JLX384160G-9805-PL 使用说明书

(带字库 IC)

目 录

1．概述

晶联讯电子专注于液晶屏及液晶模块的研发、制造。所生产 JLX384160G-9805-PL 型液晶模块由于使用方便、显示清晰，广泛应用于各种人机交流面板。

JLX384160G-9805-PL 可以显示 384 列*160 行点阵单色或 4 灰度级的图片，或显示 12 个/行*5 行 32*32 点阵或显示 16 个/行*6 行 24*24 点阵的汉字，或显示 24 个/行*10 行 16*16 点阵的汉字，或显示 8*16 点阵的英文、数字、字符 48 个*10 行，或显示 5*8 点阵的英文、数字、字符 64 个*20 行。

2．JLX384160G-9805-PL 图像型点阵液晶模块的特性

2.1 结构牢，焊接式 FPC，带铁框。

2.2 IC 采用矽创公司 ST7586S,功能强大，稳定性好

2.3 功耗:不带背光 6.6mW（3.3V*2mA）,带背光不大于 490mW（3.0V*160mA）；

2.4 接口简单方便:可采用 4 线 SPI 串行接口、并行接口。

2.5 工作温度宽：-20℃∽+70℃；

2.6 储存温度宽：-30℃∽+80℃；

2.7 显示内容:

- ●384*160 点阵单色或 4 灰度级图片；
- ●或显示 12 个×5 行 32*32 点阵的汉字；
- ●或显示 16 个×6 行 24*24 点阵的汉字；
- ●或显示 24 个×10 行 16*16 点阵的汉字；
- ●或显示 32 个×13 行 12*12 点阵的汉字；
- ●或显示 48 个×10 行 8*16 点阵的汉字；
- ●或显示 64 个×20 行 5*8 点阵的汉字；
- ●或显示其他的 ASCII 码等；

2.8 液晶模块是易碎的玻璃盒，请小心使用，轻拿轻放

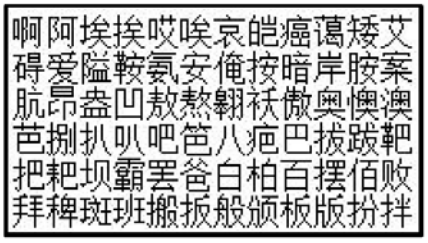
2.9 字库 IC(IC 型号：JLX-GB2312-3205，此 IC 为可选的配件)自带字库内容：

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

15X16 点 GB2312 汉字



24X24 点 GB2312 汉字

32X32 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：

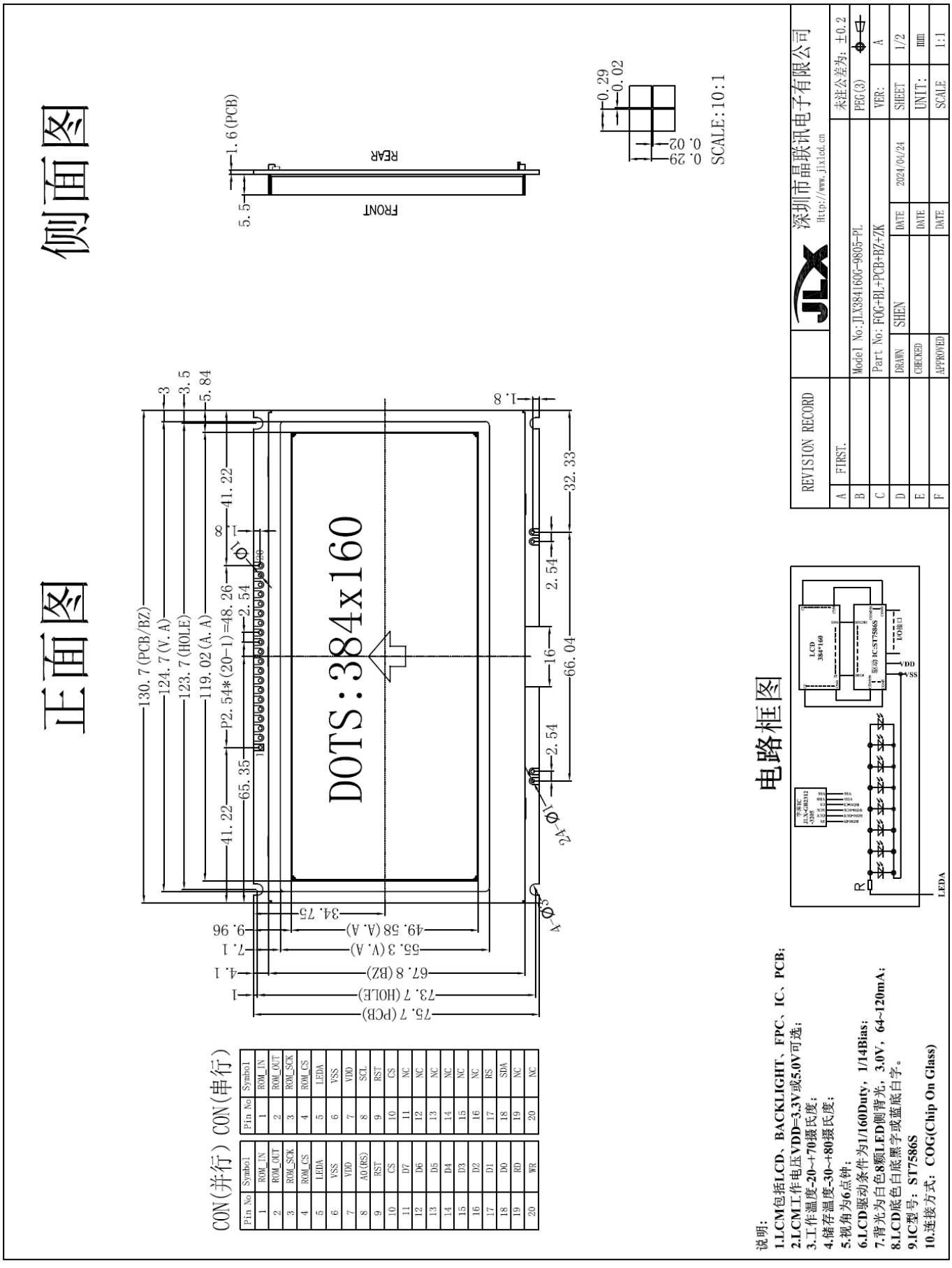


图 1. 液晶模块外形尺寸

模块的接口**既可以当成并口用，也可以当成串口用（PCB 内部跳线选择串口/并口）：**

并口引脚功能说明：

引 线 号	符 号	名 称	功 能
1	ROM_IN	字库 IC 接口 SI	字库 IC:串行数据输入。不带字库时：空脚
2	ROM_OUT	字库 IC 接口 SO	字库 IC:串行数据输出。不带字库时：空脚
3	ROM_SCK	字库 IC 接口 SCLK	字库 IC:串行时钟。不带字库时：空脚
4	ROM_CS	字库 IC 接口 CS#	字库 IC:片选输入。不带字库时：空脚
5	LEDA	背光电源正极	背光电源正极，常亮可短接到VDD
6	VSS	接地	供电电源负极
7	VDD	供电电源正极	供电电源正极（购买时需选择3.3V或5.0V供电）
8	A0（RS）	寄存器选择信号	H:数据寄存器 0:指令寄存器
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	D7	IO	数据总线
12	D6	IO	数据总线
13	D5	IO	数据总线
14	D4	IO	数据总线
15	D3	IO	数据总线
16	D2	IO	数据总线
17	D1	IO	数据总线
18	D0	IO	数据总线
19	E(RD)	使能信号（读）	6800 时序时：E:使能信号 8080 时序时：读信号
20	R/W(WR)	读/写（写）	6800 时序时：RW: H:读信号 L:写信号 8080 时序时：写信号

