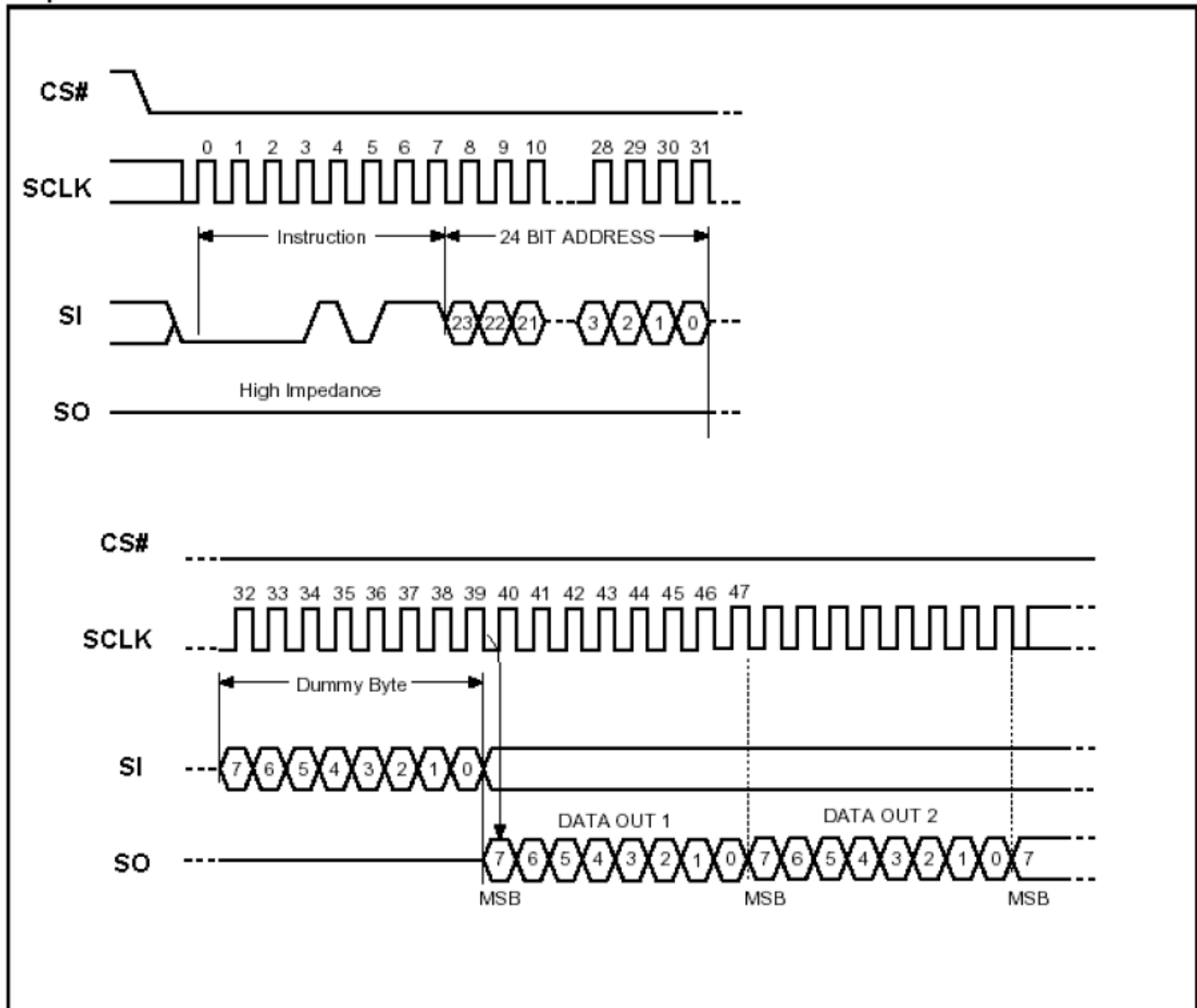
■ 首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

■ 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-923-PN”的中文说明书

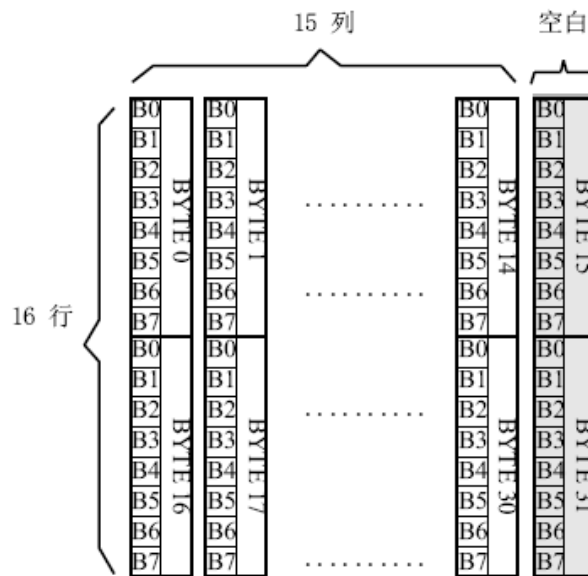
6 字库调用方法

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。

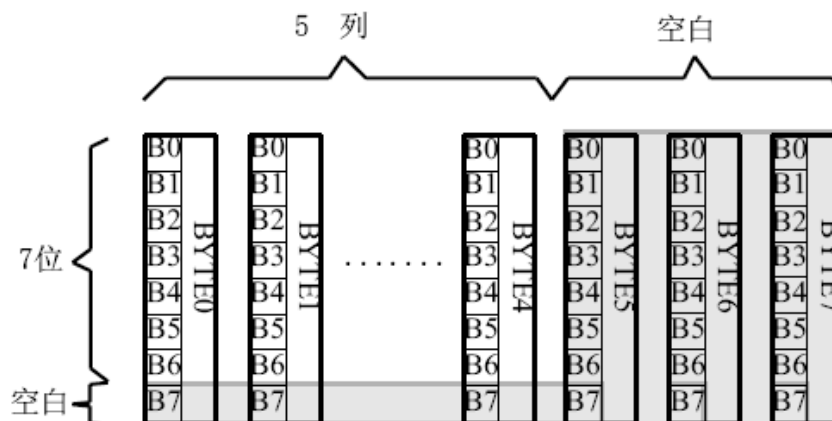
6.1.1 15X16 点汉字排列格式

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：



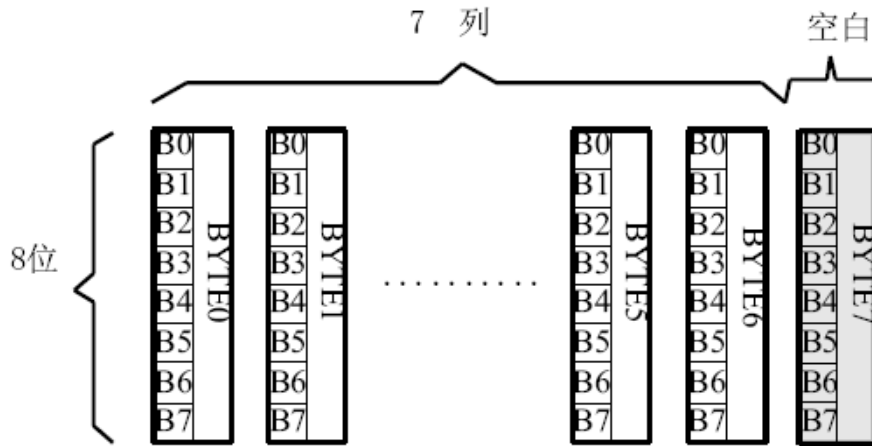
6.1.2 5X7 点 ASCII 字符排列格式

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

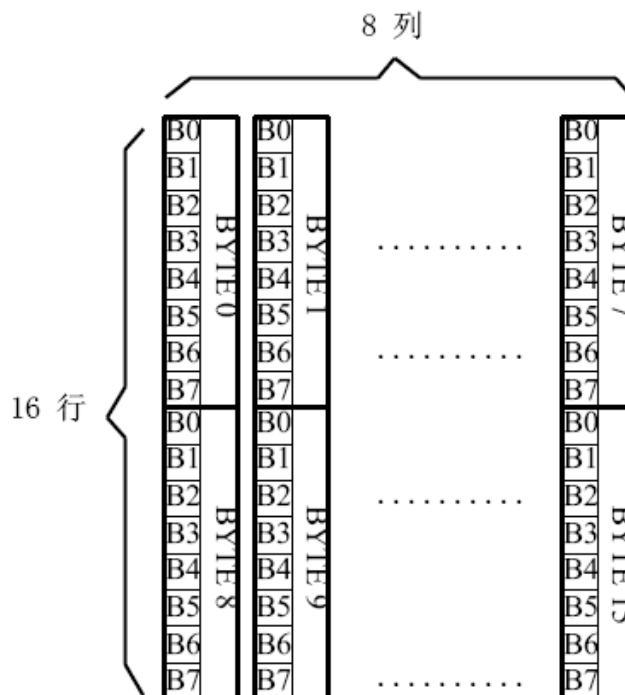
适用于此种排列格式的字有:

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。

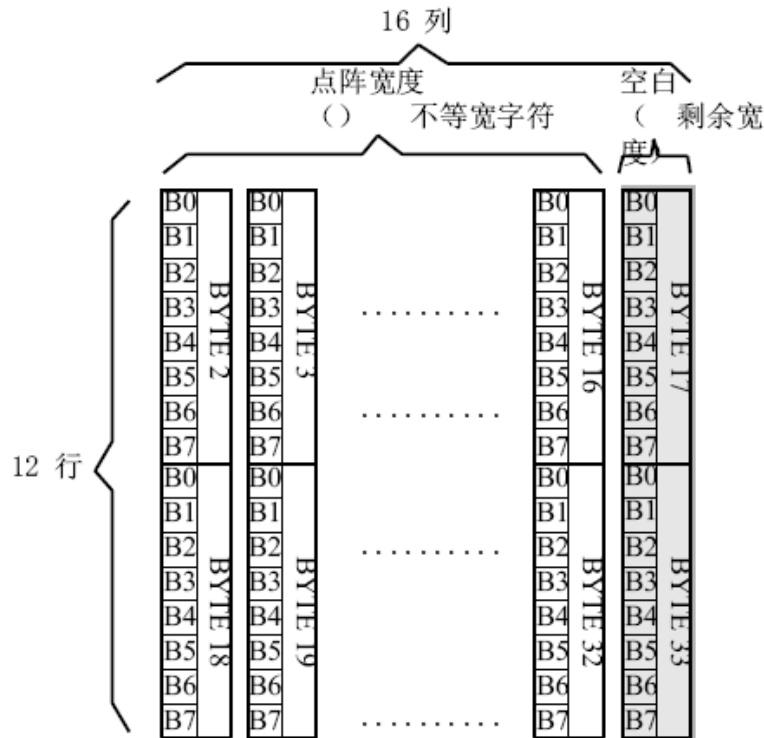
■ 存储格式

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 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:

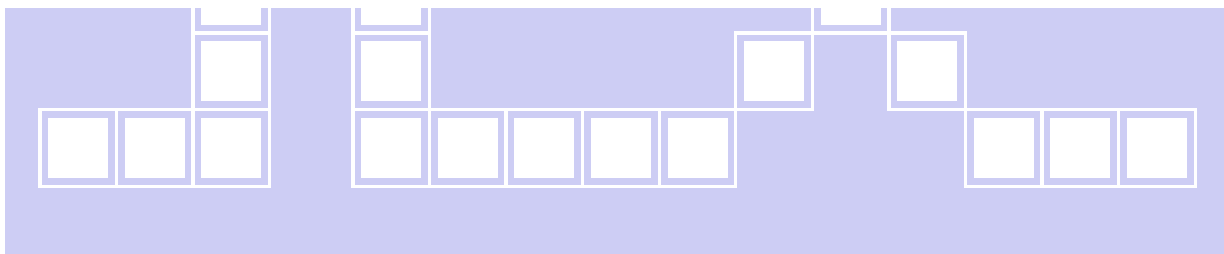
ASCIICode: 表示 ASCII 码 (8bits)

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。

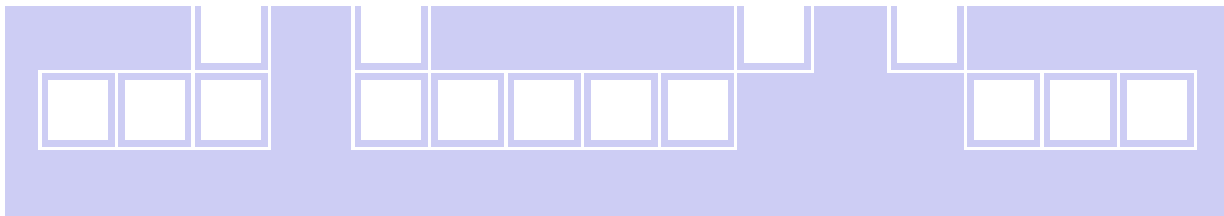
Address: ASCII 字符点阵在芯片中的字节地址。

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	∩	⊙	∫	∫	≡	≈	≈	∞	∞	≠	≠	≠	≠	∞	∞	∞
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:

点亮液晶模块的步骤

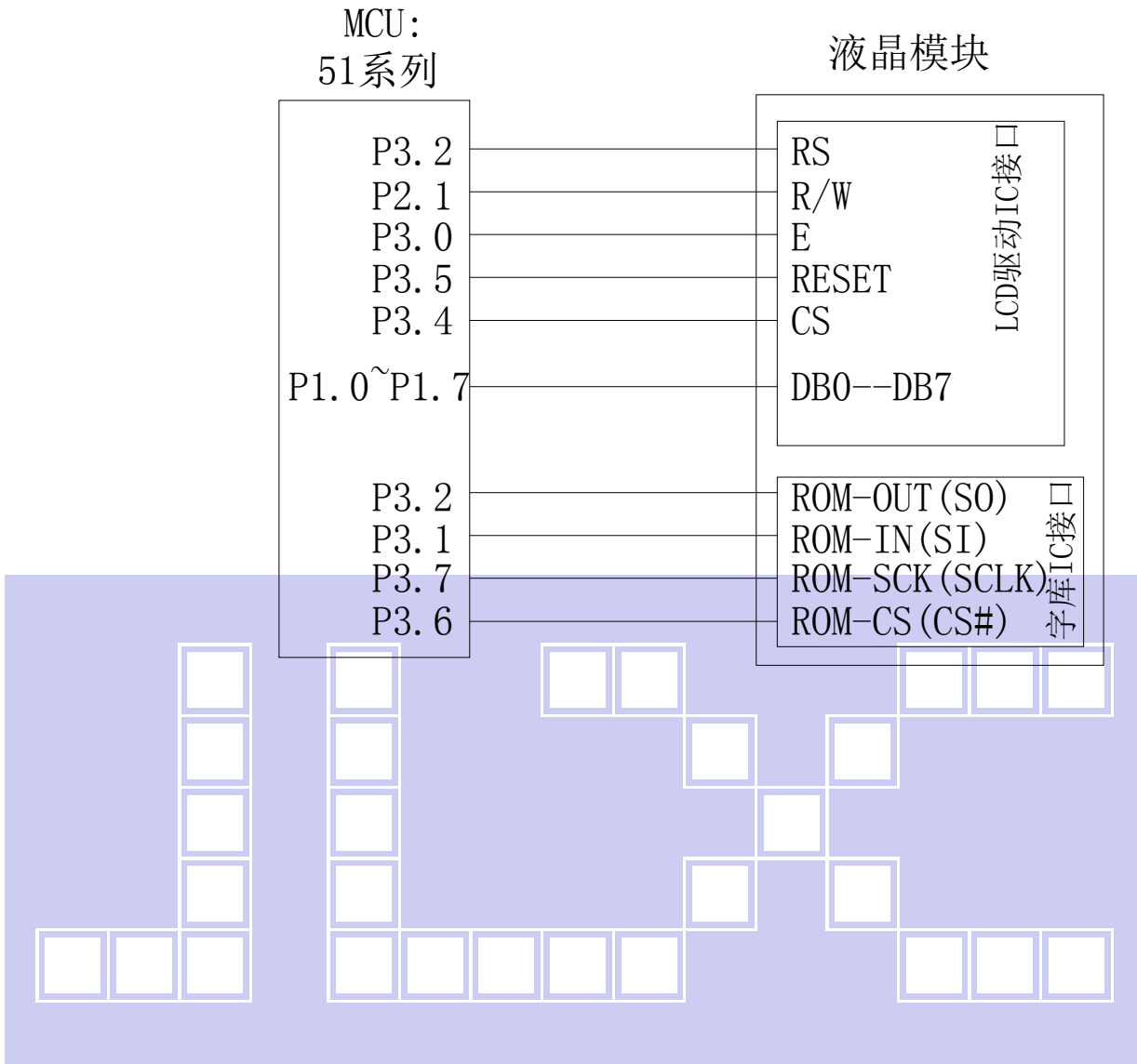
硬件准备:
开发板 (或专门设计的主板)、单片机、电源、连接线、仿真器或程序下载器 (又名烧录器)

