

JLX144-016-PC 使用说明书

(带字库 IC, 3.3V 供电)

目 录

序号	内 容 标 题	页码
1	字库	2~3
2	外形及接口引脚功能	4~5
3	基本原理	5~6
4	技术参数	6
5	字库 IC 的操作指令及点阵数据的调用方法	7~8
6	字库的使用方法	9~19
7	指令功能及硬件接口与编程案例	20~末页

1. 字库

字库 IC (IC 型号: JLX-GB2312-1602, 此 IC 为可选的配件) 自带字库内容:

分类	字库内容	编码体系 (字符集)	字符数
汉字及字符	11X12 点 GB2312 标准点阵字库	GB2312	6763+376
	15X16 点 GB2312 标准点阵字库	GB2312	6763+376
	6X12 点国标扩展字符	GB2312	126
	8X16 点国标扩展字符	GB2312	126
ASCII 字符	5X7 点 ASCII 字符	ASCII	96
	7X8 点 ASCII 字符	ASCII	96
	6X12 点 ASCII 字符	ASCII	96
	8X16 点 ASCII 字符	ASCII	96
	12 点阵不等宽 ASCII 方头 (Arial) 字符	ASCII	96
	16 点阵不等宽 ASCII 方头 (Arial) 字符	ASCII	96



2. 字型样张:

11X12 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
把把坝霸罢爸白柏百摆佰败拜裨斑班
搬扳般颁扳版扮拌伴瓣半办絆邦帮榔
榜膀绑棒磅蚌镑傍滂苞胞包褓剥薄雹
保堡饱宝抱报暴豹鲍爆杯碑悲卑北辈
背贝钡倍狈备惫焙奔苯本笨崩绷甬

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
把把坝霸罢爸白柏百摆佰败拜裨斑班
搬扳般颁扳版扮拌伴瓣半办絆邦帮榔
榜膀绑棒磅蚌镑傍滂苞胞包褓剥薄雹
保堡饱宝抱报暴豹鲍爆杯碑悲卑北辈
背贝钡倍狈备惫焙奔苯本笨崩绷甬

5x7 点 ASCII 字符

!"#\$%&'()*+,-./0123456789:
=>?@ABCDEFGHIJKLMNPOQRSTUVWXYZ
YZ[\]^_`abcdefghijklmnopqrstuvwxyz

7x8 点 ASCII 字符

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

6x12 点 ASCII 字符

!"#\$%&'()*+,-./0123456789:;
=>?@ABCDEFGHIJKLMNPOQRSTUVW
XYZ[\]^_`abcdefghijklmnopqrstuvwxyz
uvwxyz{|}~āāāāēēēēīīīīōōōōū

8x16 点 ASCII 字符

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

12 点阵不等宽 ASCII 方头

!"#\$%&'()*+,-./0123456789:;<=>?@ABC
DEFGHIJKLMNOPQRSTUVWXYZ[\]^_`
abcdefghijklmnopqrstuvwxyz{|}~
!"#\$%&'()*+,-./0123456789:;<=>?@ABC

16 点阵不等宽 ASCII 方头

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

2. 外形尺寸及接口引脚功能

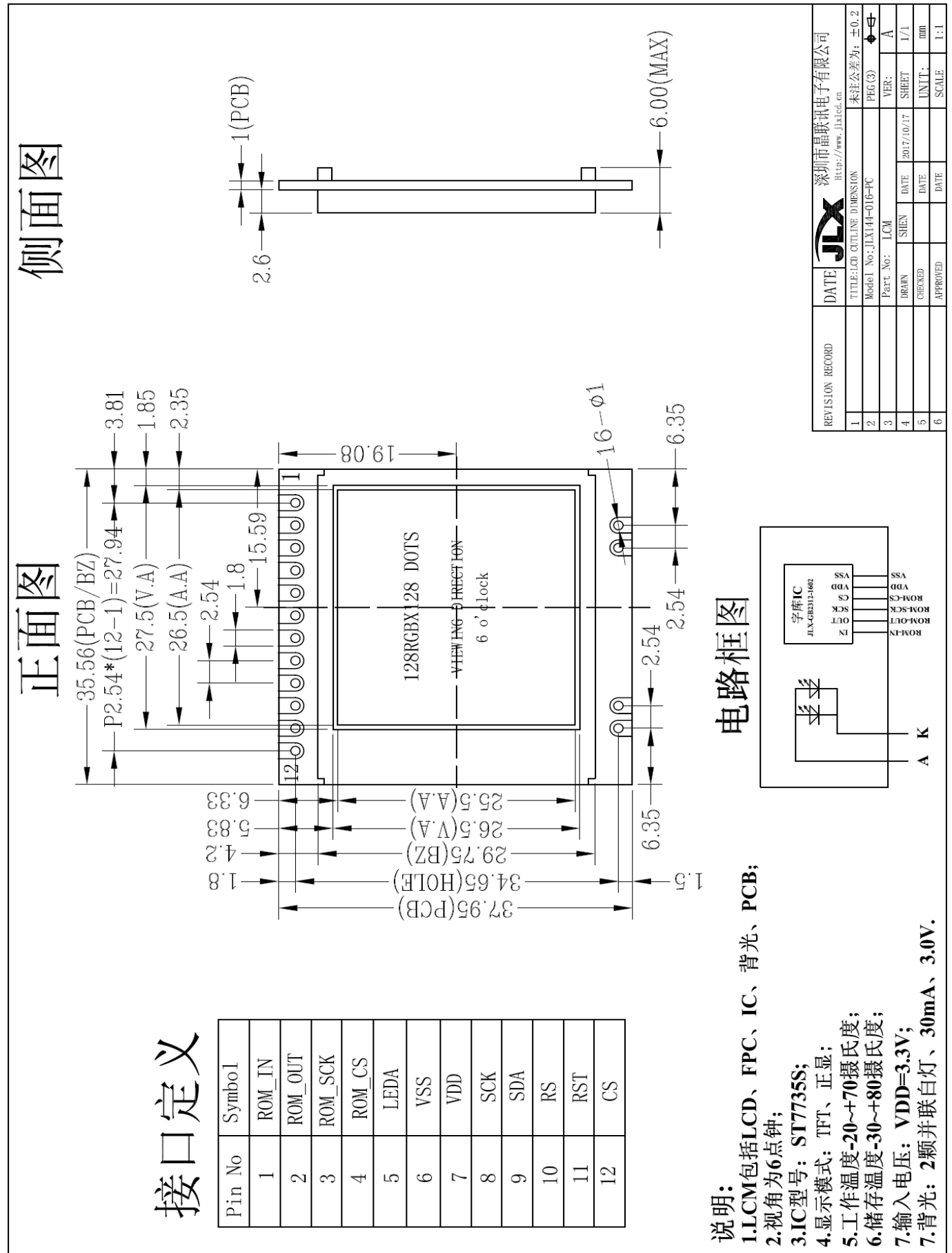


图 1. 外形尺寸

模块的接口引脚功能

表 1：模块的接口引脚功能

引 线 号	符 号	名 称	功 能	
1	ROM_IN	字库 IC 接口 SI	串行数据输入	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	
5	LDEA	背光电源	背光电源正极，（同 VDD 电压 3.3V）	
6	VSS	接地	0V	
7	VDD	电源电路	3.3V	
8	SCK	I/O	串行时钟	
9	SDA	I/O	串行数据	
10	RS	寄存选择信号	H：数据存储器 0：指令存储（IC 资料上缩写为“A0”）	
11	RST	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	
12	CS	片选	低电平片选	

3. 基本原理

3.1 液晶屏 (LCD)

在 LCD 上排列着 160×128 点阵, 160 个列信号与驱动 IC 相连, 128 个行信号也与驱动 IC 相连, IC 邦定在 LCD 玻璃上 (这种加工工艺叫 COG) .

