JLX12864OLED-242 中文使用说明书

目 录

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 JLX12864OLED-242 型液晶模块由于使用方便、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-242 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-242 图像型点阵液晶模块的特性

2.1 结构牢：焊接式 FPC。
2.2 IC 采用 SSD1309,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℃～+70℃;
2.8 储存温度宽：-40℃～+80℃;
2.9 底色可选：蓝色、白色。

3. 外形尺寸及接口引脚功能

3.1 外形图

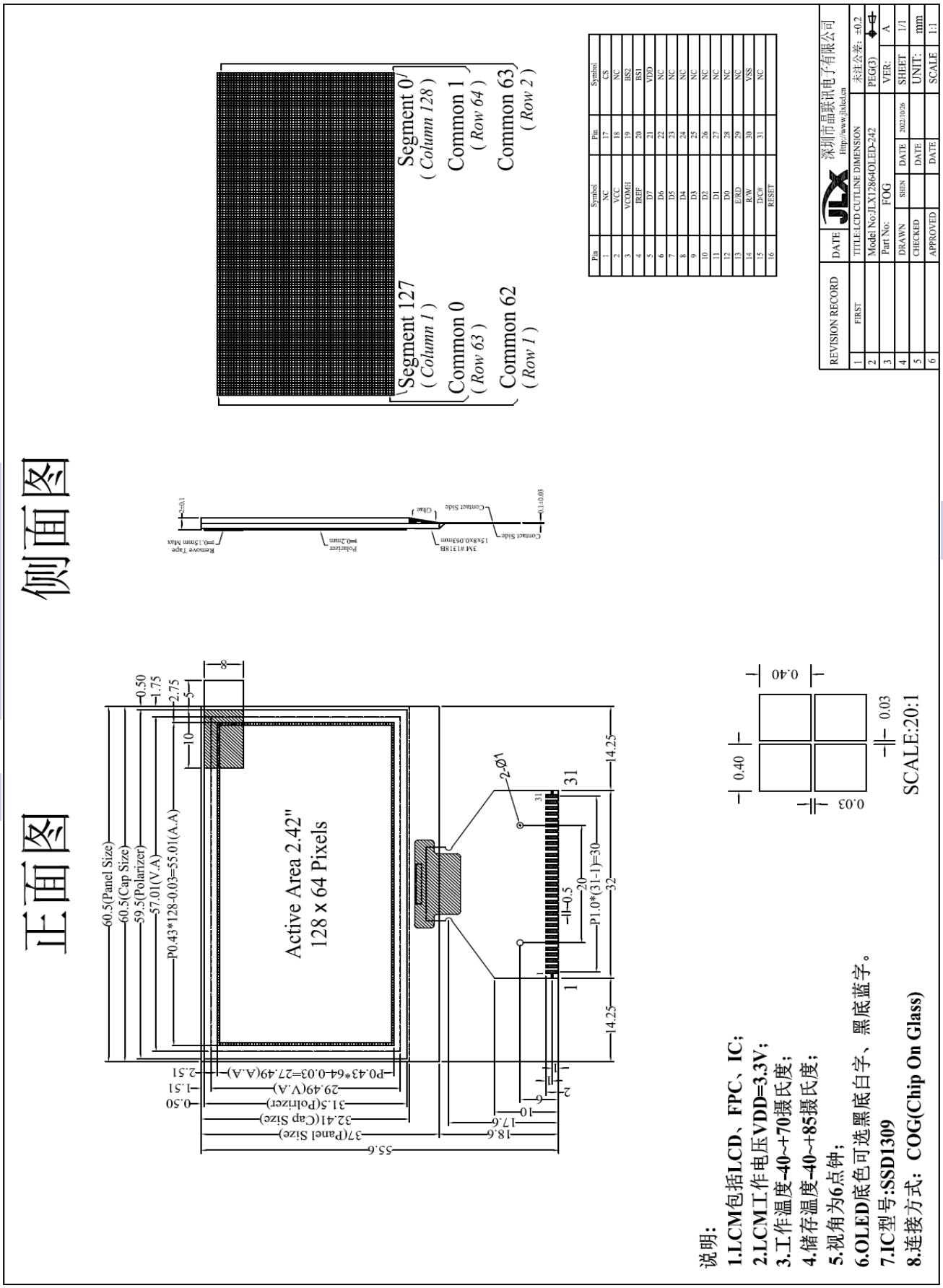
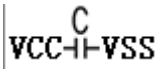
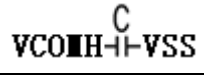
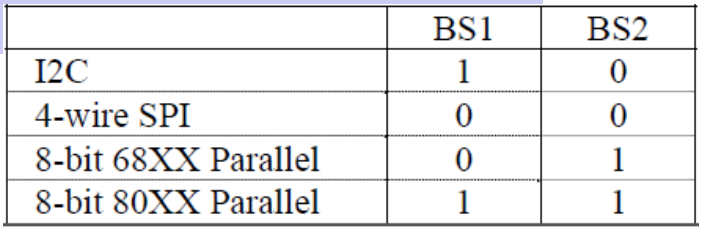


