

JLX177-00608-PC 使用说明书

(带字库 IC)

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	7~10
6	字库的使用方法	11~21
5	指令功能及硬件接口与编程案例	22~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
 搬扳般颁板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
 背贝钡倍狈备惫焙被奔笨本笨崩绷甬

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
 搬扳般颁板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
 背贝钡倍狈备惫焙被奔笨本笨崩绷甬

5x7 点 ASCII 字符

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

7x8 点 ASCII 字符

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

6x12 点 ASCII 字符

```
!"#$%&'()*+,-./0123456789:;
=>?@ABCDEFGHIJKLMN O PQRSTU V W
YZ[\]^_`abcdefghijklmnopqrs
uvwxyz{|}~āáâãäēěèìíîïōóòû
```

8x16 点 ASCII 字符

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

12 点阵不等宽 ASCII 方头

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

16 点阵不等宽 ASCII 方头

```
!"#$%&'()*+,-./0123456789:;<=>
DEFGHIJKLMN O PQRSTU VWX
abcdefghijklmnopqrs tuvwxyz{|
```

2. 外形尺寸及接口引脚功能

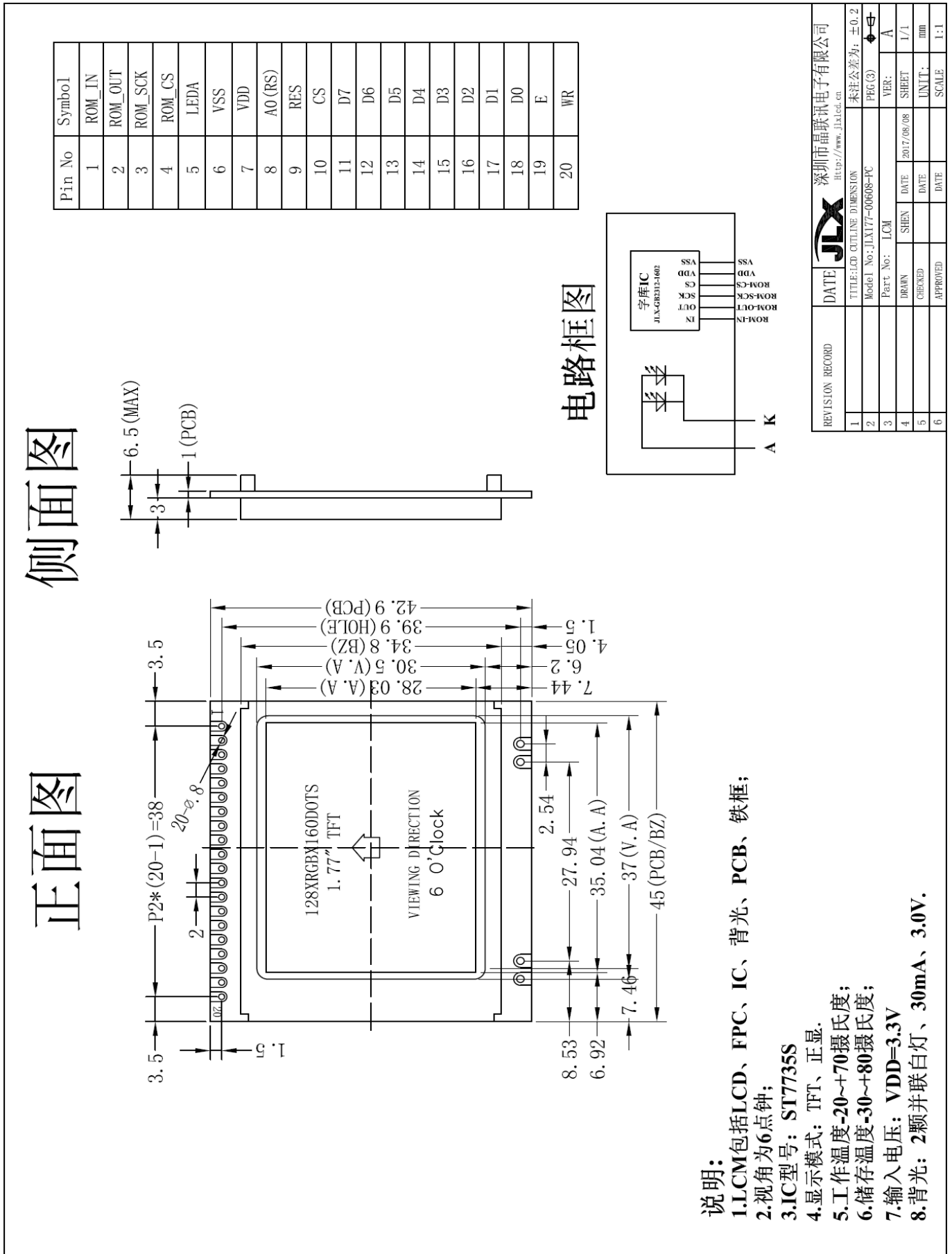


图 1. 外形尺寸

模块的接口引脚功能

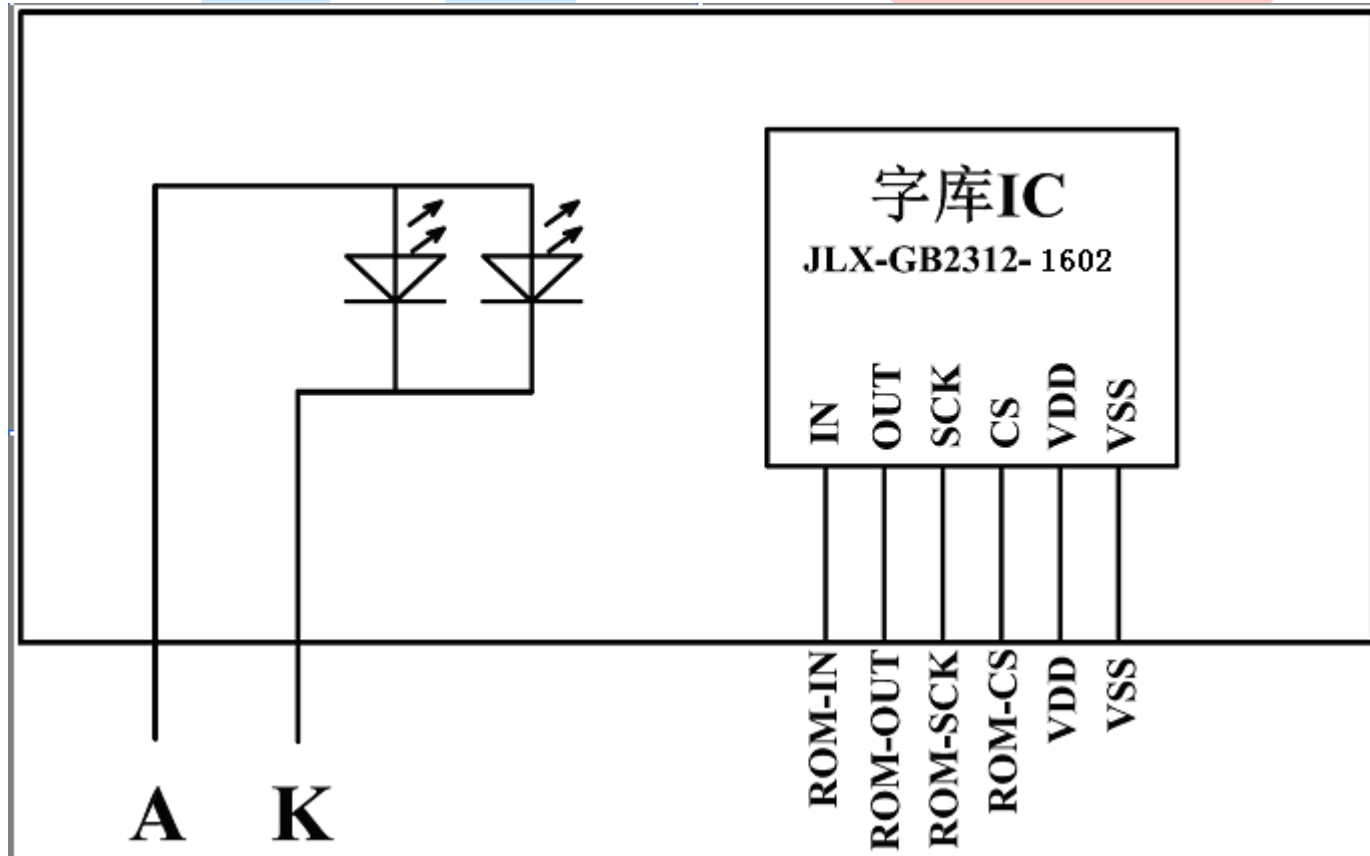
表 1：模块的接口引脚功能

引线号	符号	名称	功能
1	ROM_IN	字库 IC 接口	字库串行数据输入。
2	ROM_OUT	字库 IC 接口	字库串行数据输出。
3	ROM_SCK	字库 IC 接口	字库串行时钟。
4	ROM_CS	字库 IC 接口	字库片选输入。
5	LEDA	背光电源	背光电源正极，3.3V
6	VSS	接地	0V
7	VDD	电路电源	3.3V
8	A0 (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	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
10	CS	片选	低电平片选
11-18	D7-D0	I/O	数据总线 DB7-DB0
19	RD (E)	使能信号	使能信号
20	WR	读/写	H: 读数据 0: 写数据

