

# JLX19232G-9808-PN 使用说明书

## 目 录

序号	内 容 标 题	页 码
1	概述	2
2	特点	2
3	外形及接口引脚功能	3~4
4	基本原理	4~5
5	技术参数	5~6
6	时序特性	6~10
7	指令功能及硬件接口与编程案例	10~末 页

## 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 JLX19232G-9808 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。

JLX19232G-9808 可以显示 192 列\*32 行点阵单色图片，或显示 16\*16 点阵的汉字 12 字\*2 行或者 2 字 12 行，或显示 8\*16 点阵的英文、数字、符号 24 个\*2 行或者 4 个 12 行。或显示 5\*8 点阵的英文、数字、符号 32 个\*4 行或者 4 个 24 行。

## 2. JLX19232G-9808 图像型点阵液晶模块的特性

2.1 结构牢：背光带有挡墙，焊接式 FPC。

2.2 IC 采用 ST7525，功能强大，稳定性好

2.3 功耗低：不带背光 1mW (3.3V\*0.3mA)，带背光不大于 661mW (3.3V\*120mA)；

2.4 显示内容：

(1) 192\*32 点阵单色图片或者 32\*192 点阵单色图片，或其它小于 192\*32 点阵或 32\*192 的单色图片；

(2) 可选用 16\*16 点阵或其他点阵的图片来自编汉字，按照 16\*16 点阵汉字来计算可显示 12 字\*2 行或 2 字 12 行；

(3) 按照 8\*16 点阵汉字来计算可显示 24 个\*2 行或 4 个 12 行；

(4) 按照 5\*8 点阵汉字来计算可显示 32 个\*4 行或 4 个 24 行；

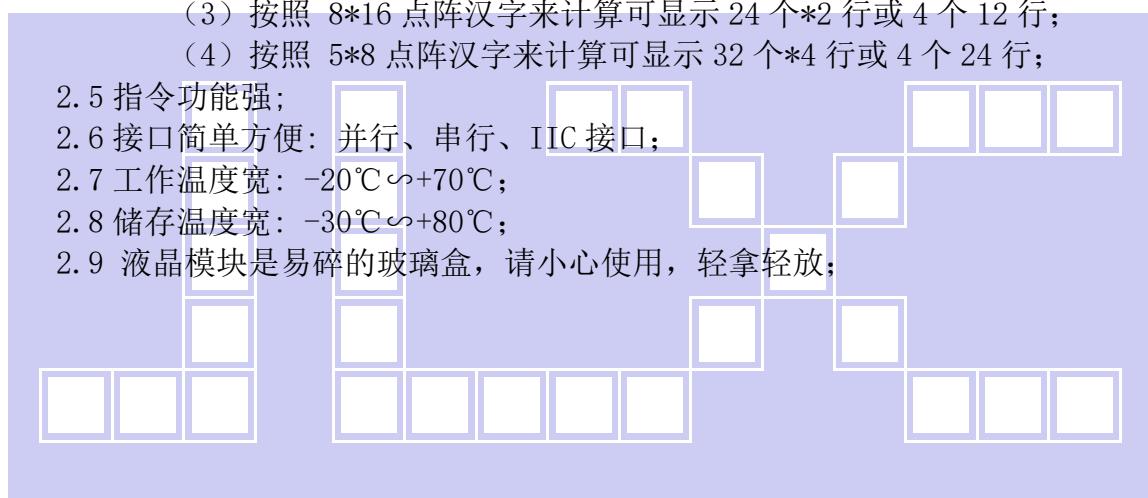
2.5 指令功能强；

2.6 接口简单方便：并行、串行、IIC 接口；

2.7 工作温度宽：-20°C ~ +70°C；

2.8 储存温度宽：-30°C ~ +80°C；

2.9 液晶模块是易碎的玻璃盒，请小心使用，轻拿轻放；



### 3. 外形尺寸及接口引脚功能

侧面图

正面图

**接口定义:**

Pin No	Symbol
1	NC
2	NC
3	NC
4	NC
5	LEDA
6	VSS
7	VDD
8	A0(RS)
9	RST
10	CS
11	D7
12	D6
13	D5
14	D4
15	D3
16	D2
17	D1
18	D0
19	E
20	WR

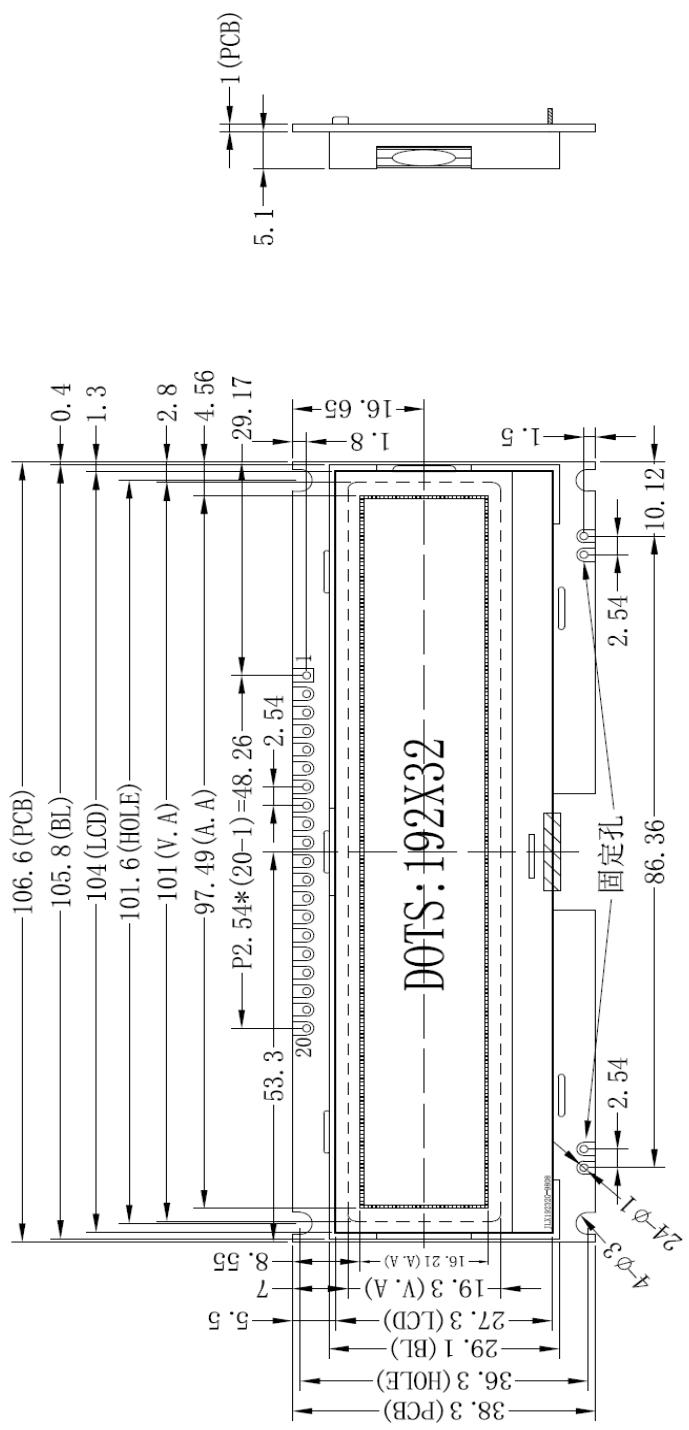


图 1. 外形尺寸

- 说明:
1. LCM包括LCD、BACKLIGHT、FPC、IC、PCB；
  2. LCM工作电压: 3.3V或5.0V可选；
  3. 工作温度-20~+70摄氏度；
  4. 储存温度-30~+80摄氏度；
  5. 视角为6点钟；
  6. LCD驱动条件为1/33Duty, 1/6Bias；
  7. 背光为白色6颗LED背光, 3.0V, 48~120mA；
  8. LCD底色可选白底黑字或蓝底白字或黑底白字；
  9. IC型号: ST7525
  10. 连接方式: COG(Chip On Glass)