图 1. 液晶模块外形尺寸

模块的接口引脚功能

引 线 号	符 号	名 称	功 能
1	NC	NC	空脚
2	VCC	面板电源	 VCC 为 OLED 驱动电压接口，需外接+8—12V
3	VCOMH	VCOMH	
4	IREF	升压电容	接电阻调整模块亮度（即限电流）
5	D7	I/O	数据总线（串行接口时：空脚或接地）
6	D6	I/O	数据总线（串行接口时：空脚或接地）
7	D5	I/O	数据总线（串行接口时：空脚或接地）
8	D4	I/O	数据总线（串行接口时：空脚或接地）
9	D3	I/O	数据总线（串行接口时：空脚或接地）
10	D2	I/O	数据总线（串行接口时：空脚或接地，I2C 接口时：SDA）
11	D1	I/O	串行数据（SDA）
12	D0	I/O	串行时钟（SCL）
13	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 VSS 或悬空
14	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. 串行接口时：接 VSS 或悬空
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”） IIC 接口：接 VSS，从属地址 0x78
16	RESET	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
17	CS	片选	低电平片选
18	NC	NC	空脚
19	BS2	BS2	
20	BS1	BS1	
21	VDD	供电电源正极	供电电源正极 3.3V
22～29	NC	NC	空脚
30	VSS	供电电源负极	供电电源负极
31	NC	NC	空脚

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

表 1：模块的接口引脚功能

4. 基本原理

4.1 OLED

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 邦定在玻璃上（这种加工工艺叫 COG）.

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	3.3	4.0	V
OLED 驱动电压	VCC	8	12	17	V
静电电压		—	—	100	V
工作温度		-40		+70	℃
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	VDD		2.7	3.3	4.0	V
输入高电平	V_{IHC}		0.8xVDD	—	VDD	V
输入低电平	V_{ILC}		—	—	0.2xVDD	V
输出高电平	V_{OHC}	IOH = 0.2mA	0.9xVDD	—	VDD	V
输出低电平	V_{OHL}	IOO = 1.2mA	—	—	0.1xVDD	V
模块工作电流	I_{DD}	VDD = 3.3V	0.3	150	300	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 SSD1309（Writing Data from CPU to SSD1309）

