

JLX240160G-668-BN 使用说明书

(不带字库 IC)

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	电路框图	5
5	背光参数	5
6	指令表及硬件接口、编程案例	5-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 JLX240160G-668 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。

JLX240160G-668-BN 可以显示 240 列*160 行点阵单色或 4 灰度级的图片, 或显示 7 个/行*5 行 32*32 点阵或显示 10 个/行*6 行 24*24 点阵的汉字, 或显示 15 个/行*10 行 16*16 点阵的汉字。

2. JLX240160G-668-BN 图像型点阵液晶模块的特性

2.1 结构牢。

2.2 IC 采用矽创公司 ST7586S, 功能强大, 稳定性好

2.3 功耗低。

2.4 接口简单方便: 可采用 4 线 SPI 串行接口, 或选择并行接口。

2.5 工作温度宽: -20℃ - 70℃;

2.6 储存温度宽: -30℃ - 80℃;

2.7 显示内容:

- 240*160 点阵单色或 4 灰度级图片;
- 或显示 7 个×5 行 32*32 点阵的汉字;
- 或显示 10 个×6 行 24*24 点阵的汉字;
- 或显示 15 个×10 行 16*16 点阵的汉字;
- 或显示 20 个×13 行 12*12 点阵的汉字;
- 或显示其他的 ASCII 码等;

3. 外形尺寸及接口引脚功能:

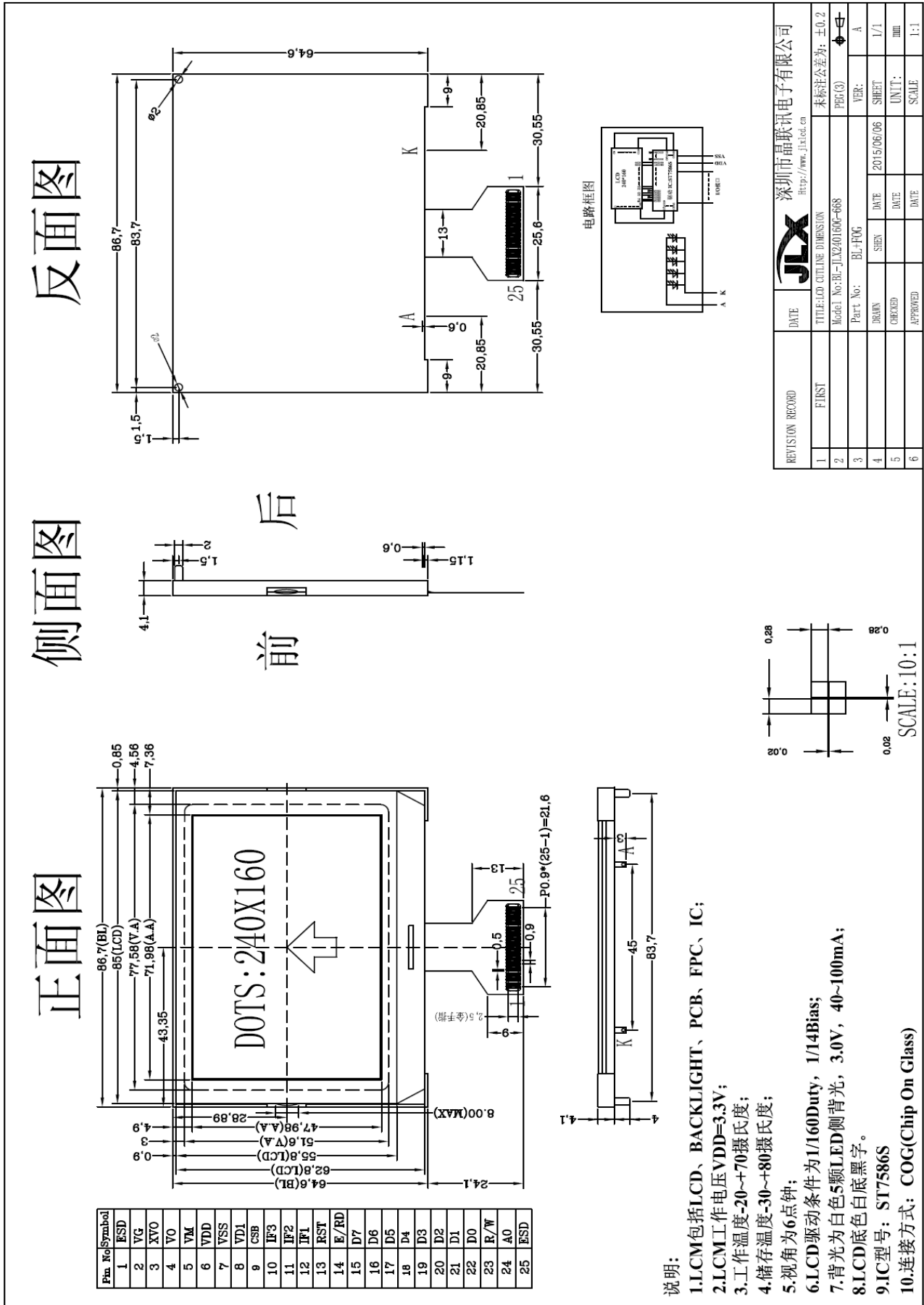
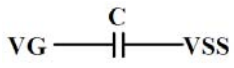
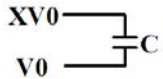
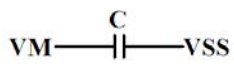


图 1. 液晶模块外形尺寸

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）：

◆当并行时，CON1 功能如下：

引线号	符号	名称	功能																				
1	ESD	空脚	空脚																				
2	VG	升压电容																					
3	XV0	升压电容																					
4	V0	升压电容																					
5	VM	升压电容																					
6	VDD	电路电源	3.3V																				
7	VSS	接地	0V																				
8	VD1	电路电源	3.3V																				
9	CSB	片选	低电平片选																				
10	IF3	IF3	<table border="1"> <thead> <tr> <th>IF3</th> <th>IF2</th> <th>IF1</th> <th>MPU interface type</th> </tr> </thead> <tbody> <tr> <td>H</td> <td>H</td> <td>L</td> <td>80 series 8-bit parallel</td> </tr> <tr> <td>H</td> <td>L</td> <td>L</td> <td>68 series 8-bit parallel</td> </tr> <tr> <td>L</td> <td>H</td> <td>H</td> <td>8-bit serial (4-Line)</td> </tr> <tr> <td>L</td> <td>H</td> <td>L</td> <td>9-bit serial (3-Line)</td> </tr> </tbody> </table>	IF3	IF2	IF1	MPU interface type	H	H	L	80 series 8-bit parallel	H	L	L	68 series 8-bit parallel	L	H	H	8-bit serial (4-Line)	L	H	L	9-bit serial (3-Line)
IF3	IF2	IF1		MPU interface type																			
H	H	L		80 series 8-bit parallel																			
H	L	L		68 series 8-bit parallel																			
L	H	H	8-bit serial (4-Line)																				
L	H	L	9-bit serial (3-Line)																				
11	IF2	IF2																					
12	IF1	IF1																					
13	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																				
14	E(RD)	使能信号 (读)	6800 时序时：E: 使能信号 8080 时序时：读信号																				
15	D7	I/O	数据总线																				
16	D6	I/O	数据总线																				
17	D5	I/O	数据总线																				
18	D4	I/O	数据总线																				
19	D3	I/O	数据总线																				
20	D2	I/O	数据总线																				
21	D1	I/O	数据总线 (串口时为 RS)																				
22	D0	I/O	数据总线 (串口时为 SDA)																				
23	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																				
24	A0 (RS)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器 (串口时为 SCK)																				
25	ESD	空脚	空脚																				

