JLX25696G-966-PL

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：	3～4
3	外形尺寸及接口引脚功能	5～7
4	工作电路框图	7
5	指令	8～9
6	字库排置	10～11
7	点阵数据验证	12
8	附录	13～17
9	硬件设计及例程：	18～页末

1．概述

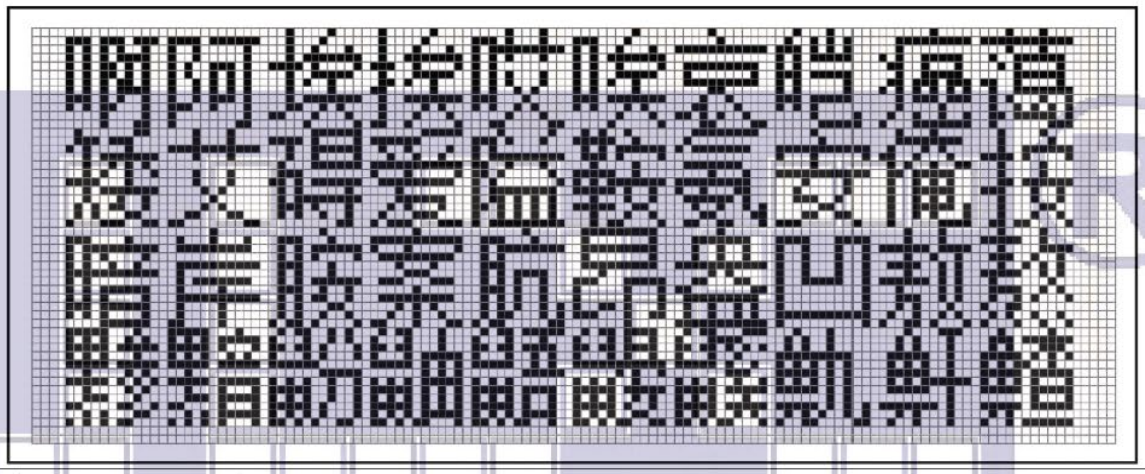
JLX25696G-966-PL 型液晶显示模块既可以当成普通的图像型液晶显示模块使用(即显示普通图像型的单色图片功能)，又含有 JLX-GB2312-3207 字库 IC，可以从字库 IC 中读出内置的字库的点阵数据写入到 LCD 驱动 IC 中，以达到显示汉字的目的。

此字库 IC 存储内容如下表所述： 表 1

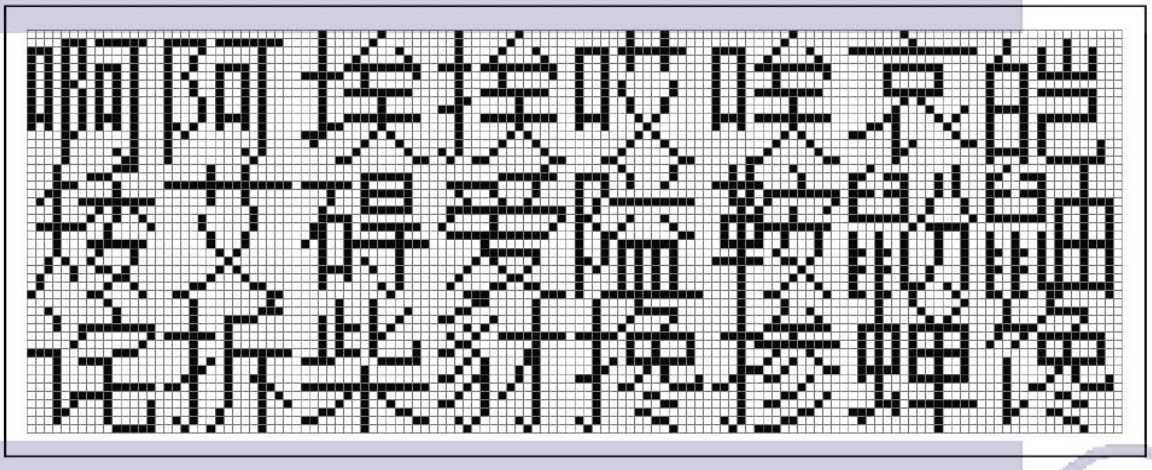
分类	字库	字号	字符数	字体	排列方式	备注
ASSII 字符集	ASCII	5x7	96	标准	Y-竖置横排	
	ASCII	7X8	96	标准	Y-竖置横排	
	ASCII	6X12	96	标准	Y-竖置横排	
	ASCII	8X16	96	标准	Y-竖置横排	
	ASCII	12X24	96	标准	Y-竖置横排	
	ASCII	16X32	96	标准	Y-竖置横排	
	ASCII	12 点阵不等宽	96	Arial（方头）	Y-竖置横排	
	ASCII	12 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排	
	ASCII	16 点阵不等宽	96	Arial（方头）	Y-竖置横排	
	ASCII	16 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排	
	ASCII	24 点阵不等宽	96	Arial（方头）	Y-竖置横排	
	ASCII	24 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排	
	ASCII	32 点阵不等宽	96	Arial（方头）	Y-竖置横排	
	ASCII	32 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排	
汉字字符集	GB2312 汉字	12X12	6763	宋体	Y-竖置横排	
		16X16	6763	宋体	Y-竖置横排	
		24X24	6763	宋体	Y-竖置横排	
		32X32	6763	宋体	Y-竖置横排	
	GB2312 字符	12X12	846	宋体	Y-竖置横排	
		16X16	846	宋体	Y-竖置横排	
		24X24	846	宋体	Y-竖置横排	
		32X32	846	宋体	Y-竖置横排	
	国际扩展字符	6X12	126	宋体	Y-竖置横排	
		8X16	126	宋体	Y-竖置横排	
		12X24	126	宋体	Y-竖置横排	
		16X32	126	宋体	Y-竖置横排	

