

JLX12864OLED-242-PC

带字库 IC 的编程说明书

目 录

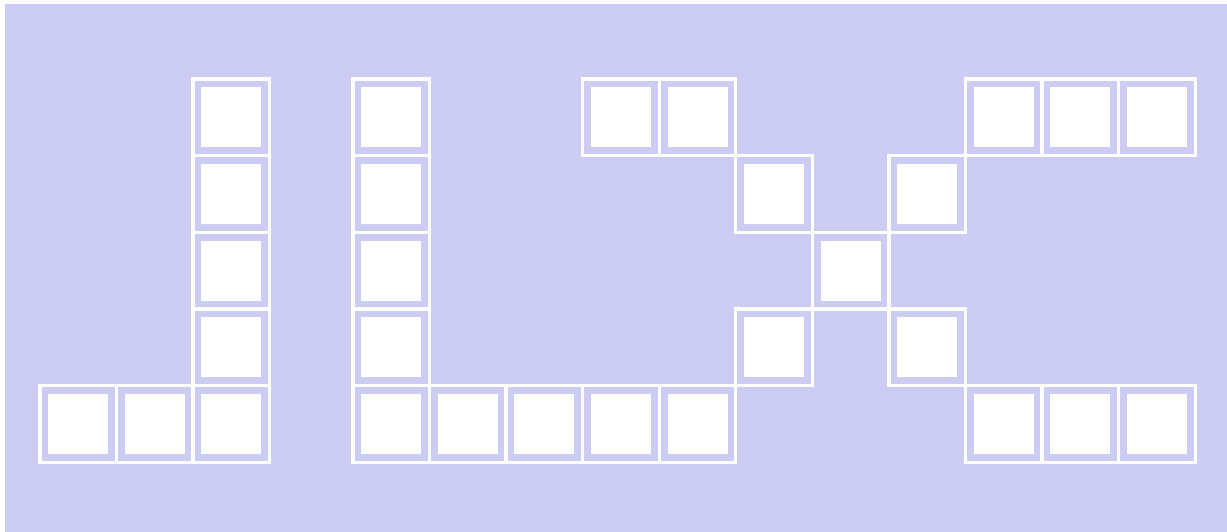
序号	内 容 标 题	页 码
1	概述	2
2	字型样张：	3
3	外形尺寸及接口引脚功能	4~6
4	工作电路框图	6
5	指令	7~9
6	字库的调用方法	10~18
7	硬件设计及例程：	19~尾页

1. 概述

JLX12864OLED-242-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:
=>?@ABCDEFGHIJKLMN OPQRSTU
VWXYZ[\]^_`abcdefghijklmnopqr

7x8 点 ASCII 字符

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

8x16 点 ASCII 字符

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

8x16 点 ASCII 粗体字符

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

16 点阵不等宽 ASCII 方头

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

16 点阵不等宽 ASCII 白正

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

3. 外形尺寸及接口引脚功能

3.1 外形图:

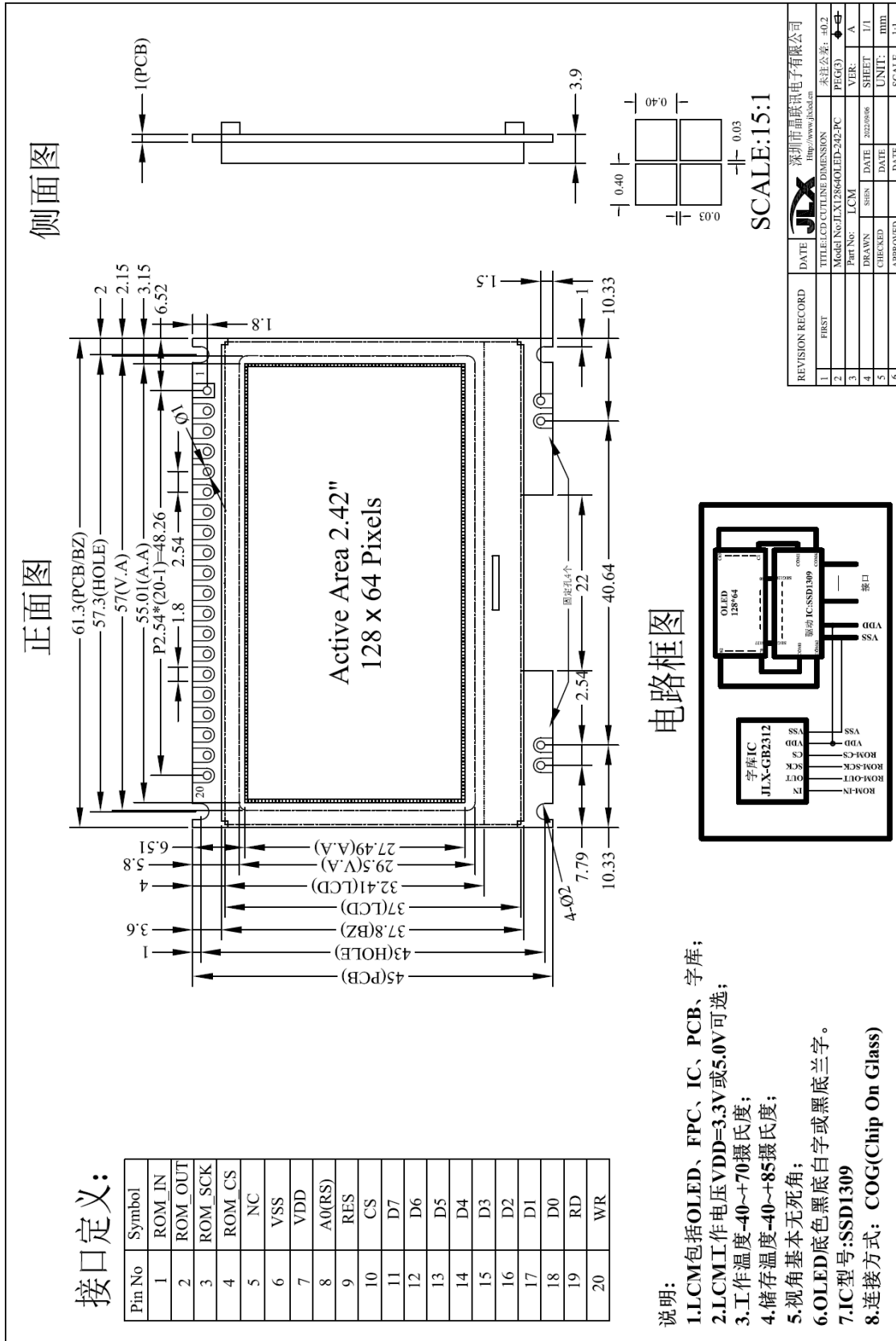


图 1. 外形尺寸

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
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	NC	NC	空脚
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-DO	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 1: 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
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	NC	NC	空脚
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	SDA	I/O	串行数据
12	SCK	I/O	串行时钟
13~20	空	空	空

表 2: 模块串行接口引脚功能

3.2.3 I²C 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
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
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#			
5	NC	NC	空脚
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	RS	接 VSS, 从属地址: 0X78; 接 VDD, 从属地址: 0x7A
9	RES	NC	空脚
10	CS	NC	空脚
11	D7	I/O	串行数据 SDA
12	D6	I/O	串行时钟 SCL
13~20	空	空	空

