

JLX12864G-08602-PC

带字库 IC 的编程说明书

目 录

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

1. 概述

JLX12864G-08602-PC 型液晶显示模块既可以当成普通的图像型液晶显示模块使用(即显示普通图像型的单色图片功能), 又含有 JLX-GB2312 字库 IC, 可以从字库 IC 中读出内置的字库的点阵数据写入到 LCD 驱动 IC 中, 以达到显示汉字的目的。

此字库 IC 存储内容如下表所述:

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



2. 字型样张:

15X16 点 GB2312 汉字

啊阿埃挨哎唉哀皑癌蔼矮艾
碍爱隘鞍氨安俺按暗岸胺案
肮昂盎凹敖熬翱袄傲奥懊澳
芭捌扒叭吧芭八疤巴拔跋靶
把耙坝霸罢爸白柏百摆佰败
拜裨斑班搬扳般颁板版扮拌

8x16 点国标扩展字符

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

5x7 点 ASCII 字符

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

7x8 点 ASCII 字符

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

8x16 点 ASCII 字符

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

8x16 点 ASCII 粗体字符

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

16 点阵不等宽 ASCII 方头

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

16 点阵不等宽 ASCII 白正

!"#\$%&'()*+,-./0123456789
:;<=>?@ABCDEFGHIJKLM
NOPQRSTUVWXYZ[\]^_`a
bcdefghijklmnopqrstuvwxyz{}

3. 外形尺寸及接口引脚功能

3.1 外形图:

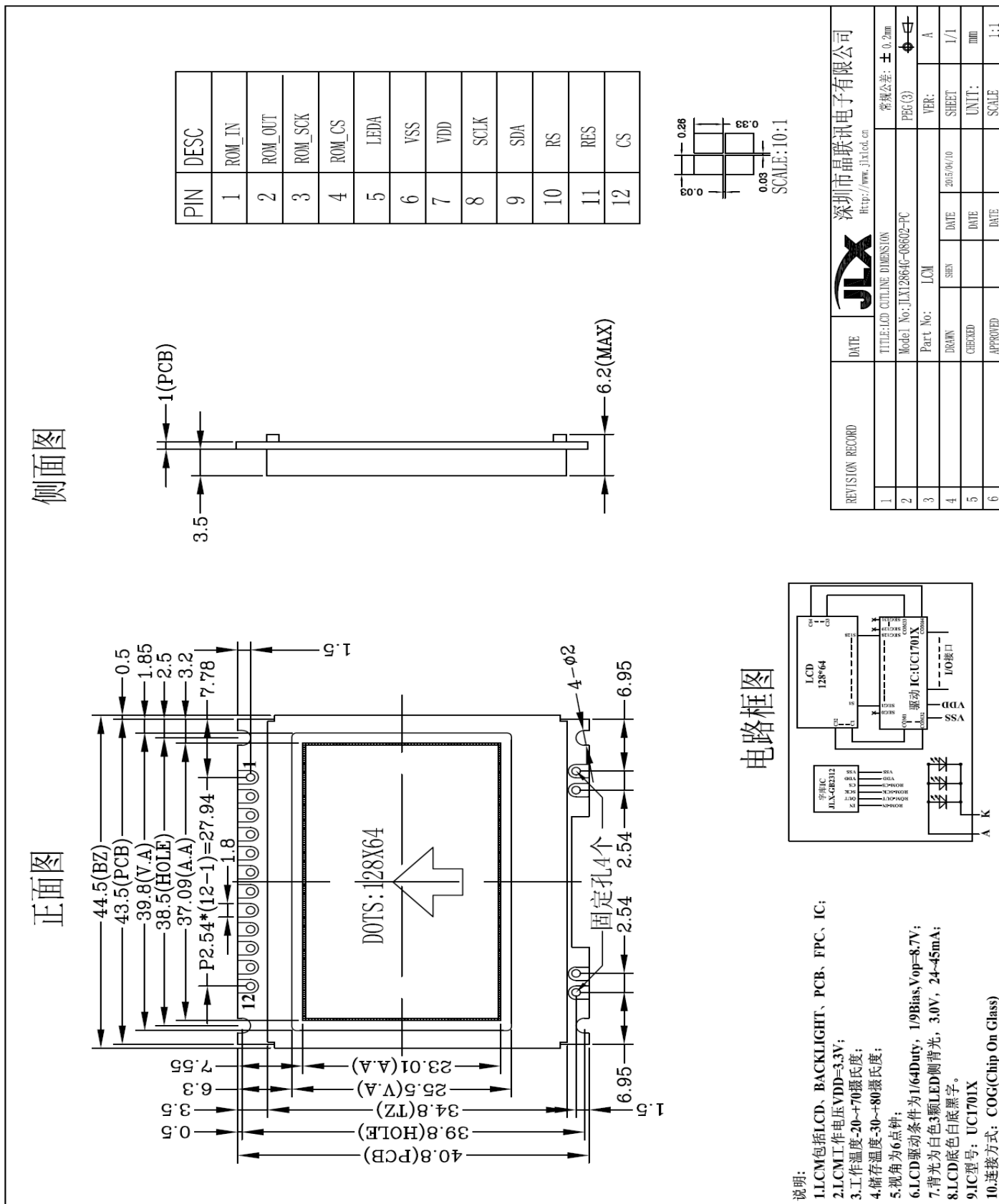


图 1. 外形尺寸

3.2 模块的接口引脚功能

3.2.1 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	LEDA	背光电源	背光电源正极，同 VDD 电压（5V 或 3.3V）	
6	VSS	接地	0V	
7	VDD	电路电源	5V, 或 3.3V 可选	
8	SCLK	I/O	串行时钟	
9	SDA	I/O	串行数据	
10	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” ）	
11	RESET	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	

表 2: 模块串行接口引脚功能

4. 工作电路框图:

见图 2, 模块由 LCD 驱动 IC UC1701X、字库 IC、背光组成。
电路框图

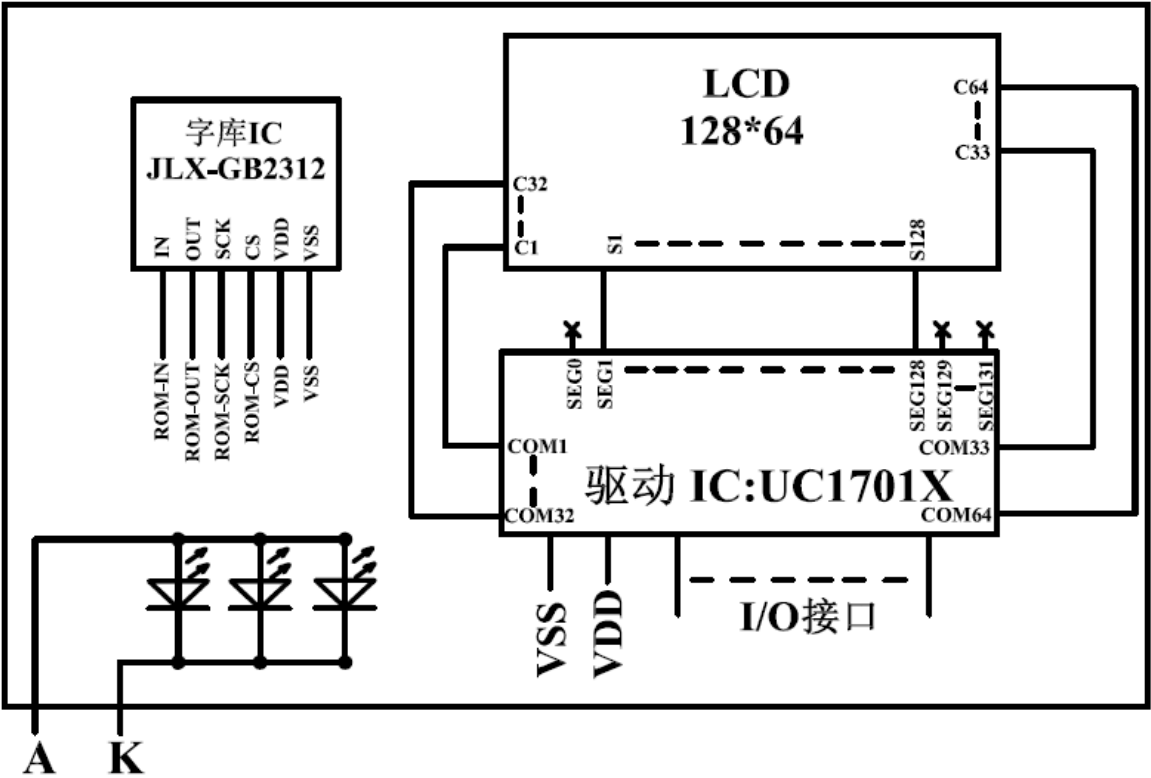


图 2: 电路框图

5. 指令:

5.1 字库 IC (JLX-GB2312) 指令表

Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
READ	Read Data Bytes	0000 0011	03 h	3	-	1 to ∞
FAST_READ	Read Data Bytes at Higher Speed	0000 1011	0B h	3	1	1 to ∞

所有对本芯片的操作只有 2 个, 那就是 Read Data Bytes (READ "一般读取") 和 Read Data Bytes at Higher Speed (FAST_READ "快速读取点阵数据")。

Read Data Bytes (一般读取):

Read Data Bytes 需要用指令码来执行每一次操作。READ 指令的时序如下(图):

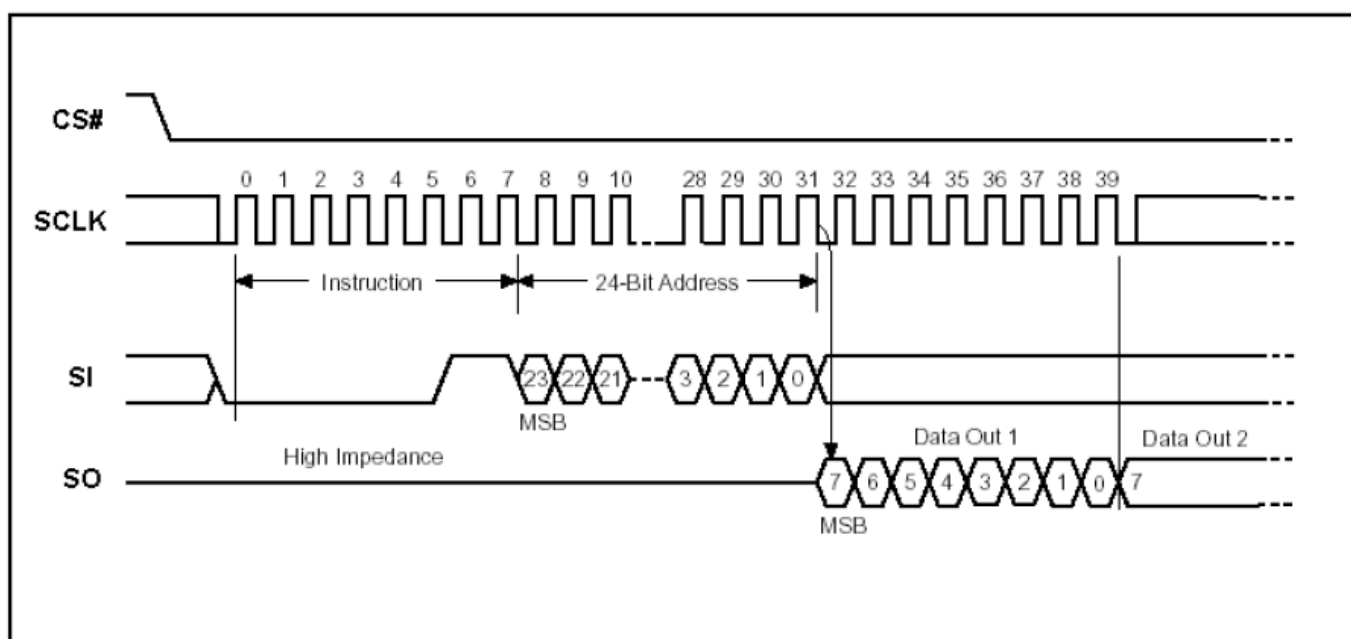
■ 首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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

■ 读取字节数据后, 则把片选信号 (CS#) 变为高, 结束本次操作。

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

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



Read Data Bytes at Higher speed (快速读取):

