

JLX19232G-9809-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	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	11~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-9809 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-9809 可以显示 192 列*32 行点阵单色图片，或显示 12 个/行*2 行 16*16 点阵的汉字，或显示 24 个/行*4 行 5*8 点阵的英文、数字、符号。

2. JLX19232G-9809 图像型点阵液晶模块的特性

2.1 结构轻、薄、带背光、焊接式 FPC。

2.2 IC 采用 ST7525, 功能强大，稳定性好

2.3 功耗低: 当电压为 3.3V 时，功耗低: 不带背光 1.32mW (3.3V*0.4mA)，带背光不大于 333mW (3.3V*100.4mA)；

2.4 显示内容:

(1) 192*32 点阵单色图片，或其它小于 192*32 点阵的单色图片；

(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；

(3) 按照 8*16 点阵汉字来计算可显示 24 字*2 行；

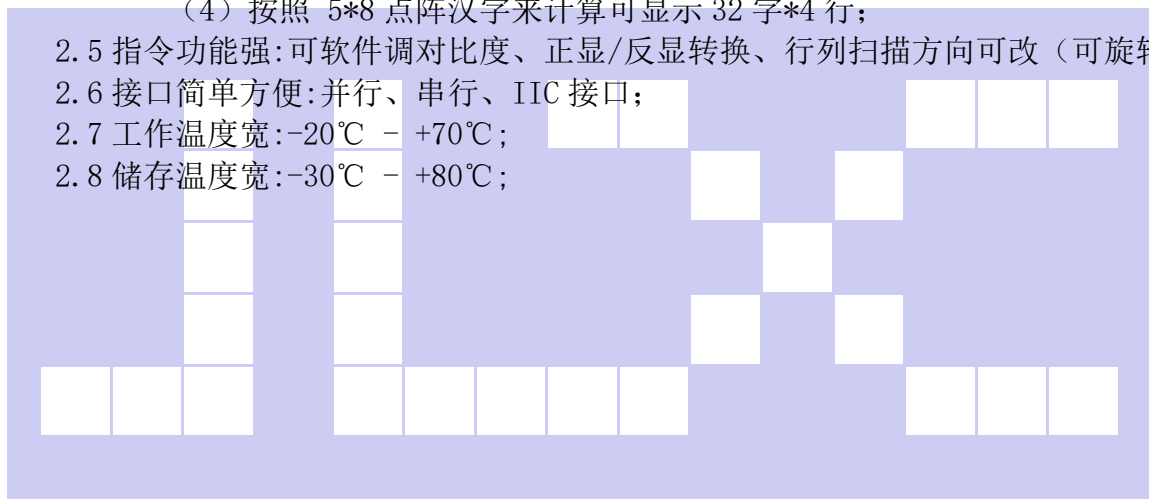
(4) 按照 5*8 点阵汉字来计算可显示 32 字*4 行；

2.5 指令功能强: 可软件调对比度、正显/反显转换、行列扫描方向可改 (可旋转 180 度使用)。

2.6 接口简单方便: 并行、串行、IIC 接口；

2.7 工作温度宽: -20℃ - +70℃；

2.8 储存温度宽: -30℃ - +80℃；



3. 外形尺寸及接口引脚功能

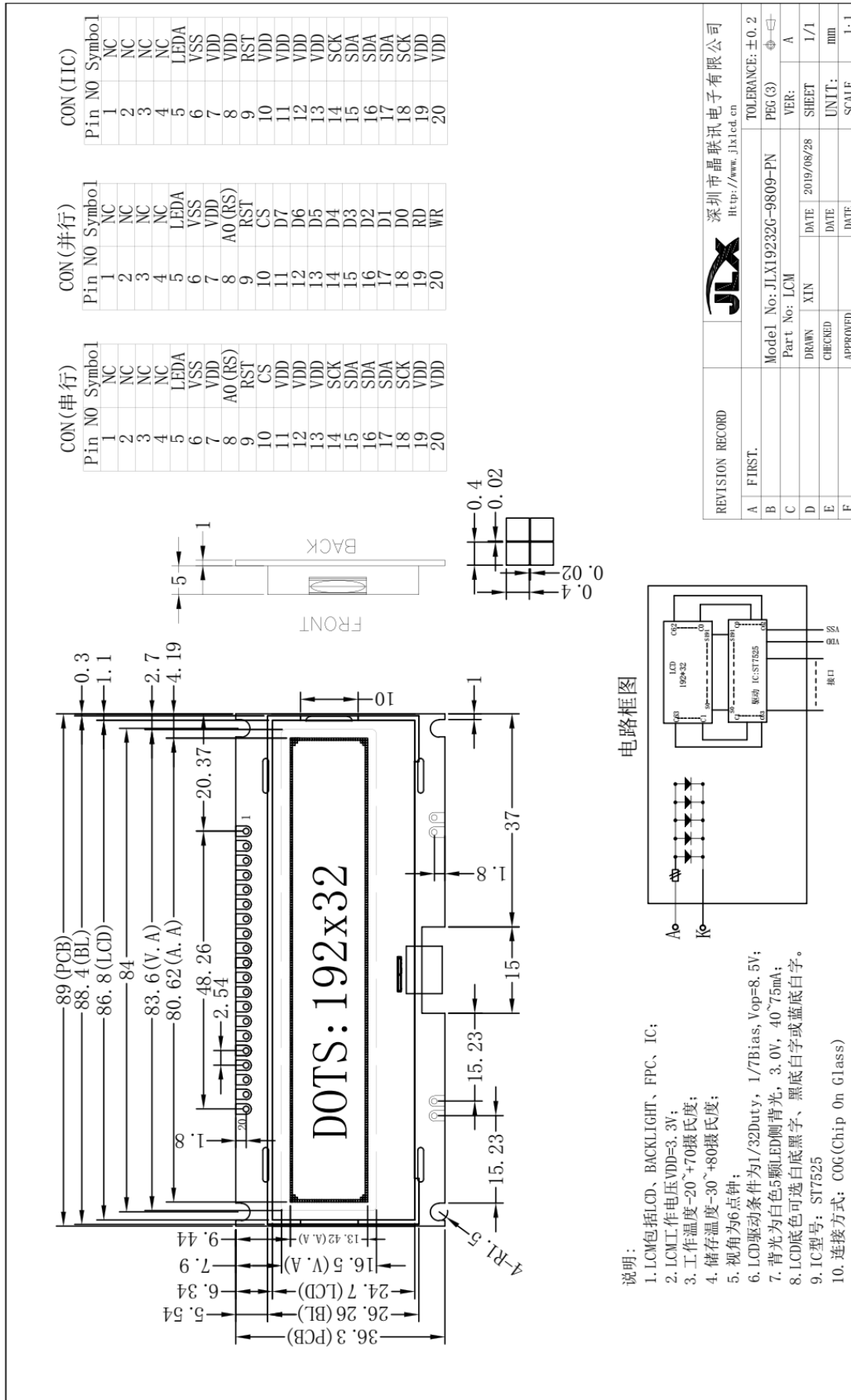


图 1. 外形尺寸

模块的接口引脚功能:

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: 接 VDD
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
12	D6	I/O	并行接口时: 数据总线 DB6 IIC/串行接口时: 接 VDD
13	D5	I/O	并行接口时: 数据总线 DB5 IIC/串行接口时: 接 VDD
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 短接一起做 SCK)
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 短接一起做 SDA)
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 短接一起做 SDA)
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 短接一起做 SDA)
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 短接一起做 SCK)
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: 接 VDD
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: 接 VDD

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 192×64 点阵, 192 个列信号与驱动 IC 相连, 64 个行信号也与驱动 IC 相连, IC 邦定在 LCD 玻璃上 (这种加工工艺叫 COG)。

4.2 工作电图:

电路框图

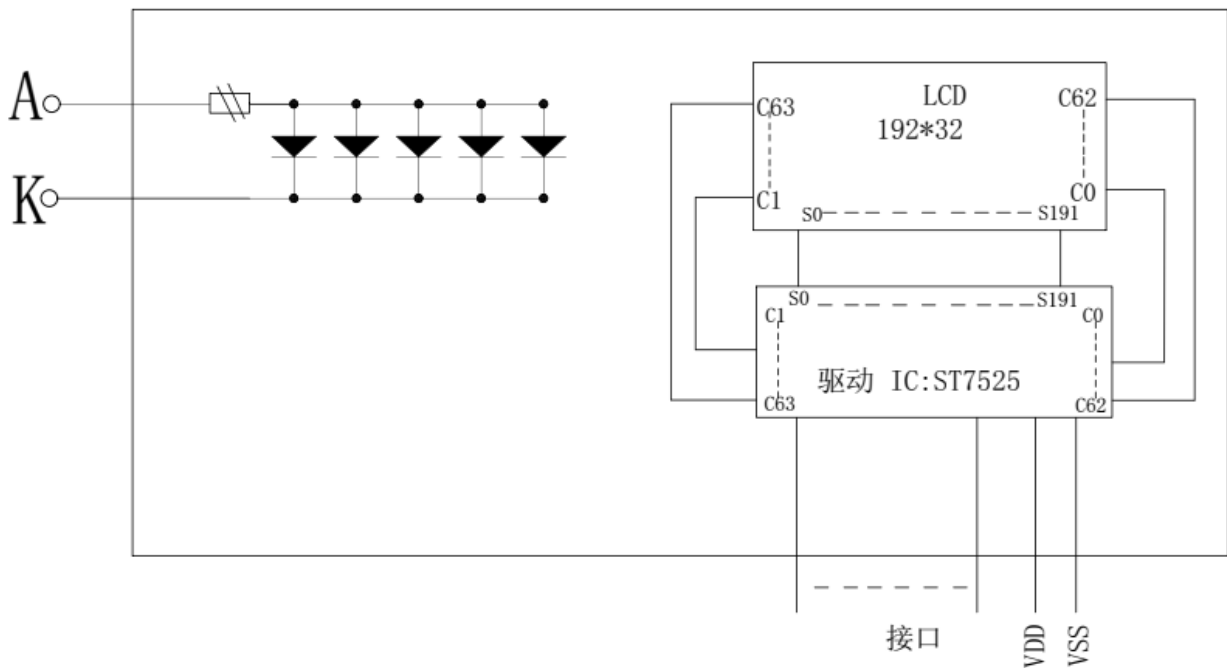


图 2: JLX19232G-9809 图像点阵型液晶模块的电路框图

4.2 背光参数

该型号液晶模块带 LED 背光源。它的性能参数如下:

背光板可选择白色。

正常工作电流为: 48~120mA (LED 灯数共 5 颗);

工作电压: 3.0V

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	
电路电源	VDD - VSS	-0.3		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		2.4	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	-	0.8xVDD		VDD	V
输入低电平	VIO	-	VSS		0.6	V
输出高电平	VOH	IOH = 0.2mA	0.8xVDD		VDD	V
输出低电平	VOO	IOO = 1.2mA	VSS		0.2xVDD	V
模块工作电流	IDD	VDD = 3.0V	-		0.3	mA
背光工作电流	ILED	VLED=3.0V	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)

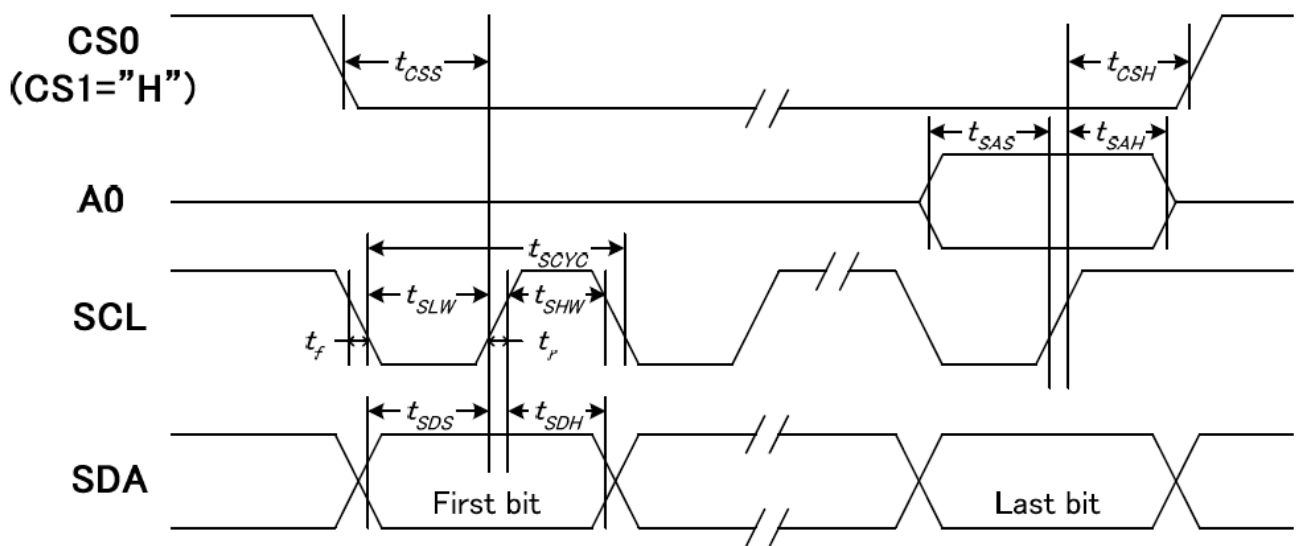


图 3. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

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	-	
Address hold time		tSAH		10	-	
Data setup time	SDA	tSDS		20	-	
Data hold time		tSDH		10	-	
CS0 setup time	CS0	tCSS		20	-	
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)