2．字型样张：

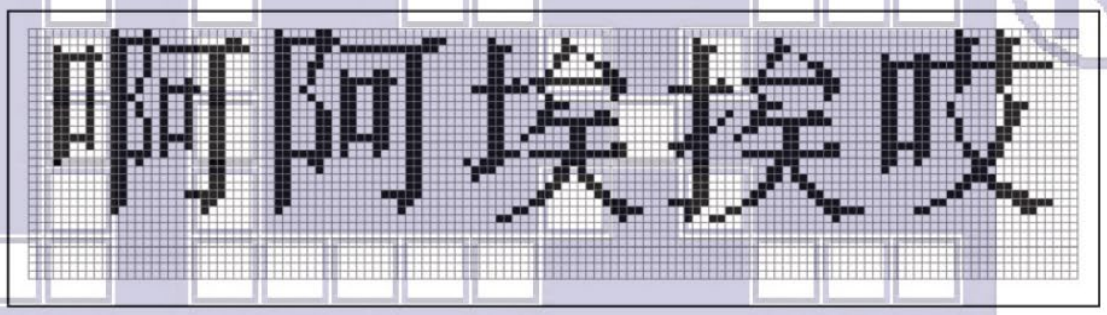
12x12 点阵 GB2312 汉字



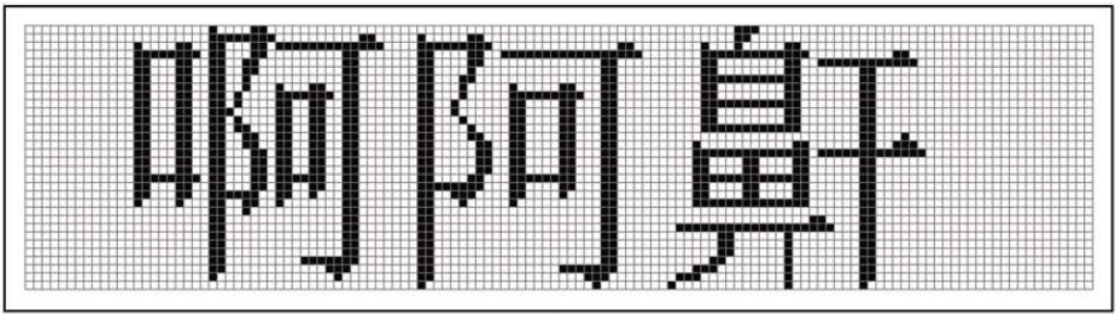
16x16 点阵 GB2312 汉字



24x24 点阵 GB2312 汉字

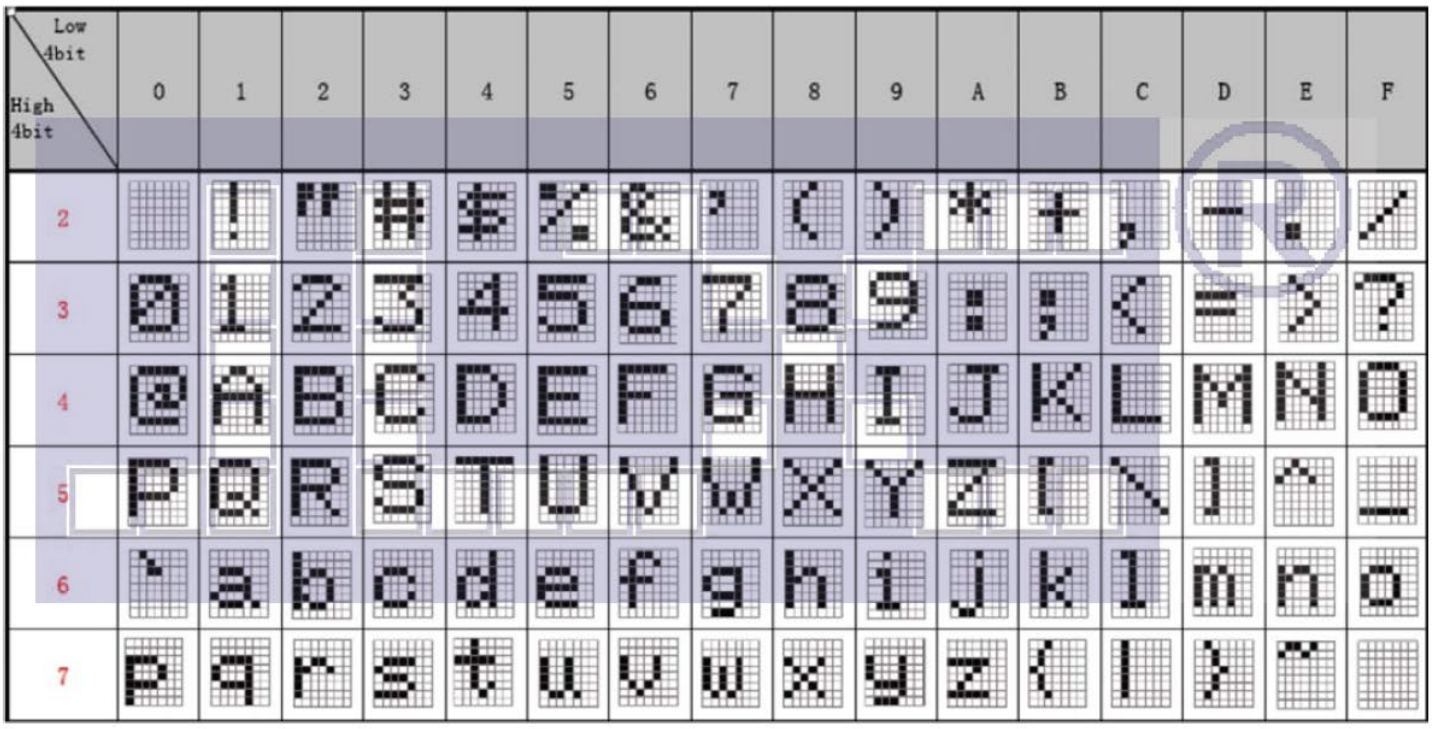


32x32 点阵 GB2312 汉字



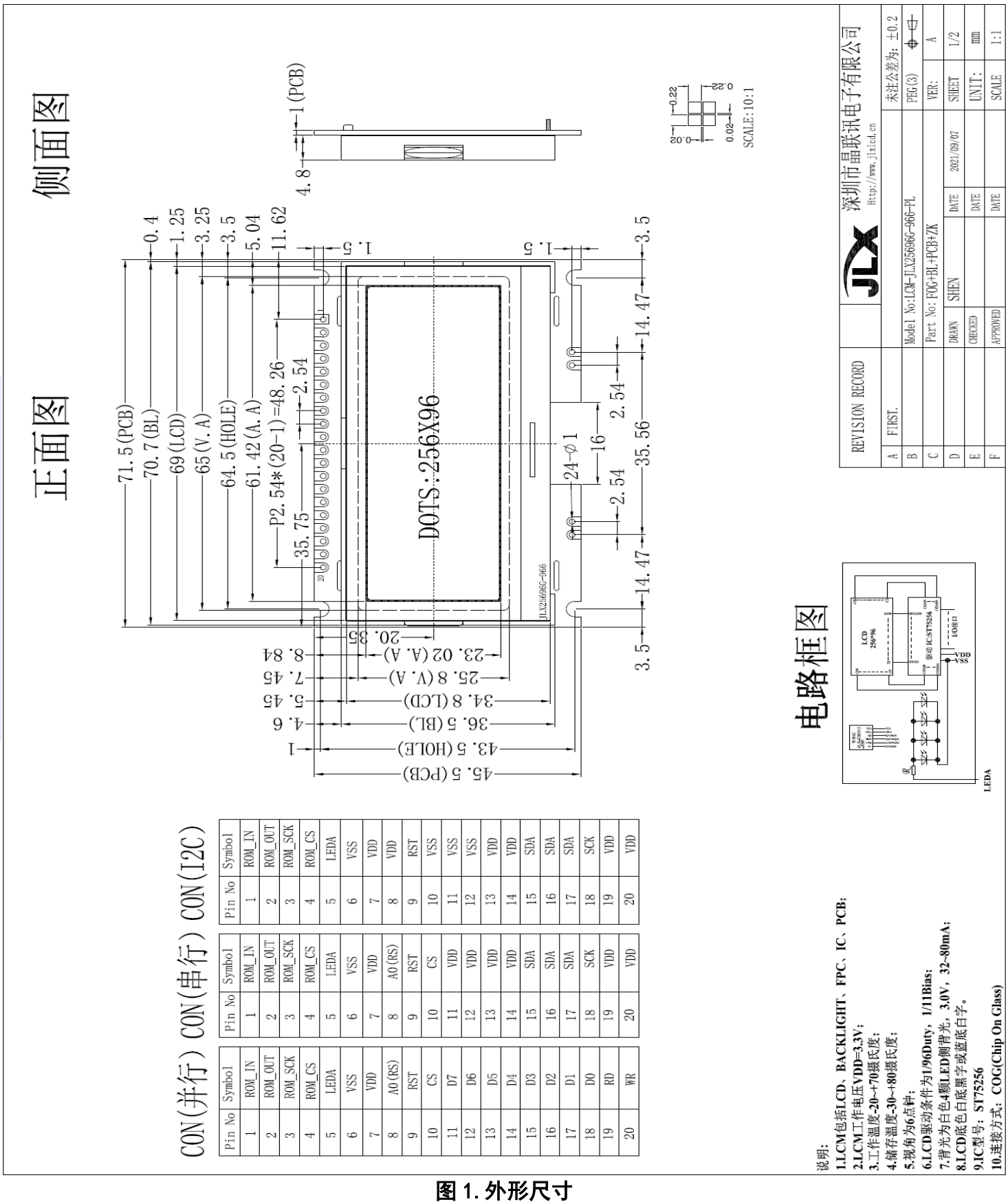
5x7 点阵 ASCII 标准字符

7x8 点阵 ASCII 标准字符



3. 外形尺寸及接口引脚功能

3.1 外形图：



CON(并行) CON(串行) CON(I2C)

Pin No	Symbol	Pin No	Symbol	Pin No	Symbol
1	ROM_IN	1	ROM_IN	1	ROM_IN
2	ROM_OUT	2	ROM_OUT	2	ROM_OUT
3	ROM_SCK	3	ROM_SCK	3	ROM_SCK
4	ROM_CS	4	ROM_CS	4	ROM_CS
5	LEDA	5	LEDA	5	LEDA
6	VSS	6	VSS	6	VSS
7	VDD	7	VDD	7	VDD
8	A0(RS)	8	A0(RS)	8	VDD
9	RST	9	RST	9	RST
10	CS	10	CS	10	VSS
11	D7	11	VDD	11	VSS
12	D6	12	VDD	12	VSS
13	D5	13	VDD	13	VDD
14	D4	14	VDD	14	VDD
15	D3	15	SDA	15	SDA
16	D2	16	SDA	16	SDA
17	D1	17	SDA	17	SDA
18	D0	18	SCK	18	SCK
19	RD	19	VDD	19	VDD
20	WR	20	VDD	20	VDD

