JLX280-012-PC 使用说明书

(带字库 IC)

目 录

1．字库

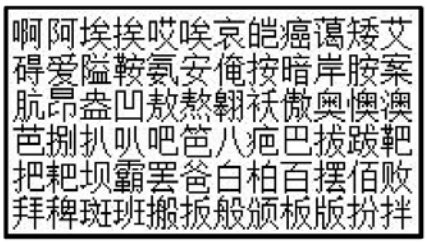
字库 IC(IC 型号：JLX-GB2312-3205，此 IC 为可选的配件)自带字库内容：

分类	字库内容	编码体系（字符集）	字符数
汉字字符	11X12 点 GB2312 标准点阵字库	GB2312	6763+846
	15X16 点 GB2312 标准点阵字库	GB2312	6763+846
	24X24 点 GB2312 标准点阵字库	GB2312	6763+846
	32X32 点 GB2312 标准点阵字库	GB2312	6763+846
	6X12 点国标扩展字符	GB2312	126
	8X16 点国标扩展字符	GB2312	126
	12X24 点国标扩展字符	GB2312	126
	16X32 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
	8X16 点粗体 ASCII 字符	ASCII	96
	12 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96
	12 点阵不等宽 ASCII 白正（Times New Roman）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（Times New Roman）字符	ASCII	96
	24 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96
	24 点阵不等宽 ASCII 白正（Times New Roman）字符	ASCII	96
	32 点阵不等宽 ASCII 方头（Arial）字符	ASCII	96
	32 点阵不等宽 ASCII 白正（Times New Roman）字符	ASCII	96
输入法码表	全拼输入法码表	GB2312	

字型样张

11X12 点 GB2312 汉字

15X16 点 GB2312 汉字



24X24 点 GB2312 汉字

32X32 点 GB2312 汉字

5x7 点 ASCII 字符

7x8 点 ASCII 字符

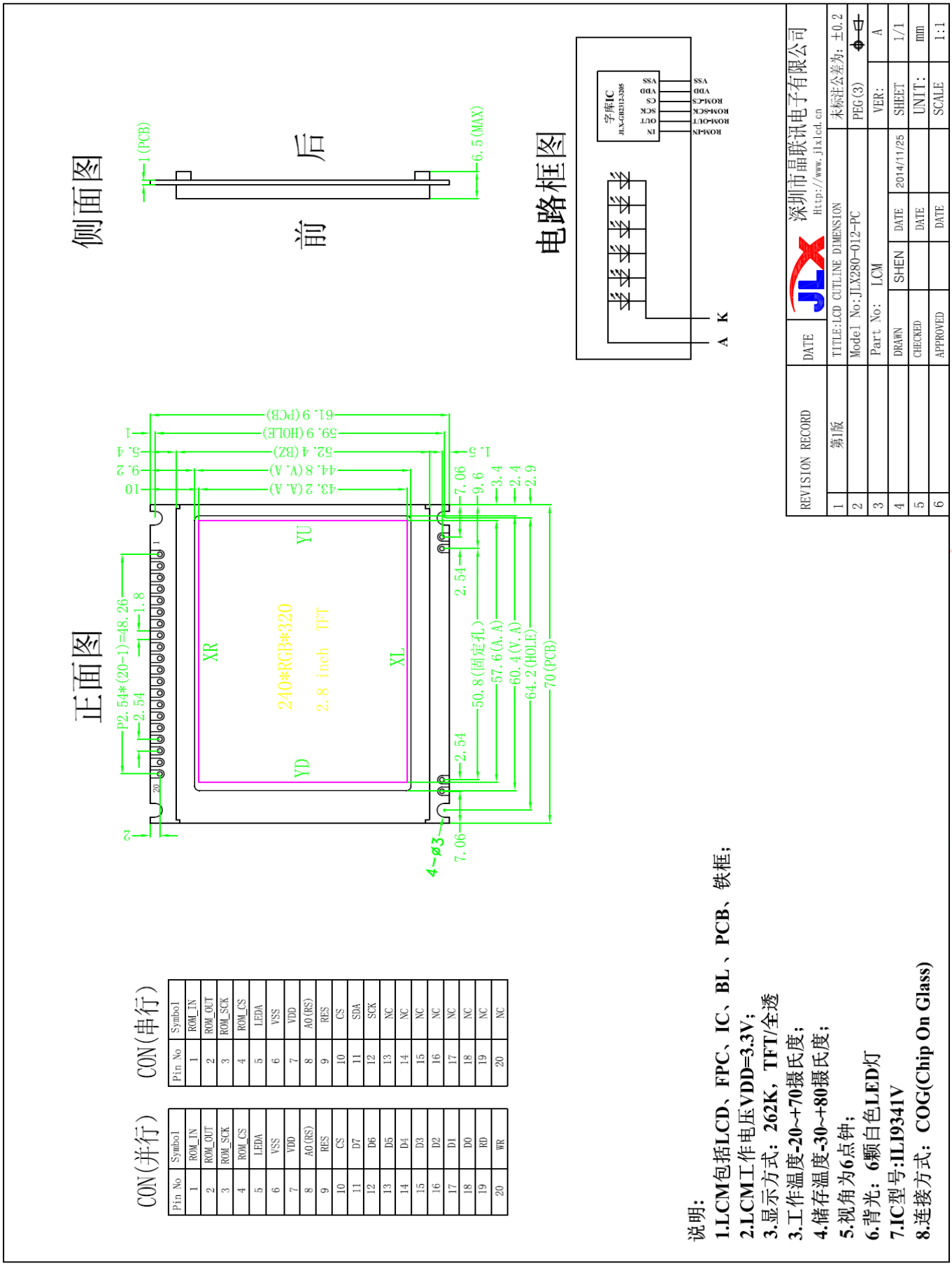
6x12 点 ASCII 字符

8x16 点 ASCII 字符

12 点阵不等宽 ASCII 方头

16 点阵不等宽 ASCII 方头

2. 外形尺寸及接口引脚功能



CON(并行)

Pin No	Symbol
1	ROM_IN
2	ROM_OUT
3	ROM_SCK
4	ROM_CS
5	LEDA
6	VSS
7	VDD
8	AO (RS)
9	RES
10	CS
11	D7
12	D6
13	D5
14	D4
15	D3
16	D2
17	D1
18	D0
19	RD
20	WR

CON(串行)

Pin No	Symbol
1	ROM_IN
2	ROM_OUT
3	ROM_SCK
4	ROM_CS
5	LEDA
6	VSS
7	VDD
8	AO (RS)
9	RES
10	CS
11	SDA
12	SCK
13	NC
14	NC
15	NC
16	NC
17	NC
18	NC
19	NC
20	NC

