JLX128640LED-13015 中文使用说明书

（插接式 FPC）

目　　录

1．概述

晶联讯电子专注于 OLED 屏及液晶模块的研发、制造。所生产 JLX12864OLED-13015 型 OLED 模块由于使用方便、无需背光、视角宽、显示清晰、超薄，广泛应用于各种人机交流面板。

JLX12864OLED-13015 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．JLX12864OLED-13015 图像型点阵 OLED 模块的特性

2.1 结构牢：插接式 FPC。

2.2 IC 采用 SH1106,功能强大，稳定性好

2.3 功耗低。

2.4 显示内容:

●128*64 点阵单色图片;

●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 8 字/行*4 行。

2.5 指令功能强:可组合成各种输入、显示、移位方式以满足不同的要求;

2.6 接口方式：4 线 SPI 串行接口、并口、I^2C 接口。

2.7 工作温度宽：-40℃～+85℃;

2.8 储存温度宽：-40℃～+85℃;

3. 外形尺寸及接口引脚功能

3.1 外形图

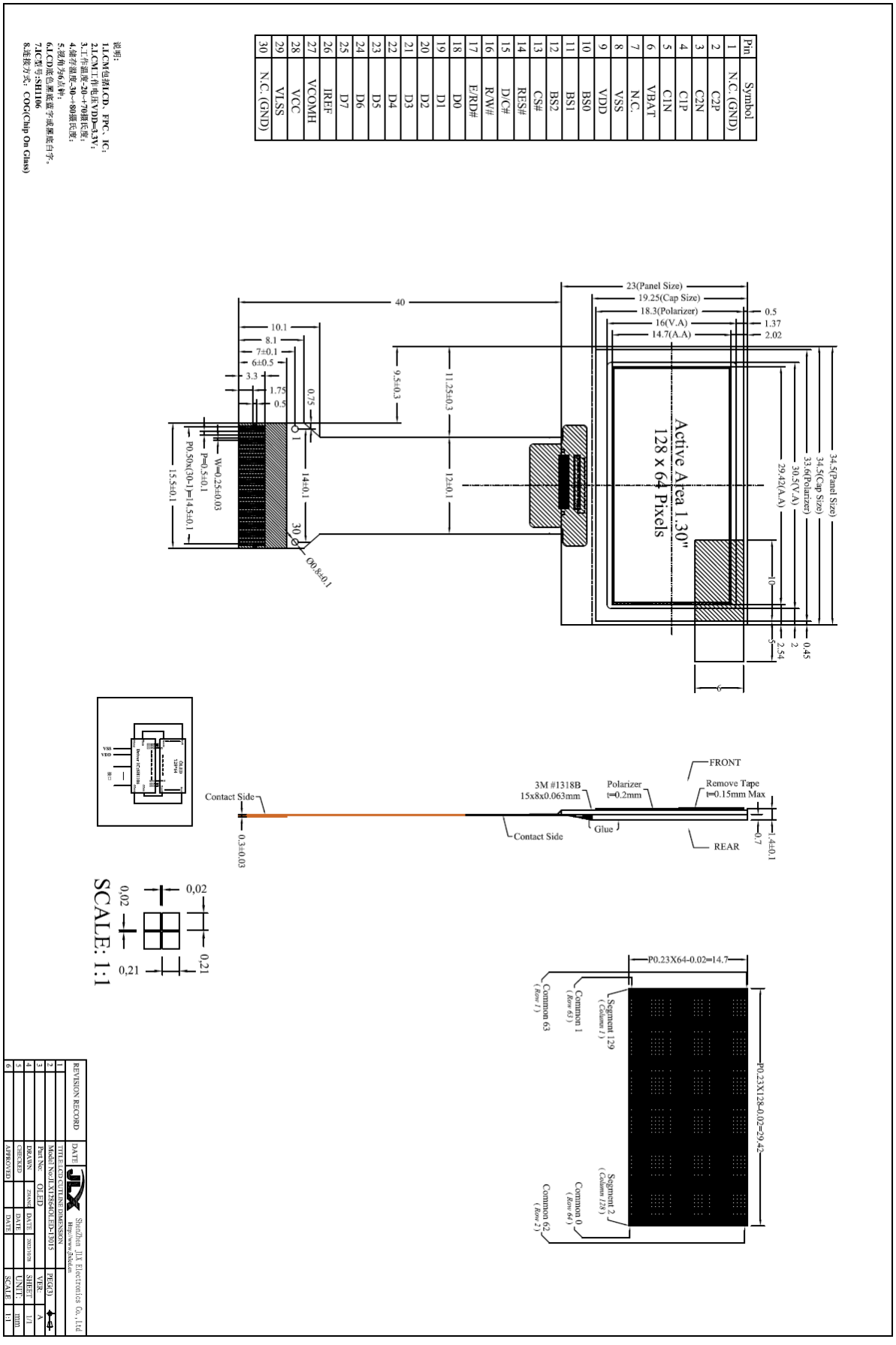
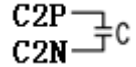
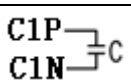
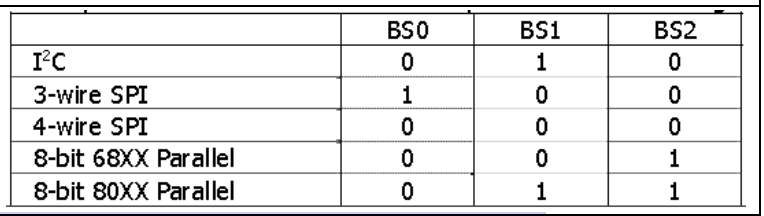
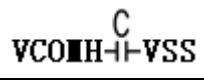
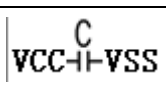


图 1. OLED 模块外形尺寸

模块的接口引脚功能

引 线 号	符 号	名 称	功 能
1	NC(GND)	NC	悬空或接地
2	C2P	升压电容	
3	C2N	升压电容	
4	C1P	升压电容	
5	C1N	升压电容	
6	VBAT	VBAT	接 VDD
7	NC	NC	空脚
8	VSS	接地	0V
9	VDD	电源电路	3.3V
10	BS0	BS0	
11	BS1	BS1	
12	BS2	BS2	
13	CS#	片选	低电平片选（I2C 接口接地）
14	RES#	复位	低电平复位，复位完成后，回到高电平，OLED 模块开始工作
15	D/C#	寄存选择信号	H：数据存储器 0：指令存储（IC 资料上缩写为“A0”） （I2C 接口接地（SA0））
16	R/W#	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. 当选择串行或 I2C 接口时，此引脚必须接 VSS
17	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. 当选择串行或 I2C 接口时，此引脚必须接 VSS
18	D0(SCL)	I/O	数据总线（串口或 I2C 接口做串行时钟 SCL）
19	D1(SDA)	I/O	数据总线（串口或 I2C 接口做串行数据 SDA）
20	D2	I/O	数据总线（串口时：空脚）
21	D3	I/O	数据总线（串口或 I2C 接口接地）
22	D4	I/O	数据总线（串口或 I2C 接口接地）
23	D5	I/O	数据总线（串口或 I2C 接口接地）
24	D6	I/O	数据总线（串口或 I2C 接口接地）
25	D7	I/O	数据总线（串口或 I2C 接口接地）
26	IREF	IREF	接 390K 电阻到 VSS
27	VCOMH	VCOMH	
28	VCC	升压电源	
29	VLSS	VLSS	接地
30	NC(GND)	NC	悬空或接地

	BS0	BS1	BS2
I²C	0	1	0
3-wire SPI	1	0	0
4-wire SPI	0	0	0
8-bit 68XX Parallel	0	0	1
8-bit 80XX Parallel	0	1	1

表 1：模块的接口引脚功能

4. 基本原理

4.1 OLED 屏

在 **OLED** 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）.