表 1：并行接口功能

串口引脚功能说明：

引 线 号	符 号	名 称	功 能
1	ROM_IN	字库 IC 接口 SI	字库 IC:串行数据输入。不带字库时：空脚
2	ROM_OUT	字库 IC 接口 SO	字库 IC:串行数据输出。不带字库时：空脚
3	ROM_SCK	字库 IC 接口 SCLK	字库 IC:串行时钟。不带字库时：空脚
4	ROM_CS	字库 IC 接口 CS#	字库 IC:片选输入。不带字库时：空脚
5	LEDA	背光电源正极	背光电源正极，常亮可短接到VDD
6	VSS	接地	供电电源负极
7	VDD	供电电源正极	供电电源正极（购买时需选择3.3V或5.0V供电）
8	A0（RS）	SCL，串行时钟	串行时钟SCL
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	D7	悬空或接VDD	悬空或接VDD
12	D6	悬空或接VDD	悬空或接VDD
13	D5	悬空或接VDD	悬空或接VDD

14	D4	悬空或接VDD	悬空或接VDD
15	D3	悬空或接VDD	悬空或接VDD
16	D2	悬空或接VDD	悬空或接VDD
17	D1	RS，寄存器选择信号	RS：H:数据寄存器 0:指令寄存器
18	D0	SDA，串行数据	串行数据，SDA
19	E(RD)	悬空或接VDD	悬空或接VDD
20	R/W(WR)	悬空或接VDD	悬空或接VDD

表 2：串口引脚功能

4. 电路框图

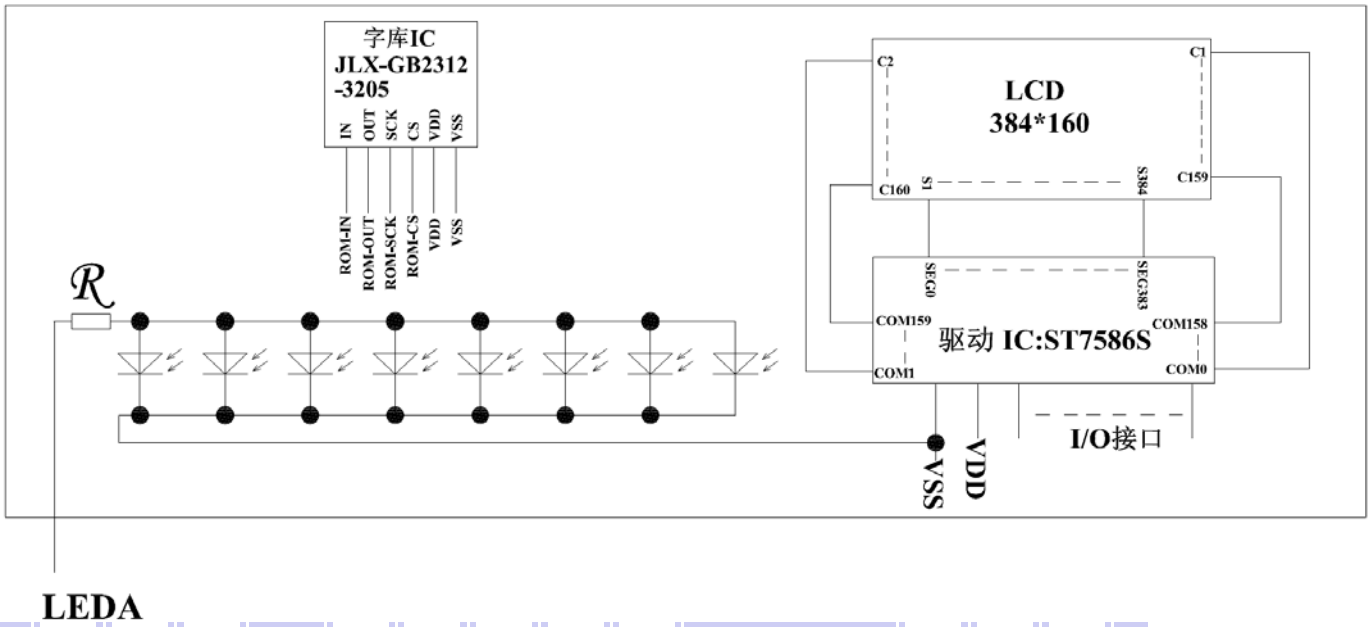


图 2:JLX384160G-9805 图像点阵型液晶模块的电路框图

5. 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℃∽+70℃
背光颜色：白色。
正常工作电流为：(8∽15)×8=64∽120mA（LED 灯数共 8 颗);
工作电压：3.0V;（PCB 已加限流电阻，工作电压同 VDD）

6. 指令表及硬件接口、编程案例

液晶模块中有两个 IC，一个 IC 在 LCD 玻璃上，叫 ST7586S,是液晶屏的驱动 IC,另一个 IC 在 PCB 上，叫 JLX-GB2312-3205,是标准的汉字库存储芯片，是不可改写的存储器(ROM).

不带字库 IC 时，单片机（MCU）也可以通过控制驱动 IC（ST7586S）使液晶屏显示。

带字库 IC 时，单片机（MCU）可以从字库 IC 中读取汉字的点阵数据，再将点阵数据写入驱动 IC（ST7586S）使液晶屏显示。

6.1 LCD 驱动 IC（ST7586S）的指令表：

请参考 JLX384160G-9805-PN 的中文说明书

详细说明请参考 ST7586S 的 IC 资料.