### 模块的接口引脚功能

引线号	符号	名称	功 能
1	NC	NC	空脚
2	NC	NC	空脚
3	NC	NC	空脚
4	NC	NC	空脚
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短接一起)
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短接一起)
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短接一起)
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短接一起)
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短接一起)
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: 模块的接口引脚功能

## 4. 基本原理

### 4.1 液晶屏 (LCD)

在LCD上排列着192×32点阵, 192个列信号与驱动IC相连, 32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)。

#### 4.2 工作电路框图:

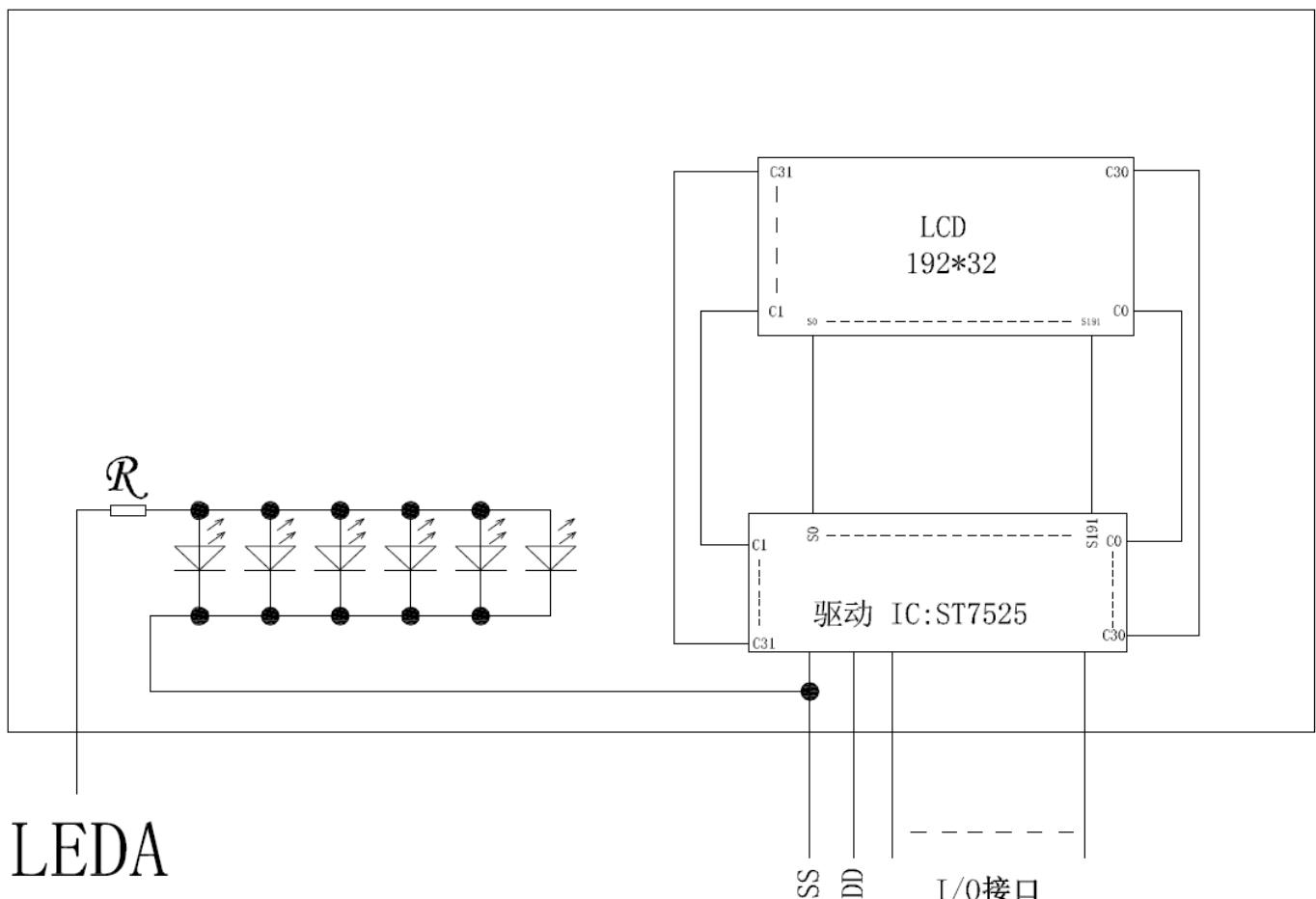


图 2: JLX19232G-9808 图像点阵型液晶模块的电路框图

#### 4.3 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°C ~ +70°C;

存储温度: -30°C ~ +80°C;

背光白色;

正常工作电流为: 48~90mA (LED 灯数共 6 颗);

工作电压: 3.0V; (PCB 已加限流电阻, 供电同 VDD 电压 (LEDA 可与 VDD 短接))

### 5. 技术参数

#### 5.1 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	
电源	VDD – VSS	-0.3	3.3	3.6	V
工作温度		-20		+70	°C
储存温度		-30		+80	°C

表 2: 最大极限参数

#### 5.2 直流 (DC) 参数

名称	符 号	测 试 条 件	标 准 值			单 位
			MIN	TYPE	MAX	
工作电压	VDD		2.6	3.3	3.6	V

背光工作电压	VLED		2.9	3.0	3.1	V
输入高电平	VIH	-	0.8VDD	-	VDD	V
输入低电平	VIO	-	0	-	0.2VDD	V
输出高电平	VOH	$I_{OH} = 0.2\text{mA}$	0.8VDD		VDD	V
输出低电平	VOO	$I_{OO} = 1.2\text{mA}$	0	-	0.2VDD	V
模块工作电流	IDD	$VDD = 3.0\text{V}$	-	0.3	1.0	mA
背光工作电流	ILED	$V_{LED}=3.0\text{V}$ (共 8 颗 LED 灯并联)	48	90	120	mA