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)。



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： $-20\sim+70^{\circ}\text{C}$ ；

存储温度： $-30\sim+80^{\circ}\text{C}$ ；

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	3.3	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		2.8	3.0	3.3	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“JLX177-00608”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):

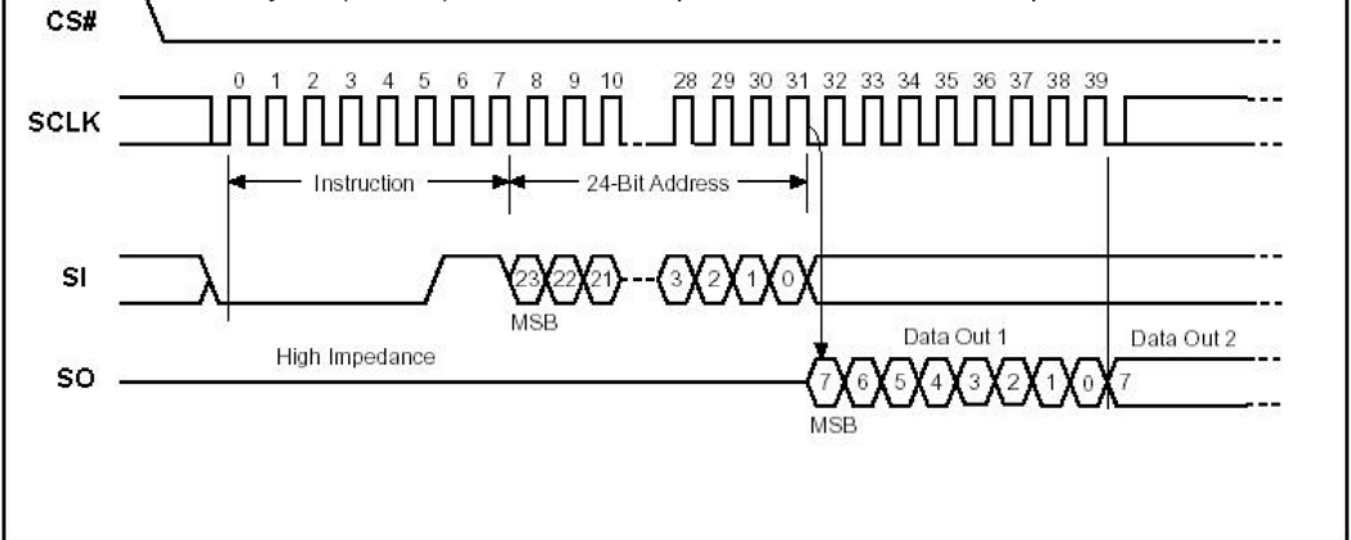
首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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):

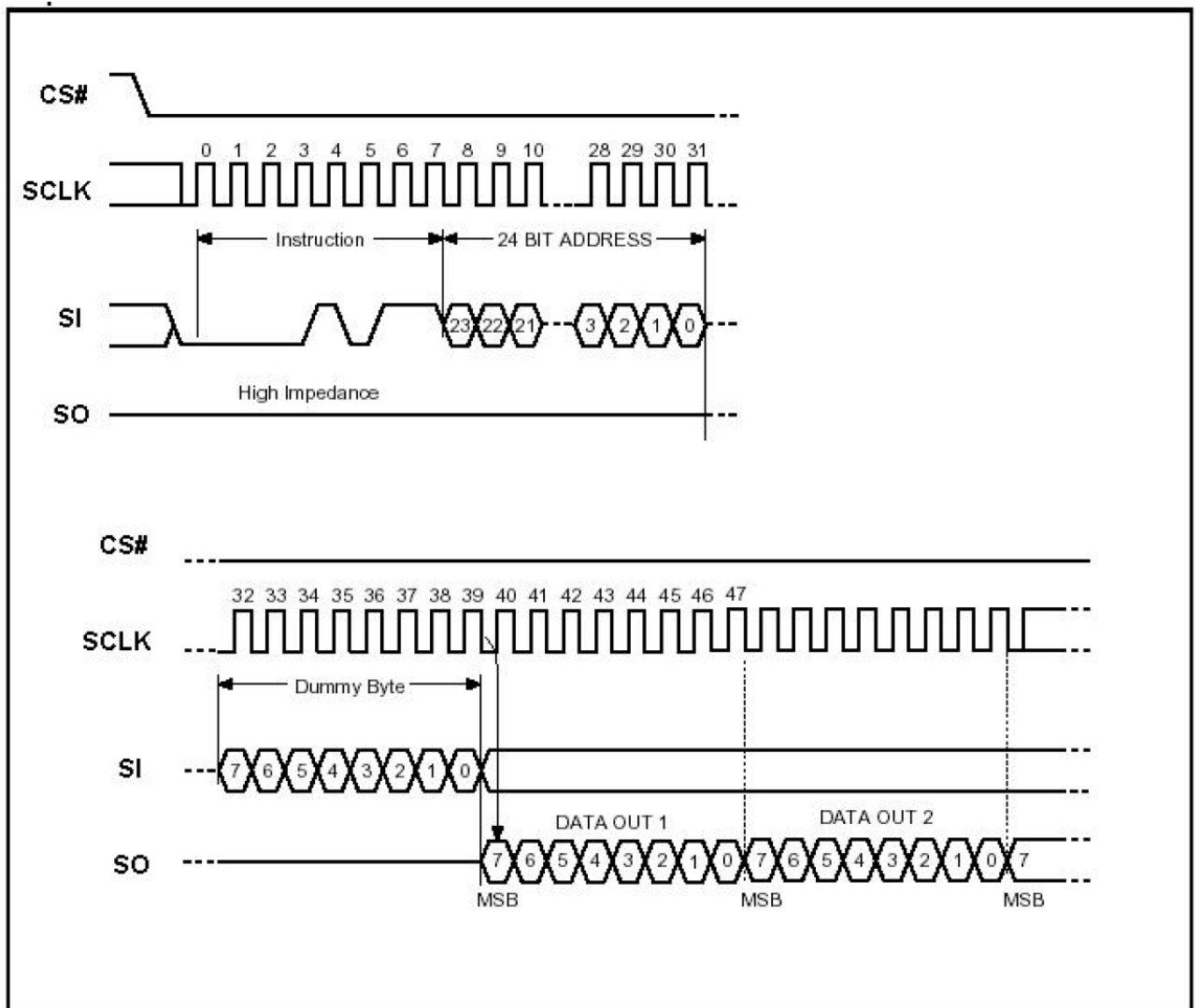
首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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