图 1. 外形尺寸

3.2.1 模块的并行接口引脚功能

引 线 号	符 号	名 称	功 能	
1	ROM-IN	字库 IC 接口 SI	串行数据输入	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	供电电源负极	供电电源负极	
7	VDD	供电电源正极	供电电源正极（注意：购买时须选择 3.3V 或者是 5V 供电）	
8	RS	寄存器选择信号	H:数据寄存器 0:指令寄存器（IC 资料上所写为”A0”）	
9	RES	复位	低电平复位，复位完成后，回到高电平，液晶模块开始工作	
10	CS	片选	低电平片选	
11～18	D7-D0	I/O	数据总线 DB7-DB0	
19	E	使能信号	并行时：使能信号	
20	R/W	读/写	并行时：H:读数据 0:写数据	

表 2：模块并行接口引脚功能

3.2.2 串行时接口引脚功能

引 线 号	符 号	名 称	功 能	
1	ROM-IN	字库 IC 接口 SI	串行数据输入	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	供电电源负极	供电电源负极	
7	VDD	供电电源正极	供电电源正极（注意：购买时须选择 3.3V 或者是 5V 供电）	
8	RS	寄存器选择信号	H:数据寄存器 0:指令寄存器（IC 资料上所写为”A0”）	
9	RES	复位	低电平复位，复位完成后，回到高电平，液晶模块开始工作	
10	CS	片选	低电平片选	
11-14	D7-D4	I/O	串行接口，D7-D4 引脚接 VDD	
15-17	D3-D1 (SDA)	I/O	串行数据（D1、D2、D3 短接一起作为 SDA）	
18	D0(SCK)	I/O	串行时钟	
19	E	使能信号	悬空或接 VDD	
20	R/W	读/写	悬空或接 VDD	

表 3：模块串行接口引脚功能

3.2.3 I2C 时接口引脚功能

引 线 号	符 号	名 称	功 能	
1	ROM-IN	字库 IC 接口 SI	串行数据输入	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	供电电源负极	供电电源负极	
7	VDD	供电电源正极	供电电源正极（注意：购买时须选择 3.3V 或者是 5V 供电）	
8	RS(A0)	寄存器选择信号	悬空或接 VDD	
9	RES	复位	低电平复位，复位完成后，回到高电平，液晶模块开始工作	
10	CS	片选	IIC 接口，此引脚必须接 VSS	
11	D7	I/O	IIC 接口，从属地址接 VSS	
12	D6	I/O	IIC 接口，从属地址接 VSS	
13	D5	I/O	悬空或接 VDD	
14	D4	I/O	悬空或接 VDD	
15-17	D3-D1(SDA)	I/O	串行数据（D1、D2、D3 短接一起作为 SDA）	
18	D0(SCK)	I/O	串行时钟	
19	RD(E)	使能信号	悬空或接 VDD	
20	WR	读/写	悬空或接 VDD	

表 4：模块 IIC 接口引脚功能

4. 工作电路框图：

见图 2，模块由 LCD 驱动 IC ST75256、字库 IC、背光组成。

电路框图

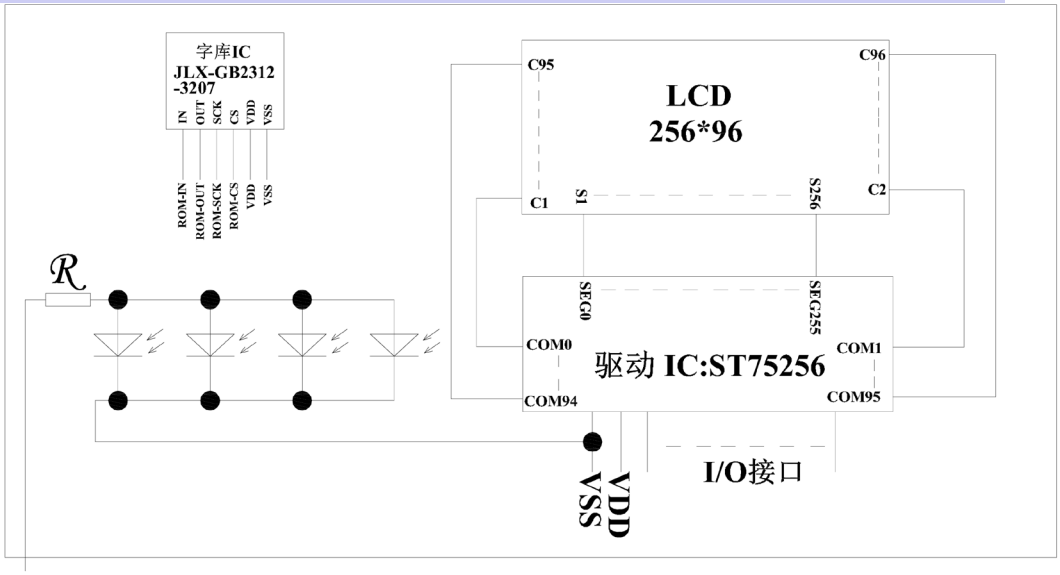


图 2:JLX25696G-966-PL 电路框图

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)：

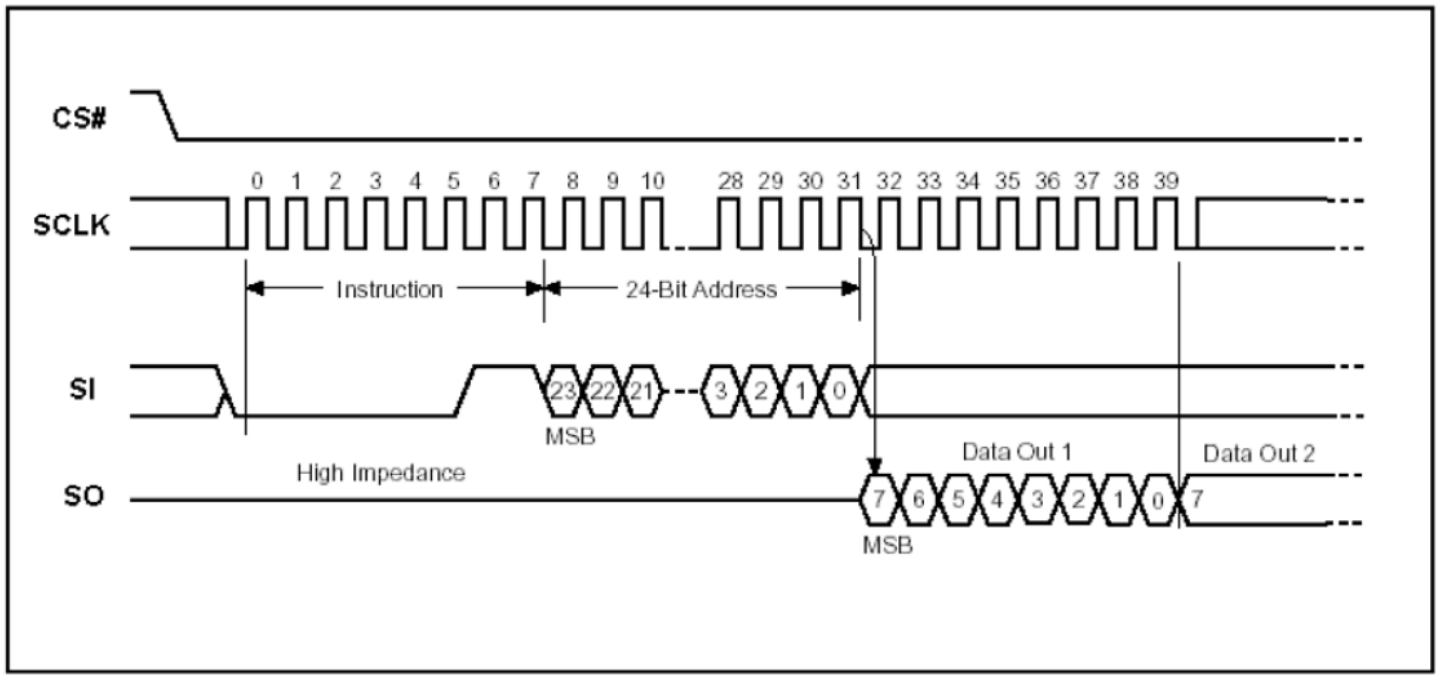
首先把片选信号（CS#）变为低，紧跟着的是1 个字节的命令字（03 h）和3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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)：