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

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

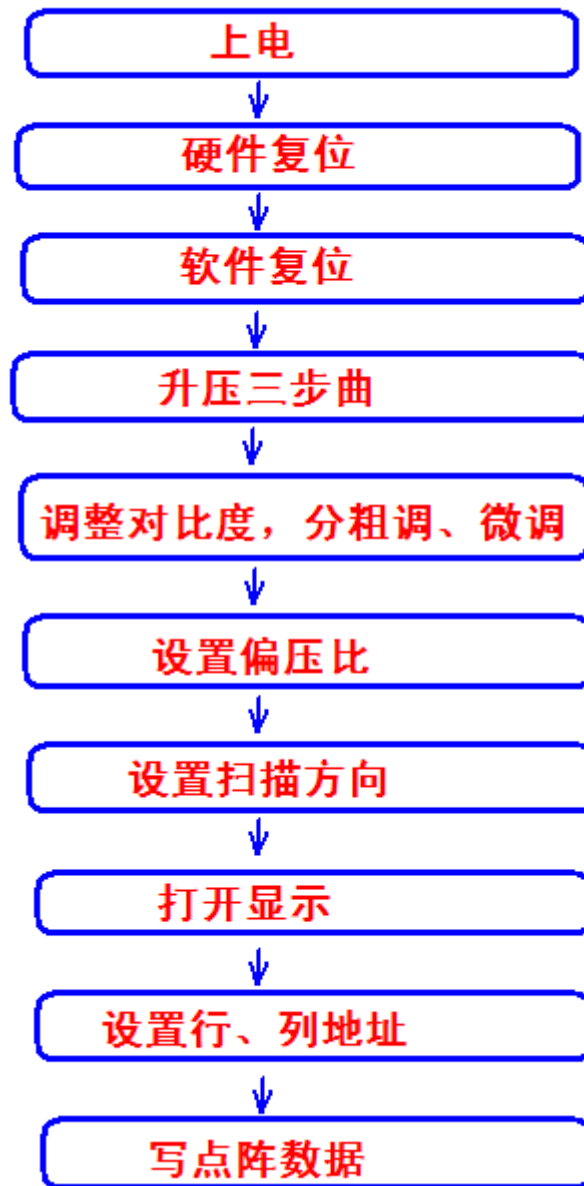


7.1.1 硬件接口： 下图为并行方式的硬件接口：



7.2 程序

点亮液晶模块的编程步骤



7.2.2 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-923-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>

sbit key=P2^0;
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 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
    
```

```

uchar code Chinese_text_16x16[];
uchar code Chinese_code_16x16[];
uchar code jing2[];
uchar code lian2[];
uchar code xun2[];
uchar code dian2[];
uchar code zi2[];
uchar code bmp1[];
uchar code ascii_table_8x16[95][16];
uchar code ascii_table_5x8[95][5];
    
```

```

/*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(5000);
}
    
```



}

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

void transfer_command(int data1)

```

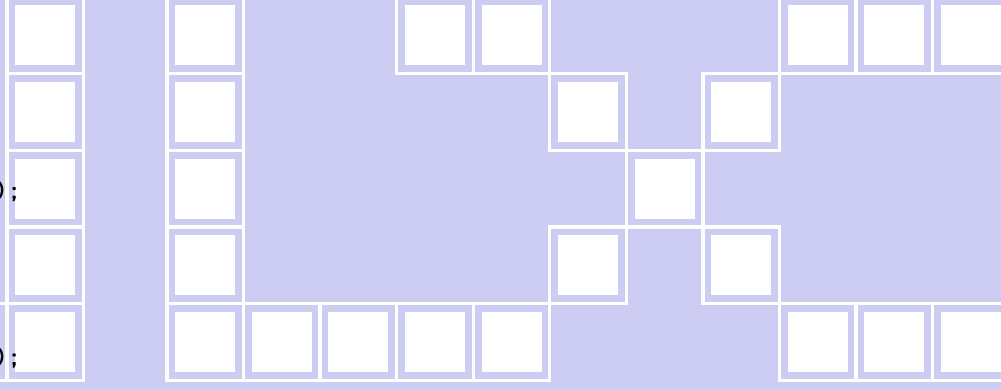
{
    cs1=0;
    rs=0;
    rd=0;
    delay_us(1);
    wr=0;
    P1=data1;
    rd=1;
    delay_us(1);
    cs1=1;
    rd=0;
}
    
```

//-----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 initial_lcd()

```

{
    reset=0;
    delay(100);
    reset=1;
    delay(100);

    transfer_command(0x30); //EXT1=0, EXT0=0,表示选择了“扩展指令表1”
    transfer_command(0x94); //退出睡眠
    transfer_command(0x31); //EXT=1
    transfer_command(0xD7); //Autoread disable
    transfer_data(0X9F); //

    transfer_command(0x32); //LCD 偏压比设置
}
    
```

```

transfer_data(0x00); //振荡频率的调整
transfer_data(0x01); //升压电容器的频率->6KHz
transfer_data(0x03); //Bias=1/11

transfer_command(0x20); //灰度设置
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); // EXT1=0, EXT0=0, 表示选择了“扩展指令表 1”
transfer_command(0x75); // 页地址设置
transfer_data(0x00); // 起始页地址: YS=0x00
transfer_data(0x14); // 结束页地址: YE=0x1F 每 4 行为一页, 第 0~3 行为第 0 页, 第 124~127 行为第 31 页 (31=0x1f)
transfer_command(0x15); // 列地址设置
transfer_data(0x00); // 起始列地址: XS=0
transfer_data(0xbf); // 结束列地址: XE=256 (0xff)

transfer_command(0xBC); //数据扫描方向
transfer_data(0x00); //MX. MY=Normal
transfer_data(0xA6);

transfer_command(0x0c); //数据格式选择, 0x0C 是低位在前 D0-D7, 0x08 是高位在前 D7-D0

transfer_command(0xCA); //显示控制
transfer_data(0x00); //设置 CL 驱动频率: CLD=0
transfer_data(0x5F); //占空比: Duty=128
transfer_data(0x20); //N 行反显: Nline=off

transfer_command(0xF0); //显示模式
transfer_data(0x10); //如果设为 0x11: 表示选择 4 灰度级模式, 如果设为 0x10: 表示选择黑白模式

transfer_command(0x81); //设置对比度, “0x81” 不可改动, 紧跟着的 2 个数据是可改的, 但“先微调后粗调”这个顺序别乱了
transfer_data(0x20); //对比度微调, 可调范围 0x00~0x3f, 共 64 级
    
```



```

transfer_data(0x03);          //对比度粗调, 可调范围 0x00~0x07, 共 8 级
transfer_command(0x20);      //Power control
transfer_data(0x0B);         //D0=regulator ; D1=follower ; D3=booste,  on:1 off:0
delay_us(100);
transfer_command(0xAF);      //打开显示
}
    
```

/*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(1, 1, 256, 17);
    for(i=0; i<17; i++)
    {
        for(j=0; j<256; j++)
        {
            transfer_data(0x00);
        }
    }
}
    
```

/*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(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 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 column, uchar page, uchar reverse, uchar *text)

```