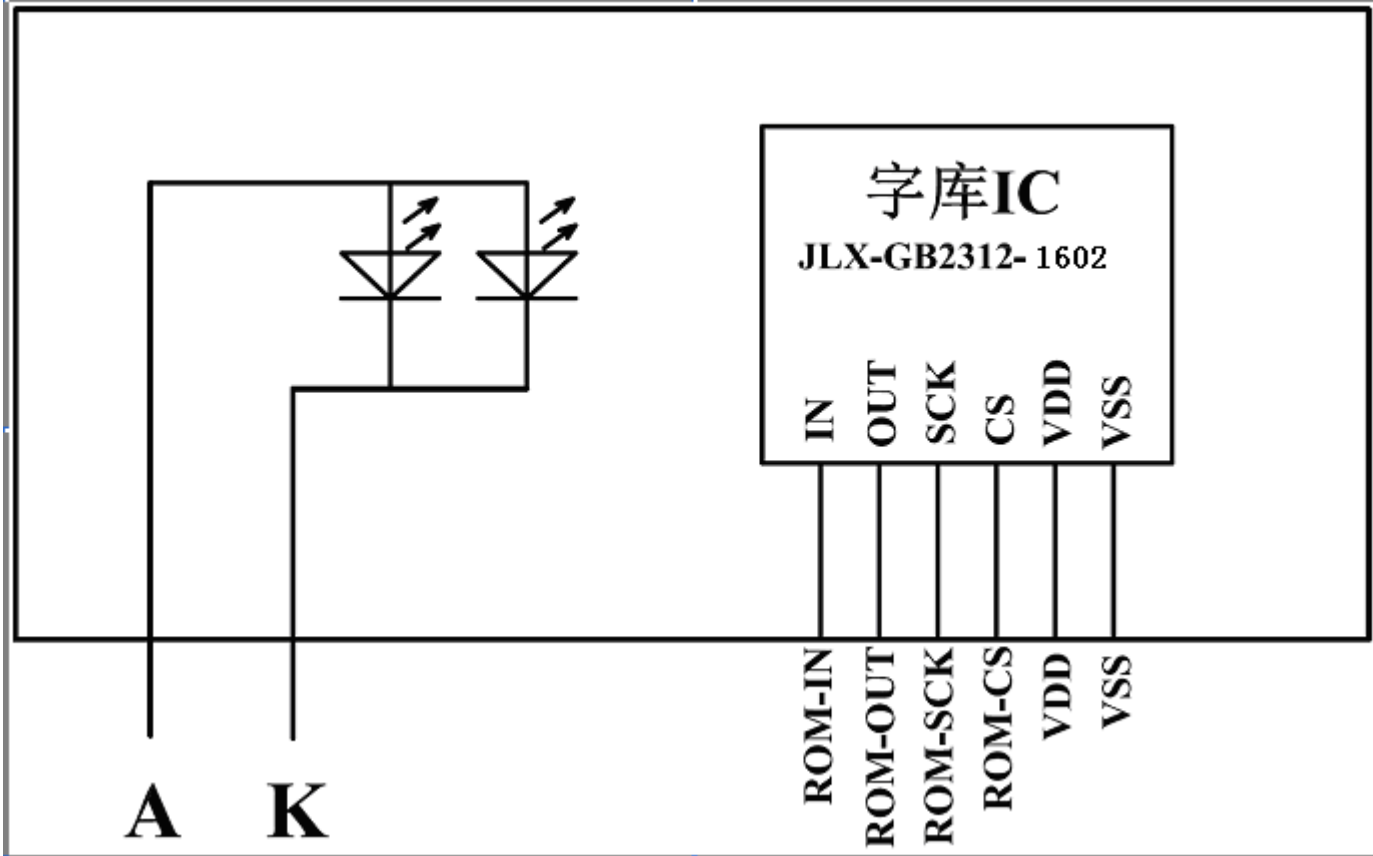


图 2，电路框图

3.3 背光参数

该型号液晶模块带 LED 背光源。它的性能参数如下:

工作温度: $-20^{\circ}\text{C} \sim +70^{\circ}\text{C}$;

存储温度: $-30^{\circ}\text{C} \sim +80^{\circ}\text{C}$;

背光板是白色。

正常工作电流为: $16 \sim 40\text{mA}$ (LED 灯数共 2 颗, 每颗灯是 $8 \sim 20\text{mA}$)

工作电压: 同 VDD 电压 (LED 灯本身的电压是 3.0V , 但是因为在 PCB 上已加了限流电阻, 所以可以同 VDD 电压);

4. 技术参数

4.1 最大极限参数 (超过极限参数则会损坏液晶模块)

名称	符号	标准值			单位
		最小	典型	最大	
电路电源	VDD	-0.3	3.3	3.6	V
工作温度		-20		+70	$^{\circ}\text{C}$
储存温度		-30		+80	$^{\circ}\text{C}$

表 2: 最大极限参数

4.2 直流 (DC) 参数

名称	符号	测试条件	标准值			单位
			最小	典型值	最大	
工作电压	VDD		3.0	3.3	3.6	V
背光工作电压	VLED		2.9	3.0	3.1	V
背光工作电流	ILED	VLED=3.0V, 共 2 颗 LED 灯并联	16	30	40	mA

表 3: 直流 (DC) 参数

4.3 LCD 驱动 IC 指令表详见 “JLX144-016-PN” 的中文说明书

5.1 字库 IC（JLX-GB2312-1602）的操作指令及点阵数据的调用方法：

5.1.1 字库 IC 的操作指令只有两条，两条只选一条进行使用，操作指令表如下：

Instruction Set

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	at Higher Speed	0000 1011	0B h	3	1	1 to ∞

Read Data

Bytes

所有对本芯片 SPI 接口的操作只有 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”)。

以下分别介绍一般读取和快速读取：

6. 2.1.1 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)：

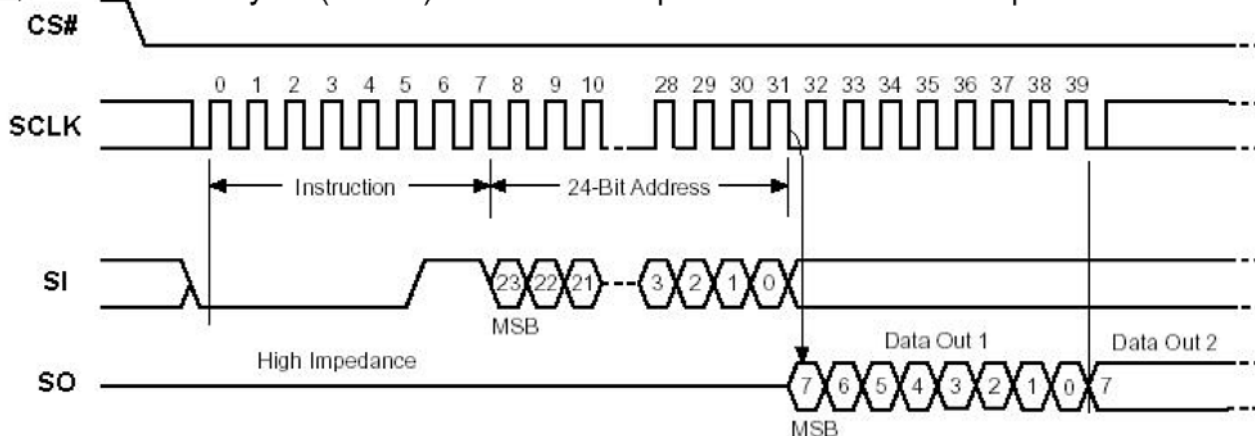
首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（03 h）和 3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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

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



5.2.1.2 Read Data Bytes at Higher Speed (快速读取点阵数据)

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

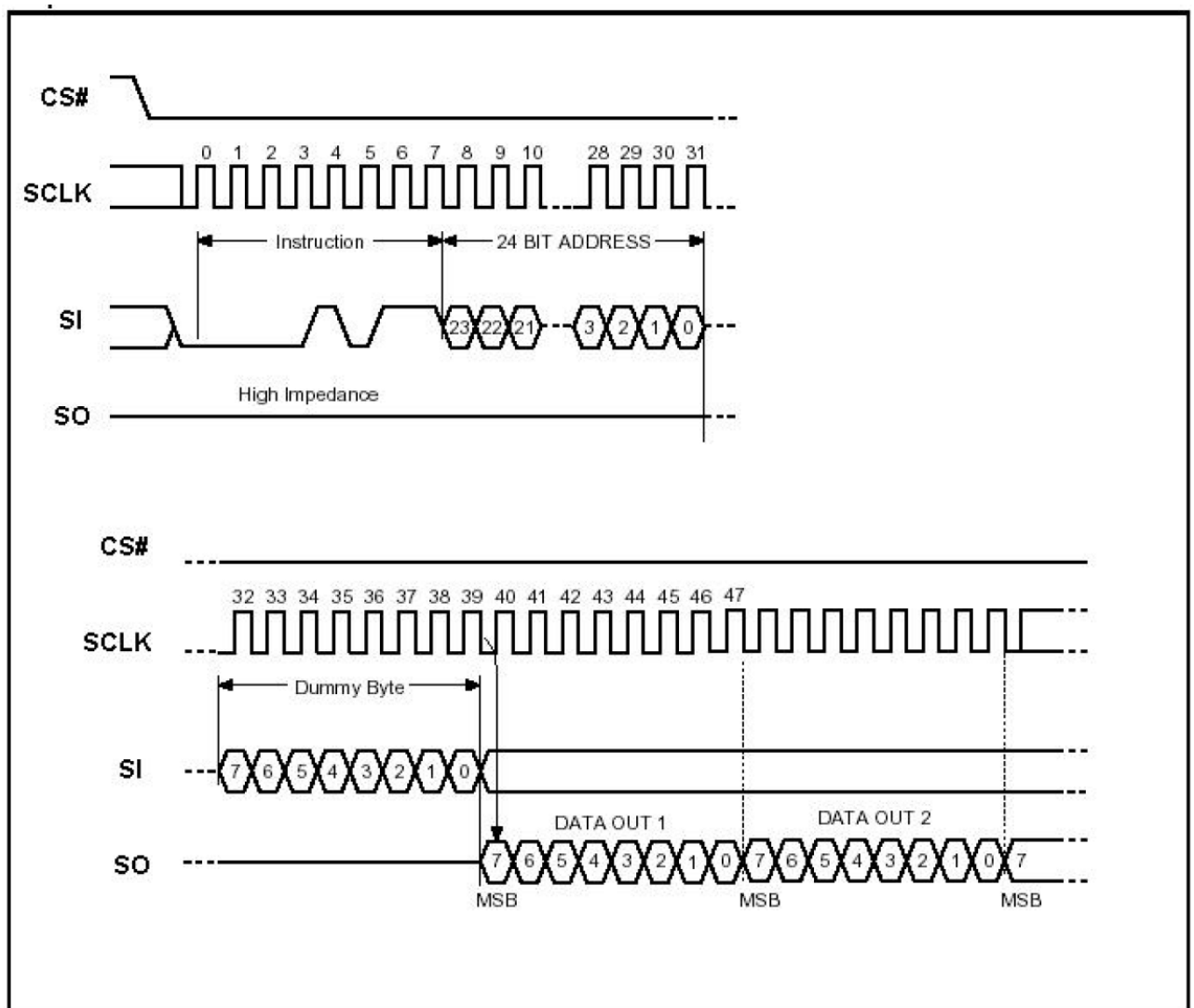
首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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