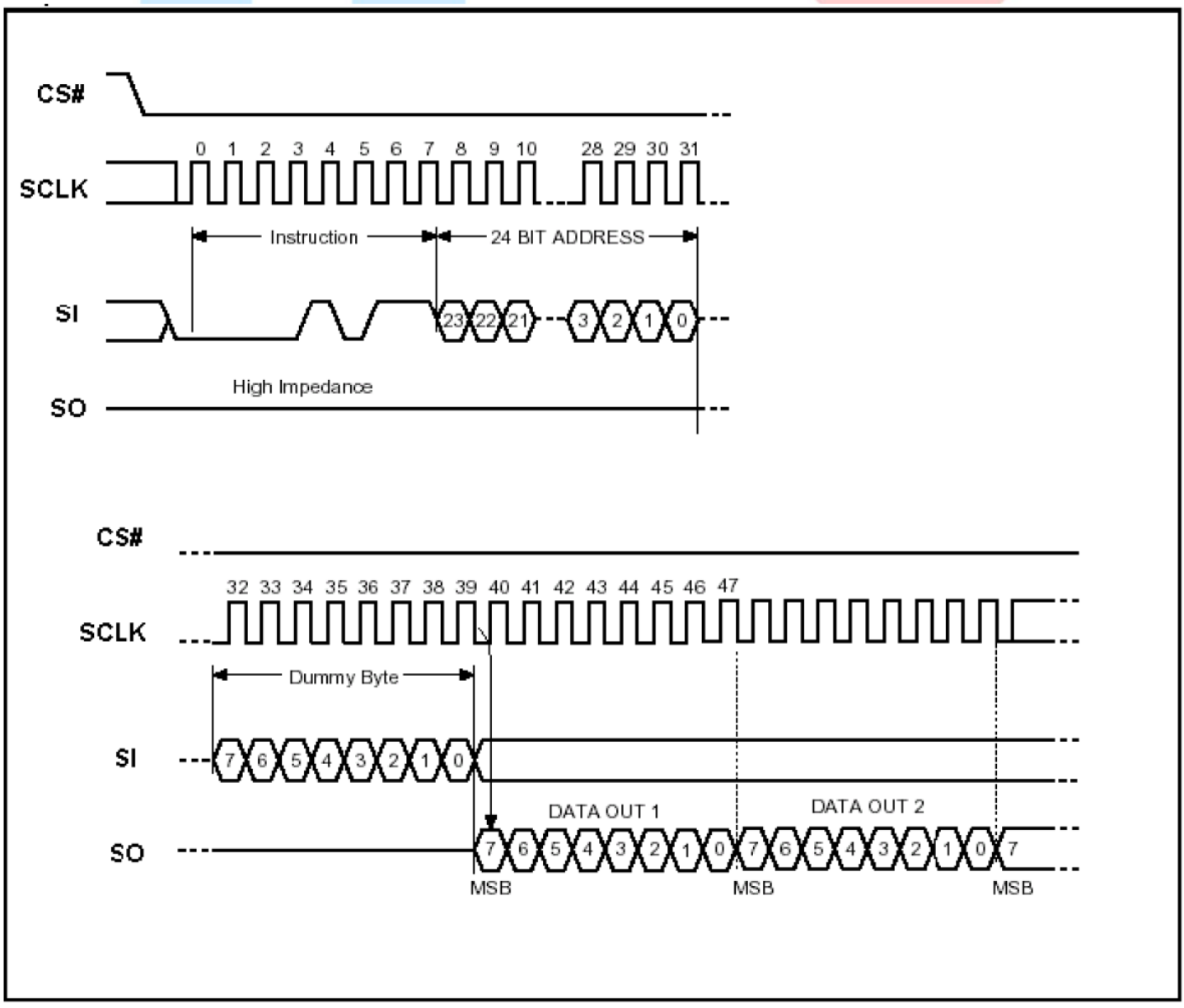
首先把片选信号（CS#）变为低，紧跟着的是1 个字节的命令字（0B h）和3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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:



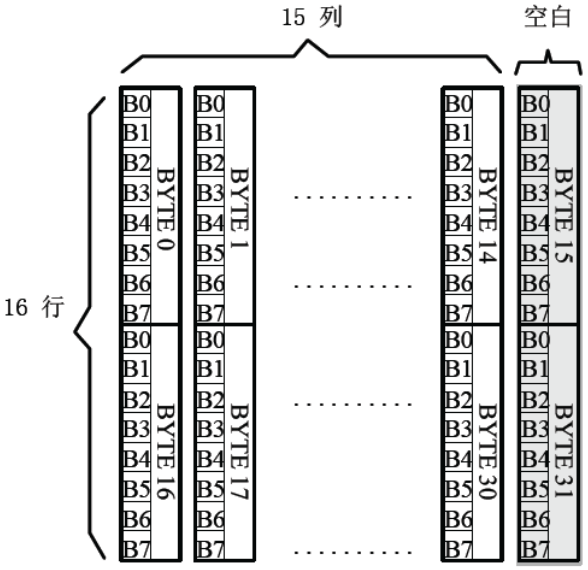
6 字库排置（竖置横排）

6.1 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：即一个字节的高位表示下面的点，低位表示上面的点（如果用户按 16bit 总线宽度读取点阵数据，请注意高低字节的顺序），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。

6.2 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竖置横排的，其具体排列结构如下图：

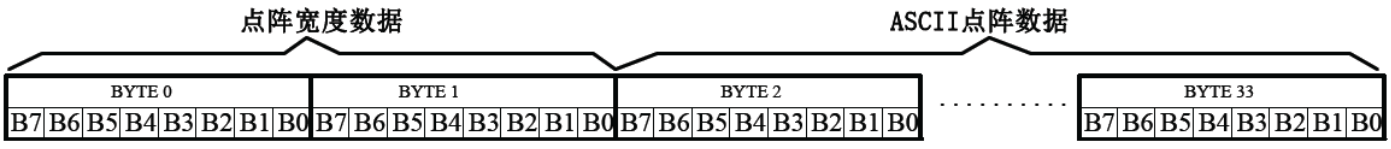


6.3 16 点阵不等宽 ASCII 方头（Arial）字符排列格式

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

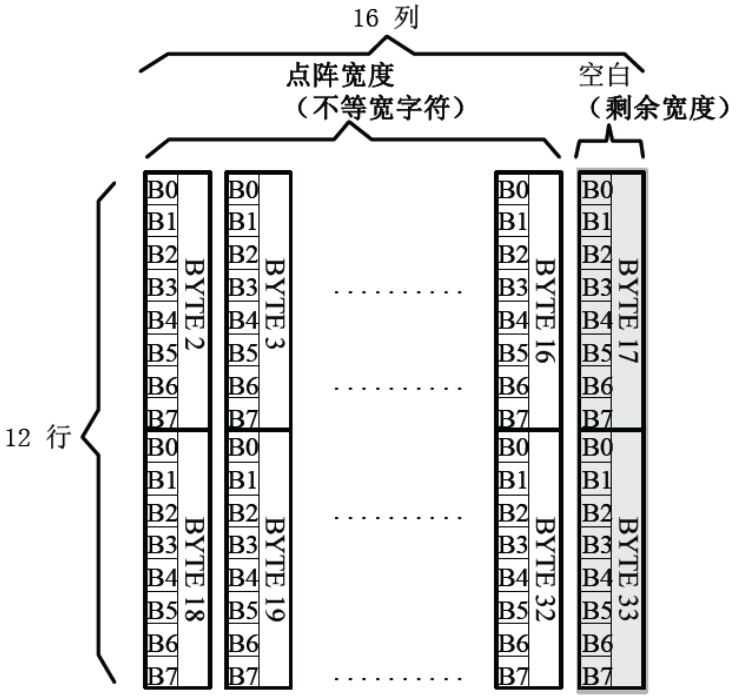
■ 存储格式

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 存放竖置横排点阵数据。具体格式见下图：



■ 存储结构

点阵存储宽度固定为 16，根据不同字符，其实际点阵宽度会小于 16，并会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的宽度数据，可以对还原下一个字的显示或排版留作参考。



例如：ASCII 方头字符 B

0-33BYTE 的点阵数据是: 00 0C 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 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。（见下图）

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 63 67 3E 1C 00 00 00 00 00 为 ASCII 方头字符 B 的点阵数据。

7 点阵数据验证（客户参考用）

客户将芯片内“A”的数据调出与以下进行对比。若一致，表示 SPI 驱动正常工作；若不一致，请重新编写驱动。

排置：Y（竖置横排）点阵大小 8X16
字母"A"
点阵数据：00 E0 9C 82 9C E0 00 00 0F 00 00 00 00 00 0F 00

排置：W（横置横排）点阵大小 8X16
字母"A"
点阵数据：00 10 28 28 28 44 44 7C 82 82 82 82 00 00 00 00

8 附录

8.1 GB23121 区字符（846 字符）

GB2312 标准点阵字符 1 区对应码位的 A1A1~A9EF 共计 846 个字符；

GB2312 1 区

GB2312 1区

A4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		あ	あ	い	い	う	う	え	え	お	お	か	が	き	ぎ	く
B	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
C	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
D	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
E	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
F	ゐ	ゑ	を	ん												

