

JLX400-042-PC 使用说明书

(带字库 IC)

(IPS 全视角)

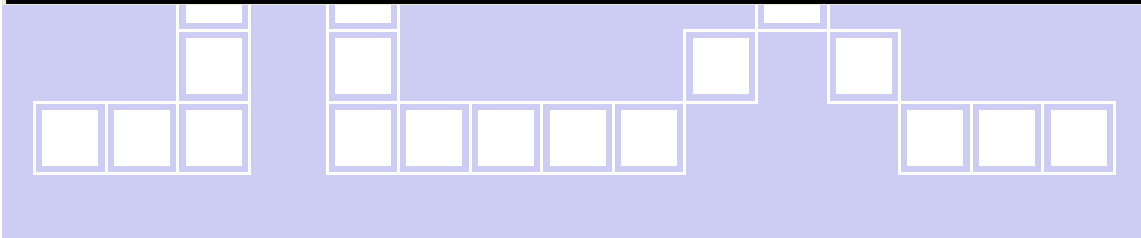
目 录

| 序号 | 内 容 标 题 | 页 码 |
|----|----------------|------|
| 1 | 字库 | 2~3 |
| 2 | 外形及接口引脚功能 | 4~5 |
| 3 | 基本原理 | 5 |
| 4 | 技术参数 | 5~6 |
| 5 | 指令功能及硬件接口与编程案例 | 6~末页 |

1. 字库

字库 IC(IC 型号: JLX-GB2312-3205, 此 IC 为可选的配件) 自带字库内容:

| 分类 | 字库内容 | 编码体系 (字符集) | 字符数 |
|--|--|------------|----------|
| 汉字字符 | 11X12 点 GB2312 标准点阵字库 | GB2312 | 6763+846 |
| | 15X16 点 GB2312 标准点阵字库 | GB2312 | 6763+846 |
| | 24X24 点 GB2312 标准点阵字库 | GB2312 | 6763+846 |
| | 32X32 点 GB2312 标准点阵字库 | GB2312 | 6763+846 |
| | 6X12 点国标扩展字符 | GB2312 | 126 |
| | 8X16 点国标扩展字符 | GB2312 | 126 |
| | 12X24 点国标扩展字符 | GB2312 | 126 |
| | 16X32 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 |
| | 8X16 点粗体 ASCII 字符 | ASCII | 96 |
| | 12 点阵不等宽 ASCII 方头 (Arial) 字符 | ASCII | 96 |
| | 12 点阵不等宽 ASCII 白正 (Times New Roman) 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 (Times New Roman) 字符 | ASCII | 96 |
| | 24 点阵不等宽 ASCII 方头 (Arial) 字符 | ASCII | 96 |
| | 24 点阵不等宽 ASCII 白正 (Times New Roman) 字符 | ASCII | 96 |
| | 32 点阵不等宽 ASCII 方头 (Arial) 字符 | ASCII | 96 |
| 32 点阵不等宽 ASCII 白正 (Times New Roman) 字符 | ASCII | 96 | |
| 输入法码表 | 全拼输入法码表 | GB2312 | |



字型样张

11X12 点 GB2312 汉字

啊阿埃挨哎唉哀皑癌藹矮艾碍爱隘鞍
 氨安俺按暗岸胺案肮盎凹敖熬翱袄
 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
 搬扳般颁板版扮拌伴瓣半办絆邦帮榔
 榜膀绑棒棒蚌傍傍苞胞包褒剥薄苞
 保堡宝抱报暴豹鲍爆杯碑悲卑北辈
 背贝狈倍狻备惫焙奔笨笨崩绷甬

15X16 点 GB2312 汉字

啊阿埃挨哎唉哀皑癌藹矮艾碍爱隘鞍
 氨安俺按暗岸胺案肮盎凹敖熬翱袄
 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
 搬扳般颁板版扮拌伴瓣半办絆邦帮榔
 榜膀绑棒棒蚌傍傍苞胞包褒剥薄苞
 保堡宝抱报暴豹鲍爆杯碑悲卑北辈
 背贝狈倍狻备惫焙奔笨笨崩绷甬

24X24 点 GB2312 汉字

啊阿埃挨哎唉哀皑
 癌藹矮艾碍爱隘鞍
 氨安俺按暗岸胺案
 肮盎盎凹敖熬翱袄

32X32 点 GB2312 汉字

啊阿埃挨哎唉
 哀皑癌藹矮艾
 碍爱隘鞍氨安

5x7 点 ASCII 字符

```
!"#$%&'()*+,-./0123456789:;  
=>?@ABCDEFGHIJKLMN O PQRSTU V  
WXY[\]^_`abcdefghijklmnopqrstuvwxyz
```

7x8 点 ASCII 字符

```
!"#$%&'()*+,-./01234  
56789:;<=>?@ABCDEFGHIJ  
KLMNOPQRSTUUVWXYZ[\]^_`  
bcdefghijklmnopqrstuv  
wxyz{|}~áâãäåæçèéíîïðóôõ
```

6x12 点 ASCII 字符

```
!"#$%&'()*+,-./0123456789:;  
=>?@ABCDEFGHIJKLMN O PQRSTU V W  
XYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~  
áâãäåæçèéíîïðóôõ
```

8x16 点 ASCII 字符

```
!"#$%&'()*+,-./0123456789:;  
=>?@ABCDEFGHIJKLMN O PQRSTU V  
WXY[\]^_`abcdefghijklmnopqrstuvwxyz
```

12 点阵不等宽 ASCII 方头

```
!"#$%&'()*+,-./01234  
56789:;<=>?@ABCDEFGHIJ  
KLMNOPQRSTUUVWXYZ[\]^_`  
bcdefghijklmnopqrstuv  
wxyz{|}~áâãäåæçèéíîïðóôõ
```

16 点阵不等宽 ASCII 方头

```
!"#$%&'()*+,-./0123456789:;  
=>?@ABCDEFGHIJKLMN O PQRSTU V W  
XYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~  
áâãäåæçèéíîïðóôõ
```