表 3: 直流 (DC) 参数

## 6. 读写时序特性

### 6.1 串行接口:

从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

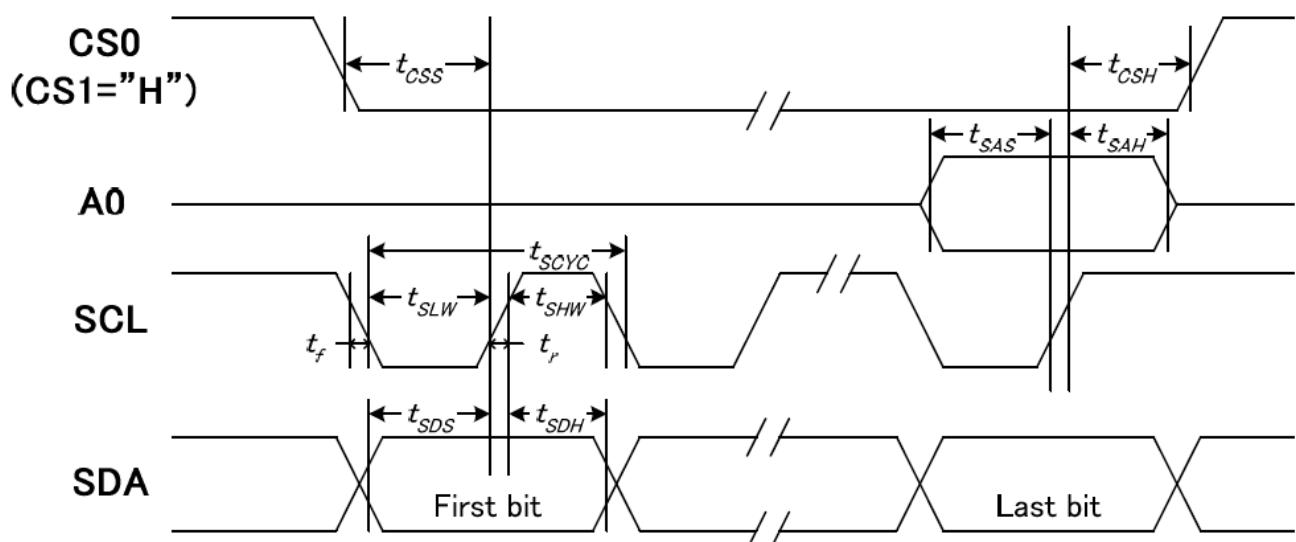


图 3. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

## 6.2 串行接口: 时序要求 (AC 参数):

写数据到 ST7525 的时序要求:

Item	Signal	Symbol	Condition	Min.	Max.	Unit
Serial clock period	SCL	tSCYC		110	-	ns
SCL "H" pulse width		tSHW		40	-	
SCL "L" pulse width		tSLW		40	-	
Address setup time	A0	tSAS		10	-	ns
Address hold time		tSAH		10	-	
Data setup time	SDA	tSDS		20	-	ns
Data hold time		tSDH		10	-	
CS0 setup time	CS0	tCSS		20	-	ns
CS0 hold time		tCSH		10	-	

表 4

## 6.3 并行接口: (8080)

从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

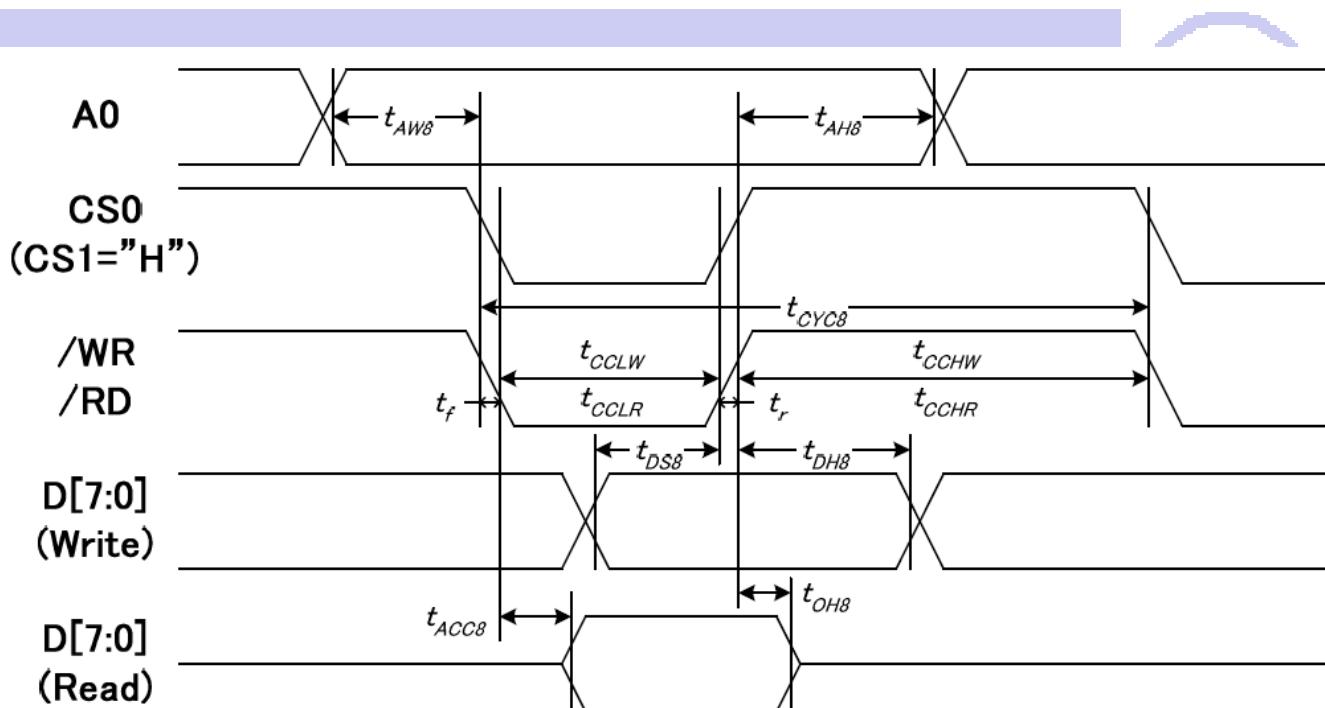


图 4. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

## 6.4 并行接口: 时序要求 (AC 参数):

写数据到 ST7525I 的时序要求: (8080 系列 MPU)

Item	Signal	Symbol	Condition	Min.	Max.	Unit
Address setup time	A0	tAW8		5	-	ns
Address hold time		tAH8		10	-	
System write cycle time	WR	tCYC8		190	-	ns
Write L pulse width		tCCLW		80	-	
Write H pulse width		tCCHW		80	-	
Read L pulse width	RD	tCCLR		100	-	ns
Read H pulse width		tCCHR		100	-	
Data setup time (Write)	D[7:0]	tDS8		60	-	ns
Write Data hold time (Write)		tDH8		5	-	