```

{

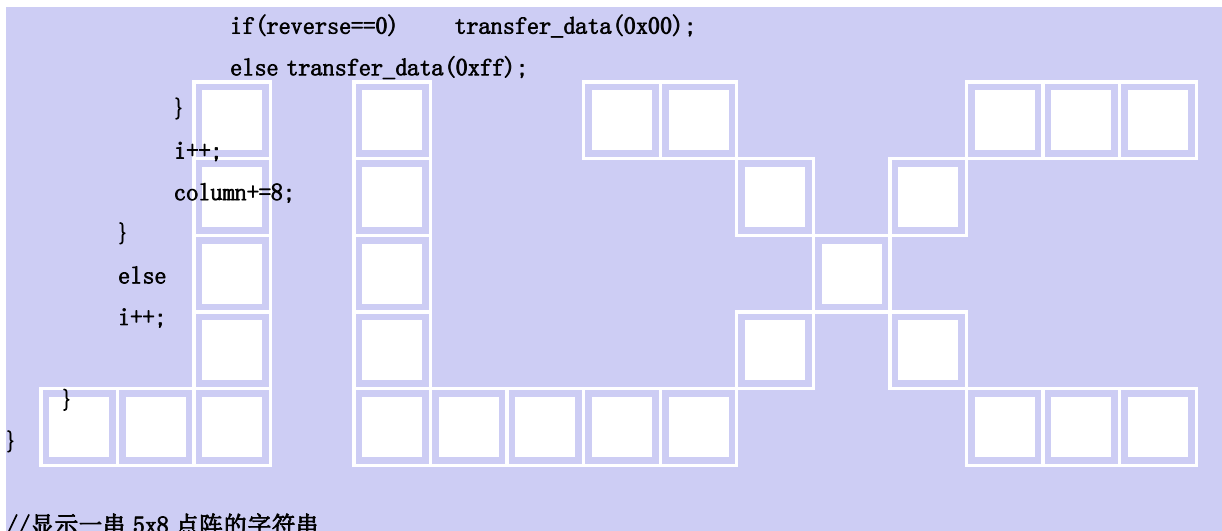
```

```

uchar data1;
uint i=0, j, k, n;

while(text[i]>0x00)
{
    if((text[i]>=0x20)&&(text[i]<=0x7e))
    {
        j=text[i]-0x20;
        lcd_address(column, page, 9, 2);
        for(n=0;n<2;n++)
        {
            for(k=0;k<8;k++)
            {
                if(reverse==1) data1=~ascii_table_8x16[j][k+8*n];
                else data1=ascii_table_8x16[j][k+8*n];
                transfer_data(data1);
            }

```



//显示一串 5x8 点阵的字符串

//括号里的参数分别为（页，列，是否反显，数据指针）

```
void display_string_5x8(uint column,uint page,uchar reverse,uchar *text)
```

```

{
    uchar i=0, j, k, data1;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(column, page, 7, 1);
            for(k=0;k<5;k++)
            {
                if(reverse==1) data1=~ascii_table_5x8[j][k];
                else data1=ascii_table_5x8[j][k];
                transfer_data(data1);
            }

```

```

        if(reverse==1)    transfer_data(0xff);
        else transfer_data(0x00);
        i++;
        column+=6;
    }
    else
        i++;
}
}

```

/*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 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) ***/

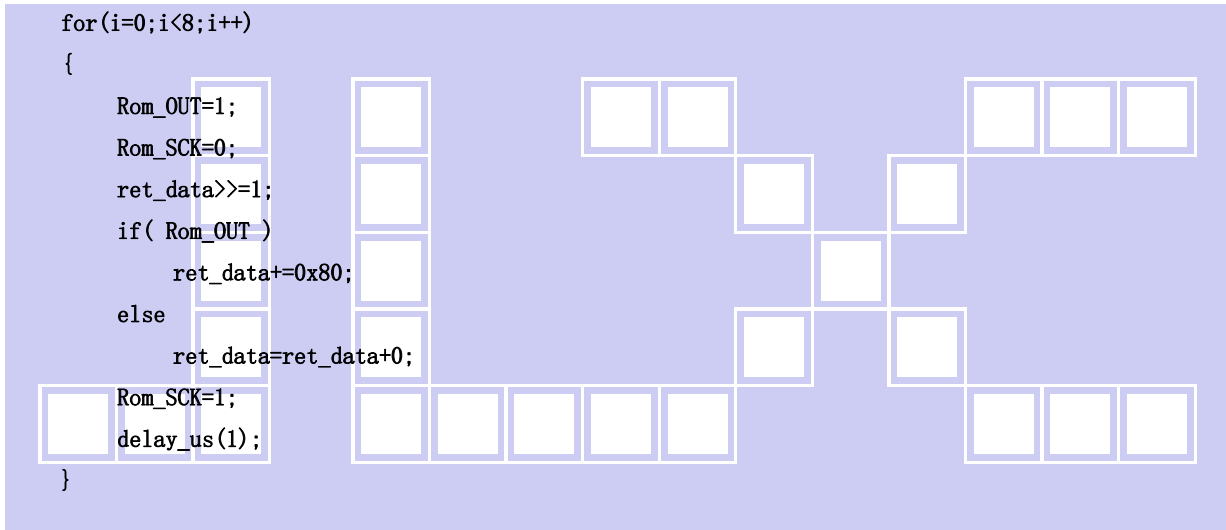
```
static uchar get_data_from_ROM( )
```

```

{
    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<=192)&&column>192)
{
    //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<=192)&&column>182)

{
    //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<=192)&&column>182)
{
    //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();
        transfer_command(0x08);

        display_GB2312_string(1, 1, "深圳市晶联讯电子有限公司");
        display_GB2312_string(1, 3, "JLX19296G-923, 带中文字库");
        display_GB2312_string(1, 5, "GB2312 简体字库及图型功能");
        display_GB2312_string(1, 7, "16X16 简体汉字库, 或 8X16 点");
        display_GB2312_string(1, 9, "阵 ASCII, 或 5X7 点阵 ASCII 码");
        display_GB2312_string(1, 11, "8x16 可以显示 6 行每行 24 个。");
        waitkey();
        transfer_command(0x0c);
        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_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);
display_string_16x16(1, 5, "深圳市晶联讯电子有限公司");
display_string_8x16(1, 7, 1, "JLX19296G-923");
display_string_5x8(1, 9, 0, "JLX19296G-923");
display_string_5x8(1, 11, 0, "01234567890123456789012345678901");
waitkey();
}
}

```

```

uchar code Chinese_text_16x16[]=
{
    "深圳市晶联讯电子有限公司"
};

```

```

uchar code Chinese_code_16x16[]=
{

```

```

/*- 文字: 深 -*/
/*- 新宋体 12: 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x10, 0x61, 0x06, 0xE0, 0x00, 0x26, 0x22, 0x1A, 0x02, 0xC2, 0x0A, 0x12, 0x32, 0x06, 0x02, 0x00,
0x04, 0xFC, 0x03, 0x20, 0x20, 0x11, 0x11, 0x09, 0x05, 0xFF, 0x05, 0x09, 0x19, 0x31, 0x10, 0x00,

```

```

/*- 文字: 圳 -*/
/*- 新宋体 12: 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x10, 0x10, 0x10, 0xFE, 0x10, 0x10, 0xFE, 0x00, 0x00, 0xFC, 0x00, 0x00, 0x00, 0xFE, 0x00, 0x00,
0x08, 0x08, 0x04, 0x47, 0x24, 0x18, 0x07, 0x00, 0x00, 0x1F, 0x00, 0x00, 0x00, 0x7F, 0x00, 0x00,

```

```

/*- 文字: 市 -*/
/*- 新宋体 12: 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x04, 0x04, 0x04, 0xE4, 0x24, 0x24, 0x25, 0xFE, 0x24, 0x24, 0x24, 0x24, 0xE4, 0x04, 0x04, 0x00,
0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x10, 0x20, 0x1F, 0x00, 0x00, 0x00,

```

```

/*- 文字: 晶 -*/
/*- 新宋体 12: 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00, 0x00, 0x00, 0x00, 0x7E, 0x2A, 0x2A, 0x2A, 0x2A, 0x2A, 0x2A, 0x7E, 0x00, 0x00, 0x00, 0x00,
0x00, 0x7F, 0x25, 0x25, 0x25, 0x25, 0x7F, 0x00, 0x00, 0x7F, 0x25, 0x25, 0x25, 0x25, 0x7F, 0x00,

```