2. 外形尺寸及接口引脚功能

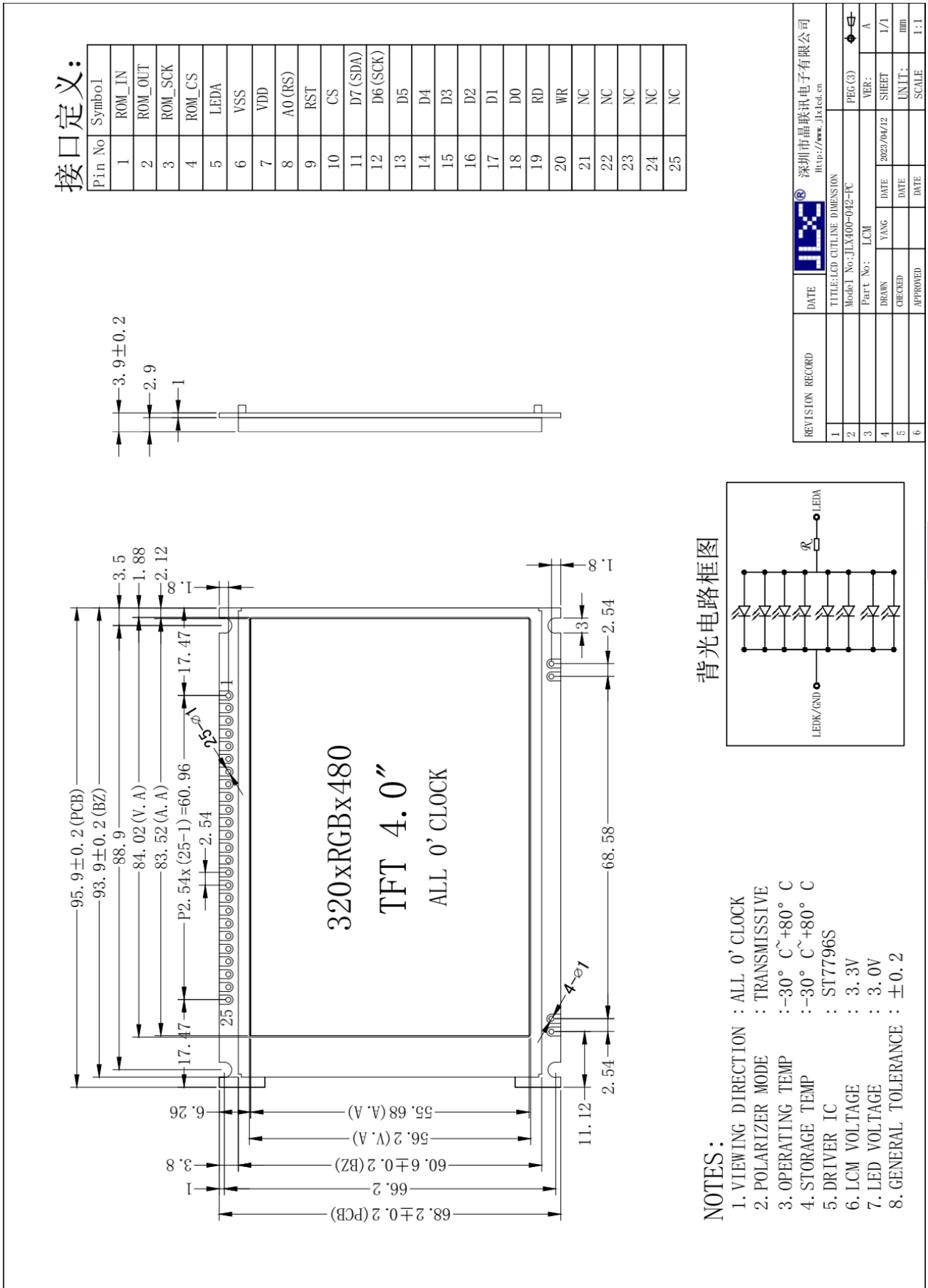


图 1. 外形尺寸

模块的接口引脚功能

表 1: 模块的接口引脚功能

| 引线号 | 符号 | 名称 | 功能 |
|-------|----------|----------|------------------------------------|
| 1 | ROM_IN | 字库 IC 接口 | 字库串行数据输入。 |
| 2 | ROM_OUT | 字库 IC 接口 | 字库串行数据输出。 |
| 3 | ROM_SCK | 字库 IC 接口 | 字库串行时钟。 |
| 4 | ROM_CS | 字库 IC 接口 | 字库片选输入。 |
| 5 | LEDA | 背光电源 | 背光电源正极, 同 VDD 电压 3.3V |
| 6 | VSS | 接地 | 0V |
| 7 | VDD | 电路电源 | 3.3V |
| 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 "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 | D7 | I/O | 并行时: 数据总线 DB7 串行时: 串行数据 (SDA) |
| 12 | D6 | I/O | 并行时: 数据总线 DB6 串行时: 串行时钟 (SCK) |
| 13-18 | D5-D0 | I/O | 并行时: 数据总线 DB0~DB5 串行时: 空 |
| 19 | E (RD) | 使能信号 | 并行时: 使能信号 串行时: 空 |
| 20 | WR (R/W) | 读/写 | 并行时: H: 读数据 0: 写数据 串行时: 空 |
| 21 | 空脚 | 空脚 | 空脚 |
| 22 | 空脚 | 空脚 | 空脚 |
| 23 | 空脚 | 空脚 | 空脚 |
| 24 | 空脚 | 空脚 | 空脚 |
| 25 | 空脚 | 空脚 | 空脚 |

3. 基本原理

3.1 背光参数

该型号液晶模块带 LED 背光源。它的性能参数如下:

工作温度: $-20 \sim +70^{\circ} \text{C}$;

存储温度: $-30 \sim +80^{\circ} \text{C}$;

背光板是白色。

正常工作电流为: $64 \sim 120 \text{mA}$ (LED 灯数共 8 颗, 每颗灯是 $8 \sim 15 \text{mA}$)

工作电压: 3.0V。 (PCB 已加限流电阻, LEDA 脚可以供电 3.3V)

4. 技术参数

4.1 最大极限参数 (超过极限参数则会损坏液晶模块)

| 名称 | 符号 | 标准值 | | | 单位 |
|------|-----|------|-----|-----|----|
| | | 最小 | 典型 | 最大 | |
| 电路电源 | VDD | -0.3 | 3.3 | 3.6 | V |

| | | | | | |
|------|--|-----|-----|-----|----|
| 工作温度 | | -20 | +25 | +70 | °C |
| 储存温度 | | -30 | +25 | +80 | °C |

表 2: 最大极限参数

4.2 直流 (DC) 参数

| 名称 | 符号 | 测试条件 | 标准值 | | | 单位 |
|--------|------|-----------------------------|-----|-----|-----|----|
| | | | 最小 | 典型值 | 最大 | |
| 工作电压 | VDD | | 2.7 | 3.3 | 3.6 | V |
| 背光工作电压 | VLED | | 2.6 | 2.8 | 3.0 | V |
| 背光工作电流 | ILED | VLED=3.0V, 共 3 颗 LED 灯并联 | 48 | 90 | 120 | mA |

表 3: 直流 (DC) 参数

4.3 LCD 驱动 IC 指令表详见“JLX400-042-PN”的中文说明书

5.1 字库 IC (JLX-GB2312-3205) 的操作指令及点阵数据的调用方法:

5.1.1 字库 IC 的操作指令只有两条, 两条只选一条进行使用, 操作指令表如下:

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 | at Higher Speed | 0000 1011 | 0B h | 3 | 1 | 1 to ∞ |

Read Data

Bytes

所有对本芯片 SPI 接口的操作只有 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”)。

以下分别介绍一般读取和快速读取:

5.2.1.1 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):