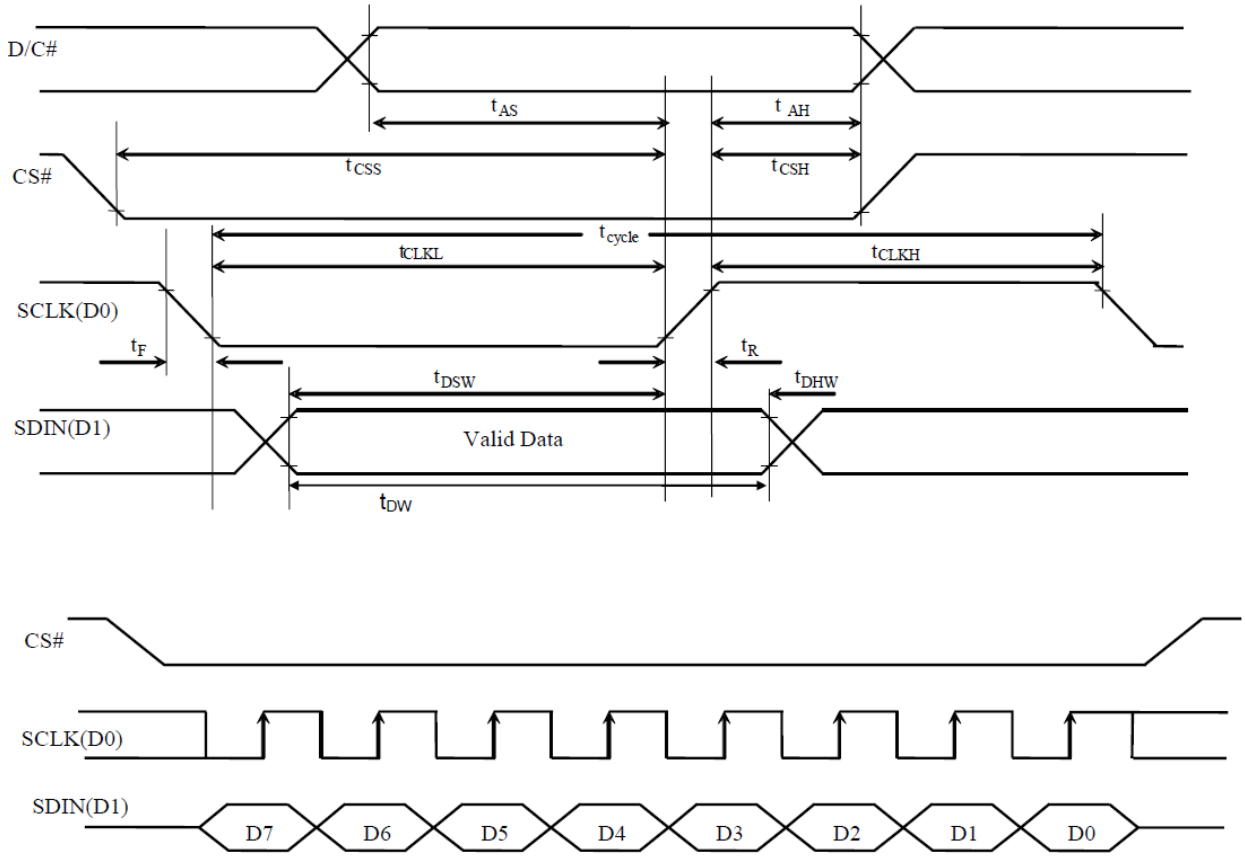


图 2. 从 CPU 写到 SSD1309（Writing Data from CPU to SSD1309）

6.1.2 串行接口：时序要求（AC 参数）：

写数据到 SSD1309 的时序要求： **表 4.**

Symbol	Parameter	Min	Typ	Max	Unit
t_{cycle}	Clock Cycle Time	100	-	-	ns
t_{AS}	Address Setup Time	15	-	-	ns
t_{AH}	Address Hold Time	15	-	-	ns
t_{CSS}	Chip Select Setup Time	20	-	-	ns
t_{CSH}	Chip Select Hold Time	50	-	-	ns
t_{DW}	Data Write Time	55	-	-	ns
t_{DSW}	Write Data Setup Time	15	-	-	ns
t_{DHW}	Write Data Hold Time	15	-	-	ns
t_{CLKL}	Clock Low Time	50	-	-	ns
t_{CLKH}	Clock High Time	50	-	-	ns
t_R	Rise Time	-	-	40	ns
t_F	Fall Time	-	-	40	ns

