

JLX12864G-331-BN 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	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 JLX12864G-331 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。

JLX12864G-331 可以显示 128 列*64 行点阵单色图片，或显示 16*16 点阵的汉字 8 个*4 行，或显示 8*16 点阵的英文、数字、符号 16 个*4 行。或显示 5*8 点阵的英文、数字、符号 21 个*8 行。

2. JLX12864G-331 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 采用矽创公司 ST7565R, 功能强大，稳定性好

2.3 功耗低：1~100mW（关掉背光：[0.3mA@3.3V](#), 打开背光不大于 100mW）；

2.4 显示内容：

- 128*64 点阵单色图片；

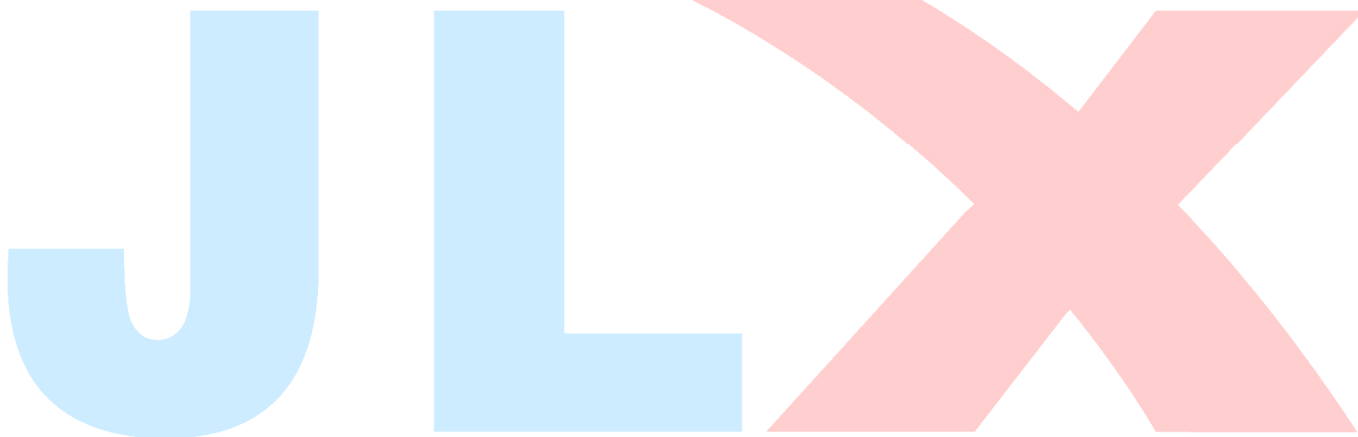
- 可選用 16*16 点阵或其他点阵的图片来自编汉字，按照 16*16 点阵汉字来计算可显示 8 字/行*4 行。按照 12*12 点阵汉字来计算可显示 10 字/行*4 行。

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）。

并口时：可以“读-改-写”；

2.6 接口简单方便：可采用 4 线 SPI 串口，或选择并口（6800 时序和 8080 时序可选）。

2.7 工作温度宽：-20℃ - 70℃；



3. 外形尺寸及接口引脚功能

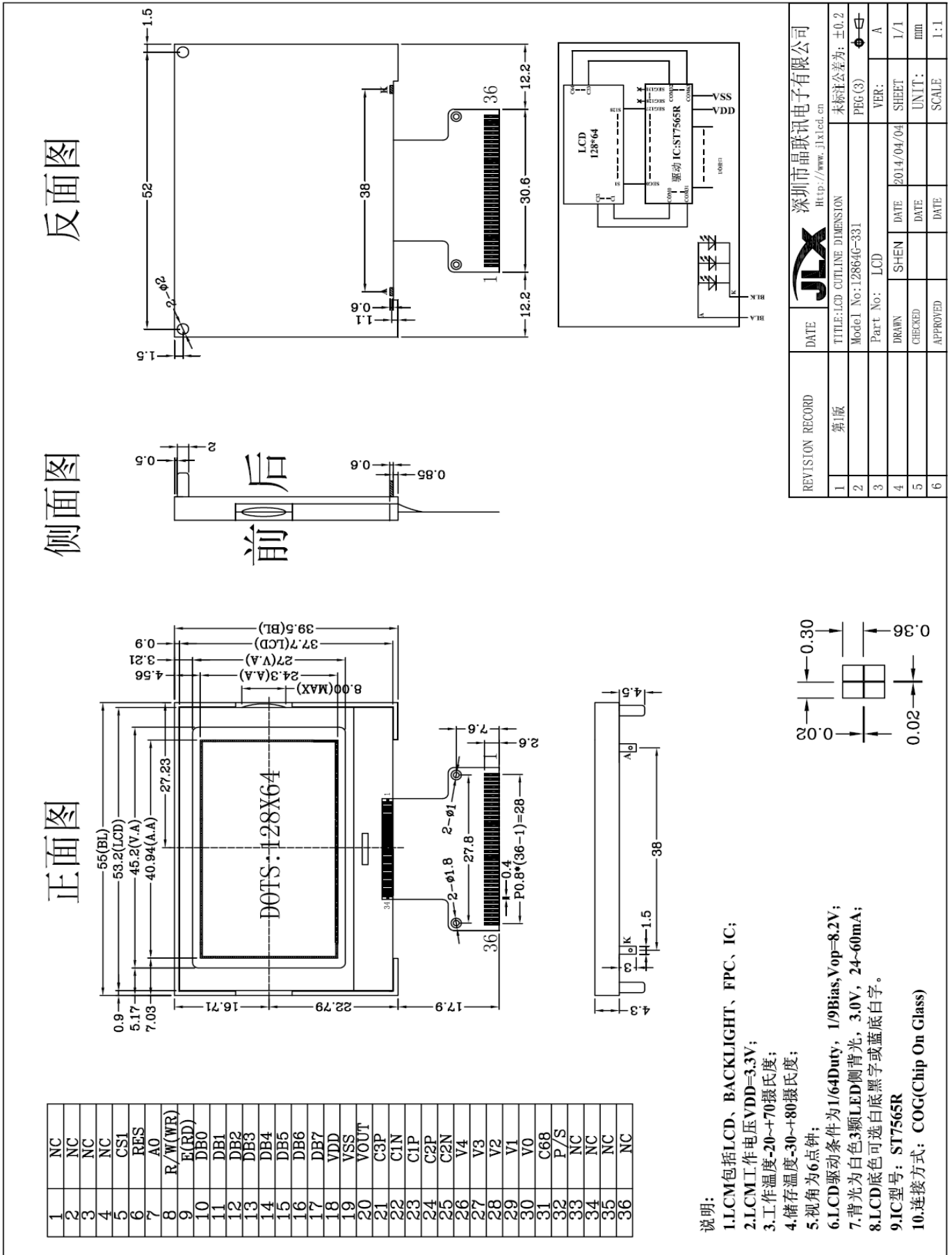


图 1. 外形尺寸

模块的接口引脚功能

引线号	符号	名称	功能
1-4	NC	空脚	空脚
5	CS1	片选	低电平片选
6	RES	复位	低电平复位, 复位完成后, 回到高电平, 液晶模块开始工作
7	A0 (RS)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器
8	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. 串行接口时: 接 VDD 或悬空
9	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. 串行接口时: 接 VDD 或悬空
10-15	DB0-DB5	I/O	数据总线 DB0~DB5 串行接口时: 空脚
16	DB6 (SCLK)	I/O	并行接口时: 数据总线 DB6 串行接口时: 串行时钟 (SCLK)
17	DB7 (SDA)	I/O	并行接口时: 数据总线 DB7 串行接口时: 串行数据 (SDA)
18	VDD	供电电源正极	供电电源正极
19	VSS	接地	0V
20	VOUT	LCD 倍压输出	外接升压电容, 如下图:
21	C3P	倍压电路	
22	C1N	倍压电路	
23	C1P	倍压电路	
24	C2P	倍压电路	
25	C2N	倍压电路	
26	V4	偏置电压	 <p>电压关系: $VOUT > V0 > V1 > V2 > V3 > V4 > VSS$.</p>
27	V3	偏置电压	
28/	V2	偏置电压	
29	V1	偏置电压	
30	V0	偏置电压	
31	C68	选择 6800 或 8080	并行接口时: H: 6800 系统, L: 8080 系统. 串行接口时: 接 VDD
32	P/S	选串并控制接口	接 VDD: 选择并行接口, 接 VSS: 选择串行接口
33-36	NC	空脚	空脚

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 128×64 点阵, 128 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 内部电路框图:

图 2 是 JLX12864G-331 图像点阵型模块的电路框图, 它由驱动 IC ST7565R 及几个电阻电容组成。

电路框图

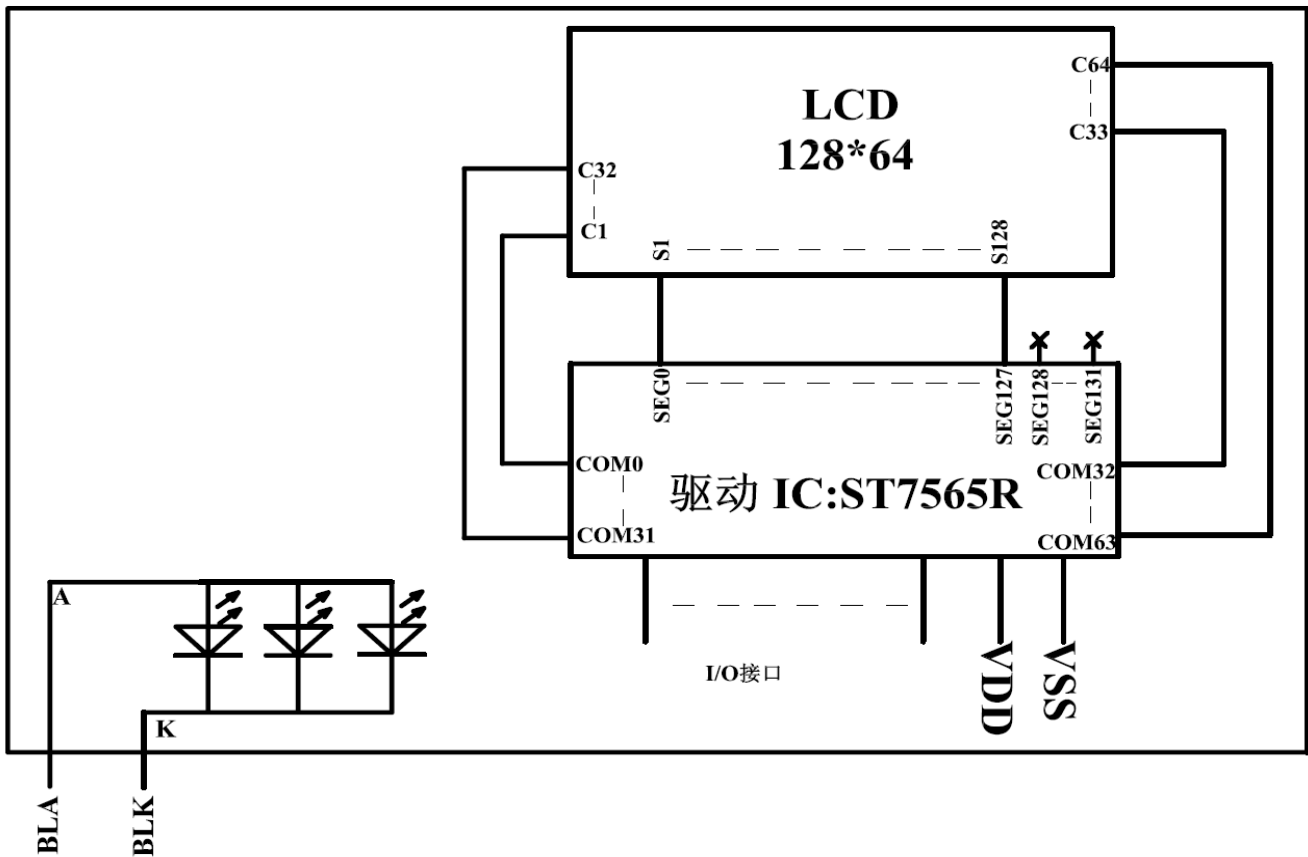


图 2: JLX12864G-331 图像点阵型液晶模块的电路框图

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:

背光板可选择绿色、白色。

正常工作电流为: 24~60mA (LED 灯数共 3 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
LCD 驱动电压	V0、VOUT	-0.3		13.5	V

LCD 驱动电压	V1\V2\V3\V4	-0.3		V0	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	V _{IHC}	-	0.8xVDD	-	VDD	V
输入低电平	V _{ILC}	-	VSS	-	0.2xVDD	V
输出高电平	V _{OHC}	I _{OH} = 0.2mA	0.8xVDD	-	VDD	V
输出低电平	V _{OHC}	I _{OO} = 1.2mA	VSS	-	0.2xVDD	V
模块工作电流	I _{DD}	VDD = 3.0V	-		0.3	mA
背光工作电流	I _{LED}	VLED=3.0V	24	45	60	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 ST7565R (Writing Data from CPU to ST7565R)

The 4-line SPI Interface

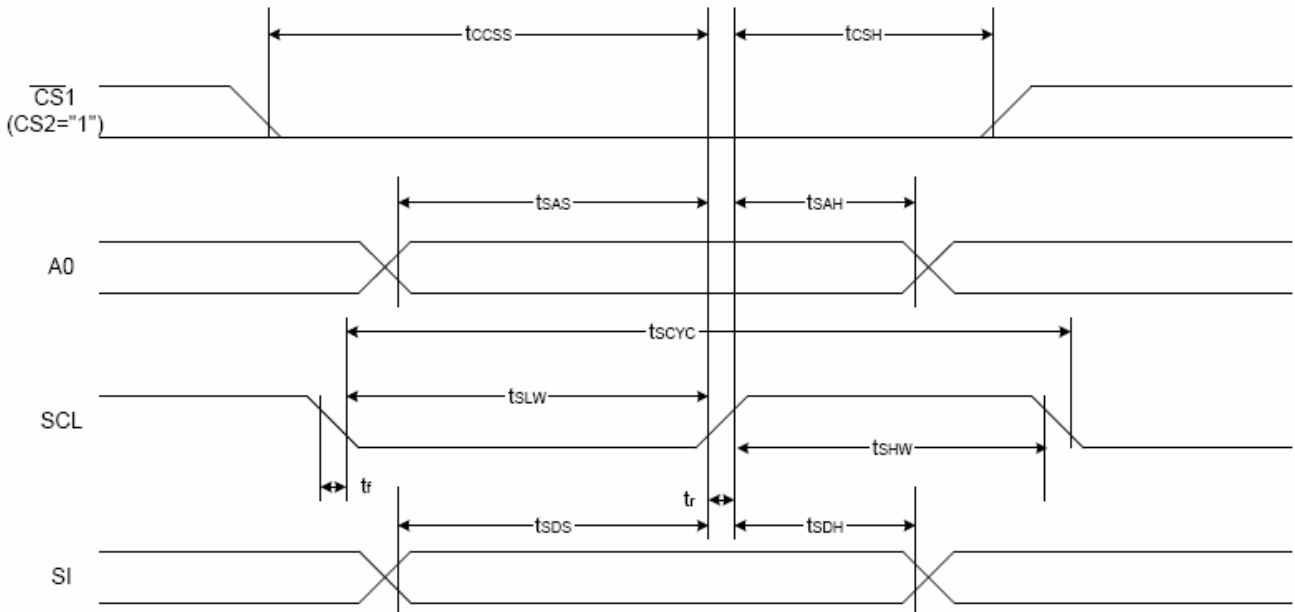


图 4. 从 CPU 写到 ST7565R (Writing Data from CPU to ST7565R)

6.2 串行接口: 时序要求 (AC 参数):

写数据到 ST7565R 的时序要求:

表 4.