A5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
B	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
C	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
D	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
E	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
F	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									

A6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
B	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
C		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
D	π	ρ	σ	τ	υ	φ	χ	ψ	ω	'	°	ˋ	:	;	!	?
E	⏜	⏝	⏜	⏝	⏜	⏝	⏞	⏟	⎴	⎵	⏤	⏤	⏤	⏤	▬	▬
F	⏜	⏝	∣	⋮	∣	⎪										

GB2312 1区

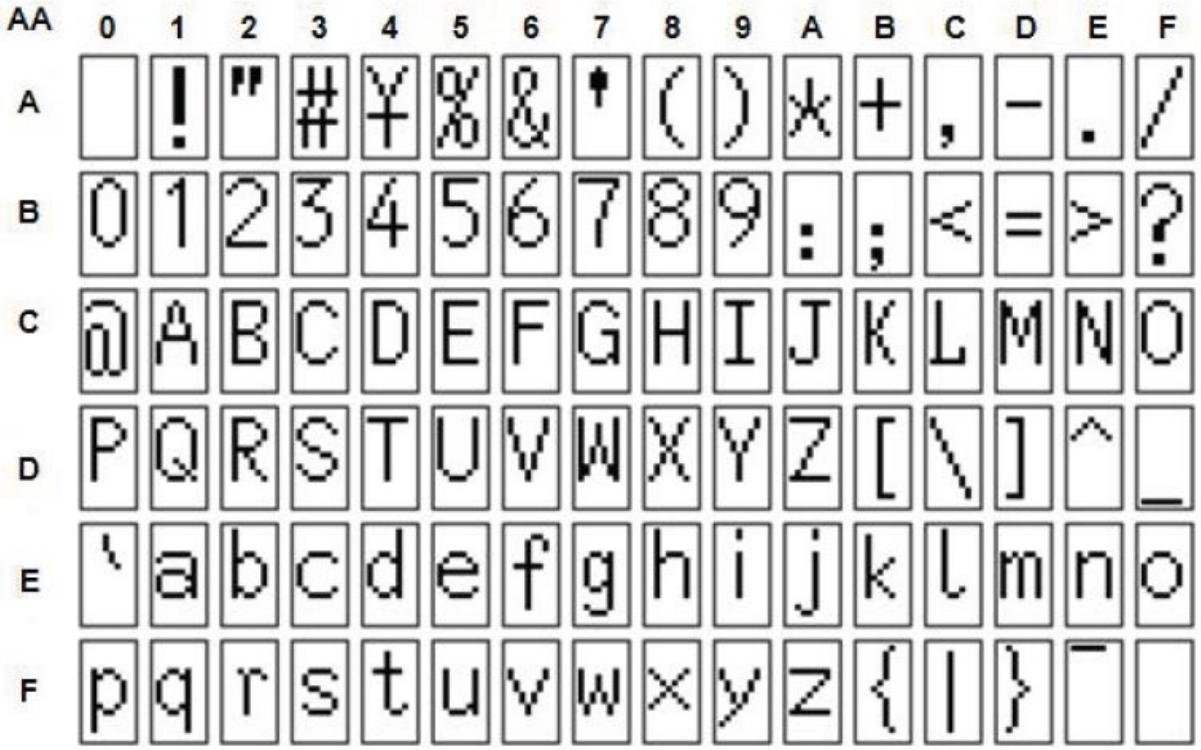
A7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
B	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
C	Ю	Я														
D		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
E	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
F	ю	я														

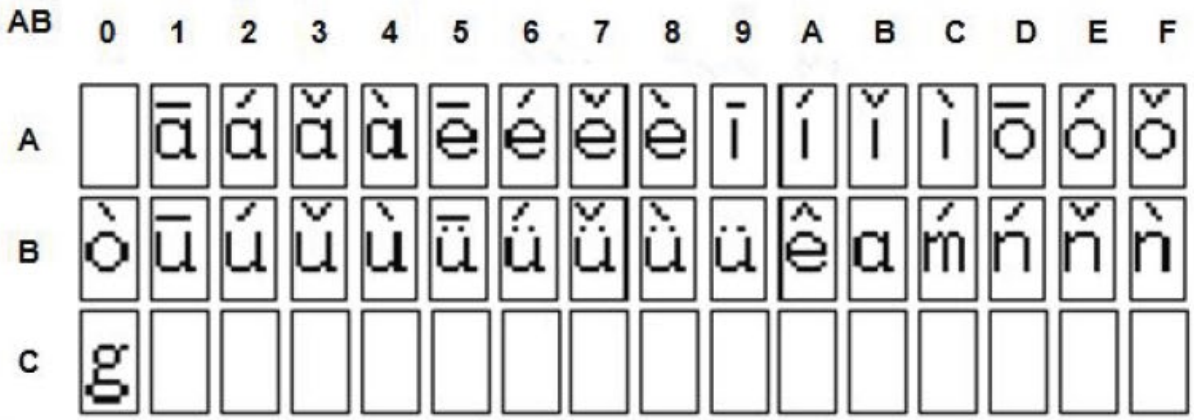
A8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ǖ	ǘ	ǚ	ǜ	ü	ê	ɑ	ḿ	ń	ň	ǹ
C	ɡ					ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ
D	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄚ	ㄛ	ㄜ	ㄝ	ㄞ	ㄟ
E	ㄠ	ㄡ	ㄢ	ㄣ	ㄤ	ㄥ	ㄦ	ㄧ	ㄨ	ㄩ						
F																

A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A					—	━	∣	┃	┄	┅	┆	┇	┈	┉	┊	┋
B	┌	┍	┎	┏	┐	┑	┒	┓	└	┕	┖	┗	┘	┙	┚	┛
C	├	┝	┞	┟	┠	┡	┢	┣	┤	┥	┦	┧	┨	┩	┪	┫
D	┬	┭	┮	┯	┰	┱	┲	┳	┴	┵	┶	┷	┸	┹	┺	┻
E	┼	┽	┾	┿	╀	╁	╂	╃	╄	╅	╆	╇	╈	╉	╊	╋
F																