4. 电路框图

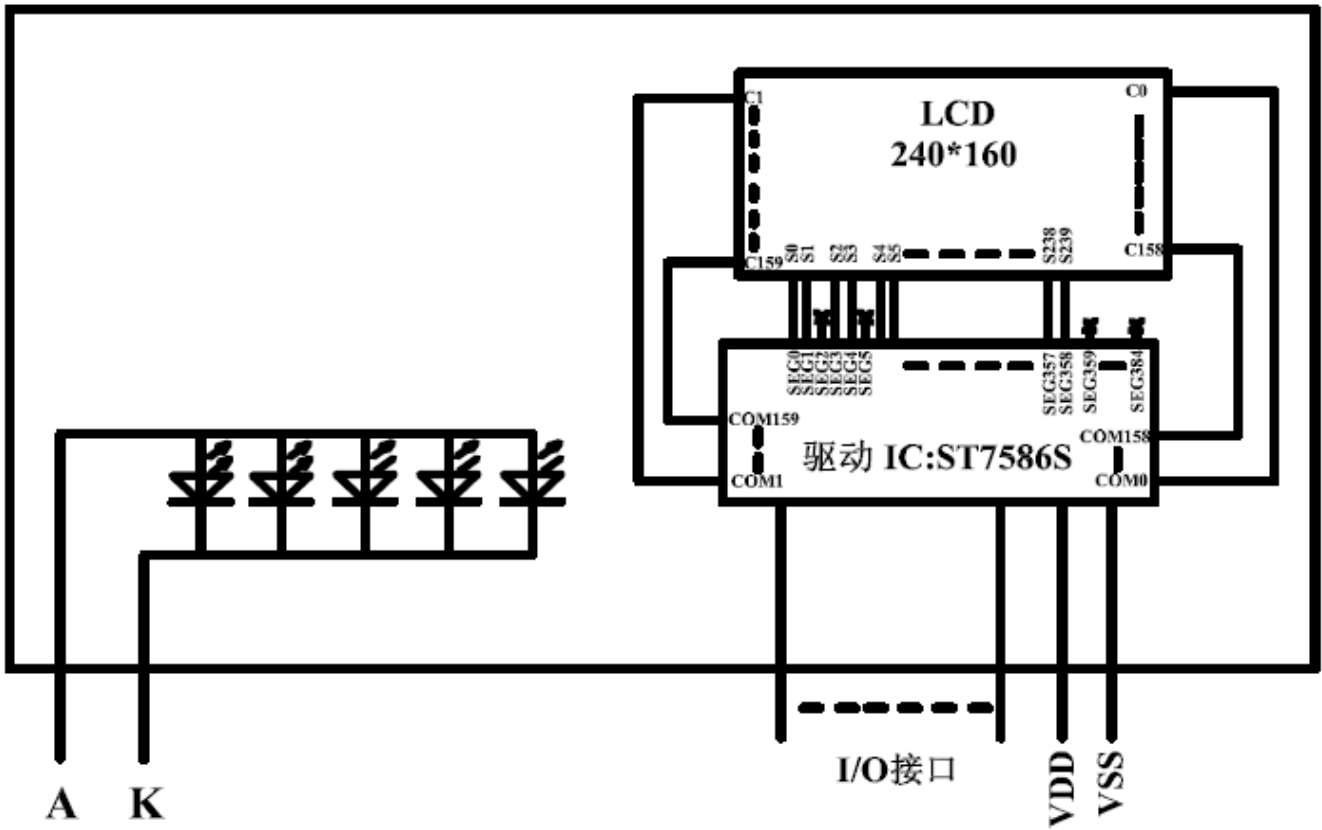


图 2: JLX240160G-668-BN 图像点阵型液晶模块的电路框图

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

背光颜色：白色。

正常工作电流为： $(8\sim 20)\times 5=40\sim 100\text{mA}$ （LED 灯数共 5 颗）；

工作电压：5V 或 3.3V，由你选择的 VDD 电源电压（5V 或 3.3V）决定；

6. 指令表及硬件接口、编程案例

液晶模块中有一个 IC 在 LCD 玻璃上，叫 ST7586S，是液晶屏的驱动 IC。

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。

本说明书只讲述不带字库 IC 时，单片机（MCU）如何通过控制驱动 IC（ST7586S）使液晶屏显示汉字\ASCII 码字符\图片等。

6.1 LCD 驱动 IC（ST7586S）的指令表：

INSTRUCTION TABLE

INSTRUCTION	A0	R/W	COMMAND BYTE								DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
NOP	0	0	0	0	0	0	0	0	0	0	No operation
RESET	0	0	0	0	0	0	0	0	0	1	Software reset
Power Save	0	0	0	0	0	1	0	0	0	SLP	Set power save mode SLP=0: Sleep in mode SLP=1: Sleep out mode
Partial Mode	0	0	0	0	0	1	0	0	1	PTL	Set partial mode PTL=0: Partial mode on PTL=1: Partial mode off
Inverse Display	0	0	0	0	1	0	0	0	0	INV	Set inverse display mode INV=0: Normal display INV=1: Inverse display
All Pixel ON/OFF	0	0	0	0	1	0	0	0	1	AP	Set all pixel on mode AP=0: All pixel off mode AP=1: All pixel on mode
Display ON/OFF	0	0	0	0	1	0	1	0	0	DSP	Set LCD display DSP=0: Display off DSP=1: Display on
Set Column Address	0	0	0	0	1	0	1	0	1	0	Set column address Starting column address: 00h ≤ XS ≤ 7Fh Ending column address: XS ≤ XE ≤ 7Fh
	1	0	XS15	XS14	XS13	XS12	XS11	XS10	XS9	XS8	
	1	0	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	
	1	0	XE15	XE14	XE13	XE12	XE11	XE10	XE9	XE8	
Set Row Address	0	0	0	0	1	0	1	0	1	1	Set row address Starting row address: 00h ≤ YS ≤ 9Fh Ending row address: YS ≤ YE ≤ 9FH
	1	0	YS15	YS14	YS13	YS12	YS11	YS10	YS9	YS8	
	1	0	YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0	
	1	0	YE15	YE14	YE13	YE12	YE11	YE10	YE9	YE8	
Write Display Data	0	0	0	0	1	0	1	1	0	0	Write display data to DDRAM
	1	0	D7	D6	D5	D4	D3	D2	D1	D0	
Read Display Data	0	0	0	0	1	0	1	1	1	0	Read display data from DDRAM
	1	1	D7	D6	D5	D4	D3	D2	D1	D0	
Partial Display Area	0	0	0	0	1	1	0	0	0	0	Set partial area Partial display address start: 00h ≤ PTS ≤ 9Fh Partial display address end: 00h ≤ PTE ≤ 9Fh Display Area: 64 ≤ Duty ≤ 160
	1	0	PTS15	PTS14	PTS13	PTS12	PTS11	PTS10	PTS9	PTS8	
	1	0	PTS7	PTS6	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0	
	1	0	PTE15	PTE14	PTE13	PTE12	PTE11	PTE10	PTE9	PTE8	
Scroll Area	0	0	0	0	1	1	0	0	1	1	Set scroll area Top Area: TA=00h~A0h Scrolling Area: SA=00h~A0h Bottom Area: BA=00h~A0h TA+SA+BA=160
	1	0	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0	
	1	0	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	
	1	0	BA7	BA6	BA5	BA4	BA3	BA2	BA1	BA0	
Display Control	0	0	0	0	1	1	0	1	1	0	Set scan direction of COM and SEG MY=0: COM0→COM159 MY=1: COM159→COOM0 MX=0: SEG0→SEG383 MX=1: SEG383→SEG0
	1	0	MY	MX	0	0	0	0	0	0	
Start Line	0	0	0	0	1	1	0	1	1	1	Set display start line S=00h~9Fh
	1	0	S7	S6	S5	S4	S3	S2	S1	S0	

