JLX128128G-338-PC

带字库 IC 的编程说明书

目 录

1．概述

JLX128128G-338-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-GB2312-1602 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。

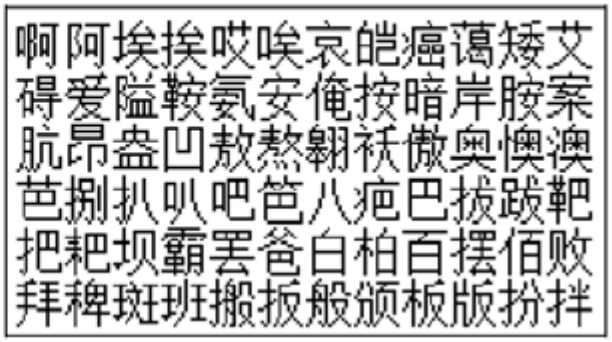
此字库 IC 存储内容如下表所述：

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

2．字型样张：

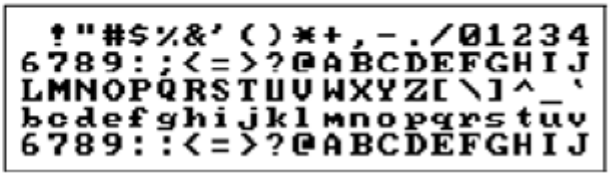
11X12 点 GB2312 汉字

15X16 点 GB2312 汉字



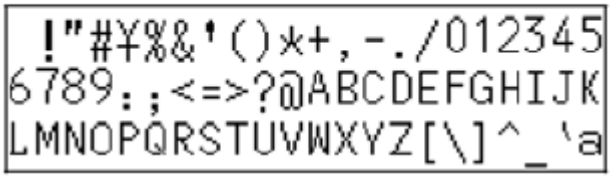
5x7 点 ASCII 字符

7x8 点 ASCII 字符



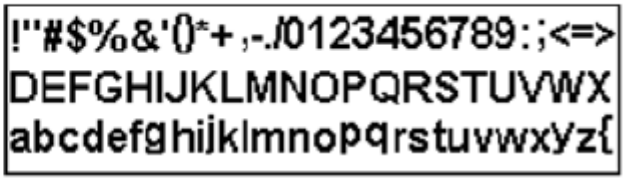
6x12 点 ASCII 字符

8x16 点 ASCII 字符



12 点阵不等宽 ASCII 方头

16 点阵不等宽 ASCII 方头



3. 外形尺寸及接口引脚功能

3.1 外形图：

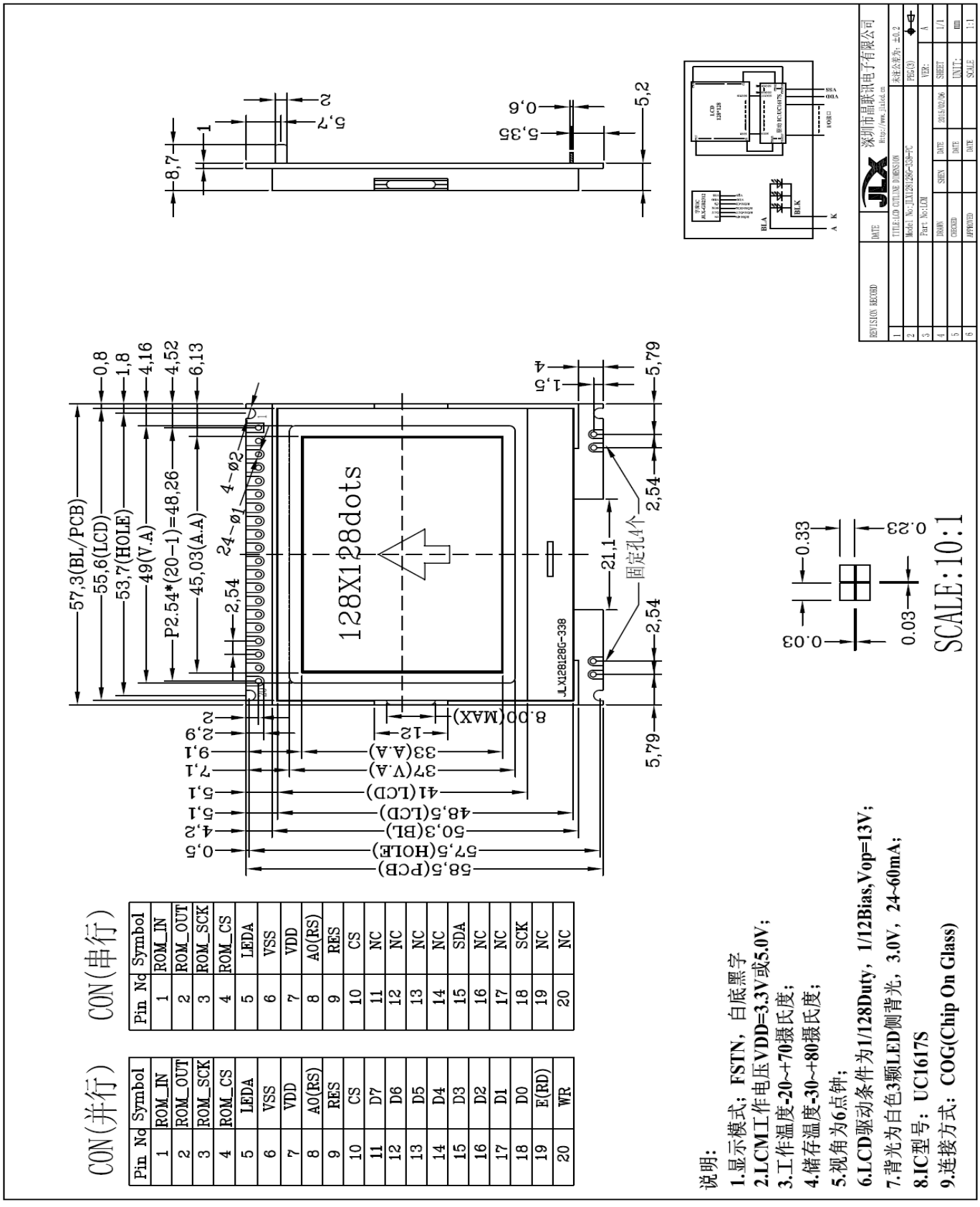


图 1. 外形尺寸

3.2 模块的接口引脚功能

3.2.1 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	1.当选择带字库的产品，请参阅： (1)字库 IC:JLX-GB2312 说明书 (2)JLX128128G-338-PC 的中文字库编程说明书 2.当不用字库时为空
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	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,请注意 11 脚是高位 D7,18 脚是低位 D0	
19	E(RD)	使能信号（读）	6800 时序：使能信号 （8080 时序时：读）	
20	RW(WR)	读/写（写）	6800 时序：H:读数据 0:写数据 （8080 时序时：写）	

表 1：模块并行接口引脚功能

3.2.2 串行接口引脚功能

引线号	符 号	名 称	功 能	
1	ROM-IN	即字库 IC 接口(SI)	串行数据输入	1.当选择带字库的产品，请参阅： (1)字库 IC:JLX-GB2312 说明书 (2)JLX128128G-338-PC 的中文字库编程说明书 2.当不用字库时为空
2	ROM-OUT	即字库 IC 接口(SO)	串行数据输出	
3	ROM-SCK	即字库 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	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“CD”）	
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	NC	空		
12	NC	空		
13	NC	空		
14	NC	空		
15	SDA(D3)	数据	串行数据	
16	NC	空		
17	NC	空		
18	SCK(D0)	时钟	串行时钟	
19	空	空		
20	空	空		

表 2：4 线 SPI 串行接口引脚功能

3.2.3 I²C 总线时接口引脚功能

引线号	符 号	名 称	功 能	
1	ROM-IN	即字库 IC 接口(SI)	串行数据输入	1.当选择带字库的产品，请参阅： (1)字库 IC:JLX-GB2312 说明书 (2)JLX128128G-338-PC 的中文字库编程说明书 2.当不用字库时为空
2	ROM-OUT	即字库 IC 接口(SO)	串行数据输出	
3	ROM-SCK	即字库 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	NC	空		
9	NC	空		
10	NC	空		
11	NC	空		
12	NC	空		
13	NC	空		
14	NC	空		
15	SDA(D3)	数据	串行数据	
16	NC	空		
17	NC	空		
18	SCK(D0)	时钟	串行时钟	
19	NC	空		
20	NC	空		