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

如果不需要继续读取数据, 则把片选信号 (CS#) 变为高, 结束本次操作。

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



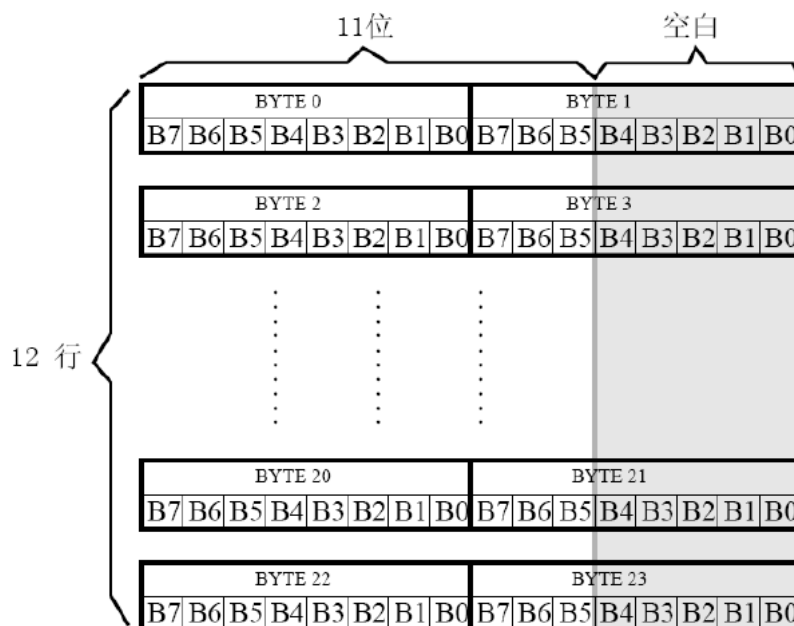
6 字库调用方法

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。点阵排列格式为横置横排：即一个字节的低位表示左面的点，高位表示右面的点，排满一行的点后再排下一行。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。

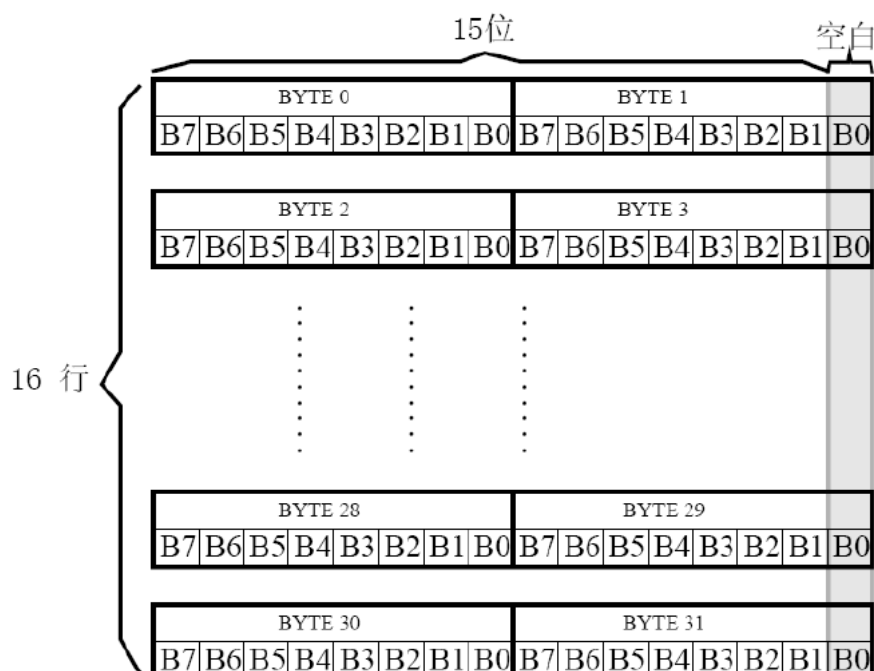
6.1.1 11X12 点汉字排列格式

11X12 点汉字的信息需要24 个字节（BYTE 0 – BYTE 23）来表示。该11X12 点汉字的点阵数据是横置横排的，其具体排列结构如下图：



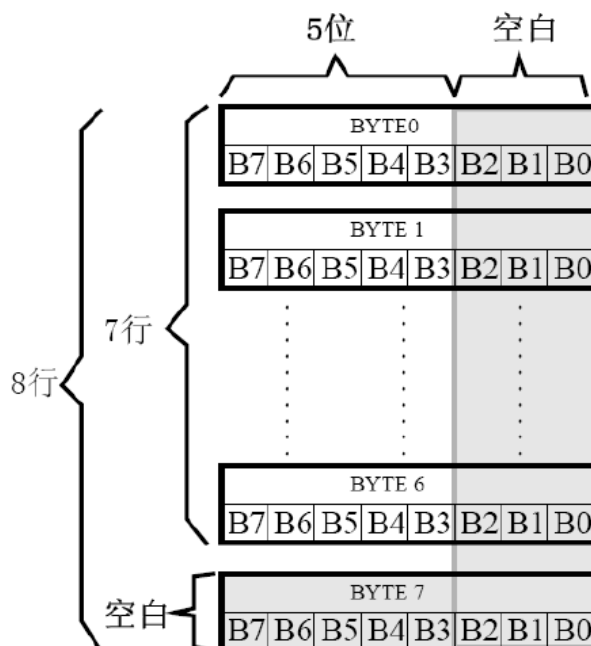
6.1.2 15X16 点汉字排列格式

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



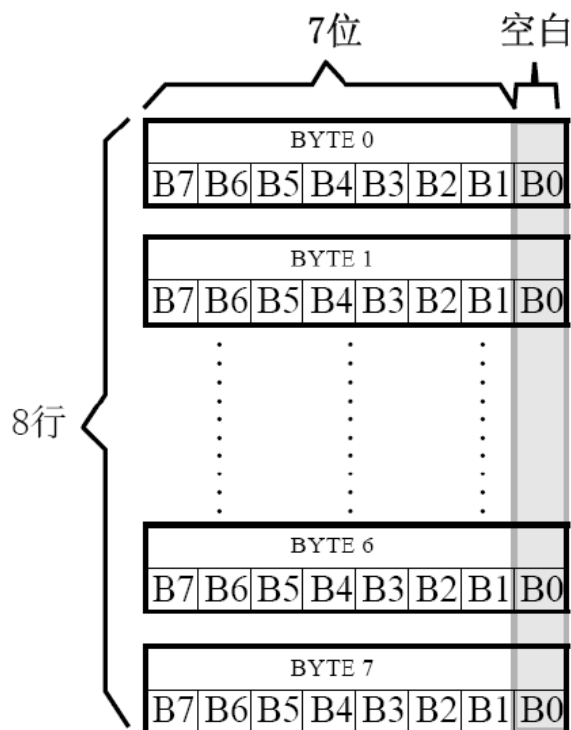
6.1.3 5X7 点ASCII 字符排列格式

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



6.1.4 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 点阵数据是横置横排的，其具体排列结构如下图：