表 5

## 6.5 并行接口: (6800)

从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

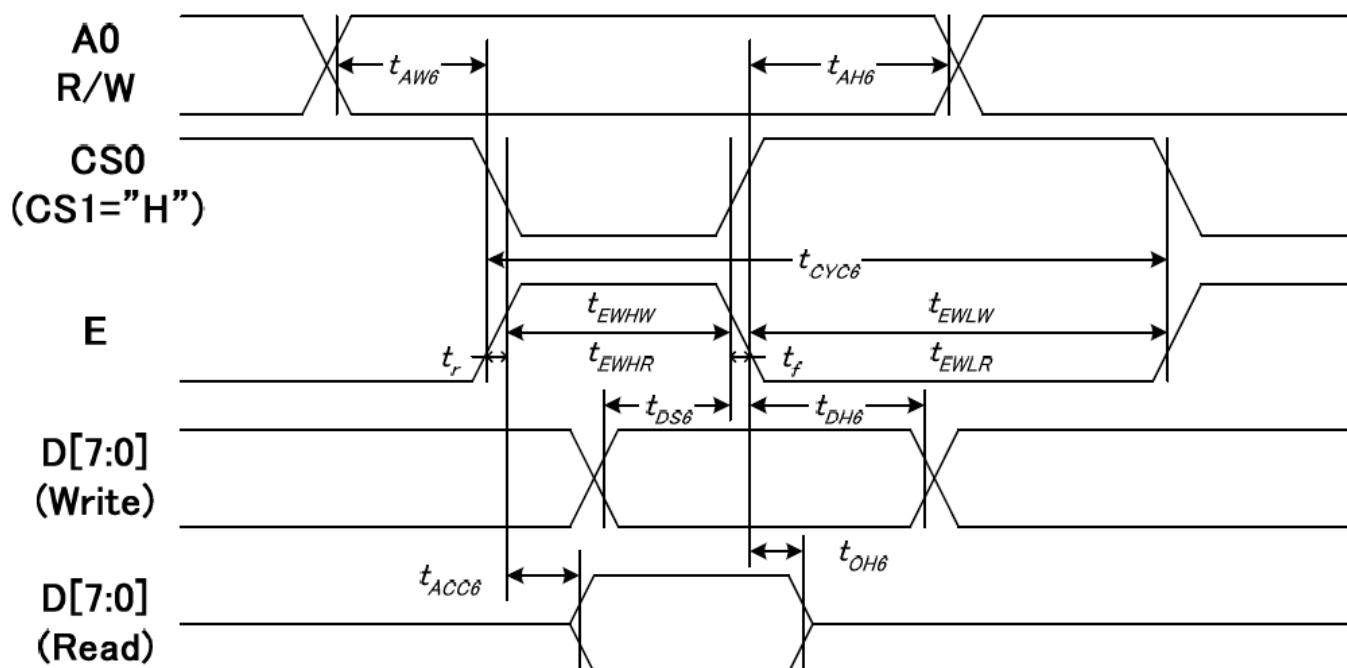


图 5. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

## 6.6 并行接口: 时序要求 (AC 参数):

写数据到 ST7525 的时序要求: (6800 系列 MPU)

Item	Signal	Symbol	Condition	Min.	Max.	Unit
Control setup time	E	tAW6		5	-	ns
Control hold time		tAH6		10	-	
System cycle time		tCYC6		190	-	
Enable H pulse width (WRITE)		tEWHW		80	-	
Enable L pulse width (WRITE)		tEWLW		100	-	
Enable H pulse width (READ)		tEWHR		100	-	
Enable L pulse width (READ)		tEWLR		100	-	
Write data setup time		tDS6		60	-	
Write data hold time	D[7:0]	tDH6		5	-	

表 6

## 6.7 IIC 接口:

从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

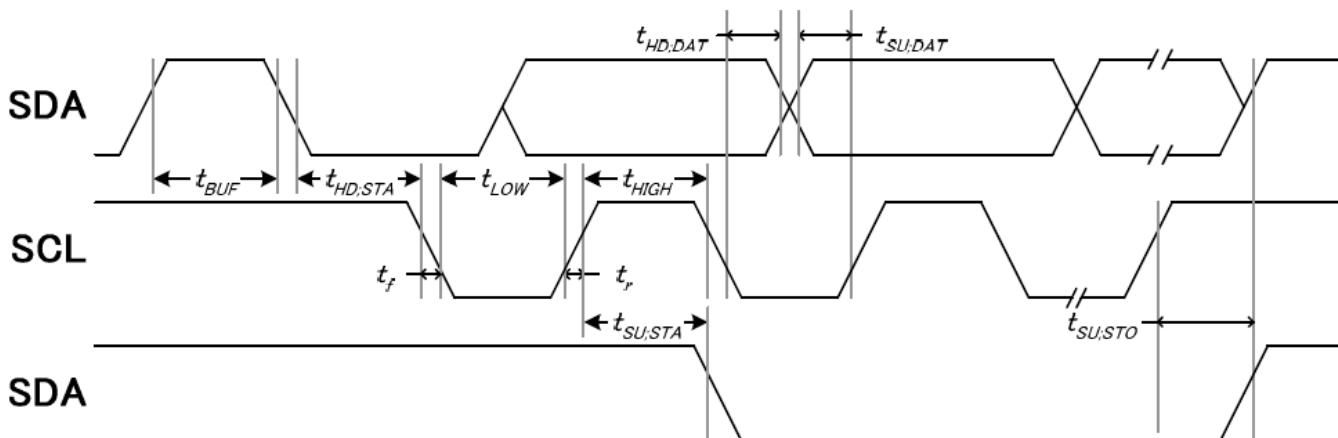


图 6. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

## 6.8 IIC 接口: 时序要求 (AC 参数):

写数据到 ST7525 的时序要求:

Item	Signal	Symbol	Condition	Rating		Unit
				Min.	Max.	
SCL clock frequency	SCL	fSCL		-	400	kHz
SCL clock low period		tLOW		1.3	-	
SCL clock high period		tHIGH		0.6	-	
Data set-up time	SDA	tSU;Data		0.1	-	us
Data hold time		tHD;Data		0	0.9	
Setup time for a repeated START condition		tSU;STA		0.6	-	
Start condition hold time		tHD;STA		0.6	-	
Setup time for STOP condition		tSU;STO		0.6	-	
Bus free time between a STOP and START		tBUF		0.1	-	
Signal rise time	SCL	tr		20+0.1Cb	300	ns
Signal fall time		tf		20+0.1Cb	300	
Capacitive load represented by each bus line		Cb		-	400	pF
Tolerable spike width on bus		tSW		-	50	ns

表 7

## 6.9 电源启动后复位的时序要求 (RESET CONDITION AFTER POWER UP):

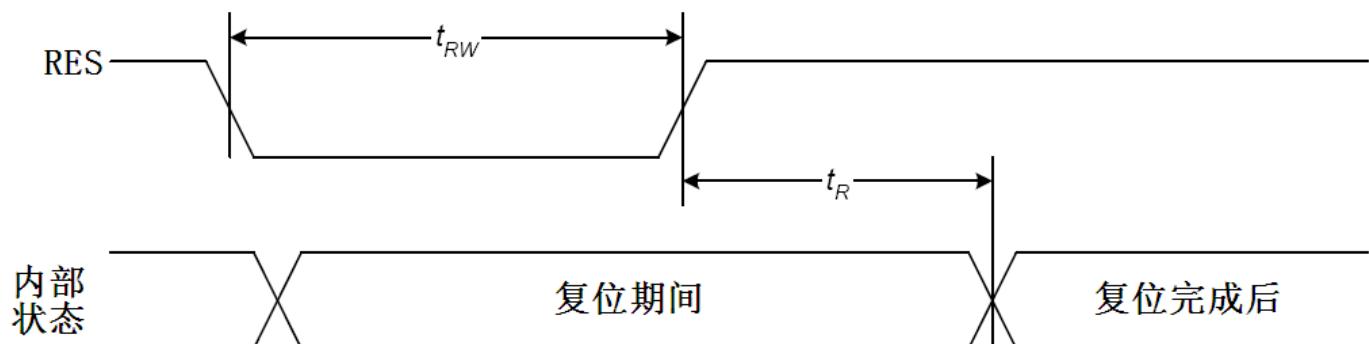


图 7: 电源启动后复位的时序

表 8: 电源启动后复位的时序要求

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
复位时间	$t_R$		100	—	—	ms
复位保持低电平的时间	$t_{RW}$	引脚: RES	100	—	—	ms

## 7. 指令功能:

### 7.1 指令表

表 9.