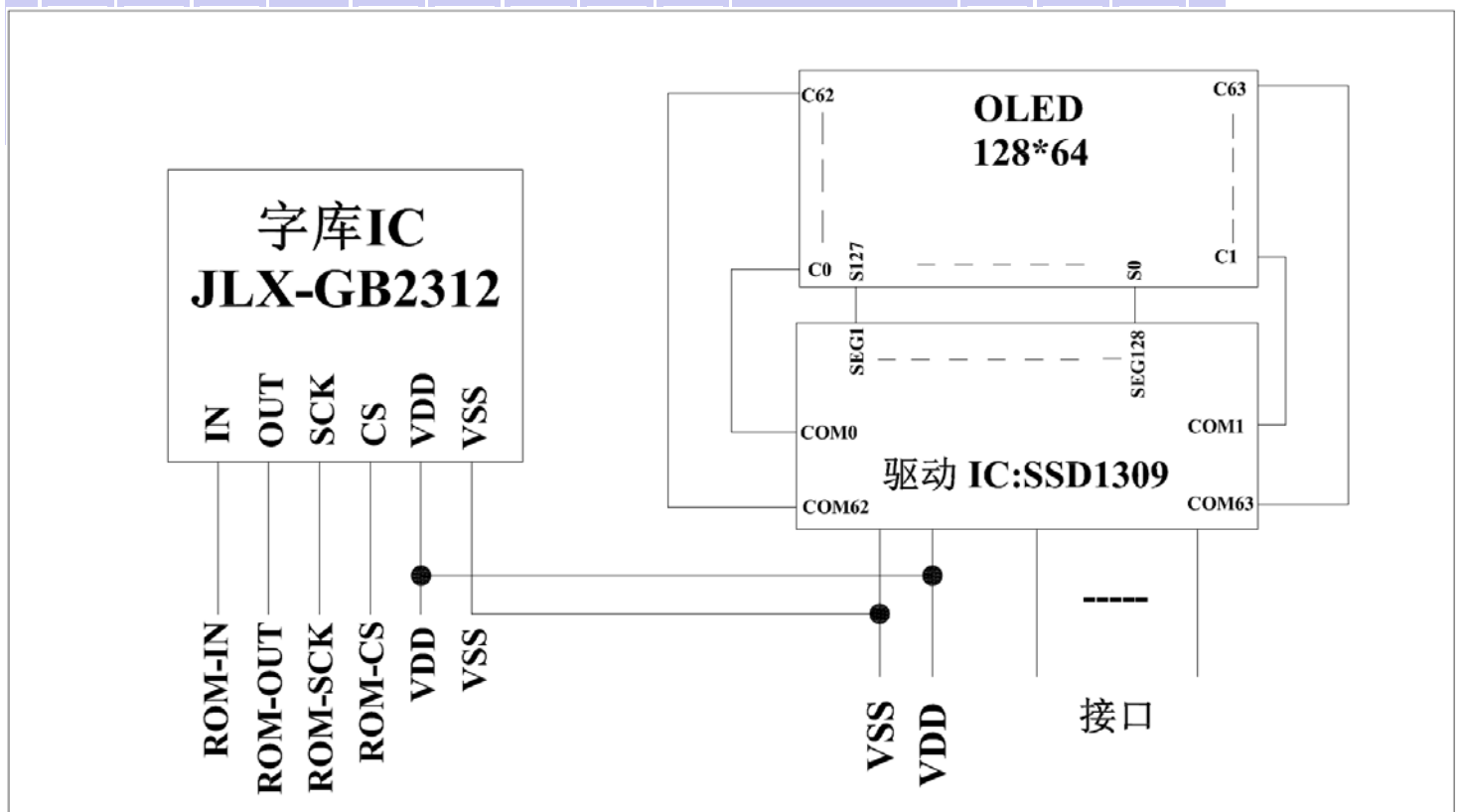
表 3: 模块串行接口引脚功能

4. 基本原理

4.1 液晶屏 (LCD)

在 LCD 上排列着 128×64 点阵, 128 个列信号与驱动 IC 相连, 64 个行信号也与驱动 IC 相连, IC 邦定在 LCD 玻璃上 (这种加工工艺叫 COG)。

电路框图



5. 指令:

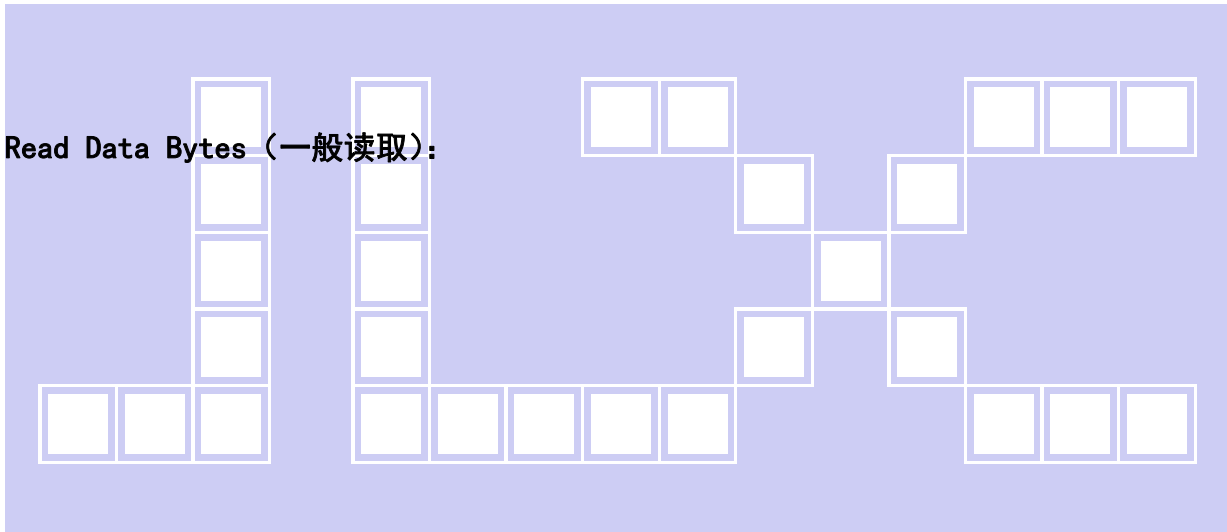
5.1 字库 IC (JLX-GB2312) 指令表

Instruction Set

Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
READ	Read Data Bytes	0000 0011	03 h	3	-	1 to ∞
FAST_READ	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):

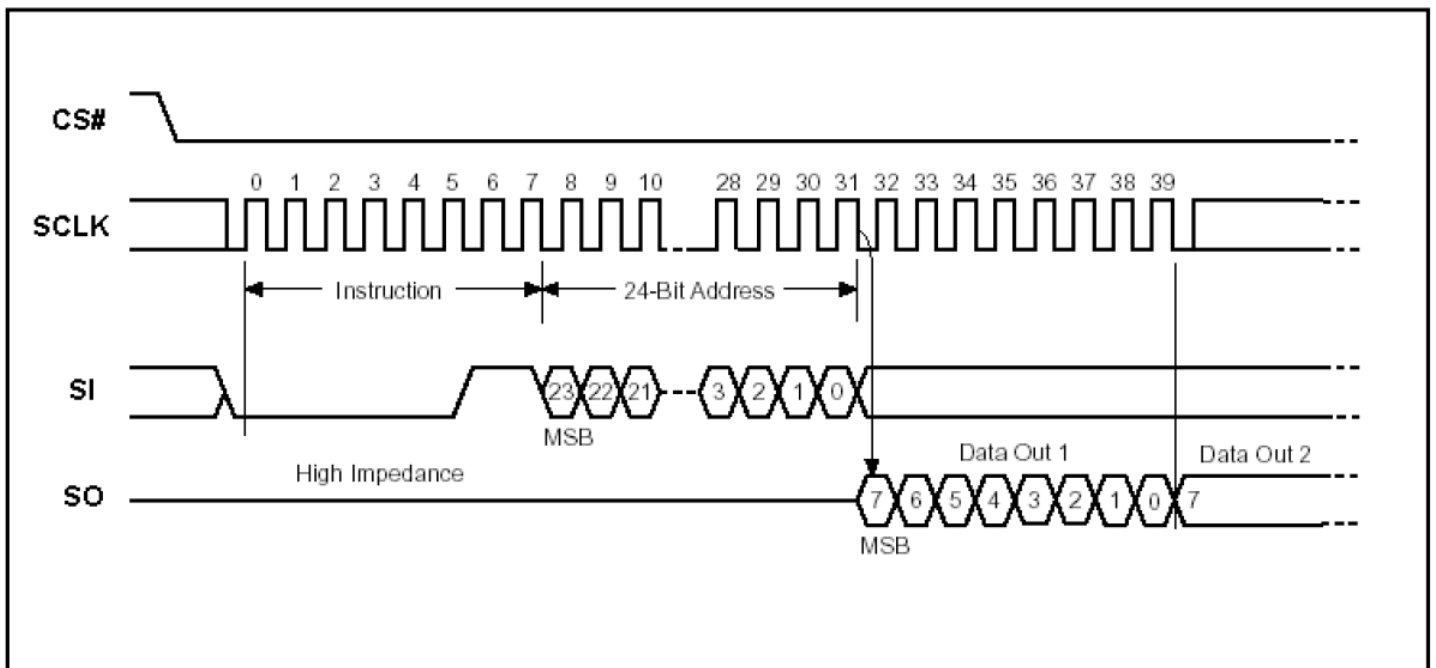
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