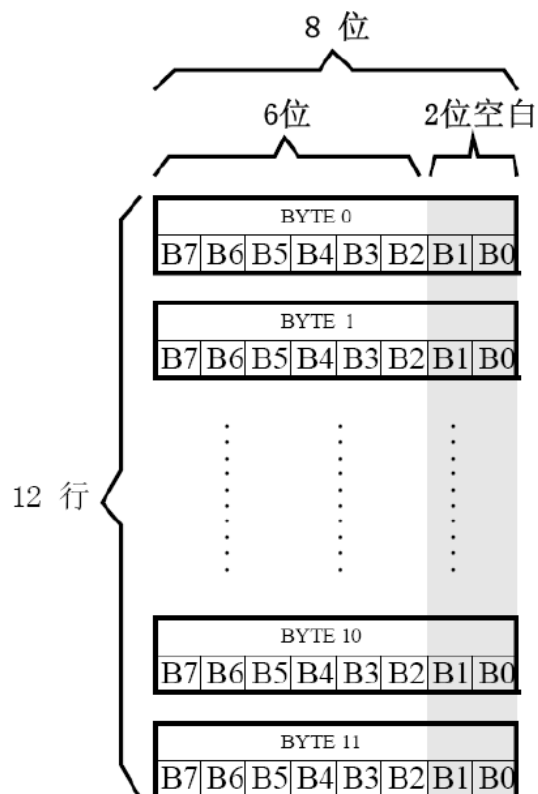
6.1.5 6X12 点字符排列格式

适用于此种排列格式的字体有：

6X12 点ASCII 字符

6X12 点国标扩展字符

6X12 点ASCII 的信息需要12 个字节（BYTE 0 – BYTE11）来表示。该ASCII 点阵数据是横置横排的，其具体排列结构如下图：



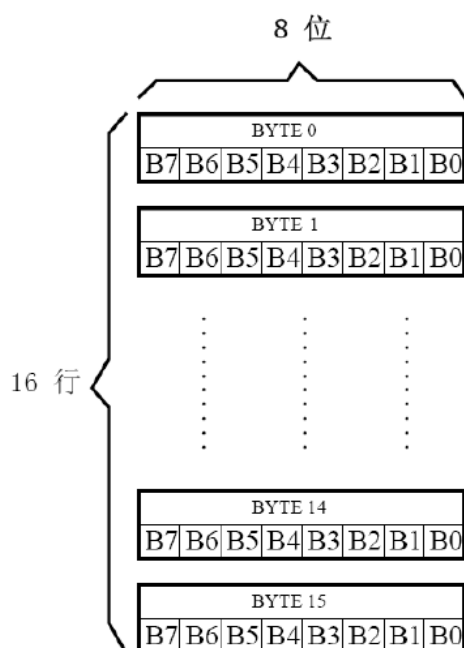
6.1.6 8X16 点字符排列格式

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。该点阵数据是横置横排的，其具体排列结构如下图：



6.1.7 12 点阵不等宽ASCII 方头 (Arial) 字符排列格式

12 点阵不等宽字符的信息需要26 个字节（BYTE 0 – BYTE25）来表示。

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，BYTE2-25 存放横置横排点阵数据。

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

6.1.4 8X16 点字符排列格式

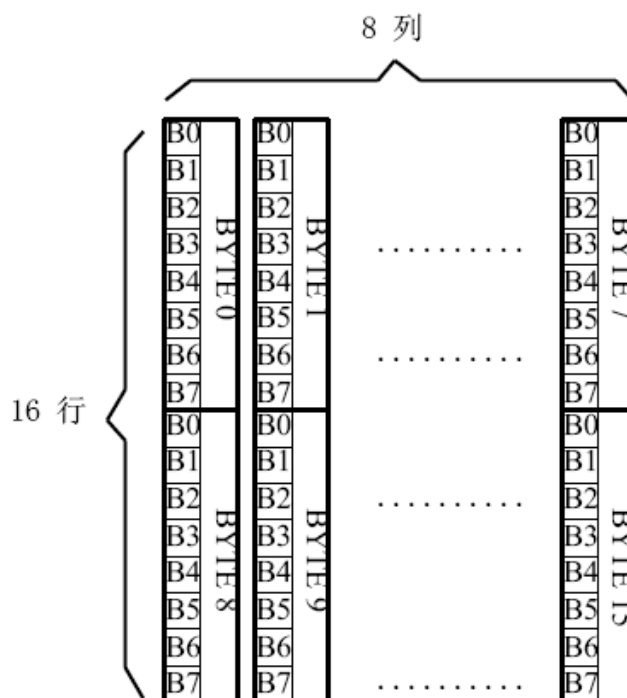
适用于此种排列格式的字体有:

8X16 点 ASCII 字符

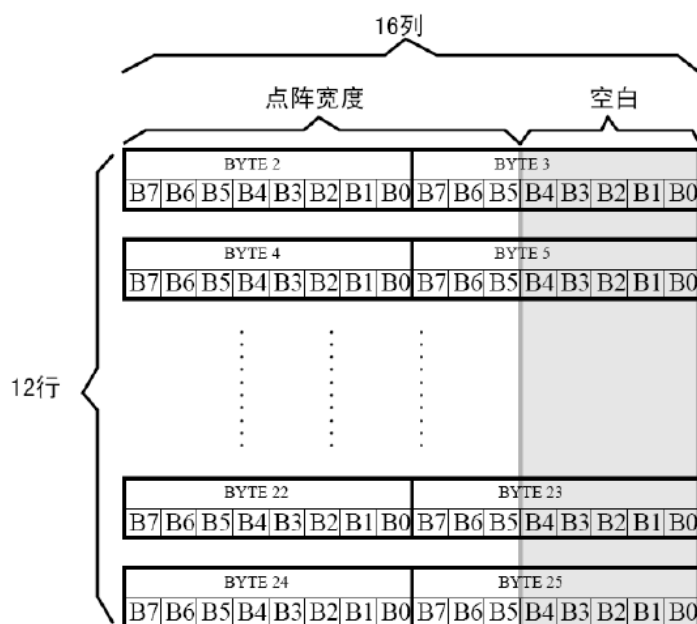
8X16 点 ASCII 粗体字符

8X16 点国标扩展字符

8X16 点字符信息需要 16 个字节 (BYTE 0 - BYTE 15) 来表示。该点阵数据是竖置横排的, 其具体排列结构如下图:



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。

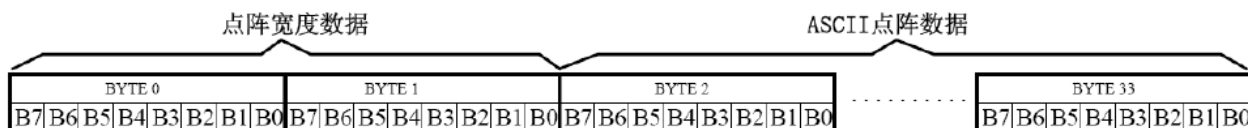


6.1.8 16 点阵不等宽ASCII 方头 (Arial) 字符排列格式

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

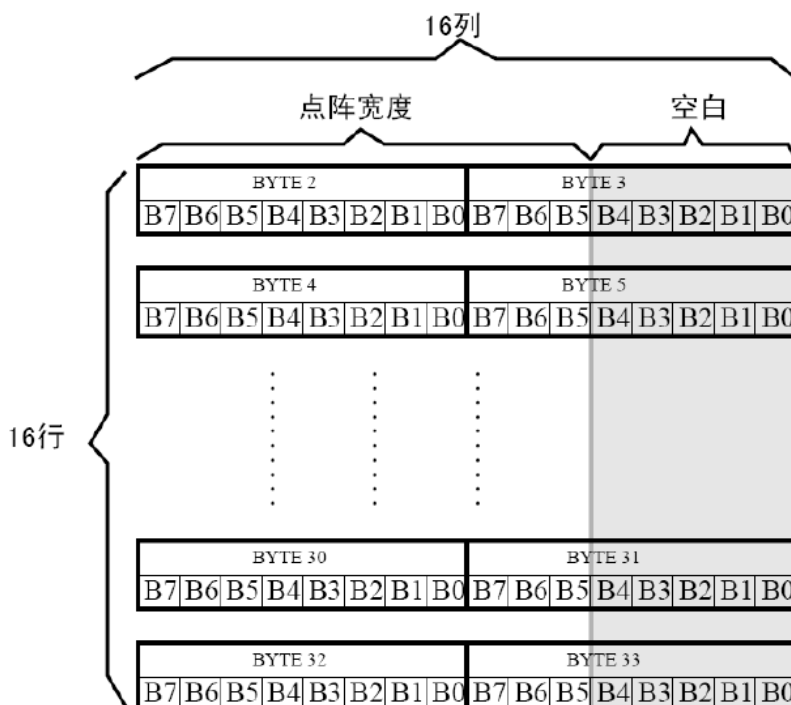
■ 存储格式

由于字符是不等宽的，因此在存储格式中BYTE0~ BYTE1 存放点阵宽度数据，BYTE2-33 存放横置横排点阵数据。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 00 00 00 00 00 7F 80 7F C0 60 C0 60 C0 60 C0 7F 80 7F C0
60 E0 60 60 60 60 7F C0 7F 80 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 00 00 00 00 00 7F 80 7F C0 60 C0 60 C0 60 C0 7F 80 7F C0 60 E0 60 60 60 60
7F C0 7F 80 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	参考算法
1	15X16 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+376	00000	3B7BF	6.3.1.2
2	GB2312 到 Unicode 内码转换表				2F00	66BF	6.4
3	7X8 点 ASCII 字符	ASCII	20~7F	96	66C0	69BF	6.3.2.2
4	8X16 点国标扩展字符	GB2312	AAA1-ABC0	126	3B7D0	3BFBF	6.3.1.4
5	8X16 点 ASCII 字符	ASCII	20~7F	96	3B7C0	3BFBF	6.3.2.4
6	5X7 点 ASCII 字符	ASCII	20~7F	96	3BFC0	3C2BF	6.3.2.1
7	16 点阵不等宽 ASCII 方头 (Arial) 字符	ASCII	20~7F	96	3C2C0	3CF7F	6.3.2.6
8	11X12 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+376	3CF80	66D3F	6.3.1.3
9	6X12 点国标扩展字符	GB2312	AAA1-ABC0	126	66D4C	6733F	6.3.1.3
10	6X12 点 ASCII 字符	ASCII	20~7F	96	66D40	6733F	6.3.2.3
11	12 点阵不等宽 ASCII 方头 (Arial) 字符	ASCII	20~7F	96	67340	67CFF	6.3.2.5
12	保留区				67D00	67D6F	
13	Unicode 到 GB2312 内码转换表				67D70	7278F	6.5
14	GT 快捷拼音输入法码表				72790	7FA33	
15	保留区				7FA33	7FFFF	