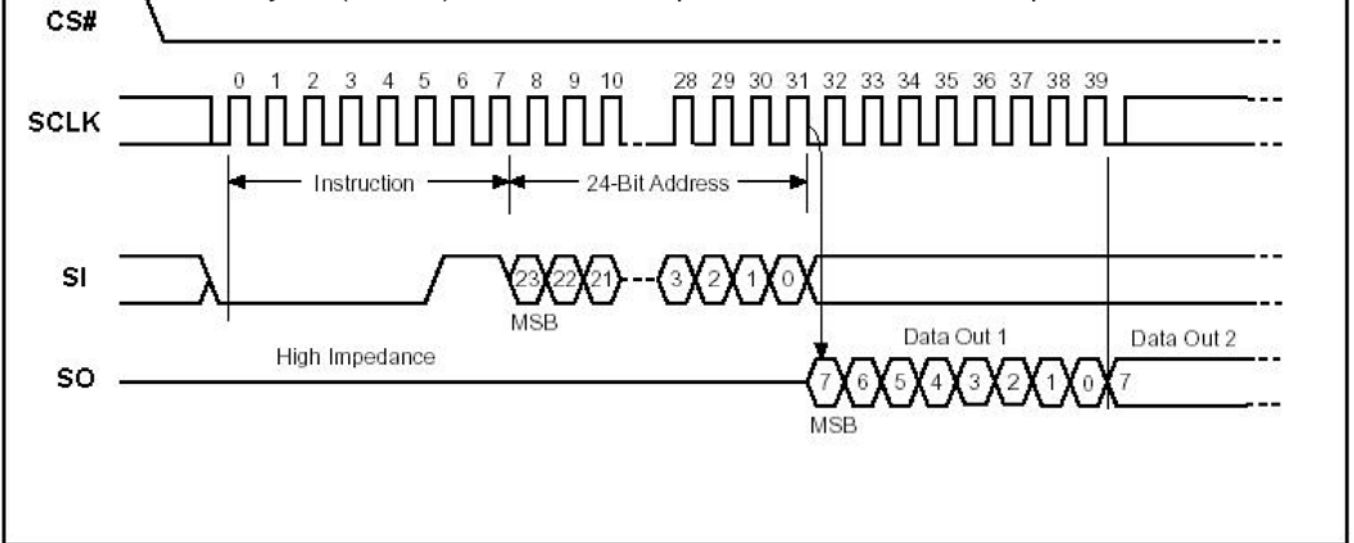
首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (03 h) 和 3 个字节的地址和通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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。

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



5.2.1.2 Read Data Bytes at Higher Speed (快速读取点阵数据)

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):

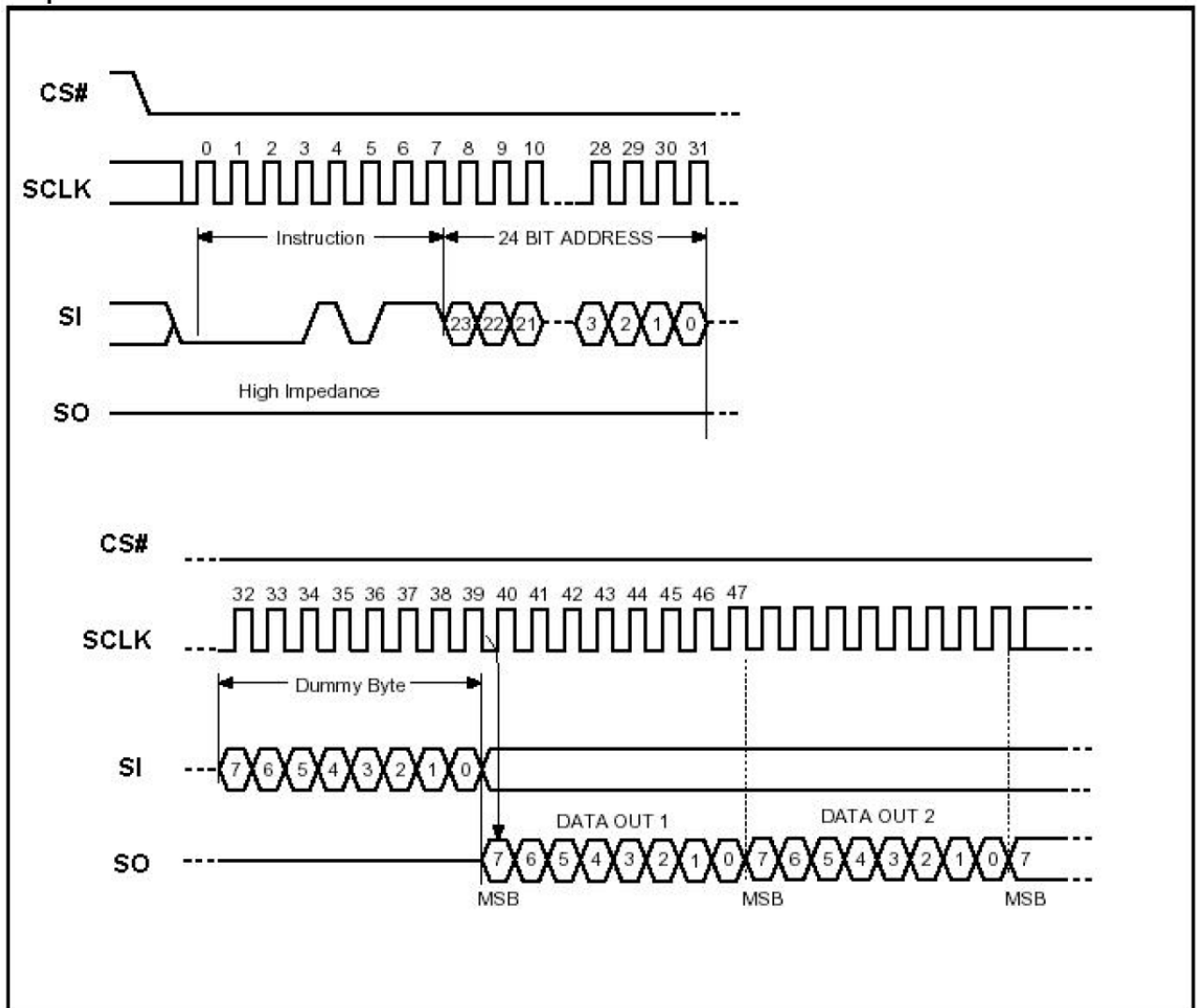
首先把片选信号 (CS#) 变为低, 紧跟着的是 1 个字节的命令字 (0B h) 和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚 (SI) 移位输入, 每一位在串行时钟 (SCLK) 上升沿被锁存。

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



5.2.1 字库调用方法:

5.2.2.1 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: 即一个字节的低位表示左面的点, 高位表示右面的点 (如果用户按 word mode 读取点阵数据, 请注意高低字节的顺序), 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。

5.2.2.2 11X12 点、15X16点、24X24点、32X32点汉字及5X7 点、7X8 点、6X12点、12X24 点字符、12 点阵不等宽字符、16点阵不等宽字符的排列格式: 详见字库IC资料 (JLX-GB2312-3205) 的第19-26页。