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

6.2.1 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：

6.2.1.1 Read Data Bytes（一般读取）

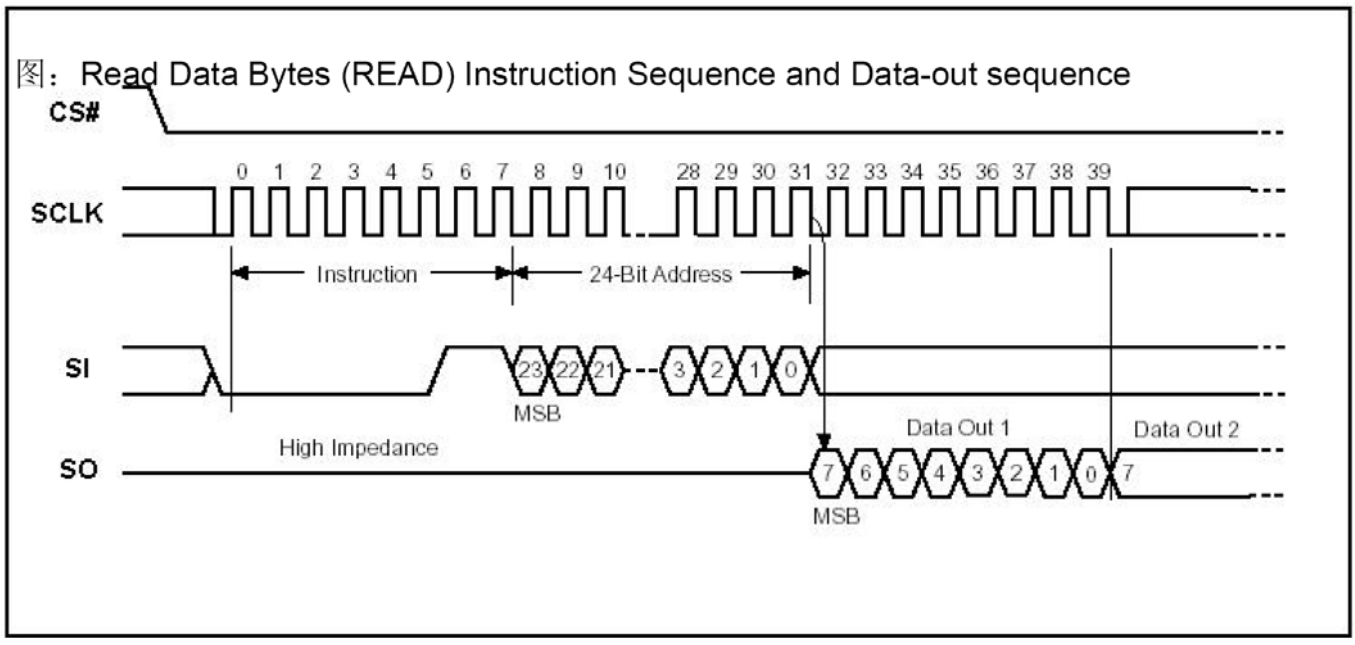
Read Data Bytes 需要用指令码来执行每一次操作。READ 指令的时序如下(图)：

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。



6.2.1.2 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)：

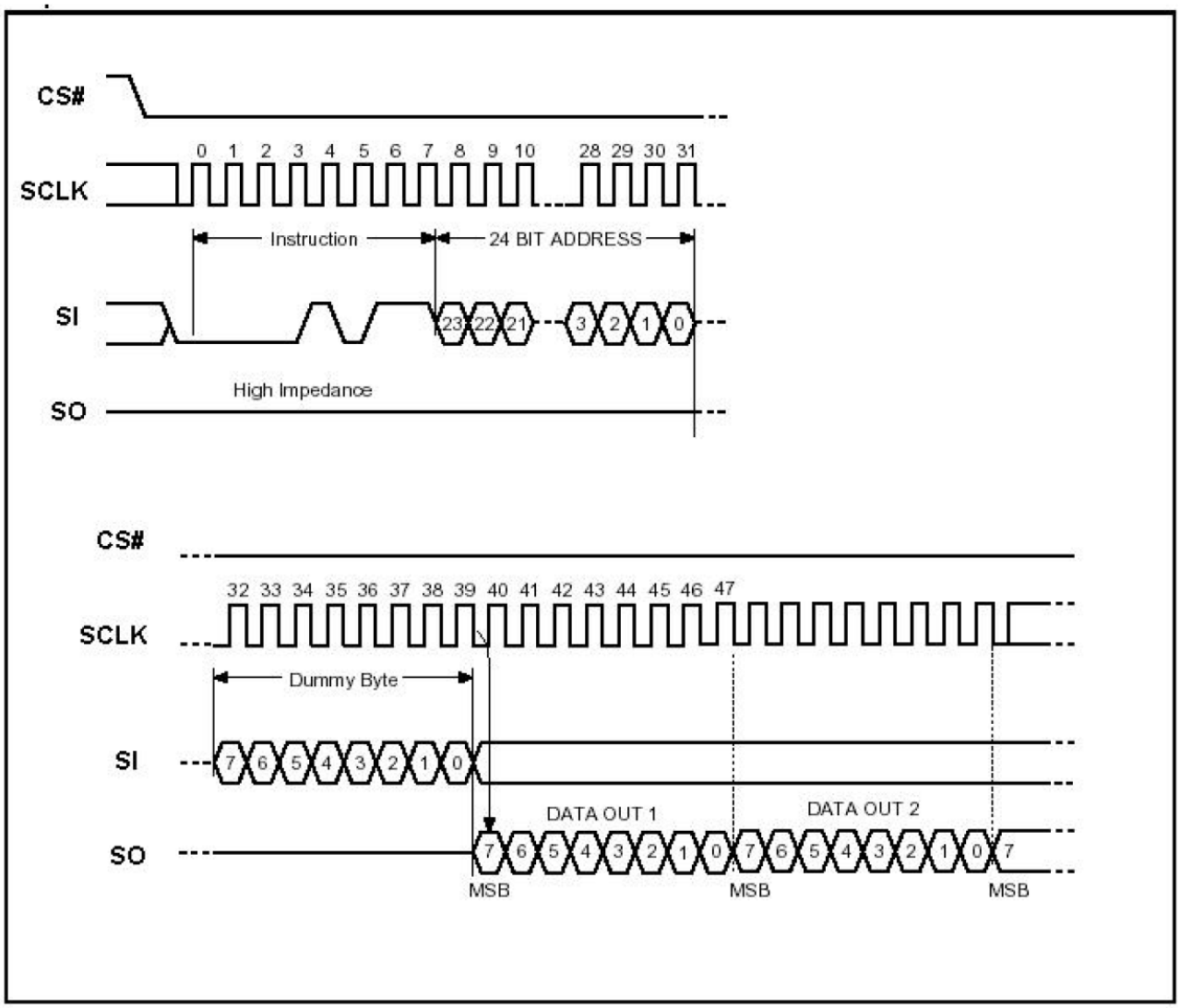
首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（0B h）和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。

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



6.2.2字库调用方法：

6.2.2.1 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：即一个字节的高位表示左面的点，低位表示右面的点（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。

6.2.2.2 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。

6.2.2.3 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 ASCIII 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	