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

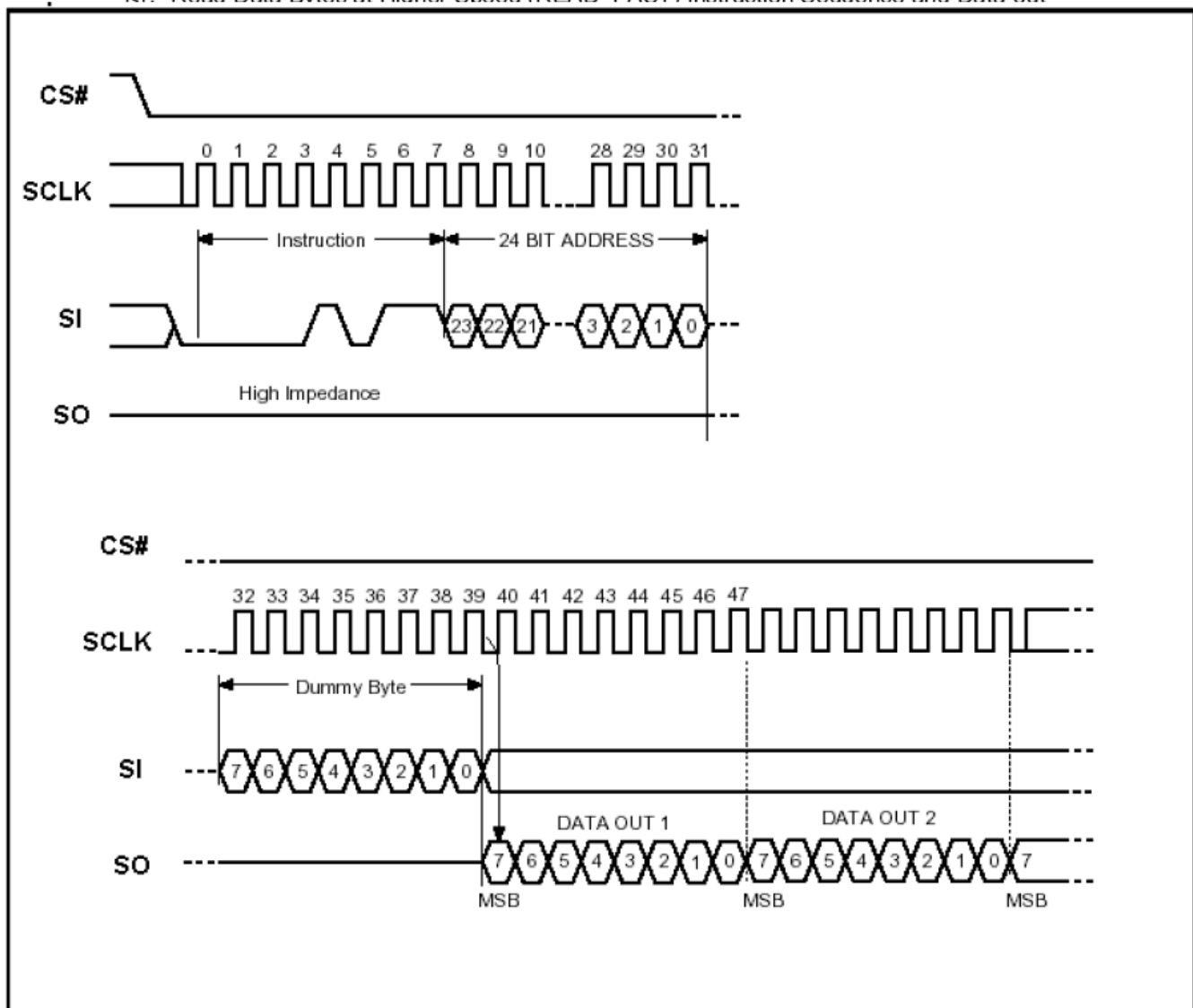
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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

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

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

图: Read Data Bytes at Higher Speed (READ FAST) Instruction Sequence and Data-out



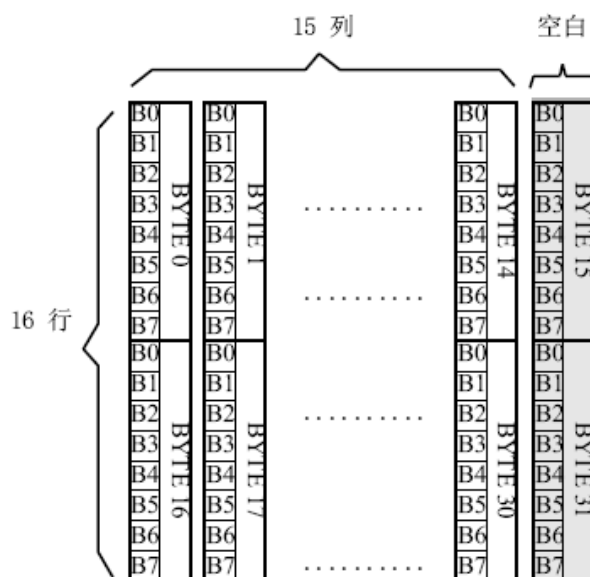
6 字库调用方法

6.1 汉字点阵排列格式

每个汉字在芯片中是以汉字点阵字模的形式存储的，每个点用一个二进制位表示，存 1 的点，当显示时可以在屏幕上显示亮点，存 0 的点，则在屏幕上不显示。点阵排列格式为竖置横排：即一个字节的低位表示下面的点，高位表示上面的点（如果用户按 16bit 总线宽度读取点阵数据，请注意高低字节的序），排满一行后再排下一行。这样把点阵信息用来直接在显示器上按上述规则显示，则将出现对应的汉字。