5.2.2.3 汉字点阵字库地址表如下:

| | 字库内容 | 编码体系 | 码位范围 | 字符数 | 起始地址 | 参考算法 |
|----|------------------------------|--------|-----------|----------|--------|----------|
| 1 | 11X12 点 GB2312 标准点阵字库 | GB2312 | A1A1-F7FE | 6763+846 | 0000 | 6.3.1.1 |
| 2 | 15X16 点 GB2312 标准点阵字库 | GB2312 | A1A1-F7FE | 6763+846 | 2C9D0 | 6.3.1.2 |
| 3 | 24X24 点 GB2312 标准点阵字库 | GB2312 | A1A1-F7FE | 6763+846 | 68190 | 6.3.1.3 |
| 4 | 32X32 点 GB2312 标准点阵字库 | GB2312 | A1A1-F7FE | 6763+846 | EDF00 | 6.3.1.4 |
| 5 | 6X12 点国标扩展字符 | GB2312 | A1A1-ABC0 | 126 | 1DBE0C | 6.3.1.5 |
| 6 | 6X12 点 ASCII 字符 | ASCII | 20~7F 96 | | 1DBE00 | 6.3.2.3 |
| 7 | 12 点阵不等宽 ASCII 方头 (Arial) 字符 | ASCII | 20~7F 96 | | 1DC400 | 6.3.2.7 |
| 8 | 12 点阵不等宽 ASCII 白正字符 | ASCII | 20~7F | 96 | 1DCDC0 | 6.3.2.8 |
| 9 | 8X16 点国标扩展字符 | GB2312 | A1A1-ABC0 | 126 | 1DD790 | 6.3.1.6 |
| 10 | 8X16 点 ASCII 字符 | ASCII | 20~7F 96 | | 1DD780 | 6.3.2.4 |
| 11 | 5X7 点 ASCII 字符 | ASCII | 20~7F 96 | | 1DDF80 | 6.3.2.1 |
| 12 | 7X8 点 ASCII 字符 | ASCII | 20~7F 96 | | 1DE280 | 6.3.2.2 |
| 13 | 16 点阵不等宽 ASCII 方头 (Arial) 字符 | ASCII | 20~7F 96 | | 1DE580 | 6.3.2.9 |
| 14 | 16 点阵不等宽 ASCII 白正字符 | ASCII | 20~7F | 96 | 1DF240 | 6.3.2.10 |
| 15 | 12X24 点国标扩展字符 | GB2312 | A1A1-ABC0 | 126 | 1DFF30 | 6.3.1.8 |
| 16 | 12X24 点 ASCII 字符 | ASCII | 20~7F 96 | | 1DFF00 | 6.3.2.5 |
| 17 | 24 点阵不等宽 ASCII 方头 (Arial) 字符 | ASCII | 20~7F 96 | | 1E22D0 | 6.3.2.11 |
| 18 | 24 点阵不等宽 ASCII 白正字符 | ASCII | 20~7F | 96 | 1E3E90 | 6.3.2.12 |
| 19 | 16X32 点国标扩展字符 | GB2312 | A1A1-ABC0 | 126 | 1E5A90 | 6.3.1.9 |
| 20 | 16X32 点 ASCII 字符 | ASCII | 20~7F 96 | | 1E5A50 | 6.3.2.6 |
| 21 | 32 点阵不等宽 ASCII 方头 (Arial) 字符 | ASCII | 20~7F 96 | | 1E99D0 | 6.3.2.13 |
| 22 | 32 点阵不等宽 ASCII 白正字符 | ASCII | 20~7F | 96 | 1ECA90 | 6.3.2.14 |
| 23 | 保留区 | | | | 1EFB50 | |
| 29 | 输入法码表 | GB2312 | | | 1F36F0 | |
| 32 | 保留区 | | | | 1F7CC8 | |

5.2.2.4 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。

举例说明: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=0x2C9D0;

if(MSB >=0xA1 && MSB <= 0xA9 && 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;

5.3 初始化方法

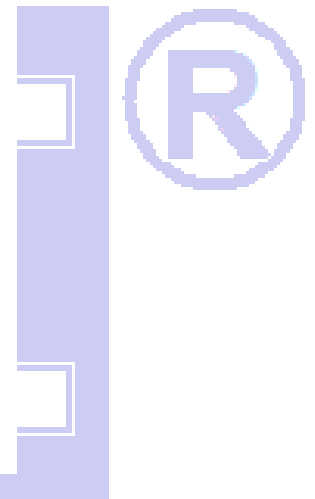
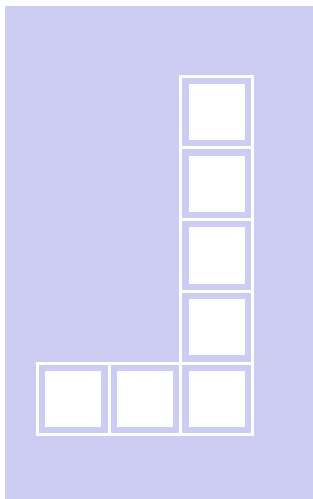
用户所编的显示程序, 开始必须进行初始化, 否则模块无法正常显示, 过程请参考程序

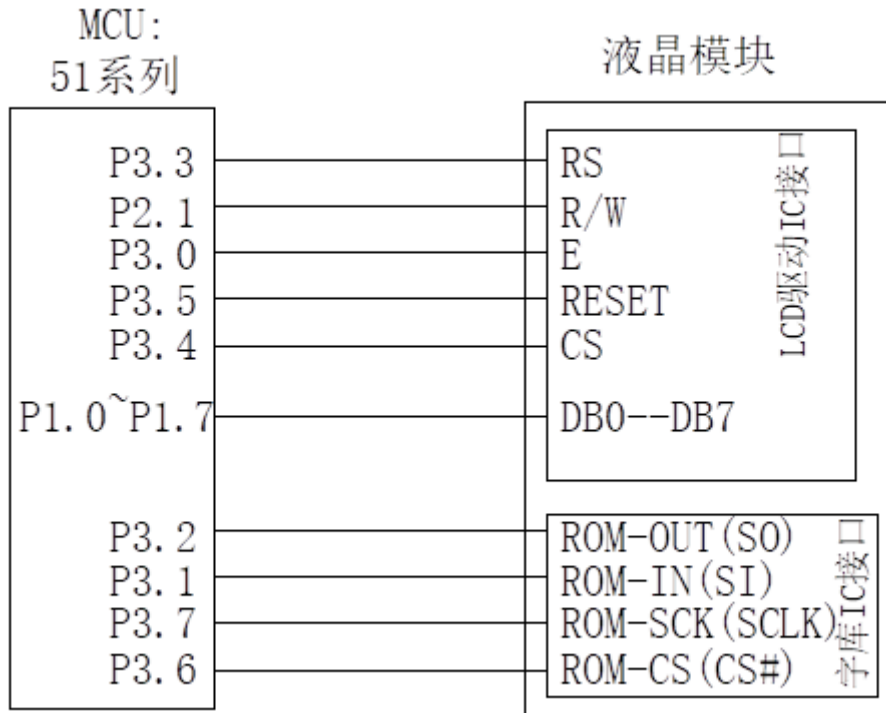
点亮液晶模块的步骤

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、IO端口 (接口)
IO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
背光给合适的直流电可以点亮, 但液晶屏里面没有程序, 只给电不能让液晶屏显示 (我们通常说“点亮”), 程序须另外编写, 并烧录 (下载) 到单片机里液晶模块才能工作。