表 3：I²C 总线接口引脚功能

4. 工作电路框图：

见图 2，模块由 LCD 驱动 IC UC11617S、字库 IC、背光组成。

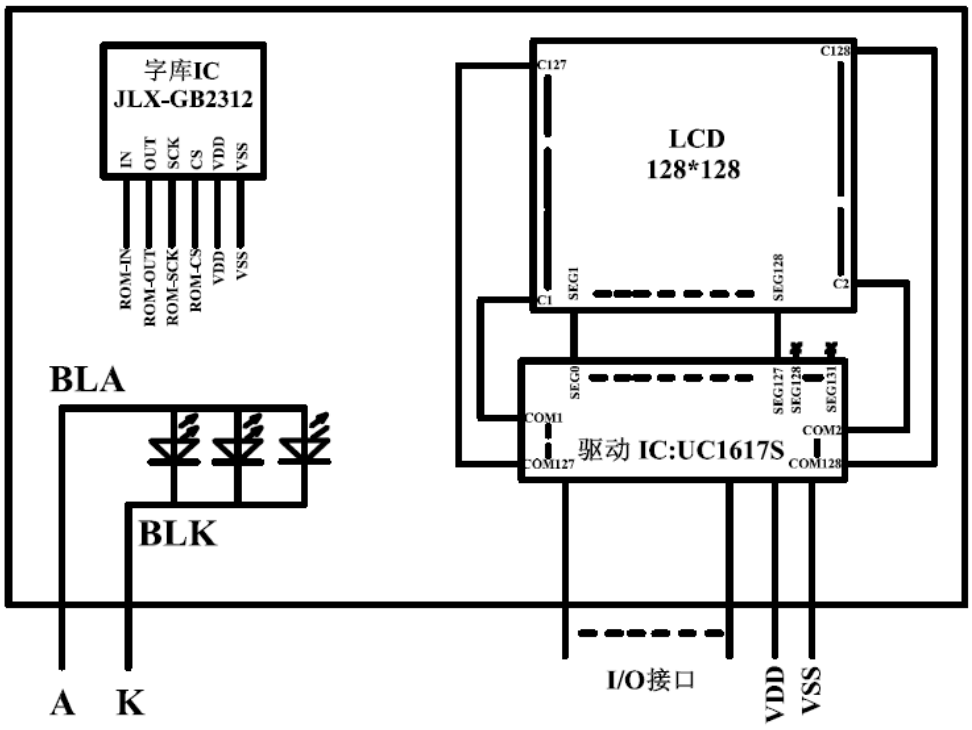


图 2:电路框图

5. 指令：

5.1 字库 IC（JLX-GB2312-16S2W）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)：

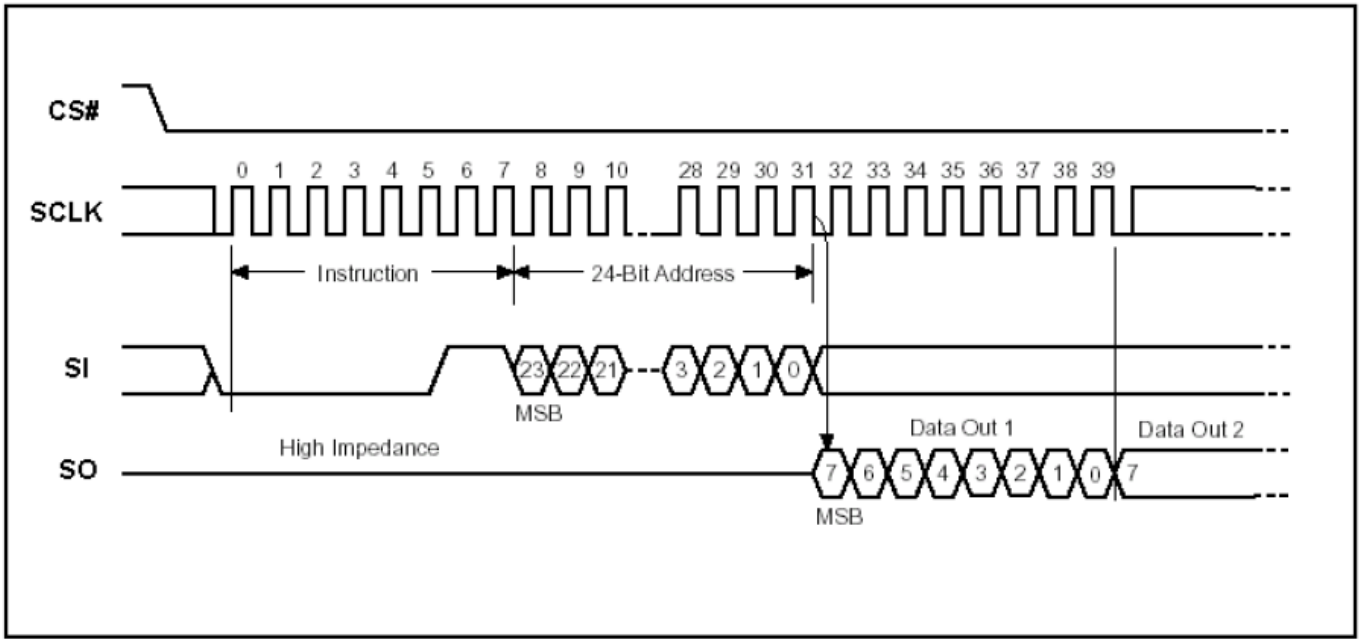
■首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（03 h）和 3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

■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)：

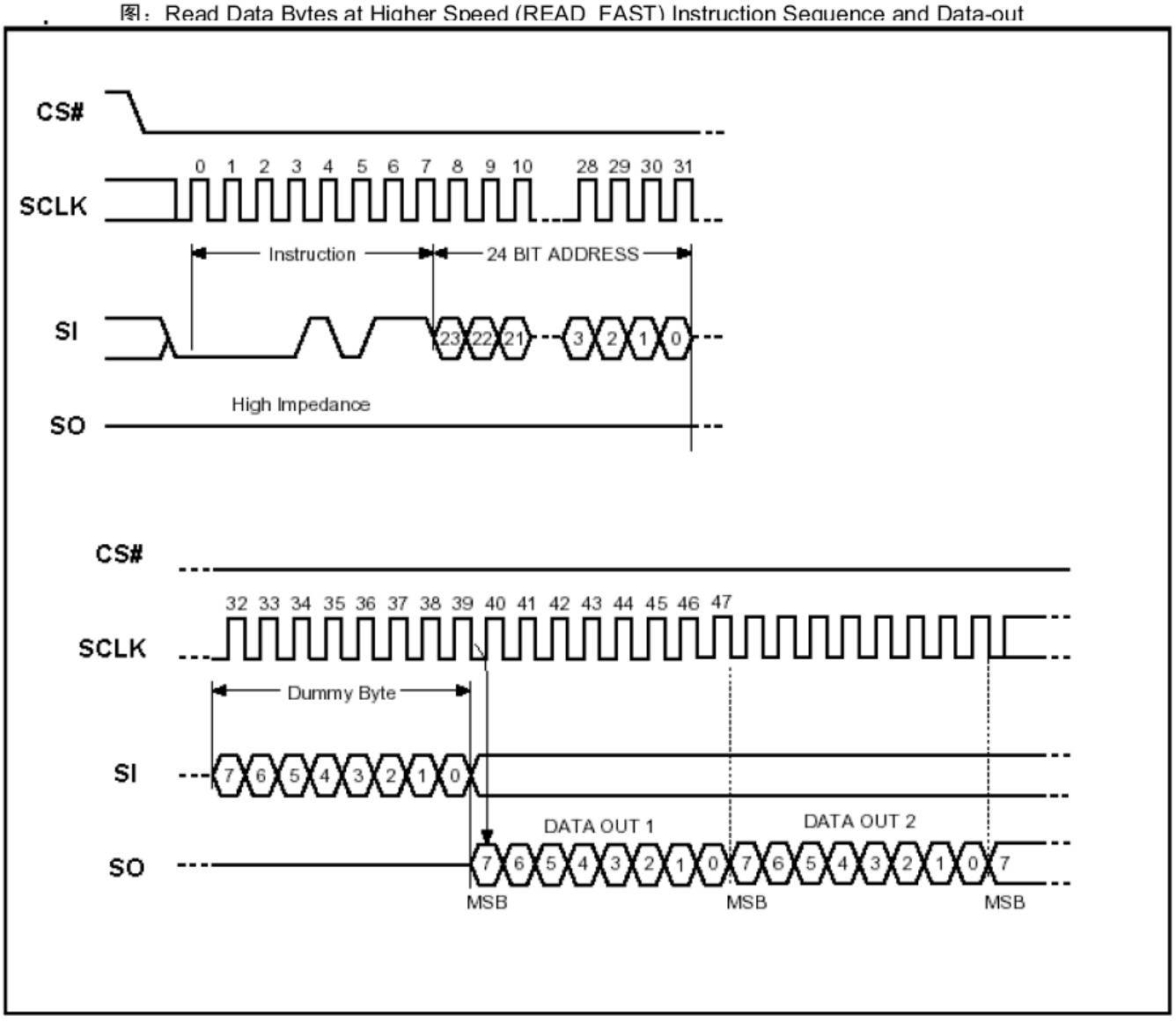
■首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（0B h）和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