指令名称	指令码										说明
	A0	WR	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(1) 写显示数据到液晶屏 (Write Data)	1	0	8 位显示数据								从 CPU 写数据到液晶屏, 每一位对应一个点阵, 1 个字节对应 8 个竖置的点阵
(2) 读液晶屏的显示数据(Read Data)	1	1	8 位显示数据								并口时: 读已经显示到液晶屏上的点阵数据。串口时不能用此指令。
(3) 读取状态字 (并行接口) Read Status Byte (parallel interface)	0	1	ID0	MX	MY	WA	DE	0	0	0	读取状态字节 仅用于并行接口
			0	0	0	0	0	o	ID2	ID1	
(4)	列地址低 4 位设置	0	0	0	0	0	列地址的低 4 位				高 4 位与低 4 位共同组成列地址, 指定 192 列中的其中一列。比如液晶模块的第 100 列地址十六进制为 <b>0x64</b> , 那么此指令由 2 个字节来表达: <b>0x16,0x04</b>
	列地址高 4 位设置		0	0	0	1	列地址的高 4 位				
(5) 设置滚动线 (Set Scroll Line)	0	0	0	1	SL5	SL4	SL3	SL2	SL1	SL0	为指定行地址 DDRAM 第一显示行 (垂直滚动)
(6) 设置页地址 ( Set Page Address)	0	0	1	0	1	1	显示页地址, 共 4 位				设置页地址。每 8 行为一个页, 64 行分为 8 个页, 可设置值为: <b>0XB0~0XB8</b> 分别对应第一页到第九页, 第九页是一个单独的一行图标, 本液晶屏没有这一行图标, 所以设置值为 <b>0XB0~0XB7</b> 分别对应第一页~第八页。

(7) 设置对比度 ( Set Contrast)	内部设置液晶电压模式 设置的电压值	0	0	1	0	0	0	0	0	1	上面一条指令 0x81 是不改的, 下面一条指令可设置范围为: 0x00~0xFF, 数值越大对比度越浓, 越小越淡
(8) 设置显示模式 (Set Partial Screen Mode)		0	0	1	0	0	0	0	1	0	0x74, 全部显示 0x75, 部分显示
(9) 设置RAM地址控 ( Set RAM Address Control)		0	0	1	0	0	0	1	AC2	AC1	AC0
(10) 设置帧频率 ( Set Frame Rate)		0	0	1	0	1	0	0	FR1	FR0	0xA0: 76 帧/秒; 0xA1: 95 帧/秒; 0xA2: 132 帧/秒; 0xA3: 168 帧/秒;
(11) 显示全部点阵 (Set All Pixel ON)		0	0	1	0	1	0	0	1	0	0 1 0xA4: 常规 0xA5: 显示全部点阵
(12) 显示正显 / 反显 ( Set Inverse Display)		0	0	1	0	1	0	0	1	1	0 1 0xA6: 常规: 正显 0xA7: 反显
(13) 显示开 / 关 ( Set Display Enable)		0	0	1	0	1	0	1	1	0	0 1 0XAЕ: 关, 0XA贲: 开
(14) 扫描方向 ( Scan Direction)		0	0	1	1	0	0	0	MY	MX	0 MY=0 : COM0 ➔ COM63 MX=0 : SEG0 ➔ SEG191 MY=1: COM63 ➔ COM0 MX=1: SEG191 ➔ SEG0
(15) 软件复位 ( Software Reset)		0	0	1	1	1	0	0	1	0	0XE2 :软件复位。
(16) 空指令 ( NOP)		0	0	1	1	1	0	0	1	1	空操作
(17) 设置偏压比 ( Set Bias)		0	0	1	1	1	0	1	0	BR1	BR0  BR1 0 0 1 1 BR0 0 1 0 1 Bias 1/6 1/7 1/8 1/9
(18) 设置 COM 结束 ( Set COM End)		0	0	1	1	1	1	0	0	1	0xF1: 此双字节指令将显示负载设置在 1/(9+1) 到 1/(64+1) 的范围内, 以实现局部显示。
		0	0	-	-	CEN5	CEN4	CEN3	CEN2	CEN1	CENO
(19) 设置部分起始地址 ( Set Partial Start Address)		0	0	1	1	1	1	0	0	1	0 0xF2: 此指令可选择部分屏幕显示的起始行地址。
		0	0	-	-	DST5	DST4	DST3	DST2	DST1	DST0
(20) 设置部分结束地址 ( Set Partial End Address)		0	0	1	1	1	1	0	0	1	1 0xF3: 此指令可选择部分屏幕显示的结束行地址。
		0	0	-	-	DEN5	DEN4	DEN3	DEN2	DEN1	DENO
(21) 设置测试控制 ( Set Test Control)		0	0	1	1	1	1	0	0	0	0 0xF0: 该指令可以选择测试命令表。
		0	0	-	-	-	-	-	-	H1	H0
(22) 读取状态字节 (适用于 4_SPI) (Read Status Byte (for 4 line SPI))		0	0	1	1	1	1	1	1	0	0 0xFE: 指示四线 SPI 使用的状态
		0	1	ID0	MX	MY	WA	DE	0	0	0
		0	1	0	0	0	0	0	0	ID2	ID1

(23) 读数据 (适用于 4_SPI) (Read Data (for 4 line SPI))	0	0	1	1	1	1	1	1	1	1	<b>0xFF:</b> 可以从列地址和页面地址指定的 RAM 位置读取显示数据的 8 位数据到微处理器。
	1	1	D7	D6	D5	D4	D3	D2	D1	D0	

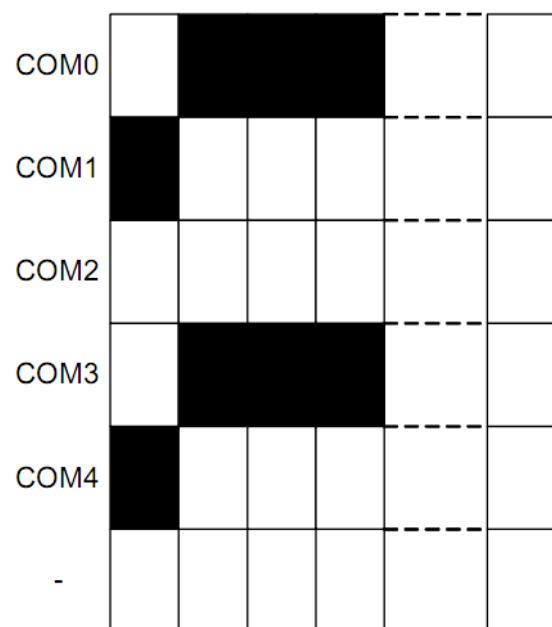
## 7.2 点阵与 DD RAM(显示数据存储器) 地址的对应关系

请留意页的定义: PAGE, 与平时所讲的“页”并不是一个意思, 在此表示 **8 个行就是一个“页”**, 一个 192\*32 点阵的屏分为 4 个“页”, 从第 0 “页” 到第 3 “页”。

**DB7—DB0 的排列方向:** 数据是从下向上排列的。最低位 D0 是在最上面, 最高位 D7 是在最下面。每一位 (bit) 数据对应一个点阵, 通常“1”代表点亮该点阵, “0”代表关掉该点阵. 如下图所示:

D0	0	1	1	1			0
D1	1	0	0	0			0
D2	0	0	0	0			0
D3	0	1	1	1			0
D4	1	0	0	0			0
-							

Display data RAM  
(显示数据存储器)