■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):

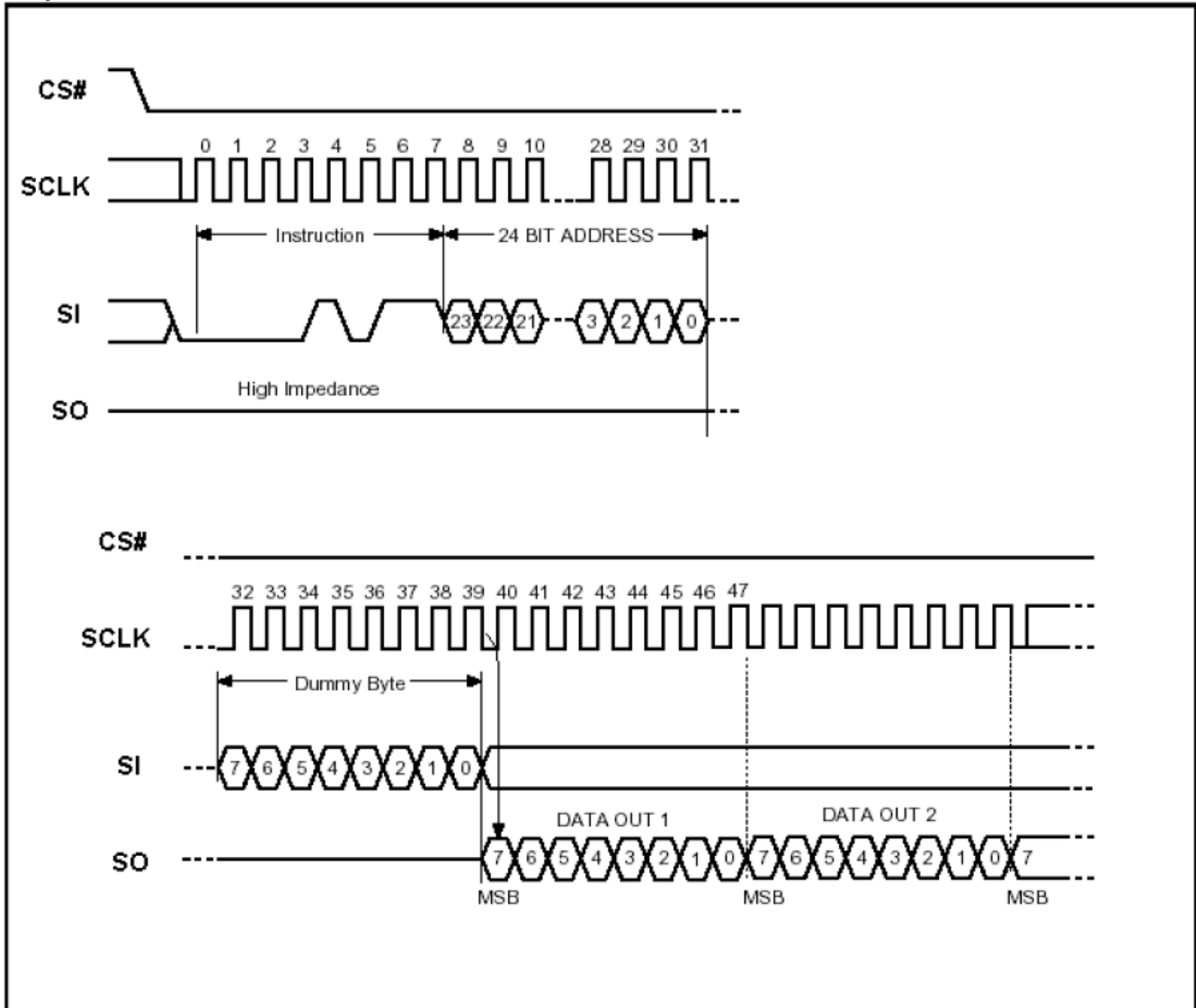
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

■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



5.2 LCD 驱动 IC 指令表详见“JLX12864OLED-242-PN”的中文说明书

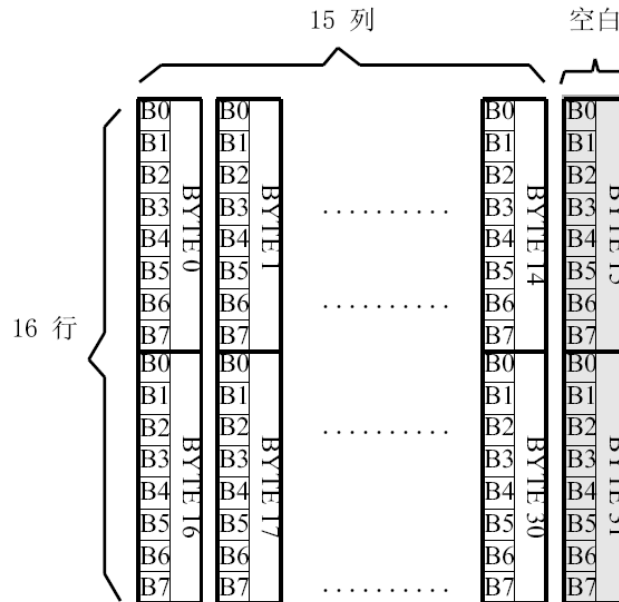
6 字库调用方法

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 7 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。