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
4线 SPI 串口时钟周期 (4-line SPI Clock Period)	T _{scyc}	引脚: SCK	50	--	25	ns

保持SCK高电平脉宽 (SCK "H" pulse width)	T_{shw}	引脚: SCK	25			ns
保持SCK低电平脉宽 (SCK "L" pulse width)	T_{SLW}	引脚: SCK	25			ns
地址建立时间 (Address setup time)	T_{SAS}	引脚: RS	20	--	--	ns
地址保持时间 (Address hold time)	T_{sah}	引脚: RS	10	--	--	ns
数据建立时间 (Data setup time)	T_{sds}	引脚: SI	20	--	--	ns
数据保持时间 (Data hold time)	T_{SDH}	引脚: SI	10	--	--	ns
片选信号建立时间 (CS-SCL time)	T_{css}	引脚: CS	20			ns
片选信号保持时间 (CS-SCL time)	T_{csh}	引脚: CS	40			ns

VDD = 3.0V ± 5%, Ta = 25°C

6.3 并行接口:

从 CPU 写到 ST7565R (Writing Data from CPU to ST7565R)

System Bus Read/Write Characteristics 1 (For the 8080 Series MPU)

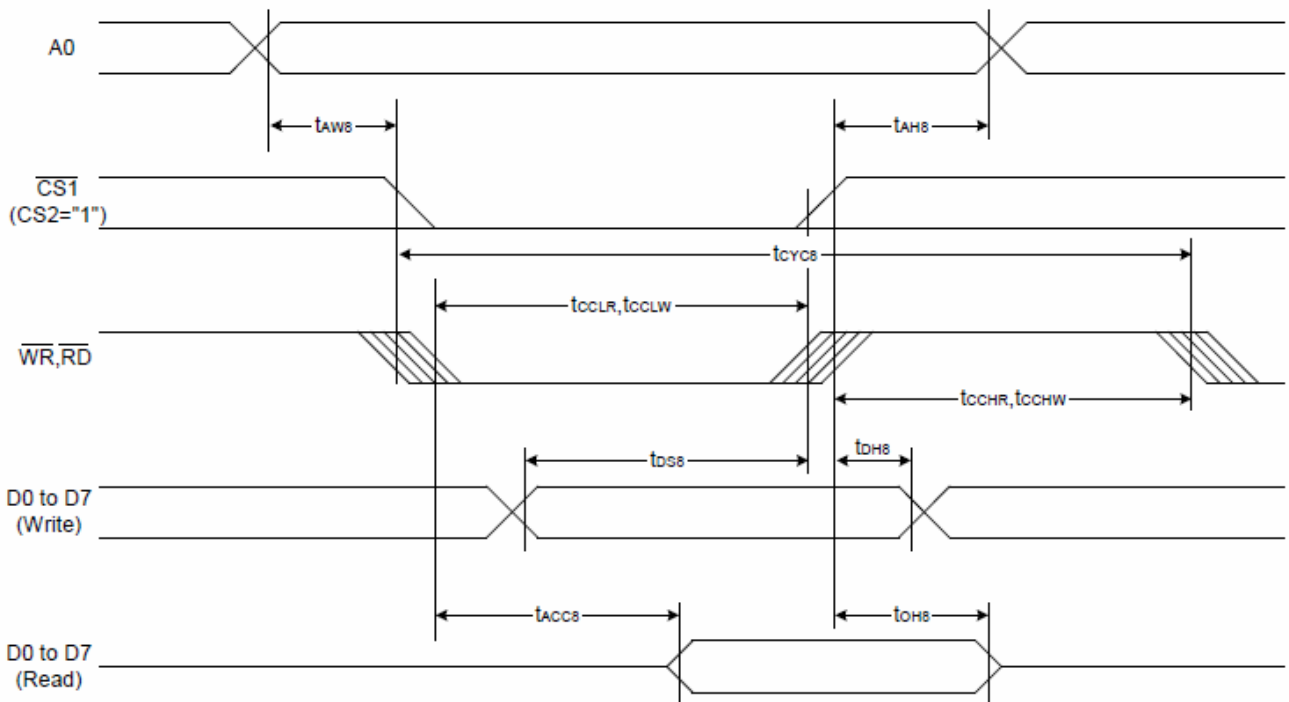


图 5. 从 CPU 写到 ST7565R (Writing Data from CPU to ST7565R)

System Bus Read/Write Characteristics 2 (For the 6800 Series MPU)

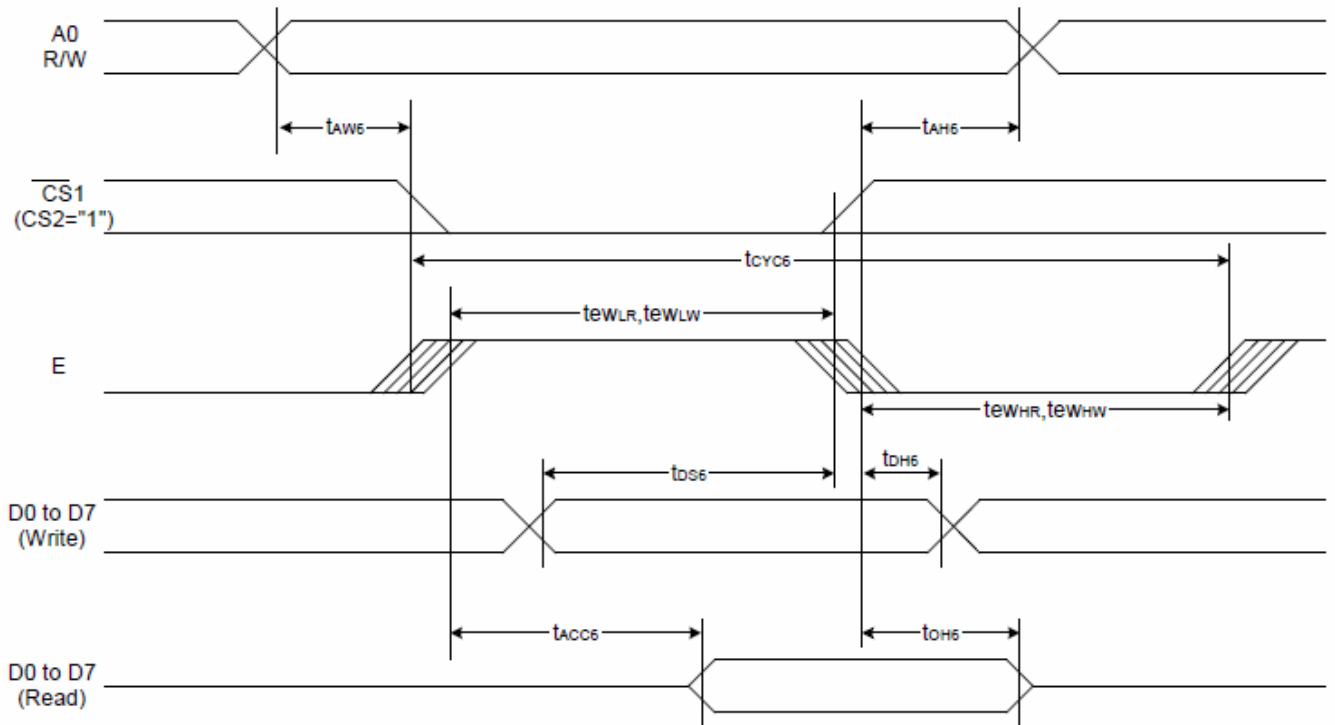


图 6. 从 CPU 写到 ST7565R (Writing Data from CPU to ST7565R)

6.4 并行接口：时序要求 (AC 参数):

写数据到 ST7565R 的时序要求: (8080 系列 MPU)

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
地址保持时间	A0	tAH8	0	--	--	ns
地址建立时间		tAW8	0	--	--	ns
系统循环时间		tCYC8	240	--	--	ns
使能“低”脉冲(写)	WR	tCCLW	80	--	--	ns
使能“高”脉冲(写)		tCCHW	80	--	--	ns
使能“低”脉冲(读)	RD	tCCLR	140	--	--	ns
使能“高”脉冲(读)		tCCHR	80	--	--	ns
写数据建立时间	D0-D7	tDS8	40	--	--	ns
写数据保持时间		tDH8	0	--	--	ns
读时间		tACC8	--	--	70	ns
读输出允许时间		tOH8	5	--	50	ns

写数据到 ST7565R 的时序要求：（6800 系列 MPU）

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
地址保持时间	A0	tAH6	0	--	--	ns
地址建立时间		tAW6	0		--	ns
系统循环时间		tCYC6	240		--	ns
使能“低”脉冲（写）	WR	tEWLW	80	--	--	ns
使能“高”脉冲（写）		tEWHW	80	--	--	ns
使能“低”脉冲（读）	RD	tEWLR	80	--	--	ns
使能“高”脉冲（读）		tEWHR	140	--		ns
写数据建立时间	D0-D7	tDS6	40		--	ns
写数据保持时间		tDH6	0		--	
读时间		tACC6	--		70	
读输出允许时间		tOH6	5		50	ns

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

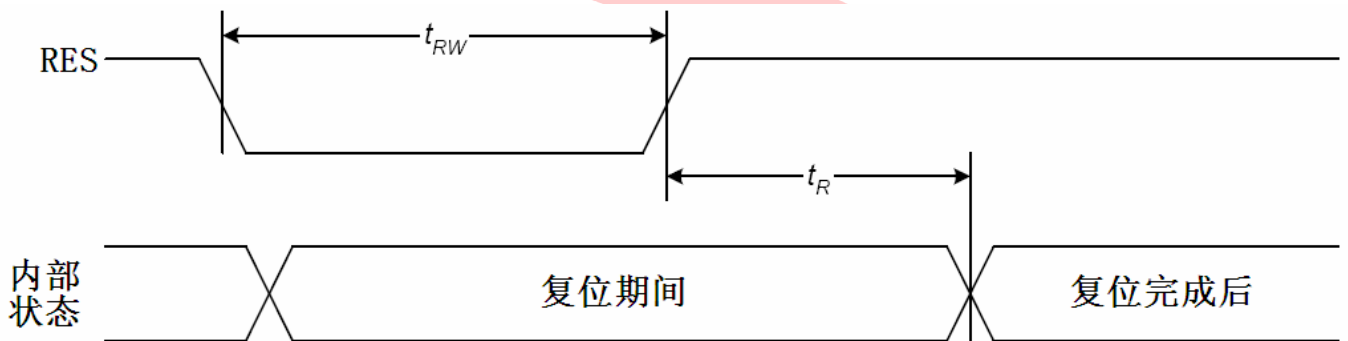


图 7：电源启动后复位的时序

表 6：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		—	—	1.0	us
复位保持低电平的时间	t _{RW}	引脚：RES	1.0	—	—	us

6.6 省电模式

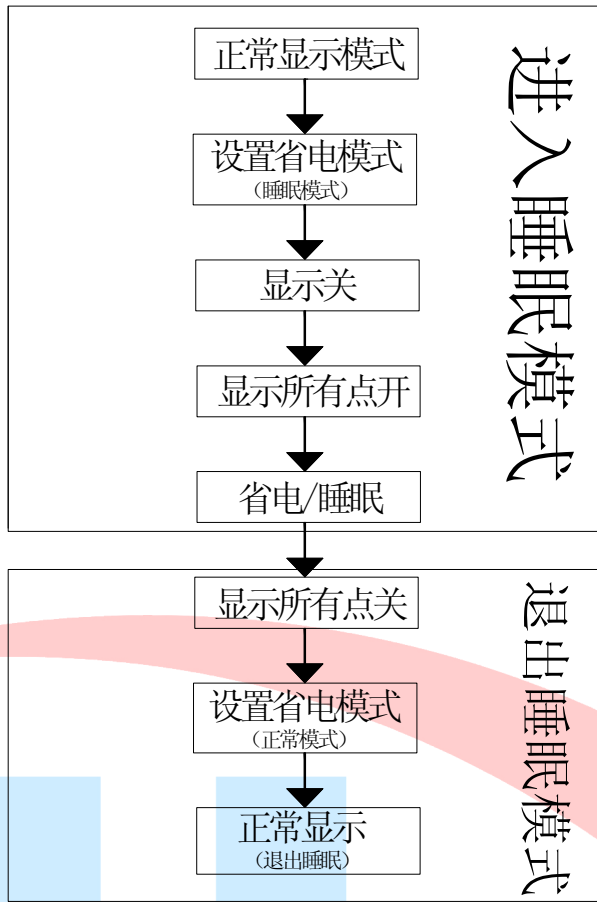


图 8 省电模式流程图

7. 指令功能:

7.1 指令表

指令表

表 8.