说明：

1.LCM包括LCD、FPC、IC、BL、PCB、铁框；
2.LCM工作电压VDD=3.3V；
3.显示方式：262K，TFT/全透
3.工作温度-20~+70摄氏度；
4.储存温度-30~+80摄氏度；
5.视角为6点钟；
6.背光：6颗白色LED灯
7.IC型号:ILI9341V
8.连接方式：COG(Chip On Glass)

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，3.0V
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 320×240 点阵,240 个列信号与驱动 IC 相连，320 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∽+70° C;
存储温度:-30∽+80° C;
背光板是白色。
正常工作电流为：48∽120mA（LED 灯数共 6 颗,每颗灯是 10～20 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	℃
储存温度		-30		+80	℃

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 5 颗 LED 灯并联	48	90	120	mA

表 3：直流（DC）参数

4.3 LCD 驱动 IC 指令表详见“JLX320-00206”的中文说明书

5.1 字库 IC（JLX-GB2312-3204）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：

5.2.1.1 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)：

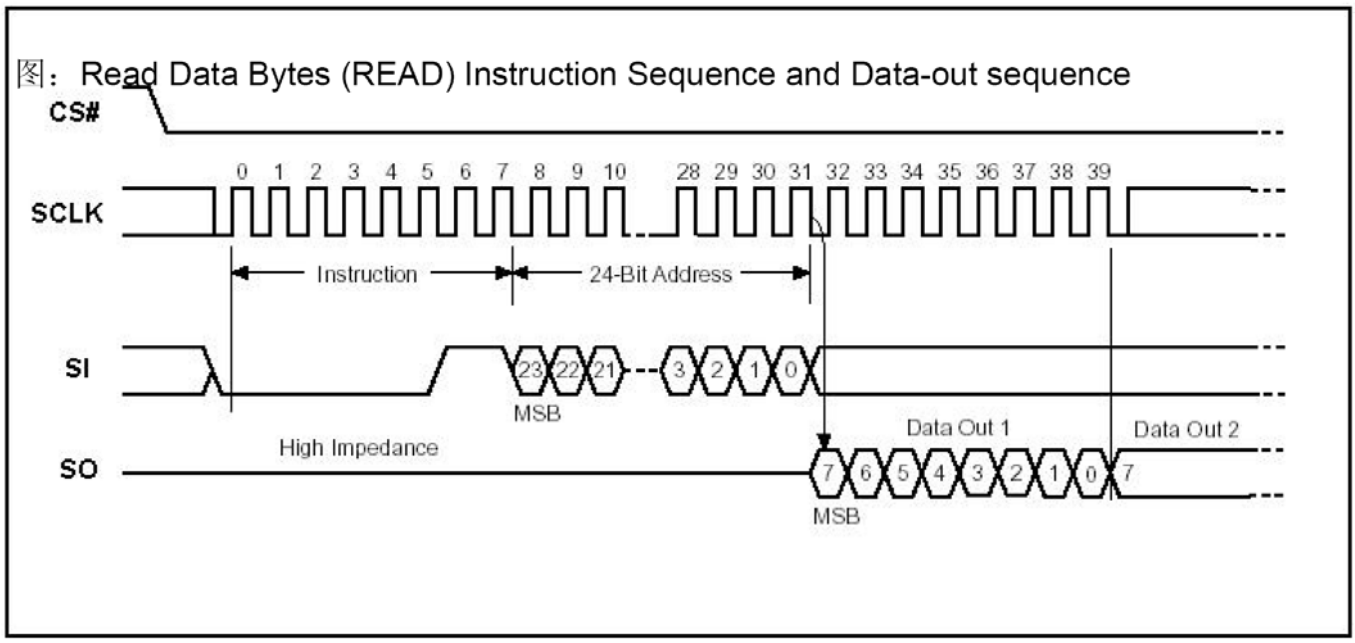
首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（03 h）和 3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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)：

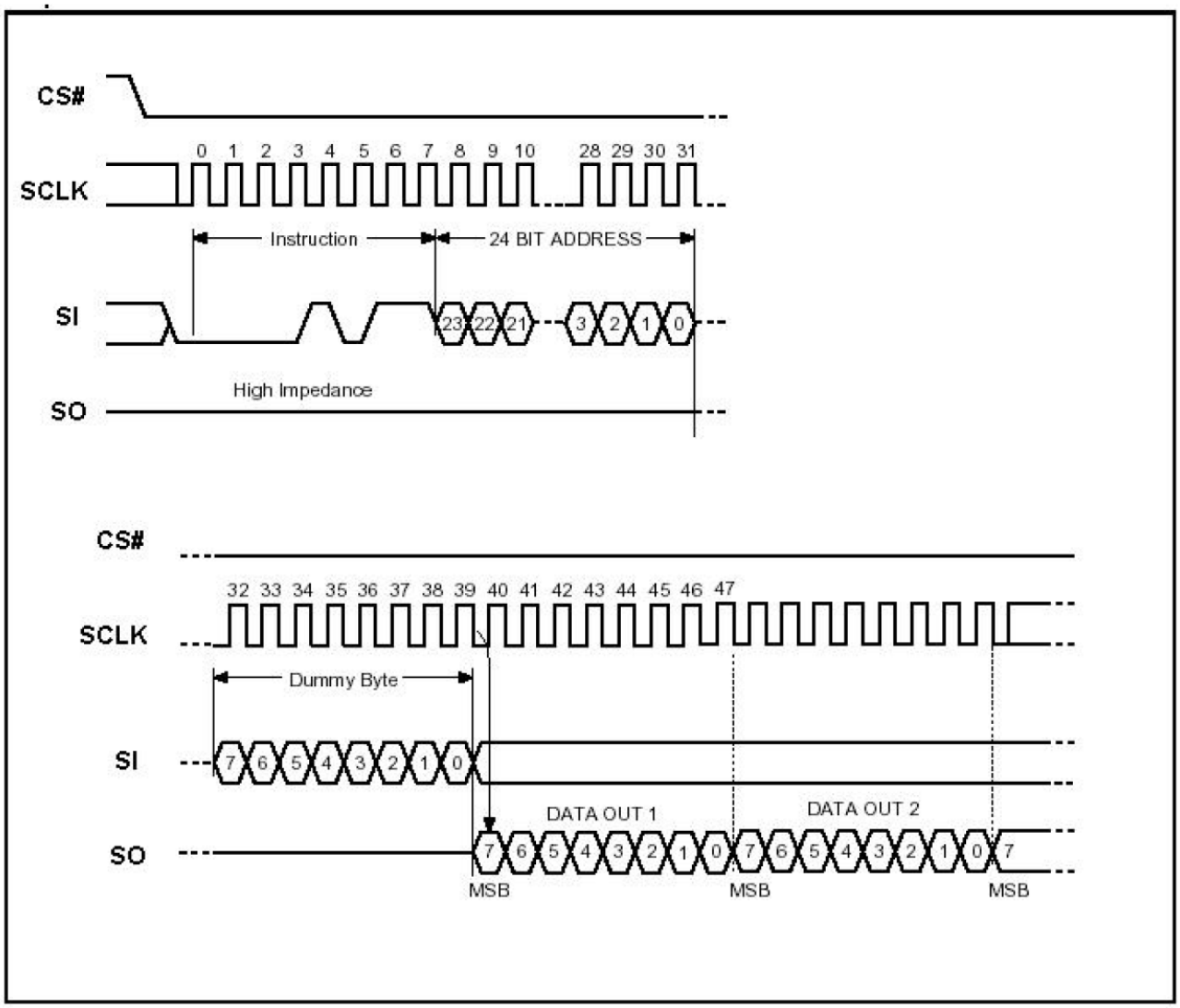
首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（0B h）和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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