*（VDD =1.65V～3.3V, Ta = 25℃）

6.2.1 6800 并行接口：

从 CPU 写到 SSD1309（Writing Data from CPU to SSD1309）

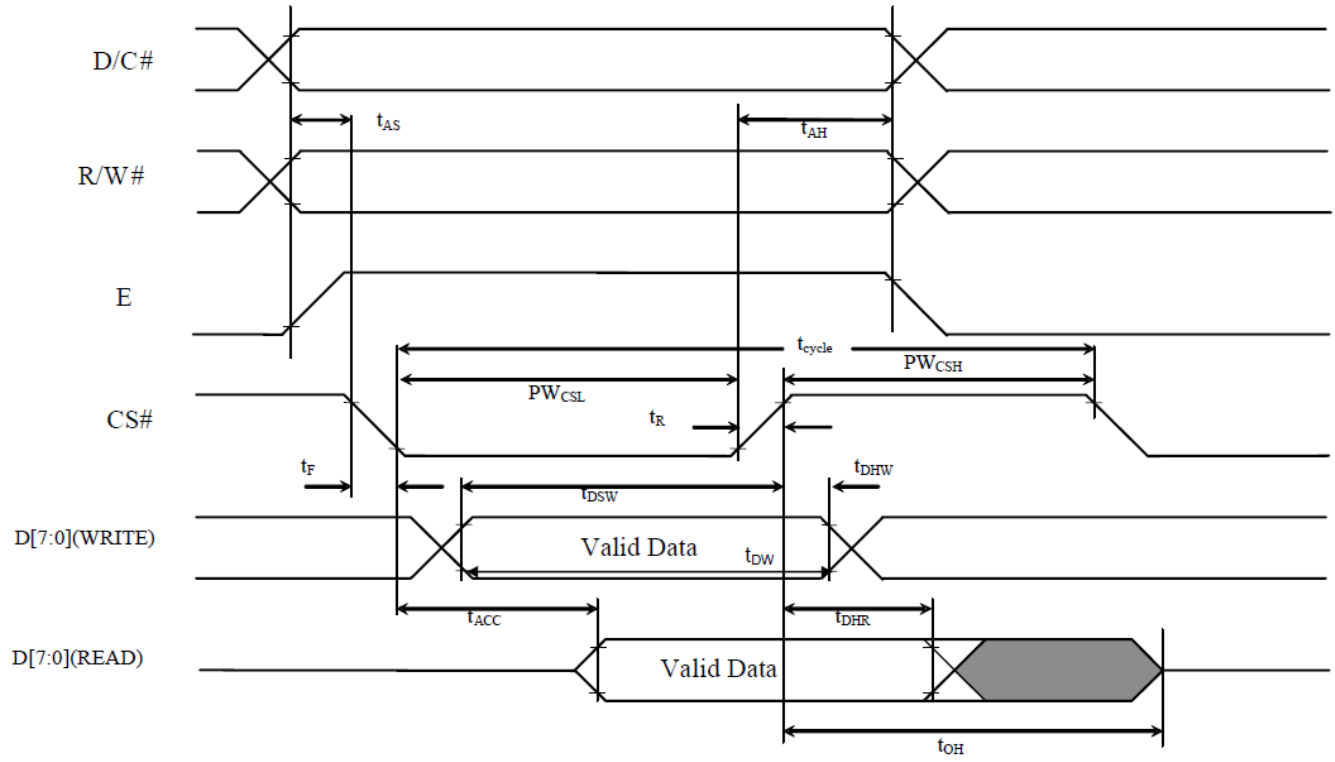


图 3. 从 CPU 写到 SSD1309（Writing Data from CPU to SSD1309）

6.2.2 6800 并行接口：时序要求（AC 参数）：

写数据到 SSD1309 的时序要求： 表 5.

Symbol	Parameter	Min	Typ	Max	Unit
t_{cycle}	Clock Cycle Time	300	-	-	ns
t_{AS}	Address Setup Time	20	-	-	ns
t_{AH}	Address Hold Time	0	-	-	ns
t_{DW}	Data Write Time	80	-	-	ns
t_{DSW}	Write Data Setup Time	40	-	-	ns
t_{DHW}	Write Data Hold Time	20	-	-	ns
t_{DHR}	Read Data Hold Time	20	-	-	ns
t_{OH}	Output Disable Time	-	-	70	ns
t_{ACC}	Access Time	-	-	140	ns
PW_{CSL}	Chip Select Low Pulse Width (read) Chip Select Low Pulse Width (write)	120 60	-	-	ns
PW_{CSH}	Chip Select High Pulse Width (read) Chip Select High Pulse Width (write)	60 60	-	-	ns
t_R	Rise Time	-	-	40	ns
t_F	Fall Time	-	-	40	ns