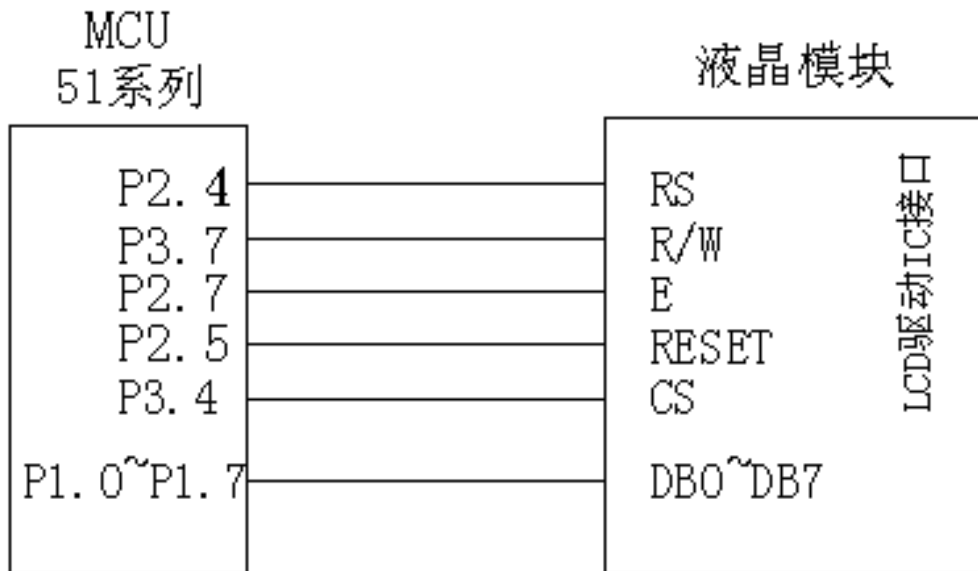
INSTRUCTION	A0	R/W	COMMAND BYTE								DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
Display Mode	0	0	0	0	1	1	1	0	0	M	Set display mode M=0: Gray mode M=1: Monochrome mode
Enable DDRAM Interface	0	0	0	0	1	1	1	0	1	0	Enable DDRAM interface
	1	0	0	0	0	0	0	0	1	0	
Display Duty	0	0	1	0	1	1	0	0	0	0	Set display duty DT=03h~9Fh
	1	0	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	
First Output COM	0	0	1	0	1	1	0	0	0	1	Set first output COM FC=00h~9Fh
	1	0	FC7	FC6	FC5	FC4	FC3	FC2	FC1	FC0	
FOSC Divider	0	0	1	0	1	1	0	0	1	1	Set FOSC dividing ratio
	1	0	0	0	0	0	0	0	FOD1	FOD0	
Partial Display	0	0	1	0	1	1	0	1	0	0	Set partial display mode
	1	0	1	0	1	0	0	0	0	0	
N-Line Inversion	0	0	1	0	1	1	0	1	0	1	Set N-Line inversion
	1	0	M	0	0	NL4	NL3	NL2	NL1	NL0	
Read Modify Write	0	0	1	0	1	1	1	0	0	RMW	Read modify write control RMW=0: Enable read modify write RMW=1: Disable read modify write
Set Vop	0	0	1	1	0	0	0	0	0	0	Set Vop
	1	0	Vop7	Vop6	Vop5	Vop4	Vop3	Vop2	Vop1	Vop0	
	1	0	-	-	-	-	-	-	-	Vop8	
Vop Increase	0	0	1	1	0	0	0	0	0	1	Vop increase one step
Vop Decrease	0	0	1	1	0	0	0	0	1	0	Vop decrease one step
BIAS System	0	0	1	1	0	0	0	0	1	1	Set BIAS system
	1	0	-	-	-	-	-	BS2	BS1	BS0	
Booster Level	0	0	1	1	0	0	0	1	0	0	Set booster level
	1	0	-	-	-	-	-	BST2	BST1	BST0	
Vop Offset	0	0	1	1	0	0	0	1	1	1	Set Vop offset
	1	0	0	VOF6	VOF5	VOF4	VOF3	VOF2	VOF1	VOF0	
Analog Control	0	0	1	1	0	1	0	0	0	0	Enable analog circuit
	1	0	0	0	0	1	1	1	0	1	
Auto Read Control	0	0	1	1	0	1	0	1	1	1	Auto read control XARD=0: Enable auto read XARD=1: Disable auto read
	1	0	1	0	0	XARD	1	1	1	1	
OTP WR/RD Control	0	0	1	1	1	0	0	0	0	0	OTP WR/RD control WR/RD=0: Enable OTP read WR/RD=1: Enable OTP write
	1	0	0	0	WR/RD	0	0	0	0	0	
OTP Control Out	0	0	1	1	1	0	0	0	0	1	OTP control out
OTP Write	0	0	1	1	1	0	0	0	1	0	OTP programming procedure
OTP Read	0	0	1	1	1	0	0	0	1	1	OTP up-load procedure
OTP Selection Control	0	0	1	1	1	0	0	1	0	0	OTP selection control Ctrl=0: Disable OTP Ctrl=1: Enable OTP
	1	0	0	Ctrl	0	1	1	0	0	1	
OTP Programming Setting	0	0	1	1	1	0	0	1	0	1	OTP programming setting
	1	0	0	0	0	0	1	1	1	1	

INSTRUCTION	A0	R/W	COMMAND BYTE								DESCRIPTION
			D7	D6	D5	D4	D3	D2	D1	D0	
Frame Rate	0	0	1	1	1	1	0	0	0	1	Frame rate setting in different temperature range
	1	0	-	-	-	FRA4	FRA3	FRA2	FRA1	FRA0	
	1	0	-	-	-	FRB4	FRB3	FRB2	FRB1	FRB0	
	1	0	-	-	-	FRC4	FRC3	FRC2	FRC1	FRC0	
	1	0	-	-	-	FRD4	FRD3	FRD2	FRD1	FRD0	
Temperature Range	0	0	1	1	1	1	0	0	1	0	Temperature range setting
	1	0	-	TA6	TA5	TA4	TA3	TA2	TA1	TA0	
	1	0	-	TB6	TB5	TB4	TB3	TB2	TB1	TB0	
	1	0	-	TC6	TC5	TC4	TC3	TC2	TC1	TC0	
Temperature Gradient Compensation	0	0	1	1	1	1	0	1	0	0	Set temperature gradient compensation coefficient
	1	0	MT13	MT12	MT11	MT10	MT03	MT02	MT01	MT00	
	1	0	MT33	MT32	MT31	MT30	MT23	MT22	MT21	MT20	
	1	0	MT53	MT52	MT51	MT50	MT43	MT42	MT41	MT40	
	1	0	MT73	MT72	MT71	MT70	MT63	MT62	MT61	MT60	
	1	0	MT93	MT92	MT91	MT90	MT83	MT82	MT81	MT80	
	1	0	MTB3	MTB2	MTB1	MTB0	MTA3	MTA2	MTA1	MTA0	
	1	0	MTD3	MTD2	MTD1	MTD0	MTC3	MTC2	MTC1	MTC0	
1	0	MTF3	MTF2	MTF1	MTF0	MTE3	MTE2	MTE1	MTE0		