电路框图

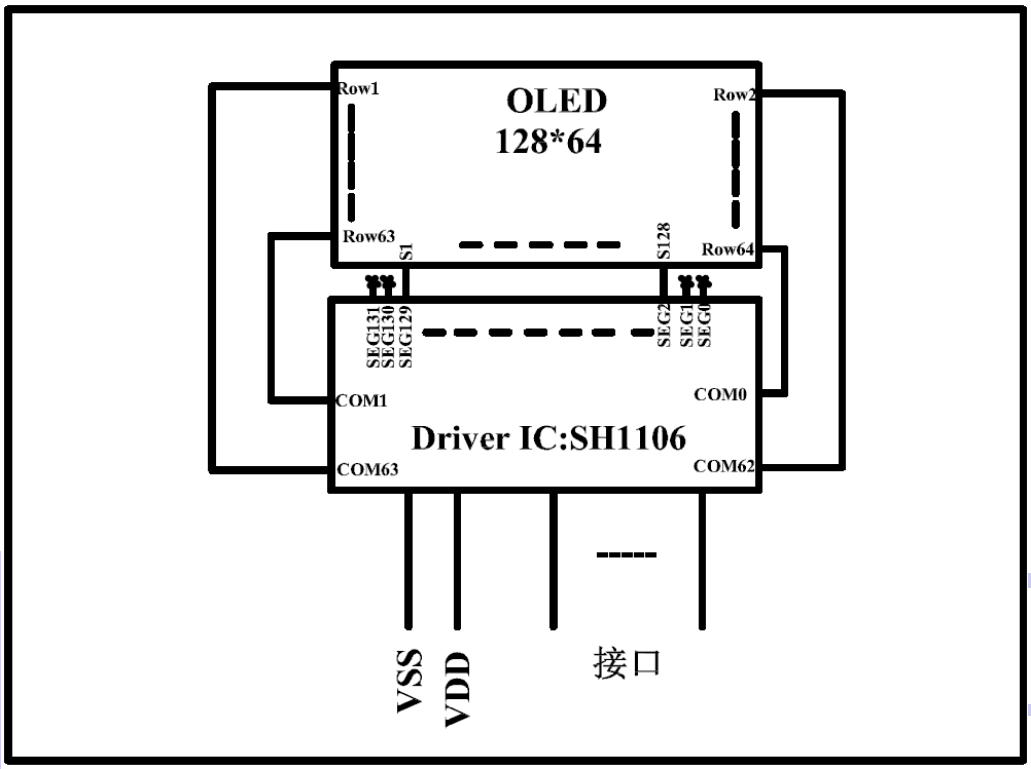


图 2：电路框图

5. 技术参数

5.1 最大极限参数（超过极限参数则会损坏 OLED 模块）

名称	符号	标准值			单位
		最小	典型	最大	
电路电源	VDD – VSS	2.7	3.3	3.5	V
OLED 驱动电压	VCC	6.4	—	14	V
静电电压		—	—	100	V
工作温度		-40		+85	℃
储存温度		-40		+85	℃

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（当 3.3V 供电时）	VDD		2.7	3.3	3.5	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}		0.8xVDD	—	VDD	V
输出低电平	V_{OLC}		VSS	—	0.2xVDD	V
模块工作电流	I_{DD}	VDD = 3.3V	0.3	75	200	mA

表 3：直流（DC）参数

6. 读写时序特性

6.1.1 串行接口：

从 CPU 写到 SH1106（Writing Data from CPU to SH1106）

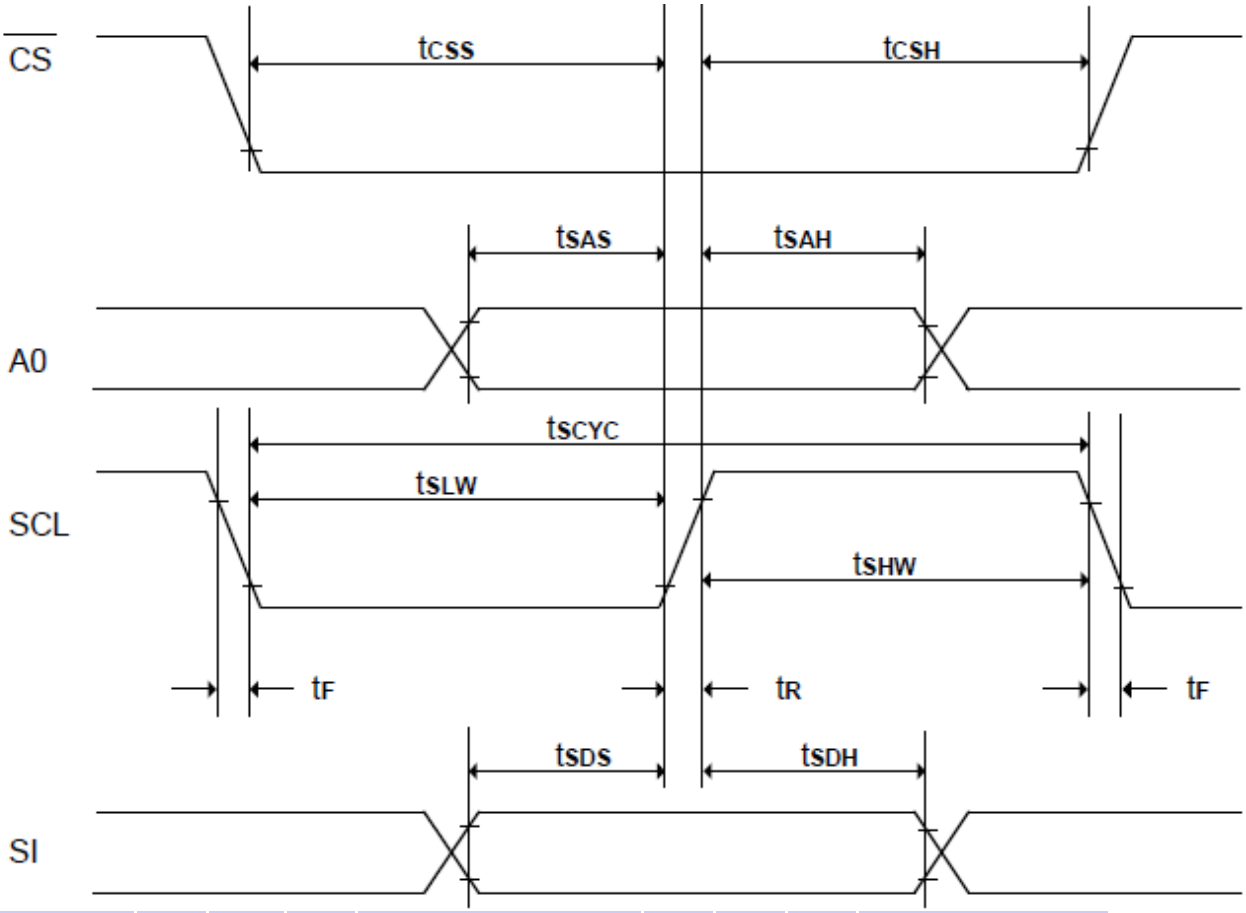


图 3. 从 CPU 写到 SH1106（Writing Data from CPU to SH1106）

6.1.2 串行接口：时序要求（AC 参数）：

写数据到 SH1106 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）	Tscyc		250	—	—	ns
地址建立时间（Address setup time)	TSAS		150	—	—	ns
地址保持时间 (Address hold time)	Tsah		150	—	—	ns
数据建立时间 (Data setup time)	Tsds		100	—	—	ns
数据保持时间 (Data hold time)	TSDH		100	—	—	ns
片选信号建立时间（CS-SCL time）	Tcss		120	—	—	ns
片选信号保持时间（CS-SCL time）	Tcsh		60	—	—	ns

保持SCK高电平脉宽 （SCK “H” pulse width）	T_{shw}		100	—	—	ns
保持SCK低电平脉宽 （SCK “L” pulse width）	T_{SLW}		100	—	—	ns
上升时间 （Rise time）	T_R				15	ns
下降时间 （Fall time）	T_F				15	ns