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 11X12 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=0x3cf80;

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

Address = (MSB - 0xA1) * 94 + (LSB - 0xA1)*24+ BaseAdd;

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

Address = (282 + (LSB - 0xA1))*24+ BaseAdd;

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

Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 376)*24+ BaseAdd;

6.3.1.2 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;

6.3.1.3 6X12 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=0x66d4c

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

ByteAddress = (FontCode-0xAAA1) * 12+BaseAdd

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

ByteAddress = (FontCode-0xABA1 + 95) * 12+BaseAdd

6.3.1.4 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 6X12 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=0x66d40

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
Address = (ASCIICode - 0x20) * 12 + BaseAdd

6.3.2.4 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

6.3.2.5 12 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=0x67340

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
Address = (ASCIICode - 0x20) * 26 + BaseAdd

6.3.2.6 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

6.4 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	—	~		...	‘	’
B	“	”	(	)	<	>	《	》	「	」	『	』	【	】	【	】
C	±	×	÷	:	^	v	Σ	Π	U	∩	ε	::	√	⊥	//	∠
D	∩	⊙	∫	ℳ	=	≈	≈	∞	≠	≠	≠	≠	≠	≠	∞	∴
E	∴	↑	♀	°	'	”	℃	\$	⊗	⊗	⊗	⊗	⊗	⊗	☆	★
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	ɑ	ǻ	ǻ	ǻ	ǻ
C	g															

5.3 初始化方法

用户所编的显示程序, 开始必须进行初始化, 否则模块无法正常显示, 过程请参考程序

点亮液晶模块的步骤

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

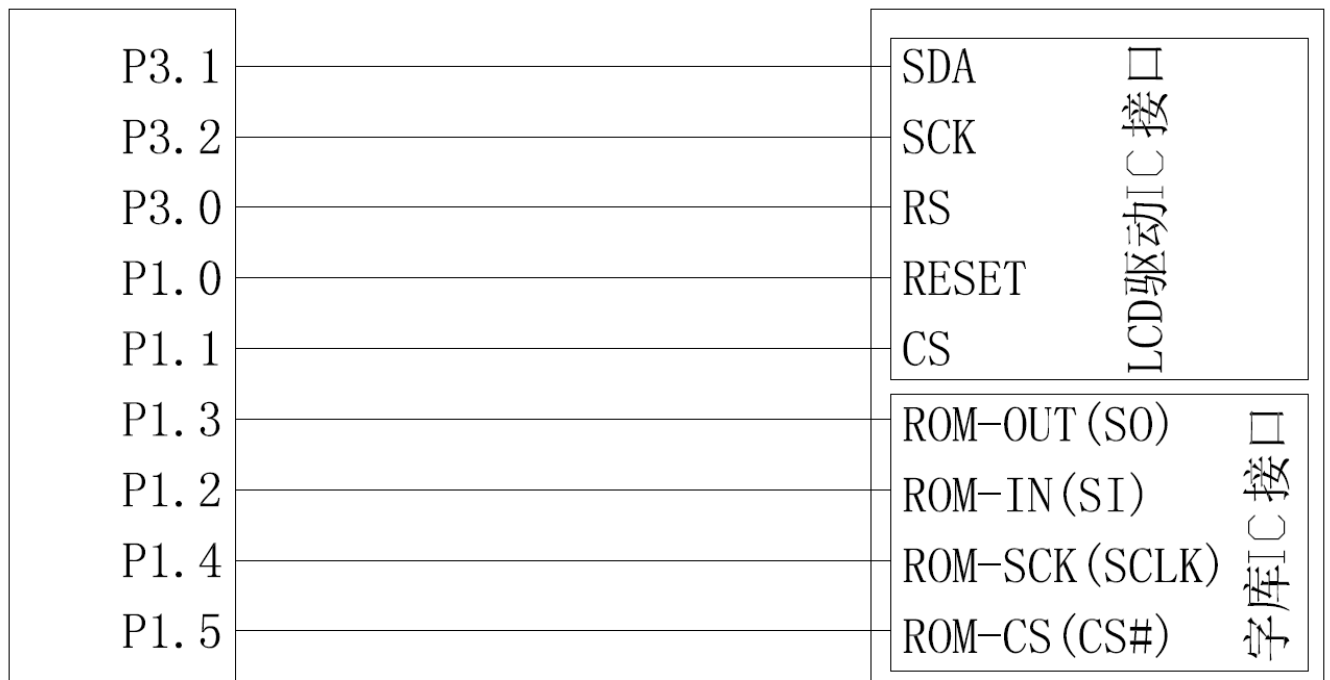
正确地接线
根据说明书正确地与开发板连接, 连接的线包括: 液晶模块电源线、背光电源线、IO端口 (接口)
IO端口包括: 并口时: CS、RESET、RW、E、RS、DO--D7, 串口时: CS、SCLK、SDA、RESET、RS

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。

MCU:

51系列

液晶模块



//通过指令可以正常竖屏、顺时针旋转 90 度、逆时针旋转 90 度、倒转 180 度竖屏

//本程序针对:JLX144-016-PC TFT 彩屏

// IC: ST7735S

//版权所有: 深圳市晶联讯电子有限公司, 网址: www.jlxlcd.cn

```
#include <reg52.h>
```

```
#include <stdio.h>
```

```
#include <128128_22.h>
```

```
#include <ASCII_TABLE_5X8_8X16_horizontal.h>
```

