

JLX19264G-901-PC

带字库 IC 的编程说明书

目 录

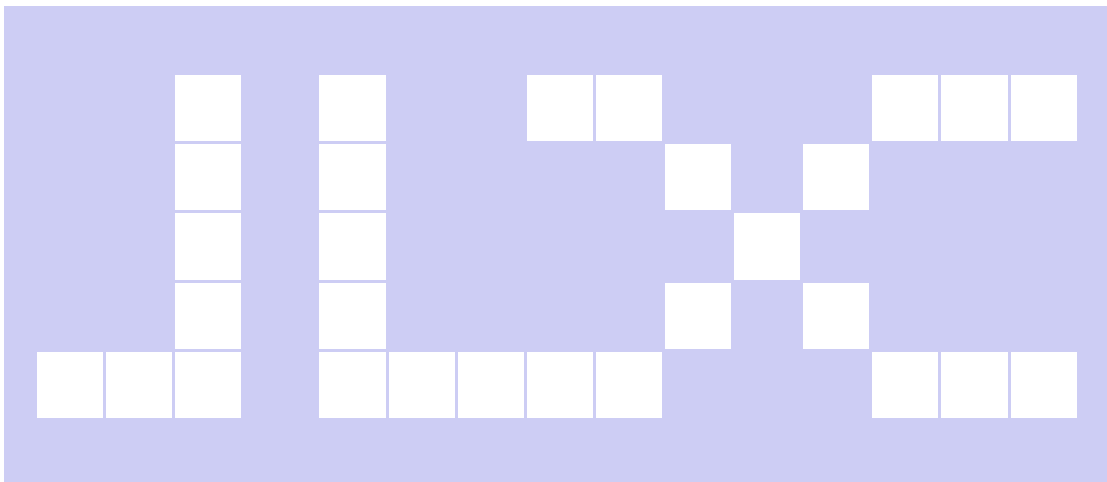
序号	内 容 标 题	页 码
1	概述	2
2	字型样张:	3
3	外形尺寸及接口引脚功能	4~5
4	工作电路框图	5~6
5	指令	6~8
6	字库的调用方法	9~17
7	硬件设计及例程:	18~末页

1. 概述

JLX19264G-901-PC 型液晶显示模块既可以当成普通的图像型液晶显示模块使用(即显示普通图像型的单色图片功能),又含有 JLX-GB2312 字库 IC, 可以从字库 IC 中读出内置的字库的点阵数据写入到 LCD 驱动 IC 中, 以达到显示汉字的目的。

此字库 IC 存储内容如下表所述:

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



2. 字型样张:

15X16 点 GB2312 汉字

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

8x16 点国标扩展字符

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

5x7 点 ASCII 字符

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

7x8 点 ASCII 字符

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

8x16 点 ASCII 字符

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

8x16 点 ASCII 粗体字符

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

16 点阵不等宽 ASCII 方头

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

16 点阵不等宽 ASCII 白正

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

3. 外形尺寸及接口引脚功能

3.1 外形图:

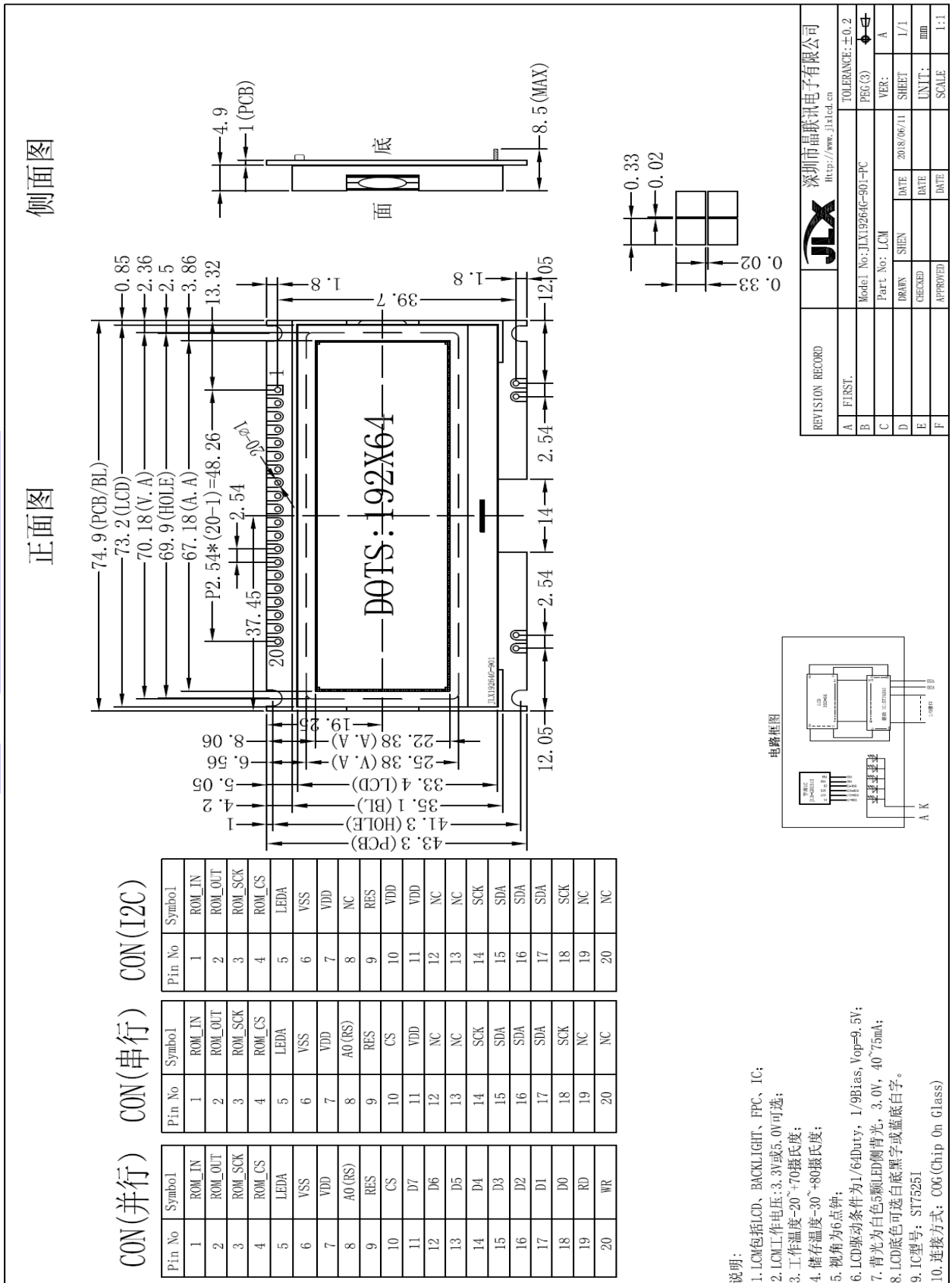


图 1. 外形尺寸

3.2 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
4	ROM_CS	字库 IC 接口 CS#	片选输入
5	LEDA	背光电源	供电电源正极 (同 VDD 电压)
6	VSS	接地	0V
7	VDD	电路电源	供电电源正极接 (5V 或 3.3V 购买时须选择 3.3V 还是 5.0 供电)
8	A0(RS)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器 (IC 资料上所写为 “CD”) IIC 接口时: 悬空
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 IIC 接口时: 接 VDD
11	D7	I/O	并行接口时: 数据总线 DB7 IIC/串行接口时: 接 VDD 或悬空
12	D6	I/O	并行接口时: 数据总线 DB6 IIC/串行接口时: 接 VDD 或悬空
13	D5	I/O	并行接口时: 数据总线 DB5 IIC/串行接口时: 接 VDD 或悬空
14	D4	I/O	并行接口时: 数据总线 DB4 IIC/串行接口时: 为 SCK 串行时钟 (D0 和 D4 短接)
15	D3	I/O	并行接口时: 数据总线 DB3 IIC/串行接口时: 为 SDA 串行数据 (D1、D2、D3 短接)
16	D2	I/O	并行接口时: 数据总线 DB2 IIC/串行接口时: 为 SDA 串行数据 (D1、D2、D3 短接)
17	D1	I/O	并行接口时: 数据总线 DB1 IIC/串行接口时: 为 SDA 串行数据 (D1、D2、D3 短接)
18	D0	I/O	并行接口时: 数据总线 DB0 IIC/串行接口时: 为 SCK 串行时钟 (D0 和 D4 短接)
19	E(/RD)	6800 时序: 使能 8080 时序: 读	并行接口时并且选择 6800 时序时: 使能信号, 高电平有效. 并行接口时并且选择 8080 时序时: 读数据, 低电平有效. IIC/串行接口时: 悬空
20	R/W(/WR)	6800 时序: 读/写 8080 时序: 写	并行接口时并且选择 6800 时序时: H: 读数据 L: 写数据 并行接口时并且选择 8080 时序时: 写数据, 低电平有效. IIC/串行接口时: 悬空

表 1: 模块接口引脚功能

4. 工作电路框图:

见图 2, 模块由 LCD 驱动 IC ST7525、字库 IC、背光组成。

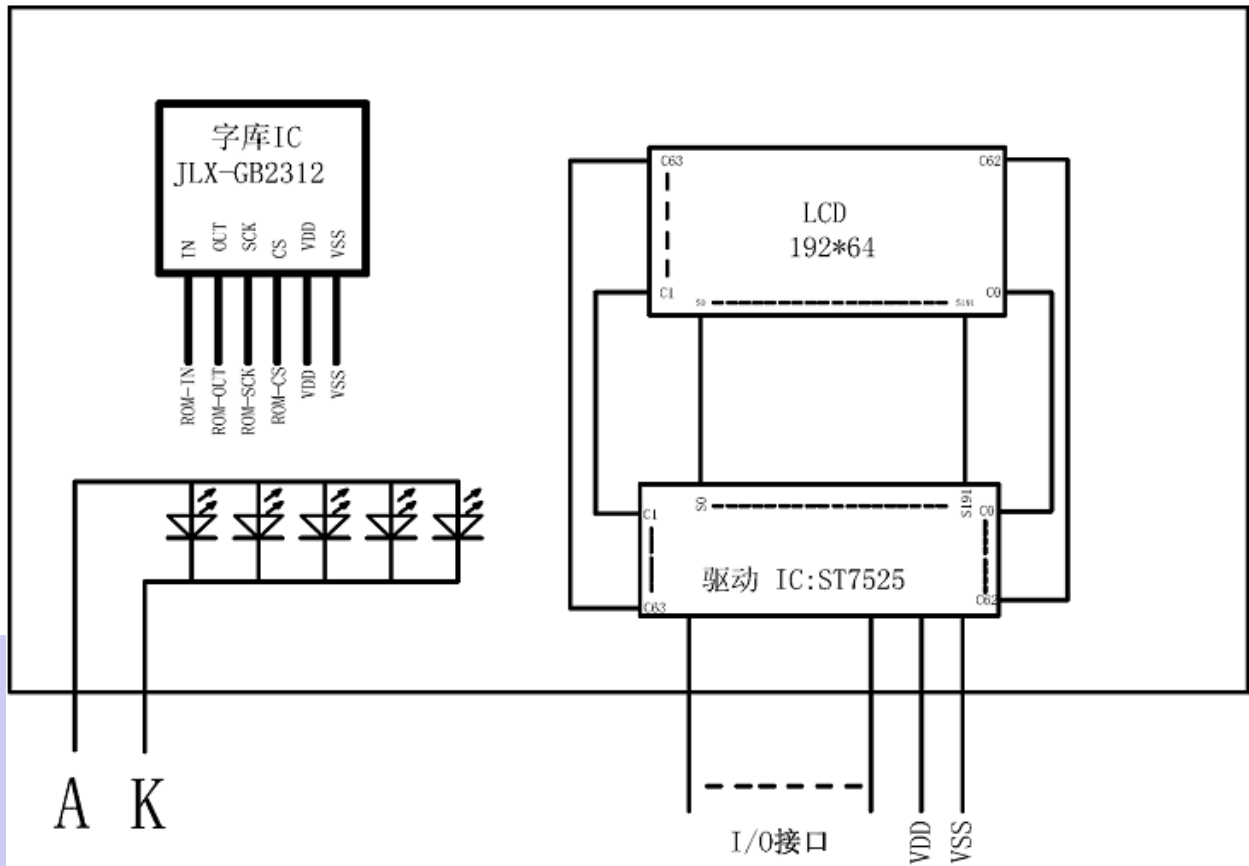


图 2: 电路框图

5. 指令:

5.1 字库 IC (JLX-GB2312) 指令表

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

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

Read Data Bytes (一般读取):

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

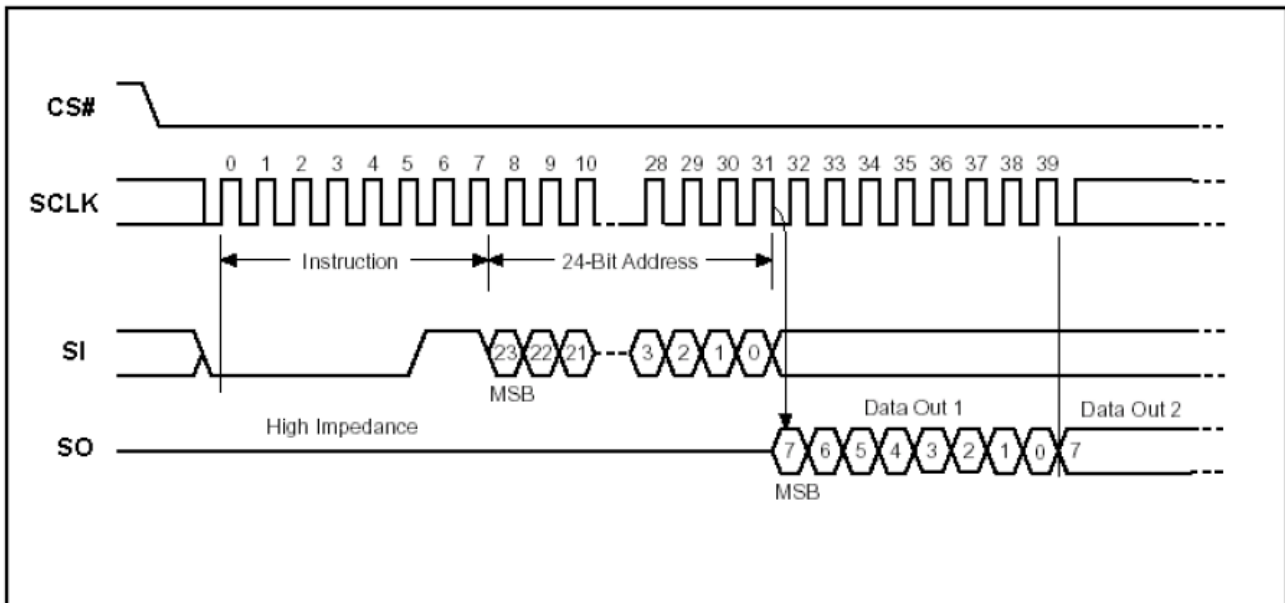
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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

■读取字节数据后, 则把片选信号 (CS#) 变为高, 结束本次操作。

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

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



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

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

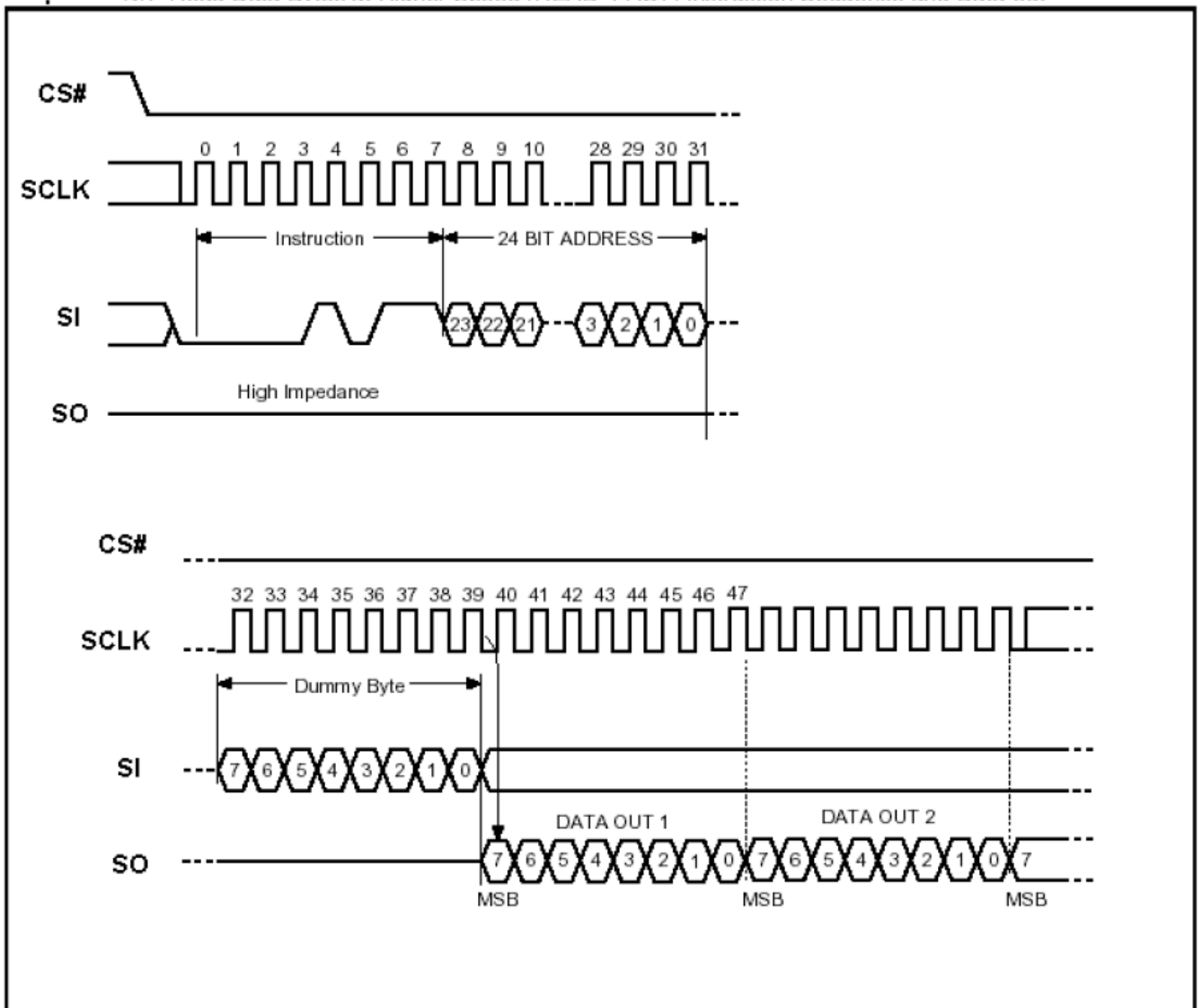
■首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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