*（VDD =2.4V～3.5V, Ta = 25℃）

6.2.1 并行接口：

6800 时序：

从 CPU 写到 SH1106（Writing Data from CPU to SH1106）

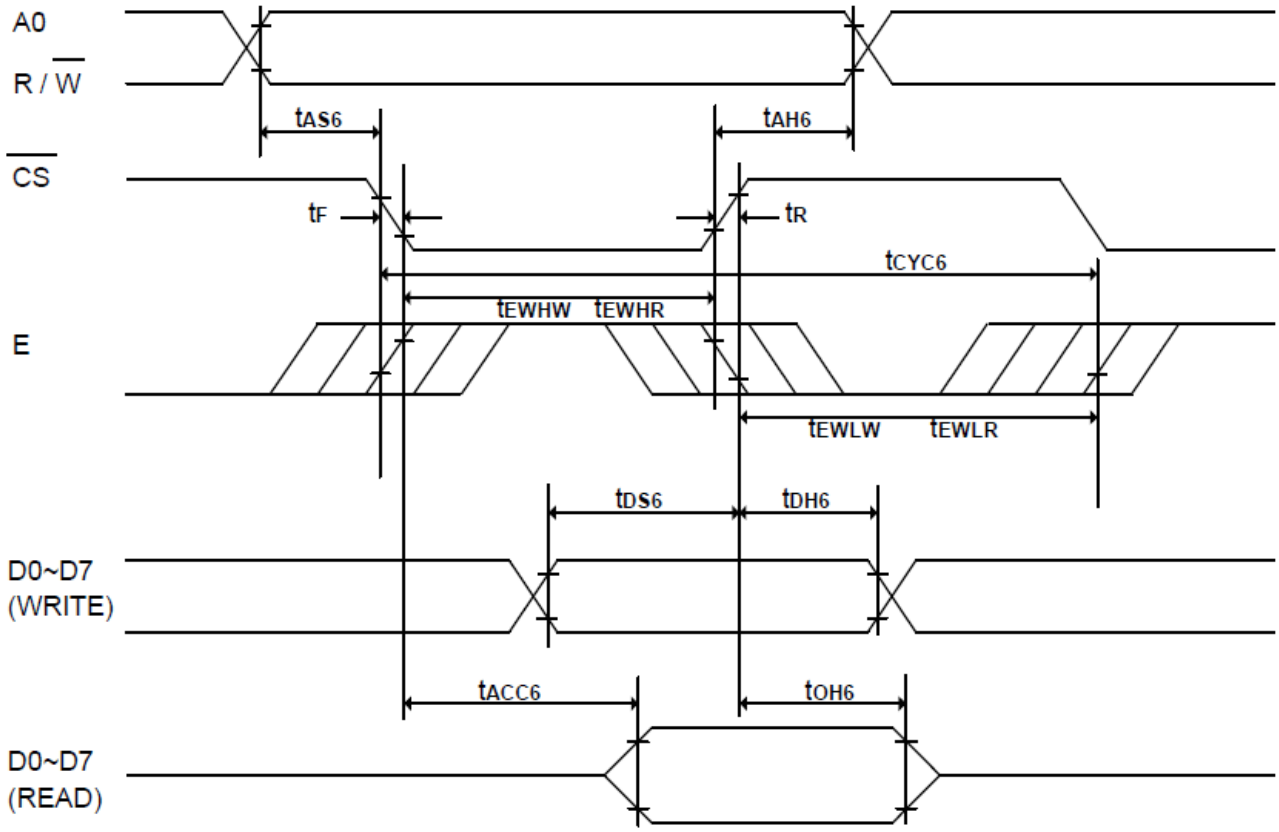


图 4. **从 CPU 写到 SH1106（Writing Data from CPU to SH1106）**

8080 时序：

从 CPU 写到 SH1106（Writing Data from CPU to SH1106）

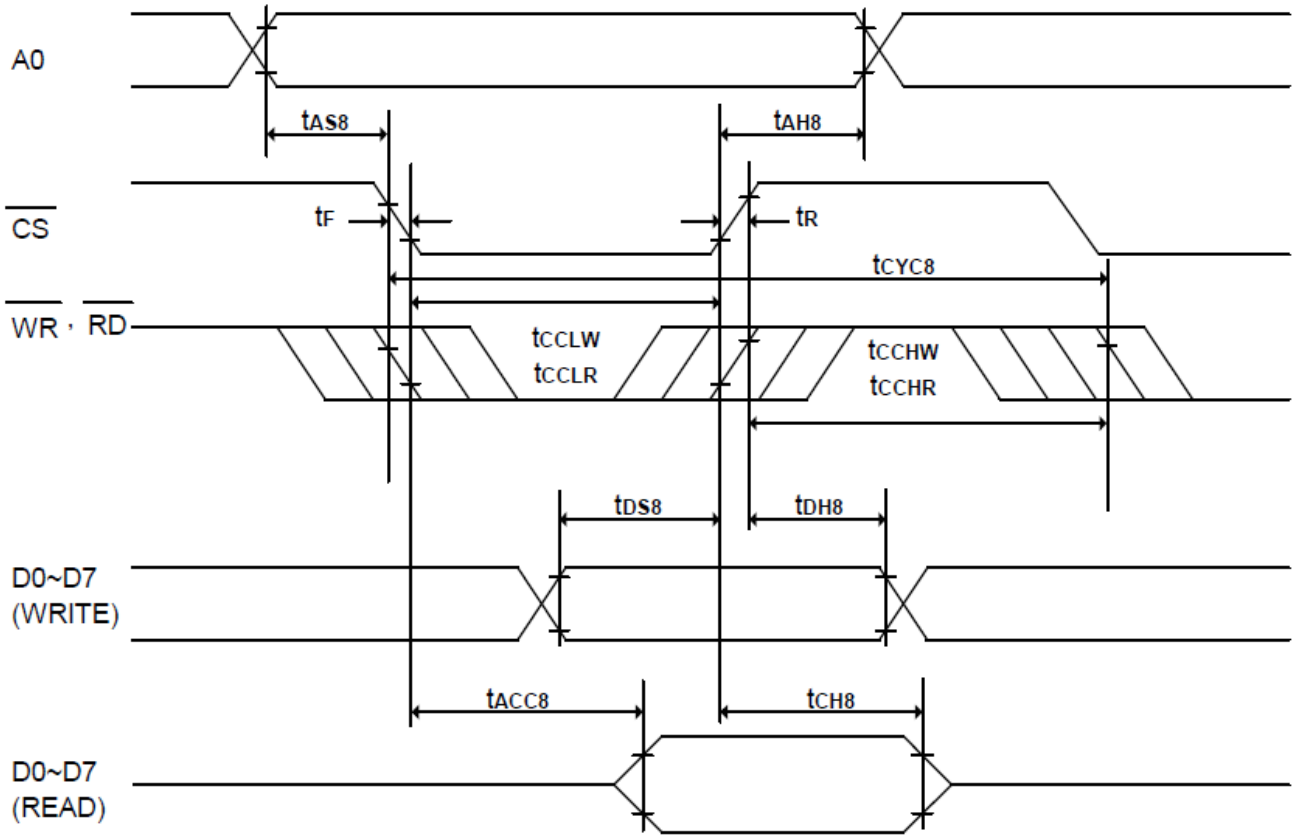


图 5. **从 CPU 写到 SH1106（Writing Data from CPU to SH1106）**

6.2.2 并行接口：

6800 时序要求（AC 参数）：

写数据到 SH1106 的时序要求： **表 5.**

项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
时钟周期（System cycle time）	T_{cyc6}		300	—	—	ns
地址建立时间（Address setup time)	T_{AS6}		0	—	—	ns
地址保持时间 (Address hold time)	T_{ah6}		0	—	—	ns
数据建立时间 (Data setup time)	T_{ds6}		40	—	—	ns
数据保持时间 (Data hold time)	T_{DH6}		15	—	—	ns
输出禁用时间（Output disable time）	T_{OH6}	CL = 100pF	10	—	70	ns
访问时间（Access time）	T_{Acc6}	CL = 100pF	—	—	140	ns
启用 H 脉冲宽度（写）（Enable H pulse width (Write)）	T_{EWHW}		100	—	—	ns

启用 H 脉冲宽度（读）（Enable H pulse width (Read)）	T_{EWHR}		120	—	—	ns
启用 L 脉冲宽度（写）Enable L pulse width (Write)	T_{EWLW}		100			ns
启用 L 脉冲宽度（读）Enable L pulse width (Read)	T_{EWLR}		100			ns
上升时间（Rise time）	T_R				15	ns
下降时间（Fall time）	T_F				15	ns

*（VDD =2.4V～3.5V, Ta = 25℃）

8080 时序要求（AC 参数）：

写数据到 SH1106 的时序要求： **表 6.**

项 目	符 号	测试条件	极限值			单位
			MIN	TYPE	MAX	
时钟周期（System cycle time）	T_{cyc8}		300	—	—	ns
地址建立时间（Address setup time)	T_{AS8}		0	—	—	ns
地址保持时间 (Address hold time)	T_{ah8}		0	—	—	ns
数据建立时间 (Data setup time)	T_{ds8}		40	—	—	ns
数据保持时间 (Data hold time)	T_{DH8}		15	—	—	ns
输出禁用时间（Output disable time）	T_{OH8}	CL = 100pF	10	—	70	ns
访问时间（RD access time）	T_{Acc8}	CL = 100pF	—	—	140	ns
控制L脉冲宽度（WR）(Control L pulse width (WR))	T_{CCLW}		100	—	—	ns
控制L脉冲宽度（RD）(（Control L pulse width (RD)）	T_{CCLR}		120	—	—	ns
控制H脉冲宽度（WR）(Control H pulse width (WR))	T_{CCHW}		100			ns
控制H脉冲宽度（RD）(Control H pulse width (RD))	T_{CCHR}		100			ns
上升时间（Rise time）	T_R				15	ns
下降时间（Fall time）	T_F				15	ns