■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



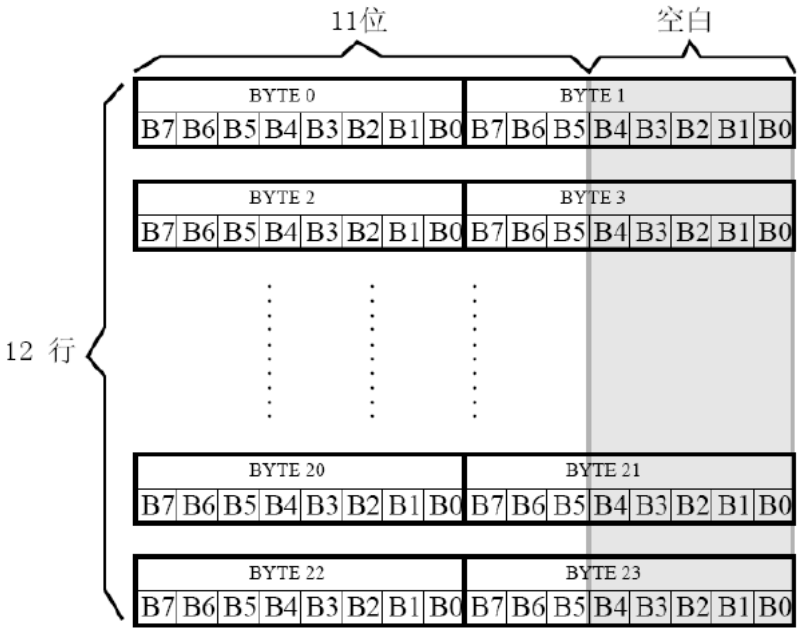
6 字库调用方法

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，排满一行的点后再排下一行。这样把点阵信息用来直接在显示器上按上述规则显示，则将出现对应的汉字。