6.2.2.4 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;
```

6.3 接口方式及程序:

6.3.1 液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:

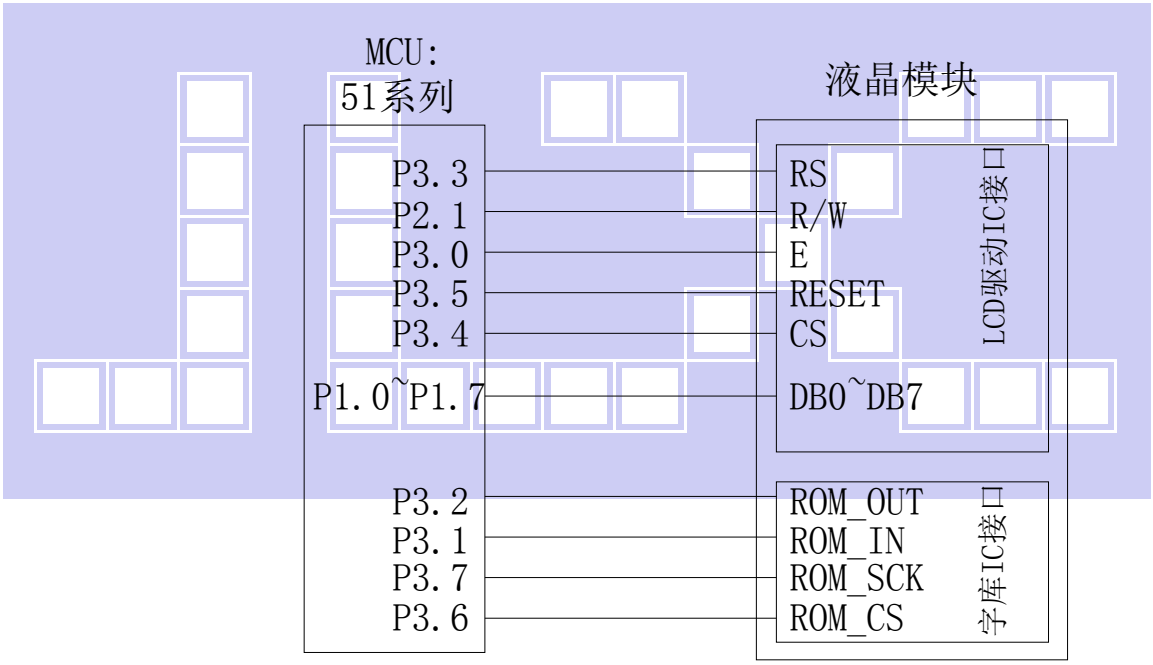


图 3：并行接口图

6.3.2 并行程序:

```
/*   液晶模块型号：JLX384160G-9805-PL,
     并行接口,6800 时序,
     驱动 IC 是:ST7586S(or compatible),
     字库 IC:JLX-GB2312-3205
     版权所有：晶联讯电子：网址  http://www.jlxlcd.cn;
*/
//#include <reg51.H>
#include <STC15F2K60S2.H>
#include <intrins.h>
#include <Ctype.h>
#include <chinese_code.h>
```

```
sbit lcd_cs1 =  P3^4;  //对应 LCD 的 CS 引脚
sbit lcd_reset= P3^5;  //对应 LCD 的 RST 引脚
sbit lcd_rs  =  P3^3;  //对应 LCD 的 RS 引脚
sbit lcd_rw  =  P2^1;  //对应 LCD 的 R\W 引脚
sbit lcd_e   =  P3^0;  //对应 LCD 的 E 引脚

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#

/*另外：DB0~DB7 与 P1.0~P1.7 相连*/
sbit key  =  P2^0;     //按键：我的主板上是 P2.0 口与 GND 之间接一个按键

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

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

//等待一个按键，我的主板是用 P2.0 与 GND 之间接一个按键
void waitkey()
{
 repeat:
     if (key==1) goto repeat;
     else delay_ms (2000);
}

//写指令到 LCD 模块
void transfer_command_lcd(int data1)
{

     lcd_cs1=0;
     lcd_rs=0;
     lcd_e=0;
     lcd_rw=0;
     P1=data1;
     lcd_e=1;
     delay_us(1);
     lcd_cs1=1;
     lcd_e=0;
}

//写数据到 LCD 模块
void transfer_data_lcd(int data1)
{
```

```
      lcd_cs1=0;
      lcd_rs=1;
      lcd_e=0;
      lcd_rw=0;
      P1=data1;
      lcd_e=1;
//    delay_us(1);
      lcd_cs1=1;
      lcd_e=0;
}

/*LCD 模块初始化*/
void initial_lcd()
{
      lcd_reset=1;
      lcd_reset=0;                          //硬件复位
      delay_ms (100);
      lcd_reset=1;                          //硬件复位完成后置高
      delay_ms (200);

      transfer_command_lcd(0x11);     //退出睡眠模式

      transfer_command_lcd(0xC0);     // 设置 VOP
      transfer_data_lcd(0x2c);        // 设置 VOP 的值的低 8 位（总共 9 位）,每调一级是 0.03667V
      transfer_data_lcd(0x01);        // 设置 VOP 的值的第 9 位，也是最高一位
      transfer_command_lcd(0xC3);     // 设置 BIAS
      transfer_data_lcd(0x02);        // 00：BIAS = 1/14  02 = 1/12
      transfer_command_lcd(0xC4);     // 设置升压倍数
      transfer_data_lcd(0x07);         // 07：8 倍压

      transfer_command_lcd(0xD0);     // 允许模拟电路
      transfer_data_lcd(0x1D);         // 允许模拟电路

      transfer_command_lcd(0xB5);     // N-Line = 13
      transfer_data_lcd(0x00);         // 8d


      transfer_command_lcd(0x38);     // 0x38：设置为灰度模式； 0x39:  设置为黑白模式。
      transfer_command_lcd(0x3A);     // 允许 DDRAM 接口：单色模式、4 灰度级、16 灰度级；
      transfer_data_lcd(0x02);         // 0x03:16 灰度级；0x02:4 灰度级或单色模式。