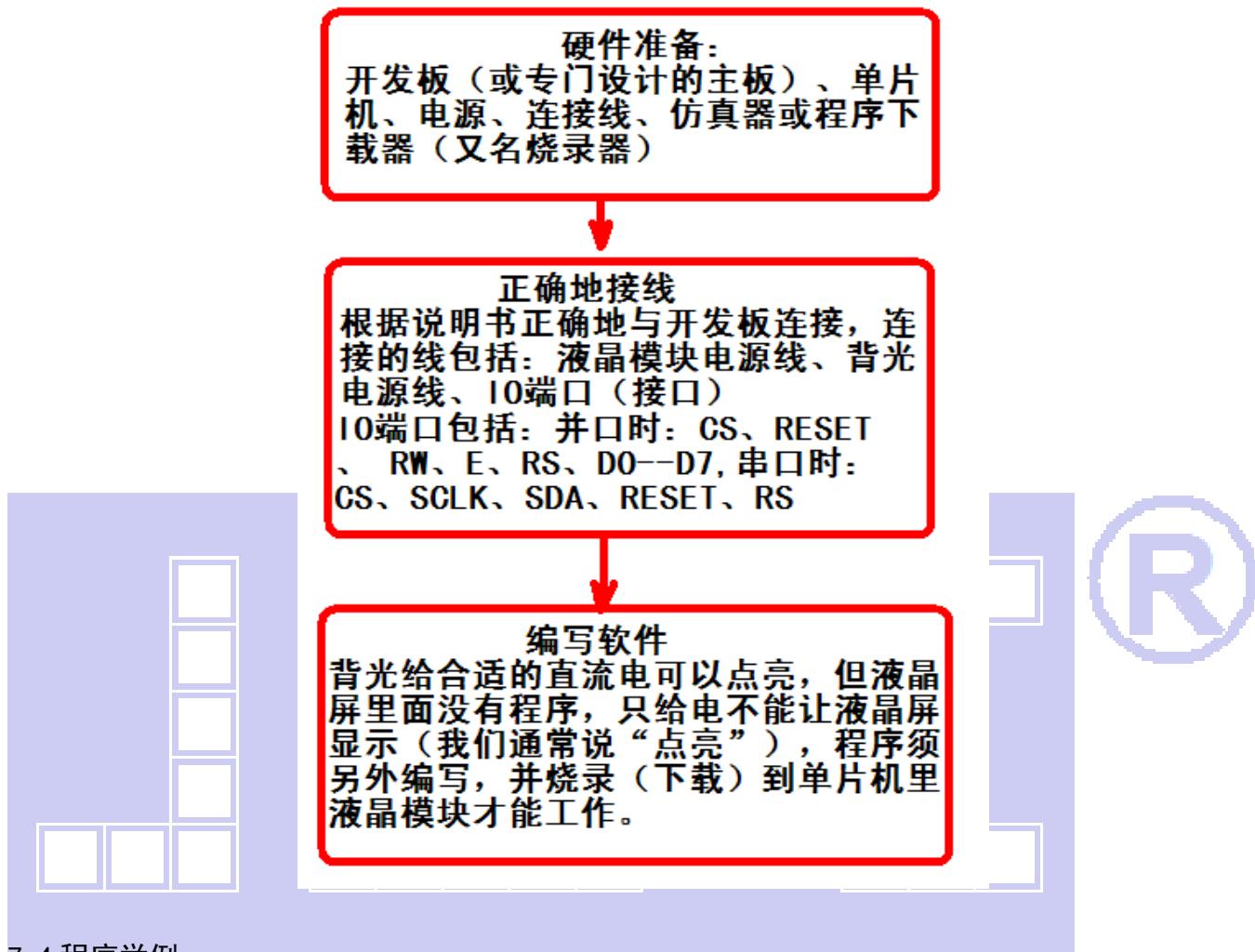


Liquid crystal display  
(液晶屏)

## 7.3 初始方法

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



## 7.4 程序举例:

液晶模块与 MCU(以 8051 系列单片机为例) 接口图如下:

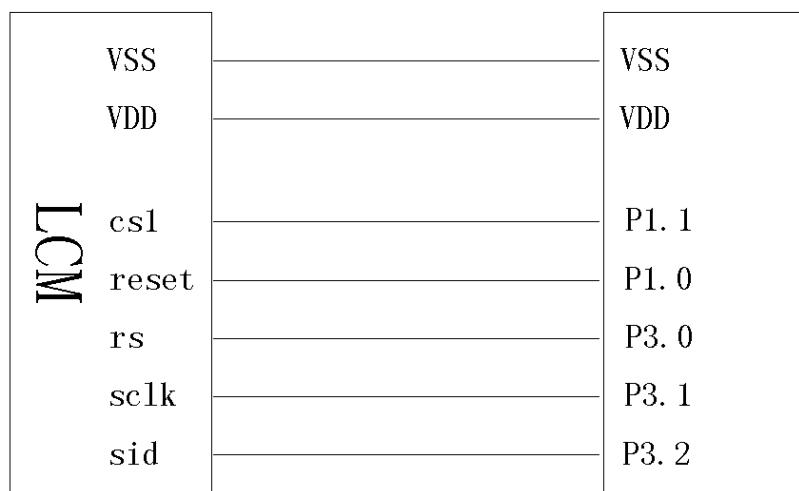


图 8. 串行接口

```
// JLX19232G-9808-PN-S
// 串行接口!!!
// 驱动 IC 是:ST7525

#include <STC15F2K60S2.H>
#include <intrins.h>
#include <Chinese_code.h>

sbit cs1=P1^1; /*对应 LCD 的 CS 引脚*/
sbit reset=P1^0; /*对应 LCD 的 RST 引脚*/
sbit rs=P3^0; /*对应 LCD 的 RS 引脚*/
sbit sclk=P3^1; //对应 LCD 的 SCK(D0)
sbit sid=P3^2; //对应 LCD 的 SDA(D1)

sbit key=P2^0;
```

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

```
//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;
        delay_us(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;
//      delay_us(1);
        data1=data1<<=1;
    }
    cs1=1;
}

//延时 1
void delay_ms(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_ms(2500);
}

//LCD 模块初始化
void initial_lcd()
{
    reset=0;          //低电平复位
    delay_ms(100);
    reset=1;          //复位完毕
    delay_ms(200);
    transfer_command(0xe2); //软复位
    delay_ms(200);
    transfer_command(0xa0); //
    transfer_command(0x81); //微调对比度
    transfer_command(0xb5); //微调对比度的值，可设置范围 0x00~0xFF
    transfer_command(0xe9); //1/7 偏压比 (bias)
    transfer_command(0xc4); //行列扫描顺序：从上到下
}

```

```

    transfer_command(0xf1); //设置部分显示
    transfer_command(0x1F); //设置显示范围: 0--31 行
    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<4;i++)
    {
        lcd_address(1+i, 1);
        for(j=0;j<192;j++)
        {
            transfer_data(0x00);
        }
    }
}

```

```

void display_graphic_192x32(uchar *dp)
{
    uchar i, j;
    for(i=0;i<4;i++)
    {
        lcd_address(i+1, 1);
        for(j=0;j<192;j++)
        {
            transfer_data(*dp);
            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++;  
        }  
    }  
}
```