8.2 8x16 点国际扩展字符（126 字符）

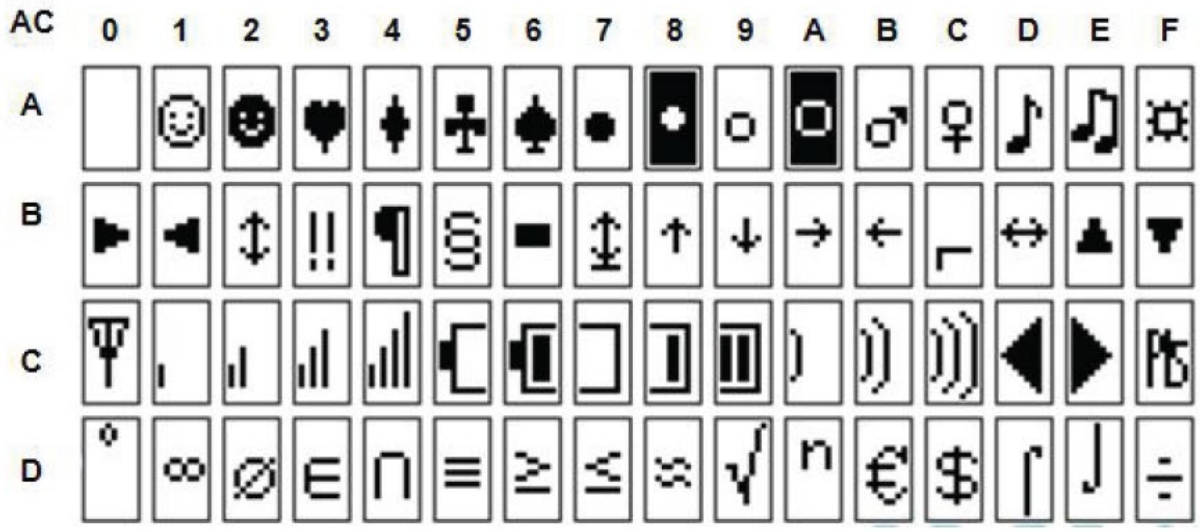
内码组成为 AAA1~ABC0 共计 126 个字符





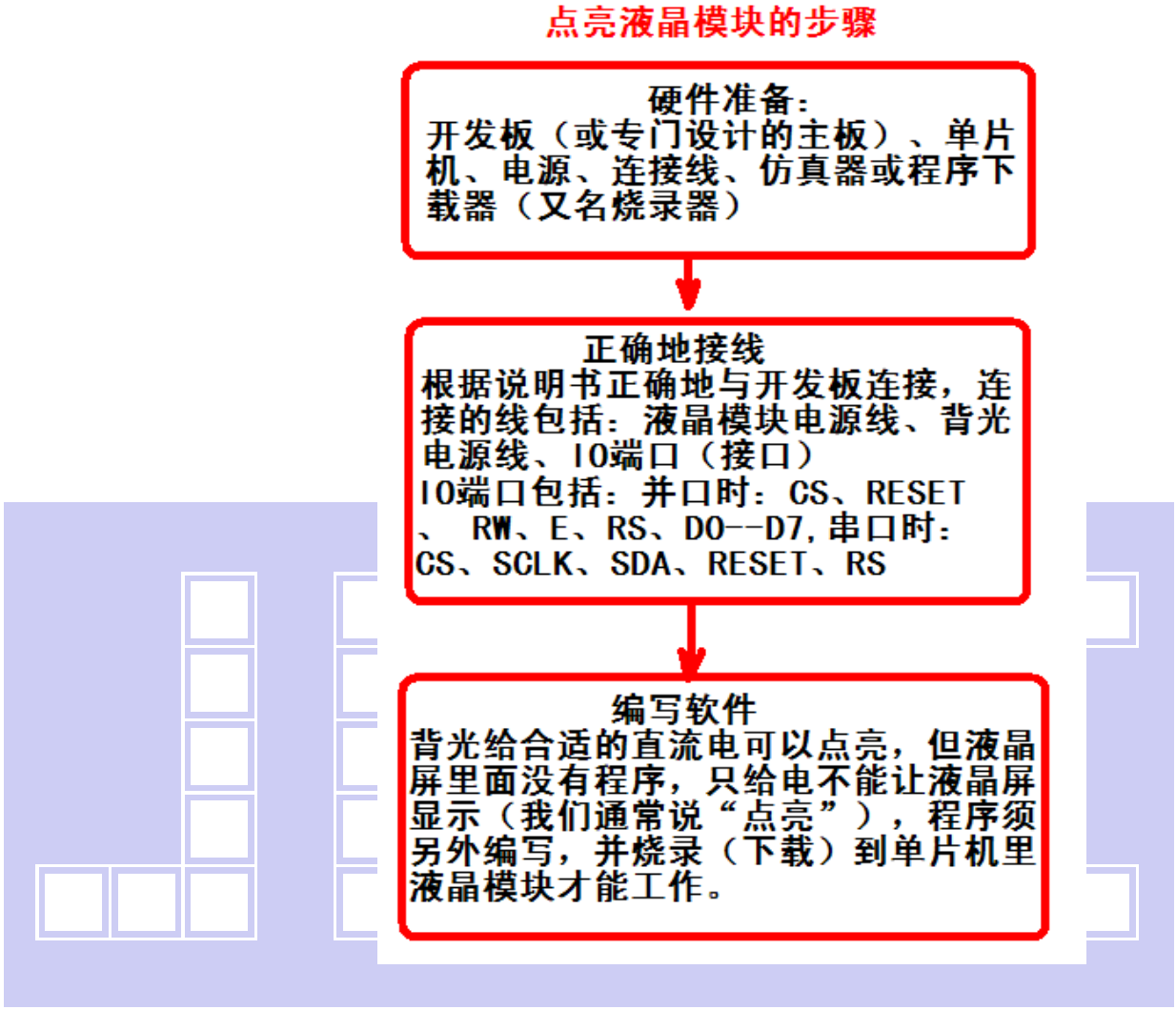
8.3 8x16 点特殊字符（64 字符）

内码组成为 ACA1~ACDF 共计 64 个字符

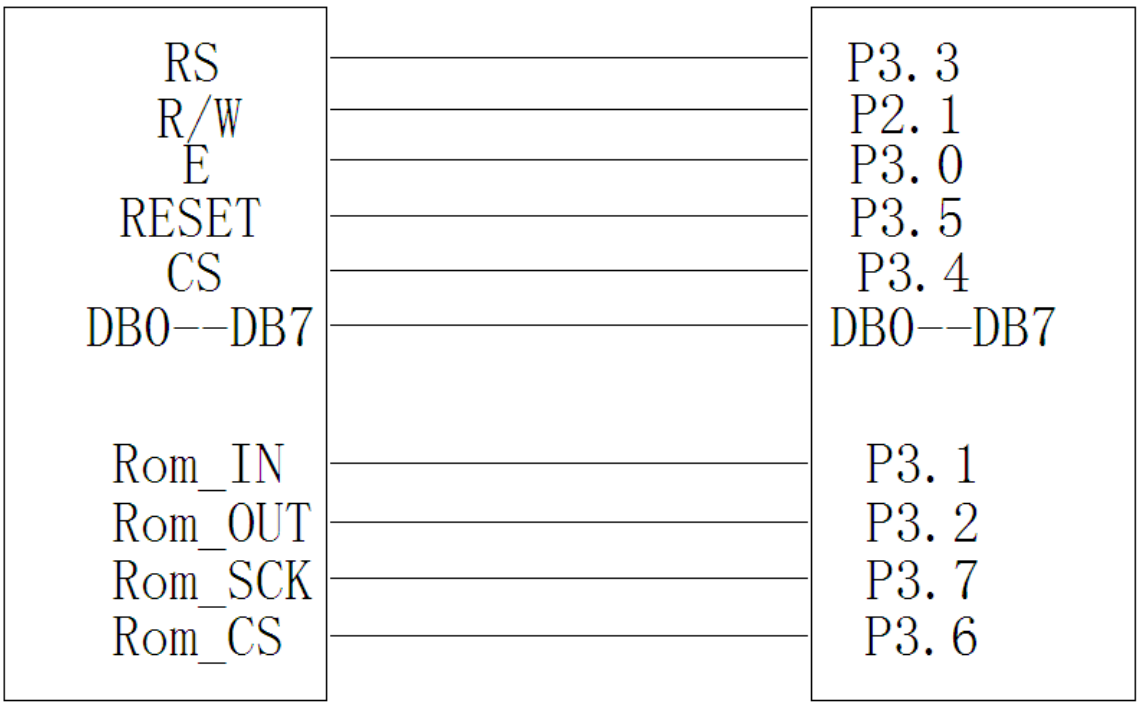


9. 硬件设计及例程：

7.1 当 LCD 驱动 IC 采用并行接口方式时的硬件设计及例程：



9.2 硬件接口： 下图为并行方式的硬件接口：



9.2 并行接口

```
/*   液晶模块型号：JLX25696G-966
     并行接口:6800 时序
     驱动 IC 是:ST75256
     版权所有：晶联讯电子：网址  http://www.jlxlcd.cn;
*/
#include <STC15F2K60S2.H>
#include <intrins.h>
#include <chinese_code.h>

sbit cs1=P3^4;          /*对应 LCD 引脚 CS*/
sbit reset=P3^5;        /*对应 LCD 引脚 RST*/
sbit rs=P3^3;           /*对应 LCD 引脚 RS*/
sbit rd=P3^0;           /*对应 LCD 引脚 RD*/
sbit wr=P2^1;           /*对应 LCD 引脚 WR，另外 P1.0~1.7 对应 DB0~DB7*/

sbit Rom_IN=P3^1;     /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI*/
sbit Rom_OUT=P3^2;    /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 SO*/
sbit Rom_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;          /*按键接口，P2.0 口与 GND 之间接一个按键*/

#define uchar unsigned char
```

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

/*延时：1 毫秒的 i 倍*/
void delay_ms(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<110;k++);
}

/*延时：1us 的 i 倍*/
void delay_us(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<1;k++);
}

/*等待一个按键，我的主板是用 P2.0 与 GND 之间接一个按键*/
void waitkey()
{
 repeat:
    if (key==1) goto repeat;
    else delay_ms(3000);
}

//============transfer command to LCM============
void transfer_command_lcd(int data1)
{
    cs1=0;
    rs=0;
    rd=0;
    delay_us(1);
    wr=0;
    P1=data1;
    rd=1;
    delay_us(1);
    cs1=1;
    rd=0;
}