//液晶屏 IC 所需要的信号线的接口定义

```
sbit cs1=P1^1;
```

```
sbit reset=P1^0;
```

```
sbit rs=P3^0;
```

```
sbit sclk=P3^2;
```

```
sbit sid=P3^1;
```

```
sbit key=P2^0; //P2.0 口与 GND 之间接一个按键
```

```
sbit Rom_IN=P1^2; /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI*/
```

```
sbit Rom_OUT=P1^3; /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 S0*/
```

```
sbit Rom_SCK=P1^4; /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
```

```
sbit Rom_CS=P1^5; /*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

//定义彩屏旋转方向
#define normal 0xc8
#define CW90 0x68
#define CCW90 0xa8
#define CW180 0x08

#define red 0xf800 //定义红色
#define blue 0x001f //定义蓝色
#define green 0x07e0 //定义绿色
#define deep_green 0x0600 //定义深绿色
#define white 0xffff //定义白色
#define black 0x0000 //定义黑色
#define orange 0xfc08 //定义橙色
#define yellow 0xffe0 //定义黄色
#define pink 0xf3f3 //定义粉红色
#define purple 0xa1d6 //定义紫色
#define brown 0x8200 //定义棕色
#define gray 0xc618 //定义灰色

uchar code jiong1[]={//横向取模
/* 文字: 囧 */
/* 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 */
0x00,0x00,0x7F,0xFC,0x44,0x84,0x46,0x44,0x24,0x48,0x34,0x50,0x14,0x6F,0xE4,
0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x7F,0xFC,0x00,0x00,
};

uchar code lei1[]={//横向取模
/* 文字: 晶 */
/* 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 */
0x1F,0xF0,0x11,0x10,0x1F,0xF0,0x11,0x10,0x11,0x10,0x1F,0xF0,0x00,0x00,0xFE,0xFE,
0x92,0x92,0x92,0x92,0xFE,0xFE,0x92,0x92,0x92,0x92,0xFE,0xFE,0x82,0x82,0x00,0x00,
};

uchar code jing[]={ //横向取模
/* 文字: 晶 */
/* 宋体 23; 此字体下对应的点阵为: 宽 x 高=31x31 */
/* 宽度不是 8 的倍数, 现调整为: 宽度 x 高度=32x31 */
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x00,0x00,0xF0,0x07,0x80,
0x00,0xFF,0xFF,0x80,0x00,0xF0,0x07,0x00,0x00,0xF0,0x07,0x00,0x00,0xF0,0x07,0x00,
0x00,0xFF,0xFF,0x00,0x00,0xF0,0x07,0x00,0x00,0xF0,0x07,0x00,0x00,0xF0,0x07,0x00,
0x00,0xFF,0xFF,0x00,0x00,0xF0,0x07,0x00,0x00,0x00,0x00,0x18,0x0C,0x40,0x30,
0x1F,0xFE,0x7F,0xF8,0x1C,0x1E,0x70,0x78,0x1C,0x1C,0x70,0x70,0x1C,0x1C,0x70,0x70,
0x1C,0x1C,0x70,0x70,0x1C,0x1C,0x70,0x70,0x1F,0xFC,0x7F,0xF0,0x1C,0x1C,0x70,0x70,
```

```
0x1C, 0x1C, 0x70, 0x70, 0x1C, 0x1C, 0x70, 0x70, 0x1C, 0x1C, 0x70, 0x70, 0x1F, 0xFC, 0x7F, 0xF0,
0x1C, 0x1C, 0x70, 0x70, 0x18, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
```

```
uchar code lian[]={ //横向取模
/*— 文字: 联 —*/
/*— 宋体 23; 此字体下对应的点阵为: 宽 x 高=31x31 —*/
/*— 宽度不是 8 的倍数, 现调整为: 宽度 x 高度=32x31 —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x81, 0xC0, 0x00, 0x0D, 0xC1, 0xF0,
0x00, 0x1E, 0xE1, 0xC0, 0x7F, 0xFB, 0xF3, 0x80, 0x1C, 0x70, 0x73, 0x80, 0x1C, 0x70, 0x77, 0x00,
0x1C, 0x70, 0x06, 0x70, 0x1C, 0x70, 0x0E, 0xF8, 0x1C, 0x7F, 0xFF, 0xD8, 0x1F, 0xF0, 0x1C, 0x00,
0x1C, 0x70, 0x1C, 0x00, 0x1C, 0x70, 0x1C, 0x00, 0x1C, 0x70, 0x1C, 0x18, 0x1C, 0x70, 0x1C, 0x3C,
0x1C, 0x7F, 0xFF, 0xFE, 0x1F, 0xF0, 0x1E, 0x00, 0x1C, 0x70, 0x1E, 0x00, 0x1C, 0x70, 0x3E, 0x00,
0x1C, 0x70, 0x3F, 0x00, 0x1C, 0x7E, 0x3B, 0x00, 0x1D, 0xF0, 0x3B, 0x80, 0x1F, 0xF0, 0x73, 0x80,
0x7C, 0x70, 0x71, 0xC0, 0x30, 0x70, 0xE1, 0xE0, 0x00, 0x71, 0xC0, 0xF0, 0x00, 0x73, 0x80, 0x7C,
0x00, 0x77, 0x00, 0x3C, 0x00, 0x7C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
```

```
uchar code xun[]={ //横向取模
/*— 文字: 讯 —*/
/*— 宋体 23; 此字体下对应的点阵为: 宽 x 高=31x31 —*/
/*— 宽度不是 8 的倍数, 现调整为: 宽度 x 高度=32x31 —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x00, 0x00, 0x80,
0x07, 0x80, 0x01, 0xC0, 0x03, 0xFF, 0xFF, 0xE0, 0x03, 0xC0, 0x01, 0xC0, 0x01, 0x81, 0xE1, 0xC0,
0x00, 0x01, 0xE1, 0xC0, 0x00, 0x01, 0xC1, 0xC0, 0x00, 0x01, 0xC1, 0xC0, 0x00, 0x01, 0xC1, 0xC0,
0x03, 0xC1, 0xC1, 0xC0, 0x7F, 0xE1, 0xCD, 0xC0, 0x03, 0x81, 0xDF, 0xC0, 0x03, 0xBF, 0xFB, 0xC0,
0x03, 0x81, 0xC1, 0xC0, 0x03, 0x81, 0xC1, 0xC0, 0x03, 0x81, 0xC1, 0xC0, 0x03, 0x81, 0xC1, 0xC0,
0x03, 0x81, 0xC1, 0xC0, 0x03, 0x9D, 0xC1, 0xC0, 0x03, 0xB9, 0xC1, 0xEC, 0x03, 0xF1, 0xC0, 0xFC,
0x03, 0xE1, 0xC0, 0xFC, 0x03, 0xE1, 0xC0, 0xFC, 0x03, 0xC1, 0xC0, 0x7C, 0x01, 0x81, 0xC0, 0x3C,
0x00, 0x01, 0xC0, 0x1C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
```

```
uchar code jing1[]={//横向取模
/*— 文字: 晶 —*/
/*— 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 —*/
0x00, 0x00, 0x0F, 0xF0, 0x08, 0x10, 0x0F, 0xF0, 0x08, 0x10, 0x0F, 0xF0, 0x08, 0x10, 0x00, 0x00,
0x7E, 0x7E, 0x42, 0x42, 0x7E, 0x7E, 0x42, 0x42, 0x42, 0x42, 0x7E, 0x7E, 0x42, 0x42, 0x00, 0x00,
};
```

```
uchar code lian1[]={//横向取模
/*— 文字: 联 —*/
/*— 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 —*/
0x01, 0x08, 0xFE, 0x8C, 0x44, 0x48, 0x44, 0x50, 0x7F, 0xFE, 0x44, 0x20, 0x44, 0x20, 0x7C, 0x20,
0x47, 0xFE, 0x44, 0x20, 0x4E, 0x20, 0xF4, 0x20, 0x44, 0x50, 0x04, 0x48, 0x04, 0x86, 0x05, 0x04];
```

```
uchar code xun1[]={//横向取模
```

```
/*— 文字: 讯 —*/
```

/*— 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 —*/

0x40, 0x00, 0x27, 0xF8, 0x31, 0x08, 0x21, 0x08, 0x01, 0x08, 0xF1, 0x08, 0x17, 0xE8, 0x11, 0x08,
0x11, 0x08, 0x11, 0x08, 0x11, 0x08, 0x11, 0x0A, 0x15, 0x0A, 0x19, 0x0A, 0x11, 0x04, 0x00, 0x00];

```
void delay(long int i)
{
    long int j,k;
    for (j=0;j<i;j++)
        for (k=0;k<110;k++);
}
```

//等待按键

```
void waitkey()
{
repeat:
    if(key==1) goto repeat;
    else delay(1000);
}
```

/*写指令到 LCD 模块*/

```
void transfer_command(int data1)
{
    char i;
    cs1=0;
    rs=0;
    for (i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
}
```

/*写数据到 LCD 模块*/

```
void transfer_data(int data1)
{
    char i;
    cs1=0;
    rs=1;
    for (i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
```

```
        data1=data1<<=1;
    }
}

//连写 2 个字节（即 16 位）数据到 LCD 模块
void transfer_data_16(uint data2)
{
    transfer_data(data2>>8);
    transfer_data(data2);
}

//LCD 初始化
void LCD_initial()
{
    delay(50);
    reset=0;           //低电平：复位
    delay(1);
    reset=1;           //高电平：复位结束
    delay(10);
    //开始初始化：
    transfer_command(0x11);

    transfer_command(0xb1);
    transfer_data(0x01);
    transfer_data(0x2c);
    transfer_data(0x2d);

    transfer_command(0xb2);
    transfer_data(0x01);
    transfer_data(0x2c);
    transfer_data(0x2d);

    transfer_command(0xb3);
    transfer_data(0x01);
    transfer_data(0x2c);
    transfer_data(0x2d);
    transfer_data(0x01);
    transfer_data(0x2d);
    transfer_data(0x2d);

    transfer_command(0xb4);
    transfer_data(0x02);

    transfer_command(0xb6);
    transfer_data(0xb4);
    transfer_data(0xf0);
```

```
transfer_command(0xc0);
transfer_data(0xa2);
transfer_data(0x02);
    transfer_data(0x84);

transfer_command(0xc1);
transfer_data(0xc5);

transfer_command(0xc2);
transfer_data(0x0a);
transfer_data(0x00);

transfer_command(0xc3);
transfer_data(0x8a);
transfer_data(0x2a);

transfer_command(0xc4);
transfer_data(0x8a);
transfer_data(0xee);

transfer_command(0xc5);
transfer_data(0x0e);

transfer_command(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
transfer_data(CW180); //MX=1 (行地址顺序: 从左到右), MY=1 (列地址顺序: 从上到下), MV=0 (竖放), ML=0 (纵向刷新: 从上到下), RGB=1 (依
次为 RGB), MH=0 (横向刷新顺序: 从左到右)
//定义: "normal"就是 "0xc8" ——正常竖放;
//定义: "CW180"就是 "0x08"——在正常竖放基础上转 180 度竖放;
//定义: "CCW90" 就是 "0xa8"——在竖放基础上逆时针转 90 度横放;
//定义: "CW90" 就是 "0x68"——在竖放基础上顺转 90 度横放;

transfer_command(0xff);
transfer_data(0x40);
transfer_data(0x03);
transfer_data(0x1a);

transfer_command(0xfc);
transfer_data(0x11);
transfer_data(0x17);

transfer_command(0xf0);
transfer_data(0x01);

transfer_command(0x3a);
transfer_data(0x05);

transfer_command(0xf6);
transfer_data(0x00);
```

```
transfer_command(0xe0);  
transfer_data(0x02);  
transfer_data(0x1c);  
transfer_data(0x07);  
transfer_data(0x12);  
transfer_data(0x37);  
transfer_data(0x32);  
transfer_data(0x29);  
transfer_data(0x2d);  
transfer_data(0x29);  
transfer_data(0x25);  
transfer_data(0x2b);  
transfer_data(0x39);  
transfer_data(0x00);  
transfer_data(0x01);  
transfer_data(0x03);  
transfer_data(0x10);
```

```
transfer_command(0xe1);  
transfer_data(0x0b);  
transfer_data(0x14);  
transfer_data(0x09);  
transfer_data(0x26);  
transfer_data(0x27);  
transfer_data(0x22);  
transfer_data(0x1c);  
transfer_data(0x20);  
transfer_data(0x1d);  
transfer_data(0x1a);  
transfer_data(0x25);  
transfer_data(0x2d);  
transfer_data(0x06);  
transfer_data(0x06);  
transfer_data(0x02);  
transfer_data(0x0f);
```

```
transfer_command(0x2a);    //定义 X 地址的开始及结束位置  
transfer_data(0x00);  
transfer_data(0x00);  
transfer_data(0x00);  
transfer_data(0x7F);
```

```
transfer_command(0x2b);    //定义 Y 地址的开始及结束位置  
transfer_data(0x00);  
transfer_data(0x00);  
transfer_data(0x00);
```

```
transfer_data(0x7F);

transfer_command(0x29);    //开显示
}

//地址函数
//定义窗口坐标: 开始坐标 (XS,YS)以及窗口大小 (x_total,y_total)
void lcd_address(int XS,int YS,int x_total,int y_total)
{
    int XE,YE;
    XS=XS+2;                //起始坐标 X 轴修正值
    YS=YS+1;                //起始坐标 Y 轴修正值
    XE=XS+x_total-1;
    YE=YS+y_total-1;
    transfer_command(0x2a);    // 设置 X 开始及结束的地址
    transfer_data_16(XS);      // X 开始地址(16 位)
    transfer_data_16(XE);      // X 结束地址(16 位)
    transfer_command(0x2b);    // 设置 Y 开始及结束的地址
    transfer_data_16(YS);      // Y 开始地址(16 位)
    transfer_data_16(YE);      // Y 结束地址(16 位)
    transfer_command(0x2c);    // 写数据开始
}

//将单色的 8 位的数据 (代表 8 个像素点) 转换成彩色的数据传输给液晶屏
void mono_transfer_data(int mono_data,int font_color,int back_color)
{
    int i;
    for(i=0;i<8;i++)
    {
        if(mono_data&0x80)
        {
            transfer_data_16(font_color);    //当数据是 1 时, 显示字体颜色
        }
        else
        {
            transfer_data_16(back_color);    //当数据是 0 时, 显示底色
        }
        mono_data<<=1;
    }
}

//显示单一色彩
void display_color(int XS,int YS,int x_total,int y_total,int color)
{
    int i,j;
    lcd_address(XS,YS,x_total,y_total);
    for(i=0;i<128;i++)
```

```
{
    for(j=0; j<128; j++)
    {
        transfer_data_16(color);
    }
}
```