5.2.1字库调用方法：

5.2.2.1 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：即一个字节的高位表示左面的点，低位表示右面的点（如果用户按 word mode 读取点阵数据，请注意高低字节的顺序），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。

5.2.2.2 11X12 点、15X16点、24X24点、32X32点汉字及5X7 点、7X8 点、6X12点、12X24点字符、12 点阵不等宽字符、16点阵不等宽字符的排列格式：详见字库IC资料（JLX-GB2312-32S4W）的第19-26页。

5.2.2.3 汉字点阵字库地址表如下：

	字库内容	编码体系	码位范围	字符数	起始地址	参考算法
1	11X12 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+846	0000	6.3.1.1
2	15X16 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+846	2C9D0	6.3.1.2
3	24X24 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+846	68190	6.3.1.3
4	32X32 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+846	EDF00	6.3.1.4
5	6X12 点国标扩展字符	GB2312	A1A1-ABC0	126	1DBE0C	6.3.1.5
6	6X12 点 ASCII 字符	ASCII	20~7F 96		1DBE00	6.3.2.3
7	12 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F 96		1DC400	6.3.2.7
8	12 点阵不等宽 ASCII 白正字符	ASCII	20~7F	96	1DCDC0	6.3.2.8
9	8X16 点国标扩展字符	GB2312	A1A1-ABC0	126	1DD790	6.3.1.6
10	8X16 点 ASCII 字符	ASCII	20~7F 96		1DD780	6.3.2.4
11	5X7 点 ASCII 字符	ASCII	20~7F 96		1DDF80	6.3.2.1
12	7X8 点 ASCII 字符	ASCII	20~7F 96		1DE280	6.3.2.2
13	16 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F 96		1DE580	6.3.2.9
14	16 点阵不等宽 ASCIII 白正字符	ASCII	20~7F	96	1DF240	6.3.2.10
15	12X24 点国标扩展字符	GB2312	A1A1-ABC0	126	1DFF30	6.3.1.8
16	12X24 点 ASCII 字符	ASCII	20~7F 96		1DFF00	6.3.2.5
17	24 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F 96		1E22D0	6.3.2.11
18	24 点阵不等宽 ASCII 白正字符	ASCII	20~7F	96	1E3E90	6.3.2.12
19	16X32 点国标扩展字符	GB2312	A1A1-ABC0	126	1E5A90	6.3.1.9
20	16X32 点 ASCII 字符	ASCII	20~7F 96		1E5A50	6.3.2.6
21	32 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F 96		1E99D0	6.3.2.13
22	32 点阵不等宽 ASCII 白正字符	ASCII	20~7F	96	1ECA90	6.3.2.14
23	保留区				1EFB50	
29	输入法码表	GB2312			1F36F0	
32	保留区				1F7CC8	