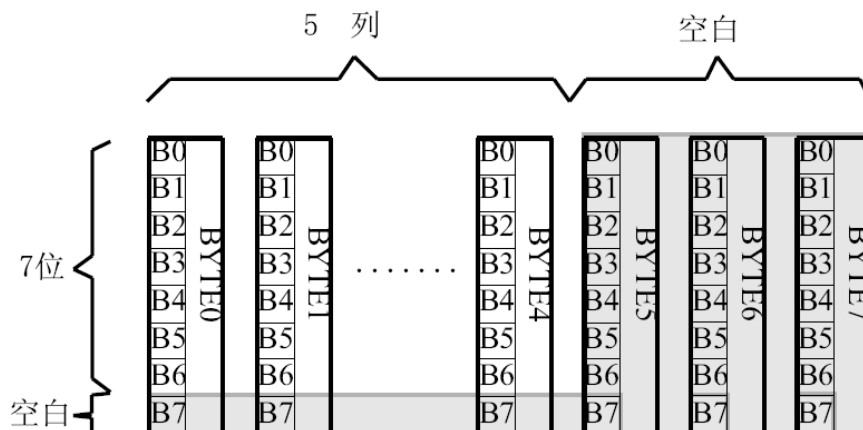
6.1.1 15X16 点汉字排列格式

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：



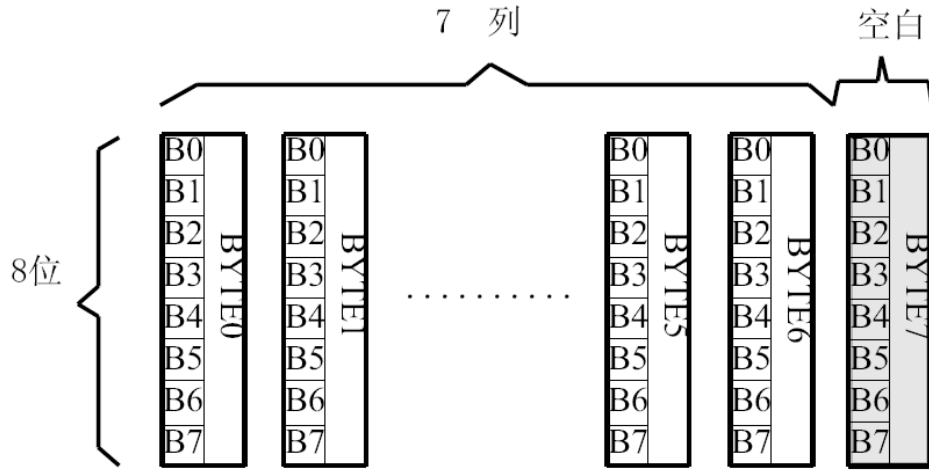
6.1.2 5X7 点 ASCII 字符排列格式

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

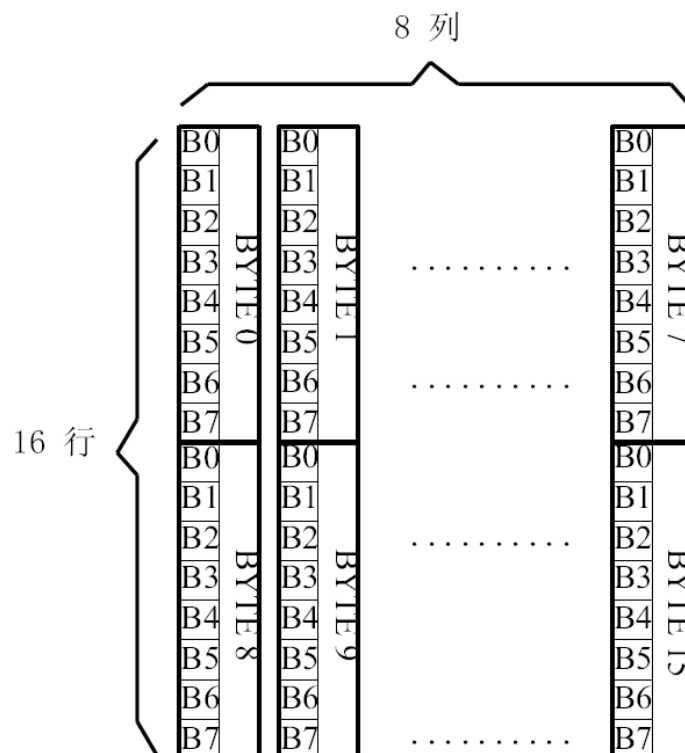
适用于此种排列格式的字体有:

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:

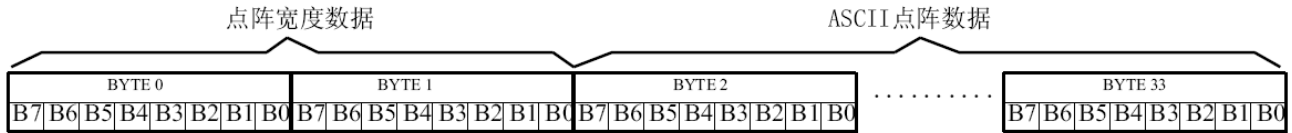


6.1.5 16 点阵不等宽 ASCII 方头 (Arial)、白正 (Times New Roman) 字符排列格式

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

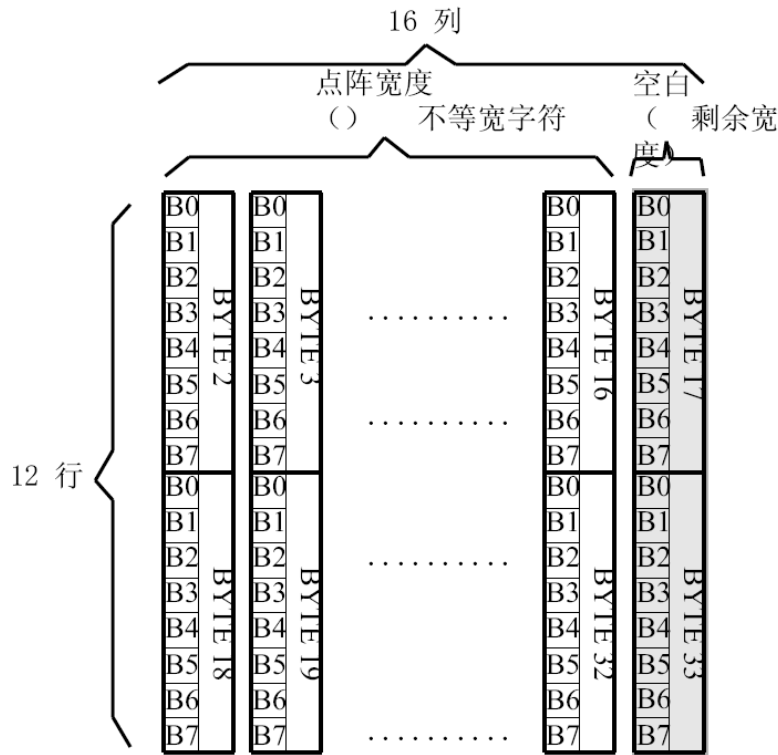
■ 存储格式

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 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 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 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

6.3.1 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:

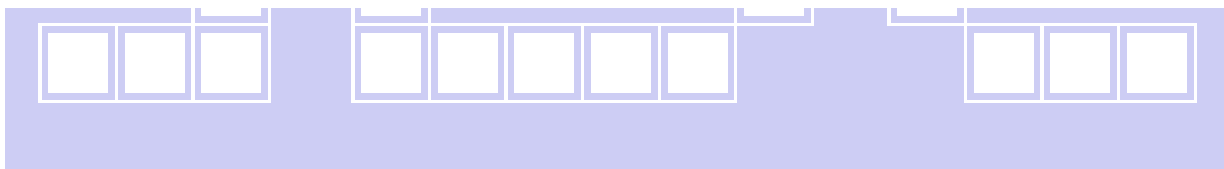
ASCIICode: 表示 ASCII 码 (8bits)

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。

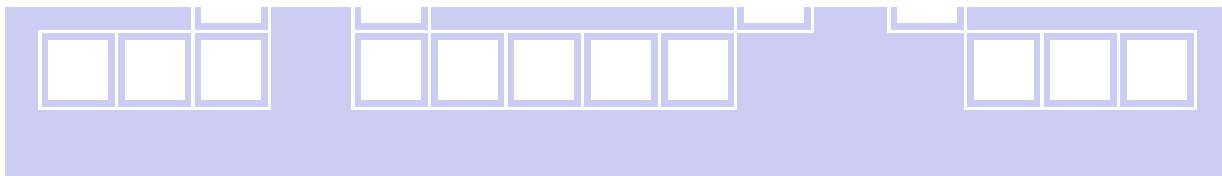
Address: ASCII 字符点阵在芯片中的字节地址。

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。

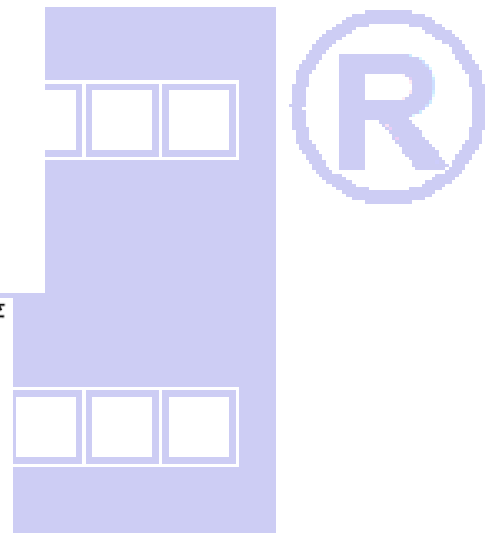
Address: ASCII 字符点阵在芯片中的字节地址。

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	—	~		…	‘	’
B	“	”	()	<	>	《	》	「	」	『	』	【	】	【	】
C	±	×	÷	:	∧	∨	Σ	Π	U	∩	€	::	√	⊥	//	∠
D	∩	⊙	∫	∫	=	≈	≈	∞	≠	<	>	<	>	∞	::	
E	∴	↑	♀	°	'	”	℃	\$	∩	∅	£	%	§	No	☆	★
F	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	=	

A2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A																
B		1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
C	16.	17.	18.	19.	20.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
D	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	①	②	③	④	⑤	⑥	⑦
E	⑧	⑨	⑩	€		(一)	(二)	(三)	(四)	(五)	(六)	(七)	(八)	(九)	(十)	
F		I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII			

A3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		!	”	#	¥	%	&	'	()	*	+	,	-	.	/
B	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
C	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
D	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
E	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
F	p	q	r	s	t	u	v	w	x	y	z	{		}	—	

A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A					—	—			---	---	!	!	---	---	!	!
B	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌	┌
C	└	└	└	└	└	└	└	└	└	└	└	└	└	└	└	└
D	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐	┐
E	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
F																