*（VDD =2.4V～3.5V, Ta = 25℃）

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

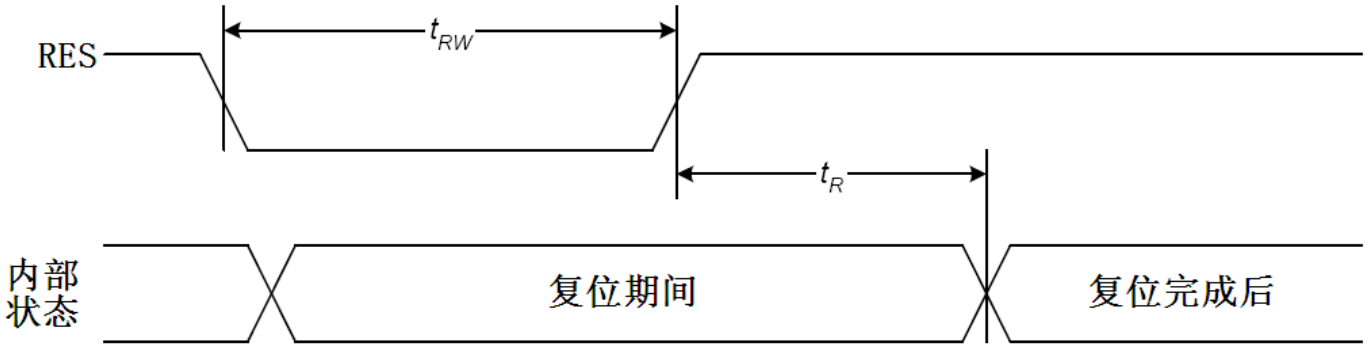


图 6. **电源启动后复位的时序**

表 7：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		500	—	—	ms
复位保持低电平的时间	t_{RW}	引脚：RES	500	—	—	ms

7．指令功能：

7.1 指令表

表 8

指令名称	指令码									说明
	RS	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(1)列地址低4位设置	0	0	0	0	0	列地址的低 4 位				高 4 位与低 4 位共同组成列地址，指定 128 列中的其中一列。比如 OLED 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
(2)列地址高4位设置		0	0	0	1	列地址的高 4 位				
(3)设定升压峰值（Set Pump voltage value）	0	0	0	1	1	0	0	A1	A0	A1 A0 升压峰值 指令 0 0 6.4V 0x30 0 1 7.4V 0x31 1 0 8.0V 0x32 1 1 9.0V 0x33
(4)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 OLED 屏一般设置为 0x40
(5) 设置对比度	0	1	0	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：0x00～0xFF,数值越大对比度越浓，越小越淡
(5) 设置对比度值	0	8 位电压值数据，0～255 共 256 级								
(6) 显示列地址增减(ADC select)	0	1	0	1	0	0	0	0	ADC 0 1	显示列地址增减： 0xA0：常规：列地址从右到左， 0xA1：反转：列地址从左到右
(7) 设置常规/打开全部点阵(Set Entire Display OFF/ON)	0	1	0	1	0	0	1	0	D 0 1	设置常规显示/打开全部点阵 0xA4：常规显示，写什么内容显示什么 0xA5：全部点阵点亮,之前的显示会被覆盖
(8)显示正显/反显(Display normal/reverse)	0	1	0	1	0	0	1	1	D 0 1	显示正显/反显: 0xA6：常规：正显 0xA7：反显
(9)设置显示行数（Set Multiplex Ration）	0	1	0	1	0	1	0	0	0	0xA8：设置显示行数
	0	*	*	共 6 位，0～63 共 64 级						设置范围：00～3f 针对本型号为 0x3f，64 行
(10) 设置内部/外部升压Set DC-DC OFF/ON: (Double Bytes Command)	0	1	0	1	0	1	1	0	1	0x8d：设置内部/外部升压
	0	1	0	0	0	1	0	1	D 0 1	0x8A：使用外部升压 0x8B：使用内部升压
(11)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
(12)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：0XB0～0XB7分别对应第一页到第八页。
(13) 行扫描顺序选择(Common output mode	0	1	1	0	0	D 0	0	0	0	行扫描顺序选择： 0XC0:普通扫描顺序：从下到上

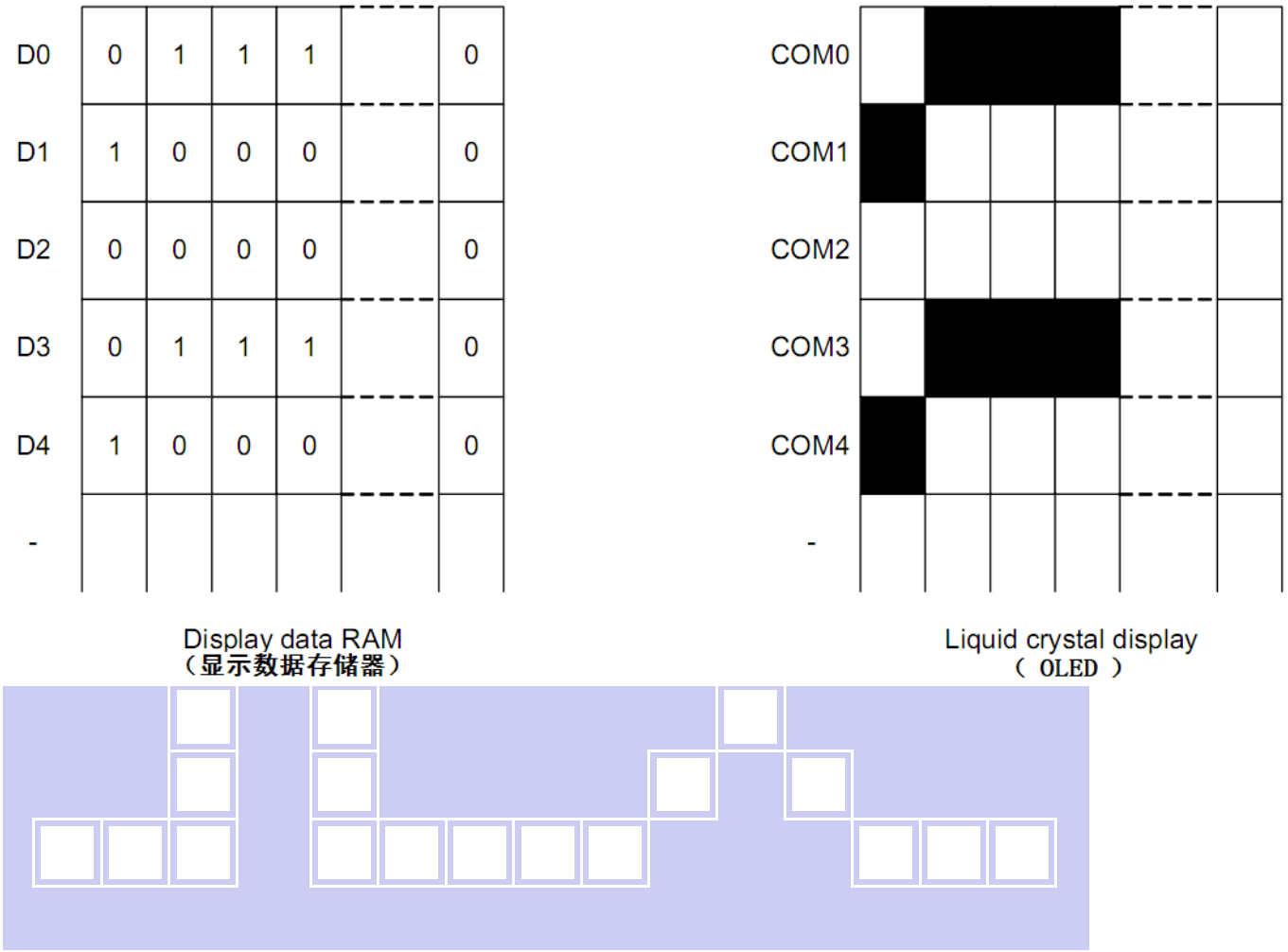
select)						1				0XC8:反转扫描顺序：从上到下
(14) 设置显示行偏移（Set Display Offset: (Double Bytes Command)）	0	1	1	0	1	0	0	1	1	0XD3:设置显示行偏移
	0	*	*	共 6 位，0～63 共 64 级						0X00:默认，范围：00—3f
(15) OLED 振荡频率设置 （Oscillator Frequency(Double Bytes Command)）	0	1	1	0	1	0	1	0	1	0Xd5:振荡频率设置
	0	共 8 位，0～255 共 256 级								0X80:默认值，范围：00—ff
(16)设置预充电周期（Set Dis-charge/Pre-charge Period: (Double Bytes Command)）	0	1	1	0	1	1	0	0	1	预充电周期模式设置： 0XD9:
	0	共 8 位，0～255 共 256 级								设置预充电时间： 0X1f:默认值，范围：00—ff
(17)设置COM硬件配置（Set Common pads hardware configuration: (Double Bytes Command)）	0	1	1	0	1	1	0	1	0	设置COM硬件配置 0XDA:
	0	0	0	0	D 0 1	0	0	1	0	设置COM配置模式 0X02: 0X12:本款型号配置
(18)设置VCOM（Set VCOM Deselect Level: (Double Bytes Command)）	0	1	1	0	1	1	0	1	1	设置VCOM 0XDB:
	0	共 8 位，0～255 共 256 级								0X40: 本款型号配置 范围：00—ff
(19)读-改-写（Read-Modify-Write）	0	1	1	1	0	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28页
(20)退出上述“读-改-写”(End)	0	1	1	1	0	1	1	1	0	0XEE :上述“读-改-写”指令结束 详情请参考IC资料第28页
(21)空指令（ NOP）	0	1	1	1	0	0	0	1	1	0XE3:空操作
(22) 写显示（Write Display Data）	1	8 位显示数据								从CPU写数据到OLED屏，每一位对应一个点阵，1个字节对应8个竖置的点阵
(23) 读状态 (Status read)	0	BUSY	ON/OFF	*	*	*	0	0	0	暂不可用
(24) 读OLED屏的显示数据 (Read Display Data)	1	8 位显示数据								并口时：读已经显示到OLED屏上的点阵数据。