//    transfer_command_lcd(0x39);     // 39:  设置为黑白模式
//    transfer_command_lcd(0x3A);     // 允许 DDRAM 接口
//    transfer_data_lcd(0x02);         // 允许 DDRAM 接口
      transfer_command_lcd(0x36);     // 扫描顺序设置
      transfer_data_lcd(0x00);         // 扫描顺序设置:MX=1,MY=1: 从左到右，从上到下的扫描顺序
      transfer_command_lcd(0xB0);     // Duty 设置
      transfer_data_lcd(0x9f);        // Duty 设置:1/160
      transfer_command_lcd(0x20);     // 反显设置：OFF

      transfer_command_lcd(0xf1);      //温度补偿，温度变化改变帧频
      transfer_data_lcd(0x15);
      transfer_data_lcd(0x15);
      transfer_data_lcd(0x15);
      transfer_data_lcd(0x15);

      transfer_command_lcd(0xb1);     // 扫描起始行设置
      transfer_data_lcd(0x00);         // 扫描起始行设置：从 COM0 开始

      transfer_command_lcd(0x29);     // 打开显示：DISPLAY ON
```

```
}

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

      x_end=x+(x_total-1)/3;
      y_end=y+y_total-1;

      transfer_command_lcd(0x2A);
      transfer_data_lcd((x>>8)&0x00ff);
      transfer_data_lcd(x&0x00ff);
      transfer_data_lcd(x_end>>8&0x00ff);
      transfer_data_lcd(x_end&0x00ff);
      transfer_command_lcd(0x2B);
      transfer_data_lcd((y>>8)&0x00ff);
      transfer_data_lcd(y&0x00ff);
      transfer_data_lcd(y_end>>8&0x00ff);
      transfer_data_lcd(y_end&0x00ff);
}

//传送同一个地址的 3 个点阵的黑白的数据：比如 SEG0、SEG1、SEG2（这 3 个点阵是同一个列地址，无法分开）
//送数据时左起第 1 列的数据是“D7 D6 D5 D4 D3 D2 D1 D0” 中的高 3 位---D7 D6 D5, 第 2 列是中 3 位---D4 D3 D2,第 3 列是低两位---D1 D0。
void transfer_mono_data_3pixel(uchar mono_data)
{
      uchar gray_data=0;

      if(mono_data&0x80)
      {
            gray_data=0xe0;   //二进制 11100000，就是给 D7、D6、D5 赋值
      }
      else
      {
            gray_data=0;
      }
      mono_data<<=1;
      if(mono_data&0x80)
      {
            gray_data+=0x1c;  //二进制 00011100，就是给 D4、D3、D2 赋值
      }
      else;
      mono_data<<=1;
      if(mono_data&0x80)
      {
            gray_data+=0x03; //二进制 00000011，就是给 D1、D0 赋值
      }
      else;
      transfer_data_lcd(gray_data);     //display 3 dots (seg_N,seg_N+1,seg_N+2)

}


//显示 6 个点阵
void transfer_mono_data_6pixel(uchar dat1)
{
      transfer_mono_data_3pixel(dat1);
      transfer_mono_data_3pixel(dat1<<3);
}
```

```
//显示 8 个点阵
void transfer_mono_data_8pixel(uchar dat1)
{
    transfer_mono_data_3pixel(dat1);//传送 dat1 的 D7\D6\D5 这 3 位，对应 3 个点阵(第 1、2、3 个）会显示出来；列地址是自动+1 的
    transfer_mono_data_3pixel(dat1<<3);//传送 dat1 的 D4\D3\D2 这 3 位，对应 3 个点阵(第 4、5、6 个）会显示出来；列地址是自动+1 的
    transfer_mono_data_3pixel(dat1<<6);//传送 dat1 的 D1\D0 这 2 位，对应 3 个点阵(第 7、8、9 个）会显示出来
    //这个液晶驱动 IC 的每个列地址管 3 个点阵，无法分开，所以第 7、8 个点阵会连累到第 9 个点阵，结果是每次显示 9 个点阵，只不过第 9 个点阵会补“0”
    //如果第 9 个点阵本来有显示内容，就会被无情地清掉
}

//显示 9 个点阵
void transfer_mono_data_9pixel(uchar dat1,uchar dat2)
{
    transfer_mono_data_6pixel(dat1);//先显示 6 个点阵
    transfer_mono_data_3pixel((dat1<<6)|(dat2>>2));   //显示 dat1 的 D1、D0 和 dat2 的 D7 位，对应 3 个点阵(第 7、7、9 个）会显示出来；列地址是自动+1 的
}

//显示 12 个点阵
void transfer_mono_data_12pixel(uchar dat1,uchar dat2)
{
    transfer_mono_data_9pixel(dat1,dat2);//先显示 9 个点阵
    transfer_mono_data_3pixel(dat2<<1);          //传送 dat2 的 D6\D5\D4 这 3 位，对应第 10、11、12 个个点阵会显示出来；列地址是自动+1 的
}

//显示 15 个点阵
void transfer_mono_data_15pixel(uchar dat1,uchar dat2)
{
    transfer_mono_data_12pixel(dat1,dat2);       //先显示 12 个点阵
    transfer_mono_data_3pixel(dat2<<4);          //传送 dat2 的 D3\D2\D1 这 3 位，对应第 13、14、15 个点阵会显示出来；列地址是自动+1 的
}

//显示 16 个点阵
 void transfer_mono_data_16pixel(uchar dat1,uchar dat2)
{
    transfer_mono_data_15pixel(dat1,dat2); //先显示 15 个点阵
    transfer_mono_data_3pixel(dat2<<7);          //显示第 16 个点阵,对应 dat2 的 D0 位。
    //这个液晶驱动 IC 的每个列地址管 3 个点阵，无法分开，所以第 16 个点阵会连累到第 17、18 个点阵，结果是每次显示 18 个点阵，只不过第 17、18 个点阵会补“0”
    //如果第 17、18 个点阵本来有显示内容，就会被无情地清掉
}

//显示 18 个点阵
void transfer_mono_data_18pixel(uchar dat1,uchar dat2,uchar dat3)
{
    transfer_mono_data_15pixel(dat1,dat2);             //先显示 15 个点阵
    transfer_mono_data_3pixel((dat2<<7)|(dat3>>1));   //传送 dat2 的 D0 和 dat3 的 D7、D6 这 3 位，对应第 16、17、18 个点阵会显示出来；列地址是自动+1 的
}

//显示 21 个点阵
void transfer_mono_data_21pixel(uchar dat1,uchar dat2,uchar dat3)
{
    transfer_mono_data_18pixel(dat1,dat2,dat3); //先显示 18 个点阵
    transfer_mono_data_3pixel(dat3<<2);          //传送 dat3 的 D5、D4、D3 这 3 位，对应第 19、20、21 个点阵会显示出来；列地址是自动+1 的
```

```
}

//显示 24 个点阵。方法一：
void transfer_mono_data_24pixel(uchar dat1,uchar dat2,uchar dat3)
{
     transfer_mono_data_21pixel(dat1,dat2,dat3); //先显示 21 个点阵
     transfer_mono_data_3pixel(dat3<<5);         //传送 dat3 的 D2、D1、D0 这 3 位，对应第 22、23、24 个点阵会显示出来；列地址是自动+1 的
}