6.4.2 8×16点国标扩展字符

内码组成为 AAA1~ABC0 共计 126 个字符

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

A		!	"	#	¥	%	&	†	()	*	+	,	-	.	/
B	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
C	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
D	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
E	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
F	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

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

A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ǘ	ú	ǚ	ù	ü	ê	á	ń	ň	ñ	ň
C	g															

7. 硬件设计及例程：

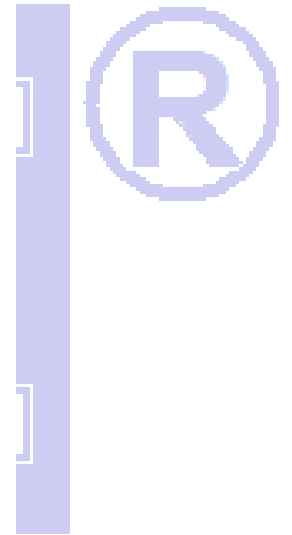
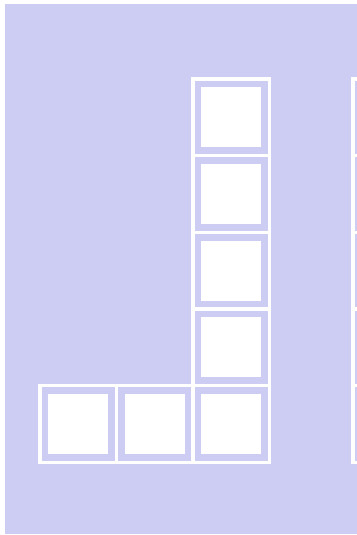
7.1 用户所编的显示程序, 开始必须进行初始化, 否则模块无法正常显示, 过程请参考程序

点亮OLED模块的步骤

硬件准备：
开发板（或专门设计的主板）、单片机、电源、连接线、仿真器或程序下载器（又名烧录器）

正确地接线
根据说明书正确地与开发板连接，连接的线包括：液晶模块电源线、背光电源线、I/O端口（接口）
I/O端口包括：并口时：CS、RESET、RW、E、RS、D0--D7，串口时：CS、SCLK、SDA、RESET、RS

编写软件
OLED 屏里面没有程序，只给电不能让 **OLED** 显示（我们通常说“点亮”），程序须另外编写，并烧录（下载）到单片机里 **OLED** 才能工作。



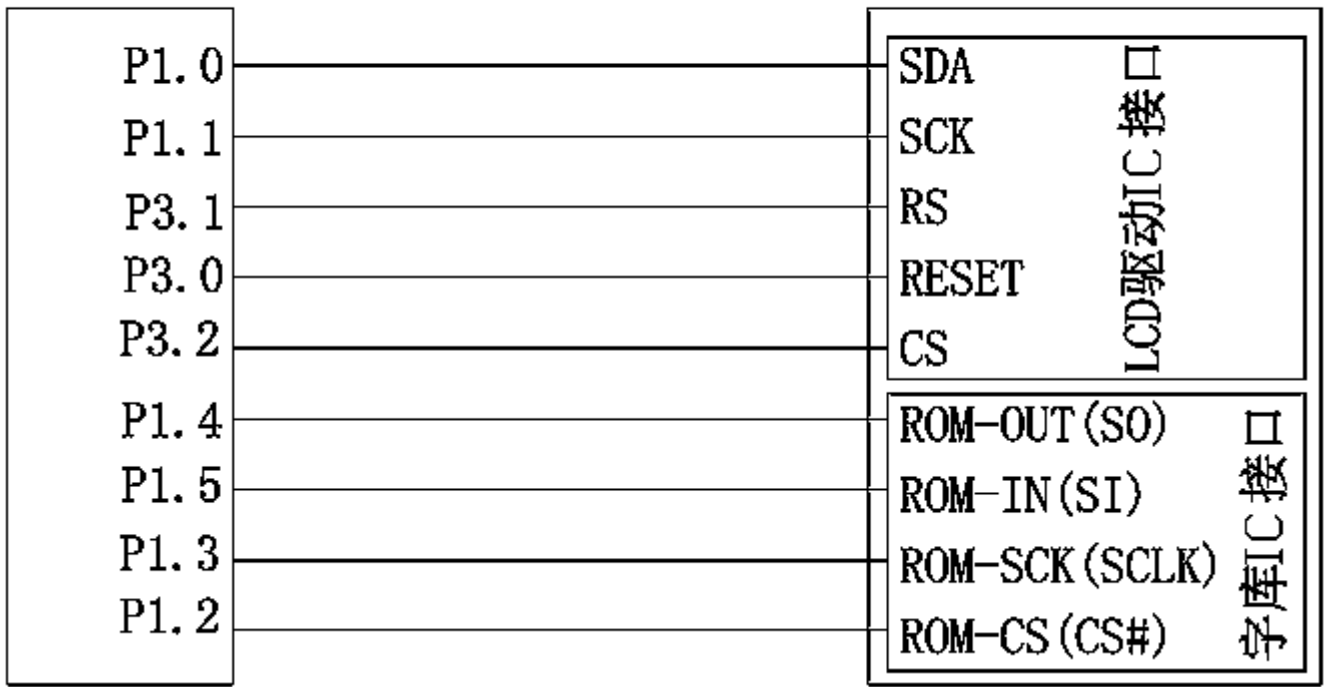
7.1.1 硬件接口：

下图为串行方式的硬件接口：

MCU：

51系列

OLED模块



7.2 程序：

点亮OLED模块的编程步骤



```

// 液晶演示程序
// 液晶模块型号：JLX12864OLED-242-PC，串行接口！
// 版权所有：晶联讯电子；网址 http://www.jlxlcd.cn;
  
```

```

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

```

//=====
sbit lcd_sclk =P1^1; //接口定义:lcd_sclk 就是 LCD 的 SCLK
sbit lcd_sda =P1^0; //接口定义:lcd_sda 就是 LCD 的 SDA
sbit lcd_reset=P3^0; //接口定义:lcd_reset 就是 LCD 的 RESET
  
```

```

sbit lcd_dc =P3^1; //接口定义:lcd_dc 就是 LCD 的 D/C
sbit lcd_cs1=P3^2; //接口定义:lcd_cs1 就是 LCD 的 CS

sbit Rom_OUT=P1^4; /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 S0*/
sbit Rom_IN=P1^5; /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 S1*/
sbit Rom_SCK=P1^3; /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
sbit Rom_CS=P1^2; /*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: P2.0 口与 GND 之间接一个按键
//=====

```

```

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

#include <Chinese_And_Graphic.H>

```

//延时

```

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

```

//等待按键: P2.0 口与 GND 之间接一个按键

```

void waitkey()
{
repeat:  if(key==1) goto repeat;
        else delay(1200);
}

```

//写指令到 OLED 显示模块

```

void transfer_command(int data1)
{
    uchar i;
    lcd_cs1=0;
    lcd_dc= 0;
    for(i=0;i<8;i++)
    {
        lcd_sclk = 0;
        if (data1 & 0x80) lcd_sda = 1;
        else lcd_sda = 0;
        lcd_sclk = 1;
        data1 <<= 1;
    }
    lcd_cs1=1;
}

```



//写数据到 OLED 显示模块

```
void transfer_data(int data1)
```

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

//OLED 显示模块初始化

```
void initial_lcd0
```

```
{
    delay(200);
    lcd_cs1=0;
    Rom_CS = 1;
    lcd_reset=0; //低电平复位
    delay(200);
    lcd_reset=1; //复位完毕
    delay(200);
```

```
transfer_command(0xae); //关显示
```

```
transfer_command(0xd5); //晶振频率
```

```
transfer_command(0x80);
```

```
transfer_command(0xa8); //duty 设置
```

```
transfer_command(0x3f); //duty=1/64
```

```
transfer_command(0xd3); //显示偏移
```

```
transfer_command(0x00);
```

```
transfer_command(0x40); //起始行
```

```
transfer_command(0x8d); //升压允许
```

```
transfer_command(0x14);
```

```
transfer_command(0x20); //page address mode
```

```

transfer_command(0x02);

transfer_command(0xc8); //行扫描顺序: 从上到下
transfer_command(0xa1); //列扫描顺序: 从左到右

transfer_command(0xda); //sequential configuration
transfer_command(0x12);

transfer_command(0x81); //设置对比度, 本指令的 0x81 不要改动, 改下面的值
transfer_command(0xcf); //设置对比度的值, 可设置范围 0x00~0xff

transfer_command(0xd9); //Set Pre-Charge Period
transfer_command(0xf1);

transfer_command(0xdb); //Set VCOMH Deselect Level
transfer_command(0x40);
    