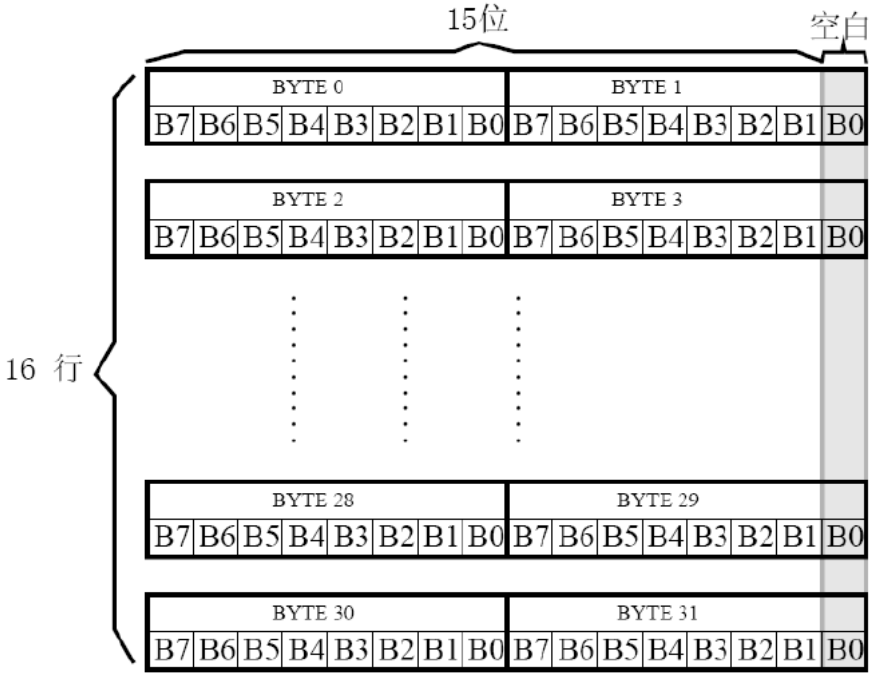
6.1.1 11X12 点汉字排列格式

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



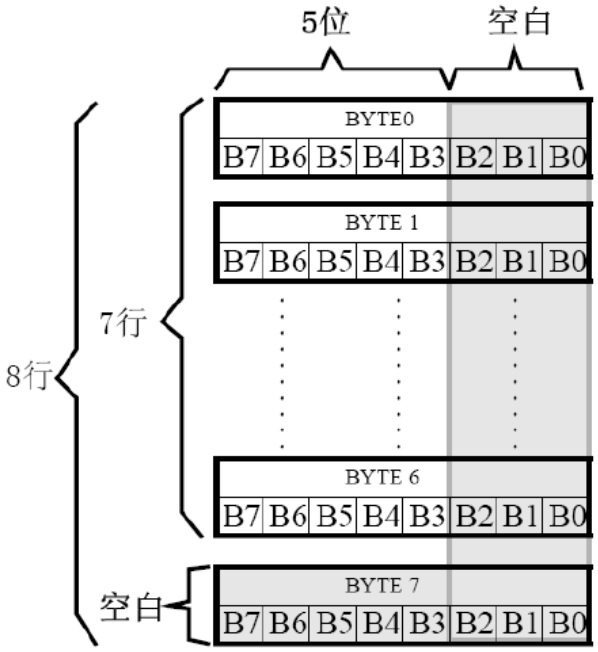
6.1.2 15X16 点汉字排列格式

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



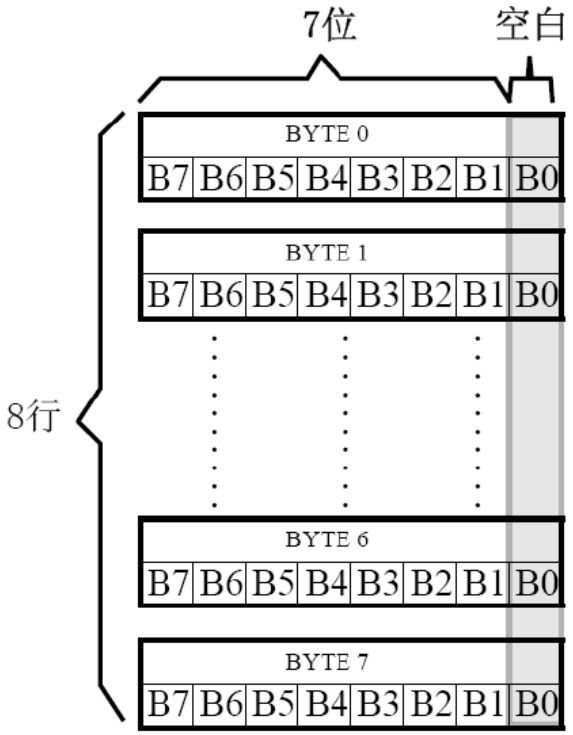
6.1.3 5X7 点ASCII 字符排列格式

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



6.1.4 7X8 点ASCII 字符排列格式

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



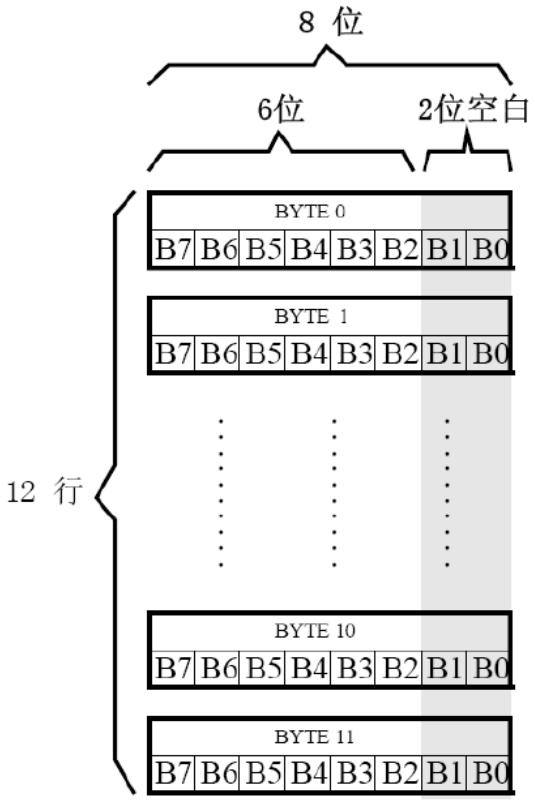
6.1.5 6X12 点字符排列格式

适用于此种排列格式的字体有：

6X12 点ASCII 字符

6X12 点国标扩展字符

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



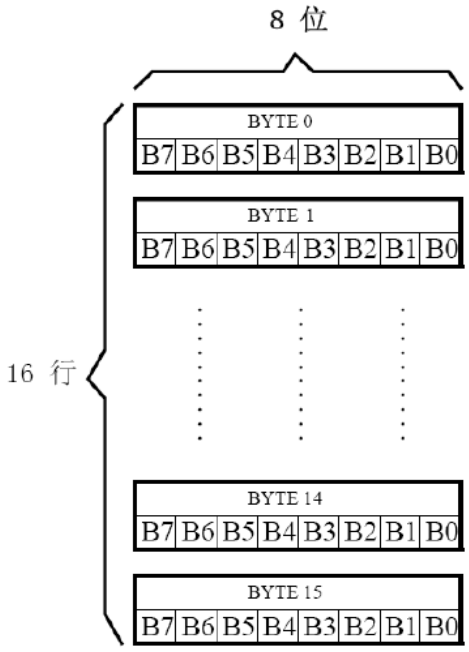
6.1.6 8X16 点字符排列格式

适用于此种排列格式的字体有：

8X16 点ASCII 字符

8X16 点国标扩展字符

8X16 点字符信息需要16 个字节（BYTE 0 – BYTE15）来表示。该点阵数据是横置横排的，其具体排列结构如下图：



6.1.7 12 点阵不等宽ASCII 方头（Arial）字符排列格式

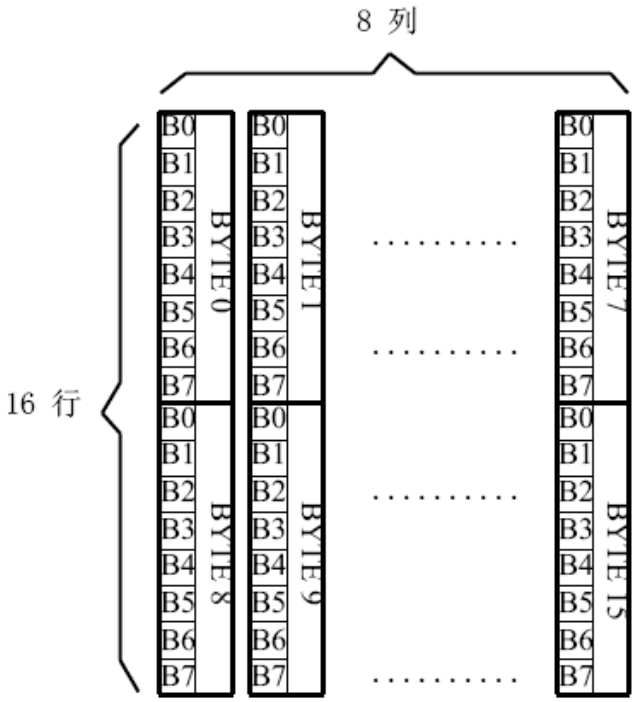
12 点阵不等宽字符的信息需要26 个字节（BYTE 0 – BYTE25）来表示。

由于字符是不等宽的，因此在存储格式中BYTE0~ BYTE1 存放点阵宽度数据，BYTE2-25 存放横置横排点阵数据。

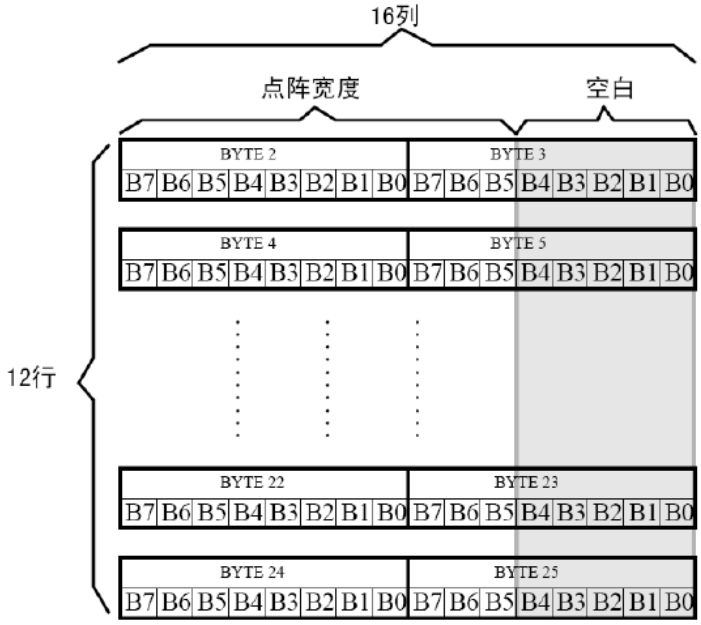
不等宽字符的点阵存储宽度是以BYTE 为单位取整的，根据不同字符宽度会出现相应的空白区。根据

6.1.4 8X16 点字符排列格式

适用于此种排列格式的字体有：
8X16 点 ASCII 字符
8X16 点 ASCII 粗体字符
8X16 点国标扩展字符
8X16 点字符信息需要 16 个字节（BYTE 0 - 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：



BYTE0~ BYTE1 所存放点阵的实际宽度数据，可以对还原下一个字的显示或排版留作参考。

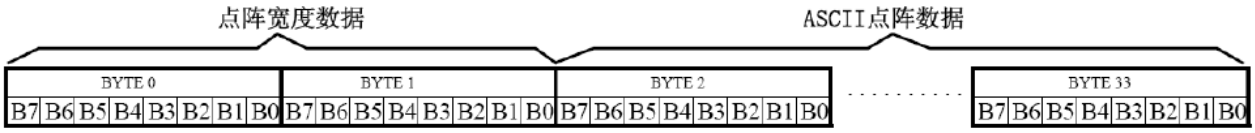


6.1.8 16 点阵不等宽ASCII 方头（Arial）字符排列格式

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

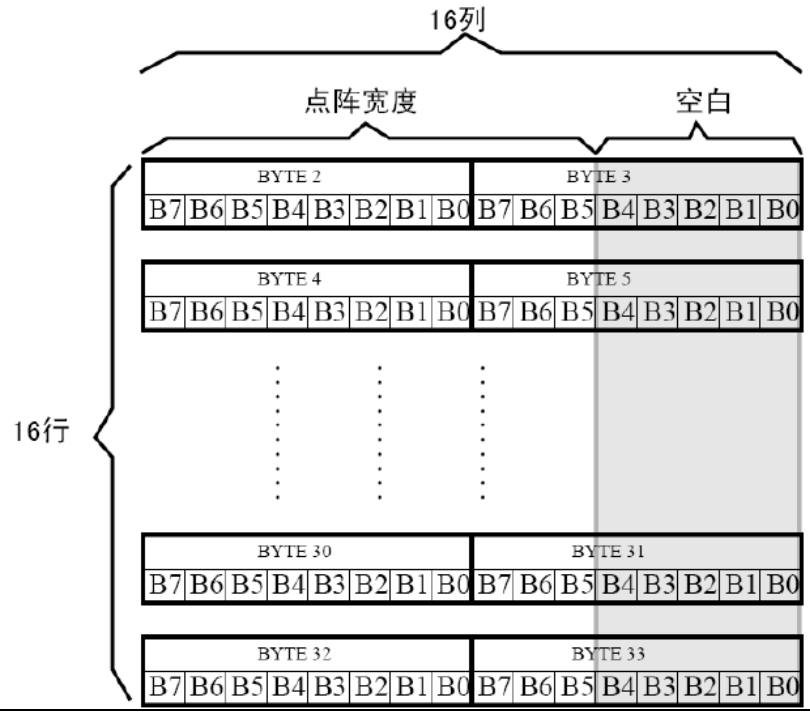
▮ 存储格式

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



▮ 存储结构

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



例如：ASCII 方头字符B

0-33BYTE 的点阵数据是： 00 0C 00 00 00 00 00 00 7F 80 7F C0 60 C0 60 C0 60 C0 7F 80 7F C0 60 E0 60 60 60 60 7F C0 7F 80 00 00

其中：

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

BYTE2-33: 00 00 00 00 00 00 7F 80 7F C0 60 C0 60 C0 60 C0 7F 80 7F C0 60 E0 60 60 60 60 7F C0 7F 80 00 00 为ASCII 方头字符B 的点阵数据。