指令名称	指令码									说明	
	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
(1)显示开/关 (display on/off)	0	1	0	1	0	1	1	1	0 1	显示开/关: 0XAE :关, 0XAF : 开	
(2)显示初始行设置 (Display start line set)	0	0	1	显示初始行地址, 共 6 位							设置显示存储器的显示初始行,可设置值为 0X40~0X7F ,分别代表第 0~63 行, 针对该液晶屏一般设置为 0x60
(3)页地址设置 (Page address set)	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: 0XB0~0XB8 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 0XB0~0XB7 分别对应第一页~第八页。
(4)	列地址高4位设置	0	0	0	0	1	列地址的高4位			高4位与低4位共同组成列地址,指定128列中的其中一列。比如液晶模块的第100列地址十六进制为 0x64 ,那么此指令由2个字节来表达: 0x16, 0x04
	列地址低4位设置		0	0	0	0	列地址的低4位			
(5)	读状态 (Status read)	0	状态			0	0	0	0	串口时: 读驱动 IC 的当前状态,串口时不能用此指令
(6)	写显示数据到液晶屏 (Display data write)	1	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
(7)	读液晶屏的显示数据 (Display data read)	1	8位显示数据							串口时: 读已经显示到液晶屏上的点阵数据。串口时不能用此指令
(8)	显示列地址增减 (ADC select)		1	0	1	0	0	0	0	显示列地址增减: 0xA0 : 常规: 列地址从左到右, 0xA1 : 反转: 列地址从右到左
(9)	显示正显/反显 (Display normal/reverse)	0	1	0	1	0	0	1	1	显示正显/反显: 0xA6 : 常规: 正显 0xA7 : 反显
(10)	显示全部点阵 (Display all points)	0	1	0	1	0	0	1	0	显示全部点阵: 0xA4 : 常规 0xA5 : 显示全部点阵
(11)	LCD 偏压比设置 (LCD bias set)	0	1	0	1	0	0	0	1	设置偏压比: 0xA2 : BIAS=1/9 (常用) 0xA3 : BIAS=1/7
(12)	读-改-写 (Read-modify-write)	0	1	1	1	0	0	0	0	0XE0 : “读-改-写” 开始。 列地址的增加: 写入时: 列地址+1 读出时: 列地址不加 详情请参考IC资料第43-44页
(13)	退出上述“读-改-写”指令(End)	0	1	1	1	0	1	1	1	0XEE : 上述“读-改-写”指令结束 详情请参考 IC 资料第 43-44 页
(14)	软件复位 (Reset)	0	1	1	1	0	0	0	1	0XE2 : 软件复位。
(15)	行扫描顺序选择 (Common output mode select)		1	1	0	0	0 1	0	0	行扫描顺序选择: 0XC0 : 普通扫描顺序: 从上到下 0XC8 : 反转扫描顺序: 从下到上
(16)	电源控制 (Power control set)		0	0	1	0	1	电压操作模式选择, 共3位		选择内部电压供应操作模式: D2、D1、D0 位分别对应内部升压是否打开(1为打开,0为不打开),电压调整电路是否打开(1为打开,0为不打开),电压跟随器是否打开(1为打开,0为不打开)。 通常是 0x2C,0x2E,0x2F 三条指令按顺序紧接着写,表示依次打开内部升压、电压调整电路、电压跟随器。也可以单写 0x2F ,一次性打开三部分电路。

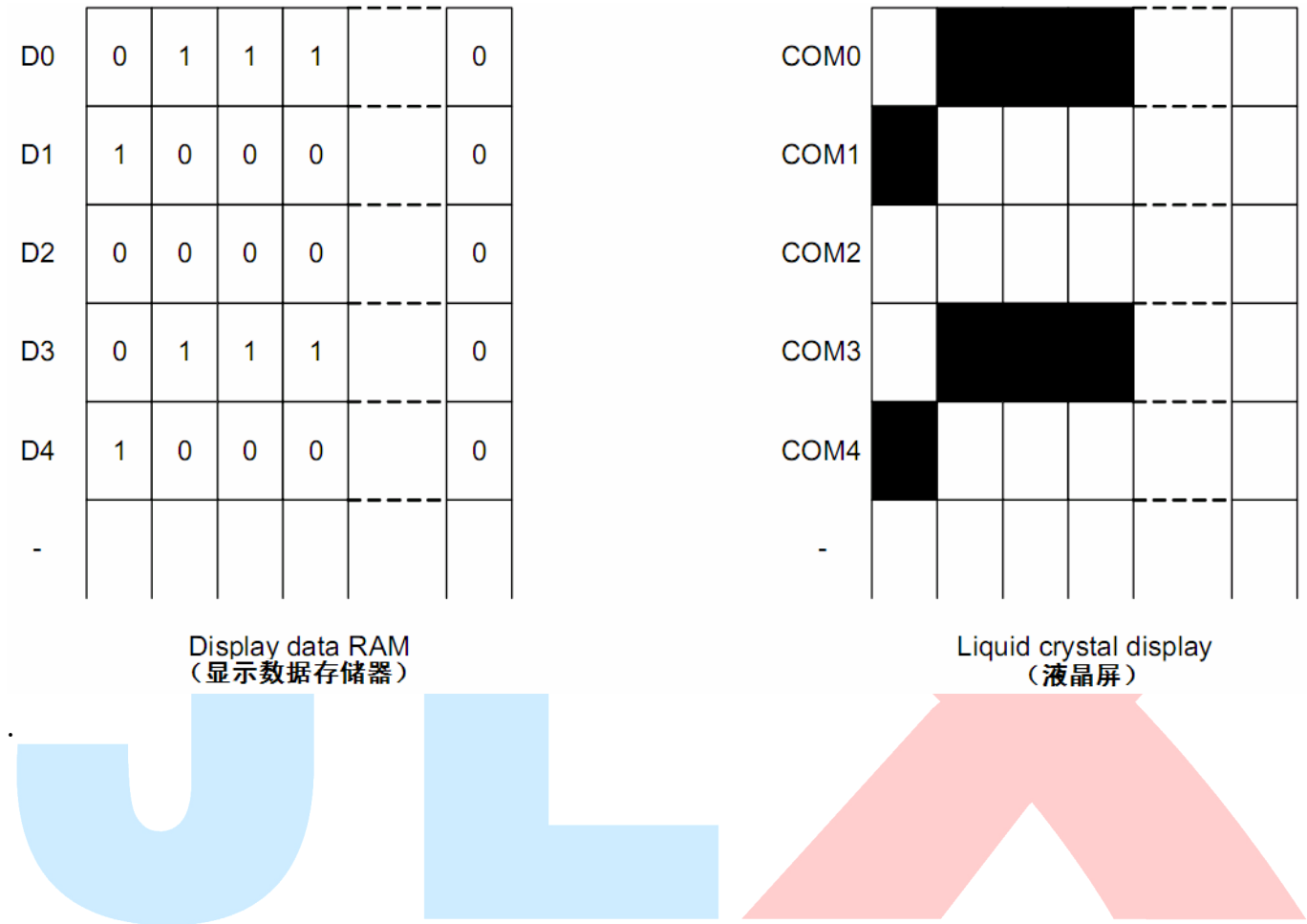
(17) 选择内部电阻比例	0	0	0	1	0	0	内部电压值电阻设置			选择内部电阻比例 (Rb/Ra): 可以理解为 粗调 对比度值。可设置范围为: 0x20~0x27 , 数值越大对比度越浓, 越小越淡
(18)	内部设置液晶电压模式	0	1	0	0	0	0	0	1	设置内部电阻微调, 可以理解为 微调 对比度值, 此两个指令需紧接着使用。上面一条指令 0x81 是不改的, 下面一条指令可设置范围为: 0x00~0x3F , 数值越大对比度越浓, 越小越淡
	设置的电压值		0	0	6 位电压值数据, 0~63 共 64 级					
(19) 静态图标显示: 开/关	0	1	0	1	0	1	1	0	0 1	静态图标的开关设置: 0xAC : 关, 0xAD : 开。 此指令在进入及退出睡眠模式时起作用
(20) 升压倍数选择 (Booster ratio set)	0	1	1	1	1	1	0	0	2 位数设置 升压倍数	选择升压倍数: 00: 2 倍, 3 倍, 4 倍 01: 5 倍 11: 6 倍。本模块外部已设置升压倍数为 4 倍, 不必使用此指令
		0	0	0	0	0	0			
(21) 省电模式 (Power save)										省电模式, 此非一条指令, 是由“(10)显示全部点阵”、(19)静态图标显示: 开/关等指令合成一个“省电功能”。详细看 IC 规格书第 47 页“POWER SAVE”
(22) 空指令 (NOP)	0	1	1	1	0	0	0	1	1	空操作
(23) 测试 (Test)	0	1	1	1	1	*	*	*	*	内部测试用, 千万别用!

温馨提示: 请详细参考 IC 资料”ST7565R_V1.9.PDF”的第 28~36 页。

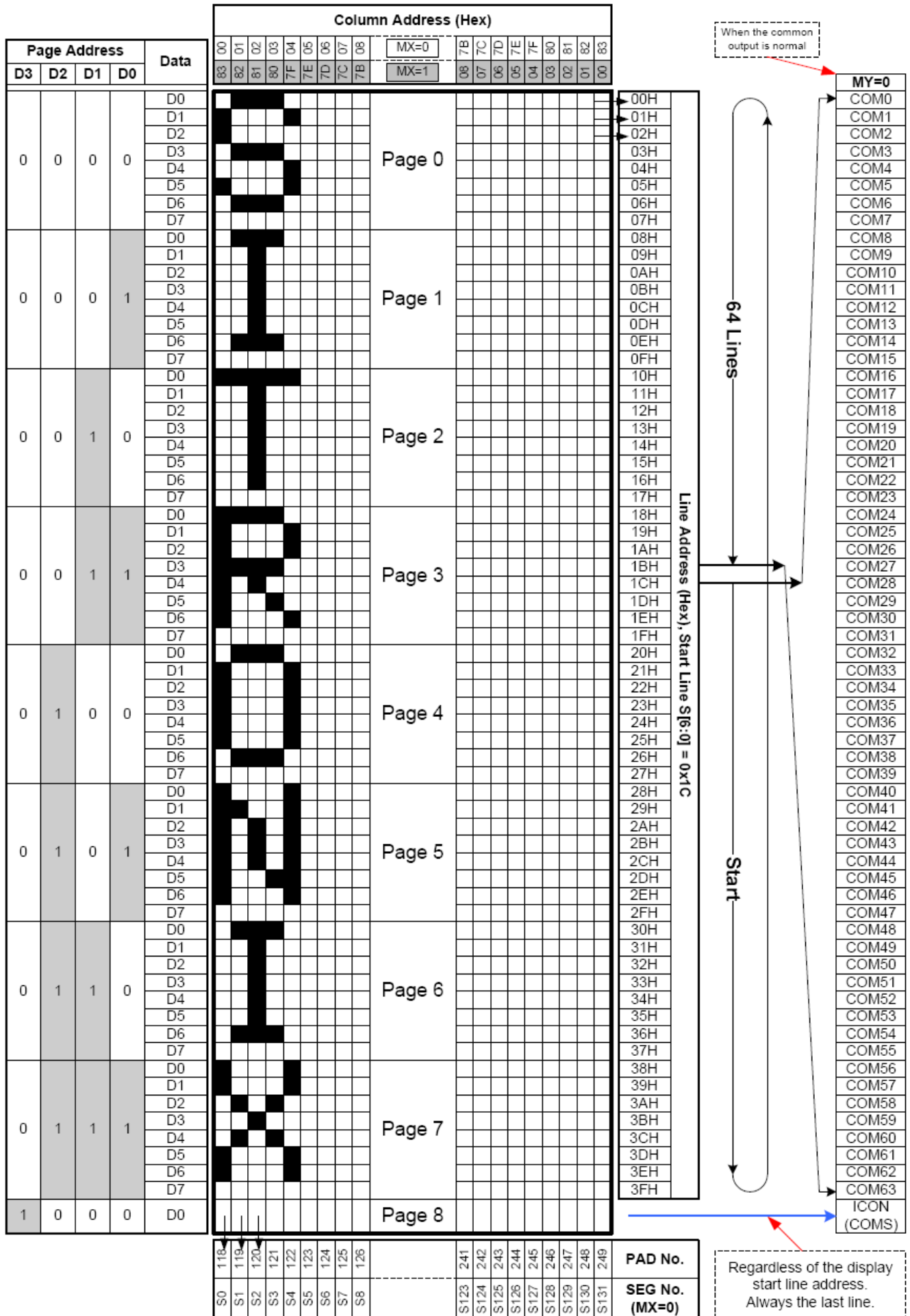
7.3 点阵与 DD RAM 地址的对应关系

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 128*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:



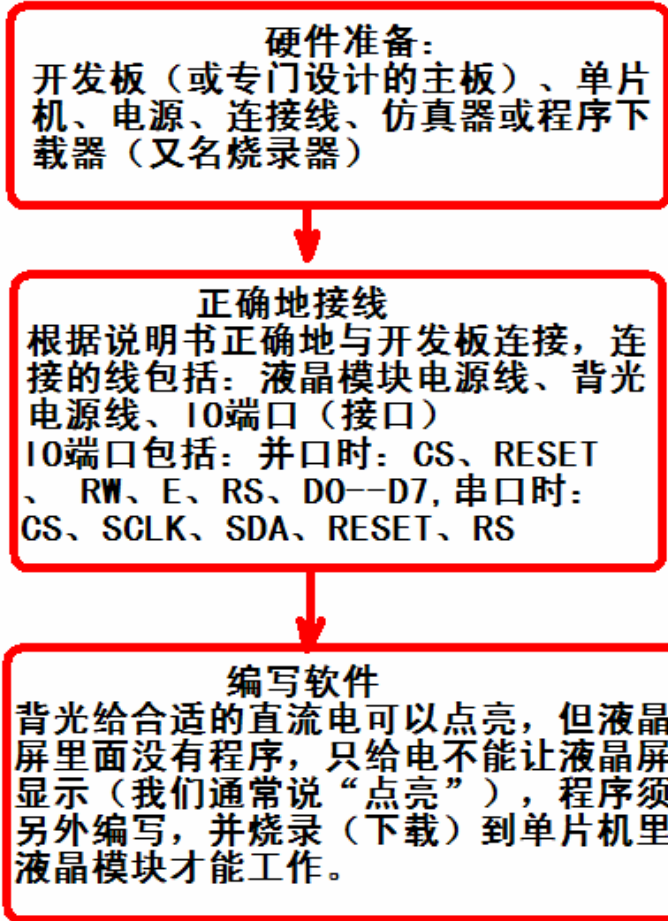
下图摘自 ST7565R IC 资料, 可通过“ST7565R_V1.9.PDF”之第 18、19 页获取最佳效果。



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



7.5 程序举例:

液晶模块与 MPU (以 8051 系列单片机为例) 接口图如下:

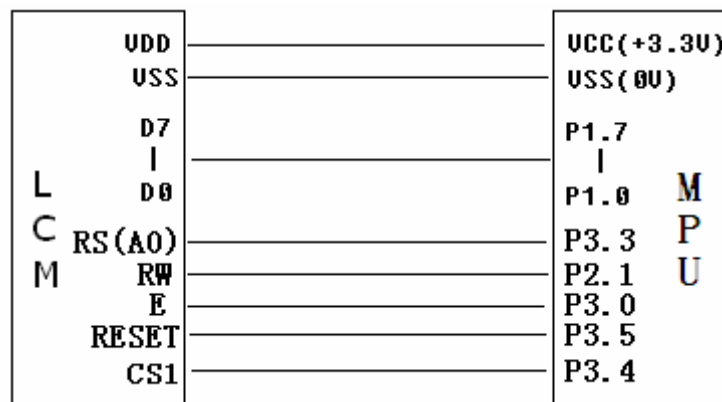
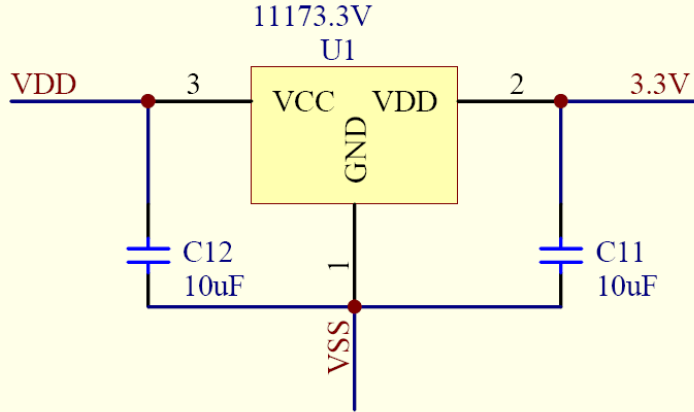


图 8. 并行接口

并行调对比度电路图

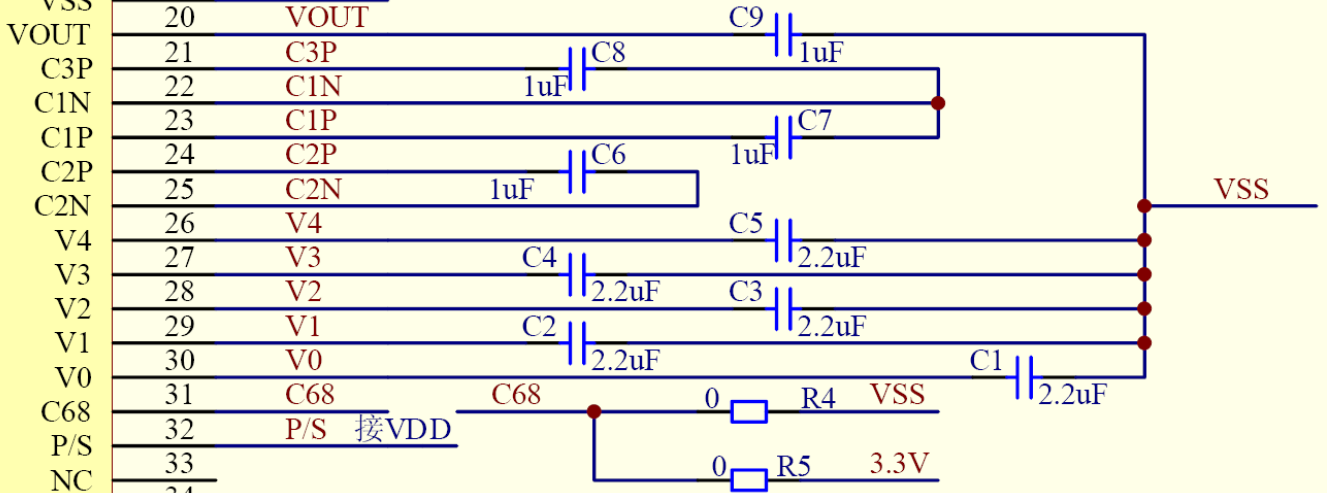
36PIN

NC	1	
NC	2	
NC	3	
NC	4	
CS1	5	CS
RES	6	RES
A0	7	RS
R/W	8	R/W
RD	9	RD
DB0	10	D0
DB1	11	D1
DB2	12	D2
DB3	13	D3
DB4	14	D4
DB5	15	D5
DB6	16	D6
DB7	17	D7
VDD	18	VDD
VSS	19	VSS
VOUT	20	VOUT
C3P	21	C3P
C1N	22	C1N
C1P	23	C1P
C2P	24	C2P
C2N	25	C2N
V4	26	V4
V3	27	V3
V2	28	V2
V1	29	V1
V0	30	V0
C68	31	C68
P/S	32	P/S 接VDD
NC	33	
NC	34	
NC	35	
NC	36	



液晶模块使用电压是2.5V-3.5V
 CPU是5V时, 用1117/3.3V, 将电压转成3.3V
 CPU是3.3V时不用1117/3.3V, 直接接3.3V

所有电容的耐压值选25V或以上

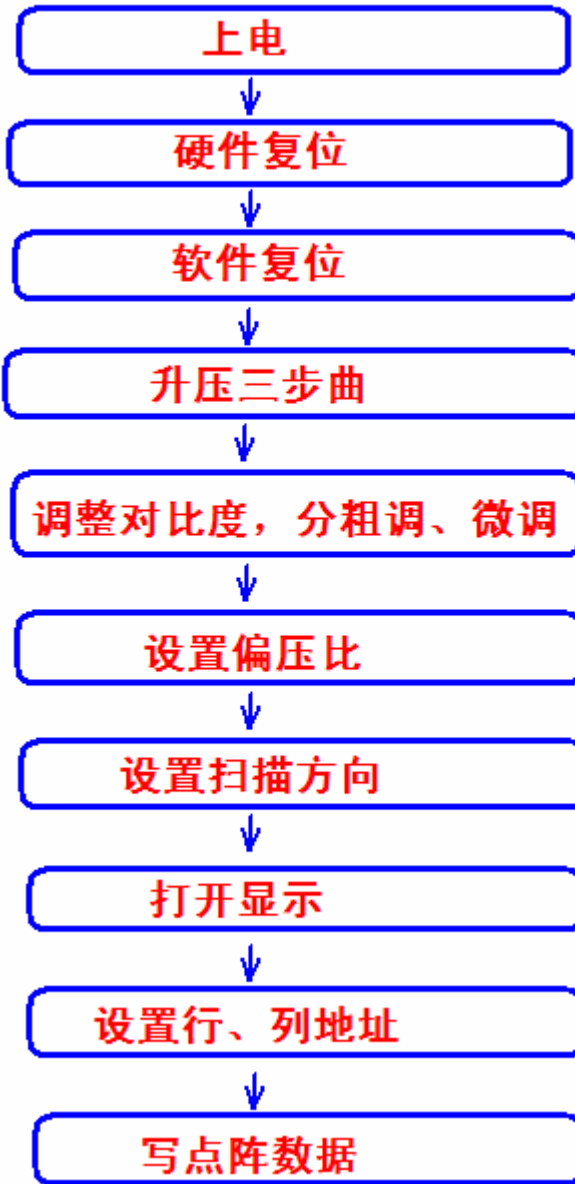


C68:6800时序;R4空, R5接0欧电阻。
 8080时序:R5空, R4接0欧电阻。
 例程:6800

36PIN

7.51、程序

点亮液晶模块的编程步骤



/* 液晶模块型号 JLX12864G-331, 并行接口, 6800 时序, 不带中文字库 IC

驱动 IC 是:ST7565R, 字库 IC:不带中文字库 IC

版权所有: 深圳市晶联讯电子有限公司: 网址 <http://www.jlxlcd.cn>;

*/

```

#include <reg51.h>
#include <intrins.h>
#include <string.h>
#include <math.h>
    
```

```

sbit rw=P2^1; //接口定义:lcd_rw 就是 LCD 的 wr,有些图纸上写“WR”
sbit e=P3^0; //接口定义:lcd_e 就是 LCD 的 rd
sbit rs=P3^3; //接口定义:lcd_rs 就是 LCD 的 rs,有些图纸上写“A0”\“DC”\“DC0”,都是它
sbit cs1=P3^4; //接口定义:lcd_cs1 就是 LCD 的 cs1
sbit reset=P3^5; //接口定义:lcd_reset 就是 LCD 的 reset
sbit key=P2^0; //P2.0 口与 GND 之间接一个按键
    
```

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

char code cheng1[];
char code gong[];
char code zhuang1[];
char code tail[];
char code shi1[];
char code yong1[];
char code hua[];
char code dian[];
char code xian[];

char code ascii_table_8x16[95][16];
char code ascii_table_5x8[95][5];
char code bmp1[];
char code bmp2[];
char code bmp3[];
char code bmp4[];

/*延时*/
void delay(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<110;k++);
}

/*短延时*/
void delay_us(int i)
{
    int j,k;
    for(j=0;j<i;j++)
        for(k=0;k<1;k++);
}

void waitkey()
{
repeat:   if(key==1) goto repeat;
          else delay(2000);
}

//写指令到LCD 模块
void transfer_command(int data1)
{
    cs1=0;
    rs=0;
    rw=0;
    P1=data1;
    e=1;
    delay_us(10);
    e=0;
    cs1=1;
    e=0;
}

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

```

cs1=0;
rs=1;
rw=0;
P1=data1;
e=1;
delay_us(2);
e=0;
cs1=1;
e=0;
}

/*LCD 模块初始化*/
void initial_lcd()
{
    reset=0;      /*低电平复位*/
    delay(200);
    reset=1;      /*复位完毕*/
    delay(50);
    transfer_command(0xe2); /*软复位*/
    delay(5);
    transfer_command(0x2c); /*升压步骤 1*/
    delay(5);
    transfer_command(0x2e); /*升压步骤 2*/
    delay(5);
    transfer_command(0x2f); /*升压步骤 3*/
    delay(5);
    transfer_command(0x24); /*粗调对比度, 可设置范围 0x20~0x27*/
    transfer_command(0x81); /*微调对比度*/
    transfer_command(0x2a); /*0x22, 微调对比度的值, 可设置范围 0x00~0x3f*/
    transfer_command(0xa2); /*1/9 偏压比 (bias) */
    transfer_command(0xc8); /*行扫描顺序: 从上到下*/
    transfer_command(0xa0); /*列扫描顺序: 从左到右*/
    transfer_command(0x40); /*起始行: 第一行开始*/
    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<9;i++)
    {
        lcd_address(1+i,1);
        for(j=0;j<132;j++)
        {
            transfer_data(0x00);
        }
    }
}

```

//==显示测试画面：例如全显示，隔行显示，隔列显示，雪花显示==

```
void test_display(uchar data1,uchar data2)
```

```
{
    int i, j;
    for(j=0;j<8;j++)
    {
        lcd_address(j+1, 0);
        for(i=0;i<128;i++)
        {
            transfer_data(data1);
            transfer_data(data2);
        }
    }
}
```

```
void display_graphic(uchar *dp)
```

```
{
    uchar i, j;
    for(j=0;j<8;j++)
    {
        lcd_address(j+1, 1);
        for(i=0;i<128;i++)
        {
            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<31;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 reverse, uchar *dp)
```

```
{
    uchar i, j;
    for(j=0;j<2;j++)
    {
        lcd_address(page+j, column);
        for (i=0;i<16;i++)
        {
            if(reverse==1)
            {
                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*/
            }
            else
                transfer_data(*dp);
            dp++;
        }
    }
}
```

```

    }
}

/*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++;
        }
    }
}

```

```

//显示一串 8x16 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_8x16(uint page,uint column,uchar reverse,uchar *text)
{
    uint i=0, j, k, n, data1;
    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);
                }
            }
            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++;
}
}
```

****从液晶屏驱动 IC 中读取数据 (1 个字节) ****/

```
uchar read_data ()
{
    uchar ret_data=0;
    P1=0xff;
    rw=1;
    rs=1;
    cs1=0;
    e=0;
    e=1;
    delay_us(1);
    ret_data=P1;
    cs1=1;
    return(ret_data);
}
```

====开始 “读取-修改-写入” 模式====

```
void Start_Read_Modify_Write()
{
    transfer_command(0xe0);
}
```

====结束 “读取-修改-写入” 模式====

```
void End_Read_Modify_Write()
{
    transfer_command(0xee);
}
```

//从液晶屏驱动 IC 中读取数据, 含一次空读和一次正式读取

```
uchar read_data_twice()
{
    uchar ret_data=0;
    P1=0xff;
    cs1=0;
    rs=1;
    rw=1;

    //一次空读-----
    e=0;
    e=1;
    delay_us(1);
```

```

//-----一次正式读-----
e=0;
e=1;
delay_us(1);
//-----
ret_data=P1;
cs1=1;
e=0;
return(ret_data);
}

//画点函数。括号里的参数分别为：坐标(column,row),row为行,共64行,最小值为1,最大值为64;column为列,共128列,最小值为1,最大值为128.
void draw_point(uint column,uint row)
{
    uchar i,dat,page;
    row--;

    column--; //我们平常的“第1行”在液晶屏里叫“第0行”,所以减1,同理,列也如此
    page=row/8;
    page++;
    lcd_address(page,column); //设置地址
    Start_Read_Modify_Write(); //开始“读取-修改-写入”模式
    i=row%8; //行位置分成了第几页和这一页内的第几位,i=y%8即是这个第几位
    dat=1;
    dat<<=i; //用移位的方法,根据这个“第几位”算出这个字节的数值来,这个就是我们要画的“点”的一个字节数据

    dat|=read_data_twice(); //要画点的数据和刚读到的原来液晶屏的位置上的数据进行“或”运算,以免覆盖原来的数据。
    transfer_data(dat);

    End_Read_Modify_Write(); //结束“读取-修改-写入”模式
}

//“画直线”函数的注意事项:
//直线的斜率k=(y2-y1)/(x2-x1),k\|x1\|y1\|x2\|y2必须是浮点型的数据,除此之外,由于终点坐标可能小于起点坐标,所以斜率还有可能是负数的。
//由于k=(y2-y1)/(x2-x1),所以在x2=x1时,导致除数为0,数学上这是行不通的,所以另外想办法。同理k_y=(x2-x1)/(y2-y1)当y2=y1时也是行不通的。
//另外int是16位的整数,uchar却只有8位。
//
void draw_line(float x1,float y1,float x2, float y2)
{
    int i;float k,k_y,x,z;
    if( (y2-y1)==0&&(x2-x1)!=0 ) //如果y2-y1=0且x2-x1不等于0,则画一条横线。
    {
        if(x2<x1) //如果x2<x1,则两个坐标互换。画横线时,从左到右与从右到左是一样的结果。
        {
            z=x2; x2=x1;x1=z;
        }
        for(i=0;i<=(x2-x1);i++)
        {
            draw_point((x1+i),y1);
        }
    }
    else if( ((x2-x1)==0)&&((y2-y1)!=0) ) //如果x2-x1=0且y2-y1不等于0,则画一条竖线。
    {
        if(y2<y1) //如果y2<y1,则两个坐标互换。画竖线时,从上到下与从下到上是一样的结果。
        {
            z=y2; y2=y1;y1=z;
        }
        for(i=0;i<=(y2-y1);i++)
        {
            draw_point(x1,(y1+i));
        }
    }
}