5.2.2.4 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。

举例说明: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。

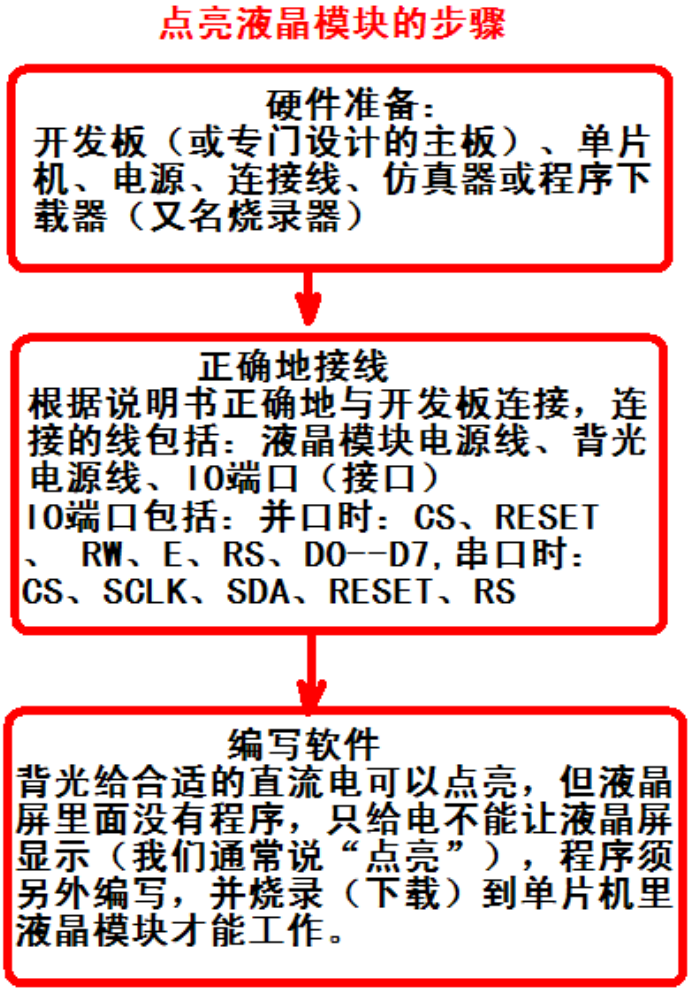
计算方法：

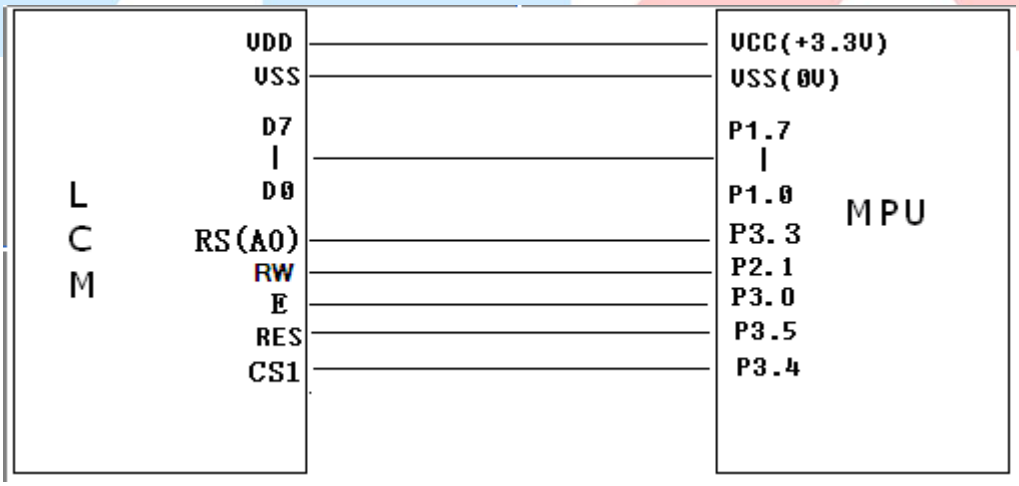
```
BaseAdd=0x2C9D0;
if(MSB >=0xA1 && MSB <= 0Xa9 && LSB >=0xA1)
    Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
    Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;
```