6.2 汉字点阵字库地址表

	字库内容	编码体系	码位范围	字符数	起始地址	结束地址	参考算法
1	15X16 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+376	00000	3B7BF	6.3.1.2
2	GB2312 到 Unicode 内码转换表				2F00	66BF	6.4
3	7X8 点 ASCII 字符	ASCII	20~7F	96	66C0	69BF	6.3.2.2
4	8X16 点国标扩展字符	GB2312	AAA1-ABC0	126	3B7D0	3BFBF	6.3.1.4
5	8X16 点 ASCII 字符	ASCII	20~7F	96	3B7C0	3BFBF	6.3.2.4
6	5X7 点 ASCII 字符	ASCII	20~7F	96	3BFC0	3C2BF	6.3.2.1
7	16 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F	96	3C2C0	3CF7F	6.3.2.6
8	11X12 点 GB2312 标准点阵字库	GB2312	A1A1-F7FE	6763+376	3CF80	66D3F	6.3.1.3
9	6X12 点国标扩展字符	GB2312	AAA1-ABC0	126	66D4C	6733F	6.3.1.3
10	6X12 点 ASCII 字符	ASCII	20~7F	96	66D40	6733F	6.3.2.3
11	12 点阵不等宽 ASCII 方头（Arial）字符	ASCII	20~7F	96	67340	67CFF	6.3.2.5
12	保留区				67D00	67D6F	
13	Unicode 到 GB2312 内码转换表				67D70	7278F	6.5
14	GT 快捷拼音输入法码表				72790	7FA33	
15	保留区				7FA33	7FFFF	

6.3 字符在芯片中的地址计算方法

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

6.3.1 汉字字符的地址计算

6.3.1.1 11X12 点GB2312 标准点阵字库

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0x3cf80;
if(MSB >=0xA1 && MSB <= 0Xa3 && LSB >=0xA1)
Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*24+ BaseAdd;
else if(MSB == 0Xa9 && LSB >=0xA1)
Address =( (282 + (LSB - 0xA1))*24+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 376)*24+ BaseAdd;
```

6.3.1.2 15X16 点GB2312 标准点阵字库

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd：说明点阵数据在字库芯片中的起始地址。

计算方法：

```
BaseAdd=0;
if(MSB == 0xA9 && LSB >=0xA1)
Address =( (282 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB >=0xA1 && MSB <= 0xA3 && LSB >=0xA1)
Address =( (MSB - 0xA1) * 94 + (LSB - 0xA1))*32+ BaseAdd;
else if(MSB >=0xB0 && MSB <= 0xF7 && LSB >=0xA1)
Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;
```

6.3.1.3 6X12 点国标扩展字符

说明：

BaseAdd：说明本套字库在字库芯片中的起始字节地址。

FontCode：表示字符内码（16bits）

ByteAddress：表示字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x66d4c

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

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

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

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

6.3.1.4 8X16 点国标扩展字符

说明：

BaseAdd：说明本套字库在字库芯片中的起始字节地址。

FontCode：表示字符内码（16bits）

ByteAddress：表示字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x3b7d0

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

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

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

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

6.3.2 ASCII 字符的地址计算

6.3.2.1 5X7 点**ASCII** 字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x3bfc0

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

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

6.3.2.2 7X8 点**ASCII** 字符

参数说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x66c0

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

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

6.3.2.3 6X12 点 **ASCII** 字符

说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：BaseAdd=0x66d40

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

6.3.2.4 8X16 点 ASCII 字符

说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x3b7c0

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

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

6.3.2.5 12 点阵不等宽ASCII 方头（Arial）字符

说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x67340

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

Address = (ASCIICode –0x20) * 26 + BaseAdd

6.3.2.6 16 点阵不等宽ASCII 方头（Arial）字符

说明：

ASCIICode：表示 ASCII 码（8bits）

BaseAdd：说明该套字库在芯片中的起始地址。

Address：ASCII 字符点阵在芯片中的字节地址。

计算方法：

BaseAdd=0x3c2c0

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

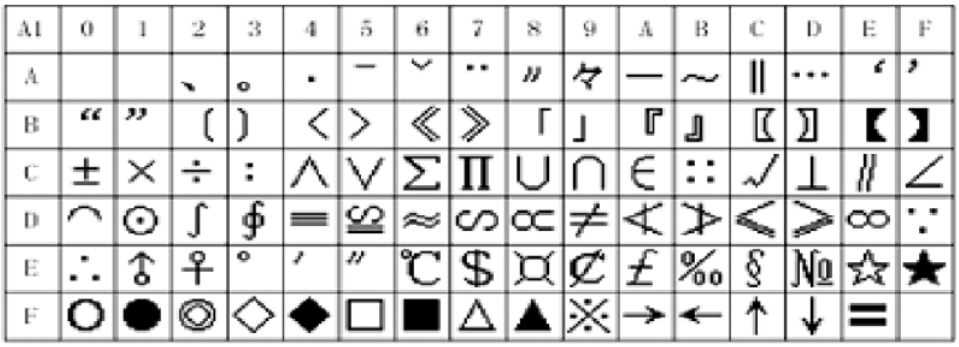
Address = (ASCIICode –0x20) * 34 + BaseAdd

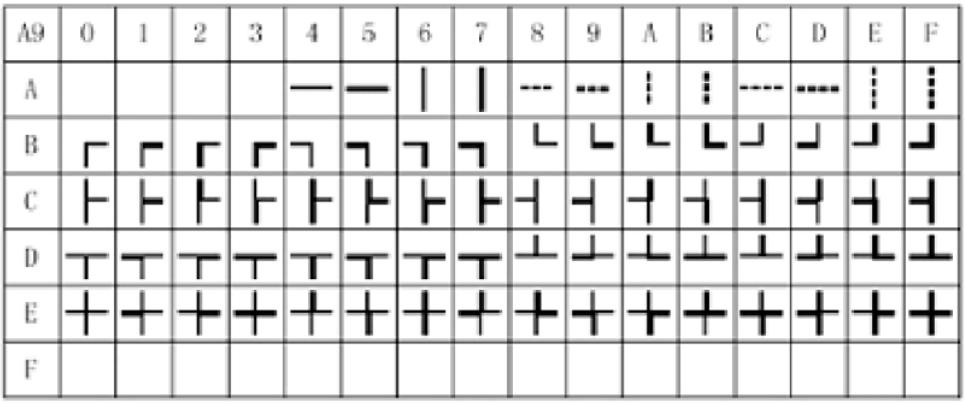
6.4 附录

6.4.1 GB2312 1 区（376 字符）

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

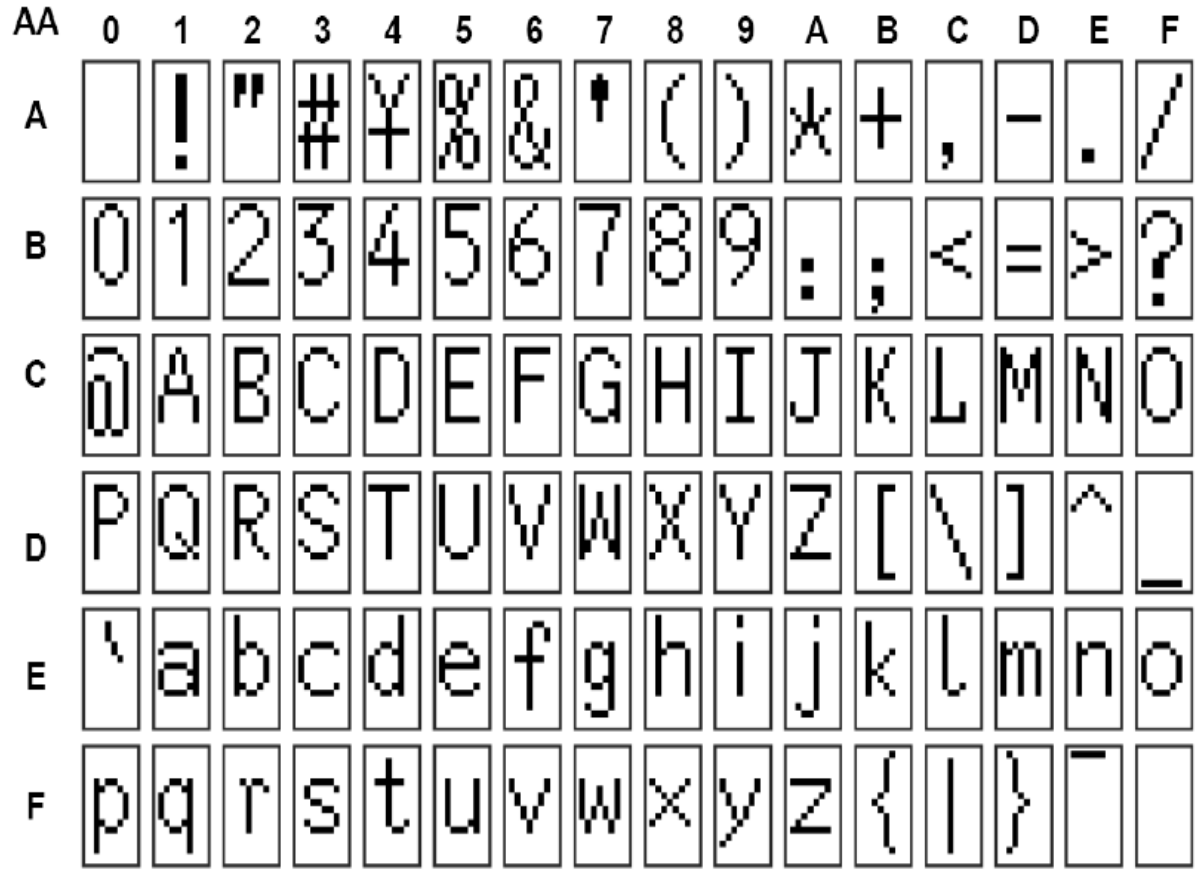
GB2312 1 区

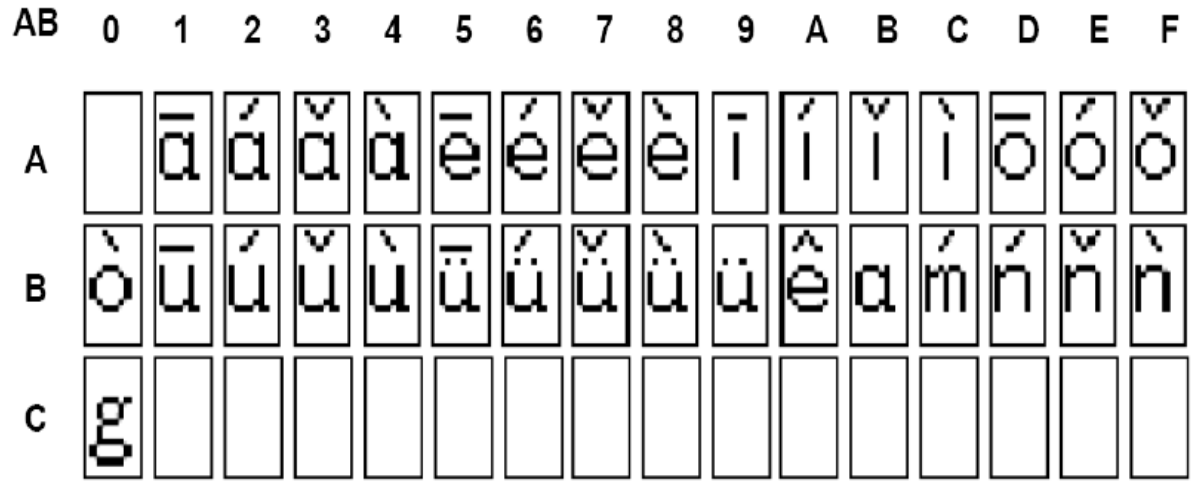




6.4.2 8×16点国标扩展字符

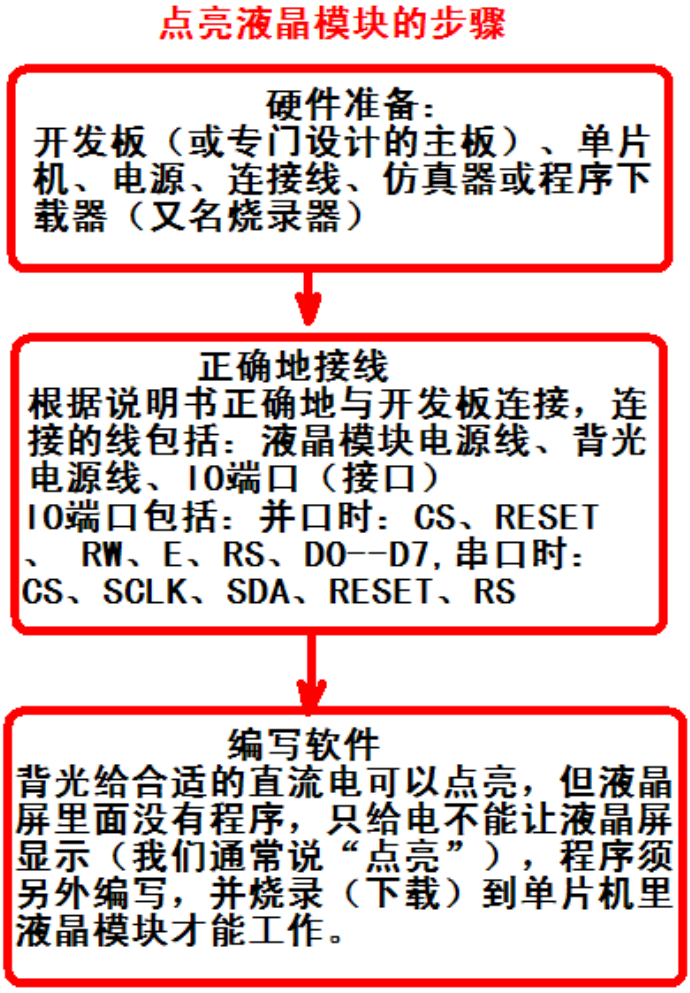
内码组成为AAA1~ABC0 共计126 个字符





7. 硬件设计及例程：

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



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： 下图为并行方式的硬件接口：

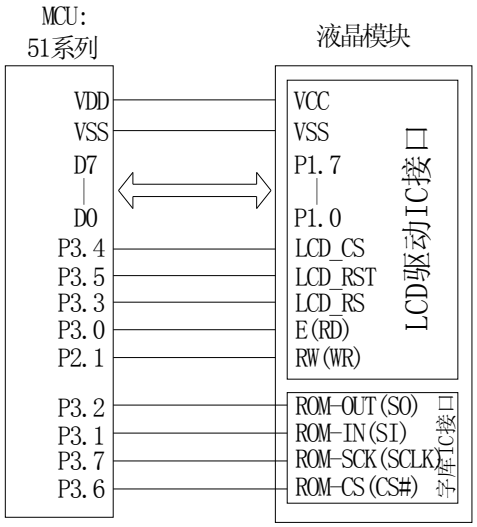


图 1：并行接口

7.1.2 例程：以下为串行方式显示汉字及 ASCII 字符的例程：