//显示 24 个点阵。方法二：
/*
void transfer_mono_data_24pixel(uchar dat1,uchar dat2,uchar dat3)  //每个字节显示 8 个点阵，显示 8*3=24 个点阵
{
     transfer_mono_data_3pixel(dat1);                  //传送 dat1 的 D7\D6\D5 这 3 位，对应第 1、2、3 个点阵会显示出来，列地址是自动+1 的
     transfer_mono_data_3pixel(dat1<<3);               //传送 dat1 的 D4\D3\D2 这 3 位，对应第 4、5、6 个点阵会显示出来，列地址是自动
+1 的

     transfer_mono_data_3pixel((dat1<<6)|(dat2>>2));   //传送 dat1 的 D1\D0 和 dat2 的 D7 位，对应第 7、8、9 个点阵会显示出来，列地址是自动+1
的

     transfer_mono_data_3pixel(dat2<<1);               //传送 dat2 的 D6\D5\D4 这 3 位，对应第 10、11、12 个个点阵会显示出来；列地址是
自动+1 的

     transfer_mono_data_3pixel(dat2<<4);               //传送 dat2 的 D3\D2\D1 这 3 位，对应第 13、14、15 个点阵会显示出来；列地址是自
动+1 的

     transfer_mono_data_3pixel((dat2<<7)|(dat3>>1));   //传送 dat2 的 D0 和 dat3 的 D7、D6 这 3 位，对应第 16、17、18 个点阵会显示出来；列地址
是自动+1 的

     transfer_mono_data_3pixel(dat3<<2);               //传送 dat3 的 D5、D4、D3 这 3 位，对应第 19、20、21 个点阵会显示出来；列地址是
自动+1 的

     transfer_mono_data_3pixel(dat3<<5);               //传送 dat3 的 D2、D1、D0 这 3 位，对应第 22、23、24 个点阵会显示出来；列地址是
自动+1 的
}
*/

//显示 27 个点阵
void transfer_mono_data_27pixel(uchar dat1,uchar dat2,uchar dat3,uchar dat4)
{
     transfer_mono_data_24pixel(dat1,dat2,dat3); //先显示 24 个点阵
     transfer_mono_data_3pixel(dat4);            //传送 dat4 的 D7、D6、D5 这 3 位，对应第 25、26、27 个点阵会显示出来；列地址是自动+1 的
}
//显示 30 个点阵
void transfer_mono_data_30pixel(uchar dat1,uchar dat2,uchar dat3,uchar dat4)
{
     transfer_mono_data_24pixel(dat1,dat2,dat3); //先显示 24 个点阵
     transfer_mono_data_6pixel(dat4);            //再显示 6 个点阵，24+6=30
}
//显示 32 个点阵
void transfer_mono_data_32pixel(uchar dat1,uchar dat2,uchar dat3,uchar dat4)
{
     transfer_mono_data_24pixel(dat1,dat2,dat3); //先显示 24 个点阵
     transfer_mono_data_8pixel(dat4);            //再显示 8 个点阵，24+8=32
     //这个液晶驱动 IC 的每个列地址管 3 个点阵，无法分开，所以第 31、32 个点阵会连累到第 33 个点阵，结果是每次显示 33 个点阵，只不过第 33 个点
阵会补“0”
     //如果第 33 个点阵本来有显示内容，就会被无情地清掉
}
```

```
//显示 33 个点阵
void transfer_mono_data_33pixel(uchar dat1,uchar dat2,uchar dat3,uchar dat4,uchar dat5)
{
      transfer_mono_data_24pixel(dat1,dat2,dat3);        //先显示 24 个点阵
      transfer_mono_data_9pixel(dat4,dat5);              //再显示 9 个点阵
}

//显示 48 个点阵
void transfer_mono_data_48pixel(uchar dat1,uchar dat2,uchar dat3,uchar dat4,uchar dat5,uchar dat6)
{
      transfer_mono_data_24pixel(dat1,dat2,dat3);        //先显示 24 个点阵
      transfer_mono_data_24pixel(dat4,dat5,dat6);        //再显示 24 个点阵
}


//传送同一个地址的 3 个点阵的 4 灰度级的数据：比如 SEG0、SEG1、SEG2,这 3 个点阵是同一个列地址，无法分开
//送灰度数据(gray_data)时，SEG0 对应高 3 位（D7、D6、D5）,SEG1 对应中 3 位（D4、D3、D2),SEG2 对应低两位（D1、D0）。
void transfer_gray_data_3pixel(uchar dat1)
{
      uchar gray_data;
      gray_data=dat1&0xc0;;    //给 gray_data 的 D7、D6 赋值(=dat1 的 D7、D6)
      if((dat1&0xc0)==0xc0)
      {
            gray_data|=0x20; //给 gray_data 的 D5 赋值，当 dat1 的 D7、D6 都是 1 的时候，gray_data 的 D5=1,当 dat1 的 D7\D6 不都是 1 的时候，gray_data
的 D5=0
      }
      gray_data|=((dat1>>1)&0x18);    //给 gray_data 的 D4、D3 赋值（=dat1 的 D5、D4)
      if((dat1&0x30)==0x30)
      {
            gray_data|=0x04; //给 gray_data 的 D2 赋值，当 dat1 的 D5、D4 都是 1 的时候，gray_data 的 D2=1,当 dat1 的 D7、D6 不都是 1 的时候，gray_data
的 D2=0
      }
      gray_data|=((dat1>>2)&0x03);       //给 gray_data 的 D1、D0 赋值(=dat1 的 D3、D2)
      transfer_data_lcd(gray_data);      //传送 1 个字节灰度数据给液晶驱动 IC,对应的 3 个点阵会显示(seg_N,seg_N+1,seg_N+2)
}


//传送同一个地址的 12 个点阵的 4 灰度的数据：比如 SEG0、SEG1、SEG2.....SEG9、SEG10、SEG11（这 12 个点阵是 4 个列地址）
//每 2 位数据对应一个点阵，12 个点阵用：2*12=24 位，即 3 个字节:dat1、dat2、dat3
void transfer_gray_data_12pixel(uchar dat1,uchar dat2,uchar dat3)
{
      transfer_gray_data_3pixel(dat1);                    //显示 3 个点阵(seg_N,seg_N+1,SEG_N+2)
      transfer_gray_data_3pixel((dat1<<6)|(dat2>>2));     //显示 3 个点阵(seg_N+3,seg_N+4,SEG_N+5)
      transfer_gray_data_3pixel((dat2<<4)|(dat3>>4));     //显示 3 个点阵(seg_N+6,seg_N+7,SEG_N+8)
      transfer_gray_data_3pixel(dat3<<2);                       //显示 3 个点阵(seg_N+9,seg_N+10,SEG_N+11)
}

/*清屏*/
void clear_screen()
{
      int i,j;
      lcd_address(0,0,384,160);
    transfer_command_lcd(0x2c);
    for(i=0;i<160;i++)
      {
          for(j=0;j<24;j++)
            {
               transfer_mono_data_18pixel(0x00,0x00,0x00); //每个字节显示 8 个点阵，显示 8*3=24 个点阵
            }
```