```

/*- 文字: 联 -*/
/*- 新宋体 12: 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x02, 0xFE, 0x92, 0x92, 0x92, 0xFE, 0x12, 0x11, 0x12, 0x1C, 0xF0, 0x18, 0x17, 0x12, 0x10, 0x00,
0x08, 0x1F, 0x08, 0x08, 0x04, 0xFF, 0x05, 0x81, 0x41, 0x31, 0x0F, 0x11, 0x21, 0xC1, 0x41, 0x00,

```

```

/*- 文字: 讯 -*/
/*- 新宋体 12: 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x20, 0x21, 0x2E, 0xE4, 0x00, 0x42, 0x42, 0xFE, 0x42, 0x42, 0x42, 0x02, 0xFE, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x7F, 0x20, 0x10, 0x00, 0x7F, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x40, 0x38, 0x00,

```




```

/*- 文字: 电 -*/
/*- 新宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00, 0x00, 0xF8, 0x48, 0x48, 0x48, 0x48, 0xFF, 0x48, 0x48, 0x48, 0x48, 0xF8, 0x00, 0x00, 0x00,
0x00, 0x00, 0x0F, 0x04, 0x04, 0x04, 0x04, 0x3F, 0x44, 0x44, 0x44, 0x44, 0x4F, 0x40, 0x70, 0x00,

```

```

/*- 文字: 子 -*/
/*- 新宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00, 0x00, 0x02, 0x02, 0x02, 0x02, 0x02, 0xE2, 0x12, 0x0A, 0x06, 0x02, 0x00, 0x80, 0x00, 0x00,
0x01, 0x01, 0x01, 0x01, 0x01, 0x41, 0x81, 0x7F, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00,

```

```

/*- 文字: 有 -*/
/*- 新宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00, 0x04, 0x84, 0x44, 0xE4, 0x34, 0x2C, 0x27, 0x24, 0x24, 0x24, 0xE4, 0x04, 0x04, 0x04, 0x00,
0x02, 0x01, 0x00, 0x00, 0xFF, 0x09, 0x09, 0x09, 0x29, 0x49, 0xC9, 0x7F, 0x00, 0x00, 0x00, 0x00,

```

```

/*- 文字: 限 -*/
/*- 新宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0xFE, 0x02, 0x32, 0x4E, 0x82, 0x00, 0xFE, 0x4A, 0xCA, 0x4A, 0x4A, 0x4A, 0x7E, 0x00, 0x00, 0x00,
0xFF, 0x00, 0x02, 0x04, 0x03, 0x00, 0xFF, 0x40, 0x20, 0x03, 0x0C, 0x12, 0x21, 0x60, 0x20, 0x00,

```

```

/*- 文字: 公 -*/
/*- 新宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00, 0x00, 0x80, 0x40, 0x30, 0x0E, 0x84, 0x00, 0x00, 0x0E, 0x10, 0x60, 0xC0, 0x80, 0x80, 0x00,
0x00, 0x01, 0x20, 0x70, 0x28, 0x24, 0x23, 0x31, 0x10, 0x10, 0x14, 0x78, 0x30, 0x01, 0x00, 0x00,

```

```

/*- 文字: 司 -*/
/*- 新宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00, 0x10, 0x92, 0x92, 0x92, 0x92, 0x92, 0x92, 0x92, 0x92, 0x12, 0x02, 0x02, 0xFE, 0x00, 0x00,
0x00, 0x00, 0x1F, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x0F, 0x00, 0x20, 0x40, 0x3F, 0x00, 0x00,
};

```

```

uchar code jing2[]={
/*- 文字: 晶 -*/
/*- 新宋体 23; 此字体下对应的点阵为: 宽 x 高=32x31 -*/
/*- 高度不是 8 的倍数, 现调整为: 宽度 x 高度=32x32 -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF8, 0xF8, 0xF8, 0xF8, 0x10, 0x10, 0x10, 0x10,
0x10, 0x10, 0x10, 0x10, 0x10, 0xF8, 0xFC, 0xFC, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x7F, 0x7F, 0x3F, 0x3F, 0x91, 0x91, 0x11, 0x11,
0x11, 0x91, 0x11, 0x11, 0x11, 0x3F, 0x3F, 0x3F, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x41, 0x41, 0x41, 0x41, 0x41, 0xFF, 0xFF, 0xFF, 0x03, 0x00,
0x00, 0xFF, 0xFF, 0xFF, 0x41, 0x41, 0x41, 0x41, 0x41, 0xFF, 0xFF, 0xFF, 0x03, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x3F, 0x3F, 0x1F, 0x08, 0x08, 0x08, 0x08, 0x08, 0x1F, 0x1F, 0x1F, 0x00, 0x00,
0x00, 0x3F, 0x3F, 0x1F, 0x08, 0x08, 0x08, 0x08, 0x08, 0x1F, 0x1F, 0x1F, 0x00, 0x00, 0x00, 0x00,
};

```



```
uchar code lian2[]={
/*-- 文字：联 --*/
/*-- 新宋体 23； 此字体下对应的点阵为：宽 x 高=32x31 --*/
/*-- 高度不是 8 的倍数，现调整为：宽度 x 高度=32x32 --*/
0x00, 0x20, 0x20, 0xE0, 0xE0, 0xE0, 0x20, 0x20, 0x20, 0xE0, 0xE0, 0xF0, 0x38, 0x18, 0x30, 0x2C,
0x3C, 0xF8, 0xF0, 0xE0, 0x00, 0x80, 0xE0, 0xFC, 0x7C, 0x1C, 0x08, 0x08, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x08, 0x08, 0x08, 0xFF, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x04, 0x04, 0x04, 0xFC, 0xFE, 0xFF, 0x07, 0x04, 0x06, 0x07, 0x83, 0xC7, 0xC6, 0x80, 0x00, 0x00,
0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x82, 0xC2, 0xC2, 0xFF, 0xFF, 0xFF, 0x21, 0x21, 0x21, 0x01,
0x01, 0x81, 0xF9, 0xFF, 0x7F, 0x1F, 0xFF, 0xF1, 0xC1, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00,
0x00, 0x01, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x3F, 0x3F, 0x3F, 0x20, 0x30, 0x18, 0x1C,
0x0E, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00, 0x03, 0x07, 0x0F, 0x1E, 0x1C, 0x18, 0x18, 0x00, 0x00,
};
```

```
uchar code xun2[]={
/*-- 文字：讯 --*/
/*-- 新宋体 23； 此字体下对应的点阵为：宽 x 高=32x31 --*/
/*-- 高度不是 8 的倍数，现调整为：宽度 x 高度=32x32 --*/
0x00, 0x00, 0x00, 0x00, 0x08, 0x18, 0x78, 0xF8, 0xF0, 0x60, 0x20, 0x20, 0x20, 0x20, 0x20, 0xA0,
0xA0, 0xA0, 0x20, 0x20, 0x20, 0x20, 0xF0, 0xF8, 0xF0, 0x20, 0x00, 0x00, 0x00, 0x00,
0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0xF0, 0xF0, 0xF0, 0x30, 0xA0, 0x80, 0x80, 0x80, 0x80, 0xFF,
0xFF, 0xFF, 0x81, 0xC0, 0xE0, 0x60, 0xC0, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x80, 0xC0, 0xE0, 0x60, 0x20, 0x00, 0xFF,
0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xFF, 0xFF, 0xC0, 0x80, 0xF0, 0xF0, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x0F, 0x07, 0x03, 0x00, 0x00, 0x00, 0x00, 0x1F,
0x1F, 0x1F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x0F, 0x1F, 0x1F, 0x1F, 0x00, 0x00,
};
```



```
uchar code dian2[]={
/*-- 文字：电 --*/
/*-- 新宋体 23； 此字体下对应的点阵为：宽 x 高=32x31 --*/
/*-- 高度不是 8 的倍数，现调整为：宽度 x 高度=32x32 --*/
0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0xFC,
0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0xFF, 0xFF, 0xFF,
0x41, 0x41, 0x41, 0x41, 0x41, 0x41, 0xFF, 0xFF, 0xFF, 0xFF, 0x02, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x7F, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0xFF, 0xFF, 0xFF,
0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x3F, 0x3F, 0x3F, 0x3F, 0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x1F, 0x1F,
0x18, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x18, 0x1F, 0x1F, 0x1E, 0x08, 0x00, 0x00,
};
```