```
//液晶模块：JLX128128G-338-PC-P
//驱动 IC:UC1617S。4 灰阶（也叫灰度级）驱动 IC
//点阵：128x128
//说明：每个点阵是有 4 灰阶的（2 的平方=4），所以每个点阵是由 2 比特来代表的。
//为了应用一般的汉字库及普通的单色无灰阶,所以增设了：“write_mono_data(uchar mono_data)”这个函数
//如果要显示 4 灰阶的图像，可以通过相关 4 灰阶的取模软件来取数据。

#include <REG52.H>

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

//=====================================================================
sbit LCD_CS=P3^4;     /*3.4 接口定义*/
sbit LCD_RST=P3^5;    /*3.3 接口定义*/
sbit LCD_RS=P3^3;     /*接口定义*/
sbit E=P3^0;      /*接口定义*/
sbit RW=P2^1;         /*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#*/
//=====================================================================
uchar code bmp1[];

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

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


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

//等待一个按键
void waitkey()
{
 repeat:
   if (key==1) goto repeat;
   else;
        delay(600);
```

```
}


//=============transfer command to LCM===============
void transfer_command_lcd(int data1)
{
   LCD_CS=0;
   LCD_RS=0;
   E=0;
   RW=0;
   P1=data1;
   E=1;
   LCD_CS=1;
   E=0;
}

//-----------transfer data to LCM----------------
void transfer_data_lcd(int data1)
{
   LCD_CS=0;
   LCD_RS=1;
   E=0;
   RW=0;
   P1=data1;
   E=1;
   LCD_CS=1;
   E=0;
}


/*传送一个字节（8 比特）黑白图像的数据*/
void write_mono_data(uchar mono_data)
{
   char i,j;
   uchar four_gray_data=0;//定义 4 灰度级的数据
   for(j=0;j<2;j++)
   {
        four_gray_data=0;
        for(i=0;i<4;i++)
        {
             four_gray_data>>=2;              //4 灰度级的数据右移 2 位
             if(mono_data&0x80)          //单色黑白数据与 0x80（二进制 10000000）进行“与”运算
             {
                  four_gray_data+=0xc0;  //4 灰度级的数据+0xc0(二进制 11000000)
             }
             else;
             mono_data<<=1;                    //单色黑白数据左移一位
        }
        transfer_data_lcd(four_gray_data);        //写进一个 8bits 的数据，驱动了 4 个像素点，因为每个像素点用了 2bits
数据
    }
}
void clear()
{
   int i,j;
   LCD_CS=0;
   Rom_CS = 1;
   transfer_command_lcd(0x70);      //行地址高 3 位
   transfer_command_lcd(0x60);     //行地址低 4 位
   transfer_command_lcd(0x00);      //列地址
```