```

```

    }
}
else if((x2-x1)==0&&(y2-y1)==0) //如果 (x2=x1)且(y2=y1)，画一个点即可
{
    draw_point(x1,y2);
}

else //否则，画斜线
{
    if(x2<x1) //如果 x2<x1，则两个坐标互换。（从起点画到终点）与（从终点画到起点）结果是一样的。
    {
        z=y2; y2=y1;y1=z;
        z=x2; x2=x1;x1=z;
    }
    else; //

    if(fabs(y2-y1)<=fabs(x2-x1)) //如果(y2-y1)的绝对值小于等于 (x2-x1) 的绝对值，就启动方案一：x 逐点扫描，y 按斜率计算，然后
画点 (x+i, y)
    {
        k=(y2-y1)/(x2-x1); //k 是斜率
        for(i=0;i<=(x2-x1);i++)
        {
            draw_point((x1+i), (y1+k*i));
        }
    }
    else //如果(y2-y1)的绝对值大于(x2-x1)的绝对值，就启动方案二：y 逐点扫描，x 按斜率计算，然后
画点 (x, y+i) 或 (x, y-i)
    {
        k_y=fabs((x2-x1)/(y2-y1)); //k_y 是反斜率（即 x 除以 y）。fabs 是浮点型数据的绝对值
        for(i=0;i<=fabs(y2-y1);i++)
        {
            x=x1+k_y*i;
            if((y2-y1)>0)
            {
                draw_point(x, (y1+i));
            }
            else
            {
                draw_point(x, (y1-i));
            }
        }
    }
}
}

void sleep()
{
    transfer_command(0xac);/*静态图标关闭*/
    transfer_command(0x00);/*静态图标寄存器设置：关闭。此指令与上述指令一起完成静态图标关闭*/
    transfer_command(0xae);/*显示：关*/
    transfer_command(0xa5);/*全屏显示：开*/
}

void wake()
{
    transfer_command(0xa4);/*全屏显示：关。进入正常模式*/
    transfer_command(0xad);/*静态图标开启*/
    transfer_command(0x03);/*静态图标寄存器设置：开。此指令与上述指令一起完成静态图标开启*/
    transfer_command(0xaf);/*显示：开*/
}

```



```

void main(void)
{
    initial_lcd();
    while(1)
    {
        clear_screen();
        draw_point(4,2); //画点
        draw_point(10,2);
        draw_point(16,2);
        draw_point(22,2);
        draw_point(28,2);
        draw_point(34,2);
        draw_point(40,2);
        draw_point(46,2);
        draw_point(52,2);
        draw_point(58,2);
        draw_point(64,2);
        draw_point(70,2);
        draw_point(76,2);
        draw_point(82,2);
        draw_point(88,2);
        draw_point(94,2);
        draw_point(100,2);
        draw_point(106,2);
        draw_point(112,2);
        draw_point(118,2);
        draw_point(124,2);

        draw_line(56,6,72,6); //画线
        draw_line(48,9,80,9);
        draw_line(40,12,88,12);
        draw_line(32,15,96,15);
        draw_line(24,18,104,18);
        draw_line(16,21,112,21);
        draw_line(8,24,120,24);
        draw_line(0,27,128,27);

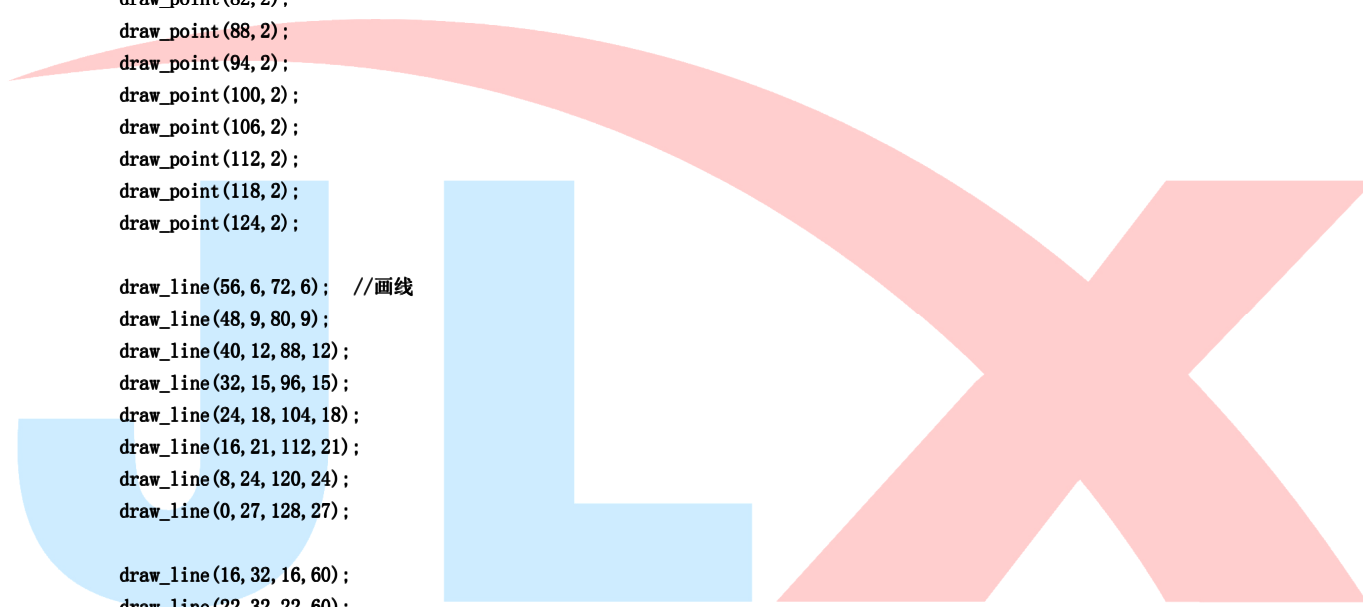
        draw_line(16,32,16,60);
        draw_line(22,32,22,60);
        draw_line(28,32,28,60);
        draw_line(34,32,34,60);
        draw_line(40,32,40,60);

        display_graphic_16x16(6,48,1,hua);
        display_graphic_16x16(6,64,1,dian);
        draw_line(47,40,47,57); //画矩形,画4条线实现
        draw_line(47,39,82,39);
        draw_line(47,58,82,58);
        draw_line(82,40,82,57);

        display_graphic_16x16(6,88,1,hua);
        display_graphic_16x16(6,104,1,xian);
        draw_line(87,40,87,57); //画矩形,画4条线实现
        draw_line(87,39,122,39);
        draw_line(87,58,122,58);
        draw_line(122,40,122,57);
        waitkey();

        clear_screen();
    }
}

```



```

display_graphic(bmp1);
waitkey();

clear_screen();
display_graphic(bmp2);
waitkey();

clear_screen(); //clear all dots
display_string_5x8(1, 1, 1, "MENU"); //显示 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)
display_string_5x8(3, 1, 0, "Select>>>>");
display_string_5x8(3, 64, 1, "1. Graphic");
display_string_5x8(4, 64, 0, "2. Chinese");
display_string_5x8(5, 64, 0, "3. Movie");
display_string_5x8(6, 64, 0, "4. Contrast");
display_string_5x8(7, 64, 0, "5. Mirror");
display_string_5x8(8, 1, 1, "PRE USER DEL NEW");
display_string_5x8(8, 19, 0, " ");
display_string_5x8(8, 65, 0, " ");
display_string_5x8(8, 97, 0, " ");
waitkey();

clear_screen();
display_graphic_32x32(1, 33, cheng1); //在第 1 页, 第 49 列显示单个汉字"成"*/
display_graphic_32x32(1, 65, gong); //在第 1 页, 第 49 列显示单个汉字"成"*/
display_graphic_16x16(5, 1, 1, zhuang1); //在第 5 页, 第 1 列显示单个汉字"状"
display_graphic_16x16(5, (1+16), 1, tail); //在第 5 页, 第 17 列显示单个汉字"态"
display_string_8x16(5, 33, 0, ":"); //在第 1 页, 第 1 列显示字符串
display_graphic_16x16(5, 41, 0, shi1); //在第 5 页, 第 41 列显示单个汉字"使"
display_graphic_16x16(5, (1+16*3+8), 0, yong1); //在第 5 页, 第 49 列显示单个汉字"用"
display_string_8x16(5, 89, 0, "00:00"); //显示 8x16 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)

waitkey();
clear_screen(); //clear all dots
display_string_8x16(1, 1, 0, "0123456789abcdef"); //显示 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)
display_string_8x16(3, 1, 0, " !@#%&'()*-+="); //同上
display_string_5x8(5, 1, 1, " !#$%&'()*+,-./01234");
display_string_5x8(6, 1, 0, "56789:;<=>?@ABCDEFGHI");
display_string_5x8(7, 1, 0, "JKLMNOPQRSTUVWXYZ[\]^");
display_string_5x8(8, 1, 0, "_`abcdefghijklmnopqrs");
test_display(0xff, 0xff);
waitkey();
test_display(0xaa, 0x55);
waitkey();
test_display(0x55, 0xaa);
waitkey();
clear_screen();
display_graphic(bmp3);
waitkey();
clear_screen();
display_graphic(bmp4);
waitkey();
sleep(); //进入睡眠模式
waitkey();
wake(); //退出睡眠模式
waitkey();
}
}

```

```
char code cheng1[]={
/*-- 文字： 成 --*/
/*-- 宋体 23； 此字体下对应的点阵为：宽 x 高=31x31 --*/
/*-- 高度不是 8 的倍数，现调整为：宽度 x 高度=32x32 --*/
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0C,
0xFC,0xFC,0x88,0x00,0x00,0x1C,0x78,0xF0,0xE0,0x00,0x80,0x80,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0xFF,0xFF,0x83,0x83,0x83,0x83,0x83,0x83,0x83,0x83,0xC3,0xC3,0x03,0x1F,
0xFF,0xFF,0x83,0x03,0x03,0x03,0xC3,0xF3,0xF3,0x63,0x03,0x03,0x00,0x00,0x00,0x00,
0x00,0x00,0xFC,0xFF,0x3F,0x00,0x80,0x00,0x00,0x80,0xFF,0xFF,0x03,0x00,0x00,0x03,
0x9F,0xFF,0xF8,0xF8,0xBE,0x1F,0x07,0x01,0x00,0x00,0xE0,0x20,0x00,0x00,0x20,0x38,
0x1F,0x07,0x01,0x00,0x00,0x01,0x01,0x07,0x07,0x23,0x31,0x18,0x0C,0x0E,0x07,0x03,
0x01,0x01,0x01,0x03,0x07,0x0F,0x0E,0x1C,0x1F,0x3F,0x30,0x00,0x00,0x00,0x00};

char code gong[]={
/*-- 文字： 功 --*/
/*-- 宋体 23； 此字体下对应的点阵为：宽 x 高=31x31 --*/
/*-- 高度不是 8 的倍数，现调整为：宽度 x 高度=32x32 --*/
0x00,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0x80,0xC0,0xC0,0xC0,0x80,
0x00,0x00,0x00,0xFC,0xFC,0xFC,0x08,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0xFF,0xFF,0xFF,0x00,0x00,0x00,0x04,0x04,0x04,0x04,
0x04,0xF4,0xFF,0xFF,0x1F,0x04,0x04,0x04,0xF4,0xFE,0xFE,0x7C,0x00,0x00,0x00,0x80,
0xC0,0xC0,0xC0,0xC0,0x60,0x7F,0x7F,0x3F,0x30,0x30,0x10,0x10,0x18,0x88,0xE0,0xFC,
0x7F,0x1F,0x07,0x00,0x00,0x00,0x00,0xFF,0xFF,0xFF,0x00,0x00,0x00,0x00,0x01,
0x01,0x00,0x00,0x00,0x00,0x20,0x20,0x30,0x18,0x1C,0x0E,0x07,0x03,0x01,0x00,0x04,
0x0C,0x0C,0x08,0x38,0x38,0x3E,0x1F,0x1F,0x07,0x00,0x00,0x00,0x00,0x00,0x00};

char code zhuang1[]={
/*-- 文字： 状 --*/
/*-- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 --*/
0x08,0x30,0x00,0xFF,0x20,0x20,0x20,0x20,0xFF,0x20,0xE1,0x26,0x2C,0x20,0x20,0x00,
0x04,0x02,0x01,0xFF,0x40,0x20,0x18,0x07,0x00,0x00,0x03,0x0C,0x30,0x60,0x20,0x00};

char code tail[]={
/*-- 文字： 态 --*/
/*-- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 --*/
0x00,0x04,0x04,0x04,0x84,0x44,0x34,0x4F,0x94,0x24,0x44,0x84,0x84,0x04,0x00,0x00,
0x00,0x60,0x39,0x01,0x00,0x3C,0x40,0x42,0x4C,0x40,0x40,0x70,0x04,0x09,0x31,0x00};

char code shi1[]={
/*-- 文字： 使 --*/
/*-- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 --*/
0x40,0x20,0xF0,0x1C,0x07,0xF2,0x94,0x94,0x94,0xFF,0x94,0x94,0x94,0xF4,0x04,0x00,
0x00,0x00,0x7F,0x00,0x40,0x41,0x22,0x14,0x0C,0x13,0x10,0x30,0x20,0x61,0x20,0x00};

char code yong1[]={
/*-- 文字： 用 --*/
/*-- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 --*/
0x00,0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0xFE,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,
0x80,0x40,0x30,0x0F,0x02,0x02,0x02,0x02,0xFF,0x02,0x02,0x42,0x82,0x7F,0x00,0x00};

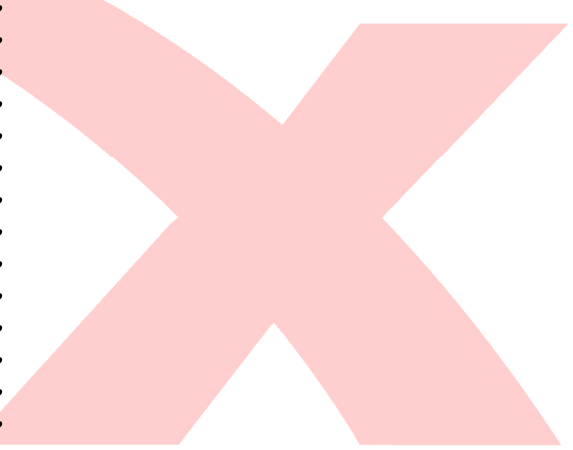
char code hua[]={
/*-- 文字： 画 --*/
/*-- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 --*/
0x02,0xF2,0x02,0x02,0xFA,0x4A,0x4A,0xFA,0x4A,0xFA,0x02,0x02,0xF2,0x02,0x00,
0x00,0x7F,0x20,0x20,0x2F,0x24,0x24,0x27,0x24,0x24,0x2F,0x20,0x20,0x7F,0x00,0x00};

char code dian[]={
/*-- 文字： 点 --*/
/*-- 宋体 12； 此字体下对应的点阵为：宽 x 高=16x16 --*/
0x00,0x00,0x00,0xE0,0x20,0x20,0x20,0x3F,0x24,0x24,0x24,0xF4,0x24,0x00,0x00,0x00,
```