*（VDD =1.65V～3.3V, Ta = 25℃）

6.3.1 8080 并行接口：

从 CPU 写到 SSD1309（Writing Data from CPU to SSD1309）

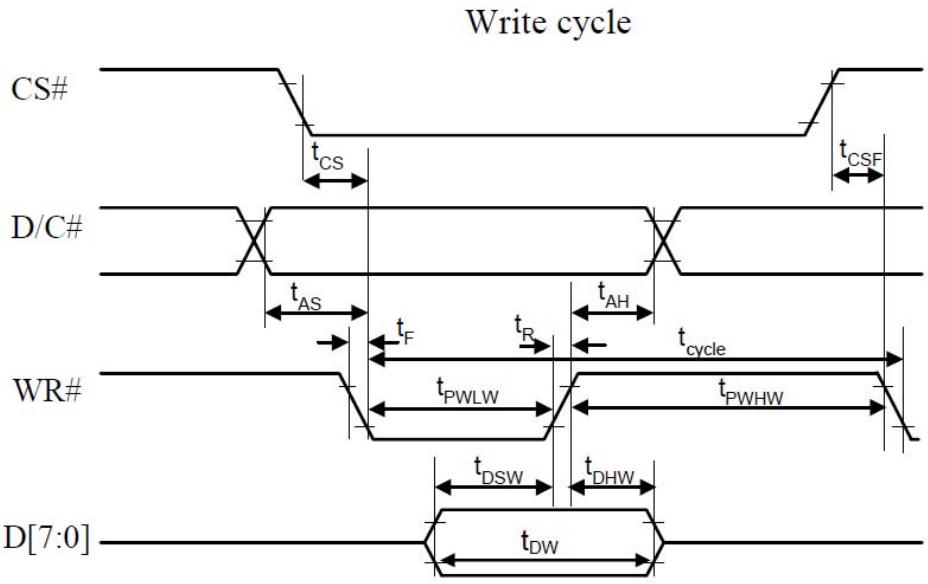


图 4. 从 CPU 写到 SSD1309（Writing Data from CPU to SSD1309）

6.3.2 8080 并行接口：时序要求（AC 参数）：

写数据到 SSD1309 的时序要求： 表 6.

Symbol	Parameter	Min	Typ	Max	Unit
t_{cycle}	Clock Cycle Time	300	-	-	ns
t_{AS}	Address Setup Time	20	-	-	ns
t_{AH}	Address Hold Time	0	-	-	ns
t_{DW}	Data Write Time	70	-	-	ns
t_{DSW}	Write Data Setup Time	40	-	-	ns
t_{DHW}	Write Data Hold Time	15	-	-	ns
t_{DHR}	Read Data Hold Time	20	-	-	ns
t_{OH}	Output Disable Time	-	-	70	ns
t_{ACC}	Access Time	-	-	140	ns
t_{PWLR}	Read Low Time	120	-	-	ns
t_{PWLW}	Write Low Time	60	-	-	ns
t_{PWHR}	Read High Time	60	-	-	ns
t_{PWHW}	Write High Time	60	-	-	ns
t_R	Rise Time	-	-	40	ns
t_F	Fall Time	-	-	40	ns
t_{CS}	Chip select setup time	0	-	-	ns
t_{CSH}	Chip select hold time to read signal	0	-	-	ns
t_{CSF}	Chip select hold time	20	-	-	ns