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

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

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



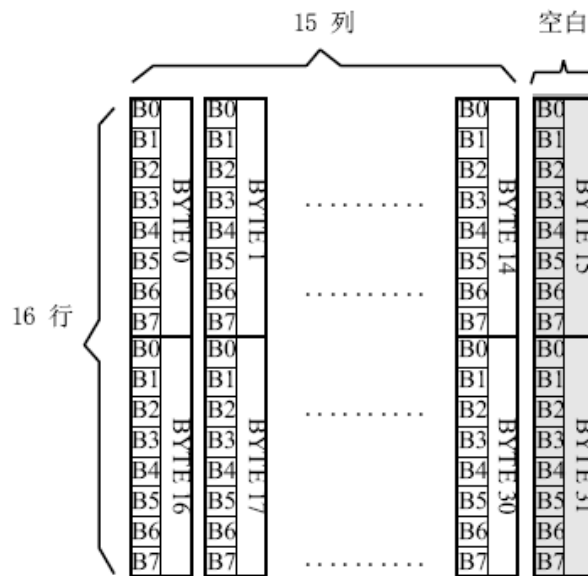
6 字库调用方法

6.1 汉字点阵排列格式

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

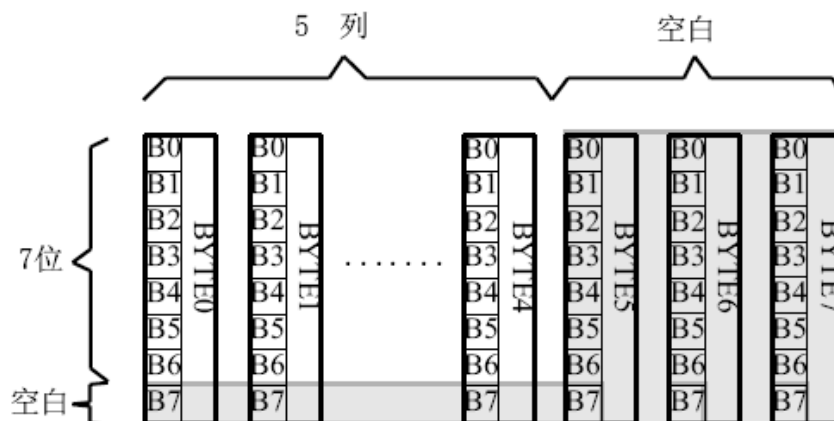
6.1.1 15X16 点汉字排列格式

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



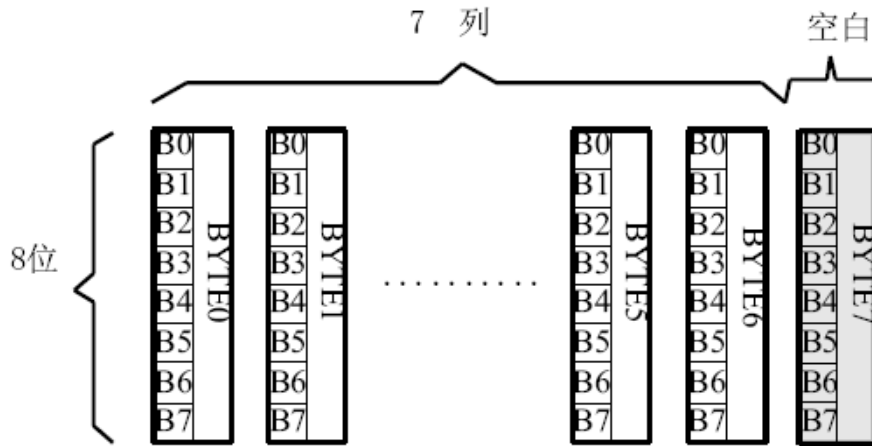
6.1.2 5X7 点 ASCII 字符排列格式

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



6.1.3 7X8 点 ASCII 字符排列格式

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



6.1.4 8X16 点字符排列格式

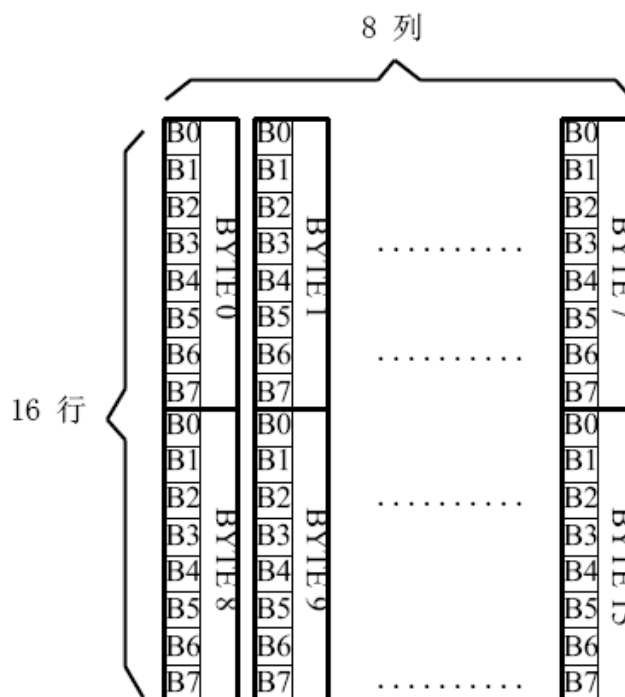
适用于此种排列格式的字有:

8X16 点 ASCII 字符

8X16 点 ASCII 粗体字符

8X16 点国标扩展字符

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



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

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

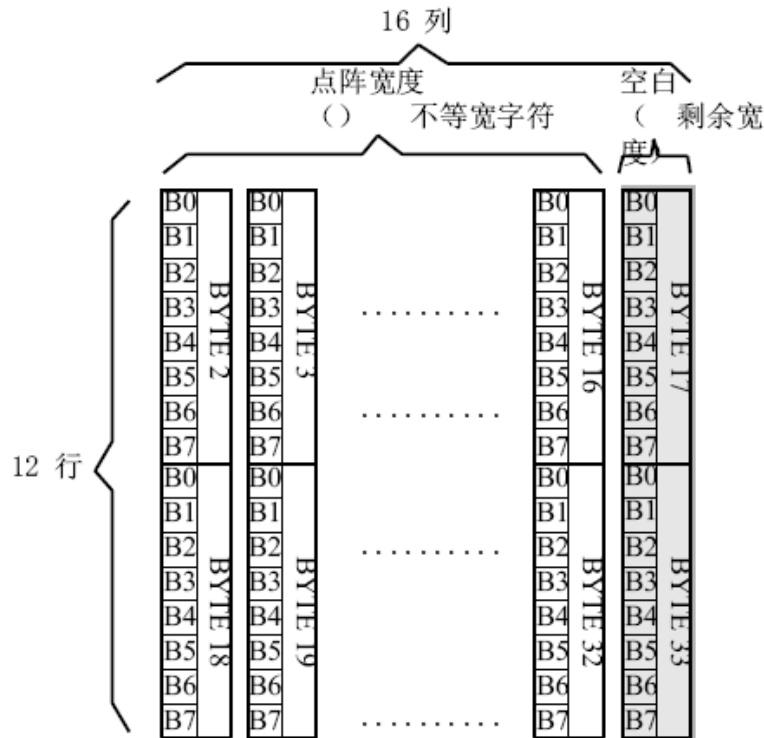
■ 存储格式

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



■ 存储结构

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



例如: ASCII

方头字符

B

0-33BYTE 的点阵数据是: 00 0C 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 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 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 字符在芯片中的地址计算方法

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

6.3.1 汉字字符的地址计算

6.3.1.1 15X16 点 GB2312 标准点阵字库

参数说明：

GBCode表示汉字内码。

MSB 表示汉字内码GBCode 的高8bits。