```
uchar code zi2[]={
```



```

0x00, 0x18, 0x04, 0x84, 0x44, 0x7C, 0xA5, 0x26, 0x24, 0xA4, 0x64, 0x24, 0x14, 0x0C, 0x04, 0x00,
0x00, 0x00, 0x00, 0xF8, 0x88, 0x88, 0x88, 0x89, 0x8A, 0x8E, 0x88, 0x88, 0x88, 0xF8, 0x00, 0x00,
0x00, 0x02, 0x42, 0x62, 0x52, 0x4A, 0x26, 0xE2, 0x22, 0x22, 0x2A, 0x72, 0x22, 0x02, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0x40, 0x40, 0x40, 0x40, 0x40, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xFE, 0x02, 0x0A, 0xCA, 0x4A, 0x6A, 0x5E, 0x49, 0x49, 0xC9, 0x09, 0x01, 0x00, 0x00,
0x40, 0x40, 0x40, 0xDF, 0x55, 0x55, 0x55, 0xD5, 0x55, 0x55, 0x55, 0xDF, 0x40, 0x40, 0x40, 0x00,
0x10, 0x08, 0x04, 0xD3, 0x56, 0x5A, 0x52, 0x5A, 0xF4, 0x53, 0x56, 0x5A, 0x72, 0x02, 0x02, 0x00,
0x00, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x0F, 0x11, 0x20, 0x20, 0x11, 0x0E, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,
0x08, 0x08, 0x05, 0x04, 0xFC, 0x46, 0x46, 0x45, 0x45, 0x46, 0x46, 0xFE, 0x04, 0x0C, 0x04, 0x00,
0x80, 0x40, 0x30, 0x0F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x20, 0x20, 0x22, 0x22, 0x22, 0x22, 0x22, 0x3F, 0x22, 0x22, 0x22, 0x22, 0x22, 0x20, 0x20, 0x00,
0x00, 0x40, 0x40, 0x40, 0x40, 0x40, 0x40, 0x7F, 0x40, 0x40, 0x40, 0x40, 0x40, 0x60, 0x40, 0x00,
0x00, 0x00, 0x58, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x80, 0x60, 0x1F, 0x00, 0x80, 0x4F, 0x20, 0x10, 0x0F, 0x10, 0x20, 0xEF, 0x40, 0x00, 0x00, 0x00,
0x40, 0x40, 0x40, 0x57, 0x55, 0x55, 0x7F, 0x55, 0x55, 0x55, 0x57, 0x50, 0x40, 0x40, 0x00,
0x00, 0x20, 0x20, 0x23, 0x12, 0x12, 0x0A, 0x06, 0xFF, 0x02, 0x02, 0x12, 0x22, 0x1E, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x18, 0x24, 0x24, 0x18, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};

```

```
uchar code ascii_table_8x16[95][16]={
```

//粗体 8x16 点阵的 ASCII 码的点阵数据，从“JLX-GB2312”型号的字库 IC 中读出来的国标的。

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //-(即“空格”) ASCII 码: 0X20
0x00, 0x00, 0x38, 0xFC, 0xFC, 0x38, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x0D, 0x00, 0x00, 0x00, 0x00, //!- ASCII 码: 0X21
0x00, 0x0E, 0x1E, 0x00, 0x00, 0x1E, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, //"-
0x20, 0xF8, 0xF8, 0x20, 0xF8, 0xF8, 0x20, 0x00, 0x02, 0x0F, 0x0F, 0x02, 0x0F, 0x0F, 0x02, 0x00, // #-
0x38, 0x7C, 0x44, 0x47, 0x47, 0xCC, 0x98, 0x00, 0x06, 0x0C, 0x08, 0x38, 0x38, 0x0F, 0x07, 0x00, // $-
0x30, 0x30, 0x00, 0x80, 0xC0, 0x60, 0x30, 0x00, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x0C, 0x0C, 0x00, // %-
0x80, 0xD8, 0x7C, 0xE4, 0xBC, 0xD8, 0x40, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, // &-
0x00, 0x10, 0x1E, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ' -
0x00, 0x00, 0xF0, 0xF8, 0x0C, 0x04, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x00, 0x00, // (-
0x00, 0x00, 0x04, 0x0C, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0C, 0x07, 0x03, 0x00, 0x00, // )-

0x80, 0xA0, 0xE0, 0xC0, 0xC0, 0xE0, 0xA0, 0x80, 0x00, 0x02, 0x03, 0x01, 0x01, 0x03, 0x02, 0x00, // *- ASCII 码: 0X2A
0x00, 0x80, 0x80, 0xE0, 0xE0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00, 0x00, 0x00, 0x00, // +-
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x1E, 0x0E, 0x00, 0x00, 0x00, 0x00, // , -
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // --
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0C, 0x00, 0x00, 0x00, 0x00, // . -
0x00, 0x00, 0x00, 0x80, 0xC0, 0x60, 0x30, 0x00, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, // / -
0xF8, 0xF8, 0x0C, 0xC4, 0x0C, 0xF8, 0xF0, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x0C, 0x07, 0x03, 0x00, // 0- ASCII 码: 0X30

```





0x00, 0x10, 0x18, 0xFC, 0xFC, 0x00, 0x00, 0x00, 0x08, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x00, // -1-
 0x08, 0x0C, 0x84, 0xC4, 0x64, 0x3C, 0x18, 0x00, 0x0E, 0x0F, 0x09, 0x08, 0x08, 0x0C, 0x0C, 0x00, // -2-
 0x08, 0x0C, 0x44, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x04, 0x0C, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // -3-

0xC0, 0xE0, 0xB0, 0x98, 0xFC, 0xFC, 0x80, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, // -4-
 0x7C, 0x7C, 0x44, 0x44, 0x44, 0xC4, 0x84, 0x00, 0x04, 0x0C, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // -5-
 0xF0, 0xF8, 0x4C, 0x44, 0x44, 0xC0, 0x80, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // -6-
 0x0C, 0x0C, 0x04, 0x84, 0xC4, 0x7C, 0x3C, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00, // -7-
 0xB8, 0xFC, 0x44, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // -8-
 0x38, 0x7C, 0x44, 0x44, 0x44, 0xFC, 0xF8, 0x00, 0x00, 0x08, 0x08, 0x08, 0x0C, 0x07, 0x03, 0x00, // -9-
 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x06, 0x06, 0x00, 0x00, 0x00, // -:-
 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x08, 0x0E, 0x06, 0x00, 0x00, 0x00, // -;-
 0x00, 0x80, 0xC0, 0x60, 0x30, 0x18, 0x08, 0x00, 0x00, 0x00, 0x01, 0x03, 0x06, 0x0C, 0x08, 0x00, // -<-
 0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, // ==-

ASCII 码: 0X34

0x00, 0x08, 0x18, 0x30, 0x60, 0xC0, 0x80, 0x00, 0x00, 0x08, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x00, // ->-
 0x18, 0x1C, 0x04, 0xC4, 0xE4, 0x3C, 0x18, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x0D, 0x00, 0x00, 0x00, // -?-
 0xF0, 0xF0, 0x08, 0xC8, 0xC8, 0xF8, 0xF0, 0x00, 0x07, 0x0F, 0x08, 0x0B, 0x0B, 0x0B, 0x01, 0x00, // -@-
 0xE0, 0xF0, 0x98, 0x8C, 0x98, 0xF0, 0xE0, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, // -A-
 0x04, 0xFC, 0xFC, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0F, 0x07, 0x00, // -B-
 0xF0, 0xF8, 0x0C, 0x04, 0x04, 0x0C, 0x18, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x08, 0x0C, 0x06, 0x00, // -C-
 0x04, 0xFC, 0xFC, 0x04, 0x0C, 0xF8, 0xF0, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x0C, 0x07, 0x03, 0x00, // -D-
 0x04, 0xFC, 0xFC, 0x44, 0xE4, 0x0C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0C, 0x0E, 0x00, // -E-
 0x04, 0xFC, 0xFC, 0x44, 0xE4, 0x0C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, // -F-
 0xF0, 0xF8, 0x0C, 0x84, 0x84, 0x8C, 0x98, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x08, 0x07, 0x0F, 0x00, // -G-