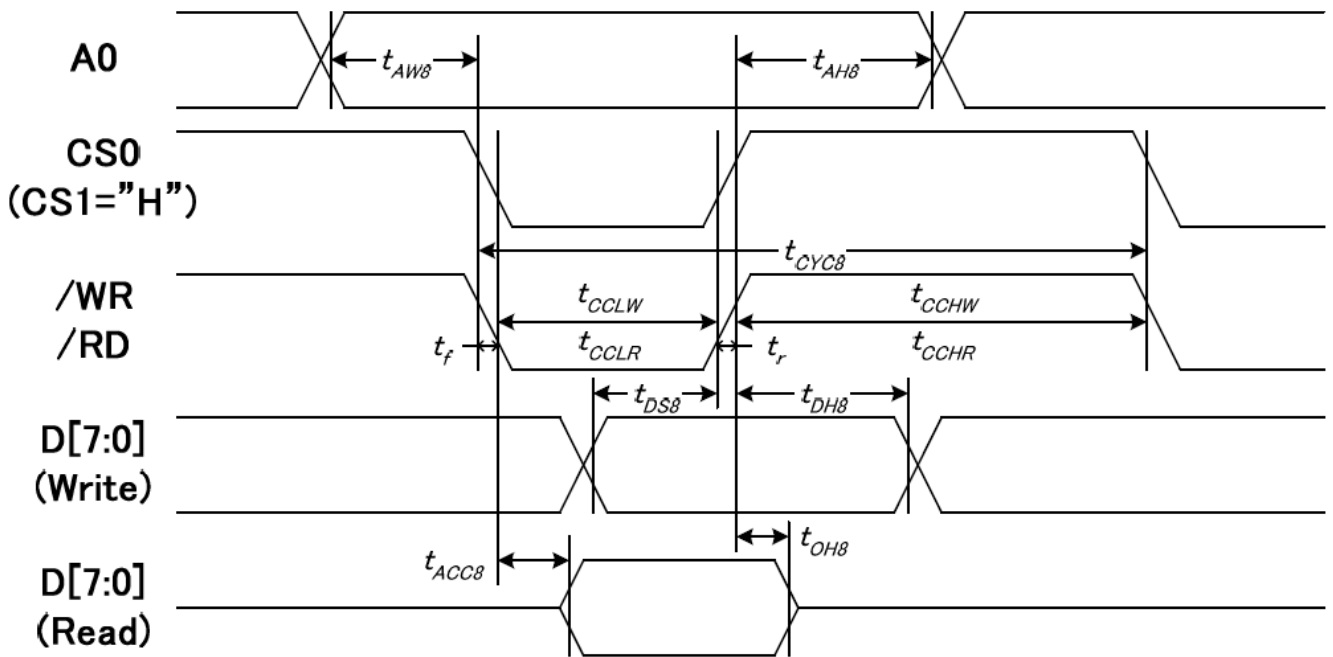


图 4. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

6.4 并行接口: 时序要求 (AC 参数):

写数据到 ST7525 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	-	
Write L pulse width		tCCLW		80	-	
Write H pulse width		tCCHW		80	-	
Read L pulse width	/RD	tCCLR		100	-	
Read H pulse width		tCCHR		100	-	
Data setup time (Write)	D[7:0]	tDS8		60	-	
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)

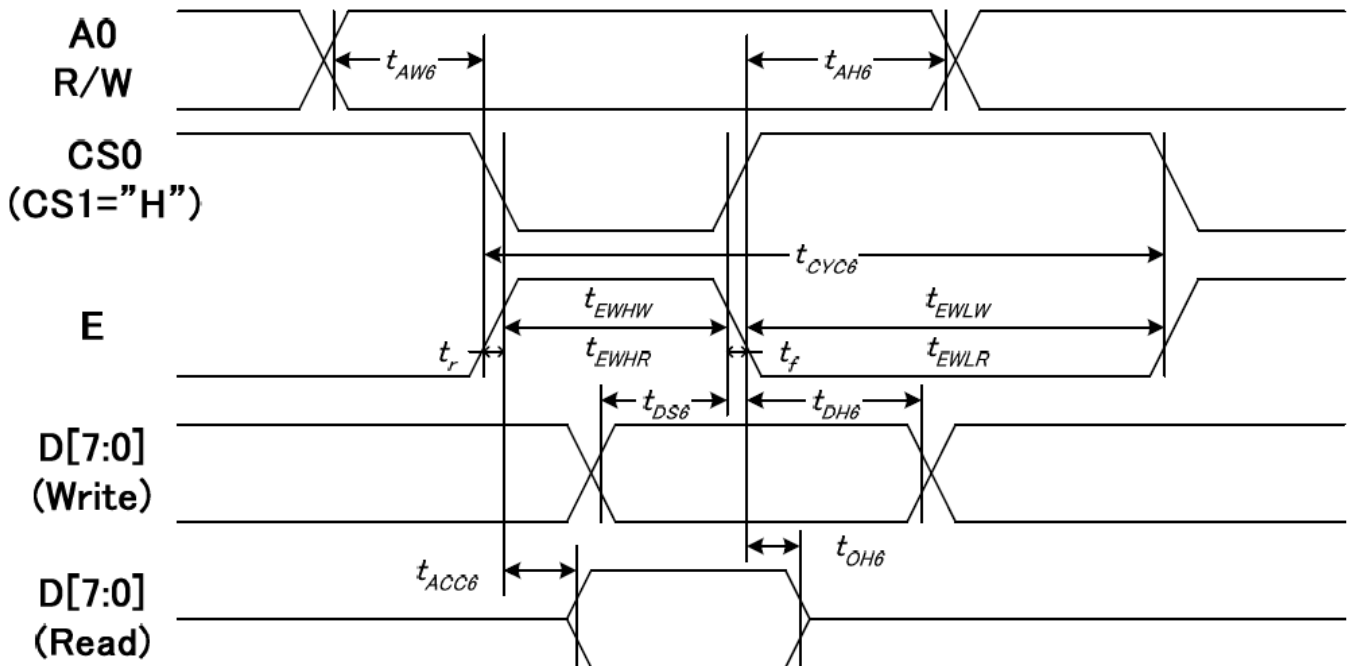


图 5. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

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	A0	tAW6		5	-	ns
Control hold time	R/W	tAH6		10	-	
System cycle time		tCYC6		190	-	
Enable H pulse width (WRITE)	E	tEHW		80	-	
Enable L pulse width (WRITE)		tEHL		100	-	
Enable H pulse width (READ)		tEWH		100	-	
Enable L pulse width (READ)		tEHL		100	-	
Write data setup time	D[7:0]	tDS6		60	-	
Write data hold time		tDH6		5	-	