LSB 表示汉字内码GBCode 的低8bits。

Address 表示汉字或ASCII字符点阵在芯片中的字节地址。

BaseAdd: 说明点阵数据在字库芯片中的起始地址。

计算方法：

BaseAdd=0;

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

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

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

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

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

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

6.3.1.2 8X16 点国标扩展字符

说明:

BaseAdd: 说明本套字库在字库芯片中的起始字节地址。

FontCode: 表示字符内码 (16bits)

ByteAddress: 表示字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3b7d0

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

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

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

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

6.3.2 ASCII 字符的地址计算

6.3.2.1 5X7 点 ASCII 字符

参数说明:

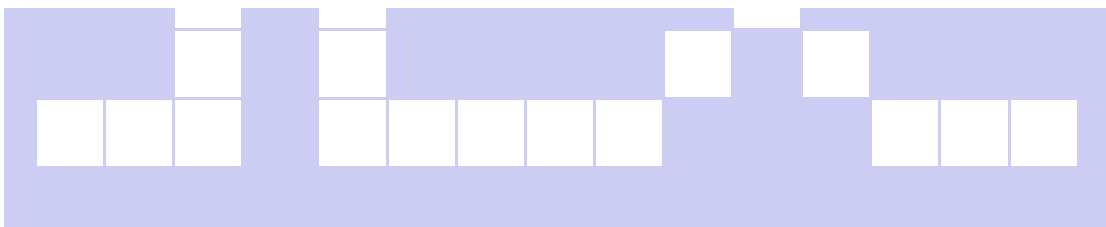
ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3bfc0



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

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

6.3.2.2 7X8 点 ASCII 字符

参数说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x66c0

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

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

6.3.2.3 8X16 点 ASCII 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

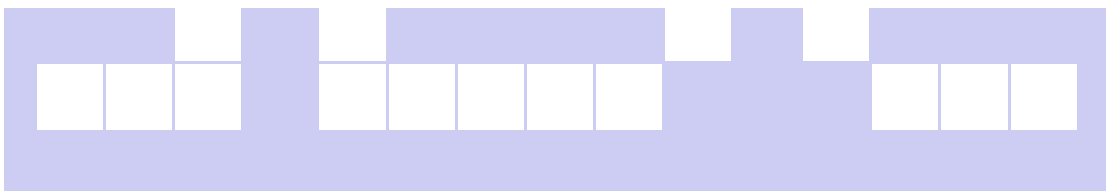
Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3b7c0