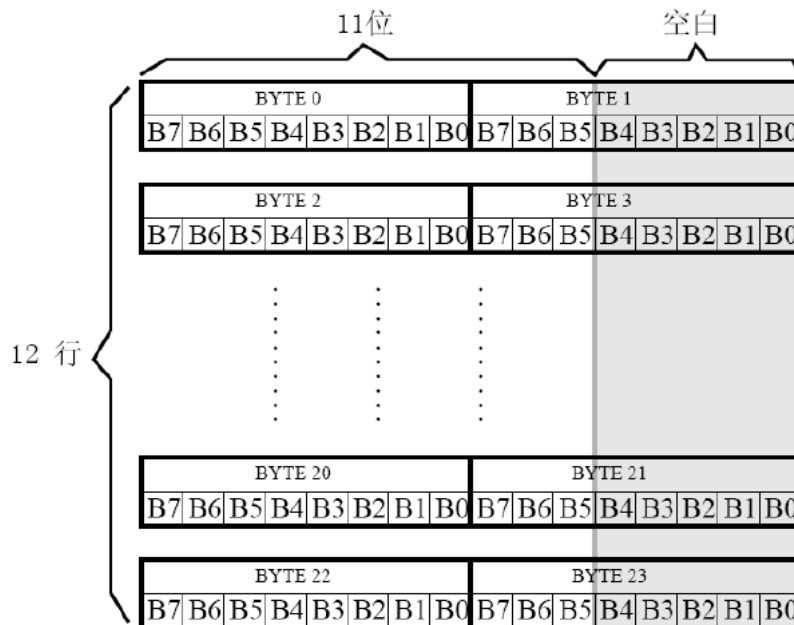
6 字库调用方法

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。

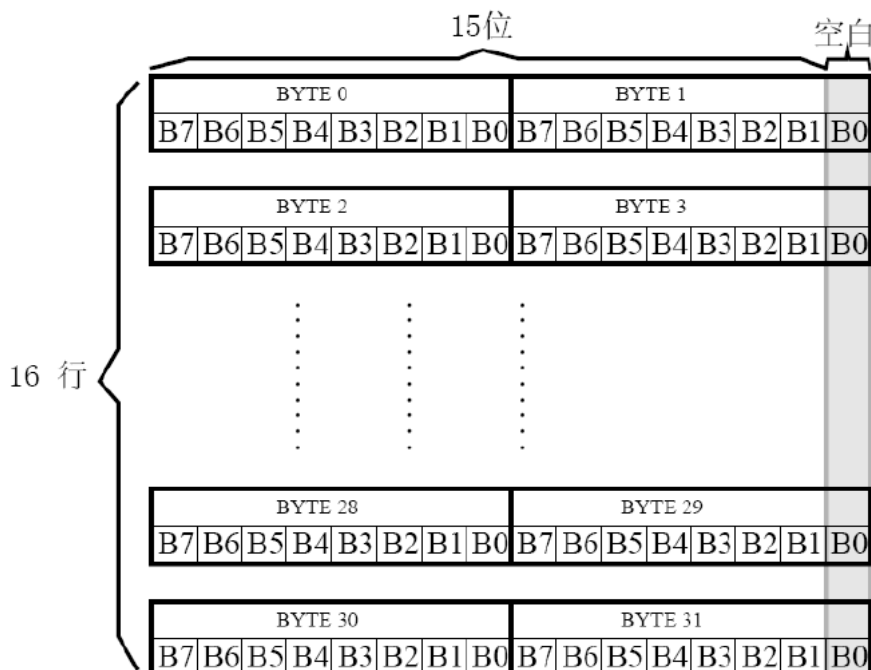
6.1.1 11X12 点汉字排列格式

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：



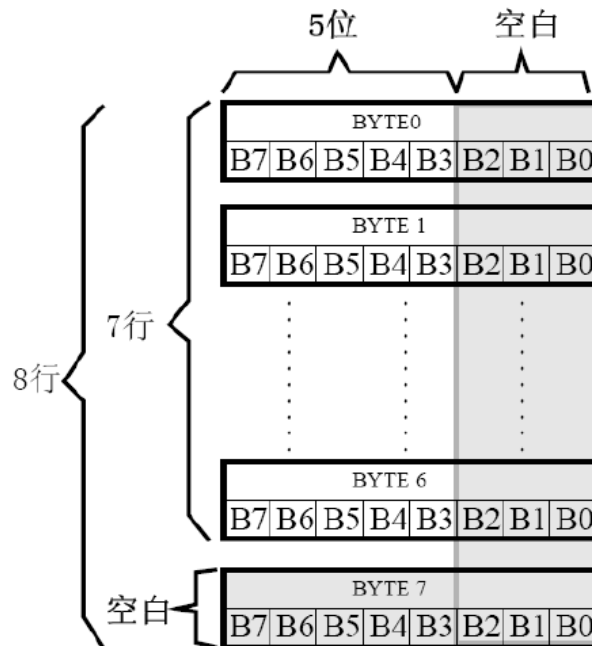
6.1.2 15X16 点汉字排列格式

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：



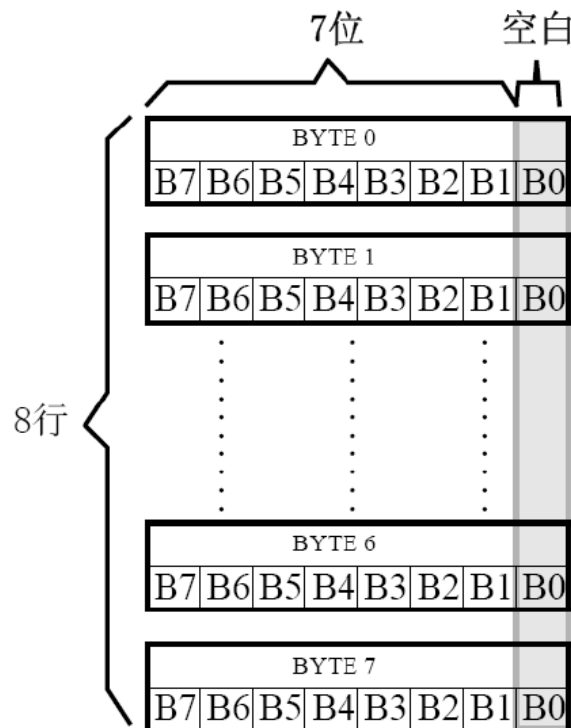
6.1.3 5X7 点ASCII 字符排列格式

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：



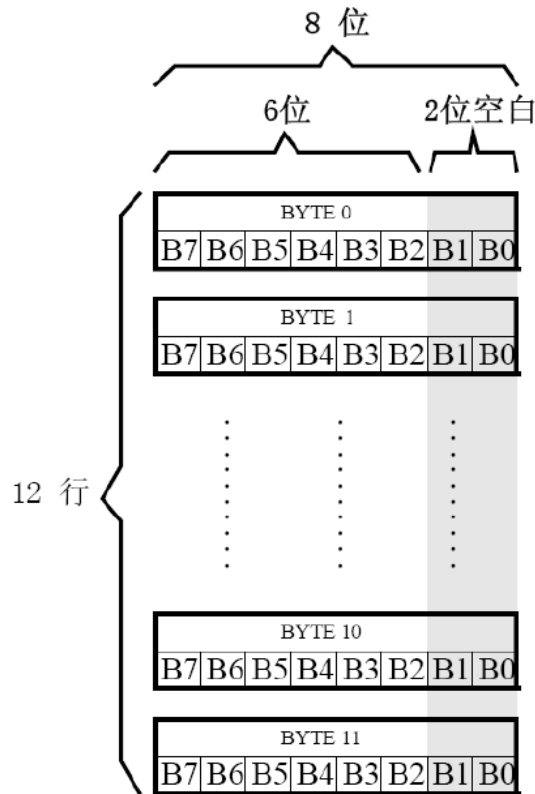
6.1.5 6X12 点字符排列格式

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：



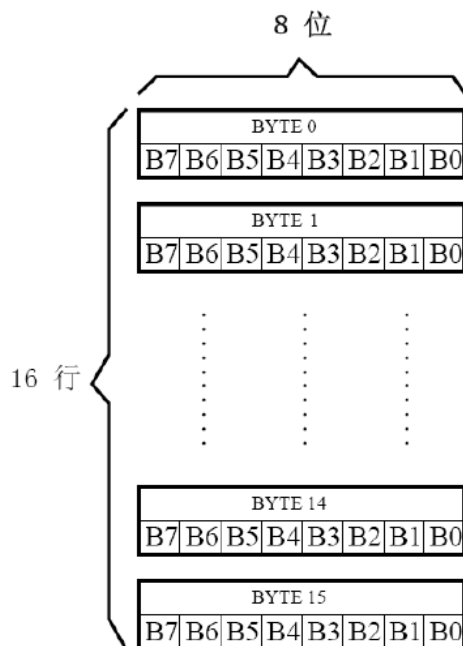
6.1.6 8X16 点字符排列格式

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

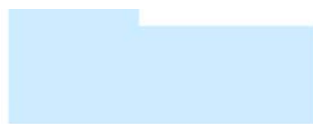
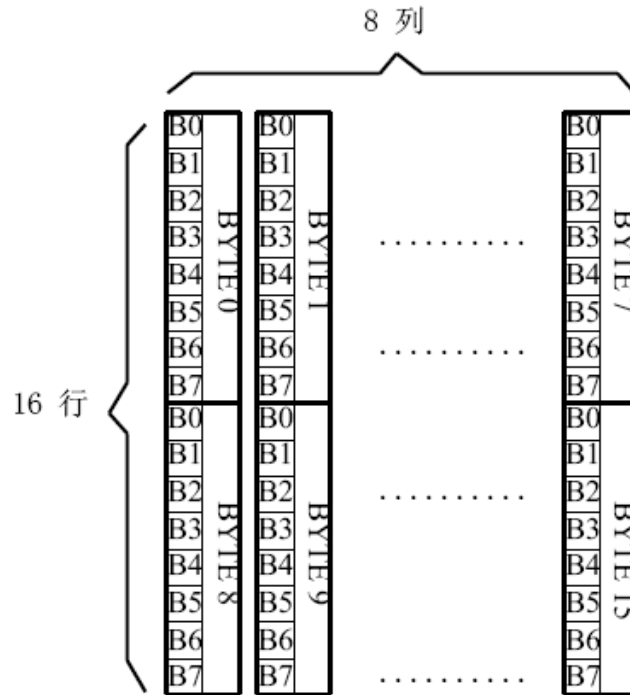
适用于此种排列格式的字有：