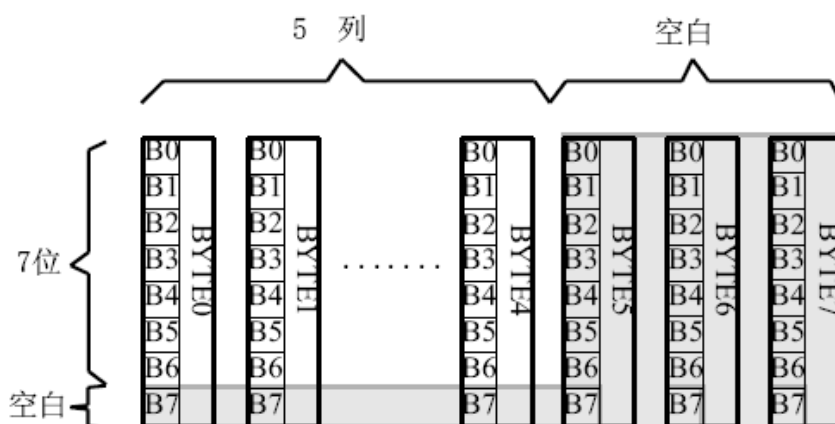
6.1.1 15X16 点汉字排列格式

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



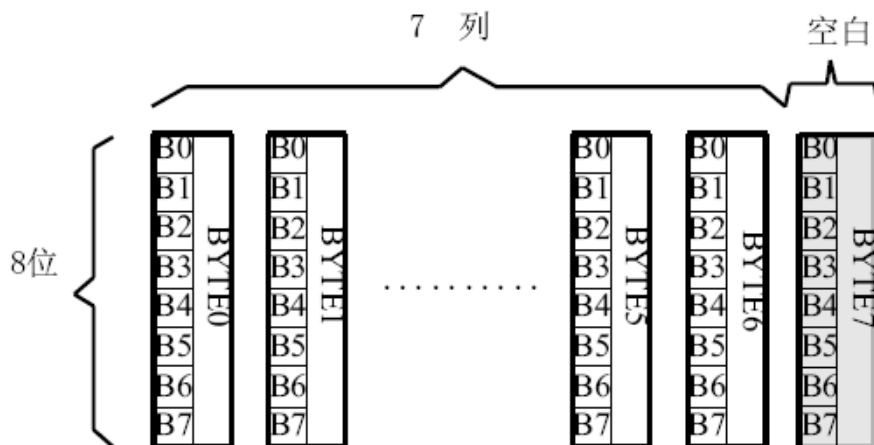
6.1.2 5X7 点 ASCII 字符排列格式

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



6.1.3 7X8 点 ASCII 字符排列格式

7X8 点 ASCII 的信息需要 8 个字节 (BYTE 0 - BYTE7) 来表示。该 ASCII 点阵数据是竖置横排的, 其具体排列结构如下图:



6.1.4 8X16 点字符排列格式

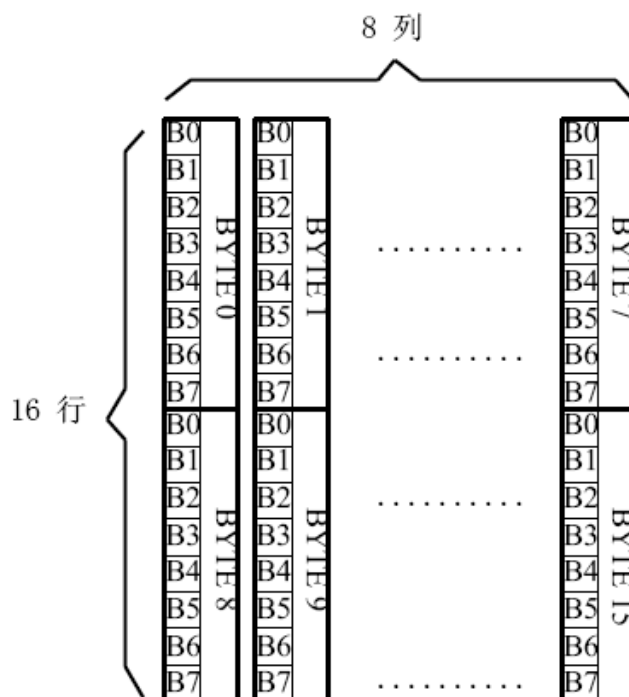
适用于此种排列格式的字体有:

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 - BYTE15) 来表示。该点阵数据是竖置横排的, 其具体排列结构如下图:



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

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

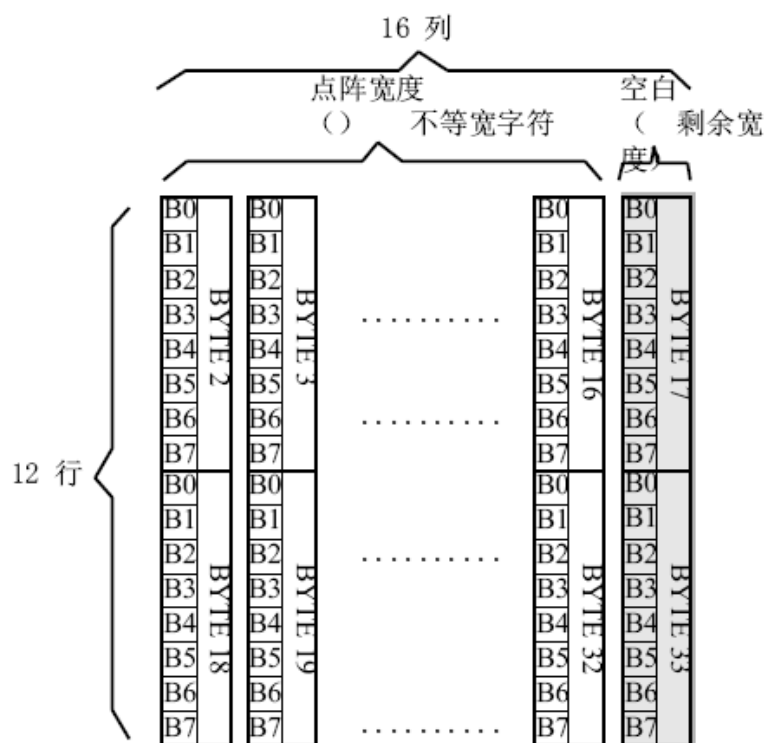
■ 存储格式

由于字符是不等宽的, 因此在存储格式中 BYTE0~ BYTE1 存放点阵宽度数据, BYTE2-33 存放竖置横排点阵数据。具体格式见下图:



■ 存储结构

不等宽字符的点阵存储宽度是以 BYTE 为单位取整的, 根据不同字符宽度会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的实际宽度数据, 可以对还原下一个字的显示或排版留作参考。



例如: ASCII

方头字符

B

0-33BYTE 的点阵数据是: 00 0C 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 7F 7F 63 63 63 63 67 3E 1C 00 00 00 00 00

其中:

BYTE0~ BYTE1: 00 0C 为 ASCII 方头字符 B 的点阵宽度数据, 即: 12 位宽度。字符后面有 4 位空白区, 可以在排版下一个字时考虑到这一点, 将下一个字的起始位置前移。