```

```

transfer_command(0xaf); //开显示
}

void lcd_address(uchar page, uchar column)
{
    column=column-1; //我们平常所说的第 1 列, 在 LCD 驱动 IC 里是第 0 列。所以在这里减去 1.
    page=page-1;
    transfer_command(0xb0+page); //设置页地址。每页是 8 行。一个画面的 64 行被分成 8 个页。我们平常所说的第 1 页, 在 LCD 驱动 IC 里是第 0 页, 所以在这里减去 1
    transfer_command(((column>>4)&0x0f)+0x10); //设置列地址的高 4 位
    transfer_command(column&0x0f); //设置列地址的低 4 位
}
    
```

//全屏清屏

```

void clear_screen()
{
    unsigned char i, j;
    for(j=0; j<8; j++)
    {
        lcd_address(1+j, 1);
        for(i=0; i<128; i++)
        {
            transfer_data(0x00);
        }
    }
}
    
```

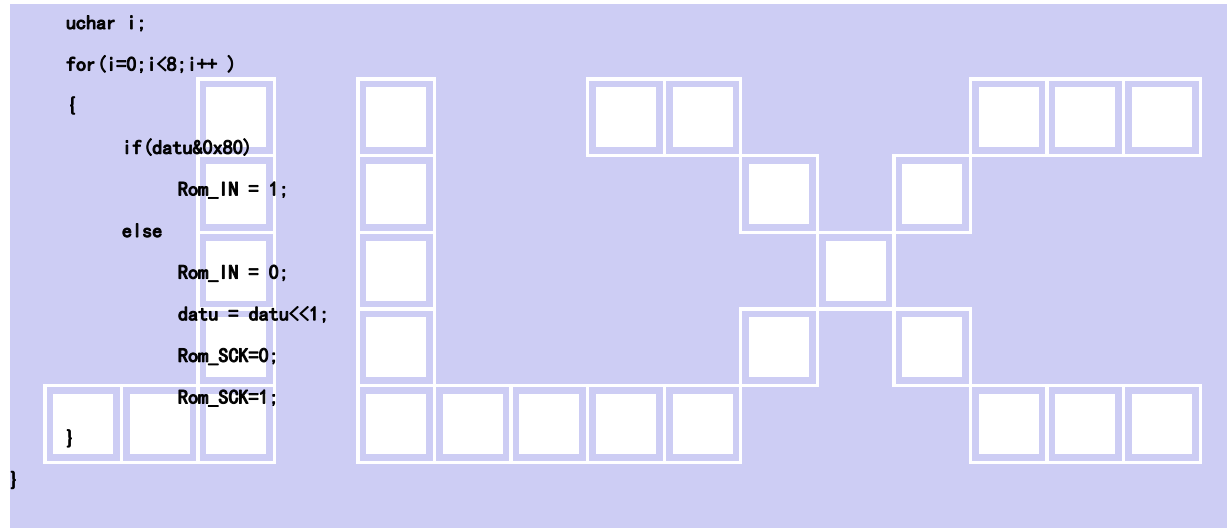
//full display test


```
void full_display(uchar data1,uchar data2)
{
    int i, j;
    for (i=0; i<8; i++)
    {
        lcd_address(i+1, 1);
        for (j=0; j<64; j++)
        {
            transfer_data(data1);
            transfer_data(data2);
        }
    }
}
```

****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_cs1=1;
    Rom_SCK=0;
    send_command_to_ROM(0x03);
    send_command_to_ROM(addrHigh);
    send_command_to_ROM(addrMid);
    send_command_to_ROM(addrLow);
    for(i = 0; i < DataLen; i++)
        *(pBuff+i) =get_data_from_ROM();
    Rom_CS = 1;
}

```

//显示 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(*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 128x16 点阵图像

```

void display_128x16(uchar page, uchar column, uchar *dp)

```

```

{
    uint i, j;
    for(j=0; j<2; j++)
    {
        lcd_address(page+j, column);
        for (i=0; i<128; i++)
        {
            transfer_data(*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 32x32 点阵图像、汉字、生僻字或 32x32 点阵的其他图标

```

void display_graphic_32x32(uchar page,uchar column,uchar *dp)
{
    uchar i,j;
    for(j=0;j<4;j++)
    {
        lcd_address(page+j,column);
        for(i=0;i<32;i++)
        {
            transfer_data(*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(uint page,uint column,uchar *dp)
{
    uint i,j;
    lcd_cs1=0;
    Rom_CS = 1;
    for(j=0;j<2;j++)
    {
        lcd_address(page+j,column);
        for(i=0;i<16;i++)
        {
            transfer_data(*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++;
        }
    }
    lcd_cs1=1;
}

```

//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)
{
    uchar i,j;
    for(j=0;j<2;j++)
    {
        lcd_address(page+j,column);
        for(i=0;i<8;i++)

```



```

{
    transfer_data(*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 5*7 点阵图像、ASCII, 或 5*7 点阵的自造字符、其他图标*/

```
void display_graphic_5x7(uint page, uchar column, uchar *dp)
```

```

{
    uint col_cnt;
    uchar page_address;
    uchar column_address_L, column_address_H;
    page_address = 0xb0+page-1;
    lcd_cs1=0;
    column_address_L =(column&0x0f)-1;
    column_address_H =((column>>4)&0x0f)+0x10;

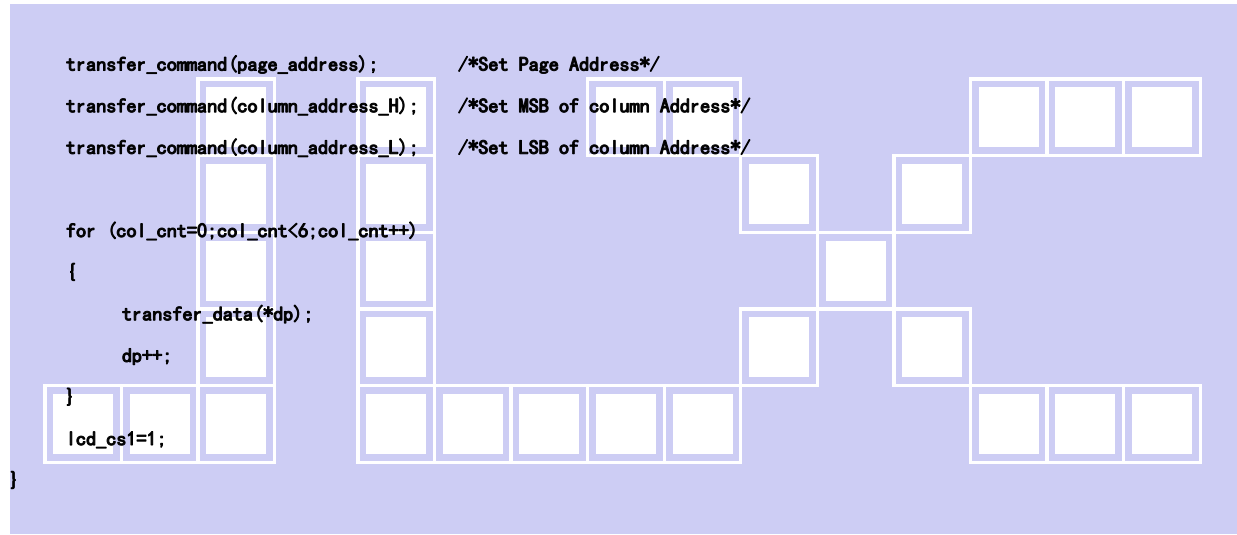
```

```

transfer_command(page_address);    /*Set Page Address*/
transfer_command(column_address_H); /*Set MSB of column Address*/
transfer_command(column_address_L); /*Set LSB of column Address*/

for (col_cnt=0;col_cnt<6;col_cnt++)
{
    transfer_data(*dp);
    dp++;
}
lcd_cs1=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))
    {
        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+=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);
```