8X16 点 ASCII 字符

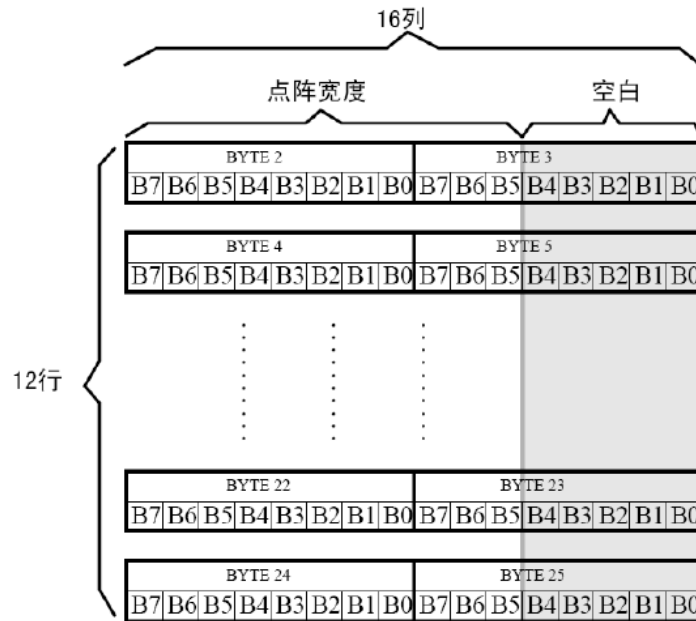
8X16 点 ASCII 粗体字符

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。

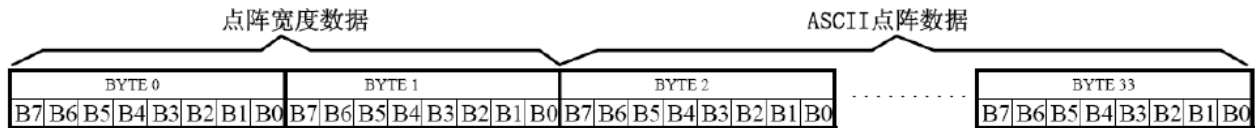


6.1.8 16 点阵不等宽ASCII 方头（Arial）字符排列格式

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

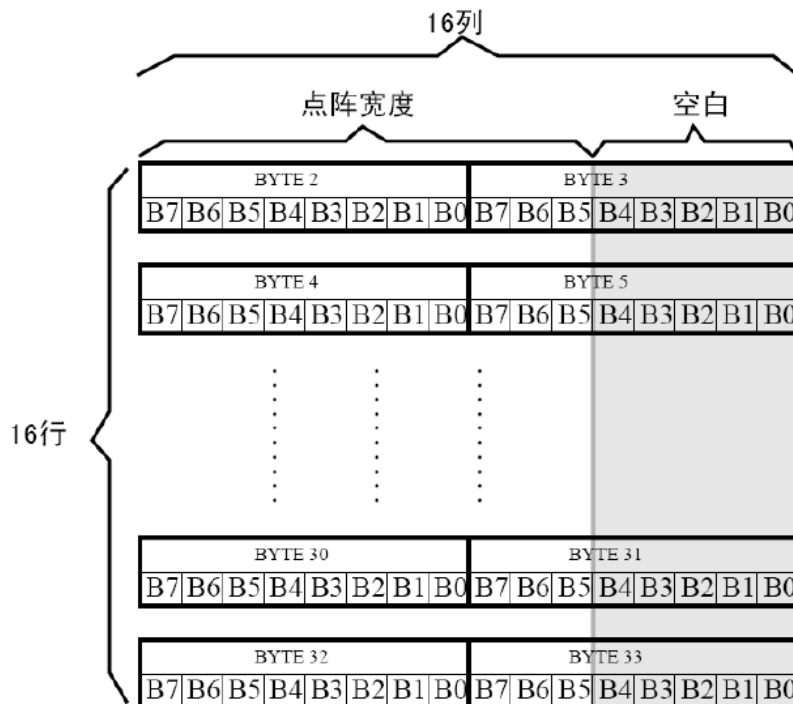
■ 存储格式

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	∩	⊙	∫	∫	=	≈	≈	∞	≠	≠	≠	≠	≠	∞	::	
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

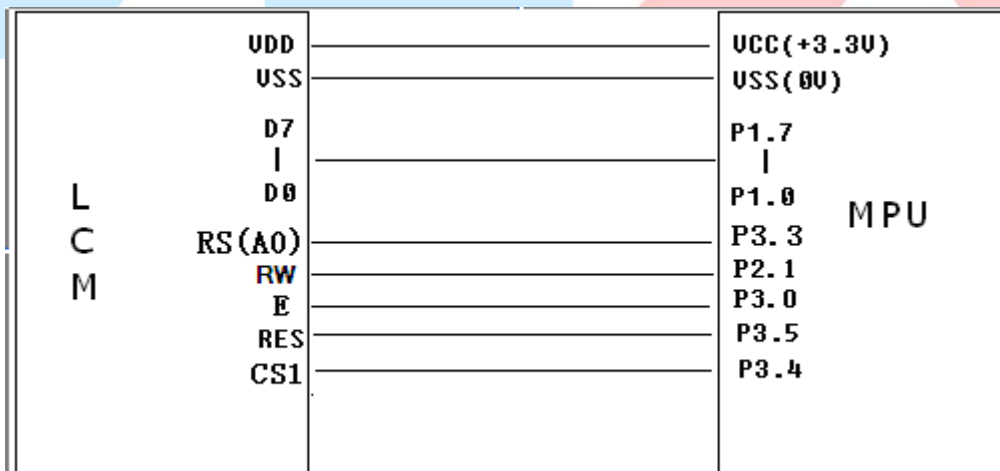
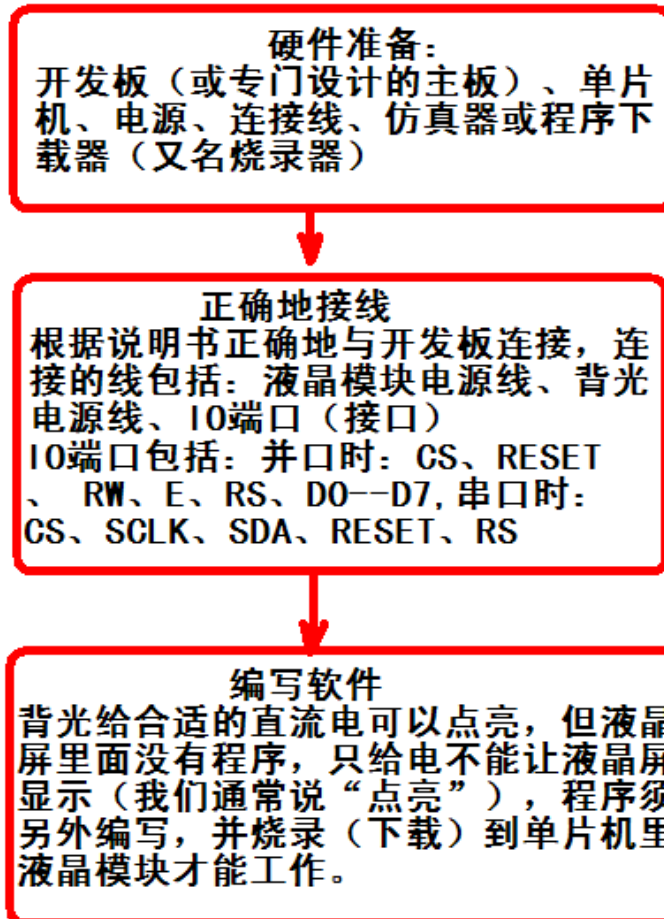
AA	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		!	"	#	¥	%	&	†	()	*	+	,	-	.	/	
B	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
C	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
D	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_		
E	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
F	p	q	r	s	t	u	v	w	x	y	z	{ }	~			

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	á	ń	ň	ñ	
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



```
#include <reg52.h>
#include <stdio.h>
#include <ASCII_TABLE_5X8_8X16_horizontal.h>
#include <128160_22.h>

//液晶屏 IC 所需要的信号线的接口定义
sbit RS=P3^3;
sbit WR=P2^1;
sbit E=P3^0;
sbit CS=P3^4;
sbit RST=P3^5;
sbit key=P2^0;

//字库 IC 所需要的 4 根信号线的接口定义
sbit Rom_CS=P3^6; //字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#
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_IN=P3^1; //字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI

//数据类型定义简化:
#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

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

```
        for (k=0;k<110;k++);
    }

//延时 2
//当用 STC12C5A60S2 单片机及晶振用 32MHz 时, K=2 就是 13 微秒
void delay_us(long int i)
{
    long int j,k;
    for (j=0;j<i;j++)
        for (k=0;k<1;k++);
}

//等待按键
void waitkey()
{
repeat:
    if(key==1) goto repeat;
    else delay_ms(300);
}