5.4 程序举例:

```
//型号:JLX400-042-PC-P;
```

```
//驱动 IC:ST7796S;
```

```
#include <STC15F2K60S2.H>
```

```
#include <chinese_code.h>
```

```
//液晶屏 IC 所需要的信号线的接口定义
```

```
sbit RS=P3^3;
```

```
sbit R/W=P2^1;
```

```
sbit E=P3^0;
```

```
sbit CS=P3^4;
```

```
sbit RESET=P3^5;
```

```
sbit key=P2^0; //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#
```

```
void transfer_command(int com1)
```

```
{
```

```
    CS = 0;
```

```
    RS = 0;
```

```
    E = 1;
```

```
    P1=com1;
```

```
    R/W = 0;
```

```
    delay_us(2);
```

```
R/W = 1;
CS = 1;
}
```

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

```
{
    CS = 0;
    RS = 1;
    E = 1;
    P1=data1;
    R/W = 0;
    R/W = 1;
    CS = 1;
}
```

//连写 2 个字节（即 16 位）数据到 LCD 模块

```
void transfer_data_16(uint data_16bit)
```

```
{
    transfer_data(data_16bit>>8);
    transfer_data(data_16bit);
}
```

```
void delay(long i)
```

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

```
void delay_us(long i)
```

```
{
    int j,k;
    for(j=0;j<i;j++);
    for(k=0;k<1;k++);
}
```

```
void Switch()
```

```
{
    repeat:
        if (key==1) goto repeat;
        else delay(1000);
}
```

```
void lcd_initial()
```

```
{
```

```

RESET=1;
delay(200); //延时 200ms
RESET=0;
delay(200); //延时 200ms
RESET=1;
delay(200); //延时 200ms
    
```

```

//***** Start Initial Sequence *****//
    
```

```

transfer_command(0x11); //Sleep Out
delay(200); //延时 200ms
transfer_command(0xf0); //
transfer_data(0xc3); //enable command 2 part 1
transfer_command(0xf0);
transfer_data(0x96); //enable command 2 part 2
transfer_command(0x36); //内存数据访问控制
transfer_data(0xe8); //0x28 旋转 180 度横屏显示
    
```

```

transfer_command(0x3a); //16bit pixel
    
```

```

transfer_data(0x55);
    
```

```

transfer_command(0xb4);
    
```

```

transfer_data(0x01);
    
```

```

transfer_command(0xb7);
    
```

```

transfer_data(0xc6);
    
```

```

transfer_command(0xe8);
    
```

```

transfer_data(0x40);
    
```

```

transfer_data(0x8a);
    
```

```

transfer_data(0x00);
    
```

```

transfer_data(0x00);
    
```

```

transfer_data(0x29);
    
```

```

transfer_data(0x19);
    
```

```

transfer_data(0xa5);
    
```

```

transfer_data(0x33);
    
```

```

transfer_command(0xc1);
    
```

```

transfer_data(0x06);
    
```

```

transfer_command(0xc2);
    
```

```

transfer_data(0xa7);
    
```

```

transfer_command(0xc5);
    
```

```

transfer_data(0x18);
    
```

```

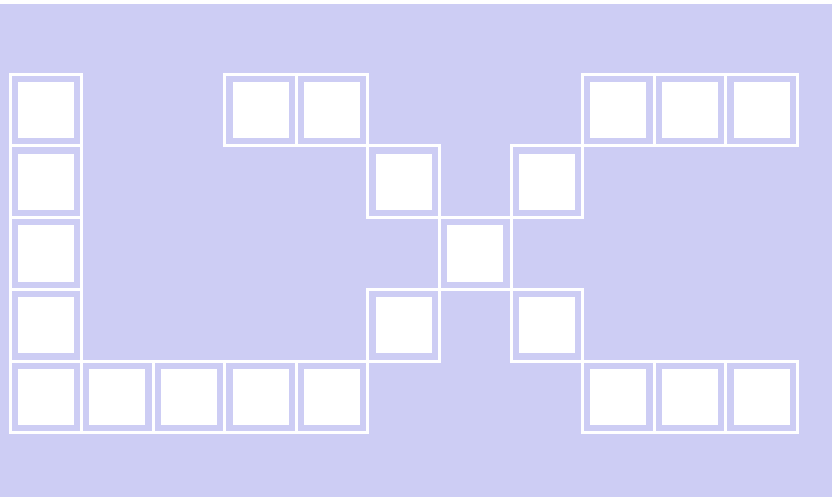
transfer_command(0xe0); //Positive Voltage Gamma Control
    
```



```
transfer_data(0xf0);
transfer_data(0x09);
transfer_data(0x0b);
transfer_data(0x06);
transfer_data(0x04);
transfer_data(0x15);
transfer_data(0x2f);
transfer_data(0x54);
transfer_data(0x42);
transfer_data(0x3c);
transfer_data(0x17);
transfer_data(0x14);
transfer_data(0x18);
transfer_data(0x1b);
```

```
transfer_command(0xe1); //Negative Voltage Gamma Coltrol
```

```
transfer_data(0xf0);
transfer_data(0x09);
transfer_data(0x0b);
transfer_data(0x06);
transfer_data(0x04);
transfer_data(0x03);
transfer_data(0x2d);
transfer_data(0x43);
transfer_data(0x42);
transfer_data(0x3b);
transfer_data(0x16);
transfer_data(0x14);
transfer_data(0x17);
transfer_data(0x1b);
```



```
transfer_command(0xf0);
transfer_data(0x3c);
```

```
transfer_command(0xf0);
transfer_data(0x69);
delay(120); //延时 120ms
transfer_command(0x21); //IPS 必须加这条指令
transfer_command(0x29); //Display ON
transfer_command(0x2c); //写数据开始
```

```
}
```