表 6

6.7 IIC 接口:

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

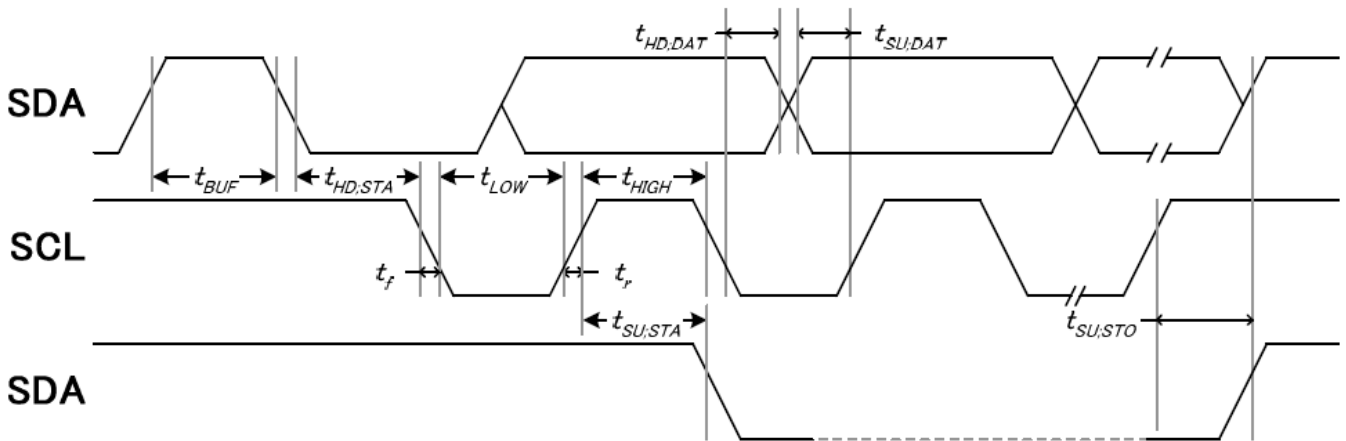


图 6. 从 CPU 写到 ST7525 (Writing Data from CPU to ST7525)

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	SDA	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):

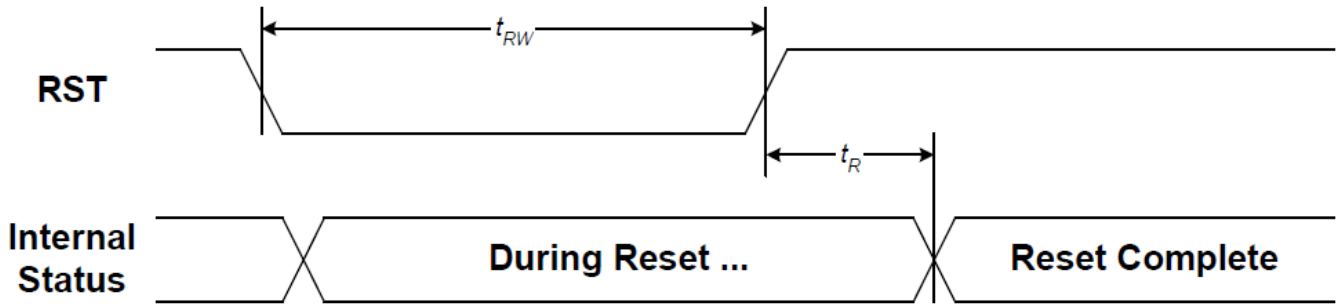


图 7: 电源启动后复位的时序

表 8

Item	Symbol	Condition	Min.	Max.	Unit
Reset time	tR		-	1	ms
Reset "L" pulse width	tRW		1	-	

7. 指令功能:

7.1 指令表

下表是“ST7525” IC 支持的指令:

CD: 0: 指令; 1: 数据 W/R: 0: 写; 1: 读 D7~D0: 有用的数据位; -: 不必理会的



表 9.