//传送 8 位的数据
void data_out(uchar data1)
{
    //8080 8bit interface
    CS = 0;
    RS= 1;
    E= 1;
    P1=data1;
    RW = 0;
    //delay_us(1);
    RW = 1;
    CS = 1;
}

//传送 8 位的命令
void comm_out(uchar com)
{
    //8080 8bit interface
    RS = 0;
    CS = 0;
    E = 1;
    P1 = com;
    RW = 0;
    delay_us(2);
    RW= 1;
    CS = 1;
}

//传 16 位数据, 16 位数据一起赋值
void data_out_16(int data_16bit)
```

```
{
    data_out(data_16bit >>8);
    data_out(data_16bit );
}
//LCD 初始化
void LCD_initial()
{
    delay_ms(150);
    RST=0;           //低电平：复位
    delay_ms(150);
    RST=1;           //高电平：复位结束
    delay_ms(30);
    //开始初始化：
    comm_out(0x11);
    //data_out(0x01,0x00);
    comm_out(0xb1);
    data_out(0x01);
    data_out(0x2c);
    data_out(0x2d);
    comm_out(0xb2);
    data_out(0x01);
    data_out(0x2c);
    data_out(0x2d);
    comm_out(0xb3);
    data_out(0x01);
    data_out(0x2c);
    data_out(0x2d);
    data_out(0x01);
    data_out(0x2d);
    data_out(0x2d);
    comm_out(0xb4);
    data_out(0x02);
    comm_out(0xb6);
    data_out(0xb4);
    data_out(0xf0);
    comm_out(0xc0);
    data_out(0xa2);
    data_out(0x02);
    data_out(0x84);
    comm_out(0xc1);
    data_out(0xc5);
    comm_out(0xc2);
    data_out(0x0a);
    data_out(0x00);
    comm_out(0xc3);
    data_out(0x8a);
    data_out(0x2a);
```

```

comm_out(0xc4);
data_out(0x8a);
data_out(0xee);
comm_out(0xc5);
data_out(0x0e);
    comm_out(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
data_out(0x08); //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)
    comm_out(0xff);
data_out(0x40);
data_out(0x03);
data_out(0x1a);
comm_out(0xfc);
data_out(0x11);
data_out(0x17);
comm_out(0xf0);
data_out(0x01);
    comm_out(0x3a);
data_out(0x05);
    comm_out(0xf6);
data_out(0x00);
comm_out(0xe0);
data_out(0x02);
data_out(0x1c);
data_out(0x07);
data_out(0x12);
data_out(0x37);
data_out(0x32);
data_out(0x29);
data_out(0x2d);
data_out(0x29);
data_out(0x25);
data_out(0x2b);
data_out(0x39);
data_out(0x00);
data_out(0x01);
data_out(0x03);
data_out(0x10);
comm_out(0xe1);
data_out(0x0b);
data_out(0x14);
data_out(0x09);
data_out(0x26);
data_out(0x27);
data_out(0x22);
data_out(0x1c);
data_out(0x20);

```



```

data_out(0x1d);
data_out(0x1a);
data_out(0x25);
data_out(0x2d);
data_out(0x06);
data_out(0x06);
data_out(0x02);
data_out(0x0f);
comm_out(0x2a); //定义 X 地址的开始及结束位置
data_out(0x00);
data_out(0x00);
data_out(0x00);
data_out(0x7F);

comm_out(0x2b); //定义 Y 地址的开始及结束位置
data_out(0x00);
data_out(0x00);
data_out(0x00);
data_out(0x9F);

comm_out(0x29); //开显示
}
//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;
    XE=XS+x_total-1;
    YE=YS+y_total-1;
    comm_out(0x2a); // 设置 X 开始及结束的地址
    data_out_16(XS); // X 开始地址(16 位)
    data_out_16(XE); // X 结束地址(16 位)

    comm_out(0x2b); // 设置 Y 开始及结束的地址
    data_out_16(YS); // Y 开始地址(16 位)
    data_out_16(YE); // Y 结束地址(16 位)

    comm_out(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_data_out(int mono_data,int font_color,int back_color)
{
    int i;
    for(i=0;i<8;i++)
    {
        if(mono_data&0x80)
        {
            data_out_16(font_color); //当数据是 1 时，显示字体颜色
        }
    }
}

```

```

    }
    else
    {
        data_out_16(back_color); //当数据是 0 时，显示底色
    }
    mono_data<<=1;
}
}
//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++) //显示 6 列，因为 5x8 点阵的字中间最好是隔 1 列，美观一点
    {
        if(mono_data&0x80)
        {
            data_out_16(font_color); //当数据是 1 时，显示字体颜色
        }
        else
        {
            data_out_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 display_graphic_5x8(int x, int y, char *dp, int font_color, int back_color)
{
    int k;
    LCD_CS=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++;
    }
}
//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;
    LCD_CS=0;
    Rom_CS = 1;
    lcd_address(x, y, 8, 16);
    for (k=0; k<16; k++)

```

```

    {
        mono_data_out(*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 display_graphic_16x16(uchar x, uchar y, char *dp, int font_color, int back_color)
```

```

{
    int i, j;
    LCD_CS=0;
    Rom_CS = 1;
    lcd_address(x, y, 16, 16);
    for (i=0; i<16; i++)
    {
        for (j=0; j<2; j++)
        {
            mono_data_out(*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;
    LCD_CS=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]”
display_graphic_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]”
display_graphic_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]”

            display_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+=8; //”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++;
}
}

```

//=====

//定义单个像素点的坐标 (x, y)

void LCD_SetPos(int x, int y)

```

{
    comm_out(0x2a); // 设置 X 开始及结束的地址

```

```

data_out_16(x); // X 开始地址(16 位)
data_out_16(x+1); // X 结束地址(16 位)

comm_out(0x2b); // 设置 Y 开始及结束的地址
data_out_16(y); // Y 开始地址(16 位)
data_out_16(y+1); // Y 结束地址(16 位)

comm_out(0x2c); // 写数据开始
}

```

//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<160; i++)
    {
        for(j=0; j<128; j++)
        {
            data_out_16(color);
        }
    }
}

```

//显示一幅全屏彩图

```

void display_image(uchar *dp)
{
    uchar i, j, k=0;
    lcd_address(0, 0, 128, 160);
    for(i=0; i<160; i++)
    {
        for(j=0; j<128; j++)
        {
            data_out(*dp); //传一个像素的图片数据的高位
            dp++;
            data_out(*dp); //传一个像素的图片数据的低位
            dp++;
        }
    }
}

```

//显示 5x8 点阵的字符串

```

void disp_string_5x8(int x, int y, uchar *text, int font_color, int back_color)

```

```

{
    uint i=0, j, k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x, y, 6, 8);
            for(k=0;k<8;k++)
            {
                mono_data_out_5x8(ascii_table_5x8[j*8+k], font_color, back_color); // 这个 "ascii_table_5x8[]" 这个数组在
"ASCII_TABLE_5X8_8X16_horizontal.h"
            }
            x+=6;
            i++;
        }
        else
            i++;
    }
}

```

//显示 8x16 点阵的字符串

```
void disp_string_8x16(int x, int y, char *text, int font_color, int back_color)
```

```

{
    int i=0, j, k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x, y, 8, 16);
            for(k=0;k<16;k++)
            {
                mono_data_out(ascii_table_8x16[j*16+k], font_color, back_color); // 这个 "ascii_table_8x16[]" 这个数组在
"ASCII_TABLE_5X8_8X16_horizontal.h"里
            }
            x+=8;
            i++;
        }
        else
            i++;
    }
}

```

//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 i, j;
lcd_address(x, y, 16, 16);
for(i=0; i<16; i++)
{
    for(j=0; j<2; j++)
    {
        mono_data_out(*dp, font_color, back_color);
        dp++;
    }
}
}

```

//画点函数

```

void draw_point(int x, int y, int color)
{
    LCD_SetPos(x, y);
    data_out_16(color);
}

```

//“画直线”函数的注意事项:

//直线的斜率 $k=(y_2-y_1)/(x_2-x_1)$, $k \setminus x_1 \setminus y_1 \setminus x_2 \setminus y_2$ 必须是浮点型的数据, 除此之外, 由于终点坐标可能小于起点坐标, 所以斜率还有可能是负数的。
//由于 $k=(y_2-y_1)/(x_2-x_1)$, 所以在 $x_2=x_1$ 时, 导致除数为 0, 数学上这是行不通的, 所以另外想办法。同理 $k_y=(x_2-x_1)/(y_2-y_1)$ 当 $y_2=y_1$ 时也是行不通的。
//另外 int 是 16 位的整数, uchar 却只有 8 位, 注意“color”是 16 位的哦! 所以别用“uchar color”