ASCII 码: 0X3E

0xFC, 0xFC, 0x40, 0x40, 0x40, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, // -H-
 0x00, 0x00, 0x04, 0xFC, 0xFC, 0x04, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, // -I-
 0x00, 0x00, 0x00, 0x04, 0xFC, 0xFC, 0x04, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x0F, 0x07, 0x00, 0x00, // -J-
 0x04, 0xFC, 0xFC, 0xC0, 0xE0, 0x3C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x01, 0x0F, 0x0E, 0x00, // -K-
 0x04, 0xFC, 0xFC, 0x04, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0C, 0x0E, 0x00, // -L-
 0xFC, 0xFC, 0x38, 0x70, 0x38, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, // -M-
 0xFC, 0xFC, 0x38, 0x70, 0xE0, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, // -N-
 0xF8, 0xFC, 0x04, 0x04, 0x04, 0xFC, 0xF8, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // -O-
 0x04, 0xFC, 0xFC, 0x44, 0x44, 0x7C, 0x38, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, // -P-
 0xF8, 0xFC, 0x04, 0x04, 0x04, 0xFC, 0xF8, 0x00, 0x07, 0x0F, 0x08, 0x0E, 0x3C, 0x3F, 0x27, 0x00, // -Q-

ASCII 码: 0X48

0x04, 0xFC, 0xFC, 0x44, 0xC4, 0xFC, 0x38, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x00, 0x0F, 0x0F, 0x00, // -R-
 0x18, 0x3C, 0x64, 0x44, 0xC4, 0x9C, 0x18, 0x00, 0x06, 0x0E, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // -S-
 0x00, 0x1C, 0x0C, 0xFC, 0xFC, 0x0C, 0x1C, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, // -T-
 0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // -U-
 0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x01, 0x03, 0x06, 0x0C, 0x06, 0x03, 0x01, 0x00, // -V-
 0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x07, 0x0F, 0x0E, 0x03, 0x0E, 0x0F, 0x07, 0x00, // -W-
 0x0C, 0x3C, 0xF0, 0xE0, 0xF0, 0x3C, 0x0C, 0x00, 0x0C, 0x0F, 0x03, 0x01, 0x03, 0x0F, 0x0C, 0x00, // -X-
 0x00, 0x0C, 0x7C, 0xC0, 0xC0, 0x7C, 0x3C, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, // -Y-
 0x1C, 0x0C, 0x84, 0xC4, 0x64, 0x3C, 0x1C, 0x00, 0x0E, 0x0F, 0x09, 0x08, 0x08, 0x0C, 0x0E, 0x00, // -Z-



ASCII 码: 0X41



```
0x00, 0x00, 0xFC, 0xFC,  0x04, 0x04, 0x00, 0x00,  0x00, 0x00, 0x0F, 0x0F,  0x08, 0x08, 0x00, 0x00,  //-[-  
  
0x38, 0x70, 0xE0, 0xC0,  0x80, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x01,  0x03, 0x07, 0x0E, 0x00,  //-\-  
0x00, 0x00, 0x04, 0x04,  0xFC, 0xFC, 0x00, 0x00,  0x00, 0x00, 0x08, 0x08,  0x0F, 0x0F, 0x00, 0x00,  //-]-  
0x08, 0x0C, 0x06, 0x03,  0x06, 0x0C, 0x08, 0x00,  0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00,  //-^-  
0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00,  0x20, 0x20, 0x20, 0x20,  0x20, 0x20, 0x20, 0x20,  //-_-  
0x00, 0x00, 0x03, 0x07,  0x04, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00,  //--^  
0x00, 0xA0, 0xA0, 0xA0,  0xE0, 0xC0, 0x00, 0x00,  0x07, 0x0F, 0x08, 0x08,  0x07, 0x0F, 0x08, 0x00,  //-a-  
0x04, 0xFC, 0xFC, 0x20,  0x60, 0xC0, 0x80, 0x00,  0x00, 0x0F, 0x0F, 0x08,  0x08, 0x0F, 0x07, 0x00,  //-b-  
0xC0, 0xE0, 0x20, 0x20,  0x20, 0x60, 0x40, 0x00,  0x07, 0x0F, 0x08, 0x08,  0x08, 0x0C, 0x04, 0x00,  //-c-  
0x80, 0xC0, 0x60, 0x24,  0xFC, 0xFC, 0x00, 0x00,  0x07, 0x0F, 0x08, 0x08,  0x07, 0x0F, 0x08, 0x00,  //-d-  
0xC0, 0xE0, 0xA0, 0xA0,  0xA0, 0xE0, 0xC0, 0x00,  0x07, 0x0F, 0x08, 0x08,  0x08, 0x0C, 0x04, 0x00,  //-e-  
  
0x40, 0xF8, 0xFC, 0x44,  0x0C, 0x18, 0x00, 0x00,  0x08, 0x0F, 0x0F, 0x08,  0x00, 0x00, 0x00, 0x00,  //-f-  
0xC0, 0xE0, 0x20, 0x20,  0xC0, 0xE0, 0x20, 0x00,  0x27, 0x6F, 0x48, 0x48,  0x7F, 0x3F, 0x00, 0x00,  //-g-  
0x04, 0xFC, 0xFC, 0x40,  0x20, 0xE0, 0xC0, 0x00,  0x08, 0x0F, 0x0F, 0x00,  0x00, 0x0F, 0x0F, 0x00,  //-h-  
0x00, 0x00, 0x20, 0xEC,  0xEC, 0x00, 0x00, 0x00,  0x00, 0x00, 0x08, 0x0F,  0x0F, 0x08, 0x00, 0x00,  //-i-  
0x00, 0x00, 0x00, 0x00,  0x20, 0xEC, 0xEC, 0x00,  0x00, 0x30, 0x70, 0x40,  0x40, 0x7F, 0x3F, 0x00,  //-j-  
0x04, 0xFC, 0xFC, 0x80,  0xC0, 0x60, 0x20, 0x00,  0x08, 0x0F, 0x0F, 0x01,  0x03, 0x0E, 0x0C, 0x00,  //-k-  
0x00, 0x00, 0x04, 0xFC,  0xFC, 0x00, 0x00, 0x00,  0x00, 0x00, 0x08, 0x0F,  0x0F, 0x08, 0x00, 0x00,  //-l-  
0xE0, 0xE0, 0x60, 0xC0,  0x60, 0xE0, 0xC0, 0x00,  0x0F, 0x0F, 0x00, 0x07,  0x00, 0x0F, 0x0F, 0x00,  //-m-  
0x20, 0xE0, 0xC0, 0x20,  0x20, 0xE0, 0xC0, 0x00,  0x00, 0x0F, 0x0F, 0x00,  0x00, 0x0F, 0x0F, 0x00,  //-n-  
0xC0, 0xE0, 0x20, 0x20,  0x20, 0xE0, 0xC0, 0x00,  0x07, 0x0F, 0x08, 0x08,  0x08, 0x0F, 0x07, 0x00,  //-o-  
  
0x20, 0xE0, 0xC0, 0x20,  0x20, 0xE0, 0xC0, 0x00,  0x40, 0x7F, 0x7F, 0x48,  0x08, 0x0F, 0x07, 0x00,  //-p-  
0xC0, 0xE0, 0x20, 0x20,  0xC0, 0xE0, 0x20, 0x00,  0x07, 0x0F, 0x08, 0x48,  0x7F, 0x7F, 0x40, 0x00,  //-q-  
0x20, 0xE0, 0xC0, 0x60,  0x20, 0xE0, 0xC0, 0x00,  0x08, 0x0F, 0x0F, 0x08,  0x00, 0x00, 0x00, 0x00,  //-r-  
0x40, 0xE0, 0xA0, 0x20,  0x20, 0x60, 0x40, 0x00,  0x04, 0x0C, 0x09, 0x09,  0x0B, 0x0E, 0x04, 0x00,  //-s-  
0x20, 0x20, 0xF8, 0xFC,  0x20, 0x20, 0x00, 0x00,  0x00, 0x00, 0x07, 0x0F,  0x08, 0x0C, 0x04, 0x00,  //-t-  
0xE0, 0xE0, 0x00, 0x00,  0xE0, 0xE0, 0x00, 0x00,  0x07, 0x0F, 0x08, 0x08,  0x07, 0x0F, 0x08, 0x00,  //-u-  
0x00, 0xE0, 0xE0, 0x00,  0x00, 0xE0, 0xE0, 0x00,  0x00, 0x03, 0x07, 0x0C,  0x0C, 0x07, 0x03, 0x00,  //-v-  
0xE0, 0xE0, 0x00, 0x80,  0x00, 0xE0, 0xE0, 0x00,  0x07, 0x0F, 0x0C, 0x07,  0x0C, 0x0F, 0x07, 0x00,  //-w-  
0x20, 0x60, 0xC0, 0x80,  0xC0, 0x60, 0x20, 0x00,  0x08, 0x0C, 0x07, 0x03,  0x07, 0x0C, 0x08, 0x00,  //-x-  
0xE0, 0xE0, 0x00, 0x00,  0x00, 0xE0, 0xE0, 0x00,  0x47, 0x4F, 0x48, 0x48,  0x68, 0x3F, 0x1F, 0x00,  //-y-  
  
0x60, 0x60, 0x20, 0xA0,  0xE0, 0x60, 0x20, 0x00,  0x0C, 0x0E, 0x0B, 0x09,  0x08, 0x0C, 0x0C, 0x00,  //-z-  
0x00, 0x40, 0x40, 0xF8,  0xBC, 0x04, 0x04, 0x00,  0x00, 0x00, 0x00, 0x07,  0x0F, 0x08, 0x08, 0x00,  //-{-  
0x00, 0x00, 0x00, 0xBC,  0xBC, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x0F,  0x0F, 0x00, 0x00, 0x00,  //-|-  
0x00, 0x04, 0x04, 0xBC,  0xF8, 0x40, 0x40, 0x00,  0x00, 0x08, 0x08, 0x0F,  0x07, 0x00, 0x00, 0x00,  //-}-  
0x08, 0x0C, 0x04, 0x0C,  0x08, 0x0C, 0x04, 0x00,  0x00, 0x00, 0x00, 0x00,  0x00, 0x00, 0x00, 0x00,  //~-  
  
};
```