```
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+=16;
```

```
}
```

```
else if((text[i]>=0x20) &&(text[i]<=0x7e))
```

```
{
```

```
unsigned char fontbuf[16];
```

```
fontaddr = (text[i]- 0x20);
```

```
fontaddr = (unsigned long) (fontaddr*16);
```

```
fontaddr = (unsigned long) (fontaddr+0x3cf80);
```

```
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+=8;
```

```
}
```

```
else
```

```
i++;
```

```
}
```

```
}
```

```
void display_string_5x7(uchar y, uchar x, uchar *text)
```

```
{
```

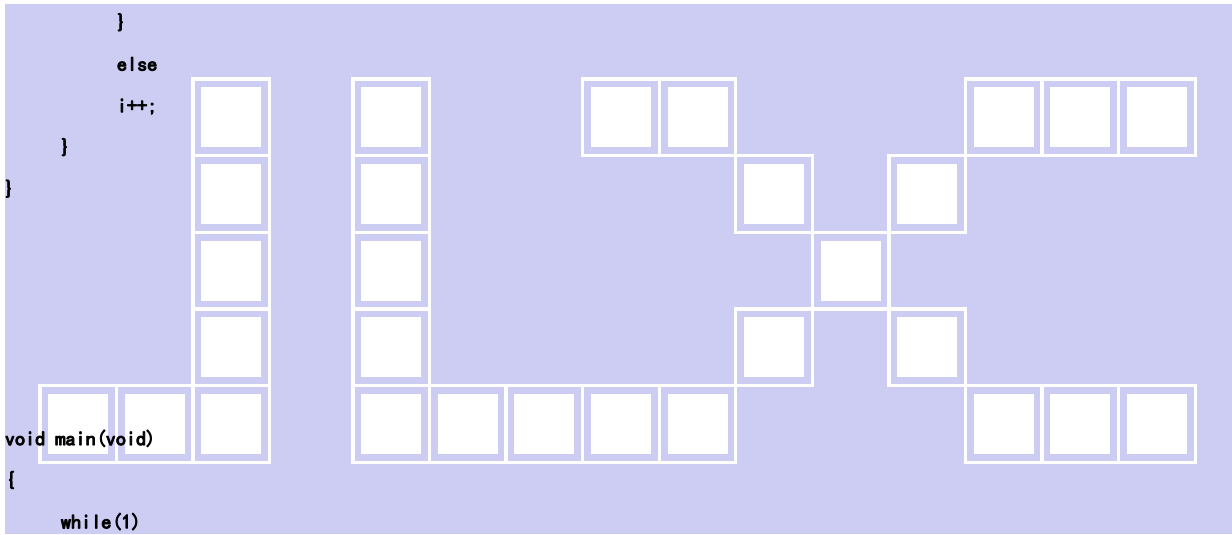
```
unsigned char i= 0;
```

```

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+0x3bfc0);
        addrHigh = (fontaddr&0xff0000)>>16;
        addrMid = (fontaddr&0xff00)>>8;
        addrLow = fontaddr&0xff;
        get_n_bytes_data_from_ROM(addrHigh, addrMid, addrLow, fontbuf, 8); /*取 8 个字节的数据, 存到"fontbuf[32]"*/

        display_graphic_5x7(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+=6;
    }
}

```



```

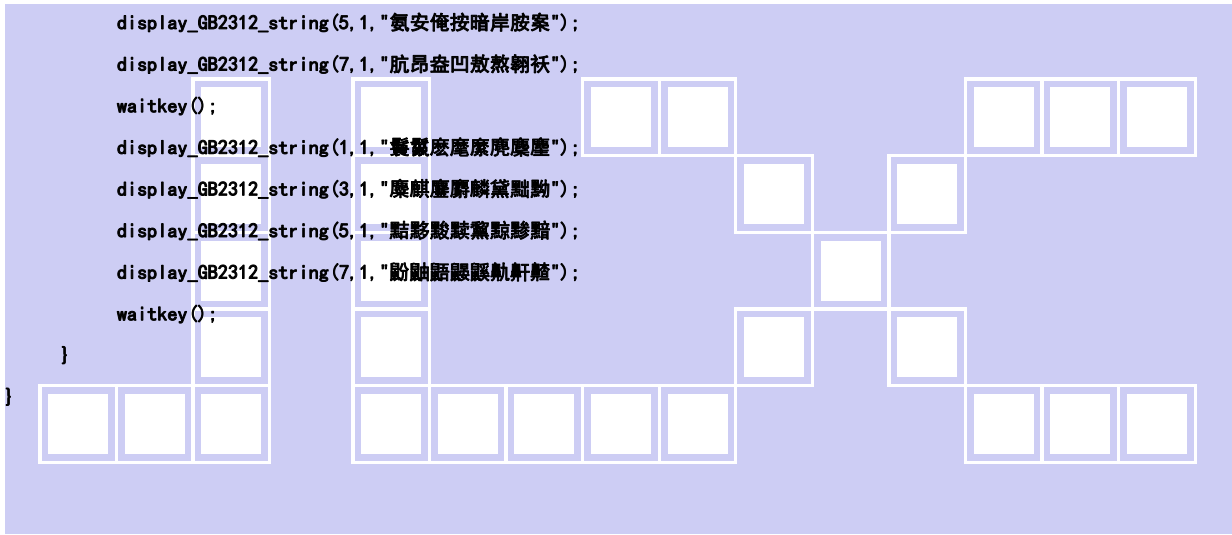
void main(void)
{
    while(1)
    {
        initial_lcd(); //初始化
        Rom_CS=1;
        lcd_cs1=0;
        clear_screen(); //clear all dots
        display_GB2312_string(1, 1, "JLXOLEDG-242-PC "); /*在第 1 页, 第 1 列, 显示一串 16x16 点阵汉字或 8x16 的 ASCII 字*/
        display_GB2312_string(3, 1, "16X16 简体汉字库,"); /*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 5X7 点阵 ASCII 码");
        waitkey();
        clear_screen();
        display_GB2312_string(1, 1, "晶联讯成立于二零");
        display_GB2312_string(3, 1, "零三年十一月七日");
        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");
        waitkey();
        display_GB2312_string(1, 1, "GB2312 简体字库及");
    }
}

```