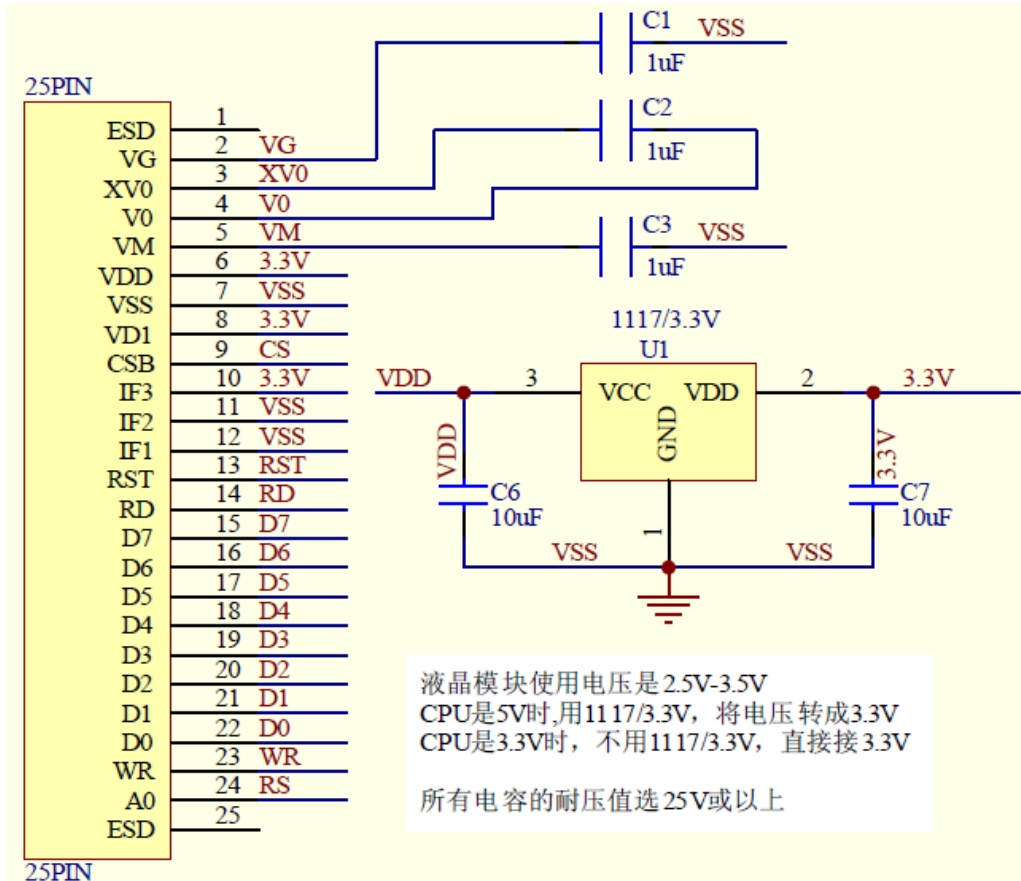
请详细参考 ST7586S 的 IC 资料.

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:



并行接口图



6.3.2 并程序序:

```
/* 液晶模块型号: JLX240160G-668,
   并行接口, 6800 时序,
   驱动 IC 是:ST7586S(or compatible),
   编写: 叶建人
   版权所有: 晶联讯电子: 网址 http://www.jlxlcd.cn;
```

```
*/
#include <reg52.H>
#include <intrins.h>
sbit lcd_cs1 = P3^4;
sbit lcd_reset= P2^5;
sbit lcd_rs = P2^4;
sbit lcd_rw = P3^7;
sbit lcd_E = P2^7;
sbit leda = P3^5; //背光控制端口: 控制 PNP 管的 B 极, 低电平有效。
//PNP 管的 E 极接电源 VDD, C 极通过一个限流电阻接背光阳极 LEDA, 背光的阴极与地线短接。
sbit key = P3^4; //按键
/*另外: DB0~DB7 与 P1.0~P1.7 相连*/
#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long
uchar code da1[]; //32*32 点阵的“大”
uchar code zi1[]; //32*32 点阵的“字”
uchar code dian1[]; //32*32 点阵的“点”
uchar code zhen1[]; //32*32 点阵的“阵”
uchar code zhong2[]; //24*24 点阵的“中”
uchar code zi2[]; //24*24 点阵的“字”
uchar code dian2[]; //24*24 点阵的“点”
uchar code zhen2[]; //24*24 点阵的“阵”
uchar code jing2[]; //24*24 点阵的“晶”
uchar code lian2[]; //24*24 点阵的“联”
uchar code xun2[]; //24*24 点阵的“讯”
```

```
uchar code xiao3[]; //16*16 点阵的“小”
uchar code zi3[]; //16*16 点阵的“字”
uchar code dian3[]; //16*16 点阵的“点”
uchar code zhen3[]; //16*16 点阵的“阵”
uchar code bmp8[];
uchar code ascii_table_16x32[];
uchar code ascii_table_12x24[];
uchar code ascii_table_8x16[];
/*写指令到LCD 模块*/
void transfer_command_lcd(int cmd)
{
    P1=cmd;
    lcd_cs1=0;
    lcd_rs=0;
    lcd_rw=0;
    lcd_E=1;
    lcd_E=0;
    lcd_cs1=1;
}
/*写数据到LCD 模块*/
void transfer_data_lcd(int dat)
{
    P1=dat;
    lcd_cs1=0;
    lcd_rs=1;
    lcd_rw=0;
    lcd_E=1;
    lcd_E=0;
    lcd_cs1=1;
}
/*延时: 1 毫秒的 i 倍*/
void delay(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<110;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(400);
}
/*LCD 模块初始化*/
void initial_lcd()
{
    lcd_reset=1;
    lcd_reset=0; //硬件复位
    delay(10);
    lcd_reset=1; //硬件复位完成后置高
    delay(10);
    transfer_command_lcd(0x11); //退出睡眠模式
    transfer_command_lcd(0xdf); //设置 VOP
    transfer_data_lcd(0x40); //设置 VOP 的值的低 8 位 (总共 9 位)
    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(0x00); //00: BIAS = 1/14
    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(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(0xc0); // 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
}
/*传送一个字节(8比特)黑白图像的数据*/
/*因为这款型号的 IC:ST7586S 原是应用于彩色 STN 屏的, IC 输出列的 1 个地址是包含 3 个列, 无法分开, 比如 SEG0、SEG1、SEG2*/
/*而且 SEG0、SEG1 是引出来的, 对应右边数起来的 S1、S2, SEG2 没引出来, 所以不关心这一列。依此类推 SEG3、SEG4 是引出来的, 对应右边数起来的 S3、S4, SEG5 没引出来, */
/*送数据时右起第 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, 低 2 位——D1 D0 不会用到。*/
void transfer_mono_data(char mono_data)
{
    char i, gray_data=0;
    for(i=0;i<4;i++)
    {
        if(mono_data&0x80)
        {
            gray_data=0x1c;
            if(mono_data&0x40)
            {
                gray_data+=0xe0;
            }
            else;
        }
        else
        {
            gray_data=0x00;
            if(mono_data&0x40)
            {
                gray_data+=0xe0;
            }
            else;
        }
        transfer_data_lcd(gray_data);
        mono_data=mono_data<<2;
    }
}
/*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, x_total, y_total)
{
    int x_ge, x_shi, y_ge, y_shi, x_end, y_end, x_end_shi, x_end_ge, y_end_shi, y_end_ge;
    x=x+8;
    x_shi=x/256;
    x_ge=x%256;
```

```
y_shi=y/256;
y_ge=y%256;
x_end=x+x_total/2-1;
y_end=y+y_total-1;
x_end_shi=x_end/256;
x_end_ge=x_end%256;
y_end_shi=y_end/256;
y_end_ge=y_end%256;
transfer_command_lcd(0x2A);
transfer_data_lcd(x_shi);
transfer_data_lcd(x_ge);
transfer_data_lcd(x_end_shi);
transfer_data_lcd(x_end_ge);
transfer_command_lcd(0x2B);
transfer_data_lcd(y_shi);
transfer_data_lcd(y_ge);
transfer_data_lcd(y_end_shi);
transfer_data_lcd(y_end_ge);
}
/*清屏*/
void clear_screen()
{
    int i, j;
    lcd_address(0, 0, 240, 160);
    transfer_command_lcd(0x2c);
    for(i=0; i<160; i++)
        for(j=0; j<120; j++)
            transfer_data_lcd(0x00);
}
/*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, char *dp)
{
    int i, j;
    lcd_address(x, y, 8, 16);
    transfer_command_lcd(0x2c);
    for(i=0; i<16; i++)
    {
        for(j=0; j<1; j++)
        {
            transfer_mono_data(*dp);
            dp++;
        }
    }
}
/*显示 16*16 点阵汉字或等同于 16*16 点阵的图像*/
void disp_16x16(int x, int y, char *dp)
{
    int i, j;
    lcd_address(x, y, 16, 16);
    transfer_command_lcd(0x2c);
    for(i=0; i<16; i++)
    {
        for(j=0; j<2; j++)
        {
            transfer_mono_data(*dp);
            dp++;
        }
    }
}
/*显示 24*24 点阵的汉字或等同于 24*24 点阵的图像*/
```

```
void disp_24x24(int x, int y, uchar *dp)
{
    int i, j;
    lcd_address(x, y, 24, 24);
    transfer_command_lcd(0x2c);
    for(i=0; i<24; i++)
    {
        for(j=0; j<3; j++)
        {
            transfer_mono_data(*dp);
            dp++;
        }
    }
}
/*显示 16*32 点阵 ASCII 码字符或等同于 16*32 点阵的图像*/
void disp_16x32(int x, int y, char *dp)
{
    int i, j;
    lcd_address(x, y, 16, 32);
    transfer_command_lcd(0x2c);
    for(i=0; i<32; i++)
    {
        for(j=0; j<2; j++)
        {
            transfer_mono_data(*dp);
            dp++;
        }
    }
}
/*显示 32*32 点阵的汉字或等同于 32*32 点阵的图像*/
void disp_32x32(int x, int y, uchar *dp)
{
    int i, j;
    lcd_address(x, y, 32, 32);
    transfer_command_lcd(0x2c);
    for(i=0; i<32; i++)
    {
        for(j=0; j<4; j++)
        {
            transfer_mono_data(*dp);
            dp++;
        }
    }
}
/*显示 240*160 点阵的图像*/
void disp_240x160(int x, int y, char *dp)
{
    int i, j;
    lcd_address(x, y, 240, 160);
    transfer_command_lcd(0x2c);
    for(i=0; i<160; i++)
    {
        for(j=0; j<30; j++)
        {
            transfer_mono_data(*dp);
            dp++;
        }
    }
}
/*显示 8x16 的 ASCII 字符串到 LCD 上,, x 为列地址, y 为行地址, 最后为数据*/
```

```
void disp_string_8x16(uchar x, uchar y, uchar *text)
{
    uint i=0, j, k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x, y, 8, 16);
            transfer_command_lcd(0x2c);
            for(k=0;k<16;k++)
            {
                transfer_mono_data(ascii_table_8x16[j*16+k]);
            }
            x+=4;
            i++;
        }
        else
            i++;
    }
}
/*显示 12x24 的 ASCII 字符串到 LCD 上,, x 为列地址, y 为行地址, 最后为数据*/
void disp_string_12x24(uchar x, uchar y, uchar *text)
{
    uint i=0, j, k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x, y, 16, 24);
            transfer_command_lcd(0x2c);
            for(k=0;k<48;k++)
            {
                transfer_mono_data(ascii_table_12x24[j*48+k]);
            }
            x+=6;
            i++;
        }
        else
            i++;
    }
}
/*显示 16x32 的 ASCII 字符串到 LCD 上,, x 为列地址, y 为行地址, 最后为数据*/
void disp_string_16x32(uchar x, uchar y, uchar *text)
{
    uint i=0, j, k;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(x, y, 16, 32);
            transfer_command_lcd(0x2c);
            for(k=0;k<64;k++)
            {
                transfer_mono_data(ascii_table_16x32[j*64+k]);
            }
            x+=8;
            i++;
        }
        else
            i++;
    }
}
```

```

    }
    else
    i++;
}
}
//-----
void main ()
{
    uchar y;
    initial_lcd();           //对液晶模块进行初始化设置
    leda=0;                  //打开 LED 背光，我的主板是通过 PNP 三极管来控制 LED 背光的开关的
    while(1)
    {
        clear_screen();     //清屏
        disp_240x160(0,0,bmp8); //显示一幅 240*160 点阵的黑白图。
        delay(600);
        clear_screen();     //清?
        y=0;
        disp_32x32(0,y,dal); //在 (x, y) 位置开始，显示“大”字,32x32 点阵
        disp_32x32(16,y,zi1); //在 (x, y) 位置开始，显示“字”字
        disp_string_16x32(32,y,"32*32");//在 (x, y) 位置开始，显示 ASCII 字符串
        disp_32x32(16*4+8,y,dian1); //在 (x, y) 位置开始，显示“点”字
        disp_32x32(16*5+8,y,zhen1); //在 (x, y) 位置开始，显示“阵”字
        y=y+32;
        disp_string_16x32(0,32,"16*32dots ASCII");//在 (x, y) 位置开始，显示 ASCII 字符串
        y=y+32;
        disp_24x24(0,y,zhong2); //在 (x, y) 位置开始，显示“中”字,24x24 点阵
        disp_24x24(12,y,zi2); //在 (x, y) 位置开始，显示“字”字
        disp_string_12x24(24,y,"24*24 ,");//在 (x, y) 位置开始，显示 ASCII 字符串
        disp_24x24(54,y,dian2); //在 (x, y) 位置开始，显示“点”字
        disp_24x24(66,y,zhen2); //在 (x, y) 位置开始，显示“阵”字
        disp_24x24(84,y,jing2); //在 (x, y) 位置开始，显示“晶”字
        disp_24x24(96,y,lian2); //在 (x, y) 位置开始，显示“联”字
        disp_24x24(108,y,xun2); //在 (x, y) 位置开始，显示“讯”字
        y=y+24;
        disp_string_12x24(0,y,"12*24dots ASCII-JLX");//在 (x, y) 位置开始，显示 ASCII 字符串
        y=y+24;
        disp_16x16(0,y,xiao3); //在 (x, y) 位置开始，显示“中”字,16x16 点阵
        disp_16x16(8,y,zi3); //在 (x, y) 位置开始，显示“字”字
        disp_string_8x16(16,y,"16*16");//在 (x, y) 位置开始，显示 ASCII 字符串
        disp_16x16(36,y,dian3); //在 (x, y) 位置开始，显示“点”字
        disp_16x16(44,y,zhen3); //在 (x, y) 位置开始，显示“阵”字
        disp_string_8x16(52,y,"8*16 dots ASCII*");//在 (x, y) 位置开始，显示 ASCII 字符串
        y=y+16;
        //ASCII 码字符显示
        disp_string_8x16(0,y,"!#$%&'()*+,-./:;<=>?@AB[^\`ab");//在 (x, y) 位置开始，显示 ASCII 字符串
        y=y+16;
        disp_string_8x16(0,y,"JLX240160,TEL:755-29784961-809");//在 (x, y) 位置开始，显示 ASCII 字符串
        waitkey();
    }
}

uchar code dal[]={
/*- 文字：大 -*/
/*- 宋体 24； 此字体下对应的点阵为：宽 x 高=32x32 -*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x03,0x80,0x00,0x00,0x03,0xC0,0x00,
0x00,0x03,0x80,0x00,0x00,0x03,0x80,0x00,0x00,0x03,0x80,0x00,0x00,0x03,0x80,0x00,
0x00,0x03,0x80,0x00,0x00,0x03,0x80,0x18,0x3F,0xFF,0xFF,0xFC,0x3F,0xFF,0xFF,0xFC,
0x00,0x03,0x80,0x00,0x00,0x03,0x80,0x00,0x00,0x03,0xC0,0x00,0x00,0x03,0xC0,0x00,
0x00,0x03,0x40,0x00,0x00,0x07,0x60,0x00,0x00,0x07,0x60,0x00,0x00,0x06,0x30,0x00,