```

//
void draw_line(float x1, float y1, float x2, float y2, int color)
{
    int i; float k, k_y, x, z;
    if( (y2-y1)==0 && (x2-x1)!=0 ) //如果 y2-y1=0 且 x2-x1 不等于 0, 则画一条横线。
    {
        if(x2<x1) //如果 x2<x1, 则两个坐标互换, 因为画横线时, 从左到右与从右到左是一样的结果。
        {
            z=x2; x2=x1; x1=z;
        }
        lcd_address( x1, y1, (x2-x1+1), 1 ); //先设定画竖线的地址范围, 从(x1, y1)出发, 总长是 x2-x1+1, 这样子画线比用画点的方式要快很多, 因为只设置一次地址, 后面的是自动+1 的。
        for(i=0; i<=(x2-x1); i++)
        {
            data_out_16(color);
        }
    }
    else if( ((x2-x1)==0) && ((y2-y1)!=0) ) //如果 x2-x1=0 且 y2-y1 不等于 0, 则画一条竖线。
    {
        if(y2<y1) //如果 y2<y1, 则两个坐标互换, 因为画竖线时, 从上到下与从下到上是一样的结果。

```

```

    {
        z=y2; y2=y1;y1=z;
    }
    lcd_address( x1, y1, 1, (y2-y1+1) ); //先设定画横线的地址范围, 从(x1, y1) 出发, 总长是 y2-y1+1, 这样子画线比用画点的方式要快很多, 因为只设置一次地址, 后面的是自动+1 的。
    for (i=0; i<=(y2-y1); i++)
    {
        data_out_16(color);
    }
}
else if ((x2-x1)==0&&(y2-y1)==0) //如果 (x2=x1)且(y2=y1), 画一个点即可
{
    draw_point(x1, y2, color);
}

else //否则, 画斜线
{
    if(x2<x1) //如果 x2<x1, 则两个坐标互换, 因为 (从起点画到终点) 与 (从终点画到起点) 结果是一样的。
    {
        z=y2; y2=y1;y1=z;
        z=x2; x2=x1;x1=z;
    }
    if (fabs(y2-y1)<=fabs(x2-x1)) //如果 (y2-y1) 的绝对值小于等于 (x2-x1) 的绝对值, 就启动方案一: x 逐点扫描, y 按斜率计算,
    然后画点 (x+i, y)
    {
        k=(y2-y1)/(x2-x1); //k 是斜率
        for (i=0; i<=(x2-x1); i++)
        {
            draw_point((x1+i), (y1+k*i), color);
        }
    }
    else //如果 (y2-y1) 的绝对值大于 (x2-x1) 的绝对值, 就启动方案二: y 逐点扫描, x 按斜率计算,
    然后画点 (x, y+i) 或 (x, y-i)
    {
        k_y=fabs((x2-x1)/(y2-y1)); //k_y 是反斜率 (即 x 除以 y)。fabs 是浮点型数据的绝对值
        for (i=0; i<=abs(y2-y1); i++)
        {
            x=x1+k_y*i;
            if((y2-y1)>0)
            {
                draw_point(x, (y1+i), color);
            }
            else
            {
                draw_point(x, (y1-i), color);
            }
        }
    }
}

```

```

        }
    }
}

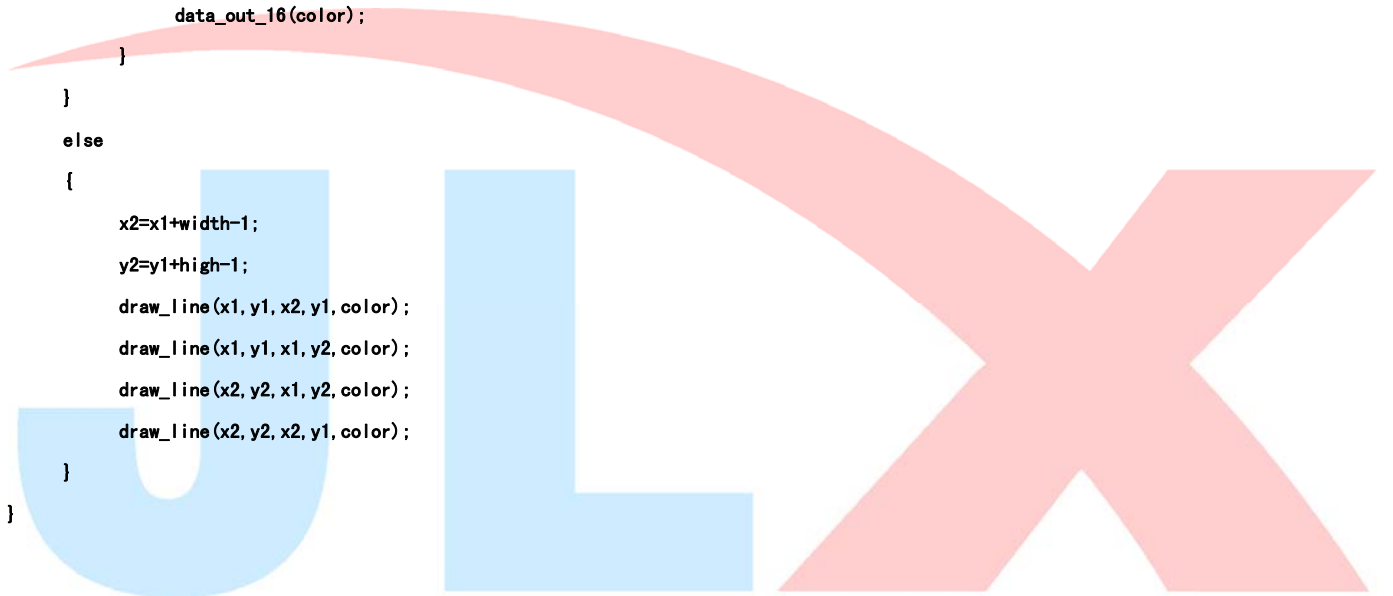
```

//画矩形函数

```

void draw_rectangle(float x1,float y1,float width,float high,int fill,int color)
{
    int i;float x2,y2;
    if(fill==1)
    {
        lcd_address(x1,y1,width,high);
        for(i=0;i<width*high;i++)
        {
            data_out_16(color);
        }
    }
    else
    {
        x2=x1+width-1;
        y2=y1+high-1;
        draw_line(x1,y1,x2,y1,color);
        draw_line(x1,y1,x1,y2,color);
        draw_line(x2,y2,x1,y2,color);
        draw_line(x2,y2,x2,y1,color);
    }
}

```



//画圆函数,算法一：以画规定起始地始及结束地址的方式画横线，逐步画出整个实心圆.速度快

```

void draw_circle(int x0,int y0,int R,int fill,int color)
{
    int x,y,RR,RS,i;//float r;
    if(fill==1)
    {
        RR=R*R+1; //补偿 1 修正方形
        RS=R*0.71; //0.71 大约是 cos45 度（45 度的余弦）。意思是先画半边，再用镜像的方法画别的。

        for(y=0;y<=RS;y++)
        {
            x=R; //初定为 R，等一下按圆的公式来计算
            while((x*x)>(RR-y*y))x--; //这是计算公式中最重要的：核对 x 平方是否大于 R 平方-y 平方，如果大于则减 1，直到满足条件。

            lcd_address((x0-x),(y0+y),2*x,1);

```

```

for (i=x0-x; i<x0+x; i++)
{
    data_out_16(color);
}
lcd_address((x0-x), (y0-y), 2*x, 1);
for (i=x0-x; i<x0+x; i++)
{
    data_out_16(color);
}
if (y>=1) //y=0 时, lcd_address(x0, (y0-x), 0, 1) 无效
{
    lcd_address((x0-y), (y0-x), 2*y, 1);
    for (i=x0-x; i<x0+x; i++)
    {
        data_out_16(color);
    }
    lcd_address((x0-y), (y0+x), 2*y, 1);
    for (i=x0-x; i<x0+x; i++)
    {
        data_out_16(color);
    }
}
else;
}
else
{
    y=R; //初定为 R, 等一下按圆的公式来计算
    RR=R*R+1; //补偿 1 修正方形
    RS=(y+(y>>1))>>1; //0.75 大约是 cos45 度, 45 度的余弦。意思是先画半边, 再用镜像的方法画别的。

    for (x=0; x<=RS; x++)
    {
        while((y*y)>(RR-x*x)) y--; //这是计算公式中最重要的: 核对 y 平方是否大于 R 平方-x 平方, 如果大于则减 1,
        直到满足条件。

        draw_point((x0+x), (y0-y), color); //第一象限的半截圆弧
        draw_point((x0+x), (y0+y), color); //第二象限的半截圆弧
        draw_point((x0-x), (y0-y), color); //第三象限的半截圆弧
        draw_point((x0-x), (y0+y), color); //第四象限的半截圆弧