//----------transfer data to LCM--------------
void transfer_data_lcd(int data1)
{
    cs1=0;
```

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

void initial_lcd()
{
    RST=0;
    delay_ms(100);
    RST=1;
    delay_ms (100);
    transfer_command_lcd(0x30);   //EXT=0
    transfer_command_lcd(0x94);   //Sleep out
    transfer_command_lcd(0x31);   //EXT=1
    transfer_command_lcd(0xD7);   //Autoread disable
    transfer_data_lcd(0X9F);      //
    transfer_command_lcd(0x32);   //Analog SET
    transfer_data_lcd(0x00);        //OSC Frequency adjustment
    transfer_data_lcd(0x01);        //Frequency on booster capacitors->6KHz
    transfer_data_lcd(0x03);        //Bias=1/11
    transfer_command_lcd(0x20);   // Gray Level

    transfer_command_lcd(0x31);   //Analog SET
    transfer_command_lcd(0xf2);   //温度补偿
    transfer_data_lcd(0x1e);      //OSC Frequency adjustment
    transfer_data_lcd(0x28);      //Frequency on booster capacitors->6KHz
    transfer_data_lcd(0x32);      //

    transfer_data_lcd(0x01);
    transfer_data_lcd(0x03);
    transfer_data_lcd(0x05);
    transfer_data_lcd(0x07);
    transfer_data_lcd(0x09);
    transfer_data_lcd(0x0b);
    transfer_data_lcd(0x0d);
    transfer_data_lcd(0x10);
    transfer_data_lcd(0x11);
    transfer_data_lcd(0x13);
    transfer_data_lcd(0x15);
    transfer_data_lcd(0x17);
```

```
    transfer_data_lcd(0x19);
    transfer_data_lcd(0x1b);
    transfer_data_lcd(0x1d);
    transfer_data_lcd(0x1f);
    transfer_command_lcd(0x30);   //EXT=0
    transfer_command_lcd(0x75);   //Page Address setting
    transfer_data_lcd(0X00);      // XS=0
    transfer_data_lcd(0X14);      // XE=159 0x28
    transfer_command_lcd(0x15);   //Clumn Address setting
    transfer_data_lcd(0X00);      // XS=0
    transfer_data_lcd(0Xff);      // XE=256
    transfer_command_lcd(0xBC);    //Data scan direction
    transfer_data_lcd(0x00);         //MX.MY=Normal
    transfer_data_lcd(0xA6);
    transfer_command_lcd(0xCA);    //Display Control
    transfer_data_lcd(0X00);         //
    transfer_data_lcd(0X9F);         //Duty=160
    transfer_data_lcd(0X20);         //Nline=off
    transfer_command_lcd(0xF0);    //Display Mode
    transfer_data_lcd(0X10);         //10=Monochrome Mode,11=4Gray
    transfer_command_lcd(0x81);    //EV control
    transfer_data_lcd(0x0a);         //VPR[5-0]  //可设置范围 0x00~0x3f,每格电压是 0.04V
    transfer_data_lcd(0x04);         //VPR[8-6]    //可设置范围 0x00~0x07
    transfer_command_lcd(0x20);    //Power control
    transfer_data_lcd(0x0B);         //D0=regulator ; D1=follower ; D3=booste,  on:1 off:0
    delay_us(100);
    transfer_command_lcd(0xAF);    //Display on
}
```

```
/*写 LCD 行列地址：X 为起始的列地址，Y 为起始的行地址，x_total,y_total 分别为列地址及行地址的起点到终点的差值 */
void lcd_address(int x,int y,x_total,y_total)
{
    x=x-1;
    y=y+7;

    transfer_command_lcd(0x15); //Set Column Address
    transfer_data_lcd(x);
    transfer_data_lcd(x+x_total-1);

    transfer_command_lcd(0x75); //Set Page Address
    transfer_data_lcd(y);
    transfer_data_lcd(y+y_total-1);
    transfer_command_lcd(0x30);
    transfer_command_lcd(0x5c);
```

```
}

/*清屏*/
void clear_screen()
{
    int i,j;
    lcd_address(1,1,256,17);
    for(i=0;i<17;i++)
    {
        for(j=0;j<256;j++)
        {
            transfer_data_lcd(0x00);
        }
    }
}

/****送指令到晶联讯字库 IC***/
void send_command_to_ROM( uchar datu )
{
    uchar i;
    for(i=0;i<8;i++ )
    {
        if(datu&0x80)
            Rom_IN = 1;
        else
            Rom_IN = 0;
        datu = datu<<1;
        Rom_SCK=0;
        Rom_SCK=1;
        delay_us(1);
    }
}

/****从晶联讯字库 IC 中取汉字或字符数据（1 个字节）***/
static uchar get_data_from_ROM( )
{
    uchar i;
    uchar ret_data=0;
    Rom_SCK=1;
    for(i=0;i<8;i++)
    {
        Rom_OUT=1;
        Rom_SCK=0;
        ret_data>>=1;
        if( Rom_OUT )
            ret_data+=0x80;
```

```
        else
            ret_data=ret_data+0;
        Rom_SCK=1;
        delay_us(1);
    }

    return(ret_data);
}


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

void get_and_write_12x24(ulong fontaddr,uchar column,uchar page)
{
    uchar i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                          //地址的低 8 位,共 24 位
    lcd_address(column,page,16,3);
    for(j=0;j<3;j++)
    {
        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;
}

void get_and_write_16x32(ulong fontaddr,uchar column,uchar page)
{
    uchar i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
    lcd_address(column,page,16,4);
    for(j=0;j<4;j++)
    {
        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_16x16(ulong fontaddr,uchar column,uchar page)
{
    uchar i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
    lcd_address(column,page,16,2);
    for(j=0;j<2;j++)
    {
        for(i=0; i<16; i++ )
      {
            disp_data=get_data_from_ROM();
            transfer_data_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_24x24(ulong fontaddr,uchar column,uchar page)
{
    uchar i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
    lcd_address(column,page,24,3);
    for(j=0;j<3;j++)
    {
        for(i=0; i<24; 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_32x32(ulong fontaddr,uchar column,uchar page)
{
    uchar i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
    lcd_address(column,page,32,4);
    for(j=0;j<4;j++)
    {
        for(i=0; i<32; 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_16x16_string(uchar column,uchar page,uchar *text)
{
    uchar i= 0;uint temp1,temp2;
    temp1=column;
    temp2=page;
    while((text[i]>0x00))
    {
        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);
            get_and_write_16x16(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=16;
            if ((temp2<=15&&temp1<=256)&&column>248)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>256 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
                page+=2;
                if (page>15)column=1;
            }
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xa3))&&(text[i+1]>=0xa1))
        {
            //国标简体（GB2312）15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);
            get_and_write_16x16(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=16;

            if ((temp2<=15&&temp1<=256)&&column>248)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>128 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
```

```
                page+=2;
                if (page>15)column=1;
            }
        }

        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long)(fontaddr*16);
            fontaddr = (unsigned long)(fontaddr+0x1dd780);

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

            if ((temp1<=15&&temp2<=256)&&column>248)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>128 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
                page+=2;
                if (page>15)column=1;
            }
        }
        else
        i++;
    }
}

//*****************************************************************
void display_GB2312_24x24_string(uchar column,uchar page,uchar *text)
{
    uchar i= 0;uint temp1,temp2;
    temp1=column;
    temp2=page;

    while((text[i]>0x00))
    {
        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*72);
```