```

```

0x00, 0x0E, 0x38, 0x00, 0x00, 0x0E, 0x18, 0x00, 0x00, 0x1C, 0x1C, 0x00, 0x00, 0x1C, 0x0E, 0x00,
0x00, 0x38, 0x07, 0x00, 0x00, 0x70, 0x03, 0x80, 0x00, 0xE0, 0x01, 0xE0, 0x01, 0xC0, 0x01, 0xFC,
0x07, 0x80, 0x00, 0x7E, 0x1E, 0x00, 0x00, 0x38, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
uchar code zil[]={
/*- 文字: 字 -*/
/*- 宋体 24; 此字体下对应的点阵为: 宽 x 高=32x32 -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00, 0x03, 0xC0, 0x00,
0x00, 0x01, 0xC0, 0x20, 0x06, 0x01, 0xC0, 0x30, 0x07, 0xFF, 0xFF, 0xF8, 0x06, 0x00, 0x00, 0x78,
0x0E, 0x00, 0x00, 0xE0, 0x1E, 0x00, 0x00, 0xC0, 0x1C, 0x00, 0x02, 0x80, 0x03, 0xFF, 0xFF, 0x00,
0x03, 0xFF, 0xFF, 0x80, 0x00, 0x00, 0x1E, 0x00, 0x00, 0x00, 0x38, 0x00, 0x00, 0x00, 0x70, 0x00,
0x00, 0x01, 0xC0, 0x00, 0x00, 0x01, 0xC0, 0x18, 0x00, 0x01, 0x80, 0x3C, 0x7F, 0xFF, 0xFF, 0xFC,
0x00, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x00,
0x00, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x01, 0x80, 0x00, 0x00, 0x71, 0x80, 0x00,
0x00, 0x3F, 0x80, 0x00, 0x00, 0x0F, 0x80, 0x00, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};
uchar code dian1[]={
/*- 文字: 点 -*/
/*- 宋体 24; 此字体下对应的点阵为: 宽 x 高=32x32 -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x00, 0x00, 0x03, 0xC0, 0x00,
0x00, 0x03, 0x80, 0x00, 0x00, 0x03, 0x80, 0x20, 0x00, 0x03, 0x80, 0x70, 0x00, 0x03, 0xFF, 0xF0,
0x00, 0x03, 0x80, 0x00, 0x00, 0x03, 0x80, 0x00, 0x00, 0x03, 0x80, 0x00, 0x00, 0x03, 0x80, 0x80,
0x01, 0x83, 0x81, 0x80, 0x01, 0xFF, 0xFF, 0xC0, 0x01, 0xC0, 0x01, 0x80, 0x01, 0xC0, 0x01, 0x80,
0x01, 0xC0, 0x01, 0x80, 0x01, 0xC0, 0x01, 0x80, 0x01, 0xC0, 0x01, 0x80, 0x01, 0xC0, 0x01, 0x80,
0x01, 0xFF, 0xFF, 0x80, 0x01, 0xC0, 0x01, 0x80, 0x01, 0xC0, 0x01, 0x80, 0x01, 0x00, 0x00, 0x00,
0x03, 0x10, 0x20, 0xC0, 0x02, 0x18, 0x30, 0x60, 0x06, 0x0C, 0x38, 0x30, 0x0E, 0x0E, 0x1C, 0x38,
0x1E, 0x0E, 0x1C, 0x38, 0x1C, 0x0E, 0x1C, 0x18, 0x18, 0x04, 0x00, 0x18, 0x00, 0x00, 0x00, 0x00,
};
uchar code zhen1[]={
/*- 文字: 阵 -*/
/*- 宋体 24; 此字体下对应的点阵为: 宽 x 高=32x33 -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x00, 0x10, 0x10, 0x38, 0x00,
0x1F, 0xF8, 0x70, 0x00, 0x18, 0x38, 0x70, 0x10, 0x18, 0x70, 0x60, 0x38, 0x18, 0x6F, 0xFF, 0xFC,
0x18, 0x60, 0xE0, 0x00, 0x18, 0xC0, 0xC0, 0x00, 0x18, 0xC0, 0xCC, 0x00, 0x19, 0x81, 0xCE, 0x00,
0x19, 0x81, 0x8C, 0x00, 0x18, 0xC3, 0x8C, 0x00, 0x18, 0x63, 0x8C, 0x00, 0x18, 0x73, 0x0C, 0x30,
0x18, 0x37, 0xFF, 0xF8, 0x18, 0x3A, 0x0C, 0x00, 0x18, 0x38, 0x0C, 0x00, 0x18, 0x38, 0x0C, 0x00,
0x18, 0x38, 0x0C, 0x08, 0x1B, 0xF0, 0x0C, 0x1C, 0x19, 0xEF, 0xFF, 0xFE, 0x18, 0xC4, 0x0C, 0x00,
0x18, 0x00, 0x0C, 0x00, 0x18, 0x00, 0x0C, 0x00, 0x18, 0x00, 0x0C, 0x00, 0x18, 0x00, 0x0C, 0x00,
0x18, 0x00, 0x0C, 0x00, 0x18, 0x00, 0x0C, 0x00, 0x18, 0x00, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00,
};
uchar code zhong2[]={
/*- 文字: 中 -*/
/*- 宋体 18; 此字体下对应的点阵为: 宽 x 高=24x24 -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00,
0x10, 0x00, 0x10, 0x10, 0x18, 0x1F, 0xFF, 0xF8, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10,
0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x1F, 0xFF, 0xF0, 0x10, 0x10, 0x10, 0x00, 0x10, 0x10,
0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00,
0x10, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00, 0x00,
};
uchar code zi2[]={
/*- 文字: 字 -*/
/*- 宋体 18; 此字体下对应的点阵为: 宽 x 高=24x24 -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x08, 0x00, 0x08, 0x18, 0x0F,
0xF7, 0xF8, 0x18, 0x00, 0x10, 0x30, 0x00, 0x20, 0x00, 0x00, 0xC0, 0x07, 0xFF, 0xE0, 0x00, 0x03,
0x00, 0x00, 0x04, 0x00, 0x00, 0x18, 0x00, 0x00, 0x18, 0x0C, 0x7F, 0xFF, 0xFE, 0x00, 0x18, 0x00,
0x00, 0x18, 0x00, 0x00, 0x18, 0x00, 0x00, 0x18, 0x00, 0x00, 0x18, 0x00, 0x00, 0x18, 0x00, 0x00,
0xF8, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x00,
};
uchar code dian2[]={
/*- 文字: 点 -*/