BYTE2-33: 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 63 63 63 67 3E 1C 00 00 00 00 00 为 ASCII 方头字符 B 的点阵数据。

6.2 汉字点阵字库地址表

	字库内容	编码体系	码位范围	字符数	起始地址	结束地址	参 考 算 法
1	15X16 点 GB2312 标准点阵字库	GB2312	A1A1-F7 FE	6763+376	00000	3B7BF	6.3.1.1
2	7X8 点 ASCII 字符	ASCII	20~7F 96		66C0	69BF	6.3.2.2
3	8X16 点国标扩展字符	GB2312	AAA1-A BC0	126	3B7D0	3BFBF	6.3.1.2
4	8X16 点 ASCII 字符	ASCII	20~7F	96	3B7C0	3BFBF	6.3.2.3
5	5X7 点 ASCII 字符 ASCII		20~7F	96	3BFC0	3C2BF	6.3.2.1
6	16 点阵不等宽 ASCII 方头 (Arial) 字符	ASCII	20~7F	96	3C2C0	3CF7F	6.3.2.4
7	8X16 点 ASCII 粗体字符 ASCII		20~7F	96	3CF80	3D57F	6.3.2.5
8	16 点阵不等宽 ASCII 白正 (TimesNewRoman) 字符	ASCII	20~7F	96	3D580	3E23F	6.3.2.6

6.3 字符在芯片中的地址计算方法

用户只要知道字符的内码, 就可以计算出该字符点阵在芯片中的地址, 然后就可从该地址连续读出点阵信息用于显示。

6.3.1 汉字字符的地址计算

6.3.1.1 15X16 点 GB2312 标准点阵字库

参数说明:

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd: 说明点阵数据在字库芯片中的起始地址。

计算方法:

BaseAdd=0;

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

Address = (282 + (LSB - 0xA1))*32+BaseAdd;

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

Address =((MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;

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

Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;

6.3.1.2 8X16 点国标扩展字符

说明:

BaseAdd: 说明本套字库在字库芯片中的起始字节地址。

FontCode: 表示字符内码 (16bits)

ByteAddress: 表示字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3b7d0

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

ByteAddress = (FontCode - 0xAAA1) * 16 + BaseAdd

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

ByteAddress = (FontCode - 0xABA1 + 95) * 16 + BaseAdd

6.3.2 ASCII 字符的地址计算

6.3.2.1 5X7 点 ASCII 字符

参数说明:

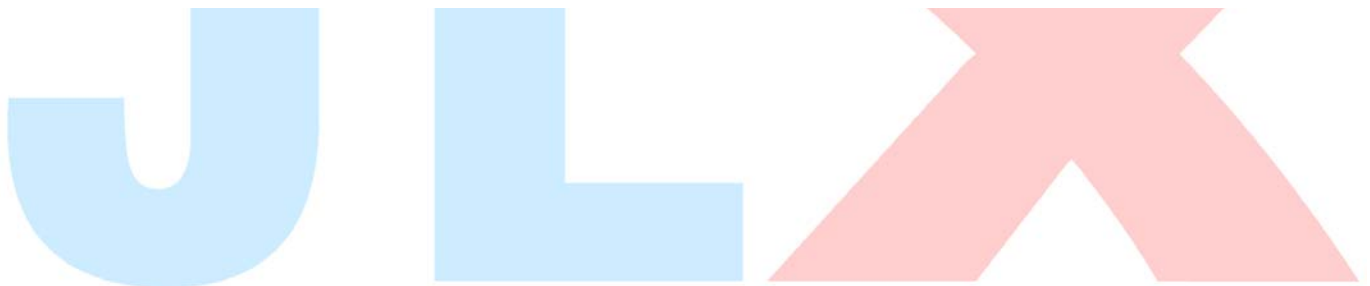
ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3bfc0



if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode - 0x20) * 8 + BaseAdd

6.3.2.2 7X8 点 ASCII 字符

参数说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x66c0

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode - 0x20) * 8 + BaseAdd

6.3.2.3 8X16 点 ASCII 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3b7c0

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode - 0x20) * 16 + BaseAdd



6.3.2.4 16 点阵不等宽 ASCII 方头 (Arial) 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3c2c0

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode - 0x20) * 34 + BaseAdd

6.3.2.5 8X16 点 ASCII 粗体字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3cf80

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode - 0x20) * 16 + BaseAdd

6.3.2.6 16 点阵不等宽 ASCII 白正 (Times New Roman) 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3d580

if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then

Address = (ASCIICode - 0x20) * 34 + BaseAdd

6.4 附录

6.4.1 GB2312 1 区 (376 字符)

GB2312 标准点阵字符 1 区对应码位的 A1A1~A9EF 共计 376 个字符;

GB2312 1 区

A1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A			、	。	·	—	√	”	々	一	~		...	‘	’	
B	“	”	{	}	<	>	《	》	「	」	『	』	【	】	【	】
C	±	×	÷	:	∧	∨	Σ	Π	U	∩	€	::	√	⊥	//	∠
D	∩	⊙	∫	∫	≡	∞	≈	∞	∞	≠	≠	≠	≠	≠	∞	∞
E	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴
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	{		}	~	

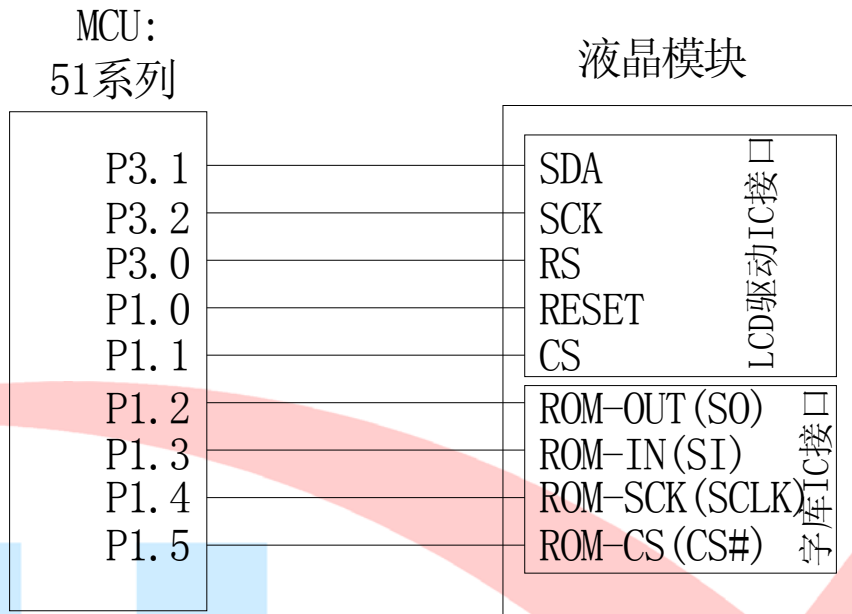
AB 0 1 2 3 4 5 6 7 8 9 A B C D E F

A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ũ	ú	ǔ	ù	ü	ê	ɑ	ǻ	ǻ	ǻ	ǻ
C	g															

7. 硬件设计及例程:

7.1 当 LCD 驱动 IC 采用串行接口方式时的硬件设计及例程:

7.1.1 硬件接口: 下图为串行方式的硬件接口:



7.1.2 例程: 以下为串行方式显示汉字及 ASCII 字符的例程:

```
//液晶屏型号: JLX12864G-08602-PC-3S
//接口: 串行
//中文字库: 带中文字库 IC: JLX-GB2312, 竖置横排
//驱动 IC: UC1701X (与 ST7565R 兼容)
//主板: 89C516-001 老主板
//单片机: 51 系列
//版权所有: 深圳市晶联讯电子有限公司; 网址 http://www.jlxlcd.cn;
//编程: 叶建人, 2013 年 10 月 11 日
```

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

sbit lcd_sclk=P3^2; //接口定义: lcd_sclk 就是 LCD 的 SCLK
sbit lcd_sid=P3^1;  //接口定义: lcd_sid 就是 LCD 的 SDA
sbit lcd_rs=P3^0;   //接口定义: lcd_rs 就是 LCD 的 RS, 也叫 "CD"
sbit lcd_reset=P1^0; //接口定义: lcd_reset 就是 LCD 的 RESET
sbit lcd_cs1=P1^1;  //接口定义: lcd_cs1 就是 LCD 的 CS1

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#

sbit key=P2^0;      //定义一个按键
```

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

uchar code bmp1[];

uchar code jiong1[]={//-- 文字: 囧 --
//-- 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 --
0x00, 0xFE, 0x82, 0x42, 0xA2, 0x9E, 0x8A, 0x82, 0x86, 0x8A, 0xB2, 0x62, 0x02, 0xFE, 0x00, 0x00,
0x00, 0x7F, 0x40, 0x40, 0x7F, 0x40, 0x40, 0x40, 0x40, 0x7F, 0x40, 0x40, 0x7F, 0x00, 0x00};

uchar code lei1[]={//-- 文字: 晶 --
//-- 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 --
0x80, 0x80, 0x80, 0xBF, 0xA5, 0xA5, 0xA5, 0x3F, 0xA5, 0xA5, 0xA5, 0xBF, 0x80, 0x80, 0x80, 0x00,
0x7F, 0x24, 0x24, 0x3F, 0x24, 0x24, 0x7F, 0x00, 0x7F, 0x24, 0x24, 0x3F, 0x24, 0x24, 0x7F, 0x00};

//延时
void delay(int n_ms)
{
    int j, k;
    for(j=0; j<n_ms; j++)
        for(k=0; k<110; k++);
}

//短延时
void delay_us(int n_us)
{
    int j, k;
    for(j=0; j<n_us; j++)
        for(k=0; k<1; k++);
}

//等待一个按键, 我的主板是用 P2.0 与 GND 之间接一个按键
void waitkey()
{
    repeat:
        if (key==1) goto repeat;
        else delay(600);
}

//写指令到 LCD 模块
void transfer_command_lcd(int data1)
{
    char i;
    lcd_cs1=0;
    lcd_rs=0;
    for(i=0; i<8; i++)
    {
        lcd_sclk=0;
        //delay_us(10); //加少量延时
        if(data1&0x80) lcd_sid=1;
        else lcd_sid=0;
        lcd_sclk=1;
        //delay_us(10); //加少量延时
        data1=data1<<1;
    }
    lcd_cs1=1;
}

//写数据到 LCD 模块
void transfer_data_lcd(int data1)
{
    char i;
```

```
lcd_cs1=0;
lcd_rs=1;
for(i=0;i<8;i++)
{
    lcd_sclk=0;
    if(data1&0x80) lcd_sid=1;
    else lcd_sid=0;
    lcd_sclk=1;
    data1=data1<<=1;
}
lcd_cs1=1;
}
```

//LCD 模块初始化

```
void initial_lcd()
{
    lcd_reset=0;          //低电平复位
    delay(100);
    lcd_reset=1;          //复位完毕
    delay(100);
    transfer_command_lcd(0xe2); //软复位
    delay(5);
    transfer_command_lcd(0x2c); //升压步骤 1
    delay(50);
    transfer_command_lcd(0x2e); //升压步骤 2
    delay(50);
    transfer_command_lcd(0x2f); //升压步骤 3
    delay(5);
    transfer_command_lcd(0x23); //粗调对比度, 可设置范围 0x20~0x27
    transfer_command_lcd(0x81); //微调对比度
    transfer_command_lcd(0x28); //微调对比度的值, 可设置范围 0x00~0x3f
    transfer_command_lcd(0xa2); //1/9 偏压比 (bias)
    transfer_command_lcd(0xc8); //行扫描顺序: 从上到下
    transfer_command_lcd(0xa0); //列扫描顺序: 从左到右
    transfer_command_lcd(0x40); //起始行: 第一行开始
    transfer_command_lcd(0xaf); //开显示
}
```

```
void lcd_address(uint page, uint column)
{
    column=column-0x01;
    transfer_command_lcd(0xb0+page-1); //设置页地址, 每 8 行为一页, 全屏共 64 行, 被分成 8 页
    transfer_command_lcd(0x10+(column>>4&0x0f)); //设置列地址的高 4 位
    transfer_command_lcd(column&0x0f); //设置列地址的低 4 位
}
```