```
    for(i=0;i<128;i++)
    {

        for(j=0;j<16;j++)
        {
            write_mono_data(0x00);
        }
    }
}
//电测用的：全屏显示黑
void display_black(void)
{
  int i,j;
   LCD_CS=0;
   Rom_CS = 1;
  for(i=0;i<128;i++)
     {
        for(j=0;j<16;j++)
        {
            write_mono_data(0xff);
        }
     }
}
//电测用的：全屏显示偶数列
void display_even_column(void)
{
    int i,j;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<128;i++)
    {
        for(j=0;j<16;j++)
        {
            write_mono_data(0x55);
        }
    }
}
//电测用的：全屏显示奇数列
void display_odd_column(void)
{
    int i,j;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<128;i++)
    {
        for(j=0;j<16;j++)
        {
            write_mono_data(0xaa);
        }
    }
}

//电测用的：全屏显示雪花 1
void display_snow1(void)
{
    int i,j;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<64;i++)
    {
```

```
        for(j=0;j<16;j++)
            write_mono_data(0x55);
        for(j=0;j<16;j++)
            write_mono_data(0xaa);
    }
}
//电测用的：全屏显示雪花 2
void display_snow2(void)
{
    int i,j;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<64;i++)
    {
        for(j=0;j<16;j++)
            write_mono_data(0xaa);
        for(j=0;j<16;j++)
            write_mono_data(0x55);
    }
}
//电测用的：全屏显示奇数行
void display_odd_row(void)
{
    int i,j;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<64;i++)
    {
        for(j=0;j<16;j++)
            write_mono_data(0xFF);
        for(j=0;j<16;j++)
            write_mono_data(0x00);
    }
}
//电测用的：全屏显示偶数行
void display_even_row(void)
{
    int i,j;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<64;i++)
    {
        for(j=0;j<16;j++)
            write_mono_data(0x00);
        for(j=0;j<16;j++)
            write_mono_data(0xff);
    }
}


//定义显示窗口大小
void window_program()
{
    transfer_command_lcd(0x70);
    transfer_command_lcd(0x60);
    transfer_command_lcd(0x10);
    transfer_command_lcd(0x00);
    transfer_command_lcd(0xF4);//set colum address start
    transfer_command_lcd(0x00);
```

```
    transfer_command_lcd(0xF6);
    transfer_command_lcd(0x1f);//set colum address end
    transfer_command_lcd(0xF5);//set row address start
    transfer_command_lcd(0x00);
    transfer_command_lcd(0xF7);//set row address end
    transfer_command_lcd(0x7f);
    transfer_command_lcd(0xF9);//set window outside mode enable
}

void display_graphic_128x128(uchar *dp)
{
    int i,j,y=0,x=0,row;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<128;i++)
    {
        transfer_command_lcd(0x00+x);              //列地址，每个地址管 4 列
        row=y+i;
        transfer_command_lcd(0x70+(row>>4)); //行地址的高 3 位
        transfer_command_lcd(0x60+(row&0x0f));    //行地址的低 4 位
        for(j=0;j<16;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
    }
}

void display_graphic_16x16(uchar y,uchar x,uchar *dp)
{
    int i,j,row;
    LCD_CS=0;
    Rom_CS = 1;
    for(i=0;i<16;i++)
    {
        transfer_command_lcd(0x00+x);              //列地址，每个地址管 4 列
        row=y+i;
        transfer_command_lcd(0x70+(row>>4)); //行地址的高 3 位
        transfer_command_lcd(0x60+(row&0x0f));    //行地址的低 4 位
        for(j=0;j<2;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
    }
}
display_graphic_8x16(y,x,uchar *dp)
{
    int i,j,row;
    LCD_CS=0;
    Rom_CS = 1;

    for(i=0;i<16;i++)
    {
        transfer_command_lcd(0x00+x);              //列地址，每个地址管 4 列
        row=y+i;
        transfer_command_lcd(0x70+(row>>4)); //行地址的高 3 位
        transfer_command_lcd(0x60+(row&0x0f));    //行地址的低 4 位
        for(j=0;j<1;j++)
        {
```

```
            write_mono_data(*dp);
            dp++;
        }
    }
}
display_graphic_8x8(y,x,uchar *dp)
{
    int i,j,row;

    for(i=0;i<8;i++)
    {
        transfer_command_lcd(0x00+x);
        row=y+i;
        transfer_command_lcd(0x70+(row>>4));
        transfer_command_lcd(0x60+(row&0x0f));
        for(j=0;j<1;j++)
        {
            write_mono_data(*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++ )
    {
        if(datu&0x80)
            Rom_IN = 1;
        else
            Rom_IN = 0;
            datu = datu<<1;
            Rom_SCK=0;
            Rom_SCK=1;
    }
}

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

```
/*从相关地址（addrHigh：地址高字节,addrMid：地址中字节,addrLow：地址低字节）中连续读出 DataLen 个字节的数据到 pBuff 的地址*/
/*连续读取*/
void get_n_bytes_data_from_ROM(uchar addrHigh,uchar addrMid,uchar addrLow,uchar *pBuff,uchar DataLen )
{
  uchar i;
  Rom_CS = 0;
  LCD_CS=1;
  Rom_SCK=0;
  send_command_to_ROM(0x03);
  send_command_to_ROM(addrHigh);
  send_command_to_ROM(addrMid);
  send_command_to_ROM(addrLow);
  for(i = 0; i < DataLen; i++ )
      *(pBuff+i) =get_data_from_ROM();
  Rom_CS = 1;
}


/*****************************************************************/

ulong  fontaddr=0;
void display_GB2312_string(uchar y,uchar x,uchar *text)
{
  uchar i= 0;
  uchar addrHigh,addrMid,addrLow ;
  uchar fontbuf[32];
  while((text[i]>0x00))
  {
      Rom_SCK=0;
      if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
      {
          /*国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：*/
          /*Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0*/
          /*由于担心 8 位单片机有乘法溢出问题，所以分三部取地址*/
          fontaddr = (text[i]- 0xb0)*94;
          fontaddr += (text[i+1]-0xa1)+846;
          fontaddr = (ulong)(fontaddr*32);

          addrHigh = (fontaddr&0xff0000)>>16;  /*地址的高 8 位,共 24 位*/
          addrMid = (fontaddr&0xff00)>>8;      /*地址的中 8 位,共 24 位*/
          addrLow = fontaddr&0xff;         /*地址的低 8 位,共 24 位*/
          get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,32 );/*取 32 个字节的数据,存到"fontbuf[32]"*/
          display_graphic_16x16(y,x,fontbuf);/*显示汉字到 LCD 上，y 为页地址，x 为列地址，fontbuf[]为数据*/
          i+=2;
          x+=4;
      }
      else if(((text[i]>=0xa1) &&(text[i]<=0xa3))&&(text[i+1]>=0xa1))
      {
          /*国标简体（GB2312）15x16 点的全角标点符号（例如“，”“。”）在晶联讯字库 IC 中的地址由以下公式来计算：*/
          /*Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0*/
          /*由于担心 8 位单片机有乘法溢出问题，所以分三部取地址*/
          fontaddr = (text[i]- 0xa1)*94;
          fontaddr += (text[i+1]-0xa1);
          fontaddr = (ulong)(fontaddr*32);

          addrHigh = (fontaddr&0xff0000)>>16;  /*地址的高 8 位,共 24 位*/
          addrMid = (fontaddr&0xff00)>>8;      /*地址的中 8 位,共 24 位*/
          addrLow = fontaddr&0xff;         /*地址的低 8 位,共 24 位*/
```

```
                get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,32 );/*取 32 个字节的数据,存到"fontbuf[32]"*/
                display_graphic_16x16(y,x,fontbuf);/*显示汉字到 LCD 上，y 为页地址，x 为列地址，fontbuf[]为数据*/
                i+=2;
                x+=4;
            }
            else if((text[i]>=0x20) &&(text[i]<=0x7e))
            {   /*ASCII 码字符的 8*16 点阵在字库 IC 中的地址*/
                unsigned char fontbuf[16];
                fontaddr = (text[i]- 0x20);
                fontaddr = (unsigned long)(fontaddr*16);
                fontaddr = (unsigned long)(fontaddr+0x3b7c0);
                addrHigh = (fontaddr&0xff0000)>>16;
                addrMid = (fontaddr&0xff00)>>8;
                addrLow = fontaddr&0xff;

                get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,16 );/*取 16 个字节的数据,存到"fontbuf[32]"*/

                display_graphic_8x16(y,x,fontbuf);/*显示 8x16 的 ASCII 字到 LCD 上，y 为页地址，x 为列地址，fontbuf[]为数据
*/
                i+=1;
                x+=2;
            }
            else
                i++;
        }

    }

    void display_string_8x8(uchar y,uchar x,uchar *text)
    {
        unsigned char i= 0;
        unsigned char addrHigh,addrMid,addrLow ;
        while((text[i]>0x00))
        {

            if((text[i]>=0x20) &&(text[i]<=0x7e))
            {
                unsigned char fontbuf[8];
                fontaddr = (text[i]- 0x20);
                fontaddr = (unsigned long)(fontaddr*8);
                fontaddr = (unsigned long)(fontaddr+0x66c0);
                addrHigh = (fontaddr&0xff0000)>>16;
                addrMid = (fontaddr&0xff00)>>8;
                addrLow = fontaddr&0xff;

                get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,8);/*取 8 个字节的数据，存到"fontbuf[32]"*/

                display_graphic_8x8(y,x,fontbuf);/*显示 5x7 的 ASCII 字到 LCD 上，y 为页地址，x 为列地址，fontbuf[]为数据*/
                i+=1;
                x+=1;
            }
            else
            i++;
        }

    }
    void LCD_INITIAL()
    {
        lcd_reset=0;
        delay(1);     //大于 0.1ms
```

```
    lcd_reset=1;
    delay(500);        //大于 50ms

    transfer_command_lcd(0xE2);//system reset
    delay(10);
    transfer_command_lcd(0x27);
    transfer_command_lcd(0x2b);
    transfer_command_lcd(0x2f); //set pump control
    transfer_command_lcd(0xeb); //set bias=1/11
    transfer_command_lcd(0x81); //set
    transfer_command_lcd(0x36); //set PM=12,vop=12.8v,4c
    transfer_command_lcd(0xa9); //set linerate mux,a2
    transfer_command_lcd(0xc8);
    transfer_command_lcd(0x0b);
    transfer_command_lcd(0x89);
    transfer_command_lcd(0xc4); //MY=1,MX=0:从左到右，再从上到下。
    transfer_command_lcd(0xf1); //f1
    transfer_command_lcd(0x7f);
    transfer_command_lcd(0xd3); //gray shade set
    transfer_command_lcd(0xd7); //gray shade set
    transfer_command_lcd(0xAf); //set display enable
}

void main(void)
{
    LCD_INITIAL();
    while(1)
    {
        clear();
        display_GB2312_string(16*0,0,"128128 带中文字库");
        display_GB2312_string(16*1,0,"16X16 点阵,○●◎");
        display_GB2312_string(16*2,0,"或 8X16 点阵 ASCII,");
        display_GB2312_string(16*3,0,"或 8X8 点阵 ASCII 码");
        display_GB2312_string(16*4,0,"GB2312 简体字库及");
        display_GB2312_string(16*5,0,"有图型功能，可自");
        display_GB2312_string(16*6,0,"编大字、图像、生");
        display_GB2312_string(16*7,0,"僻字，例如：    ");
        display_graphic_16x16(16*7,24,jiong1);          /*在第 yy 行，第 xx 列显示单个自编生僻汉字“囧”*/
        display_graphic_16x16(16*7,28,lei1);          /*显示单个自编生僻汉字"蠃“*/
        waitkey();
        clear();
        display_graphic_128x128(bmp1);
        waitkey();
        display_black();
        waitkey();
        display_odd_column();
        waitkey();
        display_even_column();
        waitkey();
        display_odd_row();
        waitkey();
        display_even_row();
        waitkey();
        display_snow1();
        waitkey();
        display_snow2();
        waitkey();
    }
}
```

```
uchar code bmp1[]={//横向取模
/*--  调入了一幅图像：E:\work\图片收藏夹\黑白屏图片\JLX128128G-338.bmp  --*/
/*--  宽度 x 高度=128x128  --*/
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x08,0x02,0x00,0x00,0x41,0x80,0x40,0x00,0x00,0x00,0xC0,0x00,0x00,0x00,0x81,
0x80,0x0F,0xFE,0x00,0x03,0x21,0x80,0x60,0x01,0x80,0x00,0x80,0x00,0x7F,0xFF,0x81,}
```