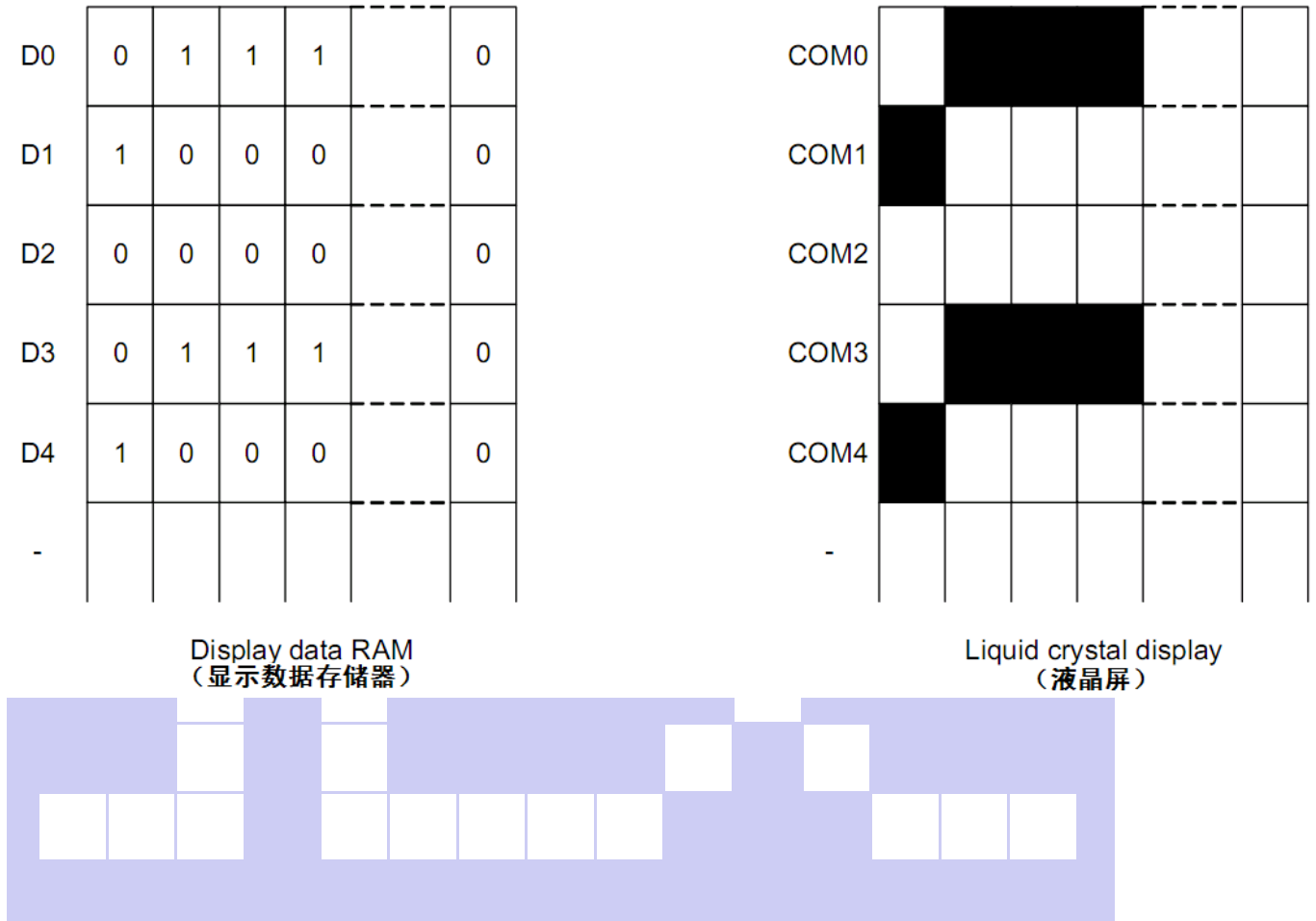
COMMAND TABLE												
INSTRUCTION	A0	R/W (RWR)	COMMAND BYTE								DESCRIPTION	
			D7	D6	D5	D4	D3	D2	D1	D0		
Write Data	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data to DDRAM	
Read Data	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from DDRAM Only for parallel interface and I ² C	
Read Status Byte (parallel interface)	0	1	ID0	MX	MY	WA	DE	0	0	0	Read status byte Only for parallel interface	
			0	0	0	0	0	0	ID2	ID1		
Set Column Address LSB	0	0	0	0	0	0	CA3	CA2	CA1	CA0	Set column address of RAM	
Set Column Address MSB	0	0	0	0	0	1	CA7	CA6	CA5	CA4		
Set Scroll Line	0	0	0	1	SL5	SL4	SL3	SL2	SL1	SL0	Specify line address for the 1 st display line of DDRAM (vertical scrolling)	
Set Page Address	0	0	1	0	1	1	PA3	PA2	PA1	PA0	Set page address of RAM	
Set Contrast	0	0	1	0	0	0	0	0	0	0	1	2-byte instruction. Set Vop voltage
			EV7	EV6	EV5	EV4	EV3	EV2	EV1	EV0		
Set Partial Screen Mode	0	0	1	0	0	0	0	1	0	PS	PS=1: Enable partial mode	
Set RAM Address Control	0	0	1	0	0	0	1	AC2	AC1	AC0	Set column and page address behavior	
Set Frame Rate	0	0	1	0	1	0	0	0	FR1	FR0	Set frame frequency	
Set All Pixel ON	0	0	1	0	1	0	0	1	0	AP	Set all display segments on	
Set Inverse Display	0	0	1	0	1	0	0	1	1	INV	Set inverse display	
Set Display Enable	0	0	1	0	1	0	1	1	1	PD	PD=0: Chip is in power down mode	
Scan Direction	0	0	1	1	0	0	0	MY	MX	0	Set COM and SEG scan direction	
Software Reset	0	0	1	1	1	0	0	0	1	0	Set software reset	
NOP	0	0	1	1	1	0	0	0	1	1	No operation	
Set Bias	0	0	1	1	1	0	1	0	BR1	BR0	Set internal bias circuit	
Set COM End	0	0	1	1	1	1	0	0	0	1	2-byte instruction. Set display duty	
			--	--	CEN5	CEN4	CEN3	CEN2	CEN1	CEN0		
Partial Start Address	0	0	1	1	1	1	0	0	1	0	Set partial start for partial display screen	
			--	--	DST5	DST4	DST3	DST2	DST1	DST0		
Partial End Address	0	0	1	1	1	1	0	0	1	1	Set partial end for partial display screen	
			--	--	DEN5	DEN4	DEN3	DEN2	DEN1	DEN0		
Test Control	0	0	1	1	1	1	0	0	0	0	Set test command table	
			--	--	--	--	--	--	H1	H0		