//写入一组 16x16 点阵的汉字字符串（字符串表格中需含有此字）

//括号里的参数: (页, 列, 汉字字符串)

```
void display_string_16x16(uchar page, uchar column, uchar reverse, uchar *text)  
{
```

```
    uchar i, j, k, data1;  
    uint address;  
    j = 0;  
    while(text[j] != '\0')  
    {  
        i=0;  
        address=1;  
        while(Chinese_text_16x16[i]> 0x7e )  
        {  
            if(Chinese_text_16x16[i] == text[j])  
            {  
                if(Chinese_text_16x16[i+1] == text[j+1])  
                {  
                    address = i*16;  
                    break;  
                }  
            }  
            i +=2;  
        }  
        if(column>191)  
        {  
            column =0;  
            page +=2;  
        }  
        if(address !=1)  
        {  
            for(k=0;k<2;k++)  
            {  
                lcd_address(page+k, column);  
            }  
        }  
    }  
}
```

```
for (i=0;i<16;i++)
{
    if(reverse==1) data1=˜Chinese_code_16x16[address];
    else data1=Chinese_code_16x16[address];
    transfer_data(data1);
    address++;
}
}
j +=2;
}
else
{
    for (k=0;k<2;k++)
    {
        lcd_address(page+k, column);
        for(i=0;i<16;i++)
        {
            if(reverse==0) transfer_data(0x00);
            else transfer_data(0xff);
        }
        j++;
    }
    column +=16;
}
}

//显示 8x16 的点阵的字符串, 括号里的参数分别为 (页, 列, 字符串指针)
void display_string_8x16_2(uchar page, uchar column, uchar reverse, uchar *text)
{
    uchar data1;
    uint i=0, j, k, n;

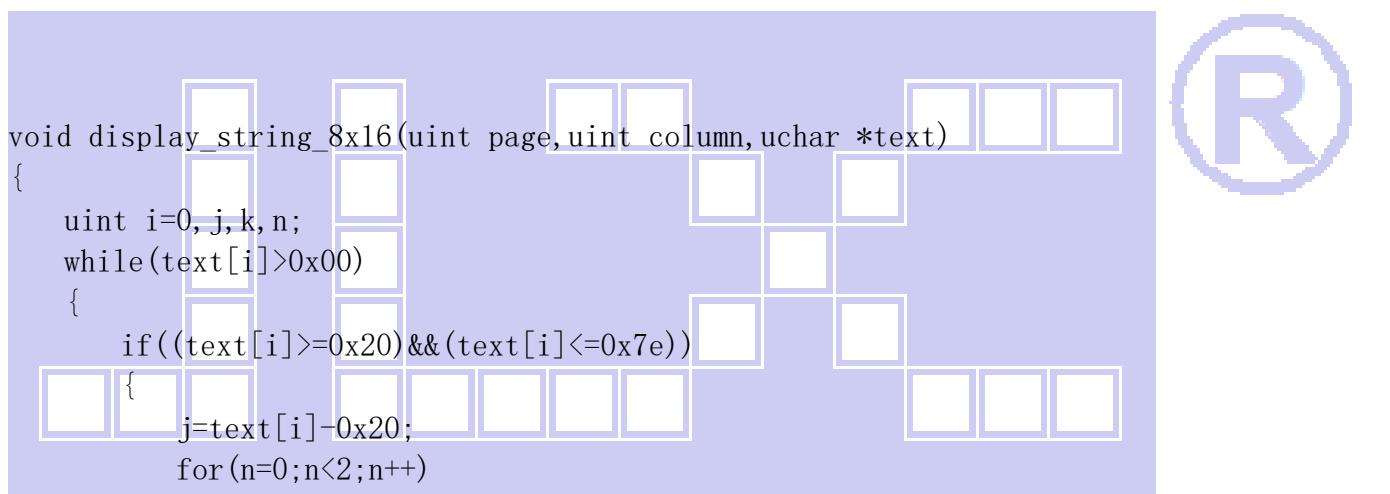
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            for(n=0;n<2;n++)
            {
                lcd_address(page+n, column);
                for(k=0;k<8;k++)
                {
                    if(reverse==1) data1=˜ascii_table_8x16[j][k+8*n];
                    else data1=ascii_table_8x16[j][k+8*n];
                    transfer_data(data1);
                }
            }
        }
    }
}
```

```

        }
        if(reverse==0)    transfer_data(0x00);
        else    transfer_data(0xff);
    }
    i++;
    column+=8;
}
else
i++;

if(column>127)
{
    column=0;
    page+=2;
}
}
}

```



```

void display_string_8x16(uint page,uint column,uchar *text)
{
    uint i=0,j,k,n;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            for(n=0;n<2;n++)
            {
                lcd_address(page+n,column);
                for(k=0;k<8;k++)
                {
                    transfer_data(ascii_table_8x16[j][k+8*n]); //显示 5x7 的 ASCII 字到 LCD
                }
            }
            i++;
            column+=8;
        }
        else
        i++;
    }
}

```

上, y 为页地址, x 为列地址, 最后为数据

```
//显示一串 5x8 点阵的字符串
//括号里的参数分别为 (页, 列, 是否反显, 数据指针)
void display_string_5x8(uint page, uint column, uchar reverse, uchar *text)
{
    uchar i=0, j, k, data1;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(page, column);
            for(k=0;k<5;k++)
            {
                if(reverse==1)    data1=~ascii_table_5x8[j][k];
                else    data1=ascii_table_5x8[j][k];
                transfer_data(data1);
            }
            if(reverse==1)    transfer_data(0xff);
            else    transfer_data(0x00);
            i++;
            column+=6;
        }
        else
            i++;
    }
}

void display_string_5x8_1(uint page, uint column, uchar *text)
{
    uint i=0, j, k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(page, column);
            for(k=0;k<5;k++)
            {
                transfer_data(ascii_table_5x8[j][k]); //显示 5x7 的 ASCII 字到 LCD 上, y 为页地址, x 为列地址, 最后为数据
            }
            i++;
            column+=6;
        }
        else
    }
}
```



```

    i++;
}

}

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
    initial_lcd(); //LCD 初始化
    while(1)
    {
        clear_screen(); //clear all dots
        display_graphic_192x32(bmp2);
        waitkey();
        clear_screen(); //clear all dots
        display_graphic_192x32(bmp1);
        waitkey();
        clear_screen();
        display_string_5x8(1, 1, 1, "      MENU      ");
        display_string_5x8(3, 1, 0, "Select>>>");
        display_string_5x8(3, 100, 1, "1. Graphic");
        display_string_5x8(4, 100, 0, "2. Chinese");
        waitkey();
        clear_screen();
        display_string_8x16_2(1, 1, 1, "--");
        display_string_16x16(1, 17, 1, "->粉尘测试      ");
        display_string_16x16(3, 33, 0, "一般测试");
        waitkey();
        clear_screen();
        display_graphic_32x32(1, 49, cheng1); //在第 1 页, 第 49 列显示单个汉字"成"
        display_graphic_32x32(1, 89, gon); //在第 1 页, 第 49 列显示单个汉字"成"

        waitkey();
        clear_screen(); //clear all dots
        display_string_8x16(1, 1, "(<\\"0123456abt^`!@#$%^\\>)"); //在第 1 页, 第 1 列显示字符串
        display_string_8x16(3, 1, "{[(<\\"*&*\|\\"@#_+=' \">)]}"); //在第*页, 第*列显示字符串
        waitkey();
        clear_screen(); //clear all dots
    }
}

```

R

字符串, 括号里的参数分别为 (页, 列, 是否反显, 数据指针)

成"

```

        display_string_5x8_1(1, 1, "[!#$%&' ()*+, -. /0123456789: ;<=>?]");
        display_string_5x8_1(2, 1, "[ABCDEFGHIJKLMNPQRSTUVWXYZabcd]");
        display_string_5x8_1(3, 1, "(abcdefghijklmnopqrstuvwxyzabcd)");
        display_string_5x8_1(4, 1, "{[(<\`' &*| \\@abcde012#_-+=` \">)]}']");
        waitkey();
    }
}