0x00, 0x40, 0x30, 0x07, 0x12, 0x62, 0x02, 0x0A, 0x12, 0x62, 0x02, 0x0F, 0x10, 0x60, 0x00, 0x00, };

```
char code xian[]={
/*-- 文字: 线 --*/
/*-- 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 --*/
0x40, 0x60, 0x58, 0xC7, 0x62, 0x00, 0x90, 0x90, 0x90, 0xFF, 0x90, 0x92, 0x9C, 0x94, 0x80, 0x00,
0x20, 0x22, 0x23, 0x12, 0x12, 0x12, 0x20, 0x20, 0x10, 0x13, 0x0C, 0x14, 0x22, 0x40, 0xF8, 0x00, };
```

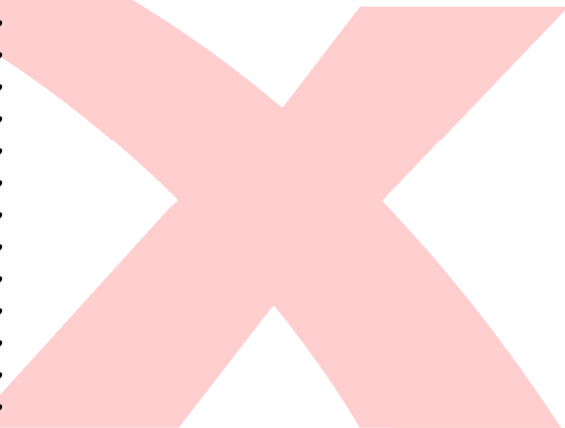
```
char code bmp2[]={
/*-- 调入了一幅图像: E:\work\图片收藏夹\黑白屏图片\JLX12864G-331.bmp --*/
/*-- 宽度 x 高度=128x64 --*/
0x10, 0x61, 0x06, 0xE0, 0x00, 0x26, 0x22, 0x1A, 0x02, 0xC2, 0x0A, 0x12, 0x32, 0x06, 0x02, 0x00,
0x10, 0x10, 0x10, 0xFE, 0x10, 0x10, 0xFE, 0x00, 0x00, 0xFC, 0x00, 0x00, 0xFE, 0x00, 0x00,
0x04, 0x04, 0x04, 0xE4, 0x24, 0x24, 0x25, 0xFE, 0x24, 0x24, 0x24, 0x24, 0xE4, 0x04, 0x04, 0x00,
0x00, 0x00, 0x00, 0x00, 0x7E, 0x2A, 0x2A, 0x2A, 0x2A, 0x2A, 0x2A, 0x7E, 0x00, 0x00, 0x00, 0x00,
0x02, 0xFE, 0x92, 0x92, 0x92, 0xFE, 0x12, 0x11, 0x12, 0x1C, 0xF0, 0x18, 0x17, 0x12, 0x10, 0x00,
0x20, 0x21, 0x2E, 0xE4, 0x00, 0x42, 0x42, 0xFE, 0x42, 0x42, 0x42, 0x02, 0xFE, 0x00, 0x00, 0x00,
0x00, 0x00, 0xF8, 0x48, 0x48, 0x48, 0x48, 0xFF, 0x48, 0x48, 0x48, 0x48, 0xF8, 0x00, 0x00, 0x00,
0x00, 0x00, 0x02, 0x02, 0x02, 0x02, 0x02, 0xE2, 0x12, 0x0A, 0x06, 0x02, 0x00, 0x80, 0x00, 0x00,
0x04, 0xFC, 0x03, 0x20, 0x20, 0x11, 0x11, 0x09, 0x05, 0xFF, 0x05, 0x09, 0x19, 0x31, 0x10, 0x00,
0x08, 0x08, 0x04, 0x47, 0x24, 0x18, 0x07, 0x00, 0x00, 0x1F, 0x00, 0x00, 0x00, 0x7F, 0x00, 0x00,
0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0xFF, 0x00, 0x00, 0x10, 0x20, 0x1F, 0x00, 0x00, 0x00,
0x00, 0x7F, 0x25, 0x25, 0x25, 0x25, 0x7F, 0x00, 0x00, 0x7F, 0x25, 0x25, 0x25, 0x7F, 0x00,
0x08, 0x1F, 0x08, 0x08, 0x04, 0xFF, 0x05, 0x81, 0x41, 0x31, 0x0F, 0x11, 0x21, 0xC1, 0x41, 0x00,
0x00, 0x00, 0x00, 0x7F, 0x20, 0x10, 0x00, 0x7F, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x40, 0x38, 0x00,
0x00, 0x00, 0x0F, 0x04, 0x04, 0x04, 0x04, 0x3F, 0x44, 0x44, 0x44, 0x44, 0x4F, 0x40, 0x70, 0x00,
0x01, 0x01, 0x01, 0x01, 0x01, 0x41, 0x81, 0x7F, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x08,
0xF8, 0x08, 0x08, 0x00, 0x08, 0xF8, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0x18, 0x68, 0x80,
0x80, 0x68, 0x18, 0x08, 0x00, 0x10, 0x10, 0xF8, 0x00, 0x00, 0x00, 0x00, 0x00, 0x70, 0x08, 0x08,
0x08, 0x88, 0x70, 0x00, 0x00, 0x70, 0x88, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0xE0, 0x10, 0x88,
0x88, 0x18, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x20, 0x10, 0xF8, 0x00, 0x00, 0xC0, 0x30, 0x08, 0x08,
0x08, 0x38, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x08, 0x88,
0x88, 0x48, 0x30, 0x00, 0x00, 0x30, 0x08, 0x88, 0x88, 0x48, 0x30, 0x00, 0x00, 0x10, 0x10, 0xF8,
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, 0xC0, 0x80, 0x80, 0x80,
0x7F, 0x00, 0x00, 0x00, 0x20, 0x3F, 0x20, 0x20, 0x20, 0x20, 0x30, 0x00, 0x20, 0x30, 0x2C, 0x03,
0x03, 0x2C, 0x30, 0x20, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00, 0x00, 0x00, 0x30, 0x28, 0x24,
0x22, 0x21, 0x30, 0x00, 0x00, 0x1C, 0x22, 0x21, 0x21, 0x22, 0x1C, 0x00, 0x00, 0x0F, 0x11, 0x20,
0x20, 0x11, 0x0E, 0x00, 0x00, 0x07, 0x04, 0x24, 0x24, 0x3F, 0x24, 0x00, 0x07, 0x18, 0x20, 0x20,
0x22, 0x1E, 0x02, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00, 0x18, 0x20, 0x20,
0x20, 0x11, 0x0E, 0x00, 0x00, 0x18, 0x20, 0x20, 0x20, 0x11, 0x0E, 0x00, 0x00, 0x20, 0x20, 0x3F,
0x20, 0x20, 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, 0xE0, 0x20, 0x20, 0x20, 0x3F, 0x24, 0x24, 0x24, 0xF4, 0x24, 0x00, 0x00,
0x00, 0xFE, 0x02, 0x12, 0x2A, 0xC6, 0x88, 0xC8, 0xB8, 0x8F, 0xE8, 0x88, 0x88, 0x88, 0x88, 0x00,
0x00, 0x00, 0x00, 0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x10, 0x10, 0xF8, 0x00, 0x00, 0x00,
0x00, 0x00, 0x70, 0x08, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x70, 0x88, 0x08, 0x08, 0x88, 0x70,
0x00, 0x00, 0x80, 0x80, 0x00, 0x80, 0x80, 0x80, 0x00, 0x00, 0xE0, 0x10, 0x88, 0x88, 0x18, 0x00,
0x00, 0x00, 0x00, 0xC0, 0x20, 0x10, 0xF8, 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, 0x40, 0x30, 0x07, 0x12, 0x62, 0x02, 0x0A, 0x12, 0x62, 0x02, 0x0F, 0x10, 0x60, 0x00,
0x00, 0xFF, 0x00, 0x02, 0x04, 0x03, 0x04, 0x04, 0x04, 0xFF, 0x04, 0x04, 0x04, 0x04,
0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00,
0x00, 0x00, 0x30, 0x28, 0x24, 0x22, 0x21, 0x30, 0x00, 0x00, 0x1C, 0x22, 0x21, 0x21, 0x22, 0x1C,
0x00, 0x00, 0x20, 0x31, 0x2E, 0x0E, 0x31, 0x20, 0x00, 0x00, 0x0F, 0x11, 0x20, 0x20, 0x11, 0x0E,
0x00, 0x00, 0x07, 0x04, 0x24, 0x24, 0x3F, 0x24, 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, 0x08, 0x08, 0x89, 0x4E, 0xAA, 0x18, 0x00, 0xFE, 0x02,
```




```

0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF0, 0xF0, 0xE0, 0xC0, 0x83, 0x0F, 0x7F, 0xDF, 0xEF, 0xFF, 0xF7,
0x19, 0x04, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0xE0, 0xF0, 0xF8, 0xFC, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE, 0xFC, 0xFC, 0xF8, 0xF8,
0xF1, 0xC3, 0x0F, 0x38, 0xE0, 0x80, 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, 0x7F, 0xD8, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x8F, 0x07, 0x63,
0x23, 0x27, 0x07, 0xFF, 0xFF, 0xFF, 0xFF, 0x7F, 0x7E, 0xFF, 0xFF, 0xFF, 0xFF, 0x8F, 0x47, 0x23,
0x23, 0x23, 0x47, 0x8F, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x7C, 0xC0, 0x7F, 0x11, 0xFF, 0x00,
0xC0, 0xF0, 0xF8, 0xFC, 0xFE, 0xFE, 0xFF, 0x7F, 0x3F, 0x3F, 0x3F, 0x3F, 0xFE, 0xFC, 0xF8,
0xF0, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xF3, 0xC1, 0x80, 0x9C, 0x88, 0x80, 0xC1, 0xFF, 0xFF, 0xFF,
0xFF, 0x7F, 0x0F, 0x80, 0x80, 0x7F, 0x40, 0x40, 0x40, 0xC0, 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, 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, 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, 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, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
};

```



//纵向取模, 适合 ST7565P, ST7565R, ST7567, UC1701X, KS0108 等驱动 IC 的液晶模块使用
char code ascii_table_8x16[95][16]={

```

//粗体 8x16 点阵的 ASCII 码的点阵数据, 从“JLX-GB2312”型号的字库 IC 中读出来的国标的。
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // - (即“空格”) ASCII 码: 0x20
0x00, 0x00, 0x38, 0xFC, 0xFC, 0x38, 0x00, 0x00, 0x00, 0x00, 0x0D, 0x0D, 0x00, 0x00, 0x00, // ! - ASCII 码: 0x21
0x00, 0x0E, 0x1E, 0x00, 0x00, 0x1E, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // " -
0x20, 0xF8, 0xF8, 0x20, 0xF8, 0xF8, 0x20, 0x00, 0x02, 0x0F, 0x0F, 0x02, 0x0F, 0x0F, 0x02, 0x00, // # -
0x38, 0x7C, 0x44, 0x47, 0x47, 0xC0, 0x98, 0x00, 0x06, 0xC0, 0x08, 0x38, 0x38, 0x0F, 0x07, 0x00, // $ -
0x30, 0x30, 0x00, 0x80, 0xC0, 0x60, 0x30, 0x00, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x0C, 0x0C, 0x00, // % -
0x80, 0xD8, 0x7C, 0xE4, 0xBC, 0xD8, 0x40, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, // & -
0x00, 0x10, 0x1E, 0x0E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // ' -
0x00, 0x00, 0xF0, 0xF8, 0x0C, 0x04, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x00, 0x00, // ( -