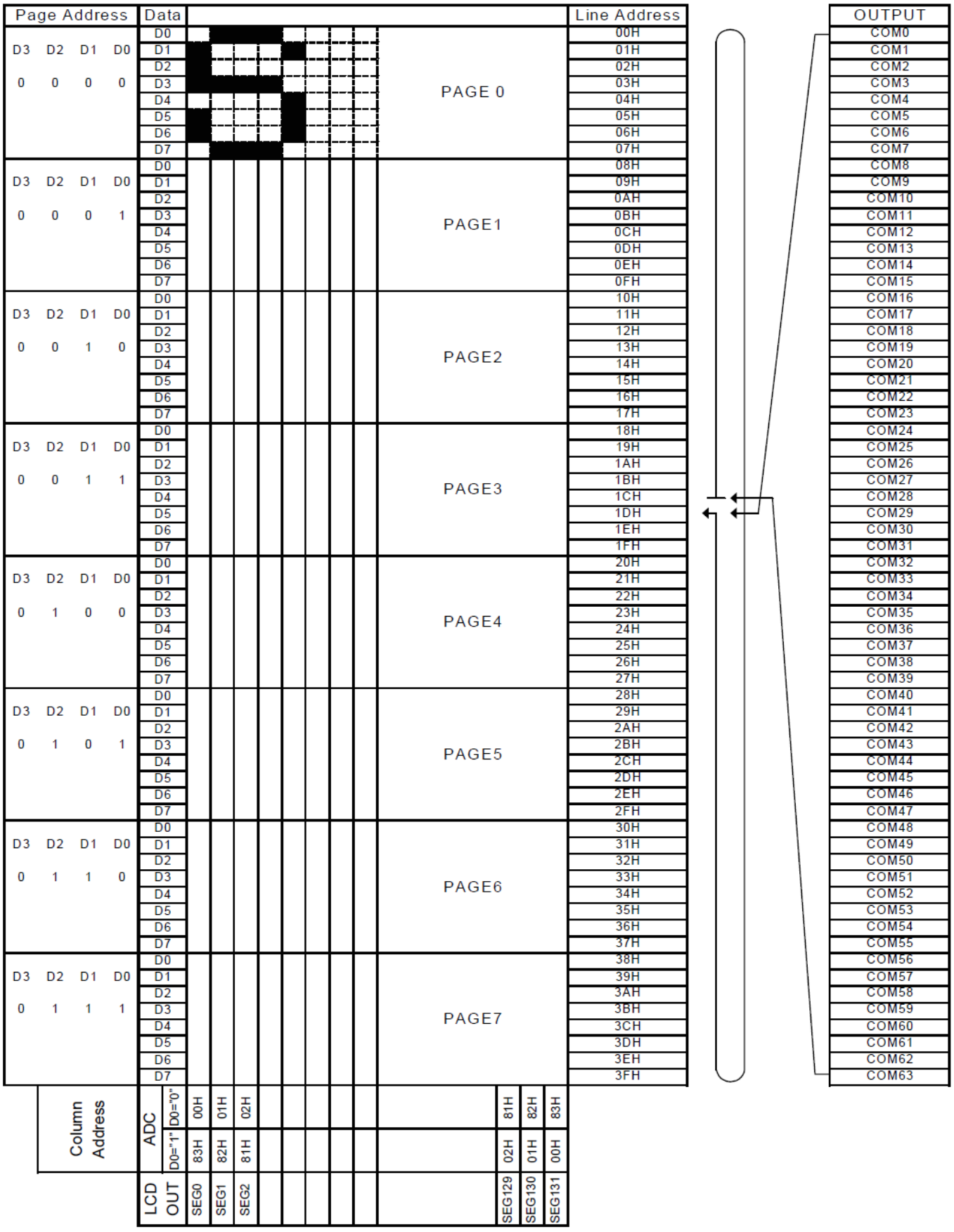
请详细参考 IC 资料”SH1106.PDF”。

7.2 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”。

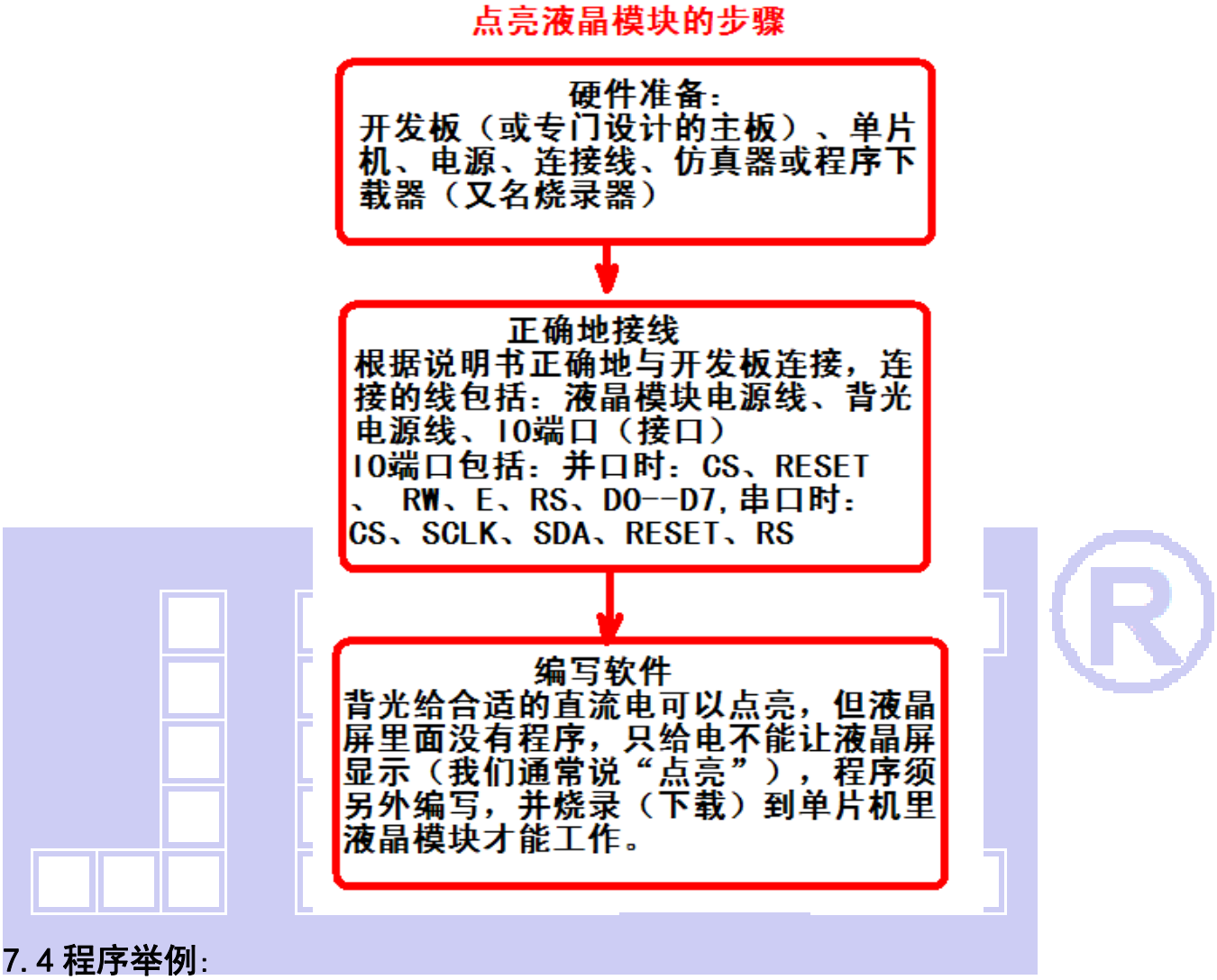
DB7--DB0 的排列方向：数据是从上向下排列的。最低位 D0 是在最上面，最高位 D7 是在最下面。每一位（bit）数据对应一个点阵，通常“1”代表点亮该点阵，“0”代表关掉该点阵. 如下图所示：