//定义窗口坐标：开始坐标 (XS,YS)以及窗口大小 (x_total,y_total)

```
void lcd_address(int XS,int YS,int x_total,int y_total)
```

```
{
```

```
int XE,YE;
```

```

XE=XS+x_total-1;
YE=YS+y_total-1;
transfer_command(0x2a);      // 设置 X 开始及结束的地址
transfer_data_16(XS); // X 开始地址(16 位)
transfer_data_16(XE); // X 结束地址(16 位)

transfer_command(0x2b);      // 设置 Y 开始及结束的地址
transfer_data_16(YS); // Y 开始地址(16 位)
transfer_data_16(YE); // Y 结束地址(16 位)

transfer_command(0x2c);      // 写数据开始
    }
    
```

```
void mono_transfer_data_16(int mono_data,int font_color,int back_color)
```

```

{
    int i;
    for(i=0;i<8;i++)
    {
        if(mono_data&0x80)
        {
            transfer_data_16(font_color); //当数据是 1 时，显示字体颜色
        }
        else
        {
            transfer_data_16(back_color); //当数据是 0 时，显示底色
        }
        mono_data<<=1;
    }
}
    
```



```
void display_black(void)
```

```

{
    int i;
    transfer_command(0x2c);      // 写数据开始
    for(i=0;i<480;i++)
    {
        lcd_address(i,0,479,0); // 显示上面一条横线
        transfer_data_16(0xffff);
    }

    for(i=0;i<480;i++)
    {
        lcd_address(i,319,479,319); // 显示下面一条横线
        transfer_data_16(0xffff);
    }
}
    
```

```

for(i=0;i<319;i++)
{
    lcd_address(0,i,0,319);    // 显示左边一条竖线
    transfer_data_16(0xffff);
}

for(i=0;i<319;i++)
{
    lcd_address(479,i,479,319); // 显示右边一条竖线
    transfer_data_16(0xffff);
}

}
    
```

```

////显示一幅彩图
//void display_image(int x,int y,uchar *dp)
//{
    
```

```

//  uchar i,j,k=0;
//  lcd_address(x,y,160,160);
//  for(i=0;i<160;i++)
//  {
//      for(j=0;j<160;j++)
//      {
//          transfer_data(*dp);    //传一个像素的图片数据的高位
//          dp++;
//          transfer_data(*dp);    //传一个像素的图片数据的低位
//          dp++;
//      }
//  }
//}
    
```



```

//全屏显示 RGB 颜色
void display_RGB(void)
{
    
```

```

    int i,j;
    lcd_address(0,0,480,320);
    for(i=0;i<160;i++)
    {
        for(j=0;j<320;j++)
        {
            transfer_data_16(red);
        }
    }

    for(i=0;i<160;i++)
    {
    
```



```

for(j=0;j<320;j++)
{
    transfer_data_16(green);
}
}

for(i=0;i<160;i++)
{
    for(j=0;j<320;j++)
    {
        transfer_data_16(blue);
    }
}
}

```

函数名: Gray16

功能: 显示 16 灰阶

```

void display_Gray16(void) //color: WHITE, RED, GREEN, BLUE
{
    uint dataaa;
    uint i, j, k;

    for(i = 0; i < 320; i++)
    {
        for(j = 0; j < 16; j++)
        {
            dataaa = ((2 * j) << 11) | ((4 * j) << 5) | (2 * j);
            for(k = 0; k < 480 / 16; k++)
            {
                transfer_data_16(dataaa);
            }
        }
    }
}

```



//全屏显示一种颜色

```

void display_color(uint color_data)
{
    int i,j;
    lcd_address(0,0,480,320);
    for(i=0;i<480;i++)
    {
        for(j=0;j<320;j++)
        {

```

```

        transfer_data_16(color_data);
    }
}

```

/****送指令到晶联讯字库 IC****/

```
void send_command_to_ROM( uint datu )
```

```

{
    uint 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()
```

```

{
    uint i;
    uint 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;
        delay_us(1);
    }
    return(ret_data);
}

```

//从指定地址读出 32x32 点阵字符的数据写到液晶屏指定 (y,x)座标中

```
void get_and_write_32x32(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
```

```
{
```

```

uint i,j,disp_data;
Rom_CS = 0;
send_command_to_ROM(0x03);
send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位,共 24 位
send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位,共 24 位
send_command_to_ROM(fontaddr&0xff); //地址的低 8 位,共 24 位

for(j=0;j<32;j++)
{
    lcd_address(y,x+j,32,32);
    for(i=0;i<4;i++)
    {
        disp_data=get_data_from_ROM();
        mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行 8 个像素点的数据。
    }
}
Rom_CS=1;
}

```

//从指定地址读出 16x32 点阵字符的数据写到液晶屏指定 (y,x)座标中

void get_and_write_16x32(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)

```

{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff); //地址的低 8 位,共 24 位

    for(j=0;j<32;j++)
    {
        lcd_address(y,x+j,32,16);
        for(i=0; i<2; i++)
        {
            disp_data=get_data_from_ROM();
            mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行 8 个像素点的数据。
        }
    }
    Rom_CS=1;
}

```

//从指定地址读出 24x24 点阵字符的数据写到液晶屏指定 (y,x)座标中

void get_and_write_24x24(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)

```

{
    uint i,j,disp_data;

```

```

Rom_CS = 0;
send_command_to_ROM(0x03);
send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位,共 24 位
send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位,共 24 位
send_command_to_ROM(fontaddr&0xff); //地址的低 8 位,共 24 位

for(j=0;j<24;j++)
{
    lcd_address(y,x+j,24,24);
    for(i=0; i<3; i++ )
    {
        disp_data=get_data_from_ROM();
        mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行 8 个像素点的数据。
    }
}
Rom_CS=1;
}

```

//从指定地址读出 12x24 点阵字符的数据写到液晶屏指定 (y,x)座标中

```

void get_and_write_12x24(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff); //地址的低 8 位,共 24 位

    for(j=0;j<24;j++)
    {
        lcd_address(y,x+j,24,12);
        for(i=0; i<2; i++ )
        {
            disp_data=get_data_from_ROM();
            mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行 8 个像素点的数据。
        }
    }
    Rom_CS=1;
}

```

//从指定地址读出 16x16 点阵字符的数据写到液晶屏指定 (y,x)座标中

```

void get_and_write_16x16(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;

```



```

Rom_CS = 0;
send_command_to_ROM(0x03);
send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位,共 24 位
send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位,共 24 位
send_command_to_ROM(fontaddr&0xff); //地址的低 8 位,共 24 位

for(j=0;j<16;j++)
{
    lcd_address(y,x+j,16,16);
    for(i=0; i<2; i++ )
    {
        disp_data=get_data_from_ROM();
        mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行 8 个像素点的数据。
    }
}
Rom_CS=1;

```

```

//从指定地址读出 8x16 点阵字符的数据写到液晶屏指定 (y,x)座标中
void get_and_write_8x16(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位,共 24 位
    send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位,共 24 位
    send_command_to_ROM(fontaddr&0xff); //地址的低 8 位,共 24 位

    for(j=0;j<16;j++)
    {
        lcd_address(y,x+j,16,8);
        for(i=0; i<1; i++ )
        {
            disp_data=get_data_from_ROM();
            mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行 8 个像素点的数据。
        }
    }
    Rom_CS=1;
}

```

```

//从指定地址读出 12x12 点阵字符的数据写到液晶屏指定 (y,x)座标中
void get_and_write_12x12(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;

```

```

send_command_to_ROM(0x03);
send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高8位,共24位
send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中8位,共24位
send_command_to_ROM(fontaddr&0xff); //地址的低8位,共24位

for(j=0;j<12;j++)
{
    lcd_address(y,x+j,12,12);
    for(i=0; i<2; i++)
    {
        disp_data=get_data_from_ROM();
        mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行8个像素点的数据。
    }
}
Rom_CS=1;
}

```

//从指定地址读出 6x12 点阵字符的数据写到液晶屏指定 (y,x)座标中

```

void get_and_write_6x12(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint i,j,disp_data;
    Rom_CS = 0;
    send_command_to_ROM(0x03);
    send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高8位,共24位
    send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中8位,共24位
    send_command_to_ROM(fontaddr&0xff); //地址的低8位,共24位

    for(j=0;j<12;j++)
    {
        lcd_address(y,x+j,12,6);
        for(i=0; i<1; i++)
        {
            disp_data=get_data_from_ROM();
            mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行8个像素点的数据。
        }
    }
    Rom_CS=1;
}