```


0x00, 0x00, 0x04, 0x0C, 0xF8, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x08, 0x0C, 0x07, 0x03, 0x00, 0x00,	//-) -	
0x80, 0xA0, 0xE0, 0xC0, 0xC0, 0xE0, 0xA0, 0x80, 0x00, 0x02, 0x03, 0x01, 0x01, 0x03, 0x02, 0x00,	//-* -	ASCII 码: 0X2A
0x00, 0x80, 0x80, 0xE0, 0xE0, 0x80, 0x80, 0x00, 0x00, 0x00, 0x03, 0x03, 0x00, 0x00, 0x00,	//+ -	
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x10, 0x1E, 0x0E, 0x00, 0x00, 0x00,	//, -	
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	//_ -	
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x0C, 0x00, 0x00, 0x00,	//. -	
0x00, 0x00, 0x00, 0x80, 0xC0, 0x60, 0x30, 0x00, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00,	// / -	
0xF8, 0xF8, 0x0C, 0xC4, 0x0C, 0xF8, 0xF0, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x0C, 0x07, 0x03, 0x00,	// - 0 -	ASCII 码: 0X30
0x00, 0x10, 0x18, 0xFC, 0xFC, 0x00, 0x00, 0x00, 0x08, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x00,	// - 1 -	
0x08, 0x0C, 0x84, 0xC4, 0x64, 0x3C, 0x18, 0x00, 0x0E, 0x0F, 0x09, 0x08, 0x08, 0x0C, 0x0C, 0x00,	// - 2 -	
0x08, 0x0C, 0x44, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x04, 0x0C, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - 3 -	
0xC0, 0xE0, 0xB0, 0x98, 0xFC, 0xFC, 0x80, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00,	// - 4 -	ASCII 码: 0X34
0x7C, 0x7C, 0x44, 0x44, 0x44, 0xC4, 0x84, 0x00, 0x04, 0x0C, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - 5 -	
0xF0, 0xF8, 0x4C, 0x44, 0x44, 0xC0, 0x80, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - 6 -	
0x0C, 0x0C, 0x04, 0x84, 0xC4, 0x7C, 0x3C, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x00,	// - 7 -	
0xB8, 0xFC, 0x44, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - 8 -	
0x38, 0x7C, 0x44, 0x44, 0x44, 0xFC, 0xF8, 0x00, 0x00, 0x08, 0x08, 0x08, 0x0C, 0x07, 0x03, 0x00,	// - 9 -	
0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x00, 0x06, 0x06, 0x00, 0x00, 0x00,	// - : -	
0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00, 0x08, 0x0E, 0x06, 0x00, 0x00, 0x00,	// - ; -	
0x00, 0x80, 0xC0, 0x60, 0x30, 0x18, 0x08, 0x00, 0x00, 0x00, 0x01, 0x03, 0x06, 0x0C, 0x08, 0x00,	// - < -	
0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x00, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x00,	// - = -	
0x00, 0x08, 0x18, 0x30, 0x60, 0xC0, 0x80, 0x00, 0x00, 0x08, 0x0C, 0x06, 0x03, 0x01, 0x00, 0x00,	// - > -	ASCII 码: 0X3E
0x18, 0x1C, 0x04, 0xC4, 0xE4, 0x3C, 0x18, 0x00, 0x00, 0x00, 0x0D, 0x0D, 0x00, 0x00, 0x00,	// - ? -	
0xF0, 0xF0, 0x08, 0xC8, 0xC8, 0xF8, 0xF0, 0x00, 0x07, 0x0F, 0x08, 0x0B, 0x0B, 0x0B, 0x01, 0x00,	// - @ -	
0xE0, 0xF0, 0x98, 0x8C, 0x98, 0xF0, 0xE0, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00,	// - A -	ASCII 码: 0X41
0x04, 0xFC, 0xFC, 0x44, 0x44, 0xFC, 0xB8, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - B -	
0xF0, 0xF8, 0x0C, 0x04, 0x04, 0x0C, 0x18, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x08, 0x0C, 0x06, 0x00,	// - C -	
0x04, 0xFC, 0xFC, 0x04, 0x0C, 0xF8, 0xF0, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x0C, 0x07, 0x03, 0x00,	// - D -	
0x04, 0xFC, 0xFC, 0x44, 0xE4, 0x0C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0C, 0x0E, 0x00,	// - E -	
0x04, 0xFC, 0xFC, 0x44, 0xE4, 0x0C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00,	// - F -	
0xF0, 0xF8, 0x0C, 0x84, 0x84, 0x8C, 0x98, 0x00, 0x03, 0x07, 0x0C, 0x08, 0x08, 0x07, 0x0F, 0x00,	// - G -	
0xFC, 0xFC, 0x40, 0x40, 0x40, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00,	// - H -	ASCII 码: 0X48
0x00, 0x00, 0x04, 0xFC, 0xFC, 0x04, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00,	// - I -	
0x00, 0x00, 0x00, 0x04, 0xFC, 0xFC, 0x04, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x0F, 0x07, 0x00, 0x00,	// - J -	
0x04, 0xFC, 0xFC, 0xC0, 0xE0, 0x3C, 0x1C, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x01, 0x0F, 0x0E, 0x00,	// - K -	
0x04, 0xFC, 0xFC, 0x04, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x0C, 0x0E, 0x00,	// - L -	
0xFC, 0xFC, 0x38, 0x70, 0x38, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00,	// - M -	
0xFC, 0xFC, 0x38, 0x70, 0xE0, 0xFC, 0xFC, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00,	// - N -	
0xF8, 0xFC, 0x04, 0x04, 0x04, 0xFC, 0xF8, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - O -	
0x04, 0xFC, 0xFC, 0x44, 0x44, 0x7C, 0x38, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00,	// - P -	
0xF8, 0xFC, 0x04, 0x04, 0x04, 0xFC, 0xF8, 0x00, 0x07, 0x0F, 0x08, 0x0E, 0x3C, 0x3F, 0x27, 0x00,	// - Q -	
0x04, 0xFC, 0xFC, 0x44, 0xC4, 0xFC, 0x38, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x00, 0x0F, 0x0F, 0x00,	// - R -	
0x18, 0x3C, 0x64, 0x44, 0xC4, 0x9C, 0x18, 0x00, 0x06, 0x0E, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - S -	
0x00, 0x1C, 0x0C, 0xFC, 0xFC, 0x0C, 0x1C, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00,	// - T -	
0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00,	// - U -	
0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x01, 0x03, 0x06, 0x0C, 0x06, 0x03, 0x01, 0x00,	// - V -	
0xFC, 0xFC, 0x00, 0x00, 0x00, 0xFC, 0xFC, 0x00, 0x07, 0x0F, 0x0E, 0x03, 0x0E, 0x0F, 0x07, 0x00,	// - W -	
0x0C, 0x3C, 0xF0, 0xE0, 0xF0, 0x3C, 0x0C, 0x00, 0x0C, 0x0F, 0x03, 0x01, 0x03, 0x0F, 0x0C, 0x00,	// - X -	
0x00, 0x0C, 0x7C, 0xC0, 0xC0, 0x7C, 0x3C, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00,	// - Y -	
0x1C, 0x0C, 0x84, 0xC4, 0x64, 0x3C, 0x1C, 0x00, 0x0E, 0x0F, 0x09, 0x08, 0x08, 0x0C, 0x0E, 0x00,	// - Z -	
0x00, 0x00, 0xFC, 0xFC, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x08, 0x08, 0x00, 0x00,	// - [-	
0x38, 0x70, 0xE0, 0xC0, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03, 0x07, 0x0E, 0x00,	// - \ -	
0x00, 0x00, 0x04, 0x04, 0xFC, 0xFC, 0x00, 0x00, 0x00, 0x00, 0x08, 0x08, 0x0F, 0x0F, 0x00, 0x00,	// -] -	
0x08, 0x0C, 0x06, 0x03, 0x06, 0x0C, 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,	// - ^ -	

```

0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, // - _
0x00, 0x00, 0x03, 0x07, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // - `
0x00, 0xA0, 0xA0, 0xA0, 0xE0, 0xC0, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, // - a ASCII 码: 0x61
0x04, 0xFC, 0xFC, 0x20, 0x60, 0xC0, 0x80, 0x00, 0x00, 0x0F, 0x0F, 0x08, 0x08, 0x0F, 0x07, 0x00, // - b
0xC0, 0xE0, 0x20, 0x20, 0x20, 0x60, 0x40, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0C, 0x04, 0x00, // - c
0x80, 0xC0, 0x60, 0x24, 0xFC, 0xFC, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, // - d
0xC0, 0xE0, 0xA0, 0xA0, 0xA0, 0xE0, 0xC0, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0C, 0x04, 0x00, // - e

0x40, 0xF8, 0xFC, 0x44, 0x0C, 0x18, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, // - f
0xC0, 0xE0, 0x20, 0x20, 0xC0, 0xE0, 0x20, 0x00, 0x27, 0x6F, 0x48, 0x48, 0x7F, 0x3F, 0x00, 0x00, // - g
0x04, 0xFC, 0xFC, 0x40, 0x20, 0xE0, 0xC0, 0x00, 0x08, 0x0F, 0x0F, 0x00, 0x00, 0x0F, 0x0F, 0x00, // - h
0x00, 0x00, 0x20, 0xEC, 0xEC, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x08, 0x00, 0x00, // - i
0x00, 0x00, 0x00, 0x00, 0x20, 0xEC, 0xEC, 0x00, 0x00, 0x30, 0x70, 0x40, 0x40, 0x7F, 0x3F, 0x00, // - j
0x04, 0xFC, 0xFC, 0x80, 0xC0, 0x60, 0x20, 0x00, 0x08, 0x0F, 0x0F, 0x01, 0x03, 0x0E, 0x0C, 0x00, // - k
0x00, 0x00, 0x04, 0xFC, 0xFC, 0x00, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x0F, 0x08, 0x00, 0x00, // - l
0xE0, 0xE0, 0x60, 0xC0, 0x60, 0xE0, 0xC0, 0x00, 0x0F, 0x0F, 0x00, 0x07, 0x00, 0x0F, 0x0F, 0x00, // - m
0x20, 0xE0, 0xC0, 0x20, 0x20, 0xE0, 0xC0, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x0F, 0x0F, 0x00, // - n
0xC0, 0xE0, 0x20, 0x20, 0x20, 0xE0, 0xC0, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x08, 0x0F, 0x07, 0x00, // - o

0x20, 0xE0, 0xC0, 0x20, 0x20, 0xE0, 0xC0, 0x00, 0x40, 0x7F, 0x7F, 0x48, 0x08, 0x0F, 0x07, 0x00, // - p
0xC0, 0xE0, 0x20, 0x20, 0xC0, 0xE0, 0x20, 0x00, 0x07, 0x0F, 0x08, 0x48, 0x7F, 0x7F, 0x40, 0x00, // - q
0x20, 0xE0, 0xC0, 0x60, 0x20, 0xE0, 0xC0, 0x00, 0x08, 0x0F, 0x0F, 0x08, 0x00, 0x00, 0x00, 0x00, // - r
0x40, 0xE0, 0xA0, 0x20, 0x20, 0x60, 0x40, 0x00, 0x04, 0x0C, 0x09, 0x09, 0x0B, 0x0E, 0x04, 0x00, // - s
0x20, 0x20, 0xF8, 0xFC, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x0C, 0x04, 0x00, // - t
0xE0, 0xE0, 0x00, 0x00, 0xE0, 0xE0, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x07, 0x0F, 0x08, 0x00, // - u
0x00, 0xE0, 0xE0, 0x00, 0x00, 0xE0, 0xE0, 0x00, 0x00, 0x03, 0x07, 0x0C, 0x0C, 0x07, 0x03, 0x00, // - v
0xE0, 0xE0, 0x00, 0x80, 0x00, 0xE0, 0xE0, 0x00, 0x07, 0x0F, 0x0C, 0x07, 0x0C, 0x0F, 0x07, 0x00, // - w
0x20, 0x60, 0xC0, 0x80, 0xC0, 0x60, 0x20, 0x00, 0x08, 0x0C, 0x07, 0x03, 0x07, 0x0C, 0x08, 0x00, // - x
0xE0, 0xE0, 0x00, 0x00, 0x00, 0xE0, 0xE0, 0x00, 0x47, 0x4F, 0x48, 0x48, 0x68, 0x3F, 0x1F, 0x00, // - y

0x60, 0x60, 0x20, 0xA0, 0xE0, 0x60, 0x20, 0x00, 0x0C, 0x0E, 0x0B, 0x09, 0x08, 0x0C, 0x0C, 0x00, // - z //
0x00, 0x40, 0x40, 0xF8, 0xBC, 0x04, 0x04, 0x00, 0x00, 0x00, 0x00, 0x07, 0x0F, 0x08, 0x08, 0x00, // - { //
0x00, 0x00, 0x00, 0xBC, 0xBC, 0x00, 0x00, 0x00, 0x00, 0x0F, 0x0F, 0x00, 0x00, 0x00, // - | //
0x00, 0x04, 0x04, 0xBC, 0xF8, 0x40, 0x40, 0x00, 0x00, 0x08, 0x08, 0x0F, 0x07, 0x00, 0x00, 0x00, // - } //
0x08, 0x0C, 0x04, 0x0C, 0x08, 0x0C, 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, // - ~ ASCII 码: 0x7E

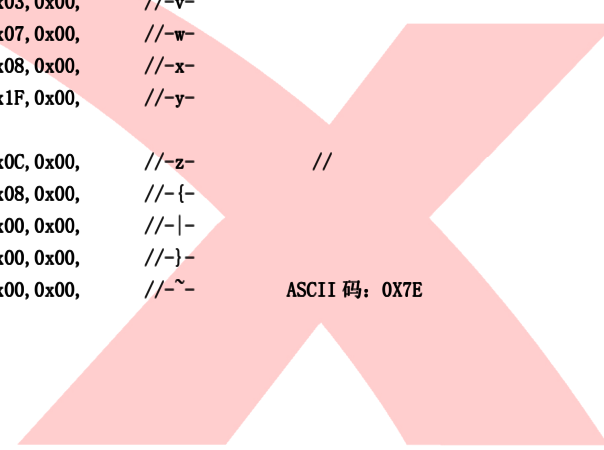
```

```

};

char code ascii_table_5x8[95][5]={
/*全体 ASCII 列表:5x8 点阵*/
0x00, 0x00, 0x00, 0x00, 0x00, // - //space
0x00, 0x00, 0x4f, 0x00, 0x00, // - !
0x00, 0x07, 0x00, 0x07, 0x00, // - "
0x14, 0x7f, 0x14, 0x7f, 0x14, // - #
0x24, 0x2a, 0x7f, 0x2a, 0x12, // - $
0x23, 0x13, 0x08, 0x64, 0x62, // - %
0x36, 0x49, 0x55, 0x22, 0x50, // - &
0x00, 0x05, 0x07, 0x00, 0x00, // - '
0x00, 0x1c, 0x22, 0x41, 0x00, // - (
0x00, 0x41, 0x22, 0x1c, 0x00, // - )
0x14, 0x08, 0x3e, 0x08, 0x14, // - *
0x08, 0x08, 0x3e, 0x08, 0x08, // - +
0x00, 0x50, 0x30, 0x00, 0x00, // - ,
0x08, 0x08, 0x08, 0x08, 0x08, // - -
0x00, 0x60, 0x60, 0x00, 0x00, // - .
0x20, 0x10, 0x08, 0x04, 0x02, // - /
0x3e, 0x51, 0x49, 0x45, 0x3e, // - 0
0x00, 0x42, 0x7f, 0x40, 0x00, // - 1
0x42, 0x61, 0x51, 0x49, 0x46, // - 2
0x21, 0x41, 0x45, 0x4b, 0x31, // - 3