*（VDD =1.65V～3.3V, Ta = 25℃）

6.4.1 I2C 接口：

从 CPU 写到 SSD1309（Writing Data from CPU to SSD1309）

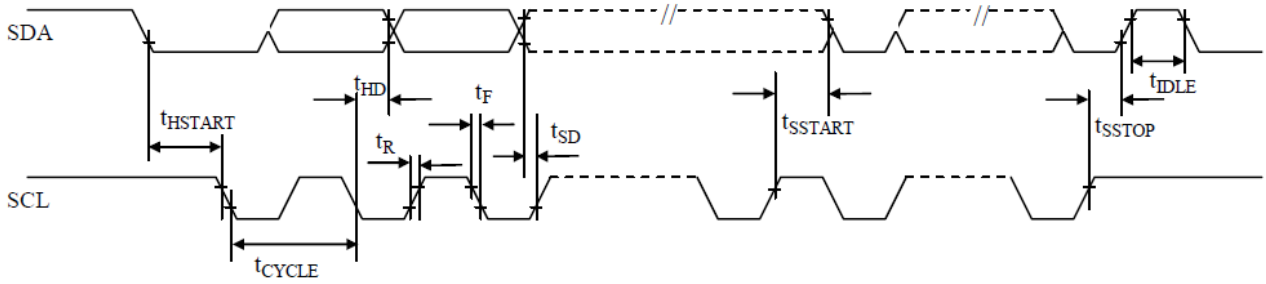


图 5. 从 CPU 写到 SSD1309（Writing Data from CPU to SSD1309）

6.4.2 I2C 接口：时序要求（AC 参数）：

写数据到 SSD1309 的时序要求： **表 7.**

Symbol	Parameter	Min	Typ	Max	Unit
t_{cycle}	Clock Cycle Time	2.5	-	-	us
t_{HSTART}	Start condition Hold Time	0.6	-	-	us
t_{HD}	Data Hold Time (for "SDA_{OUT}" pin)	0	-	-	ns
	Data Hold Time (for "SDA_{IN}" pin)	300	-	-	ns
t_{SD}	Data Setup Time	100	-	-	ns
t_{SSTART}	Start condition Setup Time (Only relevant for a repeated Start condition)	0.6	-	-	us
t_{SSTOP}	Stop condition Setup Time	0.6	-	-	us
t_R	Rise Time for data and clock pin	-	-	300	ns
t_F	Fall Time for data and clock pin	-	-	300	ns
t_{IDLE}	Idle Time before a new transmission can start	1.3	-	-	us

*（VDD =1.65V～3.3V, Ta = 25℃）

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

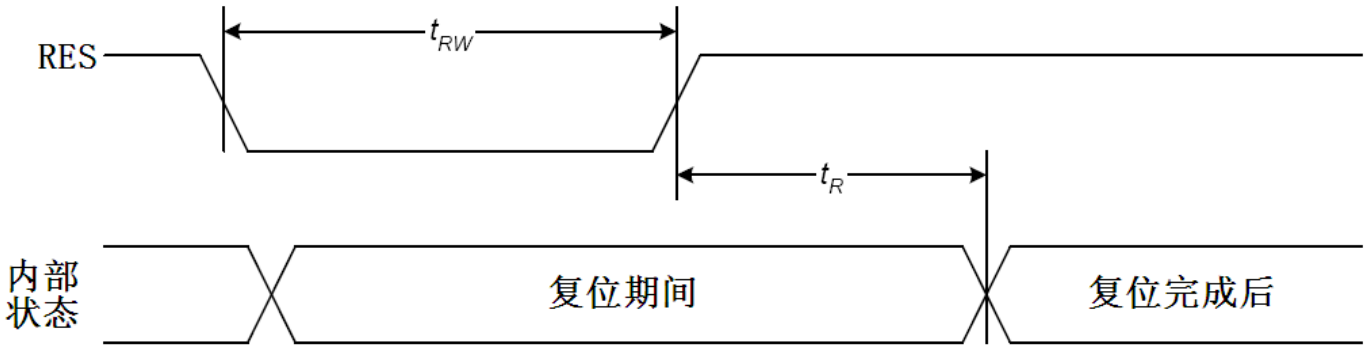


图 6：**电源启动后复位的时序**

表 8：电源启动后复位的时序要求

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

7．指令功能：

7.1 指令表

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)设定寻址方式（双指令） （Set Memory Addressing Mode）	0	0	0	1	0	0	0	0	0	**0x20**：设定寻址方式
	0	0	0	0	0	0	0	A1	A0	A1 A0 寻址方式 指令 0 0 水平 0x00 0 1 垂直 0x01 1 0 页 0x02 1 1 无效指令
(4)设置列地址起始-结束（Set Column Address）	0	0	0	1	0	0	0	0	1	**0x21**：设置列地址
	0	A7	A6	A5	A4	A3	A2	A1	A0	设置起始列地址；范围 0--127
	0	B7	B6	B5	B4	B3	B2	B1	B0	设置结束列地址；范围 0--127
(5) 设置页地址起始-结束（Set Page Address）	0	0	0	1	0	0	0	1	0	**0x22**：设置页地址
	0	0	0	0	0	0	A2	A1	A0	设置起始页地址；范围 0--7
	0	0	0	0	0	0	B2	B1	B0	设置结束页地址；范围 0--7
(6)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**
(7) 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
(7) 设置对比度值	0	8 位电压值数据，0～255 共 256 级								
(8) 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**：列地址从右到左， **0xA1**：列地址从左到右
(9) 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
(10)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
(11)设置显示行数	0	1	0	1	0	1	0	0	0	**0xA8**：设置显示行数