7.1.3、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：

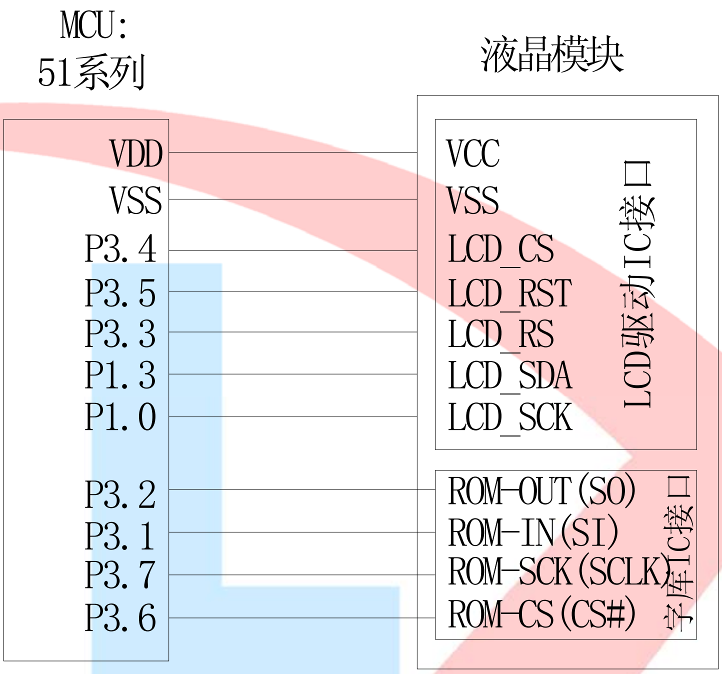


图 2：串行接口图

```
//=========================================================================
sbit LCD_RST = P3^5;
sbit LCD_RS  = P3^3;
sbit LCD_CS = P3^4;
sbit LCD_SDA = P1^3;
sbit LCD_SCL = P1^0;
sbit key=P2^0;

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#*/
//=========================================================================

//传送指令
void transfer_command_lcd(unsigned char cmd)
{
   int k;
```

```
    LCD_CS = 0;
    LCD_RS= 0;

    for (k=0; k<8; k++)
    {
         cmd = cmd<<1;
         LCD_SCL = 0;
         delay(1);
         LCD_SDA = CY;
//       delay_us(10);
         LCD_SCL = 1;
         delay(1);
    }

    LCD_CS=1;
}

//传送数据
void transfer_data_lcd(unsigned char dat)
{
    unsigned char k;
    LCD_CS = 0;
    LCD_RS = 1;
    for (k=0; k<8; k++)
    {
         dat = dat<<1;
         LCD_SDA = CY;
         LCD_SCL = 0;
//       delay_us(10);
         LCD_SCL = 1;
//       delay_us(10);
    }
    LCD_CS=1;
}
```

7.1.4、以下为 IIC 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：

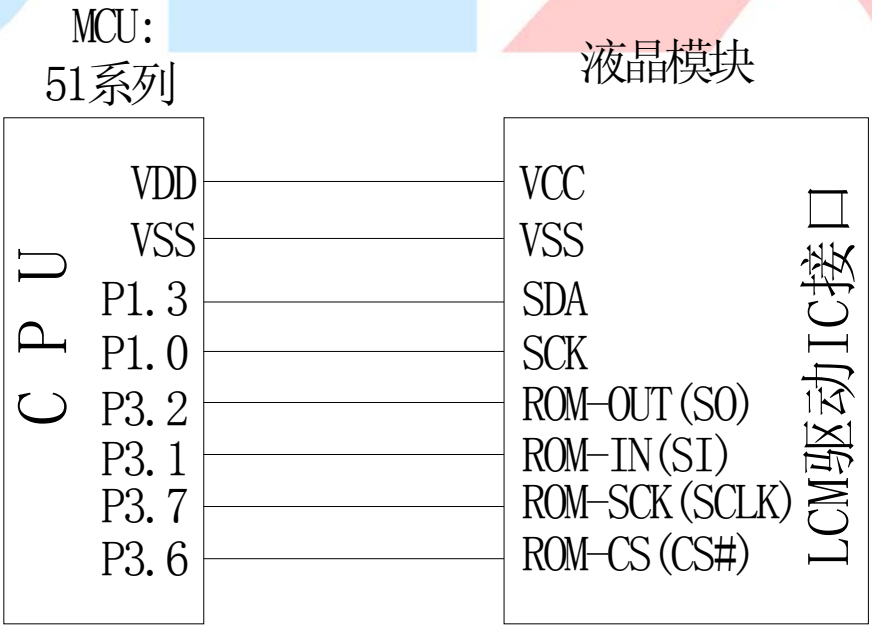


图 3：IIC 接口图

```
//=====================================================================
sbit sda = P1^3;
sbit scl = P1^0;
sbit key=P2^0;       /*按键接口，P2.0 口与 GND 之间接一个按键*/
//=====================================================================
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*/

}
```