Page Address
Data
Line Address
OUTPUT
D3 D2 D1 D0
0 0 0 0
PAGE 0
0 0 0 1
PAGE1
0 0 1 0
PAGE2
0 0 1 1
PAGE3
0 1 0 0
PAGE4
0 1 0 1
PAGE5
0 1 1 0
PAGE6
0 1 1 1
PAGE7
D0
D1
D2
D3
D4
D5
D6
D7
00H 01H 02H 03H 04H 05H 06H 07H
08H 09H 0AH 0BH 0CH 0DH 0EH 0FH
10H 11H 12H 13H 14H 15H 16H 17H
18H 19H 1AH 1BH 1CH 1DH 1EH 1FH
20H 21H 22H 23H 24H 25H 26H 27H
28H 29H 2AH 2BH 2CH 2DH 2EH 2FH
30H 31H 32H 33H 34H 35H 36H 37H
38H 39H 3AH 3BH 3CH 3DH 3EH 3FH
COM0 COM1 COM2 COM3 COM4 COM5 COM6 COM7
COM8 COM9 COM10 COM11 COM12 COM13 COM14 COM15
COM16 COM17 COM18 COM19 COM20 COM21 COM22 COM23
COM24 COM25 COM26 COM27 COM28 COM29 COM30 COM31
COM32 COM33 COM34 COM35 COM36 COM37 COM38 COM39
COM40 COM41 COM42 COM43 COM44 COM45 COM46 COM47
COM48 COM49 COM50 COM51 COM52 COM53 COM54 COM55
COM56 COM57 COM58 COM59 COM60 COM61 COM62 COM63
Column Address
ADC
D0="0"
D0="1"
00H 01H 02H
83H 82H 81H
81H 82H 83H
02H 01H 00H
LCD OUT
SEG0 SEG1 SEG2
SEG129 SEG130 SEG131

7.3 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



7.4 程序举例:

OLED 模块与 MPU(以 8051 系列单片机为例)接口图如下:

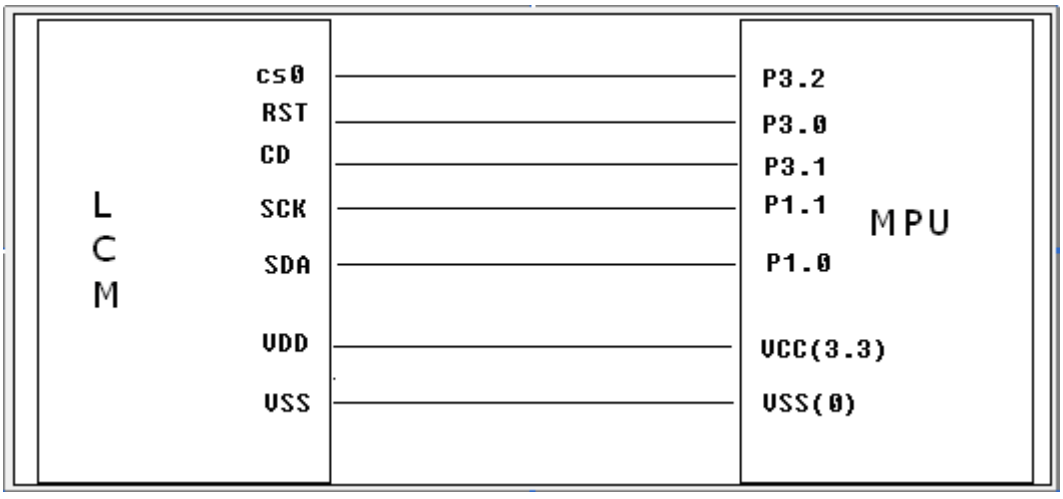
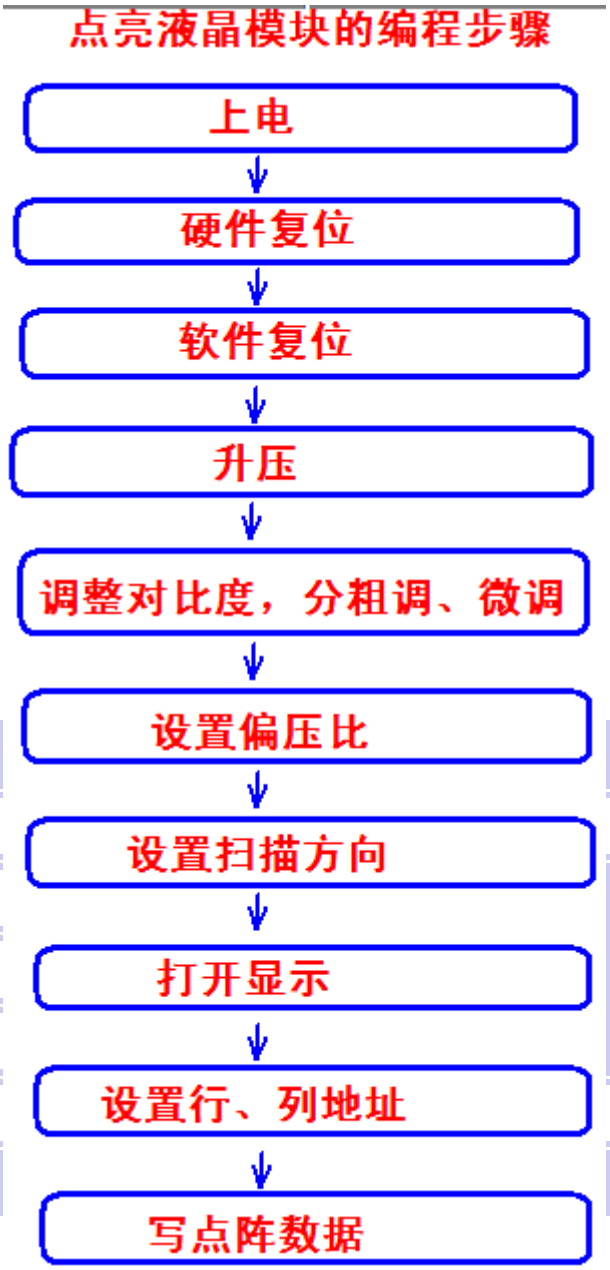
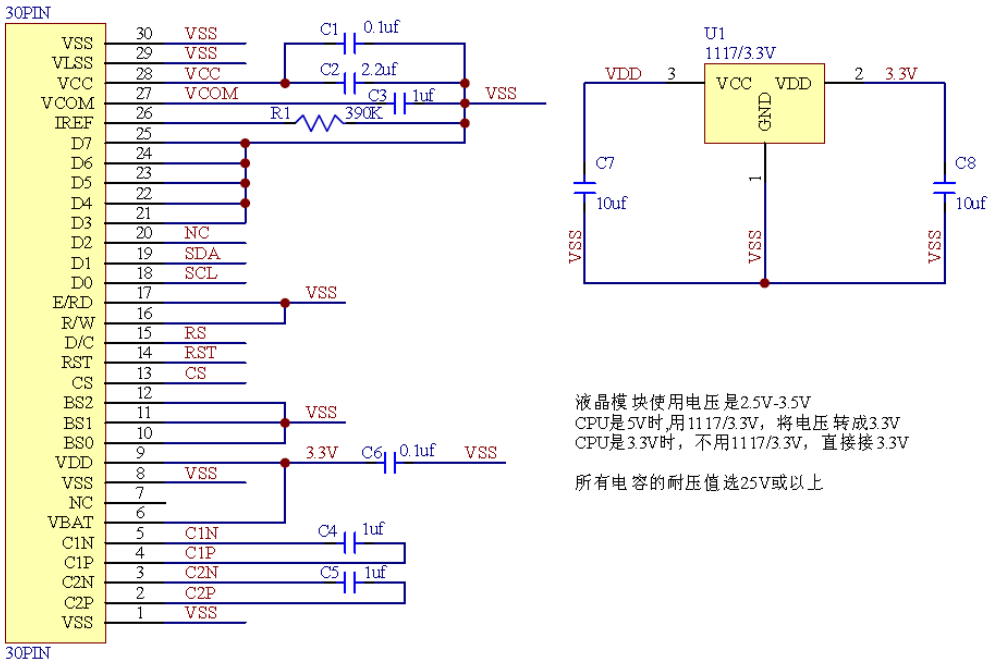


图 5.串行接口

7.4.1 程序:


点亮液晶模块的编程步骤
上电
硬件复位
软件复位
升压
调整对比度，分粗调、微调
设置偏压比
设置扫描方向
打开显示
设置行、列地址
写点阵数据




```
//    OLED 演示程序
//    OLED 模块型号：JLX12864OLED-13015，串行接口！
//    驱动 IC 是:SH1106
//    资料（源程序、驱动手册、使用说明书等）销售统一发
#include <reg52.H>
//=============================================================================
sbit lcd_sclk =P1^1;  //接口定义:lcd_sclk 就是 LCD 的 SCLK    //SCLK 接到“D0”脚
sbit lcd_sda  =P1^0;   //接口定义:lcd_sda 就是 LCD 的 SDA    //SDIN 接到“D1”脚
sbit lcd_reset=P3^0;  //接口定义:lcd_reset 就是 LCD 的 RESET
sbit lcd_dc   =P3^1;   //接口定义:lcd_dc 就是 LCD 的 D/C(RS)
sbit lcd_cs1=P3^2;    //接口定义:lcd_cs1 就是 LCD 的 CS
sbit key=P2^0;         //定义一个按键：P2.0 口与 GND 之间接一个按键
//=============================================================================
#define uchar unsigned char
#define uint unsigned int
#define ulong unsigned long
#include <ASCII_CODE_8X16_5X8_VERTICAL.H>
#include <Chinese_And_Graphic.H>
//延时
void delay_ms(int i)
{
      int j,k;
      for(j=0;j<i;j++)
            for(k=0;k<110;k++);
}

//等待按键：P2.0 口与 GND 之间接一个按键
void waitkey()
{
```

```
repeat:    if(key==1) goto repeat;
           else  delay_ms (1500);
}
//写指令到 OLED 显示模块
void transfer_command(int data1)
{
    uchar i;
    lcd_cs1=0;
    lcd_dc= 0;
      for(i=0;i<8;i++)
      {
        lcd_sclk = 0;
        if (data1 & 0x80)   lcd_sda = 1;
        else                      lcd_sda = 0;
        lcd_sclk = 1;
        data1 <<= 1;
    }
    lcd_cs1=1;
}

//写数据到 OLED 显示模块
void transfer_data(int data1)
{
    uchar i;
    lcd_cs1=0;
    lcd_dc= 1;
      for(i=0;i<8;i++)
      {
        lcd_sclk = 0;
        if (data1 & 0x80)   lcd_sda = 1;
        else                      lcd_sda = 0;
        lcd_sclk = 1;
        data1 <<= 1;
    }
    lcd_cs1=1;
}
//OLED 显示模块初始化
void initial_lcd()
{
      lcd_reset=0;          //低电平复位
      delay_ms (500);
      lcd_reset=1;                //复位完毕
      delay_ms (200);

      transfer_command(0xAE);    /*display off*/
      transfer_command(0x02);    /*set lower column address*/
      transfer_command(0x10);    /*set higher column address*/
```

```
    transfer_command(0x40);    /*set display start line*/
    transfer_command(0xB0);    /*set page address*/
    transfer_command(0x81);    /*contrast control*/
    transfer_command(0xff);
    transfer_command(0xA1);    /*set segment remap*/
    transfer_command(0xA6);    /*normal / reverse*/
    transfer_command(0xA8);    /*multiplex ratio*/
    transfer_command(0x3F);    /*duty = 1/64*/
    transfer_command(0xad);    /*set charge pump enable*/
    transfer_command(0x8b);     /*    0x8B    内供 VCC   */
    transfer_command(0x33);    /*0X30---0X33  set VPP   9V */
    transfer_command(0xC8);    /*Com scan direction*/
    transfer_command(0xD3);    /*set display offset*/
    transfer_command(0x00);   /*   0x20  */
    transfer_command(0xD5);    /*set osc division*/
    transfer_command(0x80);
    transfer_command(0xD9);    /*set pre-charge period*/
    transfer_command(0x1f);    /*0x22*/
    transfer_command(0xDA);    /*set COM pins*/
    transfer_command(0x12);
    transfer_command(0xdb);    /*set vcomh*/
    transfer_command(0x40);
    transfer_command(0xAF);    /*display ON*/
}

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(j=0;j<8;j++)
    {
        lcd_address(1+j,1);
        for(i=0;i<128;i++)
        {
            transfer_data(0x00);
        }
    }
}
```

```
//显示 128x64 点阵图像
void display_128x64(uchar *dp)
{
    uint i,j;
    for(j=0;j<8;j++)
    {
        lcd_address(j+1,1);
        for (i=0;i<128;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 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++;
        }
    }
}
//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(uint page,uint column,uchar *text)
{
    uint i=0,j,k,n;
    if(column>123)
    {
        column=1;
        page+=2;
    }
    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]);  //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
                }
            }
            i++;
            column+=8;
        }
        else
        i++;
    }
}
//显示 5x8 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_5x8(uint page,uint column,uchar reverse,uchar *text)
{
    uint i=0,j,k,disp_data;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
```

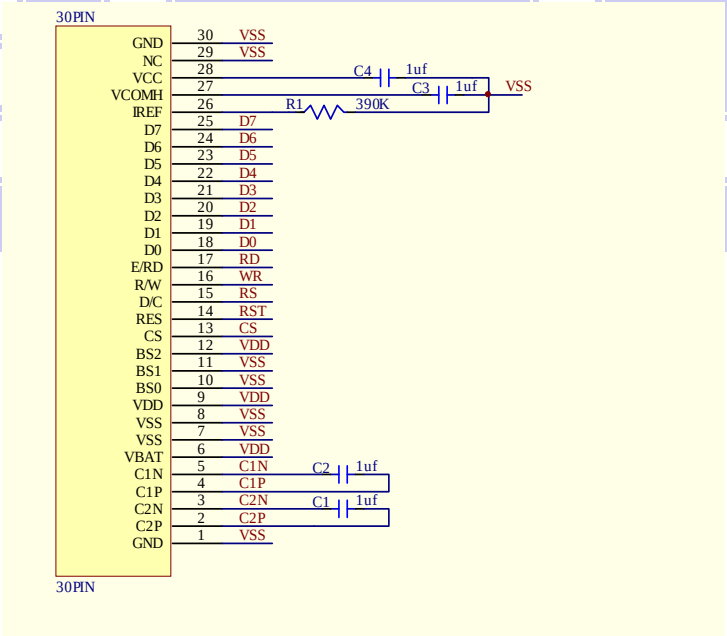
```
                lcd_address(page,column);
                for(k=0;k<5;k++)
                {
                    if(reverse==1)
                    {
                        disp_data=~ascii_table_5x8[j][k];
                    }
                    else
                    {
                        disp_data=ascii_table_5x8[j][k];
                    }
                    transfer_data(disp_data);   //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
                }
                if(reverse==1)   transfer_data(0xff);  //写入一列空白列，使得 5x8 的字符与字符之间有一列间隔，更美观
                else  transfer_data(0x00);             //写入一列空白列，使得 5x8 的字符与字符之间有一列间隔，更美观
                i++;
                column+=6;
                if(column>123)
                {
                    column=1;
                    page++;
                }
            }
            else
            i++;
        }
}
//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 *text)
{
    uchar i,j,k;
    uint address;
    j = 0;
    while(text[j] != '\0')
    {
        i = 0;
        address = 1;
        while(Chinese_text_16x16[i] > 0x7e)    // >0x7f 即说明不是 ASCII 码字符
        {
            if(Chinese_text_16x16[i] == text[j])
            {
                if(Chinese_text_16x16[i + 1] == text[j + 1])
                {
                    address = i * 16;
                    break;
                }
```

```
            }
            i += 2;
        }

        if(column > 113)
        {
            column = 0;
            page += 2;
        }

        if(address != 1)// 显示汉字
        {

                for(k=0;k<2;k++)
                {
                     lcd_address(page+k,column);
                 for(i = 0; i < 16; i++)
                 {
                     transfer_data(Chinese_code_16x16[address]);
                     address++;
                 }
                }
            j += 2;
        }
        else          //显示空白字符
        {
                for(k=0;k<2;k++)
                {
                     lcd_address(page+k,column);
                 for(i = 0; i < 16; i++)
                 {
                     transfer_data(0x00);
                 }
                }

            j++;
        }

        column+=16;
    }
}

//显示 16x16 点阵的汉字或者 ASCII 码 8x16 点阵的字符混合字符串
//括号里的参数：(页，列，字符串)
void disp_string_8x16_16x16(uchar page,uchar column,uchar *text)
{
    uchar temp[3];
```