//全屏清屏

```
void clear_screen()
{
    unsigned char i, j;

    for(i=0;i<9;i++)
    {
        transfer_command_lcd(0xb0+i);
        transfer_command_lcd(0x10);
        transfer_command_lcd(0x00);
        for(j=0;j<132;j++)
        {
            transfer_data_lcd(0x00);
        }
    }
}
```

```
//显示 128x64 点阵图像
void display_128x64(uchar *dp)
{
    uint i, j;

    for(j=0; j<8; j++)
    {
        lcd_address(j+1, 1);
        for (i=0; i<128; i++)
        {
            transfer_data_lcd(*dp);           //写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1
            dp++;
        }
    }
}
```

```
//显示 16x16 点阵图像、汉字、生僻字或 16x16 点阵的其他图标
void display_graphic_16x16(uchar page, uchar column, uchar *dp)
{
    uint i, j;
    for(j=0; j<2; j++)
    {
        lcd_address(page+j, column);
        for (i=0; i<16; i++)
        {
            transfer_data_lcd(*dp);           //写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1
            dp++;
        }
    }
}
```

```
//显示 8x16 点阵图像、ASCII, 或 8x16 点阵的自造字符、其他图标
void display_graphic_8x16(uchar page, uchar column, uchar *dp)
{
    uint i, j;
    for(j=0; j<2; j++)
    {
        lcd_address(page+j, column);
        for (i=0; i<8; i++)
        {
            transfer_data_lcd(*dp);           //写数据到 LCD, 每写完一个 8 位的数据后列地址自动加 1
            dp++;
        }
    }
}
```

```
//显示 5x8 点阵图像、ASCII, 或 5x8 点阵的自造字符、其他图标
void display_graphic_5x8(uchar page, uchar column, uchar *dp)
{
    uint i;
    lcd_address(page, column);
    for (i=0; i<6; i++)
    {
        transfer_data_lcd(*dp);
        dp++;
    }
}
```

```
//送指令到晶联讯字库 IC
void send_command_to_ROM( uchar datu )
{

```

```
uchar i;
for(i=0;i<8;i++)
{
    Rom_SCK=0;
    delay_us(10);
    if(datu&0x80)Rom_IN = 1;
    else Rom_IN = 0;
    datu = datu<<1;
    Rom_SCK=1;
    delay_us(10);
}
}

//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;
    for(i=0;i<8;i++)
    {
        Rom_OUT=1;
        Rom_SCK=0;
        //delay_us(1);
        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);
}

//从指定地址读出数据写到液晶屏指定 (page, column)座标中
void get_and_write_16x16(ulong fontaddr,uchar page,uchar column)
{
    uchar i, j, disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位, 共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位, 共 24 位
    send_command_to_ROM(fontaddr&0xff); //地址的低 8 位, 共 24 位
    for(j=0;j<2;j++)
    {
        lcd_address(page+j, column);
        for(i=0; i<16; i++)
        {
            disp_data=get_data_from_ROM();
            transfer_data_lcd(disp_data); //写数据到 LCD, 每写完 1 字节的数据后列地址自动加 1
        }
    }
    Rom_CS=1;
}

//从指定地址读出数据写到液晶屏指定 (page, column)座标中
void get_and_write_8x16(ulong fontaddr,uchar page,uchar column)
{
    uchar i, j, disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位, 共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位, 共 24 位
    send_command_to_ROM(fontaddr&0xff); //地址的低 8 位, 共 24 位
    for(j=0;j<2;j++)
```

```
{
    lcd_address(page+j, column);
    for(i=0; i<8; i++)
    {
        disp_data=get_data_from_ROM();
        transfer_data_lcd(disp_data);    //写数据到 LCD, 每写完 1 字节的数据后列地址自动加 1
    }
}
Rom_CS=1;
}

//从指定地址读出数据写到液晶屏指定 (page, column)座标中
void get_and_write_5x8(ulong fontaddr, uchar page, uchar column)
{
    uchar i, disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM(((fontaddr&0xff0000)>>16);    //地址的高 8 位, 共 24 位
    send_command_to_ROM(((fontaddr&0xff00)>>8);    //地址的中 8 位, 共 24 位
    send_command_to_ROM(fontaddr&0xff);    //地址的低 8 位, 共 24 位
    lcd_address(page, column);
    for(i=0; i<5; i++)
    {
        disp_data=get_data_from_ROM();
        transfer_data_lcd(disp_data);    //写数据到 LCD, 每写完 1 字节的数据后列地址自动加 1
    }
    Rom_CS=1;
}

//*****

ulong fontaddr=0;
void display_GB2312_string(uchar page, uchar column, uchar *text)
{
    uchar i= 0;
    while((text[i]>0x00))
    {
        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            //国标简体 (GB2312) 汉字在晶联讯字库 IC 中的地址由以下公式来计算:
            //Address = ((MSB - 0xb0) * 94 + (LSB - 0xa1) + 846)*32+ BaseAdd; BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题, 所以分三部取地址
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong) (fontaddr*32);

            get_and_write_16x16(fontaddr, page, column);    //从指定地址读出数据写到液晶屏指定 (page, column)座标中
            i+=2;
            column+=16;
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xa3))&&(text[i+1]>=0xa1))
        {
            //国标简体 (GB2312) 15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算:
            //Address = ((MSB - 0xa1) * 94 + (LSB - 0xa1))*32+ BaseAdd; BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题, 所以分三部取地址
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong) (fontaddr*32);

            get_and_write_16x16(fontaddr, page, column);    //从指定地址读出数据写到液晶屏指定 (page, column)座标中
            i+=2;
            column+=16;
        }
        else if((text[i]>=0x20) &&(text[i]<=0x7e))
```



```
{
    fontaddr = (text[i]- 0x20);
    fontaddr = (unsigned long)(fontaddr*16);
    fontaddr = (unsigned long)(fontaddr+0x3cf80);

    get_and_write_8x16(fontaddr, page, column);    //从指定地址读出数据写到液晶屏指定 (page, column)座标中
    i+=1;
    column+=8;
}
else
    i++;
}

}

void display_string_5x8(uchar page, uchar column, uchar *text)
{
    unsigned char i= 0;
    while((text[i]>0x00))
    {

        if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long)(fontaddr*8);
            fontaddr = (unsigned long)(fontaddr+0x3bfc0);

            get_and_write_5x8(fontaddr, page, column);    //从指定地址读出数据写到液晶屏指定 (page, column)座标中

            i+=1;
            column+=6;
        }
        else
            i++;
    }
}

//=====main program=====
void main(void)
{
    while(1)
    {
        initial_lcd();
        clear_screen();    //clear all dots
        display_128x64(bmp1);
        delay(2000);
        display_GB2312_string(1, 1, "12864, 带中文字库");    //在第 1 页, 第 1 列, 显示一串 16x16 点阵汉字或 8x16 的 ASCII 字
        display_GB2312_string(3, 1, "16X16 简体汉字库,");    //显示一串 16x16 点阵汉字或 8x16 的 ASCII 字. 以下雷同
        display_GB2312_string(5, 1, "或 8X16 点阵 ASCII,");
        display_GB2312_string(7, 1, "或 5x8 点阵 ASCII 码");
        delay(4000);
        clear_screen();
        display_GB2312_string(1, 1, "晶联讯成立于二零");
        display_GB2312_string(3, 1, "零四年十一月七日");
        display_GB2312_string(5, 1, "主要生产液晶模块");
        display_GB2312_string(7, 1, "品质至上真诚服务");
        delay(4000);
        display_GB2312_string(1, 1, "GB2312 简体字库及");
        display_GB2312_string(3, 1, "有图型功能, 可自");
        display_GB2312_string(5, 1, "编大字或图像或生");
        display_GB2312_string(7, 1, "僻字, 例如:    ");
        display_graphic_16x16(7, 97, jiong1);
        display_graphic_16x16(7, 113, lei1);
        delay(4000);
    }
}
```