详见字库IC资料（JLX-GB2312-2） 的第28-33页。

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





5.4 程序举例:

```
//=========================================================
//液晶模块型号：JLX280-012-PC,带字库                      |
//驱动 IC:ILI9341，字库 IC:JLX-GB2312                     |
//接口：8 位并行，8080 时序                               |
```

```
//顺时针转 90 度放置                                                    |
//版权所有：深圳市晶联讯电子有限公司，网站：http://www.jlxlcd.cn   |
//=============================================================

#include <reg51.h>
#include <intrins.h>
#include <160120_55.h>   //存储 160*120 像素的 65k COLOR 的一幅彩图
//=========================================================
sbit WRO=P2^1;   //接口定义:WRO 就是 LCD 的 wr
sbit RDO=P3^0;   //接口定义:RDO 就是 LCD 的 rd
sbit DCO=P3^3;   //接口定义:DCO 就是 LCD 的 rs
sbit lcd_cs=P3^4;   //接口定义:lcd_cs 就是 LCD 的 cs
sbit RST=P3^5;      //接口定义:RST 就是 LCD 的 reset
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_CS=P3^6;   //字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#
sbit Rom_SCK=P3^7;  //字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK
sbit key=P2^0;      //P2.0 口与 GND 之间接一个按键
//=========================================================
#define  LCD_DataPort P1   //数据口,8 位模式下只使用高 8 位，DB15~DB8 其实相当于 DB7~DB0
//另外：数据总线：P1 口
//定义彩屏旋转方向
#define normal   0x48
#define CW90     0xE8
#define CCW90    0x28
#define CW180    0x88

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

#define red    0xf800 //定义红色
#define blue   0x001f //定义蓝色
#define green  0x07e0 //定义绿色
#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
//图片数据
uchar code pic1[];
//uchar code pic2[];
//简体汉字字库
char 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,};
char code lian2[]={//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=32x32  —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x80, 0xC0, 0x00, 0x04, 0xC1, 0xE0,
0x7F, 0xFE, 0xE1, 0xC0, 0x3F, 0xFE, 0x71, 0x80, 0x0C, 0x30, 0x73, 0x00, 0x0C, 0x30, 0x73, 0x00,
0x0C, 0x30, 0x66, 0x20, 0x0C, 0x37, 0xFF, 0xF0, 0x0C, 0x37, 0xFF, 0xF8, 0x0F, 0xF0, 0x1C, 0x00,
0x0C, 0x30, 0x1C, 0x00, 0x0C, 0x30, 0x1C, 0x00, 0x0C, 0x30, 0x1C, 0x10, 0x0C, 0x30, 0x1C, 0x38,
0x0F, 0xFF, 0xFF, 0xFC, 0x0F, 0xF0, 0x1C, 0x00, 0x0C, 0x30, 0x1E, 0x00, 0x0C, 0x30, 0x1E, 0x00,
0x0C, 0x36, 0x1A, 0x00, 0x0C, 0x3E, 0x3B, 0x00, 0x0F, 0xF0, 0x33, 0x00, 0x3F, 0xB0, 0x31, 0x80,
0x7E, 0x30, 0x61, 0xC0, 0x30, 0x30, 0xE0, 0xE0, 0x00, 0x31, 0xC0, 0xF0, 0x00, 0x33, 0x80, 0x7C,
0x00, 0x37, 0x00, 0x3C, 0x00, 0x3C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,};
char code xun3[]={//横向取模
/*—  文字:  讯  —*/
/*—  宋体 47;  此字体下对应的点阵为：宽 x 高=63x63   —*/
/*—  宽度不是 8 的倍数，现调整为：宽度 x 高度=64x64  —*/
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, 0x60, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x78, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x3E, 0x00, 0x00, 0x00, 0x00, 0x60, 0x00, 0x00, 0x1F, 0x80, 0x00, 0x00, 0x00, 0xF0, 0x00,
0x00, 0x0F, 0xC3, 0xFF, 0xFF, 0xFF, 0xF8, 0x00, 0x00, 0x0F, 0xC1, 0xFF, 0xFF, 0xFF, 0xF8, 0x00,
0x00, 0x07, 0xE0, 0xE0, 0x00, 0x01, 0xF8, 0x00, 0x00, 0x03, 0xE0, 0x00, 0x00, 0x01, 0xE0, 0x00,
0x00, 0x03, 0xE0, 0x00, 0x80, 0x01, 0xE0, 0x00, 0x00, 0x03, 0xC0, 0x00, 0xE0, 0x01, 0xE0, 0x00,
0x00, 0x01, 0xC0, 0x00, 0xF8, 0x01, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x01, 0xE0, 0x00,
0x00, 0x00, 0x00, 0x00, 0xF8, 0x01, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x01, 0xE0, 0x00,
0x00, 0x00, 0x00, 0x00, 0xF0, 0x01, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x01, 0xE0, 0x00,
0x00, 0x00, 0x00, 0x00, 0xF0, 0x01, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x01, 0xE0, 0x00,
0x00, 0x00, 0x80, 0x00, 0xF0, 0x01, 0xE0, 0x00, 0x00, 0x01, 0xC0, 0x00, 0xF0, 0x01, 0xE0, 0x00,
0x3F, 0xFF, 0xE0, 0x00, 0xF0, 0x01, 0xE0, 0x00, 0x1F, 0xFF, 0xF0, 0x00, 0xF0, 0x01, 0xE0, 0x00,
0x0E, 0x03, 0xF0, 0x00, 0xF0, 0x31, 0xE0, 0x00, 0x00, 0x03, 0xE0, 0x00, 0xF0, 0x79, 0xE0, 0x00,
0x00, 0x03, 0xC0, 0x00, 0xF0, 0xFD, 0xE0, 0x00, 0x00, 0x03, 0xC7, 0xFF, 0xFF, 0xFF, 0xE0, 0x00,
0x00, 0x03, 0xC3, 0xFF, 0xFF, 0xFF, 0xE0, 0x00, 0x00, 0x03, 0xC0, 0x00, 0xF0, 0x01, 0xE0, 0x00,
0x00, 0x03, 0xC0, 0x00, 0xF0, 0x01, 0xE0, 0x00, 0x00, 0x03, 0xC0, 0x00, 0xF0, 0x01, 0xE0, 0x00,
0x00, 0x03, 0xC0, 0x00, 0xF0, 0x01, 0xE0, 0x00, 0x00, 0x03, 0xC0, 0x00, 0xF0, 0x01, 0xF0, 0x00,
0x00, 0x03, 0xC0, 0x00, 0xF0, 0x00, 0xF0, 0x00, 0x00, 0x03, 0xC0, 0x00, 0xF0, 0x00, 0xF0, 0x00,
0x00, 0x03, 0xC0, 0x00, 0xF0, 0x00, 0xF0, 0x00, 0x00, 0x03, 0xC0, 0x00, 0xF0, 0x00, 0xF0, 0x00,
0x00, 0x03, 0xC0, 0x00, 0xF0, 0x00, 0xF0, 0x20, 0x00, 0x03, 0xC0, 0x60, 0xF0, 0x00, 0xF0, 0x30,
0x00, 0x03, 0xC0, 0xE0, 0xF0, 0x00, 0xF0, 0x30, 0x00, 0x03, 0xC1, 0xC0, 0xF0, 0x00, 0xF8, 0x70,
0x00, 0x03, 0xC3, 0x80, 0xF0, 0x00, 0x78, 0x60, 0x00, 0x03, 0xC7, 0x00, 0xF0, 0x00, 0x78, 0x60,
0x00, 0x03, 0xCF, 0x00, 0xF0, 0x00, 0x7C, 0x60, 0x00, 0x03, 0xDE, 0x00, 0xF0, 0x00, 0x7C, 0x60,
0x00, 0x03, 0xFC, 0x00, 0xF0, 0x00, 0x3E, 0xE0, 0x00, 0x03, 0xFC, 0x00, 0xF0, 0x00, 0x3E, 0xE0,
0x00, 0x03, 0xF8, 0x00, 0xF0, 0x00, 0x1F, 0xE0, 0x00, 0x07, 0xF0, 0x00, 0xF0, 0x00, 0x1F, 0xE0,
0x00, 0x07, 0xE0, 0x00, 0xF0, 0x00, 0x0F, 0xE0, 0x00, 0x03, 0xE0, 0x00, 0xF0, 0x00, 0x0F, 0xE0,
```

```
0x00,0x01,0xC0,0x00,0xF0,0x00,0x07,0xF0,0x00,0x01,0xC0,0x00,0xF0,0x00,0x03,0xF0,
0x00,0x00,0x80,0x00,0xF0,0x00,0x01,0xF0,0x00,0x00,0x00,0x00,0xE0,0x00,0x00,0x70,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x10,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,};
//延时 1
void delay(long tt)
{
    uint i,j;
    for(i=0;i<tt;i++)
    {
        for(j=0;j<293;j++);
    }
}
//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<2;k++);
}
//传送 8 位的数据
void data_out(uchar data1)
{
    //8080 8bit interface
    lcd_cs = 0;
    DC0 = 1;        //RS=1:表示以下传的是“数据”
    RD0 = 1;
    P1=data1;
    WR0 = 0;
    WR0 = 1;
    lcd_cs = 1;    //传完后将片选置高，减少受其它信号的干扰
}
//传送 8 位的命令
void comm_out(uchar com)
{
    //8080 8bit interface
    DC0 = 0;
    lcd_cs = 0;
    RD0 = 1;        //RS=1:表示以下传的是“指令”
    P1 = com;
//  delay_us(1 );
    WR0 = 0;
    WR0 = 1;
    lcd_cs = 1;  //传完后将片选置高，减少受其它信号的干扰
}
```

```
//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);  //先传高 8 位
      data_out(data_16bit );     //再传低 8 位
}
//===传 16 位指令，16 位指令一起赋值
void comm_out_16(uint data_16bit)
{
      comm_out(data_16bit >>8);   //先传高 8 位
      comm_out(data_16bit );      //再传低 8 位
}
//===发送 1 个字节的指令及 1 个字节的数据==============
void Lcd_Write_Com_Data(uint com,uint val)
{
      comm_out_16(com);      //先传指令
    data_out_16(val);        //再传数据
}
void waitkey()
{
repeat:
      if(key==1) goto repeat;
      else delay(500);

}
//LCD 初始化
void LCD_initial()
{
      delay(50);
      RST=0;                 //低电平：复位
      delay(50);
      RST=1;                 //高电平：复位结束
      delay(50);
      //开始初始化：