        //镜像 45 度画另外半边圆弧
        draw_point((x0+y), (y0-x), color); //第一象限的另半截圆弧, 45 度镜像, 所以 x 变 y, y 变 x
        draw_point((x0+y), (y0+x), color); //第二象限的另半截圆弧, 45 度镜像, 所以 x 变 y, y 变 x
        draw_point((x0-y), (y0-x), color); //第三象限的另半截圆弧, 45 度镜像, 所以 x 变 y, y 变 x
        draw_point((x0-y), (y0+x), color); //第四象限的另半截圆弧, 45 度镜像, 所以 x 变 y, y 变 x
    }
}
}

```

```

}
uchar code jing[]={//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 lian[]={//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 xun[]={//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};

uchar code hua1[]={//横向取模
/*— 文字： 画 —*/
/*— 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x00,0x00,0xFF,0xFE,0x00,0x00,0x0F,0xE0,0x49,0x24,0x49,0x24,0x4F,0xE4,0x49,0x24,
0x49,0x24,0x49,0x24,0x4F,0xE4,0x48,0x24,0x40,0x04,0x7F,0xFC,0x40,0x04,0x00,0x00};

uchar code dian1[]={
/*— 文字： 点 —*/
/*— 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x01,0x00,0x01,0x00,0x01,0xF8,0x01,0x00,0x01,0x10,0x1F,0xF8,0x10,0x10,0x10,0x10,
0x10,0x10,0x1F,0xF0,0x10,0x10,0x01,0x10,0x28,0x88,0x24,0x44,0x44,0x00,0x00};

uchar code dun_hao[]={//顿号
/*— 文字： 、 —*/
/*— 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x80,0x00,0x30,0x00,0x10,0x00,0x00,0x00,0x00,0x00};

uchar code zhi1[]={
/*— 文字： 直 —*/
/*— 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x01,0x00,0x01,0x00,0x7F,0xFC,0x02,0x00,0x1F,0xF0,0x10,0x10,0x10,0x10,0x1F,0xF0,
0x10,0x10,0x1F,0xF0,0x10,0x10,0x1F,0xF0,0x10,0x10,0x10,0x10,0xFF,0xFE,0x00,0x00};

uchar code xian1[]={
/*— 文字： 线 —*/
/*— 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x10,0x40,0x18,0x50,0x10,0x4C,0x20,0x48,0x23,0xFC,0x48,0x40,0xF8,0x40,0x13,0xFE,

```

```
0x20, 0x40, 0x7C, 0x48, 0x00, 0x30, 0x00, 0x22, 0x1C, 0xD2, 0xE3, 0x0A, 0x00, 0x06, 0x00, 0x02, };
```

```
uchar code yuan1[]={
/*— 文字： 圆 —*/
/*— 宋体 12： 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x7F, 0xFC, 0x40, 0x04, 0x4F, 0xE4, 0x48, 0x24, 0x4F, 0xE4, 0x40, 0x04, 0x4F, 0xE4, 0x48, 0x24,
0x49, 0x24, 0x49, 0x24, 0x49, 0x24, 0x42, 0x84, 0x44, 0x44, 0x48, 0x24, 0x7F, 0xFC, 0x40, 0x04, };
```

```
uchar code ju1[]={
/*— 文字： 矩 —*/
/*— 宋体 12： 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x20, 0x00, 0x20, 0xFE, 0x20, 0x80, 0x3E, 0x80, 0x50, 0x80, 0x90, 0xFC, 0x10, 0x84, 0xFE, 0x84,
0x10, 0x84, 0x18, 0xFC, 0x14, 0x84, 0x22, 0x80, 0x22, 0x80, 0x40, 0x80, 0x80, 0xFE, 0x00, 0x00, };
```

```
uchar code xing1[]={
/*— 文字： 形 —*/
/*— 宋体 12： 此字体下对应的点阵为：宽 x 高=16x16 —*/
0x00, 0x04, 0x7F, 0x86, 0x12, 0x0C, 0x12, 0x10, 0x12, 0x20, 0x12, 0x08, 0xFF, 0xGC, 0x12, 0x18,
0x12, 0x20, 0x12, 0x44, 0x12, 0x86, 0x12, 0x0C, 0x22, 0x10, 0x22, 0x20, 0x42, 0x40, 0x80, 0x80, };
```

//一些难检字（在 GB2312 标准简体汉字库的 6763 个汉字里也找不到的）取模数据：

//取模方法请找客服人员索要

```
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, 0x44, 0x24, 0x48, 0x34, 0x50, 0x14, 0x6F, 0xE4,
0x48, 0x24, 0x48, 0x24, 0x48, 0x24, 0x48, 0x24, 0x48, 0x24, 0x48, 0x24, 0x7F, 0xFC, 0x00, 0x00,
};
```

```
uchar code lei1[]={//横向取模
/*— 文字： 畀 —*/
/*— 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,
};
```

//字库测试

```
void test_ZK()
{
    comm_out(0x36); //行扫描顺序，列扫描顺序，横放/竖放
    data_out(normal); //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”——在竖放基础上顺时针转 90 度横放；
    display_color(0, 0, 128, 160, 0x0000); //全屏写 0x0000，相当于清屏。参数分别是 (x_start, y_start, x_total, y_total, 颜色)
    delay_ms(10);
    display_GB2312_string(0, 16*0, "128x160 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:℃", 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);
display_graphic_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)

display_graphic_16x16(16*7, 16*7, lei1, deep_green, white);
display_string_5x8(0, 16*8+8*0, "<!@#%^&*()-+]/;.:?[ ", blue, white); //显示一串 5x8 点阵的 ASCII 字, 参数分别是 (x, y, " 字符
", 字符颜色, 背景颜色)

display_string_5x8(0, 16*8+8*1, "JLX electronics Co., ", blue, white);

display_string_5x8(0, 16*8+8*2, "http://www.jlxlcd.cn ", pink, white);
display_string_5x8(0, 16*8+8*3, "TEL:0755-29784961 ", pink, white);

waitkey();

comm_out(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
data_out(CW180); //定义: "normal"就是 "0xc8" ——正常竖放;
//定义: "CW180"就是 "0xc8" ——在正常竖放基础上转 180 度竖放;
//定义: "CW90" 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, 128, 160, 0x0000); //全屏写 0x0000, 相当于清屏。
delay_ms(10);
display_GB2312_string(0, 16*0, "128x160 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:℃", 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); //有图型功能, 可
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);
display_graphic_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)

display_graphic_16x16(16*7, 16*7, lei1, deep_green, white);
display_string_5x8(0, 16*8+8*0, "<!@#%^&*()-+]/;.:?[ ", blue, white); //显示一串 5x8 点阵的 ASCII 字, 参数分别是 (x, y, " 字符
", 字符颜色, 背景颜色)

display_string_5x8(0, 16*8+8*1, "JLX electronics Co., ", blue, white);

display_string_5x8(0, 16*8+8*2, "http://www.jlxlcd.cn ", pink, white);
display_string_5x8(0, 16*8+8*3, "TEL:0755-29784961 ", pink, white);

waitkey();

```

```

comm_out(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
data_out(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(0,0,160,128,0x0000); //全屏写 0x0000, 相当于清屏。
delay_ms(10);
display_GB2312_string(0,16*0,"160x128 带汉字库彩屏,",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 汉字或符号:②℃",white,orange);
display_GB2312_string(0,16*2,"或 8X16 点阵 ASCII,@#%$",white,deep_green);
display_GB2312_string(0,16*3,"GB2312 简体字库及自编",white,blue);
display_GB2312_string(0,16*4,"图像,生僻字,例: ",red,white);

display_graphic_16x16(16*8,16*4,jiong1,red,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)
display_graphic_16x16(16*9,16*4,lei1,red,white);

display_string_5x8(0,16*5+8*0,"<!@#%&*()_+]/;.:?>[]`~",orange,white); //显示一串 5x8 点阵的 ASCII 字,参数分别是 (x, y,"
字符 ", 字符颜色, 背景颜色)
display_string_5x8(0,16*5+8*1,"ShenZhen JLX electronics ",orange,white);
display_string_5x8(0,16*5+8*2,"http://www.jlxlcd.cn ",deep_green,white);
display_string_5x8(0,16*5+8*3,"TEL:0755-29784961 ",deep_green,white);

display_GB2312_string(0,16*7,"有画点线圆矩形的例程",blue,white);

waitkey();
comm_out(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
data_out(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(0,0,160,128,0x0000); //全屏写 0x0000, 相当于清屏。
delay_ms(10);
display_GB2312_string(0,16*0,"160x128 带汉字库彩屏,",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 汉字或符号:②℃",white,orange);
display_GB2312_string(0,16*2,"或 8X16 点阵 ASCII,@#%$",white,deep_green);
display_GB2312_string(0,16*3,"GB2312 简体字库及自编",white,blue);
display_GB2312_string(0,16*4,"图像,生僻字,例: ",red,white);

display_graphic_16x16(16*8,16*4,jiong1,red,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)

```