//显示一幅全屏彩图

```
void display_image(uchar *dp)
{
    uchar i, j;
    lcd_address(0, 0, 128, 128);
    for(i=0; i<128; i++)
    {
        for(j=0; j<128; j++)
        {
            transfer_data(*dp); //传一个像素的图片数据的高位
            dp++;
            transfer_data(*dp); //传一个像素的图片数据的低位
            dp++;
        }
    }
}
```

//显示 32x32 点阵的汉字，或相当于 16x16 点阵的图像。温馨提示，数据指针*dp 是字符型数据（char *dp）

```
void disp_32x32(int x, int y, char *dp, int font_color, int back_color) //int x X 轴坐标, int y, Y 轴坐标
{
    int i, j;
    lcd_address(x, y, 32, 32);
    for(i=0; i<32; i++)
    {
        for(j=0; j<4; j++)
        {
            mono_transfer_data(*dp, font_color, back_color);
            dp++;
        }
    }
}
```

//显示 16x16 点阵的汉字，或相当于 16x16 点阵的图像。温馨提示，数据指针*dp 是字符型数据（char *dp）

```
void disp_16x16(int x, int y, char *dp, int font_color, int back_color) //int x X 轴坐标, int y, Y 轴坐标
{
    int i, j;
    lcd_address(x, y, 16, 16);
    for(i=0; i<2; i++)
    {
```

```
        for(j=0;j<16;j++)
        {
            mono_transfer_data(*dp, font_color, back_color);
            dp++;
        }
    }
}
```

//显示 8x16 点阵的字符

```
void display_graphic_8x16(int x, int y, char *dp, int font_color, int back_color)
{
    int k;
    cs1=0;
    Rom_CS = 1;
    lcd_address(x, y, 8, 16);
    for(k=0;k<16;k++)
    {
        mono_transfer_data(*dp, font_color, back_color);
        dp++;
    }
}
```

//将单色的 8 位的数据的高 5 位（代表 5 个像素点）转换成彩色的数据传输给液晶屏

```
void mono_data_out_5x8(char mono_data, int font_color, int back_color)
```

```
{
    int i;
    for(i=0;i<6;i++)
    {
        if(mono_data&0x80)
        {
            transfer_data_16(font_color);    //当数据是 1 时，显示字体颜色
        }
        else
        {
            transfer_data_16(back_color);    //当数据是 0 时，显示底色
        }

        mono_data<<=1;
    }
}
```

//显示 5x8 点阵的字符

```
void dis_graphic_5x8(int x, int y, char *dp, int font_color, int back_color)
{
    int k;
    cs1=0;
    Rom_CS = 1;
```

```
lcd_address(x, y, 6, 8);  
for (k=0;k<8;k++)  
{  
    mono_data_out_5x8(*dp, font_color, back_color);  
    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;  
    }  
}
```

****从晶联讯字库 IC 中取汉字或字符数据 (1 个字节) ****

```
static uchar get_data_from_ROM( )
```

```
{  
    uchar i;  
    uchar ret_data=0;  
    Rom_SCK=1;  
    for (i=0;i<8;i++)  
    {  
        Rom_OUT=1;  
        Rom_SCK=0;  
        ret_data=ret_data<<1;  
        if( Rom_OUT )  
            ret_data=ret_data+1;  
        else  
            ret_data=ret_data+0;  
        Rom_SCK=1;  
    }  
    return(ret_data);  
}
```

/*从相关地址 (addrHigh: 地址高字节, addrMid: 地址中字节, addrLow: 地址低字节) 中连续读出 DataLen 个字节的数据到 pBuff 的地址*/

/*连续读取*/

```
void get_n_bytes_data_from_ROM(uchar addrHigh, uchar addrMid, uchar addrLow, uchar *pBuff, uchar DataLen )
```

```
{
    uchar i;
    Rom_CS = 0;
    cs1=1;
    Rom_SCK=0;
    send_command_to_ROM(0x03);
    send_command_to_ROM(addrHigh);
    send_command_to_ROM(addrMid);
    send_command_to_ROM(addrLow);
    for(i = 0; i < DataLen; i++)
    {
        *(pBuff+i) =get_data_from_ROM();
    }
    Rom_CS = 1;
}

//显示一串 16x16 汉字, 全角标点、符号;或 8x16 的 ASCII、半角标点、符号
ulong fontaddr=0;
void display_GB2312_string(uchar x,uchar y,uchar *text,int font_color,int back_color)
{
    int i= 0;
    uchar addrHigh,addrMid,addrLow ;
    uchar fontbuf[32];
    while((text[i]>0x00))
    {
        Rom_SCK=0;
        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);

            addrHigh = (fontaddr&0xff0000)>>16;    //地址的高 8 位, 共 24 位
            addrMid = (fontaddr&0xff00)>>8;        //地址的中 8 位, 共 24 位
            addrLow = fontaddr&0xff;              //地址的低 8 位, 共 24 位
            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,32 );//取 32 个字节的数据, 存到"fontbuf[32]"
            disp_16x16(x,y,fontbuf,font_color,back_color);    //显示汉字到 LCD 上, y 为页地址, x 为列地址, fontbuf[] 为数据
            i+=2;
            x+=16;
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xa3))&&(text[i+1]>=0xa1))
        {
            /*国标简体 (GB2312) 15x16 点的全角标点符号 (例如 “,” “。”) 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);

addrHigh = (fontaddr&0xff0000)>>16; //地址的高 8 位, 共 24 位
addrMid = (fontaddr&0xff00)>>8; //地址的中 8 位, 共 24 位
addrLow = fontaddr&0xff; //地址的低 8 位, 共 24 位
get_n_bytes_data_from_ROM(addrHigh, addrMid, addrLow, fontbuf, 32 );//取 32 个字节的数据，存到"fontbuf[32]"
disp_16x16(x, y, fontbuf, font_color, back_color); //显示汉字到 LCD 上, y 为页地址, x 为列地址, fontbuf[] 为数据
i+=2;
x=x+16;
}
else if((text[i]>=0x20) &&(text[i]<=0x7e))
{
//ASCII 码字符的 8*16 点阵在字库 IC 中的地址
unsigned char fontbuf[16];
fontaddr = (text[i]- 0x20);
fontaddr = (unsigned long) (fontaddr*16);
fontaddr = (unsigned long) (fontaddr+0x3b7c0);
addrHigh = (fontaddr&0xff0000)>>16;
addrMid = (fontaddr&0xff00)>>8;
addrLow = fontaddr&0xff;
get_n_bytes_data_from_ROM(addrHigh, addrMid, addrLow, fontbuf, 16 );//取 16 个字节的数据，存到"fontbuf[32]"
display_graphic_8x16(x, y, fontbuf, font_color, back_color); //显示 8x16 的 ASCII 字到 LCD 上, y 为页地址, x 为列地址,
fontbuf[] 为数据
i+=1;
x+=8;
}
else
i++;
}
}

```

```

void display_string_5x8(uchar x, uchar y, uchar *text, int font_color, int back_color)

```

```

{
    unsigned char i= 0;
    unsigned char addrHigh, addrMid, addrLow ;
    while((text[i]>0x00))
    {
        if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            unsigned char fontbuf[8];
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long) (fontaddr*8);
            fontaddr = (unsigned long) (fontaddr+0x3bfc0);
            addrHigh = (fontaddr&0xff0000)>>16;