Serial Read Command Table (Enabled only in 4 line SPI)												
INSTRUCTION	A0	R/W (RWR)	COMMAND BYTE								DESCRIPTION	
			D7	D6	D5	D4	D3	D2	D1	D0		
Read Status Byte	0	0	1	1	1	1	1	1	1	1	0	Read status byte
	0	1	ID0	MX	MY	WA	DE	0	0	0	0	
Read Data	0	0	1	1	1	1	1	1	1	1	1	Read data from DDRAM
	1	1	D7	D6	D5	D4	D3	D2	D1	D0		

7.3 点阵与 DD RAM(显示数据存储器)地址的对应关系

请注意页的定义: 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*64 点阵的屏分为 8 个“页”, 从第 0“页”到第 7“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:



7.4 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。

7.5 程序举例:

液晶模块与 MCU (以 8051 系列单片机为例) 接口图如下:

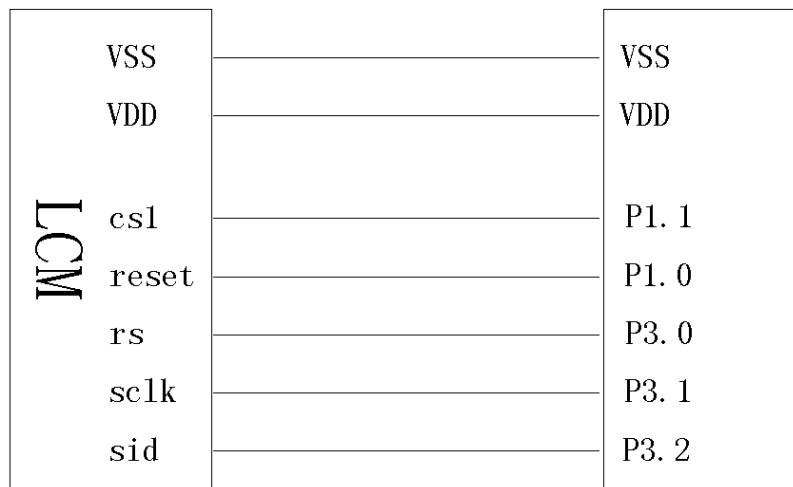


图 8. 串行接口

7.5.1 程序

//JLX19232G-9808-PN-S

```
//串行接口
//驱动 IC 是:ST7525
//单片机型号: STC15W4K56S4
#include <STC15F2K60S2.H>
#include <intrins.h>
#include <Chinese_code.h> //此文件购买后联系销售人员索要
```

```
sbit cs1=P3^4;    /*对应 LCD 的 CS 引脚*/
sbit reset=P3^5; /*对应 LCD 的 RST 引脚*/
sbit rs=P3^3;     /*对应 LCD 的 RS 引脚*/
sbit sclk=P1^0;  //对应 LCD 的 SCK(D0)
sbit sid=P1^1;   //对应 LCD 的 SDA(D1)
```

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

```
#define DataBus P1
```

```
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;
        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;
}

```

```

//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
//====对比度值 0xb3=====//
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(0x2f); //打开内部升压
    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,每格调 0.0143V
transfer_command(0xe9); //1/7 偏压比 (bias)
transfer_command(0xc4); //行列扫描顺序: 从上到下(0xc4 视角 6 点, 0xc2 视角 12 点)
transfer_command(0xf1); //局部显示
transfer_command(0x1f); //显示 1-32 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++;
        }
    }
}

```




```
//=====display a picture of 128*64 dots=====
```

```
void full_display(uchar data_left,uchar data_right)
```

```
{
    int i,j;
    for(i=0;i<4;i++)
    {
        lcd_address(i+1,1);
        for(j=0;j<96;j++)
        {
            transfer_data(data_left);
            transfer_data(data_right);
        }
    }
}
```

```
//显示 32x32 点阵图像、汉字、生僻字或 32x32 点阵的其他图标
```

```
void display_graphic_32x32(uchar page,uchar column,uchar *dp)
```

```
{
    uchar i,j;
    for(j=0;j<4;j++)
    {
        lcd_address(page+j,column);
        for (i=0;i<32;i++)
        {
            transfer_data(*dp); //写数据到 LCD,每写完一个 8 位的数据后列地址自动加 1
            dp++;
        }
    }
}
```



```
//显示 16x16 点阵图像、汉字、生僻字或 16x16 点阵的其他图标
```

```
void display_graphic_16x16(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++;
        }
    }
}
```

// 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 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 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 上, y 为页地址, x 为
                    列地址, 最后为数据
                }
            }
            i++;
            column+=8;
        }
        else
            i++;
    }
}

//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
    while(1)
    {
        initial_lcd();
        clear_screen();                //clear all dots
        display_graphic_192x32 bmp4);
        waitkey();
        clear_screen();                //clear all dots
        display_graphic_192x32 bmp1);
        waitkey();
        clear_screen();                //clear all dots
        display_graphic_192x32 bmp2);
        waitkey();
        clear_screen();                //clear all dots
        display_graphic_192x32 bmp3);
        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();
        full_display(0xff,0xff);
        waitkey();
        full_display(0x55,0xaa);
        waitkey();
        full_display(0xaa,0x55);
        waitkey();
        full_display(0xff,0x00);
        waitkey();
    }
}

```