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

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



6.3.2.4 16 点阵不等宽 ASCII 方头 (Arial) 字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3c2c0

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
    Address = (ASCIICode -0x20 ) * 34 + BaseAdd
```

6.3.2.5 8X16 点 ASCII 粗体字符

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3cf80

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
    Address = (ASCIICode -0x20 ) * 16+BaseAdd
```

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

说明:

ASCIICode: 表示 ASCII 码 (8bits)

BaseAdd: 说明该套字库在芯片中的起始地址。

Address: ASCII 字符点阵在芯片中的字节地址。

计算方法:

BaseAdd=0x3d580

```
if (ASCIICode >= 0x20) and (ASCIICode <= 0x7E) then
    Address = (ASCIICode -0x20 ) * 34 + BaseAdd
```

6.4 附录

6.4.1 GB2312 1 区 (376 字符)

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

GB2312 1 区

A1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A			、	。	·	-	√	”	々	一	~		…	‘	’	
B	“	”	{	}	<	>	《	》	「	」	『	』	【	】	【	】
C	±	×	÷	:	∧	∨	Σ	Π	U	∩	€	::	√	⊥	//	∠
D	∩	⊙	∫	∫	≡	≈	≈	∞	∞	≠	≠	≠	≠	∞	:	:
E	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴	∴
F	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	=	

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

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

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

6.4.2 8×16点国标扩展字符

内码组成为 AAA1~ABC0 共计 126 个字符

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

A		!	"	#	¥	%	&	†	()	*	+	,	-	.	/
B	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
C	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
D	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
E	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
F	p	q	r	s	t	u	v	w	x	y	z	{		}	~	

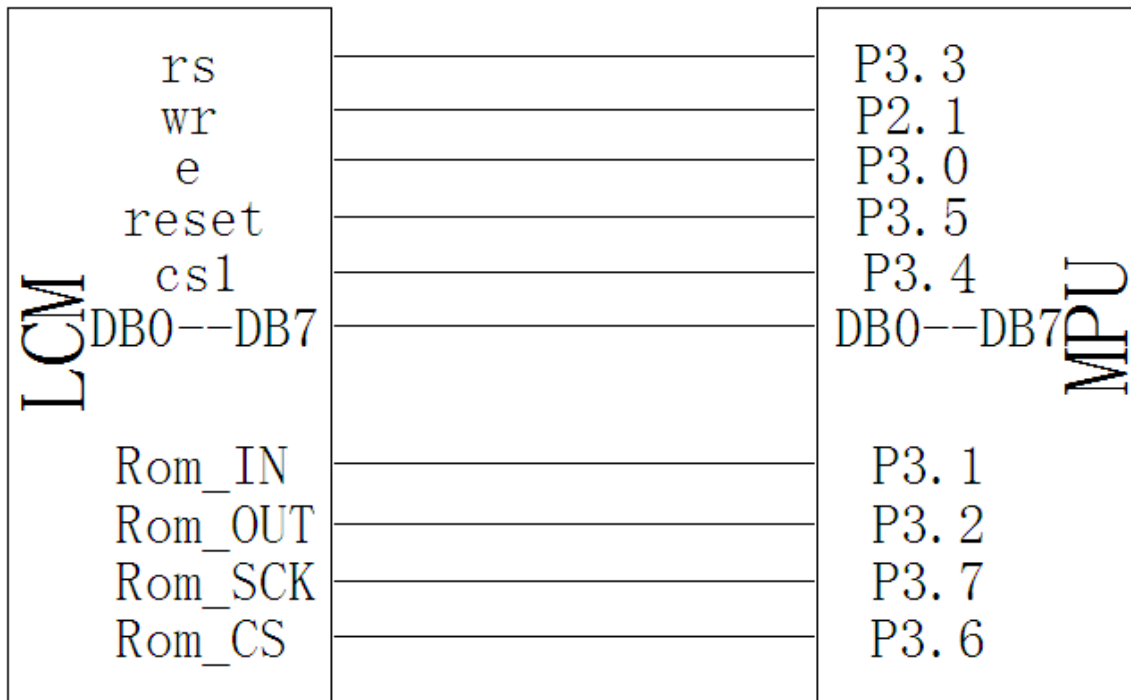
AB 0 1 2 3 4 5 6 7 8 9 A B C D E F

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

7. 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:

7.1.1 硬件接口: 下图为并行方式的硬件接口:



7.1.2 例程: 以下为并行方式显示汉字及 ASCII 字符的例程:

```
// JLX19264G-901-PC-P
// 并行接口
// 字库 IC 是:JLX-GB2312
// 驱动 IC 是:ST7525

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

sbit cs1=P3^4;    /*接口定义*/
sbit reset=P3^5; /*接口定义*/
sbit rs=P3^3;    /*接口定义*/
sbit e=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 Rom_SCK=P3^7;
sbit Rom_OUT=P3^2;
sbit Rom_IN=P3^1;
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 DataBus P1
```

```
//延时 1
```

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

```
//=====transfer command to LCM=====
```

```
void transfer_command(int data1)
```

```
{
    cs1=0;
    rs=0;
    wr=0;
    e=0;
    DataBus=data1;
    e=1;
    e=0;
    DataBus=0x00;
    cs1=1;
}
```



```
//-----transfer data to LCM-----
```

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

```
{
    cs1=0;
    rs=1;
    wr=0;
    e=0;
    DataBus=data1;
    e=1;
    e=0;
    DataBus=0x00;
    cs1=1;
}
```

```
void waitkey()
```

```

{
repeat:
    if(key==1) goto repeat;
    else delay(2500);
}

//LCD 模块初始化
void initial_lcd()
{
    reset=0;          //低电平复位
    delay(100);
    reset=1;          //复位完毕
    delay(200);
    transfer_command(0xe2); //软复位
    delay(200);
    transfer_command(0x2f); //打开内部升压
    delay(200);
    transfer_command(0xa0); //
    transfer_command(0x81); //微调对比度
    transfer_command(0x70); //微调对比度的值, 可设置范围 0x00~0xFF
    transfer_command(0xeb); //1/9 偏压比 (bias)
    transfer_command(0xc4); //行列扫描顺序: 从上到下
    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(i=0; i<8; i++)
    {
        lcd_address(1+i, 1);
        for(j=0; j<192; j++)
        {
            transfer_data(0x00);