```

//从指定地址读出数据写到液晶屏指定 (y,x)座标中

```

void get_and_write_5x8(ulong fontaddr,uint x,uint y,uint font_color,uint back_color)
{
    uint j,disp_data;
    Rom_CS = 0;

```

```

//Rom_SCK=0;
send_command_to_ROM(0x03);
send_command_to_ROM((fontaddr&0xff0000)>>16); //地址的高 8 位,共 24 位
send_command_to_ROM((fontaddr&0xff00)>>8); //地址的中 8 位,共 24 位
send_command_to_ROM(fontaddr&0xff); //地址的低 8 位,共 24 位

for(j=0;j<8;j++)
{
    lcd_address(y,x+j,8,5);
    disp_data=get_data_from_ROM();
    mono_transfer_data_16(disp_data,font_color,back_color); //这一句相当于写了一行 8 个像素点的数据。
}
Rom_CS=1;
}

```

/*-----*/

ulong fontaddr;

//显示一串 32x32 点阵的汉字或 16x32 点阵的 ASCII 码

void disp_GB2312_32x32_string(uint y,uint x,uchar *text,uint font_color,uint back_color)

```

{
    uint i=0;
    while((text[i]>0x00))
    {
        if( ((text[i]>=0xb0) && (text[i]<=0xf7) ) && (text[i+1]>=0xa1) )
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0Xedf00);

            get_and_write_32x32(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=32;
        }

        else if(( (text[i]>=0xa1) && (text[i]<=0xa9) ) && (text[i+1]>=0xa1) )
        {

            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0Xedf00);

            get_and_write_32x32(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=32;
        }
    }
}

```



```

}

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

    get_and_write_16x32(fontaddr,y,x,font_color,back_color);
    i+=1;
    x+=16;
}

else
    i++;
}
}

```

//显示一串 24x24 点阵的汉字或 12x24 点阵的 ASCII 码

```
void disp_GB2312_24x24_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
```

```

{
    uint i= 0;
    while((text[i]>0x00))
    {
        if( ((text[i]>=0xb0) && (text[i]<=0xf7) ) && (text[i+1]>=0xa1) )
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*72);
            fontaddr = (ulong)(fontaddr+0X068190);

            get_and_write_24x24(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=24;
        }

        else if(( (text[i]>=0xa1) && (text[i]<=0xa9) ) && (text[i+1]>=0xa1) )
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*72);
            fontaddr = (ulong)(fontaddr+0X068190);

            get_and_write_24x24(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=24;
        }
    }
}

```




```

}
else if( (text[i]>=0x20) && (text[i]<=0x7e) )
{
    fontaddr = (text[i]- 0x20);
    fontaddr = (ulong)(fontaddr*48);
    fontaddr = (ulong)(fontaddr+0x1dff00);
    get_and_write_12x24(fontaddr,y,x,font_color,back_color);
    i+=1;
    x+=12;
}
else
    i++;
}
}

```

//=====从字库读数据，显示 16*16 点阵的汉字或 8*16 点阵的数字=====

```
void disp_GB2312_16x16_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
```

```

{
    uint i= 0;
    while((text[i]>0x00))
    {
        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);

            get_and_write_16x16(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=16;
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);

            get_and_write_16x16(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=16;
        }
        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);

```



```

fontaddr = (ulong)(fontaddr*16);
fontaddr = (ulong)(fontaddr+0x1dd780);

get_and_write_8x16(fontaddr,y,x,font_color,back_color);
i+=1;
x+=8;
}
else
    i++;
}
}

```

//=====从字库读数据，显示 12*12 点阵的汉字或 6*12 点阵的数字

```
void disp_GB2312_12x12_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
```

```

{
    uint i= 0;
    while((text[i]>0x00))
    {
        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*24);
            fontaddr = (ulong)(fontaddr+0x00);

            get_and_write_12x12(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=12;
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*24);
            fontaddr = (ulong)(fontaddr+0x00);

            get_and_write_12x12(fontaddr,y,x,font_color,back_color);
            i+=2;
            x+=12;
        }
        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (ulong)(fontaddr*12);
            fontaddr = (ulong)(fontaddr+0x1dbe00);
        }
    }
}

```



```

        get_and_write_6x12(fontaddr,y,x,font_color,back_color);
        i+=1;
        x+=6;
    }
    else
        i++;
}
}

```

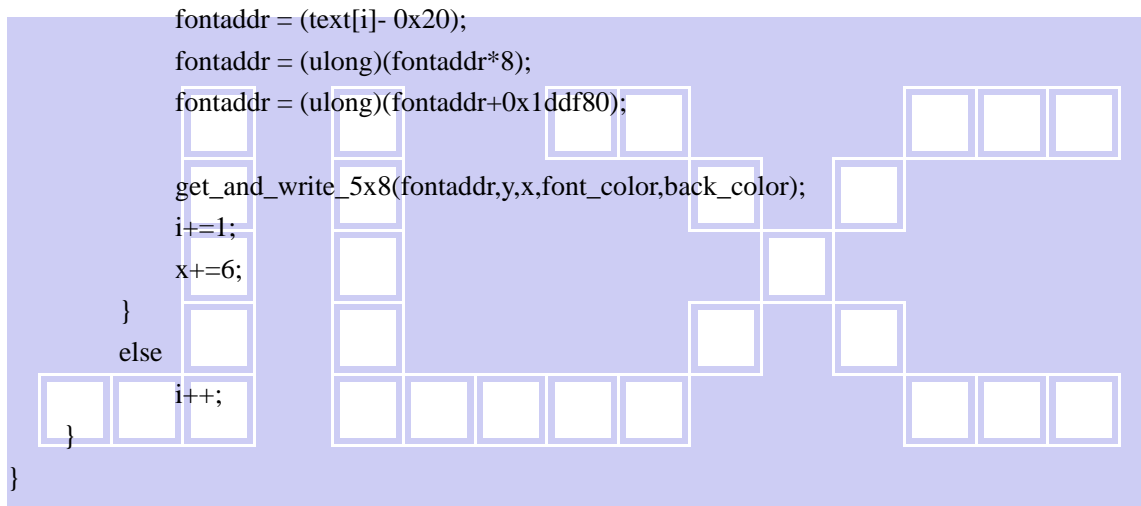
//=====从字库读数据，显示 5*8 点阵的字

```
void disp_GB2312_5x8_string(uint y,uint x,uchar *text,uint font_color,uint back_color)
```

```

{
    uint i= 0;
    while((text[i]>0x00))
    {
        if((text[i]>=0x20) &&(text[i]<=0x7e))
        {