```

```

        addrMid = (fontaddr&0xff00)>>8;
        addrLow = fontaddr&0xff;
        get_n_bytes_data_from_ROM(addrHigh, addrMid, addrLow, fontbuf, 8); //取 8 个字节的数据, 存到"fontbuf[32]"
        dis_graphic_5x8(x, y, fontbuf, font_color, back_color); //显示 5x8 的 ASCII 字到 LCD 上, x 为列地址, y 为行地址, fontbuf[]为
数据(数组)

        i+=1;
        x+=6; // "x=x+6"不用 "x=x+5" 的原因是留一点字间隔好美观一点
    }
    else
        i++;
}
}

//主程序
void main(void)
{
    while(1)
    {
        LCD_initial(); //初始化
        display_color(0, 0, 128, 128, blue);
        display_GB2312_string(0, 16*0, "128x128 带汉字库, ", white, red); //显示汉字, 参数分别是 (x, y, " 汉字 ", 字符颜色, 背景颜色)
        display_GB2312_string(0, 16*1, "16X16 字或符号:'C", white, orange);
        display_GB2312_string(0, 16*2, "或 8X16 点阵 ASCII, ", white, deep_green);
        display_GB2312_string(0, 16*3, "或 5X8 点阵 ASCII 码", white, blue);
        display_GB2312_string(0, 16*4, "GB2312 简体字库及", white, pink);
        display_GB2312_string(0, 16*5, "有图型功能, 可自", red, white);
        display_GB2312_string(0, 16*6, "编大字、图像、生", orange, white);
        display_GB2312_string(0, 16*7, "僻字, 例如: ", deep_green, white);
        disp_16x16(16*6, 16*7, jiong1, deep_green, white); //显示单个自编生僻汉字“囧”, 参数分别是 (x, y, " 汉字
", 字符颜色, 背景颜色)
        disp_16x16(16*7, 16*7, lei1, deep_green, white);
        waitkey();
    }
}

//由于 IC 是 128*160 点阵的, 旋转的时候地址会出现偏差, 在此须修改, X 和 Y 的地址参数起始值
//也可以在定义地址函数里面修改 X 和 Y 的修正值

transfer_command(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
transfer_data(0xc8); //定义: "normal"就是 "0xc8" ——正常竖放;
//定义: "CW180"就是 "0x08"——在正常竖放基础上转 180 度竖放;
//定义: "CCW90" 就是 "0xa8"——在竖放基础上逆时针转 90 度横放;
//定义: "CW90" 就是 "0x68"——在竖放基础上顺转 90 度横放;

display_color(0, 0+2, 128, 128, 0x0000); //全屏写 0x0000, 相当于清屏。
delay(10);
display_GB2312_string(0, 16*0+2, "128x128 带汉字库, ", white, red); //显示汉字, 参数分别是 (x, y, " 汉字 ", 字符颜色,
背景颜色)

display_GB2312_string(0, 16*1+2, "16X16 字或符号:'C", white, orange);
display_GB2312_string(0, 16*2+2, "或 8X16 点阵 ASCII, ", white, deep_green);
display_GB2312_string(0, 16*3+2, "或 5X8 点阵 ASCII 码", white, blue);
display_GB2312_string(0, 16*4+2, "GB2312 简体字库及", white, pink);

```

```
display_GB2312_string(0,16*5+2,"有图型功能,可自",red,white); //有图型功能,可
display_GB2312_string(0,16*6+2,"编大字、图像、生",orange,white);
display_GB2312_string(0,16*7+2,"僻字,例如: ",deep_green,white);
disp_16x16(16*6,16*7+2,jiong1,deep_green,white); //显示单个自编生僻汉字“囧”,参数分别是(x,y,汉字,字符颜色,
背景颜色)

disp_16x16(16*7,16*7+2,lei1,deep_green,white);
waitkey();

//由于 IC 是 128*160 点阵的,旋转的时候地址会出现偏差,在此须修改, X 和 Y 的地址参数起始值
//也可以在定义地址函数里面修改 X 和 Y 的修正值

transfer_command(0x36); //行扫描顺序,列扫描顺序,横放/竖放
transfer_data(CW90); //定义:"normal"就是"0xc8"—正常竖放;
//定义:"CW180"就是"0x08"—在正常竖放基础上转 180 度竖放;
//定义:"CCW90"就是"0xa8"—在竖放基础上逆时针转 90 度横放;
//定义:"CW90"就是"0x68"—在竖放基础上顺转 90 度横放;

display_color(0-1,0+1,128,128,0x0000); //全屏写 0x0000,相当于清屏。
delay(10);
display_GB2312_string(0-1,16*0+1,"128x128 带汉字库",white,red); //显示汉字,参数分别是(x,y,汉字,字符颜色,
背景颜色)

display_GB2312_string(0-1,16*1+1,"16X16 字或符号:'C",white,orange);
display_GB2312_string(0-1,16*2+1,"或 8X16 点阵 ASCII",white,deep_green);
display_GB2312_string(0-1,16*3+1,"或 5X8 点阵 ASCII 码",white,blue);
display_GB2312_string(0-1,16*4+1,"GB2312 简体字库及",white,pink);
display_GB2312_string(0-1,16*5+1,"有图型功能,可自",red,white); //有图型功能,可
display_GB2312_string(0-1,16*6+1,"编大字、图像、生",orange,white);
display_GB2312_string(0-1,16*7+1,"僻字,例如: ",deep_green,white);
disp_16x16(16*6-1,16*7+1,jiong1,deep_green,white); //显示单个自编生僻汉字“囧”,参数分别是(x,y,汉字,字符颜色,
背景颜色)

disp_16x16(16*7-1,16*7+1,lei1,deep_green,white);
waitkey();

//由于 IC 是 128*160 点阵的,旋转的时候地址会出现偏差,在此须修改, X 和 Y 的地址参数起始值
//也可以在定义地址函数里面修改 X 和 Y 的修正值

transfer_command(0x36); //行扫描顺序,列扫描顺序,横放/竖放
transfer_data(CW90); //定义:"normal"就是"0xc8"—正常竖放;
//定义:"CW180"就是"0x08"—在正常竖放基础上转 180 度竖放;
//定义:"CCW90"就是"0xa8"—在竖放基础上逆时针转 90 度横放;
//定义:"CW90"就是"0x68"—在竖放基础上顺转 90 度横放;

display_color(0+1,0+1,128,128,0x0000); //全屏写 0x0000,相当于清屏。
delay(10);
display_GB2312_string(0+1,16*0+1,"128x128 带汉字库",white,red); //显示汉字,参数分别是(x,y,汉字,字符颜色,
背景颜色)

display_GB2312_string(0+1,16*1+1,"16X16 字或符号:'C",white,orange);
display_GB2312_string(0+1,16*2+1,"或 8X16 点阵 ASCII",white,deep_green);
display_GB2312_string(0+1,16*3+1,"或 5X8 点阵 ASCII 码",white,blue);
display_GB2312_string(0+1,16*4+1,"GB2312 简体字库及",white,pink);
display_GB2312_string(0+1,16*5+1,"有图型功能,可自",red,white); //有图型功能,可
display_GB2312_string(0+1,16*6+1,"编大字、图像、生",orange,white);
display_GB2312_string(0+1,16*7+1,"僻字,例如: ",deep_green,white);
```

```
disp_16x16(16*6+1, 16*7+1, jiong1, deep_green, white); //显示单个自编生僻汉字“囧”, 参数分别是(x, y, 汉字, 字符颜色,
背景颜色)
disp_16x16(16*7+1, 16*7+1, lei1, deep_green, white);
waitkey();
transfer_command(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
transfer_data(CW180);
display_color(0, 0, 128, 160, blue);
disp_32x32(16+32*0, 32, jing, red, white);
disp_32x32(16+32*1, 32, lian, red, white);
disp_32x32(16+32*2, 32, xun, red, white);
disp_16x16(8+16*2, 8, jing1, red, white);
disp_16x16(8+16*3, 8, lian1, red, white);
disp_16x16(8+16*4, 8, xun1, red, white);
display_GB2312_string(0, 65, " JLX144-016 ", yellow, blue); //在(x, y)位置开始, 显示 ASCII 字符串
display_GB2312_string(0, 81, " 1.44 TFT ", yellow, blue); //在(x, y)位置开始, 显示 ASCII 字符串
display_string_5x8(0, 97, " JLX144-016, 1.44TFT", yellow, blue); //在(x, y)位置开始, 显示 ASCII 字符串
display_string_5x8(0, 105, " www.jlxlcd.cn", yellow, blue); //在(x, y)位置开始, 显示 ASCII 字符串
display_string_5x8(0, 113, " tel:0755-29784961", yellow, blue); //在(x, y)位置开始, 显示 ASCII 字符串
waitkey();
display_color(0, 0, 128, 128, red);
waitkey();
display_color(0, 0, 128, 128, green);
waitkey();
display_color(0, 0, 128, 128, blue);
waitkey();
display_image(pic1); //显示单幅彩图, 编码是通过专用取模工具获取的, 取模工具请找客服人员索取。
waitkey();
}
```