```



```

    }
}
}

```

```

void display_graphic_192x64(uchar *dp)
{
    uchar i, j;
    for(i=0;i<8;i++)
    {
        lcd_address(i+1,1);
        for(j=0;j<192;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}

```

//显示 16x16 点阵图像、汉字、生僻字或 16x16 点阵的其他图标

```

void display_graphic_16x16(uchar page,uchar column,uchar *dp)
{
    uchar i, j;
    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++;
        }
    }
}

```



//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 5x7 点阵图像、ASCII, 或 8x16 点阵的自造字符、其他图标
void display_graphic_5x7(uchar page,uchar column,uchar *dp)
{
    uchar i, j;
    for(j=0; j<1; 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++;
        }
    }
}

```

/*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;
```

```
    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))
```

```
{
```

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

```
{
```

```
        /*国标简体（GB2312）汉字在字库 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]>=0x20) &&(text[i]<=0x7e))
    {
        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+=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 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;
    }
    else
        i++;
}
}

void main(void)
{
    while(1)
    {
        initial_lcd();
        clear_screen(); //clear all dots
        display_graphic_192x64(bmp2);
        waitkey();
        clear_screen(); //clear all dots
        display_graphic_192x64(bmp1);
        waitkey();
        clear_screen();
        display_GB2312_string(1, 1, "JLX19264G-901, 带中文字库");
        display_GB2312_string(3, 1, "16X16 简体汉字库, 或 8X16 点");
        display_GB2312_string(5, 1, "阵 ASCII, 或 5x8 点阵 ASCII 码");
        display_GB2312_string(7, 1, "{<!@#$$%^&*()_+]/:.,?[]>}");
        waitkey();
        display_GB2312_string(1, 1, "GB2312 简体字库及有图型功");
        display_GB2312_string(3, 1, "能, 可自编大字或图像或生");
        display_GB2312_string(5, 1, "僻字, 例如: 癌藹矮艾碍爰隘");
        display_GB2312_string(7, 1, "鞍氨安俺按暗岸胺案肮昂盎");
        waitkey();
        display_GB2312_string(1, 1, "深圳晶联电子有限公司讯成");
        display_GB2312_string(3, 1, "立于二零零四年十一月七日");
        display_GB2312_string(5, 1, "主要生产销售具有高科技的");
        display_GB2312_string(7, 1, "液晶模块品质至上真诚服务");
        waitkey();
        clear_screen();
        display_GB2312_string(1, 1, " 90123456789abcdef_~!@#$"); //% /*在第 1 页，第 1 列，显示一串
16x16 点阵汉字或 8*16 的 ASCII 字*/
        display_GB2312_string(3, 1, "! #$$%^&'()*+,-./012345678");
        display_string_5x7(5, 1, "0123456789abcdef~!@#$$%^&*()_+]/:.,?[]>}"); /*显示一串 5x7 点阵的 ASCII 字*/
    }
}

```

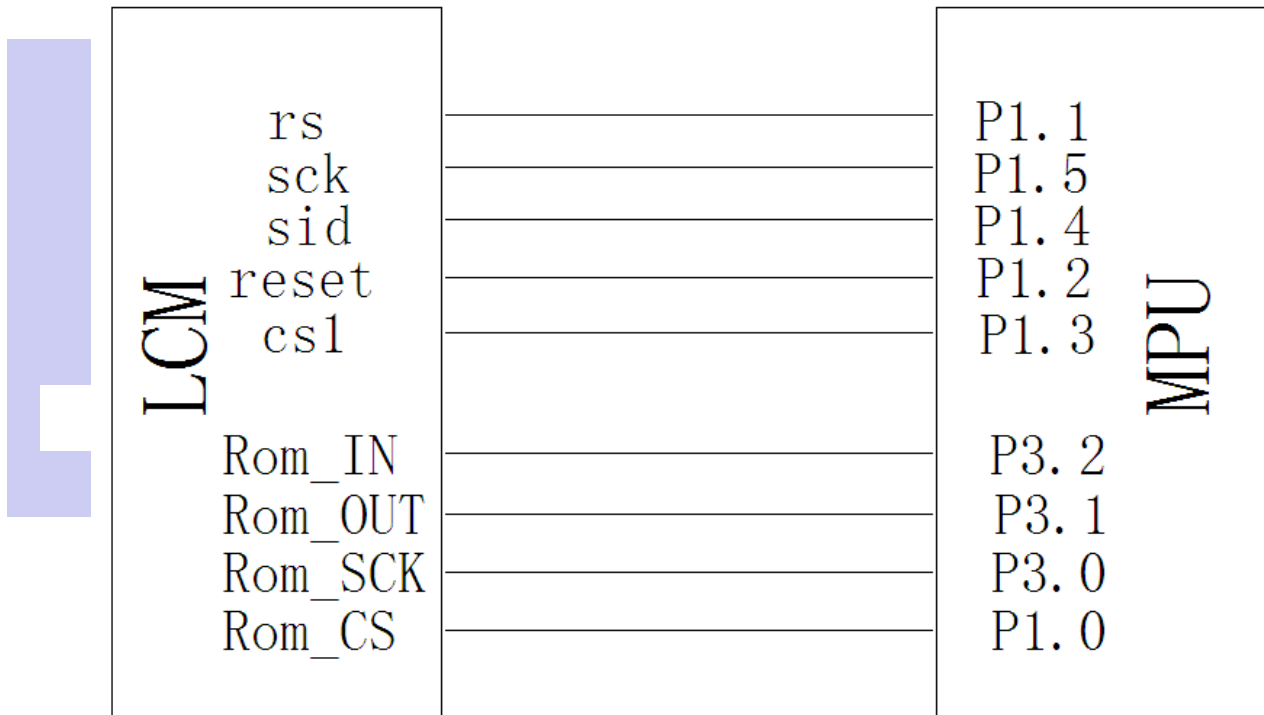


```

display_string_5x7(6,1,"JLX electronics Co.,Ltd. ");/*显示一串 5x7 点阵的 ASCII 字*/
display_string_5x7(7,1,"etd at year 2004.Focus LCM. ");/*显示一串 5x7 点阵的 ASCII 字*/
display_string_5x7(8,1,"www.jlxlcd.cn tel:0755-29784961 ");/*显示一串 5x7 点阵的 ASCII 字*/
waitkey();
clear_screen();
display_GB2312_string(1,1,"晶联讯");
display_GB2312_string(3,1,"      晶联讯");
display_GB2312_string(5,1,"      晶联讯");
display_GB2312_string(7,1,"      晶联讯");
waitkey();
}
}