```



```
void main(void)
```

```

{
    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
    lcd_initial();
    while(1)
    {
        display_color(blue); //显示全屏蓝色
        disp_GB2312_32x32_string( 0,95,"晶联讯 3.5 寸 TFT 彩屏",white,blue); //显示 32x32 点阵的字符串，括号
        内的参数分别是 (y, x, 数据指针, 字体色, 背景色)
        disp_GB2312_32x32_string(32+8,0,"①32*32 点阵大汉字库",white,blue); //显示 32x32 点阵的字符串，
        括号内的参数分别是 (y, x, 数据指针, 字体色, 背景色)
    }
}

```



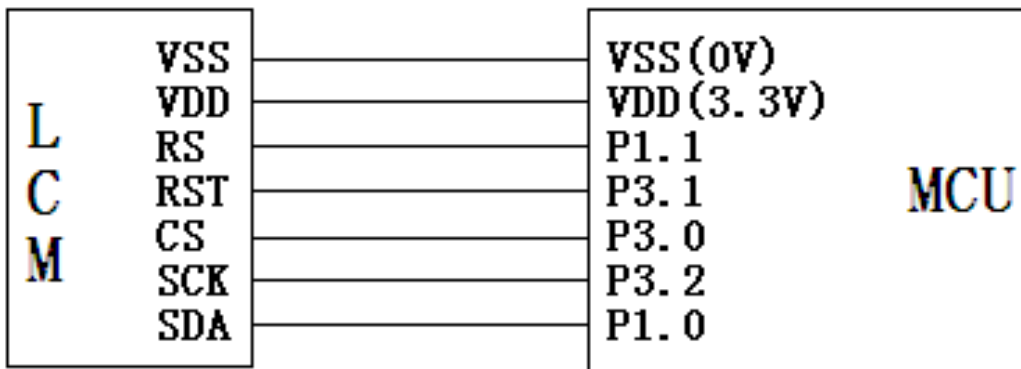
```
disp_GB2312_12x12_string(12*4,0," 自成立以来,晶联讯一直以研发液晶显示解决方案为核心,注重品质并  
专心于技术领域, ",white,blue);  
disp_GB2312_12x12_string(12*5,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*6,0,"相应形成自身的核心能力。现研发的产品有:一、TFT 彩色液晶显示模块。  
二、TN 型、STN 型",white,blue);  
disp_GB2312_12x12_string(12*7,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*8,0,"FSTN 型液晶显示屏。三、COG、COB 型液晶显示模块。其中标准化的  
COG 型液晶模块、带中文",white,blue);  
disp_GB2312_12x12_string(12*9,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*10,0,"字库的液晶模块在行业中处于领先地位。目前,晶联讯各类产品已经被  
广泛应用于新能源,",white,blue);  
disp_GB2312_12x12_string(12*11,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*12,0,"医疗、电力、通讯、金融、环保、工业自动化等多个领域。",white,blue);  
disp_GB2312_12x12_string(12*13,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*14,0,"实力打造成功,品质决定成败。晶联讯依靠长期的实践经验和强大的科  
研创新能力,在激烈",white,blue);  
disp_GB2312_12x12_string(12*15,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*16,0,"的市场竞争中稳步前行,已通过 ISO9001 质量管理体系认证,并严格按  
ISO9001 质量管理体系",white,blue);  
disp_GB2312_12x12_string(12*17,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*18,0,"的要求进行管理,已成为全国多家知名品牌和终端用户的合作伙伴。  
",white,blue);  
disp_GB2312_12x12_string(12*19,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*20,0,"为顾客创造价值,为员工提供发展空间,为社会、厂商、股东创造  
效益,是晶联讯发展的基",white,blue);  
disp_GB2312_12x12_string(12*21,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*22,0,"石和动力。多年来,晶联讯凭借稳定、可靠的产品质量,优异的性  
价比,完善的售后服务, ",white,blue);  
disp_GB2312_12x12_string(12*23,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*24,0,"树立了良好的品牌形象;高于国家标准,超出顾客期望,为广大用  
户提供了更加增值贴身的",white,blue);  
disp_GB2312_12x12_string(12*25,0,"  
",white,blue);  
disp_GB2312_12x12_string(12*26-4,0," 服 务 。
```

```

display_RGB();
Switch();
display_Gray16();
Switch();
display_color(0xf800);
Switch();
display_color(0x07e0);
Switch();
display_color(0x001f);
Switch();
display_color(0x0000);           // 全屏显示黑色
display_black();
Switch();
display_color(0xffff);
Switch();
}
}

```

TFT 模块与 MPU(以 8051 系列单片机为例)串行接口图如下:



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：
串程序序：

```

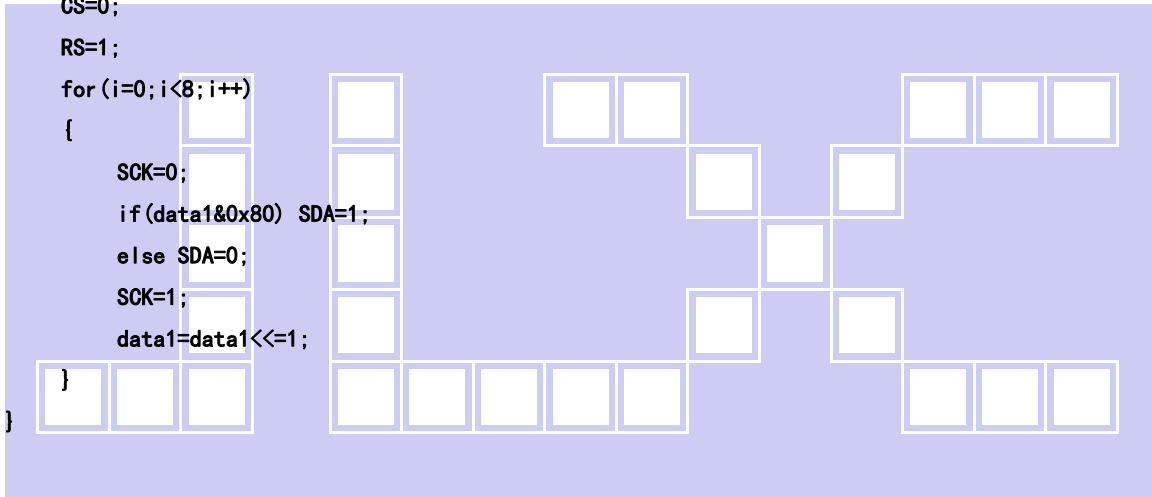
//型号: JLX400-042-PC-S、横屏；
//驱动 IC: ST7796S；
#include <STC15F2K60S2.H>
#include <chinese_code.h>
//液晶屏 IC 所需要的信号线的接口定义
sbit RS=P1^1;
sbit SCK=P3^2;
sbit SDA=P1^0;
sbit CS=P3^0;
sbit RST=P3^1;
sbit key=P2^0;           //P2.0 口与 GND 之间接一个按键

```

```
void transfer_command(int com1)
{
    char i;
    CS=0;
    RS=0;
    for (i=0; i<8; i++)
    {
        SCK=0;
        if (com1&0x80) SDA=1;
        else SDA=0;
        SCK=1;
        com1=com1<<=1;
    }
}
```

```
void transfer_data(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;
        data1=data1<<=1;
    }
}
```



-END-