```



```
/*— 宋体 18; 此字体下对应的点阵为: 宽 x 高=24x24  —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x18, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00,
0x1F, 0xF8, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x00, 0x10, 0x00, 0x02, 0x10, 0x60, 0x03, 0xEF,
0xE0, 0x02, 0x00, 0x60, 0x02, 0x00, 0x60, 0x02, 0x00, 0x60, 0x02, 0x00, 0x60, 0x03, 0xFF, 0xE0,
0x02, 0x00, 0x60, 0x00, 0x00, 0x00, 0x00, 0x84, 0x20, 0x08, 0x42, 0x18, 0x08, 0x63, 0x08, 0x18,
0x21, 0x0C, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00,
};
uchar code zhen2[]={
/*— 文字: 阵  —*/
/*— 宋体 18; 此字体下对应的点阵为: 宽 x 高=24x24  —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x00, 0x1F, 0x86, 0x00, 0x11, 0x84, 0x08, 0x11,
0x7F, 0xFC, 0x11, 0x0C, 0x00, 0x12, 0x08, 0x00, 0x12, 0x09, 0x80, 0x14, 0x19, 0x00, 0x12, 0x11,
0x00, 0x11, 0x11, 0x08, 0x10, 0xBF, 0xF0, 0x10, 0x81, 0x00, 0x10, 0x81, 0x00, 0x11, 0x81, 0x00,
0x1F, 0x81, 0x0C, 0x12, 0x7F, 0xF0, 0x10, 0x01, 0x80, 0x10, 0x01, 0x80, 0x10, 0x01, 0x80, 0x10,
0x01, 0x80, 0x10, 0x01, 0x80, 0x00, 0x00, 0x00,
};

uchar code jing2[]={
/*— 文字: 晶  —*/
/*— 宋体 18; 此字体下对应的点阵为: 宽 x 高=24x24  —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x00, 0x40, 0x01, 0xFF, 0xC0, 0x01, 0x00, 0x40, 0x01,
0x00, 0x40, 0x01, 0xFF, 0xC0, 0x01, 0x00, 0x40, 0x01, 0x00, 0x40, 0x01, 0xFF, 0xC0, 0x01, 0x00,
0x40, 0x00, 0x00, 0x00, 0x10, 0x64, 0x08, 0x1F, 0xE7, 0xFC, 0x10, 0x46, 0x08, 0x10, 0x46, 0x08,
0x10, 0x46, 0x08, 0x1F, 0xC7, 0xF8, 0x10, 0x46, 0x08, 0x10, 0x46, 0x08, 0x10, 0x46, 0x08, 0x1F,
0xC7, 0xF8, 0x10, 0x46, 0x08, 0x00, 0x00, 0x00,
};

uchar code lian2[]={
/*— 文字: 联  —*/
/*— 宋体 18; 此字体下对应的点阵为: 宽 x 高=24x24  —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x30, 0x00, 0x64, 0x30, 0x3F, 0x86, 0x20, 0x10,
0x86, 0x40, 0x10, 0x80, 0x48, 0x10, 0x80, 0x9C, 0x10, 0x9F, 0x60, 0x1F, 0x81, 0x00, 0x10, 0x81,
0x00, 0x10, 0x81, 0x04, 0x10, 0xFF, 0xFE, 0x1F, 0x81, 0x80, 0x10, 0x81, 0x80, 0x10, 0x81, 0x40,
0x10, 0xA1, 0x40, 0x13, 0xC2, 0x40, 0x3C, 0x82, 0x20, 0x20, 0x84, 0x30, 0x00, 0x8C, 0x18, 0x00,
0x98, 0x0E, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x00,
};

uchar code xun2[]={
/*— 文字: 讯  —*/
/*— 宋体 18; 此字体下对应的点阵为: 宽 x 高=24x24  —*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x0C, 0x00, 0x30, 0x06, 0x7F, 0xF0, 0x02,
0x08, 0x20, 0x00, 0x0C, 0x20, 0x00, 0x0C, 0x20, 0x00, 0x0C, 0x20, 0x02, 0x0C, 0x20, 0x3E, 0x0C,
0x20, 0x06, 0x0D, 0xA0, 0x06, 0x7E, 0x60, 0x06, 0x0C, 0x20, 0x06, 0x0C, 0x30, 0x06, 0x0C, 0x30,
0x06, 0x4C, 0x12, 0x06, 0x8C, 0x14, 0x07, 0x8C, 0x14, 0x07, 0x0C, 0x1C, 0x06, 0x0C, 0x0C, 0x02,
0x0C, 0x06, 0x00, 0x00, 0x02, 0x00, 0x00, 0x00,
};

uchar code xiao3[]={
/*— 文字: 小  —*/
/*— 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16  —*/
0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x09, 0x40, 0x0D, 0x20, 0x19, 0x10, 0x11, 0x18,
0x21, 0x0C, 0x41, 0x06, 0x81, 0x04, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x05, 0x00, 0x02, 0x00,
};

uchar code zi3[]={
/*— 文字: 字  —*/
/*— 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16  —*/
0x02, 0x00, 0x01, 0x00, 0x3F, 0xFC, 0x20, 0x04, 0x40, 0x08, 0x1F, 0xE0, 0x00, 0x40, 0x00, 0x80,
0x01, 0x00, 0x7F, 0xFE, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x05, 0x00, 0x02, 0x00,
};
```

```

};

uchar code dian3[]={
/*- 文字： 点  -*/
/*- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16  -*/
0x01, 0x00, 0x01, 0x00, 0x01, 0xF8, 0x01, 0x00, 0x01, 0x10, 0x1F, 0xF8, 0x10, 0x10, 0x10, 0x10,
0x10, 0x10, 0x1F, 0xF0, 0x10, 0x10, 0x01, 0x10, 0x28, 0x88, 0x24, 0x44, 0x44, 0x00, 0x00,
};

uchar code zhen3[]={
/*- 文字： 阵  -*/
/*- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16  -*/
0x00, 0x80, 0xF8, 0x80, 0x88, 0x80, 0x97, 0xFC, 0xA1, 0x00, 0x91, 0x40, 0x8A, 0x40, 0x8F, 0xFC,
0x88, 0x40, 0xA8, 0x40, 0x97, 0xFE, 0x80, 0x40, 0x80, 0x40, 0x80, 0x40, 0x80, 0x40, 0x80, 0x40,
};

uchar code ascii_table_16x32[]={
/*- 文字：      -*/
/*- 宋体 24； 此字体下对应的点阵为：宽 x 高=16x32  -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

/*- 文字： !  -*/
/*- 宋体 24； 此字体下对应的点阵为：宽 x 高=16x32  -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x80, 0x03, 0x80, 0x03, 0x80,
0x03, 0x80, 0x03, 0x80, 0x03, 0x80, 0x03, 0x80, 0x03, 0x80, 0x03, 0x80, 0x03, 0x80, 0x01, 0x00,
0x01, 0x00, 0x01, 0x00, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xC0,
0x03, 0xC0, 0x03, 0xC0, 0x03, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

/*- 文字： "  -*/
/*- 宋体 24； 此字体下对应的点阵为：宽 x 高=16x32  -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x38, 0x0F, 0x78, 0x0F, 0x78, 0x1E, 0xF0, 0x1C, 0xE0,
0x18, 0xC0, 0x31, 0x80, 0x21, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
.
.
//以下省略，完整程序请找客服人员索要。图片转十六进制码方法（取模方法）详见“zimo221.exe 取模方法”，找客服人员索要。
};

uchar code ascii_table_12x24[]={
/*- 文字：      -*/
/*- 宋体 18； 此字体下对应的点阵为：宽 x 高=12x24  -*/
/*- 宽度不是 8 的倍数，现调整为：宽度 x 高度=16x24  -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

/*- 文字： !  -*/
/*- 宋体 18； 此字体下对应的点阵为：宽 x 高=12x24  -*/
/*- 宽度不是 8 的倍数，现调整为：宽度 x 高度=16x24  -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x07, 0x00, 0x07, 0x00, 0x07, 0x00,
0x07, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x02, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x07, 0x00, 0x07, 0x00, 0x07, 0x00, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00,

/*- 文字： "  -*/