```

7.1.3 硬件接口： 下图为串行方式的硬件接口：



7.1.4 例程： 以下为串行方式显示汉字及 ASCII 字符的例程：

串行程序与并行只是接口定义、写数据和命令不一样，其它都一样

```

// 液晶演示程序 JLX19264G-901，并行接口！
// 驱动 IC 是:ST7525I

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

```

```
sbit cs1=P1^3;
sbit reset=P1^2;
sbit rs=P1^1;
sbit sclk=P1^5;
sbit sid=P1^4;
sbit key=P2^0;
```

```
sbit Rom_SCK=P3^0;
sbit Rom_OUT=P3^1;
sbit Rom_IN=P3^2;
sbit Rom_CS=P1^0; /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#*/
```

//写指令到 LCD 模块

```
void transfer_command(int data1)
```

```
{
    char i;
    cs1=0;
    rs=0;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
    cs1=1;
}
```



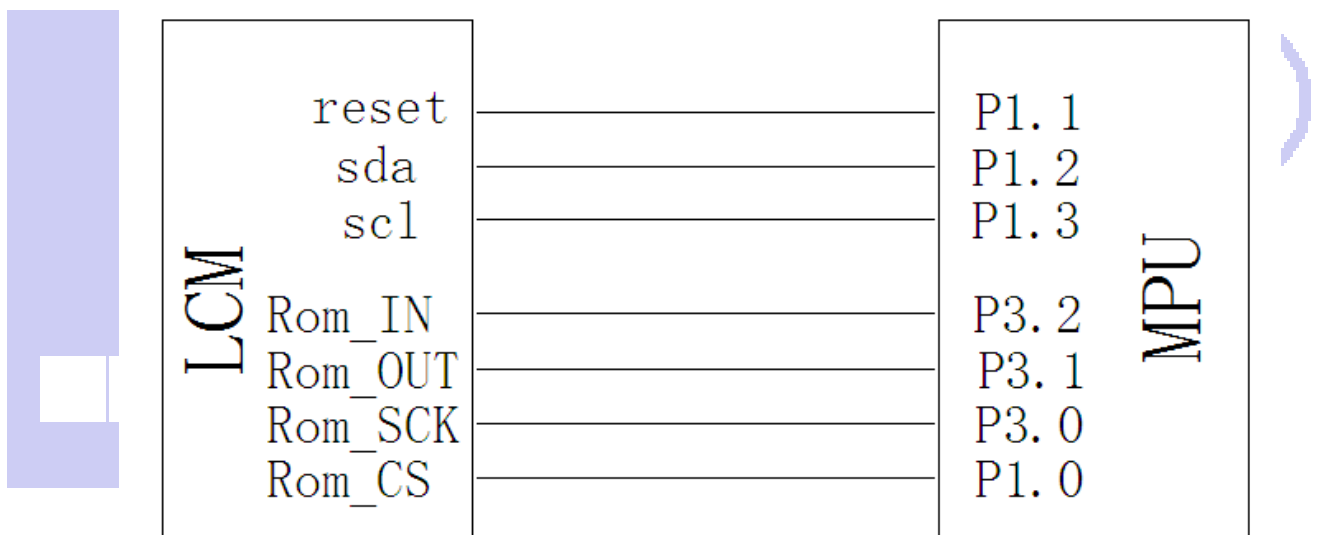
//写数据到 LCD 模块

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

```
{
    char i;
    cs1=0;
    rs=1;
    for(i=0;i<8;i++)
    {
        sclk=0;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
    cs1=1;
}
```

}

7.1.5 硬件接口: 下图为 IIC 方式的硬件接口:



7.1.6 例程: 以下为 IIC 方式显示汉字及 ASCII 字符的例程:

IIC 程序与串、并行接口定义、写数据和命令不一样，取模代码是一样的

// 液晶演示程序 JLX19264G-901, IIC 接口!

// 驱动 IC 是:ST7525I

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

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

```
sbit reset=P1^1;
```

```
sbit scl=P1^3;
```

```
sbit sda=P1^2;
```

```
sbit key=P2^0;

sbit Rom_SCK=P3^0;
sbit Rom_OUT=P3^1;
sbit Rom_IN=P3^2;
sbit Rom_CS=P1^0;

void delay_us(int i);
void delay(int i);
```

```
//延时 1
void delay(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<110;k++);
}
```

```
//延时 2
void delay_us(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<10;k++);
}
```

```
void waitkey()
{
    repeat:
        if(key==1) goto repeat;
        else delay(400);
}
```

```
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(0x7c);
    transfer(com);
    stop_flag();
}

```

//写数据到液晶显示模块

```

void transfer_data(uchar dat)

```

```

{
    start_flag();
    transfer(0x7e);
    transfer(dat);
    stop_flag();
}

```