```



0x18, 0x14, 0x12, 0x7f, 0x10, //-4-
 0x27, 0x45, 0x45, 0x45, 0x39, //-5-
 0x3c, 0x4a, 0x49, 0x49, 0x30, //-6-
 0x01, 0x71, 0x09, 0x05, 0x03, //-7-
 0x36, 0x49, 0x49, 0x49, 0x36, //-8-
 0x06, 0x49, 0x49, 0x29, 0x1e, //-9-
 0x00, 0x36, 0x36, 0x00, 0x00, //-:-
 0x00, 0x56, 0x36, 0x00, 0x00, //-;-
 0x08, 0x14, 0x22, 0x41, 0x00, //-<-
 0x14, 0x14, 0x14, 0x14, 0x14, //-==
 0x00, 0x41, 0x22, 0x14, 0x08, //->-
 0x02, 0x01, 0x51, 0x09, 0x06, //-?-
 0x32, 0x49, 0x79, 0x41, 0x3e, //-@-
 0x7e, 0x11, 0x11, 0x11, 0x7e, //-A-
 0x7f, 0x49, 0x49, 0x49, 0x36, //-B-
 0x3e, 0x41, 0x41, 0x41, 0x22, //-C-
 0x7f, 0x41, 0x41, 0x22, 0x1c, //-D-
 0x7f, 0x49, 0x49, 0x49, 0x41, //-E-
 0x7f, 0x09, 0x09, 0x09, 0x01, //-F-
 0x3e, 0x41, 0x49, 0x49, 0x7a, //-G-
 0x7f, 0x08, 0x08, 0x08, 0x7f, //-H-
 0x00, 0x41, 0x7f, 0x41, 0x00, //-I-
 0x20, 0x40, 0x41, 0x3f, 0x01, //-J-
 0x7f, 0x08, 0x14, 0x22, 0x41, //-K-
 0x7f, 0x40, 0x40, 0x40, 0x40, //-L-
 0x7f, 0x02, 0x0c, 0x02, 0x7f, //-M-
 0x7f, 0x04, 0x08, 0x10, 0x7f, //-N-
 0x3e, 0x41, 0x41, 0x41, 0x3e, //-O-
 0x7f, 0x09, 0x09, 0x09, 0x06, //-P-
 0x3e, 0x41, 0x51, 0x21, 0x5e, //-Q-
 0x7f, 0x09, 0x19, 0x29, 0x46, //-R-
 0x46, 0x49, 0x49, 0x49, 0x31, //-S-
 0x01, 0x01, 0x7f, 0x01, 0x01, //-T-
 0x3f, 0x40, 0x40, 0x40, 0x3f, //-U-
 0x1f, 0x20, 0x40, 0x20, 0x1f, //-V-
 0x3f, 0x40, 0x38, 0x40, 0x3f, //-W-
 0x63, 0x14, 0x08, 0x14, 0x63, //-X-
 0x07, 0x08, 0x70, 0x08, 0x07, //-Y-
 0x61, 0x51, 0x49, 0x45, 0x43, //-Z-
 0x00, 0x7f, 0x41, 0x41, 0x00, //-[-
 0x02, 0x04, 0x08, 0x10, 0x20, //-\-
 0x00, 0x41, 0x41, 0x7f, 0x00, //-]-
 0x04, 0x02, 0x01, 0x02, 0x04, //-^-
 0x40, 0x40, 0x40, 0x40, 0x40, //-_-
 0x01, 0x02, 0x04, 0x00, 0x00, //-^-
 0x20, 0x54, 0x54, 0x54, 0x78, //-a-
 0x7f, 0x48, 0x48, 0x48, 0x30, //-b-
 0x38, 0x44, 0x44, 0x44, 0x44, //-c-
 0x30, 0x48, 0x48, 0x48, 0x7f, //-d-
 0x38, 0x54, 0x54, 0x54, 0x58, //-e-
 0x00, 0x08, 0x7e, 0x09, 0x02, //-f-
 0x48, 0x54, 0x54, 0x54, 0x3c, //-g-
 0x7f, 0x08, 0x08, 0x08, 0x70, //-h-
 0x00, 0x00, 0x7a, 0x00, 0x00, //-i-
 0x20, 0x40, 0x40, 0x3d, 0x00, //-j-
 0x7f, 0x20, 0x28, 0x44, 0x00, //-k-
 0x00, 0x41, 0x7f, 0x40, 0x00, //-l-
 0x7c, 0x04, 0x38, 0x04, 0x7c, //-m-
 0x7c, 0x08, 0x04, 0x04, 0x78, //-n-
 0x38, 0x44, 0x44, 0x44, 0x38, //-o-



```

0x7c, 0x14, 0x14, 0x14, 0x08, //-p-
0x08, 0x14, 0x14, 0x14, 0x7c, //-q-
0x7c, 0x08, 0x04, 0x04, 0x08, //-r-
0x48, 0x54, 0x54, 0x54, 0x24, //-s-
0x04, 0x04, 0x3f, 0x44, 0x24, //-t-
0x3c, 0x40, 0x40, 0x40, 0x3c, //-u-
0x1c, 0x20, 0x40, 0x20, 0x1c, //-v-
0x3c, 0x40, 0x30, 0x40, 0x3c, //-w-
0x44, 0x28, 0x10, 0x28, 0x44, //-x-
0x04, 0x48, 0x30, 0x08, 0x04, //-y-
0x44, 0x64, 0x54, 0x4c, 0x44, //-z-
0x08, 0x36, 0x41, 0x41, 0x00, //-{-
0x00, 0x00, 0x77, 0x00, 0x00, //-|-
0x00, 0x41, 0x41, 0x36, 0x08, //-}-
0x04, 0x02, 0x02, 0x02, 0x01, //-~-
};
    
```

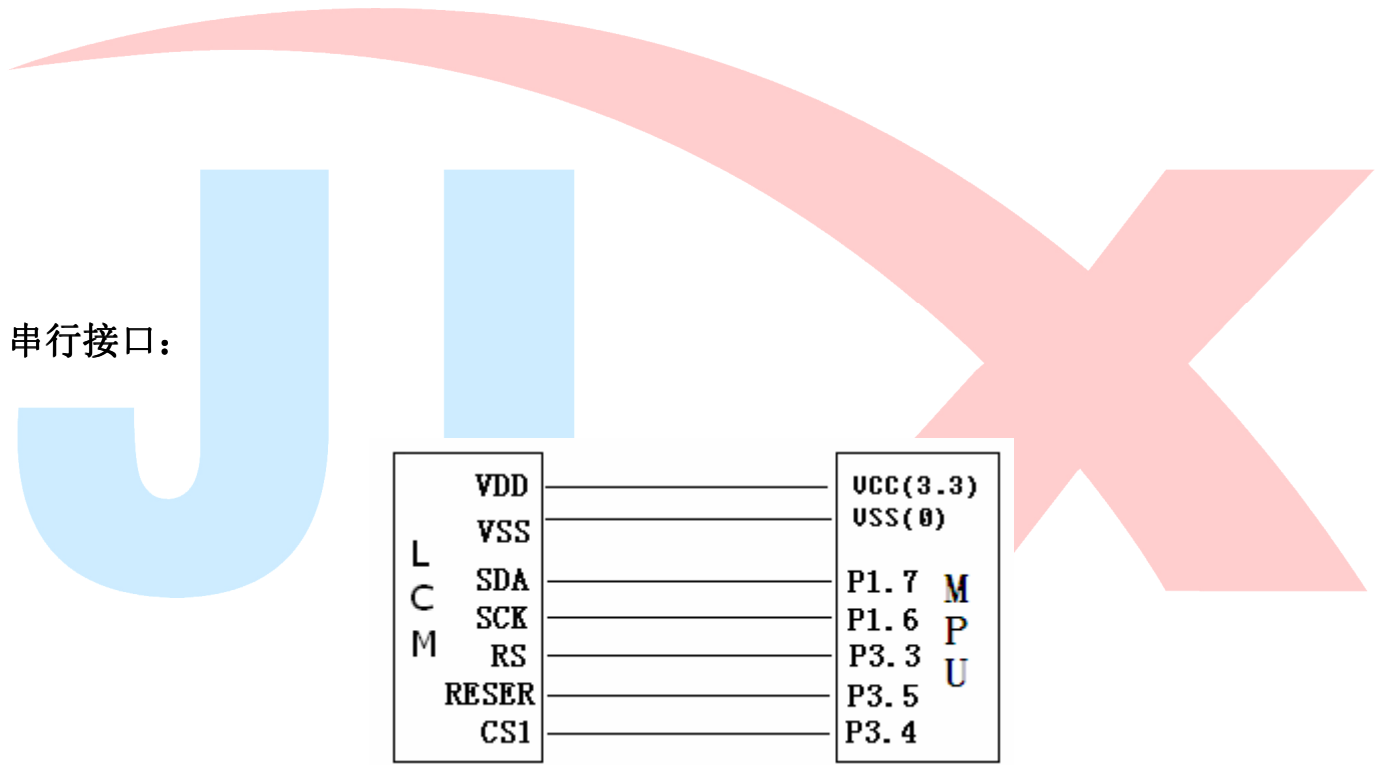
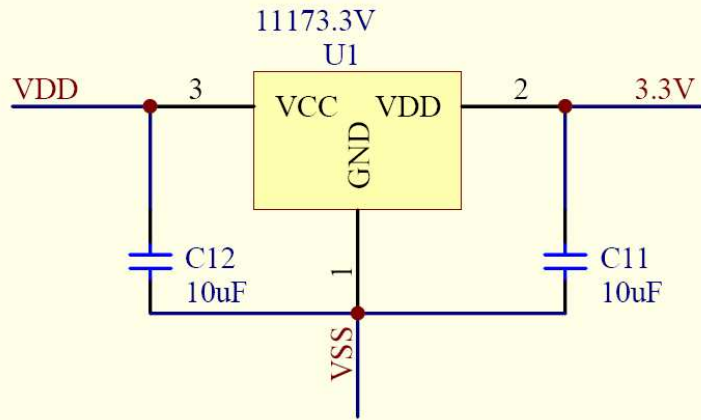
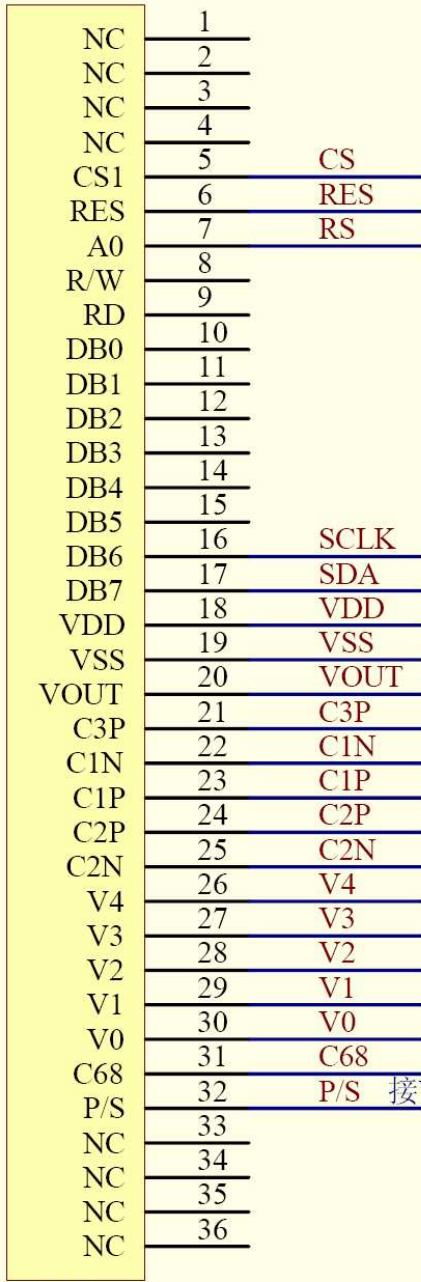


图 9. 串行接口

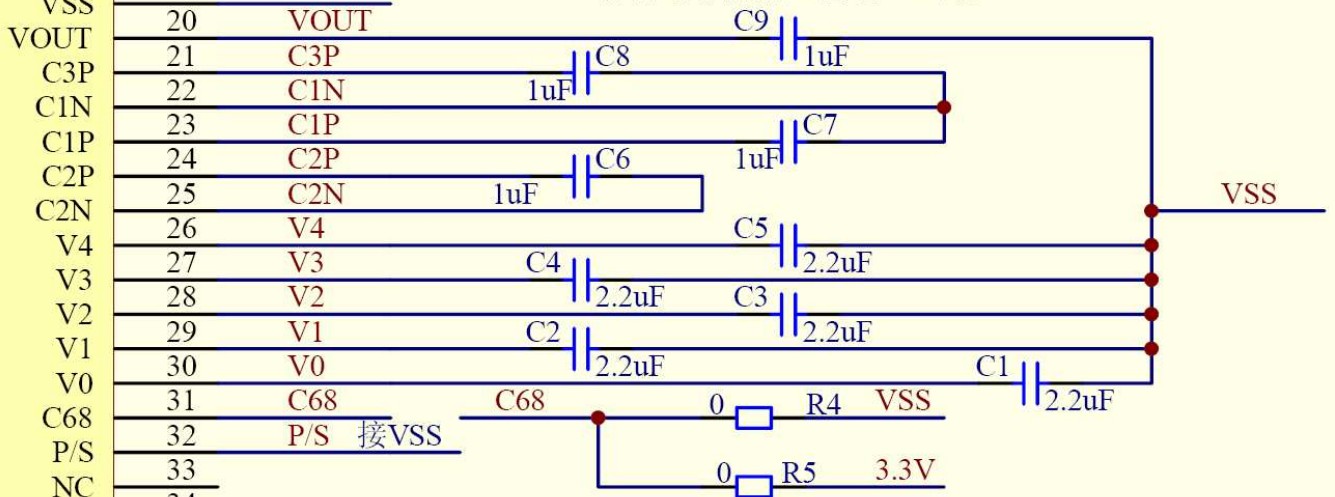
串行电路图

36PIN



液晶模块使用电压是2.5V-3.5V
CPU是5V时,用1117/3.3V,将电压转成3.3V
CPU是3.3V时不用1117/3.3V,直接接3.3V

所有电容的耐压值选25V或以上



C68:6800时序;R4空, R5接0欧电阻。
8080时序:R5空, R4接0欧电阻。
例程:6800

36PIN

7.52、以下为串行方式例程序

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:

注意:串行不能用读改写指令

```

sbit rs=P3^3; /*接口定义:LCD的rs*/
sbit sclk=P1^6; /*接口定义:LCD的sclk*/
sbit sid=P1^7; /*接口定义:LCD的sid*/
sbit reset=P3^5; /*接口定义:LCD的reset*/
sbit cs1=P3^4; /*接口定义:LCD的cs1*/
sbit key=P2^0; //P2.0口与GND之间接一个按键
    
```

```
/*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<<=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<<=1;  
    }  
    cs1=1;  
}
```