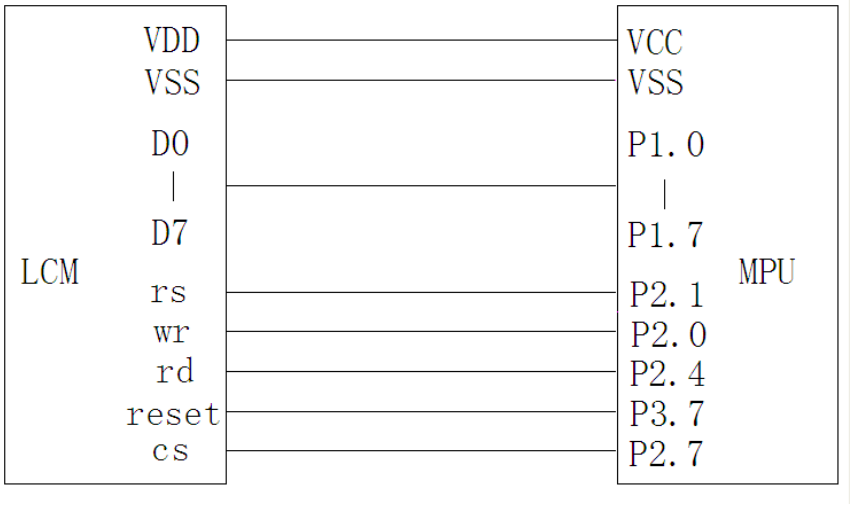
```
    uchar i = 0;

    while(text[i] != '\0')
    {
        if(text[i] > 0x7e)
        {
            temp[0] = text[i];
            temp[1] = text[i + 1];
            temp[2] = '\0';          //汉字为两个字节
            display_string_16x16(page,column,temp);  //显示汉字
            column += 16;
            i += 2;
        }
        else
        {
            temp[0] = text[i];
            temp[1] = '\0';          //字母占一个字节
            display_string_8x16(page, column, temp);  //显示字母
            column += 8;
            i++;
        }
    }
}

void main(void)
{
    while(1)
    {
        initial_lcd();                          //初始化
        clear_screen();                         //清屏

//演示 32x32 点阵的汉字，16x16 点阵的汉字，8x16 点阵的字符，5x8 点阵的字符
        display_string_5x8(1,1,0,"{(5x8dot ASCII char)}");//显示字符串，括号里的参数分别为（PAGE,列,字符串指针）
        display_string_5x8(2,1,0,"{[(<~!@#$%^&*_+=?>)]}");
        disp_string_8x16_16x16(3,1,"标准 16x16dot 汉字");      //显示 16x16 点阵汉字串或 8x16 点阵的字符串，括号里的参数分别为（页,列,
字符串指针）
        display_graphic_32x32 (5,1+32*0,jing1);                 //显示单个 32x32 点阵的汉字，括号里的参数分别为（PAGE,列,字
符指针）
        display_graphic_32x32 (5,1+32*1,lian1);
        display_graphic_32x32 (5,1+32*2,xun1);
        disp_string_8x16_16x16(5,1+32*3,"JLX:");
        disp_string_8x16_16x16(7,1+32*3,"OLED");
        waitkey();

//演示显示一页纯英文的 5x8 点阵的菜单界面
        clear_screen();                                         //clear all dots
        display_string_5x8(1,1,1,"012345678901234567890");
```

```
        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();                                          //clear all dots
        display_128x64(bmp1);
        waitkey();
        clear_screen();                                          //clear all dots
        display_128x64(bmp2);
        waitkey();
    }
}
```

7.4.2 并行接口：



OLED 模块与 MPU(以 8051 系列单片机为例)接口图如下：



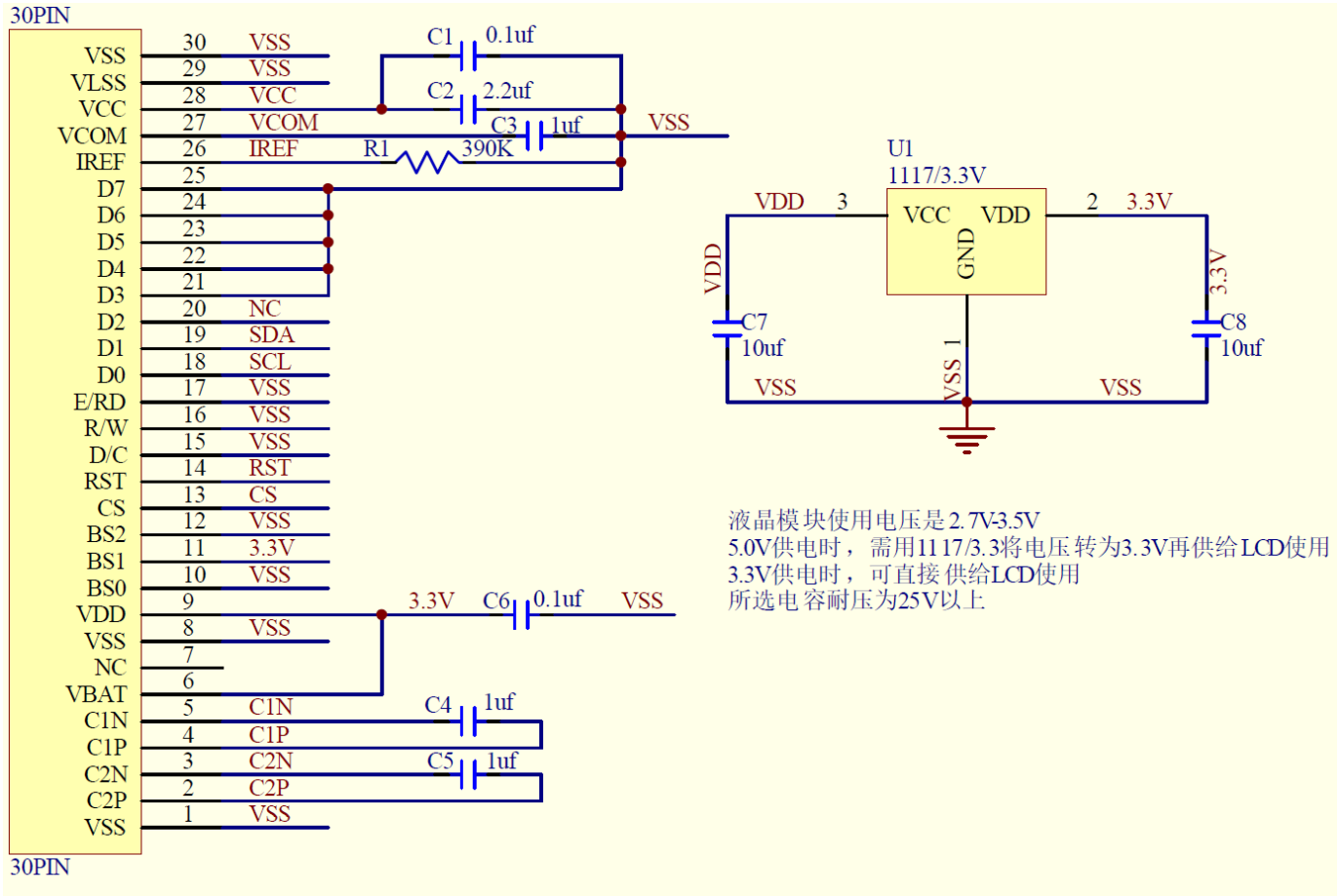
并行接口

与串行方式相比较，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：

并行程序：6800 时序

```
sbit lcd_rs=P2^1;     /*接口定义:lcd_rs 就是 OLED 的 rs*/
sbit lcd_rd=P2^4;     /*接口定义:lcd_e 就是 OLED 的 rd*/
sbit lcd_wr=P2^0;     /*接口定义:lcd_rw 就是 OLED 的 wr*/
sbit lcd_reset=P3^7; /*接口定义:lcd_reset 就是 OLED 的 reset*/
sbit lcd_cs1=P2^7;    /*接口定义:lcd_cs1 就是 OLED 的 cs1*/

//写指令到 OLED 模块
void transfer_command_lcd(int data1)
{
    lcd_cs1=0;
    lcd_rs=0;
    lcd_rd=0;
    lcd_wr=0;
    P1=data1;
    lcd_rd=1;
    lcd_cs1=1;
    lcd_rd=0;
}

//写数据到 OLED 模块
void transfer_data_lcd(int data1)
{
    lcd_cs1=0;
    lcd_rs=1;
    lcd_rd=0;
    lcd_wr=0;
    P1=data1;
    lcd_rd=1;
    lcd_cs1=1;
    lcd_rd=0;
}
```

7.4.3 I2C 接口：



OLED 模块与 MPU(以 8051 系列单片机为例)接口图如下：

I2C 接口

与串行方式相比较，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：

```
//=============================================

sbit lcd_scl  =P3^2;  //接口定义:lcd_sclk 就是 OLED 的 SCL

sbit lcd_sda  =P3^1;   //接口定义:lcd_sda 就是 OLED 的 SDA

void  start_flag()

{

     lcd_scl=1;

     delay_us(1);

     lcd_sda=1;
```

```
      delay_us(1);
      lcd_sda=0;
      delay_us(1);
      lcd_scl=0;
      delay_us(1);
}
void  stop_flag()
{
      lcd_scl=0;
      delay_us(1);
      lcd_sda=0;
      delay_us(1);
      lcd_sda=1;
      delay_us(1);
      lcd_scl=1;
      delay_us(1);
}
//传 8 位指令或数据到 OLED 显示模块
void transfer(uchar data1)
{
   unsigned char j;
   for(j=0;j<8;j++)
   {
     lcd_scl=0;
     if(data1&0x80)    lcd_sda=1;
     else
         lcd_sda=0;
     lcd_scl=1;
     lcd_scl=0;
     data1<<=1;
      delay_us(1);
    }
    lcd_sda=0;
    lcd_scl=0;
    lcd_scl=1;
}
//写指令到 OLED 显示模块
void transfer_command(uchar com)
{
      start_flag();
      transfer(0x78);
      transfer(0x00);
      transfer(com);
      stop_flag();
}
//写数据到 OLED 显示模块
void transfer_data(uchar dat)
```

```
{
    start_flag();
    transfer(0x78);
    transfer(0x40);
    transfer(dat);
    stop_flag();
}
```

-END-