```
    }
}

/*显示 8*16 点阵 ASCII 码字符或等同于 8*16 点阵的图像*/
void disp_8x16(int x,int y,uchar *dp)
{
    int i,j;
    uchar dat1;
    lcd_address(x,y,8,16);
    transfer_command_lcd(0x2c);
    for(i=0;i<16;i++)
    {
        for(j=0;j<1;j++)
        {
            dat1=*dp;dp++;
            transfer_mono_data_8pixel(dat1);
        }
    }
}

//显示 12*12 点阵的图像
void disp_12x12(int x,int y,uchar *dp)
{
    int i,j;
    uchar dat1,dat2;

    lcd_address(x,y,12,12);

    transfer_command_lcd(0x2C);

    for(i=0;i<12;i++)
    {
        for(j=0;j<1;j++)//循环 1 次，每次显示 12 个点阵
        {
            dat1=*dp;dp++;
            dat2=*dp;dp++;
            transfer_mono_data_12pixel(dat1,dat2); //每个字节显示 8 个点阵，显示 8*2=16 个点阵
        }
    }
}

//显示 16*16 点阵的图像
void disp_16x16(int x,int y,uchar *dp)
{
    int i,j;
    uchar dat1,dat2;

    lcd_address(x,y,16,16);

    transfer_command_lcd(0x2C);

    for(i=0;i<16;i++)
    {
        for(j=0;j<1;j++)//循环 1 次，每次显示 18 个点阵
        {
            dat1=*dp;dp++;
            dat2=*dp;dp++;
            transfer_mono_data_16pixel(dat1,dat2); //每个字节显示 8 个点阵，显示 8*2=16 个点阵
        }
    }
```

```
}

//显示 24*24 点阵的图像
void disp_24x24(int x,int y,uchar *dp)
{
      int i,j;
      uchar dat1,dat2,dat3;

      lcd_address(x,y,24,24);

      transfer_command_lcd(0x2C);

      for(i=0;i<24;i++)
      {
          for(j=0;j<1;j++)//循环 1 次，每次显示 24 个点阵
            {
                  dat1=*dp;dp++;
                  dat2=*dp;dp++;
                  dat3=*dp;dp++;
                  transfer_mono_data_24pixel(dat1,dat2,dat3); //每个字节显示 8 个点阵，显示 8*3=24 个点阵
            }
      }
}


//显示 32*32 点阵的图像
void disp_32x32(int x,int y,uchar *dp)
{
      int i,j;
      uchar dat1,dat2,dat3,dat4;

      lcd_address(x,y,32,32);

      transfer_command_lcd(0x2C);

      for(i=0;i<32;i++)
      {
          for(j=0;j<1;j++)//循环 1 次，每次显示 32 个点阵
            {
                  dat1=*dp;dp++;
                  dat2=*dp;dp++;
                  dat3=*dp;dp++;
                  dat4=*dp;dp++;
                  transfer_mono_data_32pixel(dat1,dat2,dat3,dat4); //每个字节显示 8 个点阵，显示 8*4=32 个点阵
            }
      }
}

//显示 384*160 点阵的图像
void disp_384x160(uchar *dp)
{
      int i,j;
      uchar dat1,dat2,dat3;

      lcd_address(0,0,384,160);

      transfer_command_lcd(0x2C);

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

```
        for(j=0;j<16;j++)//循环 16 次，每次显示 24 个点阵，合计 384 个点阵
          {
               dat1=*dp;dp++;
               dat2=*dp;dp++;
               dat3=*dp;dp++;
               transfer_mono_data_24pixel(dat1,dat2,dat3); //每个字节显示 8 个点阵，显示 8*3=24 个点阵
          }
     }
}

//显示 384*160 点阵的 4 灰度级图像
void disp_4gray_384x160(uchar *dp)
{
     uchar i,j;
     uchar dat1,dat2,dat3;
     lcd_address(0,0,384,160);  //
     transfer_command_lcd(0x2C);
     for(i=0;i<160;i++)
     {
         for(j=0;j<32;j++)//循环 26 次，每次显示 12 个点阵，合计 26*12=312 个点阵
          {
               dat1=*dp;dp++;
               dat2=*dp;dp++;
               dat3=*dp;dp++;
               transfer_gray_data_12pixel(dat1,dat2,dat3); //每个字节显示 4 个点阵，共显示 4*3=12 个点阵
          }

     }
}
/*显示 16*32 点阵 ASCII 码字符或等同于 8*16 点阵的图像*/
void disp_16x32(int x,int y,uchar *dp)
{
     int i,j;
     uchar dat1,dat2;

     lcd_address(x,y,16,132);

     transfer_command_lcd(0x2C);

     for(i=0;i<32;i++)
     {
         for(j=0;j<1;j++)//循环 1 次，每次显示 18 个点阵
          {
               dat1=*dp;dp++;
               dat2=*dp;dp++;
               transfer_mono_data_16pixel(dat1,dat2); //每个字节显示 8 个点阵，显示 8*2=16 个点阵
          }
     }
}

/****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;
            delay_us(1);
                Rom_SCK=1;
//          delay_us(1);
     }
}

/****从晶联讯字库 IC 中取汉字或字符数据（1 个字节）***/
static uchar get_data_from_ROM( )
{

     uchar i;
     uchar ret_data=0;
     Rom_SCK=1;
     for(i=0;i<8;i++)
     {
          Rom_OUT=1;
          Rom_SCK=0;
          ret_data=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_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;
}

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

ulong  fontaddr;
void disp_GB2312_32x32_string(uchar x,uchar y,uchar *text)
{
     uchar i= 0,j;
     uchar addrHigh,addrMid,addrLow ;
     uchar fontbuf[128];
     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);

        addrHigh = (fontaddr&0xff0000)>>16;
        addrMid = (fontaddr&0xff00)>>8;
        addrLow = fontaddr&0xff;

        Rom_CS = 0;
        lcd_cs1=1;

        send_command_to_ROM(0x03);
        send_command_to_ROM(addrHigh);
        send_command_to_ROM(addrMid);
        send_command_to_ROM(addrLow);

        for(j = 0; j < 128; j++ )
        {
             fontbuf[j] =get_data_from_ROM();
        }
        Rom_CS = 1;
        disp_32x32(x,y,fontbuf);
        i+=2;
        x+=11;
    }

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

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

        addrHigh = (fontaddr&0xff0000)>>16;
        addrMid = (fontaddr&0xff00)>>8;
        addrLow = fontaddr&0xff;

        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(j = 0; j < 128; j++ )
        {
             fontbuf[j] =get_data_from_ROM();
        }
        Rom_CS = 1;

        disp_32x32(x,y+11,fontbuf);
        i+=2;
        x+=11;
    }

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