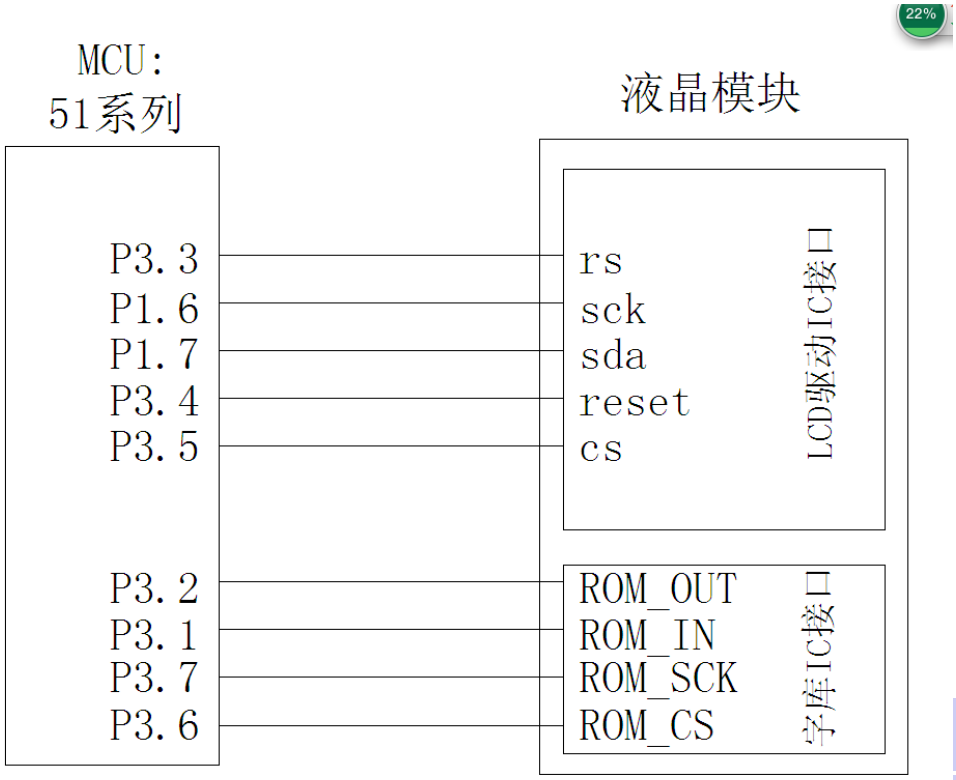
```
            fontaddr = (ulong)(fontaddr+0X068190);

            get_and_write_24x24(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=24;
        }

        else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(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*72);
            fontaddr = (ulong)(fontaddr+0X068190);

            get_and_write_24x24(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=24;
        }

        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long)(fontaddr*48);
            fontaddr = (unsigned long)(fontaddr+0x1dff00);
            get_and_write_12x24(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=1;
            column+=12;
        }
        else
        i++;
    }
}

//******************************************************************
void display_GB2312_32x32_string(uchar column,uchar page,uchar *text)
{
    uchar i= 0;uint temp1,temp2;
    temp1=column;
    temp2=page;

    while((text[i]>0x00))
    {
        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
```

```
        {
            //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0xedf00);

            get_and_write_32x32(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=32;
        }

        else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(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*128);
            fontaddr = (ulong)(fontaddr+0xedf00);

            get_and_write_32x32(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=32;
        }

        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long)(fontaddr*64);
            fontaddr = (unsigned long)(fontaddr+0x1e5a50);

            get_and_write_16x32(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=1;
            column+=16;
        }
        else
        i++;
    }
}

//-------------------------
void main()
```

```
{
    P1M1=0x00;
    P1M0=0x00;   //P1 配置为准双向
    P2M1=0x00;
    P2M0=0x00;   //P2 配置为准双向
    P3M1=0x00;
    P3M0=0x00;   //P3 配置为准双向
    initial_lcd();              //对液晶模块进行初始化设置
    while(1)
    {
        clear_screen();
        display_GB2312_16x16_string(1,1,1,"GB2312 简体字 16X16、24X24、32X32");
        display_GB2312_16x16_string(1,3,0,"简体汉字库及 8X16、12X24、16X32");
        display_GB2312_16x16_string(1,5,0,"的 ASCII 码(1)①○●◎◇◆ 1.2.‖…");
        display_GB2312_24x24_string(8,7,0,"啊阿埃挨森淼焱皑鑫蔼");
        display_GB2312_24x24_string(1,10,0,"ABCDEFGHIJKLMNOPQRSTU");
        waitkey();
        clear_screen();
        display_GB2312_32x32_string(1,1,0,"鑫森淼焱晶磊众品");
        display_GB2312_32x32_string(1,5,1,"鬓鬟麽麾縻麂麋麈");
        display_GB2312_32x32_string(1,9,0,"麋麒麈麝麟黛黜黝");
        waitkey();
        clear_screen();
        display_GB2312_32x32_string(1,1,0," 《望庐山瀑布》");
        display_GB2312_32x32_string(1,5,0,"日照香炉生紫烟，");
        display_GB2312_32x32_string(1,9,0,"遥看瀑布挂前川。");
        waitkey();
        clear_screen();             //清屏
        disp_256x96(1,1,bmp1);              //显示一幅 256*96 点阵的黑白图。
        waitkey();
        disp_256x96(1,1,bmp2);              //显示一幅 256*96 点阵的黑白图。
        waitkey();    }
}
```

9.3 当 LCD 驱动 IC 采用串行接口方式时的硬件设计及例程：

9.3.1 硬件接口：下图为串行方式的硬件接口：



9.3.2、以下为串行接口方式范例程序

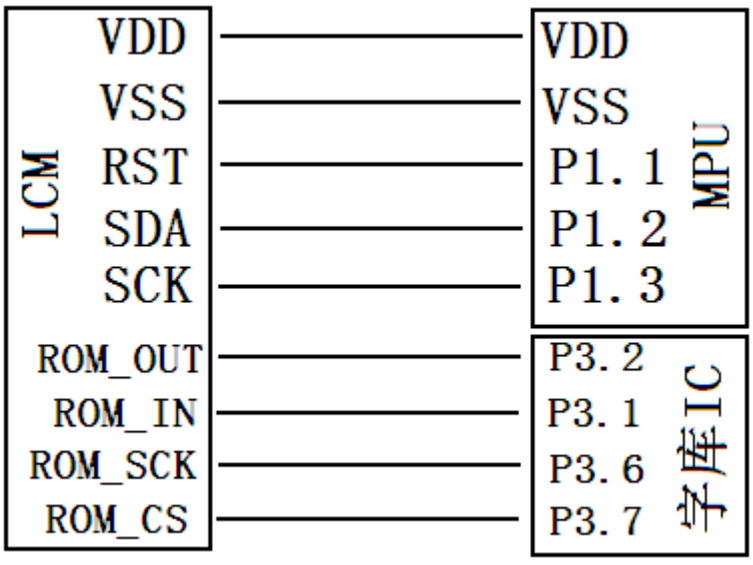
与并行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

```
#include <reg51.H>
#include <intrins.h>
#include <Ctype.h>

 sbit CS=P3^5;       /*3.4 接口定义*/
 sbit reset=P3^4;    /*3.3 接口定义*/
 sbit RS=P3^0;       /*接口定义*/
 sbit SCK=P1^6;      /*接口定义*/
 sbit SDA=P1^7;      /*接口定义。另外 P1.0~1.7 对应 DB0~DB7*/
 sbit key=P2^0;      /*按键接口，P2.0 口与 GND 之间接一个按键*/
 sbit Rom_IN=P3^1;   /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI*/
 sbit Rom_OUT=P3^2;  /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 SO*/
 sbit Rom_SCK=P3^7;  /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
 sbit Rom_CS=P3^6;   /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#*/

 //=============transfer command to LCM===============
 void transfer_command_lcd(int data1)
 {
     char i;
     CS=0;
     RS=0;
     for(i=0;i<8;i++)
     {
         SCK=0;
```

```
            if(data1&0x80) SDA=1;
            else SDA=0;
            SCK=1;
//     delay_us(1);
            data1=data1<<=1;
        }
}

//-----------transfer data to LCM---------------
void transfer_data_lcd(int data1)
{
     char i;
     CS=0;
     RS=1;
     for(i=0;i<8;i++)
     {
          SCK=0;
          if(data1&0x80) SDA=1;
          else SDA=0;
          SCK=1;
//         delay_us(1);
          data1=data1<<=1;
     }
}
```

9.4 当 LCD 驱动 IC 采用 IIC 接口方式时的硬件设计及例程：

9.4.1 硬件接口：下图为 IIC 方式的硬件接口：



9.4.2、以下为 IIC 接口方式范例程序

与串行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

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

sbit  reset=P1^1;
sbit scl=P1^3;
sbit sda=P1^2;
sbit Rom_IN=P3^1;       /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI*/
sbit Rom_OUT=P3^2;      /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 SO*/
sbit Rom_SCK=P3^7;      /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
sbit Rom_CS=P3^6;       /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#*/
sbit  key=P2^0;


#define uchar unsigned char
#define uint unsigned int

void transfer(int data1)
{
    int i;
    for(i=0;i<8;i++)
   {
        scl=0;
        if(data1&0x80) sda=1;
        else sda=0;
        scl=1;
        scl=0;
        data1=data1<<1;
   }
        sda=0;
        scl=1;
        scl=0;
}

void start_flag()
{
    scl=1;        /*START FLAG*/
    sda=1;        /*START FLAG*/
    sda=0;        /*START FLAG*/
}

void stop_flag()
{
    scl=1;        /*STOP FLAG*/
    sda=0;        /*STOP FLAG*/
```

```
    sda=1;          /*STOP FLAG*/
}

//写命令到液晶显示模块
void transfer_command(uchar com)
{
    start_flag();
    transfer(0x78);
    transfer(0x80);
    transfer(com);
    stop_flag();
}

//写数据到液晶显示模块
void transfer_data(uchar dat)
{
    start_flag();
    transfer(0x78);
    transfer(0xC0);
    transfer(dat);
    stop_flag();
}
```