      comm_out(0xCF);        //
      data_out(0x00);
      data_out(0xD9);
      data_out(0X30);
      comm_out(0xED);        //
      data_out(0x64);
      data_out(0x03);
      data_out(0X12);
      data_out(0X81);
      comm_out(0xE8);
      data_out(0x85);
      data_out(0x00);
      data_out(0x78);
```

```
comm_out(0xCB);
data_out(0x39);
data_out(0x2C);
data_out(0x00);
data_out(0x34);
data_out(0x02);
comm_out(0xF7);
data_out(0x20);
comm_out(0xEA);
data_out(0x00);
data_out(0x00);
comm_out(0xC0);        //Power control
data_out(0x1B);        //VRH[5:0]
comm_out(0xC1);        //Power control
data_out(0x12);        //SAP[2:0];BT[3:0]
comm_out(0xC5);        //VCM control
data_out(0x32);
data_out(0x3C);
comm_out(0xC7);        //VCM control2
data_out(0X9D);
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;
comm_out(0x3A);
data_out(0x55);
comm_out(0xB1);
data_out(0x00);
data_out(0x1B);
comm_out(0xB6);        //显示功能设置：列/行 显示顺序
data_out(0x0A);      //0A
data_out(0x82);        //改变 SOURCE 线的方向：0xa2：左到右，0x82：右到左
comm_out(0xF6);
data_out(0x01);
data_out(0x30);      //30
comm_out(0xF2);        // 3Gamma Function Disable
data_out(0x00);
comm_out(0x26);        //Gamma curve selected
data_out(0x01);
comm_out(0xE0);        //设置 GAMMA 值，由玻璃及 IC 原厂设置
data_out(0x0F);
data_out(0x24);
data_out(0x1F);
data_out(0x0B);
data_out(0x0F);
data_out(0x05);
```

```
    data_out(0x4A);
    data_out(0X96);
    data_out(0x39);
    data_out(0x07);
    data_out(0x11);
    data_out(0x03);
    data_out(0x11);
    data_out(0x0D);
    data_out(0x04);
    comm_out(0XE1);    //设置 GAMMA 值，由玻璃及 IC 原厂设置
    data_out(0x00);
    data_out(0x1B);
    data_out(0x20);
    data_out(0x04);
    data_out(0x10);
    data_out(0x02);
    data_out(0x35);
    data_out(0x23);
    data_out(0x46);
    data_out(0x04);
    data_out(0x0E);
    data_out(0x0C);
    data_out(0x2E);
    data_out(0x32);
    data_out(0x05);
    comm_out(0x11);    //退出睡眠
    delay(50);    //这个很重要
    comm_out(0x29);    //打开显示：display ON.
}
//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(uint mono_data,uint font_color,uint back_color)
{
    uint 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;
    }
}
///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
}
//显示 160x120 点阵的彩色图像
void disp_160x120(uint y,uint x,char *dp)
{
      int i,j;
      for(j=0;j<120;j++)
      {
            lcd_address(x,y+j,160,120);
            for(i=0;i<160;i++)
            {
                  data_out(*dp);
                  dp++;
                  data_out(*dp);          //这一句相当于写了一个像素点的数据，彩色 1 个的 16 位的数据。
                  dp++;
            }
      }
}
//=============================================================

//显示全屏单一色彩
void display_color(int color)
{
      int i,j;
      lcd_address(0,0,320,240);
      for(i=0;i<240;i++)
      {
            for(j=0;j<320;j++)
            {
                  data_out_16(color);
            }
      }
}
//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)
```

```
{
    uint i,j;
    for(j=0;j<16;j++)
    {
        lcd_address(x,y+j,16,16);
        for(i=0;i<2;i++)
        {
            mono_data_out(*dp,font_color,back_color);    //这一句相当于写了一行 8 个像素点的数据。
            dp++;
        }
    }
}
//显示 32x32 点阵的单色的字符或图像
void disp_32x32(int x,int y,char *dp,int font_color,int back_color)
{
    uint i,j;
    for(j=0;j<32;j++)
    {
        lcd_address(x,y+j,32,32);
        for(i=0;i<4;i++)
        {
            mono_data_out(*dp,font_color,back_color);    //这一句相当于写了一行 8 个像素点的数据。
            dp++;
        }
    }
}
//显示 64x64 点阵的单色的字符或图像
void disp_64x64(uint x,uint y,char *dp,uint font_color,uint back_color)
{
    uint i,j;
    for(j=0;j<64;j++)
    {
        lcd_address(x,y+j,64,64);
        for(i=0;i<8;i++)
        {
            mono_data_out(*dp,font_color,back_color);    //这一句相当于写了一行 8 个像素点的数据。
            dp++;
        }
    }
}
/****送指令到晶联讯字库 IC***/
void send_command_to_ROM( uint datu )
{
    uint 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( )
{
    uint i;
    uint ret_data=0;
    Rom_SCK=1;
    for(i=0;i<8;i++)
    {
        Rom_OUT=1;
        Rom_SCK=0;
        ret_data=ret_data<<1;
        if( Rom_OUT )
            ret_data=ret_data+1;
        else
            ret_data=ret_data+0;
        Rom_SCK=1;
        //delay_us(1);
    }
    return(ret_data);
}
//从指定地址读出 32x32 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_32x32(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);       //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
    for(j=0;j<32;j++)
    {
        lcd_address(y,x+j,32,32);
        for(i=0;i<4;i++)
        {
            disp_data=get_data_from_ROM();
            mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
        }
```

```
    }
    Rom_CS=1;
}
//从指定地址读出 16x32 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_16x32(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);      //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);         //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                //地址的低 8 位,共 24 位
    for(j=0;j<32;j++)
    {
        lcd_address(y,x+j,32,16);
        for(i=0; i<2; i++ )
        {
            disp_data=get_data_from_ROM();
            mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
        }
    }
    Rom_CS=1;
}
//从指定地址读出 24x24 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_24x24(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);      //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);         //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                //地址的低 8 位,共 24 位
    for(j=0;j<24;j++)
    {
        lcd_address(y,x+j,24,24);
        for(i=0; i<3; i++ )
        {
            disp_data=get_data_from_ROM();
            mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
        }
    }
    Rom_CS=1;
}
//从指定地址读出 12x24 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_12x24(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
```

```
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);      //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);         //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                //地址的低 8 位,共 24 位
    for(j=0;j<24;j++)
    {
        lcd_address(y,x+j,24,12);
        for(i=0; i<2; i++ )
       {
            disp_data=get_data_from_ROM();
            mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
        }
    }
    Rom_CS=1;
}
//从指定地址读出 16x16 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_16x16(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);      //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);         //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                //地址的低 8 位,共 24 位
    for(j=0;j<16;j++)
    {
        lcd_address(y,x+j,16,16);