```
            {
                fontaddr = (text[i]- 0x20);
                fontaddr = (ulong)(fontaddr*64);
                fontaddr = (ulong)(fontaddr+0x1e5a50);

                addrHigh = (fontaddr&0xff0000)>>16;
                addrMid = (fontaddr&0xff00)>>8;
                addrLow = fontaddr&0xff;

                get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,64);

                disp_16x32(x,y,fontbuf);
                i+=1;
                x+=6;
            }
            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(uchar x,uchar y,uchar *text)
{
    uchar i= 0;
    uchar addrHigh,addrMid,addrLow ;
    uchar fontbuf[64];
    ulong  fontaddr;
    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);

            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;
            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,32 );
            disp_16x16(x,y,fontbuf);
            i+=2;
            x+=5;
        }
        else if(((text[i]>=0xa1) &&(text[i]<=0xab))&&(text[i+1]>=0xa1))
        {
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*32);
            fontaddr = (ulong)(fontaddr+0x2c9d0);

            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;
            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,32 );
            disp_16x16(x,y,fontbuf);
            i+=2;
            x+=5;
        }
        else if((text[i]>=0x20) &&(text[i]<=0x7e))
```

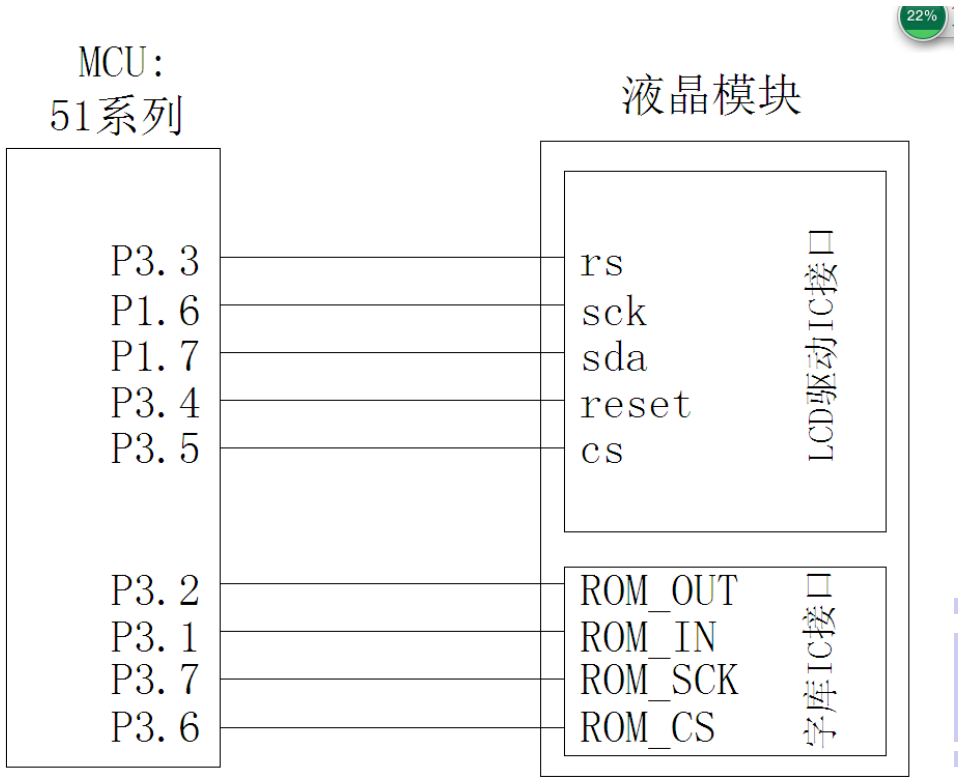
```
        {
            uchar fontbuf[16];
            fontaddr = (text[i]- 0x20);
            fontaddr = (ulong)(fontaddr*16);
            fontaddr = (ulong)(fontaddr+0x1dd780);
            addrHigh = (fontaddr&0xff0000)>>16;
            addrMid = (fontaddr&0xff00)>>8;
            addrLow = fontaddr&0xff;

            get_n_bytes_data_from_ROM(addrHigh,addrMid,addrLow,fontbuf,16 );

            disp_8x16(x,y,fontbuf);
            i+=1;
            x+=3;
        }
        else
            i++;
    }
}

//----------------------------
void main ()
{
  P1M1=0x00;
    P1M0=0x00;
    P2M1=0x00;
    P2M0=0x00;
    P3M1=0x00;
    P3M0=0x00;
    while(1)
    {
        initial_lcd();
        clear_screen();//清屏
        disp_GB2312_32x32_string(0,32*0,"   深圳市晶联讯电子");
        disp_GB2312_32x32_string(0,32*1,"  JLX384160G-9805");
        disp_GB2312_16x16_string(0,64*1,"自带 GB2312 国标汉字库,16X16、32X32 汉字库 JKLMNOP");
        disp_GB2312_16x16_string(0,80*1,"的 ASCII 码⑴①〇●◎◇◆ 1.2.‖…ABCDEFGHIQRXYZWG");
        disp_GB2312_16x16_string(0,96*1,"鑫森淼焱晶磊众品鬟靐麽麾縻麂麋麈麋麒麈麝磊琴斑昔焱,");
        disp_GB2312_16x16_string(0,112*1,"abcdefghijklmnABCDEFGHIJKLMNOPQRSTU 款堪搭塔,");
        disp_GB2312_16x16_string(0,128*1,"麟黛黜黝黠黟黢黩黧黯黯黔黜黯黯麝麟黩淼趁趋超越,");
        disp_GB2312_16x16_string(0,144*1,"△▽〇●◆□☆♀♂※▼▲√×★■@＃【】!?&±/℅§鑫淼");
        waitkey();
        clear_screen();//清屏
        disp_GB2312_32x32_string(0,32*0,"  新年作-----刘长卿");
        disp_GB2312_32x32_string(0,32*1,"乡心新岁切，天畔独清然.");
        disp_GB2312_32x32_string(0,32*2,"老至居人下，春归在客先.");
        disp_GB2312_32x32_string(0,32*3,"岭猿同旦暮，江柳共风烟.");
        disp_GB2312_32x32_string(0,32*4,"已似长沙傅，从今又几年.");
        waitkey();
        clear_screen();//清屏
        disp_384x160(bmp1);         //显示 G-9805
        waitkey();
        clear_screen();//清屏
        disp_384x160(bmp2);         //显示古风画
        waitkey();
        clear_screen();//清屏
        disp_384x160(bmp3);         //显示老虎+二维码
        waitkey();
        clear_screen();//清屏
        disp_4gray_384x160(bmp_4gray_2);          //显示一个 384x160 点阵的 4 灰度级的图片
```

```
        waitkey();
    }
}
```



串行接口图

以下为串行接口方式范例程序

与并行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

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

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

//=============transfer command to LCM===============
void transfer_command_lcd(int data1)
{
    char i;
    CS=0;
```

```
    RS=0;
    for(i=0;i<8;i++)
    {
        SCK=0;
        if(data1&0x80) SDA=1;
        else SDA=0;
        SCK=1;
//    delay_us(1);
        data1=data1<<=1;
    }
}

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

-END-