```

display_graphic_16x16(16*9, 16*4, lei1, red, white);

display_string_5x8(0, 16*5+8*0, "<@#$$%^&*()_+]/;.:?>[]`~", orange, white); //显示一串 5x8 点阵的 ASCII 字, 参数分别是 (x, y, "
字符", 字符颜色, 背景颜色)

display_string_5x8(0, 16*5+8*1, "ShenZhen JLX electronics", orange, white);
display_string_5x8(0, 16*5+8*2, "http://www.jlxlcd.cn", deep_green, white);
display_string_5x8(0, 16*5+8*3, "TEL:0755-29784961", deep_green, white);

display_GB2312_string(0, 16*7, "有画点线圆矩形的例程", blue, white);

waitkey();
}

void draw_line_circle()
{
    comm_out(0x36); //行扫描顺序, 列扫描顺序, 横放/竖放
    data_out(normal); //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;

//画线:
    draw_rectangle(0, 80, 128, 160-80, 1, black); //指定范围显示一种颜色, 如果全屏写黑色, 相当于清屏。 参数分别是 (x1, y1, 宽, 高, 颜色)

    draw_line(63, 102, 63+20, 102, red);

    draw_line(63, 102, 63+8, 102+18, orange); //从 (x1, y1) 到 (x2, y2) 画一条线段, 请注意如果 x2=x1 且 y2=y1, 表示画一个点
    draw_line(63, 102, 63+14, 102+14, yellow);
    draw_line(63, 102, 63+18, 102+8, green);

    draw_line(63, 102, 63, 102+20, blue);

    draw_line(63, 102, 63-8, 102+18, purple);
    draw_line(63, 102, 63-14, 102+14, gray);
    draw_line(63, 102, 63-18, 102+8, red);

    draw_line(63, 102, 63-20, 102, red);

    draw_line(63, 102, 63-8, 102-18, orange);
    draw_line(63, 102, 63-14, 102-14, yellow);
    draw_line(63, 102, 63-18, 102-8, green);

    draw_line(63, 102, 63, 102-20, blue);

    draw_line(63, 102, 63+8, 102-18, purple);

```

```
draw_line(63,102,63+14,102-14,gray);
draw_line(63,102,63+18,102-8,red);
```

//画矩形:

```
draw_rectangle(0,82,40,20,1,blue); //画矩形,参数分别是(x,y,宽,高,填充与否,颜色)
draw_rectangle(10,87,20,10,1,yellow);

draw_rectangle(88,82,40,20,0,blue);
draw_rectangle(98,87,20,10,0,yellow);
```

//画圆形:

```
draw_circle(25,134,25,1,pink); //画圆,(圆心x,圆心y,半径,填充与否,颜色)
draw_circle(25,134,20,1,green);
draw_circle(25,134,15,1,blue);
draw_circle(25,134,10,1,red);
draw_circle(25,134,5,1,blue);
```

```
draw_circle(102,134,25,0,pink);
draw_circle(102,134,20,0,green);
draw_circle(102,134,15,0,blue);
draw_circle(102,134,10,0,red);
draw_circle(102,134,5,0,blue);
```

//画点:

```
draw_point(22,134,white); //在(x,y)位置显示一个点,第三个参数是点的颜色
draw_point(23,134,white);
draw_point(24,134,white);
draw_point(25,134,white);
draw_point(26,134,white);
draw_point(27,134,white);
draw_point(28,134,white);
```

```
draw_point(25,131,white); //在(x,y)位置显示一个点,第三个参数是点的颜色
draw_point(25,132,white);
draw_point(25,133,white);
draw_point(25,134,white);
draw_point(25,135,white);
draw_point(25,136,white);
draw_point(25,137,white);
waitkey();
```

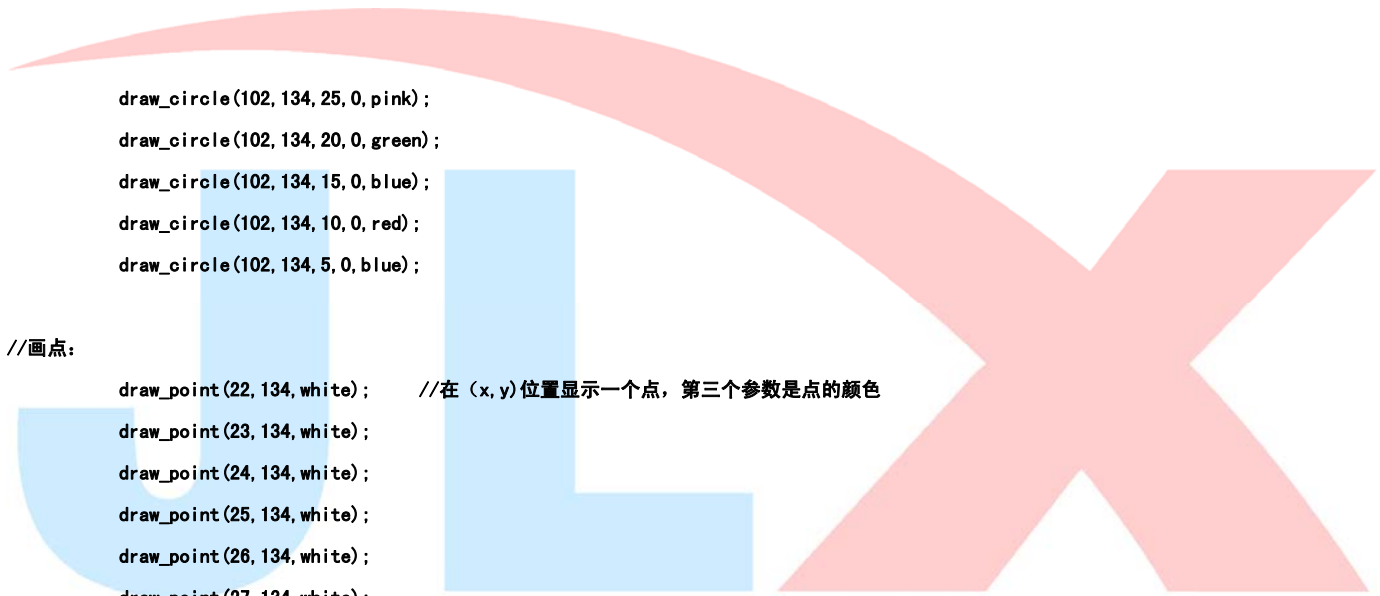
}

//全屏显示单色,及彩图

```
void disp_color_bmp()
```

```
{
```

//全屏显示单色:



```

display_color(0,0,128,160,red);
waitkey();
display_color(0,0,128,160,green);
waitkey();
display_color(0,0,128,160,blue);
waitkey();
display_color(0,0,128,160,black);
waitkey();
display_color(0,0,128,160,white);
waitkey();
//显示全屏的彩图:
display_image(pic1);
waitkey();
}

//写自编的汉字、8x16、5x8 字符:
void disp_self_create_font()
{
    comm_out(0x36); //行扫描顺序,列扫描顺序,横放/竖放
    data_out(normal); //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"——在竖放基础上顺时针转90度横放;
//写汉字、8x16、5x8 字符:
display_color(0,0,128,160,black); //全屏写0x0000,相当于清屏。
delay_ms(10);
display_color(0,0,128,160,white); //全屏写白色。
disp_16x16(8+16*2,0,jing,blue,white);
disp_16x16(8+16*3,0,lian,blue,white);
disp_16x16(8+16*4,0,xun,blue,white);

disp_16x16(16*0,16,dian1,blue,white); //显示汉字“点”
disp_16x16(16*1,16,dun_hao,blue,white);
disp_16x16(16*2,16,xian1,blue,white);
disp_16x16(16*3,16,dun_hao,blue,white);
disp_16x16(16*4,16,yuan1,blue,white);
disp_16x16(16*5,16,dun_hao,blue,white);
disp_16x16(16*6,16,ju1,blue,white);
disp_16x16(16*7,16,xing1,blue,white);

disp_string_8x16(0,32,"8*16 dots ASCII",blue,white); //在(x,y)位置开始,显示ASCII字符串
disp_string_5x8(0,48,"display 5x8 dots char",blue,white); //在(x,y)位置开始,显示ASCII字符串
disp_string_5x8(0,56,"(159,119),color=blue",blue,white); //在(x,y)位置开始,显示ASCII字符串
disp_string_5x8(0,64,"~!@#%^&*O_+|/?><.,",blue,white); //在(x,y)位置开始,显示ASCII字符串

```

```
    disp_string_5x8( 0, 72, "[]0123456789abcxyzAB", blue, white); //在 (x, y) 位置开始, 显示 ASCII 字符串
}

//主程序
void main(void)
{
    LCD_initial();           //初始化
    while(1)
    {
        test_ZK();           //调用字库 1G 的汉字、字符数据显示在液晶屏上
        disp_self_create_font(); //自编的汉字、字符数据显示在液晶屏上, 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。
        draw_line_circle();   //画点、线、圆、矩形
        disp_color_bmp();     //显示单色全屏、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。
    }
}
```