```

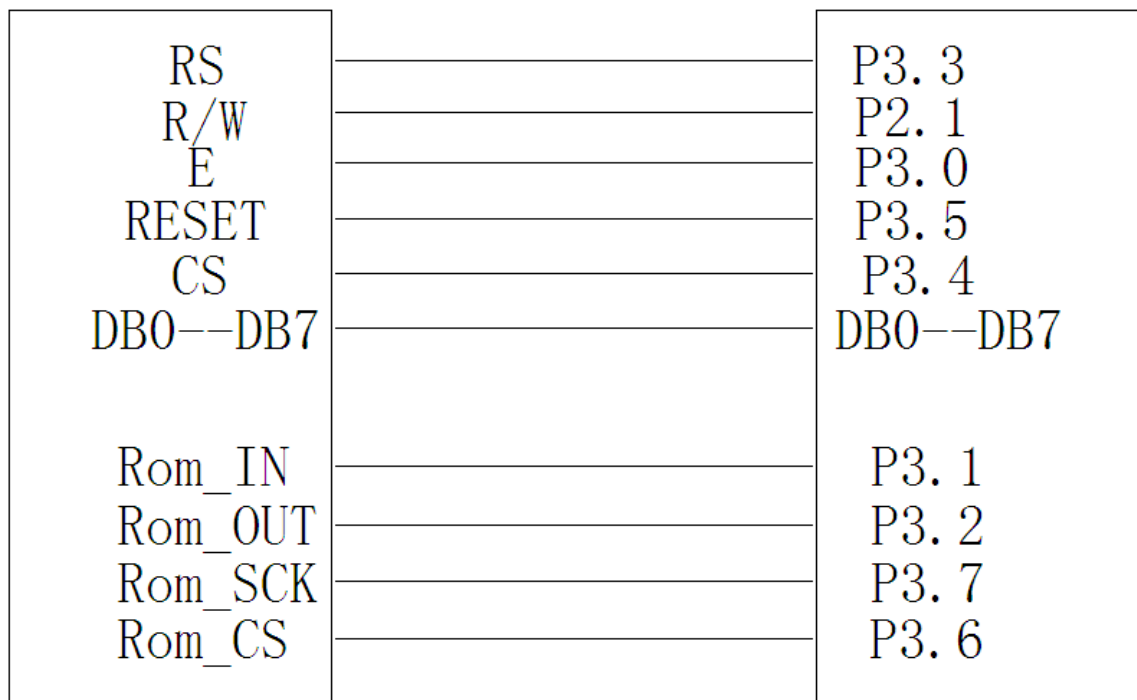
display_GB2312_string(3, 1, "有图型功能, 可自");
display_GB2312_string(5, 1, "编大字或图像或生");
display_GB2312_string(7, 1, "僻字, 例如:   ");
// display_graphic_16x16(7, 97, jiong1);           /*在第 7 页, 第 81 列显示单个自编生僻汉字"囧"*/
// display_graphic_16x16(7, 113, lei1);           /*显示单个自编生僻汉字"囧"*/
waitkey();
clear_screen();
display_GB2312_string(1, 1, "<!@#%^&*0_+]/"); /*在第 1 页, 第 1 列, 显示一串 16x16 点阵汉字或 8x16 的 ASCII 字*/
display_string_5x7(3, 1, "<!@#%^&*0_+]/;.:?["/*在第 3 页, 第 1 列, 显示一串 5x7 点阵的 ASCII 字*/
display_string_5x7(4, 1, "JLX electronics Co., "); /*显示一串 5x7 点阵的 ASCII 字*/
display_string_5x7(5, 1, "Ltd. established at "); /*显示一串 5x7 点阵的 ASCII 字*/
display_string_5x7(6, 1, "year 2004. Focus LCM. "); /*显示一串 5x7 点阵的 ASCII 字*/
display_string_5x7(7, 1, "TEL:0755-29784961-809"); /*显示一串 5x7 点阵的 ASCII 字*/
display_string_5x7(8, 1, "FAX:0755-29784964   "); /*显示一串 5x7 点阵的 ASCII 字*/
waitkey();
display_GB2312_string(1, 1, "啊阿埃挨哎哀皑"); /*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, "癌薨矮艾碍爱隘鞅"); /*显示一串 16x16 点阵汉字或 8x16 的 ASCII 字. 以下雷同*/
display_GB2312_string(5, 1, "氦安俺按暗岸胺案");
display_GB2312_string(7, 1, "肮昂盎凹敖熬翱袄");
waitkey();
display_GB2312_string(1, 1, "囊囊麽麽糜糜糜糜");
display_GB2312_string(3, 1, "麋麋麋麋麟麟麟麟");
display_GB2312_string(5, 1, "黠黠黠黠黠黠黠黠");
display_GB2312_string(7, 1, "黠黠黠黠黠黠航航");
waitkey();
}
}

```



7.3 当 LCD 驱动 IC 采用并行接口方式时的硬件设计及例程:

7.3.1 硬件接口: 下图为并行方式的硬件接口:



7.3.2、以下为并行接口方式范例程序

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：

```

//=====接口定义=====
sbit cs1=P3^4; /*3.4 接口定义*/
sbit reset=P3^5; /*3.3 接口定义*/
sbit rs=P3^3; /*接口定义*/
sbit rd=P3^0; /*接口定义*/
sbit wr=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*/

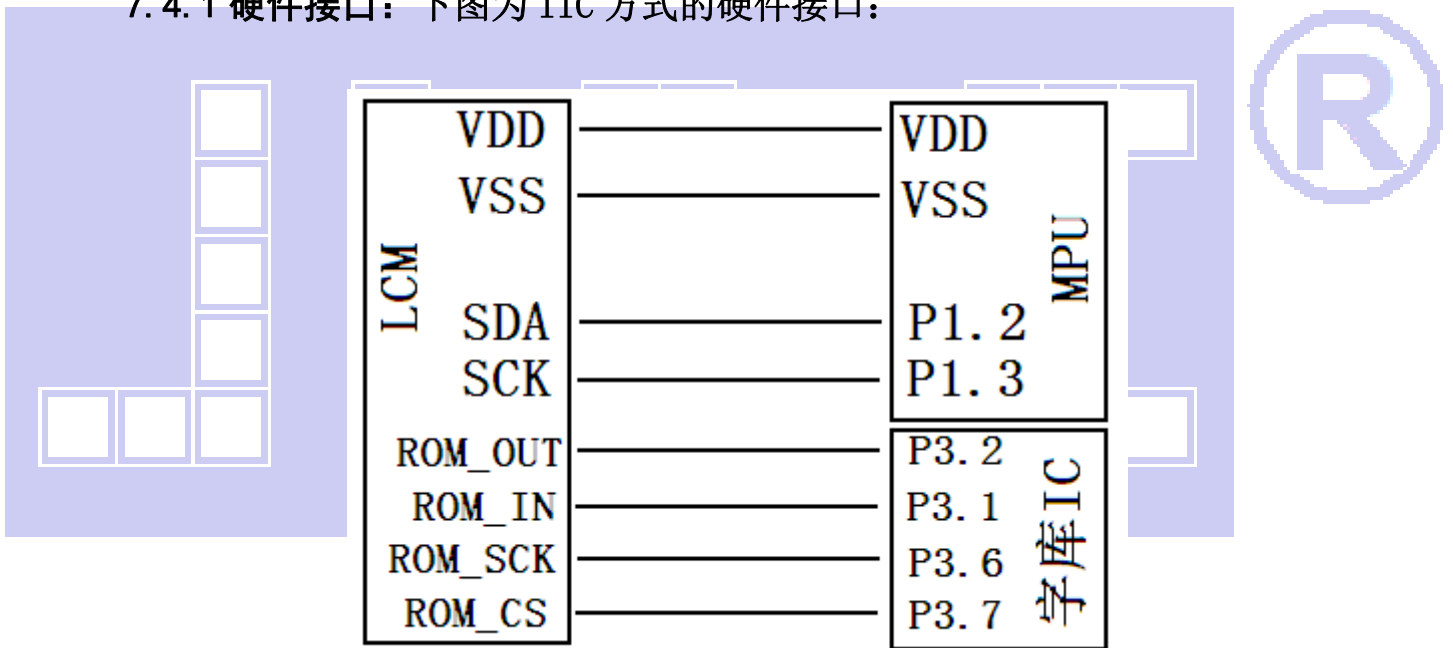
//=====transfer command to OLED=====
void transfer_command_lcd(int data1)
{
    cs1=0;
    rs=0;
    rd=0;
    wr=0;
    P1=data1;
    rd=1;
    cs1=1;
    rd=0;
}
    
```



```
//-----transfer data to OLED-----
void transfer_data_lcd(int data1)
{
    csl=0;
    rs=1;
    rd=0;
    wr=0;
    P1=data1;
    rd=1;
    csl=1;
    rd=0;
}
```

7.4 当 LCD 驱动 IC 采用 IIC 接口方式时的硬件设计及例程:

7.4.1 硬件接口: 下图为 IIC 方式的硬件接口:



7.4.2、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：

```
//=====接口定义=====
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*/
//=====
void start_flag()
{
```

```

    lcd_scl=1;
    lcd_sda=1;
    lcd_sda=0;
    lcd_scl=0;
}

```

```

void stop_flag()
{
    lcd_scl=0;
    lcd_sda=0;
    lcd_sda=1;
    lcd_scl=1;
}

```

//传 8 位指令或数据到 OLED 显示模块

```

void transfer(uchar data1)

```

```

{
    unsigned char j;
    for(j=0;j<8;j++)
    {
        lcd_scl=0;
        if(data1&0x80) lcd_sda=1;
        else lcd_sda=0;
        data1<<=1;
        lcd_scl=1;
    }
    lcd_sda =0;
    lcd_scl =0;
    lcd_scl =1;
}

```



//写指令到 OLED 显示模块

```

void transfer_command_lcd(u8 data1)

```

```

{
    start_flag();
    transfer(0x78); //7a
    transfer(0x00);
    transfer(data1);
    stop_flag();
}

```

//写数据到 OLED 显示模块

```

void transfer_data_lcd(u8 data1)

```

```

{
    start_flag();
    transfer(0x78); //7a
    transfer(0x40);
}

```



```
transfer(data1);
```

```
stop_flag0;
```

```
}
```

-END-