        for(i=0; i<2; i++ )
       {
            disp_data=get_data_from_ROM();
            mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
        }
    }
    Rom_CS=1;
}
//从指定地址读出 8x16 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_8x16(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);      //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);         //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                //地址的低 8 位,共 24 位
    for(j=0;j<16;j++)
    {
```

```
            lcd_address(y,x+j,16,8);
            for(i=0; i<1; i++ )
           {
                 disp_data=get_data_from_ROM();
                 mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
            }
      }
      Rom_CS=1;
}
//从指定地址读出 12x12 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_12x12(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
      uint i,j,disp_data;
      Rom_CS = 0;
      send_command_to_ROM(0x03);
      send_command_to_ROM((fontaddr&0xff0000)>>16);       //地址的高 8 位,共 24 位
      send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
      send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
      for(j=0;j<12;j++)
      {
            lcd_address(y,x+j,12,12);
            for(i=0; i<2; i++ )
           {
                 disp_data=get_data_from_ROM();
                 mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
            }
      }
      Rom_CS=1;
}
//从指定地址读出 6x12 点阵字符的数据写到液晶屏指定（y,x)座标中
void get_and_write_6x12(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
      uint i,j,disp_data;
      Rom_CS = 0;
      send_command_to_ROM(0x03);
      send_command_to_ROM((fontaddr&0xff0000)>>16);       //地址的高 8 位,共 24 位
      send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
      send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
      for(j=0;j<12;j++)
      {
            lcd_address(y,x+j,12,6);
            for(i=0; i<1; i++ )
           {
                 disp_data=get_data_from_ROM();
                 mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
            }
      }
```

```
    Rom_CS=1;
}
//从指定地址读出数据写到液晶屏指定（y,x)座标中
void get_and_write_5x8(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint j,disp_data;
    Rom_CS = 0;
    //Rom_SCK=0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);      //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);         //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                //地址的低 8 位,共 24 位
    for(j=0;j<8;j++)
    {
        lcd_address(y,x+j,8,5);
        disp_data=get_data_from_ROM();
        mono_data_out(disp_data,font_color,back_color);   //这一句相当于写了一行 8 个像素点的数据。
    }
    Rom_CS=1;
}
/***********/
ulong  fontaddr;
//显示一串 32x32 点阵的汉字或 16x32 点阵的 ASCII 码
void disp_GB2312_32x32_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
{
    uint i= 0;
    while((text[i]>0x00))
    {
        if( ((text[i]>=0xb0) && (text[i]<=0xf7) ) && (text[i+1]>=0xa1) )
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0Xedf00);
            get_and_write_32x32(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=32;
        }
        else if(( (text[i]>=0xa1) && (text[i]<=0xa9) ) && (text[i+1]>=0xa1) )
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0Xedf00);
            get_and_write_32x32(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=32;
```

```
            }
            else if( (text[i]>=0x20) && (text[i]<=0x7e) )
            {
                  fontaddr = (text[i]- 0x20);
                  fontaddr = (ulong)(fontaddr*64);
                  fontaddr = (ulong)(fontaddr+0x1e5a50);
                  get_and_write_16x32(fontaddr,y,x,font_color,back_color);
                  i+=1;
                  x+=16;
            }
            else
                  i++;
      }
}
//显示一串 24x24 点阵的汉字或 12x24 点阵的 ASCII 码
void disp_GB2312_24x24_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
{
      uint i= 0;
      while((text[i]>0x00))
      {
            if( ((text[i]>=0xb0) && (text[i]<=0xf7) ) && (text[i+1]>=0xa1) )
            {
                  fontaddr = (text[i]- 0xb0)*94;
                  fontaddr += (text[i+1]-0xa1)+846;
                  fontaddr = (ulong)(fontaddr*72);
                  fontaddr = (ulong)(fontaddr+0X068190);
                  get_and_write_24x24(fontaddr,y,x,font_color,back_color);
                  i+=2;
                  x+=24;
            }
            else if(( (text[i]>=0xa1) && (text[i]<=0xa9) ) && (text[i+1]>=0xa1) )
            {
                  fontaddr = (text[i]- 0xa1)*94;
                  fontaddr += (text[i+1]-0xa1);
                  fontaddr = (ulong)(fontaddr*72);
                  fontaddr = (ulong)(fontaddr+0X068190);
                  get_and_write_24x24(fontaddr,y,x,font_color,back_color);
                  i+=2;
                  x+=24;
            }

            else if( (text[i]>=0x20) && (text[i]<=0x7e) )
            {
                  fontaddr = (text[i]- 0x20);
                  fontaddr = (ulong)(fontaddr*48);
                  fontaddr = (ulong)(fontaddr+0x1dff00);
                  get_and_write_12x24(fontaddr,y,x,font_color,back_color);
```

```
                i+=1;
                x+=12;
            }

            else
                i++;
        }
}
//=====从字库读数据，显示 16*16 点阵的汉字或 8*16 点阵的数字========
void disp_GB2312_16x16_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
{
    uint i= 0;
    while((text[i]>0x00))
    {

        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);

            get_and_write_16x16(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=16;
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);

            get_and_write_16x16(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=16;
        }
        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (ulong)(fontaddr*16);
            fontaddr = (ulong)(fontaddr+0x1dd780);

            get_and_write_8x16(fontaddr,y,x,font_color,back_color);
            i+=1;
            x+=8;
        }
```

```
            else
                  i++;
      }
}
//=====从字库读数据，显示 12*12 点阵的汉字或 6*12 点阵的数字
void disp_GB2312_12x12_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
{
      uint i= 0;
      while((text[i]>0x00))
      {

            if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
            {
                  fontaddr = (text[i]- 0xb0)*94;
                  fontaddr += (text[i+1]-0xa1)+846;
                  fontaddr = (ulong)(fontaddr*24);
                  fontaddr = (ulong)(fontaddr+0x00);

                  get_and_write_12x12(fontaddr,y,x,font_color,back_color);
                  i+=2;
                  x+=12;
            }
            else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
            {