```

```

/*- 宋体 18; 此字体下对应的点阵为：宽 x 高=12x24  -*/
/*- 宽度不是 8 的倍数，现调整为：宽度 x 高度=16x24  -*/
0x00, 0x00, 0x00, 0x00, 0x06, 0x60, 0x06, 0x60, 0x0C, 0xC0, 0x19, 0x80, 0x11, 0x00, 0x22, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
.
.
//以下省略，完整程序请找客服人员索要。图片转十六进制码方法（取模方法）详见“zimo221.exe 取模方法”，找客服人员索要。

};

uchar code ascii_table_8x16[]={
/*- 文字：  -*/
/*- 宋体 12; 此字体下对应的点阵为：宽 x 高=8x16  -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,

/*- 文字： !  -*/
/*- 宋体 12; 此字体下对应的点阵为：宽 x 高=8x16  -*/
0x00, 0x00, 0x00, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x00, 0x18, 0x18, 0x00, 0x00,

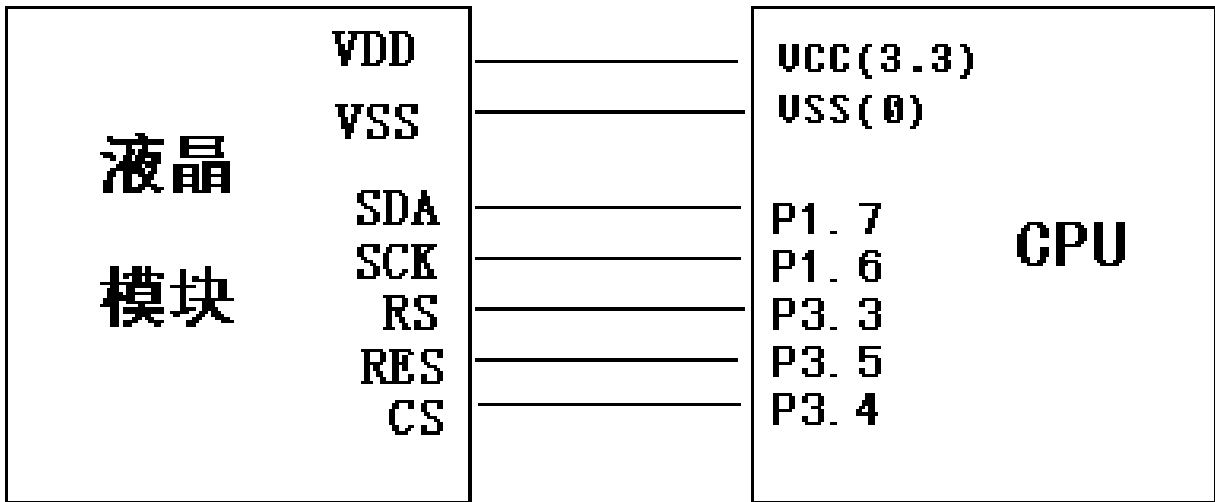
/*- 文字： "  -*/
/*- 宋体 12; 此字体下对应的点阵为：宽 x 高=8x16  -*/
0x00, 0x12, 0x36, 0x24, 0x48, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
.
.
//以下省略，完整程序请找客服人员索要。图片转十六进制码方法（取模方法）详见“zimo221.exe 取模方法”，找客服人员索要。

};

uchar code bmp8[]=
{
/*- 调入了一幅图像：D:\e\新开发部\显示图案收藏\240160-喜羊羊 2. bmp  -*/
/*- 宽度 x 高度=240x160  -*/
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
.
.
//以下省略，完整程序请找客服人员索要。图片转十六进制码方法（取模方法）详见“zimo221.exe 取模方法”，找客服人员索要。
};

```

串行接口程序



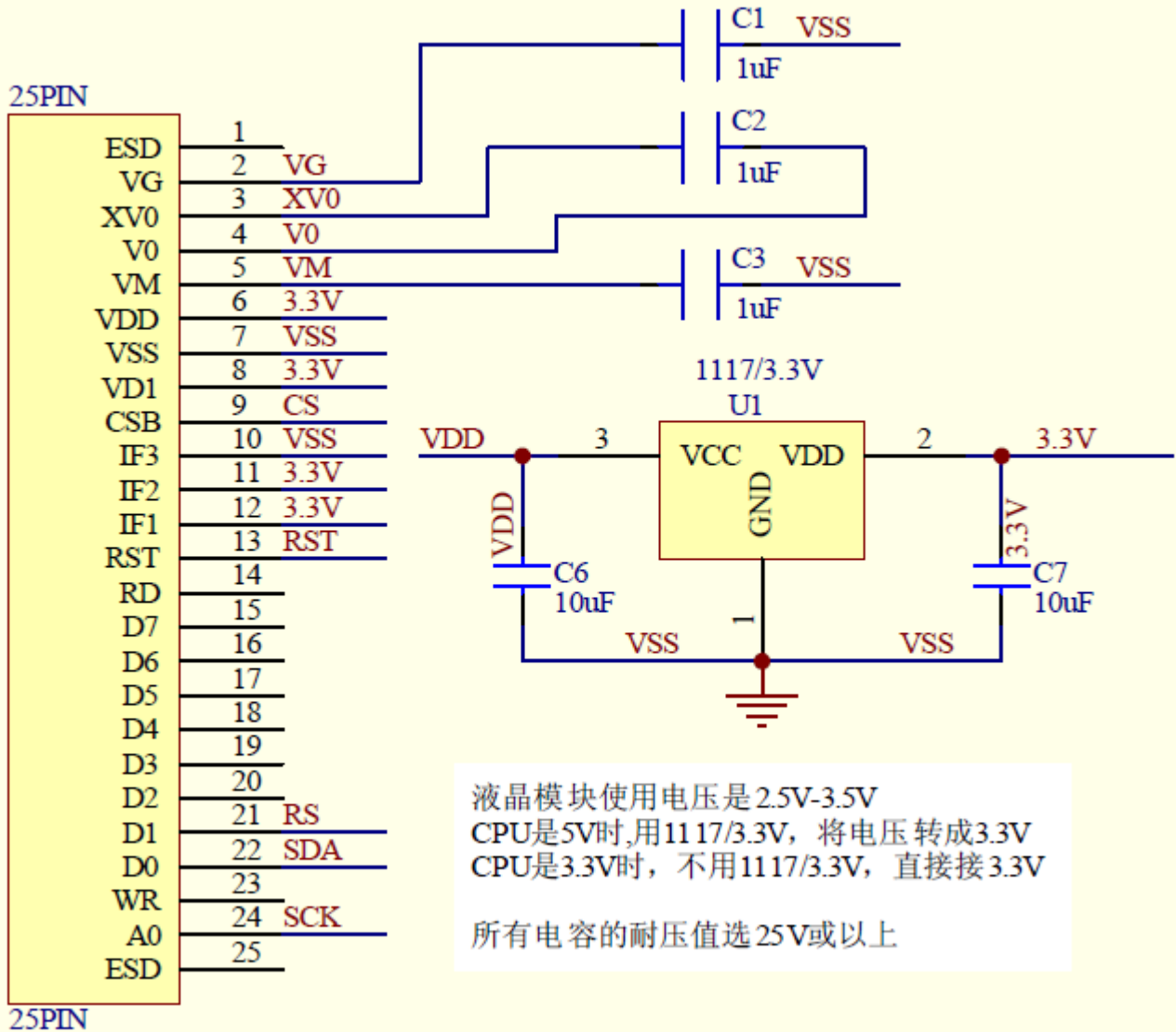


图 9. 串行接口

```

/*      液晶模块型号: JLX240160G-666,
4 线串行接口: 接口有点特别, 请注意:
“SCLK” 接在 PCB 印刷为“RS”那一个脚,
“RS” 接在印刷为“D1”那一个脚,
“SDA” 接在印刷为“D0”那一个脚。
驱动 IC 是:ST7586S
版权所有: 晶联讯电子; 网址 http://www.jlxlcd.cn;
*/

```

```

//=====
sbit lcd_cs1 = P3^4;//CS
sbit lcd_reset= P3^5;//RST
sbit lcd_sclk = P1^6;//串行时钟
sbit lcd_rs = P3^3;//RS
sbit lcd_sid = P1^7;//串行数据
//=====
//写指令到LCD 模块
void transfer_command_lcd(int data1)
{
char i;
lcd_cs1=0;
lcd_rs=0;
for(i=0;i<8;i++)

```

```
{
    lcd_sclk=0;
//    delay_us(10);    //加少量延时
    if(data1&0x80) lcd_sid=1;
    else lcd_sid=0;
    lcd_sclk=1;
//    delay_us(10);    //加少量延时
    data1<<=1;
}
lcd_cs1=1;
}

//写数据到LCD 模块
void transfer_data_lcd(int data1)
{
    char i;
    lcd_cs1=0;
    lcd_rs=1;
    for(i=0;i<8;i++)
    {
        lcd_sclk=0;
//        delay_us(10);    //加少量延时
        if(data1&0x80) lcd_sid=1;
        else lcd_sid=0;
        lcd_sclk=1;
//        delay_us(10);    //加少量延时
        data1<<=1;
    }
    lcd_cs1=1;
}
```

-END-