```

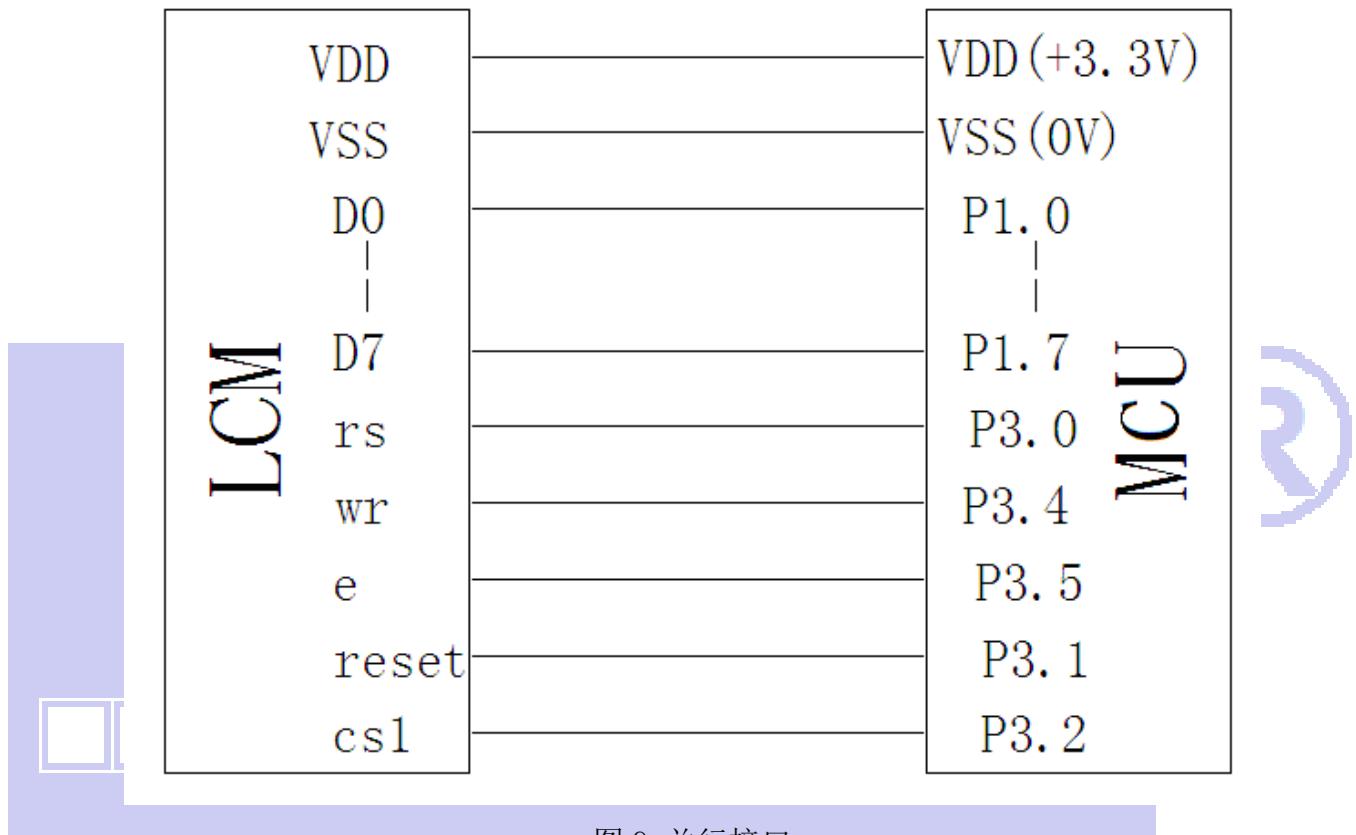


图 9. 并行接口

并行程序与串行只是接口定义、写数据和命令不一样，其它都一样

### 并行程序:

```

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

sbit cs1=P3^2;      //对应 LCD 的 CS
sbit reset=P3^1;    //对应 LCD 的 RST
sbit rs=P3^0;       //对应 LCD 的 RS
sbit e=P3^5;        //对应 LCD 的 RD (E)
sbit wr=P3^4;       //对应 LCD 的 WR
sbit key=P2^0;      /*按键接口，P2.0 口与 GND 之间接一个按键*/

```

```

//写指令到 LCD 模块
void transfer_command(int data1)
{
    cs1=0;
    rs=0;
    wr=0;
    e=0;
    P1=data1;

```

```

e=1;
e=0;
P1=0x00;
cs1=1;
}
//写数据到 LCD 模块
void transfer_data(int data1)
{
    cs1=0;
    rs=1;
    wr=0;
    e=0;
    P1=data1;
    e=1;
    e=0;
    P1=0x00;
    cs1=1;
}

```

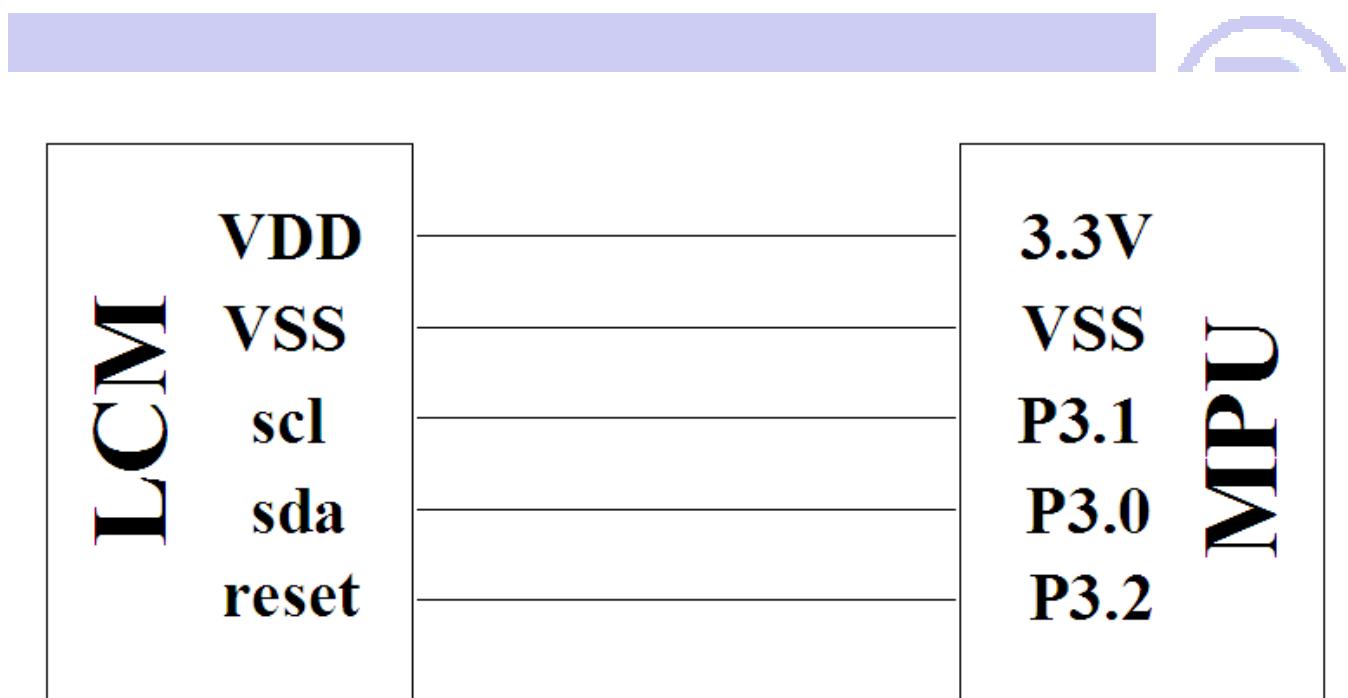


图 10. IIC 接口  
IIC 程序与串、并行接口定义、写数据和命令不一样，其他都是一样的

### IIC 程序：

```

// 液晶演示程序 JLX19232G-9808, IIC 接口!
// 驱动 IC 是:ST7525

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

sbit reset=P3^2; // 对应 LCD 的 RST

```

```

sbit scl=P3^1;      //对应 LCD 的 SCK (D0)
sbit sda=P3^0;      //对应 LCD 的 SDA (D1)
sbit key=P2^0;

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();  
}
```