                  fontaddr = (text[i]- 0xa1)*94;
                  fontaddr += (text[i+1]-0xa1);
                  fontaddr = (ulong)(fontaddr*24);
                  fontaddr = (ulong)(fontaddr+0x00);

                  get_and_write_12x12(fontaddr,y,x,font_color,back_color);
                  i+=2;
                  x+=12;
            }
            else if((text[i]>=0x20) &&(text[i]<=0x7e))
            {
                  fontaddr = (text[i]- 0x20);
                  fontaddr = (ulong)(fontaddr*12);
                  fontaddr = (ulong)(fontaddr+0x1dbe00);

                  get_and_write_6x12(fontaddr,y,x,font_color,back_color);
                  i+=1;
                  x+=6;
            }
            else
                  i++;
      }
```

```
}
//=====从字库读数据，显示 5*8 点阵的字
void disp_GB2312_5x8_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
{
    uint i= 0;
    while((text[i]>0x00))
    {
        if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (ulong)(fontaddr*8);
            fontaddr = (ulong)(fontaddr+0x1ddf80);

            get_and_write_5x8(fontaddr,y,x,font_color,back_color);
            i+=1;
            x+=6;
        }
        else
            i++;
    }

}

//主程序
void main(void)
{
    LCD_initial();
    while(1)
    {
       display_color(blue);

        disp_160x120(0,0,pic2);              //在（y，x）位置显示一幅 160x120 像素的图片
        disp_160x120(120,0,pic2);        //在（y，x）位置显示一幅 160x120 像素的图片
        disp_160x120(0,160,pic2);        //在（y，x）位置显示一幅 160x120 像素的图片
        disp_160x120(120,160,pic2);      //在（y，x）位置显示一幅 160x120 像素的图片

      waitkey();
        display_color(blue);        //显示全屏蓝色
      disp_GB2312_32x32_string(  0,0," 晶联讯 2.8 寸 TFT 彩屏 ",red,yellow);  //显示 32x32 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）
        disp_GB2312_32x32_string( 32,0,"①32*32 点阵大汉字库",white,blue); //显示 32x32 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）
        disp_GB2312_32x32_string( 64,0,"  {[(<!#$%a01^&*>)]}",white,blue);//显示 32x32 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）

        disp_GB2312_24x24_string(96,0,"②24*24 点阵中字国标汉字库",white,blue);        //显示 24x24 点阵的字符串，括号内的参数分别是(y，x，数据指针，字体色，背景色）
```

```
        disp_GB2312_24x24_string(120,0,"  {[(<!#$%ABCabc012^&*>)]}",white,blue);//显示 24x24 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）

        disp_GB2312_16x16_string(148,0,"③16*16 点阵小字国标汉字库 GB2312",white,blue);//显示 16x16 点阵的字符串，括号内的参数分别是(y，x，数据指针，字体色，背景色）
        disp_GB2312_16x16_string(164,0,"  {[(<!#$%abcdefghiABCDEFGHI01234^&*>)]}",white,blue);

        disp_GB2312_12x12_string(184,0,"④12*12 点阵微小字国标汉字库 GB2312",white,blue);//显示 12x12 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）
        disp_GB2312_12x12_string(200,0,"  {[(<!#$%abcdefghijklmnopqABCDEFGHIJKLMN01234^&*>)]}",white,blue);


        disp_GB2312_5x8_string  (220,0,"(5)5X8 dots format ASCII characters:{[(<!@#$-%^&*>)]}",white,blue); //显示 5x8 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）

        waitkey();
        display_color(blue);        //显示全屏蓝色

        disp_GB2312_24x24_string  (24*1,36,"  ((`'_\"``\"\"_'`))",white,blue);
        disp_GB2312_24x24_string  (24*2,36,"    )  -   -  (    ",white,blue);
        disp_GB2312_24x24_string  (24*3,36,"   /  (o _ o)  \\",white,blue);
        disp_GB2312_24x24_string  (24*4,36,"   \\    (o)    /" ,white,blue);
        disp_GB2312_24x24_string  (24*5,36,"   _'-._'='_.-'_ ",white,blue);
        disp_GB2312_24x24_string  (24*6,36," /`;#'#'#.-.#'#'#;`\\",white,blue);
        disp_GB2312_24x24_string  (24*7,36," \\_))   '#   ((_/" ,white,blue);
        disp_GB2312_24x24_string  (24*8,36,"   #          #   ",white,blue);
        waitkey();

        display_color(blue);        //显示全屏蓝色
      disp_GB2312_32x32_string(32*0,1, "我公司产品已广泛应用",white,blue);
      disp_GB2312_32x32_string(32*1,1, "于公共查询、监控、地",white,blue);
      disp_GB2312_32x32_string(32*2,1, "质、冶金、石油、化工",white,blue);                    //显示 16x16 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）
        disp_GB2312_32x32_string(32*3,0, "交通、能源、工业自动",white,blue);
        disp_GB2312_32x32_string(32*4,0, "化、环保、医疗、电子",white,blue);
        disp_GB2312_32x32_string(32*5,0, "电信、煤矿、航海、体",white,blue);
      disp_GB2312_32x32_string(32*6,0, "育、教育、科研、保健",white,blue);
      disp_GB2312_16x16_string(16*14,0,"                              ",white,blue);
      waitkey();
        display_color(blue);        //显示全屏蓝色
      disp_GB2312_12x12_string(12*0,0," 多年来，深圳市晶联讯电子有限公司凭借稳定、可靠的产品",white,blue);                    // 显 示 12x12 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）
      disp_GB2312_12x12_string(12*1,0,"                                                  ",white,blue);
      disp_GB2312_12x12_string(12*2,0,"质量，优异的性价比，完善的售后服务，在广大用户中树立 ",white,blue);
      disp_GB2312_12x12_string(12*3,0,"                                                  ",white,blue);
      disp_GB2312_12x12_string(12*4,0,"了良好的品牌形象。能够为客户量身订做的产品设计理念，  ",white,blue);
      disp_GB2312_12x12_string(12*5,0,"                                                  ",white,blue);
```

```
        disp_GB2312_12x12_string(12*6,0,"为广大用户提供了更加贴身的服务。产品的高可靠性和高稳 ",white,blue);
        disp_GB2312_12x12_string(12*7,0,"                                                    ",white,blue);
        disp_GB2312_12x12_string(12*8,0,"定性让每一位用户都倍感放心。                       ",white,blue);
        disp_GB2312_12x12_string(12*9,0,"                                                    ",white,blue);
        disp_GB2312_12x12_string(12*10,0,"经营宗旨：制造高品质产品及提供良好服务。          ",white,blue);
        disp_GB2312_12x12_string(12*11,0,"                                                   ",white,blue);
        disp_GB2312_12x12_string(12*12,0,"质量方针:客户至上，质量第一，持续改进，服务到位。 ",white,blue);
        disp_GB2312_12x12_string(12*13,0,"                                                   ",white,blue);
        disp_GB2312_12x12_string(12*14,0,"经营目标：做最好的模组公司，做客户信得过企业。    ",white,blue);
        disp_GB2312_12x12_string(12*15,0,"                                                   ",white,blue);
        disp_GB2312_12x12_string(12*16,0,"深圳市晶联讯电子有限公司热情欢迎新老客户垂询！    ",white,blue);
          waitkey();
          display_color(0xf800);
          waitkey();
          display_color(0x07e0);
          waitkey();
          display_color(0x0000);
          disp_GB2312_32x32_string(0,36," 自编汉字的显示",white,blue); //显示 32x32 点阵的字符串，括号内的参数分别是（y，x，数据指针，字体色，背景色）
          disp_16x16(96,96,jing1,white,blue);                    //在（y=0，x=0）位置显示一个 16x16 像素的汉字（实际是当成一幅 16x16 点的图片）
          disp_32x32(96,112,lian2,white,blue);                   //在（y，x）位置显示一个 32x32 像素的汉字（实际是当成一幅 32x32 点的图片）
          disp_64x64(96,144,xun3,white,blue);                    //在（y，x）位置显示一个 64x64 像素的汉字（实际是当成一幅 64x64 点的图片）
          waitkey();
      }
}
```