//在第 7 页, 第 81 列显示单个自编生僻汉字“囧”
//显示单个自编生僻汉字“晶”

```

clear_screen();
display_GB2312_string(1,1,"<!@#%^&*()_+]/"); //在第1页，第1列，显示一串16x16点阵汉字或8*16的ASCII字
display_string_5x8(3,1,"<!@#%^&*()_+]/;.:?[")://在第3页，第1列，显示一串5x8点阵的ASCII字
display_string_5x8(4,1,"JLX electronics Co., ");://显示一串5x8点阵的ASCII字
display_string_5x8(5,1,"Ltd. established at ");://显示一串5x8点阵的ASCII字
display_string_5x8(6,1,"year 2004.Focus LCM. ");://显示一串5x8点阵的ASCII字
display_string_5x8(7,1,"TEL:0755-29784961 ");://显示一串5x8点阵的ASCII字
display_string_5x8(8,1,"FAX:0755-29784964 ");://显示一串5x8点阵的ASCII字
delay(4000);

display_GB2312_string(1,1,"啊阿埃挨埃唉哀皑"); //在第1页，第1列，显示一串16x16点阵汉字或8x16的ASCII字
display_GB2312_string(3,1,"瘡瘍矮艾碍爰隘鞍"); //显示一串16x16点阵汉字或8x16的ASCII字。以下雷同
display_GB2312_string(5,1,"氨安俺按暗岸胺案");
display_GB2312_string(7,1,"肮昂盎凹熬熬熬袄");
delay(4000);

display_GB2312_string(1,1,"鬣鬣麼麼麼麼麼麼");
display_GB2312_string(3,1,"麋麋麋麋麋麋麋麋");
display_GB2312_string(5,1,"黠黠黠黠黠黠黠黠");
display_GB2312_string(7,1,"黝黝黝黝黝黝黝黝");
delay(4000);
}
}

```

```
uchar code bmp1[] = {  
    //--- 调入了一幅图像：D:\我的文档\My Pictures\12864-555.bmp ---  
    //--- 宽度 x 高度=128x64 ---  
  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0x60, 0xE0, 0xE0, 0xE0, 0xE0, 0x60,  
0x60, 0x60, 0x60, 0x60, 0x60, 0x60, 0xE0, 0xE0, 0xE0, 0xE0, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x80,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 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, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xE0, 0x70, 0x38, 0x18, 0x1C, 0x0C,  
0x0E, 0x07, 0x03, 0x03, 0x01, 0x81, 0xE0, 0x78, 0x1C, 0x0E, 0x07, 0x03, 0x01, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0E, 0x1C, 0x79, 0xE1, 0x83,  
0x03, 0x07, 0x0E, 0x0C, 0x1C, 0x18, 0x38, 0x70, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0,  
0xC0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,  
0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0, 0xE0,  
0xE0, 0xE0, 0xE0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0xF0, 0x78, 0x1E, 0x07, 0x03, 0x81, 0x80, 0xC0, 0xC0, 0xC0, 0xE0, 0x60,  
0x60, 0x60, 0x70, 0xF0, 0xFE, 0x3F, 0x19, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,  
0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,  
0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18, 0x18,  
0xFF, 0xF0, 0x60, 0x60, 0xE0, 0xC0, 0xC0, 0xC0, 0x80, 0x81, 0x03, 0x07, 0x1E, 0x78, 0xF0, 0x80,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0xFC, 0xFF, 0x7F, 0x07, 0x0F, 0x1F, 0x3D,  
0x79, 0x71, 0xF0, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x80, 0x80, 0xC0, 0xE0, 0xF0, 0x71, 0x79, 0x3D, 0x1F, 0x0F, 0xFF, 0xFE, 0xFC, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x80, 0xFE, 0xFF, 0xF9, 0x1C, 0x0E, 0x07, 0x03, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0xFF, 0xFF, 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, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x03, 0x07, 0x0E, 0x1C, 0xF9, 0xFF,  
0xFE, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0F, 0x0E, 0x9E, 0xFC, 0xF8, 0xF0, 0xE0, 0xC0, 0xC0, 0x80,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0xC0, 0xC0, 0xE0, 0xF0, 0xF8, 0xF8, 0xFC, 0x9E, 0x0F, 0x07,  
0x03, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x3F, 0xFF, 0xDF, 0x38, 0x70, 0xE0, 0xC0, 0xC0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0xFF, 0xFF, 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, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x80, 0x80, 0xC0, 0xC0, 0xE0, 0x70, 0x38, 0xDF, 0xFF,  
0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x7F, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,  
0x80, 0xC0, 0xE0, 0xF0, 0x78, 0x3C, 0x1E, 0x0F, 0x07, 0x03, 0x01, 0x00, 0x01, 0x01, 0x01, 0x03, 0x07, 0x0F, 0x1E,  
0x3C, 0x78, 0xF0, 0xE0, 0xC0, 0x80, 0x80, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00,  
0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x3C, 0x70, 0xE0, 0xC1, 0x81, 0x03, 0x03, 0x03, 0x03, 0x07, 0x00,
```