（Set Multiplex Ration）	0	*	*	共 6 位，0～63 共 64 级						**设置范围：00～3f** **针对本型号为 0x3f，64 行**
(12) 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
(13)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。
(14) 行扫描顺序选择 (Common output mode select)	0	1	1	0	0	D 0 1	0	0	**0**	行扫描顺序选择： **0XC0**:普通扫描顺序：从下到上 **0XC8**:反转扫描顺序：从上到下
(15) 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
(16) OLED 显示时钟/振荡频率设（Oscillator Frequency(Double Bytes Command)）	0	1	1	0	1	0	1	0	1	**0Xd5**:振荡频率设置
	0	A7	A6	A5	A4	A3	A2	A1	A0	**D3—D0：显示时钟分频** **D7—D4：显示频率** **详情请看 IC 资料第 43 页**
(17)设置预充电周（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	A7	A6	A5	A4	A3	A2	A1	A0	设置预充电时间： **0X82**:默认值，范围：00—ff
(18)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	A5	A4	0	0	1	0	设置COM配置模式 **0X02**:上下映射 **0X32**:左右映射 **0X12**:正常显示
(19)设置VCOM（Set V_{COMH} Deselect Level）	0	1	1	0	1	1	0	1	1	**0XDB**:设置VCOMH
	0	0	0	A5	A4	A3	A2	0	0	A5 A4 A3 A2 VCOMH 指令 0 0 0 0 0.64xV_{CC} 0x00 1 1 0 1 0.78xV_{CC} 0x34 1 1 1 1 0.84xV_{CC} 0x3C
(20)设置GPIO（Set GPIO）	0	1	1	0	1	1	1	0	0	**0XDC**:设置 GPIO
	0	0	0	0	0	0	0	A1	A0	A1 A0 指令 功能说明 0 0 0X00 输入禁用 0 1 0x01 输入启用 1 0 0X02 输出低 1 1 0x03 输出高
(21)空指令（ NOP）	0	1	1	1	0	0	0	1	1	**0XE3**:空操作
(22)设置指令锁（Set	0	1	1	1	1	1	1	0	1	**0XFD**: 设置指令锁