ASCII 码: 0x61



ASCII 码: 0x7E

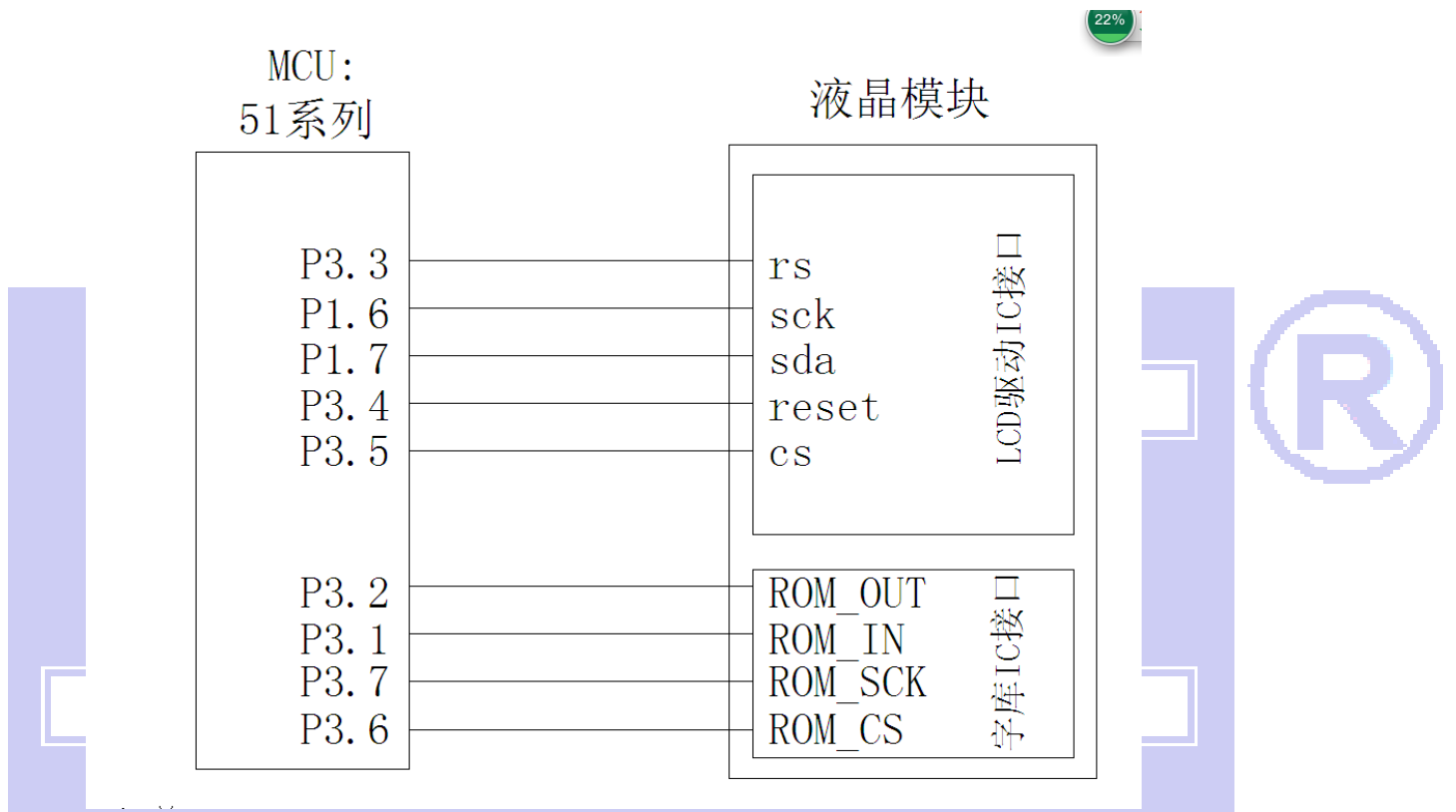
```
uchar code ascii_table_5x8[95][5]={
```



```
0x08, 0x36, 0x41, 0x41, 0x00, //{-{-
0x00, 0x00, 0x77, 0x00, 0x00, //|-|-
0x00, 0x41, 0x41, 0x36, 0x08, //-}-
0x04, 0x02, 0x02, 0x02, 0x01, //~--
};
```

7.3 当 LCD 驱动 IC 采用串行接口方式时的硬件设计及例程:

7.3.1 硬件接口: 下图为串行方式的硬件接口:



7.3.3、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)
```

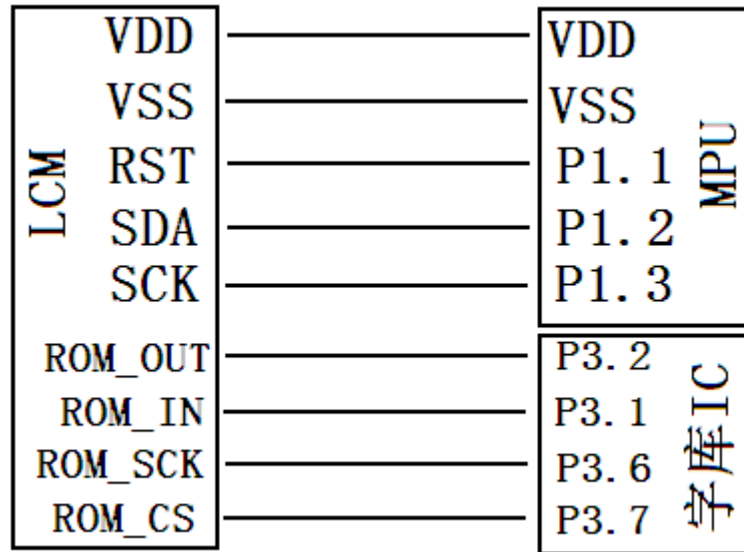
```
{
unsigned char k;
cs1=0;
rs=1;
for(k=0;k<8;k++)
{
dat=dat<<1;
sda=1;
sck=0;
sck=1;
```

```
}
cs1=1;
}
```



7.4 当 LCD 驱动 IC 采用 IIC 接口方式时的硬件设计及例程:

7.4.1 硬件接口: 下图为 IIC 方式的硬件接口:



7.4.2、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：

```
#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=P3^7; /*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*/
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();
}

```