```

        full_display(0x00,0xff);
        waitkey();
    }
}
    
```

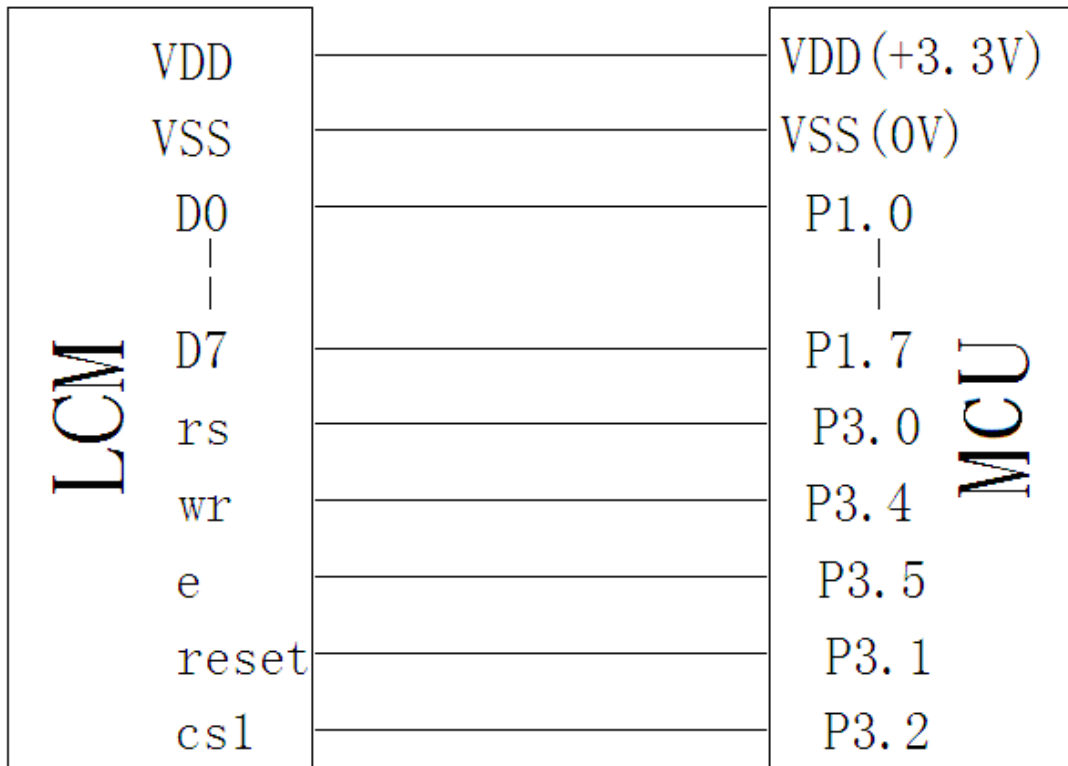


图 9. 并行接口

并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;
}
    
```

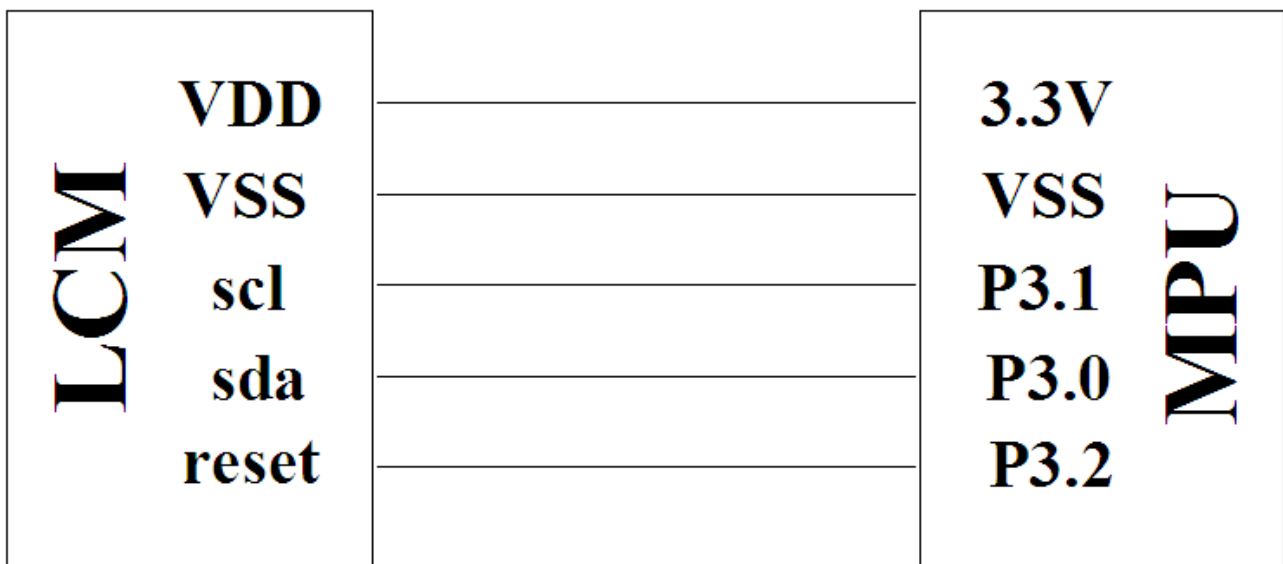


图 10. IIC 接口

IIC 程序与串、并行接口定义、写数据和命令不一样，取模代码是一样的

IIC 程序:

```

// 液晶演示程序 JLX19232G-9809, 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();
}
```