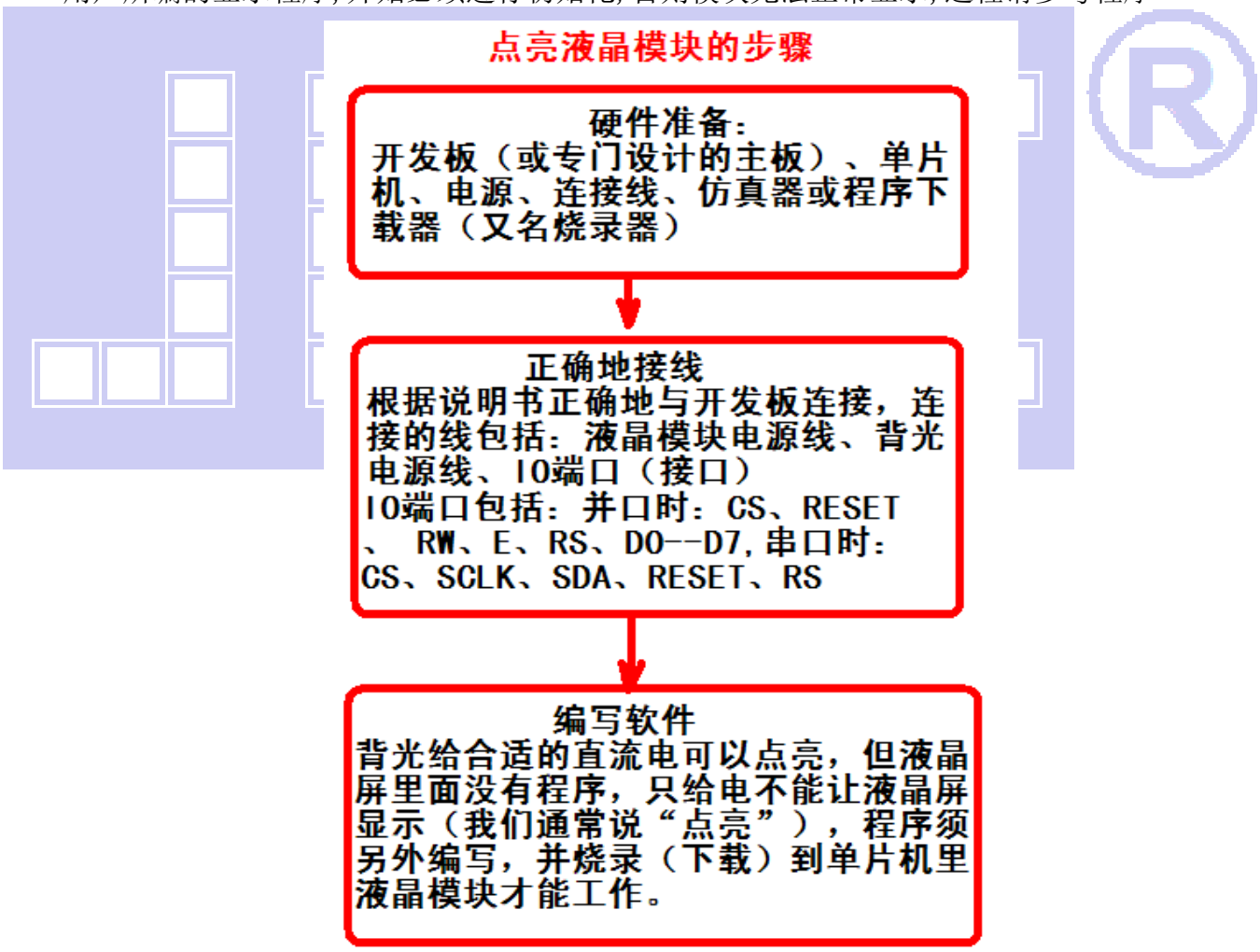
<table>
<tr><td>Command Lock）</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>A2</td><td>1</td><td>0</td><td>0X16:锁定
0X12:开锁</td></tr>
<tr><td rowspan="11">(23)水平滚动设置（Horizontal Scroll Setup）</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>X0</td><td>0x26：水平向右滚动
0x27：水平向左滚动</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0x00：虚拟指令</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>B2</td><td>B1</td><td>B0</td><td>设置起始滚动页地址，范围：00--07</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>C2</td><td>C1</td><td>C0</td><td>
<table>
<tr><th>C2</th><th>C1</th><th>C0</th><th>间隔</th><th>指令</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>5帧</td><td>0x00</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>64帧</td><td>0x01</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>128帧</td><td>0x02</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>256帧</td><td>0x03</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>2帧</td><td>0x04</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>3帧</td><td>0x05</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>4帧</td><td>0x06</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1帧</td><td>0x07</td></tr>
</table>
</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>D2</td><td>D1</td><td>D0</td><td>设置结束滚动页地址，范围：00--07</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0x00：虚拟指令</td></tr>
<tr><td>0</td><td>F7</td><td>F6</td><td>F5</td><td>F4</td><td>F3</td><td>F2</td><td>F1</td><td>F0</td><td>定义开始列：开始最小：0X00</td></tr>
<tr><td>0</td><td>G7</td><td>G6</td><td>G5</td><td>G4</td><td>G3</td><td>G2</td><td>G1</td><td>G0</td><td>定义结束列：结束最大：0X7f</td></tr>
<tr><td rowspan="11">(24) 连续垂直和水平滚动设置（Continuous Vertical and Horizontal Scroll Setup）</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>X1</td><td>X0</td><td>0x29：垂直向右水平滚动
0x2A：垂直向左水平滚动</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>A0</td><td>设置列滚动偏移的数量
0x00：无水平滚动 0x01：水平滚动 1 列</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>B2</td><td>B1</td><td>B0</td><td>设置起始滚动页地址，范围：00--07</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>C2</td><td>C1</td><td>C0</td><td>
<table>
<tr><th>C2</th><th>C1</th><th>C0</th><th>间隔</th><th>指令</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>5帧</td><td>0x00</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>64帧</td><td>0x01</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>128帧</td><td>0x02</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>256帧</td><td>0x03</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>2帧</td><td>0x04</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>3帧</td><td>0x05</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>4帧</td><td>0x06</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1帧</td><td>0x07</td></tr>
</table>
</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>D2</td><td>D1</td><td>D0</td><td>设置结束滚动页地址，范围：00--07</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>E5</td><td>E4</td><td>E3</td><td>E2</td><td>E1</td><td>E0</td><td>设置垂直滚动行偏移，范围：01—3f</td></tr>
<tr><td>0</td><td>F7</td><td>F6</td><td>F5</td><td>F4</td><td>F3</td><td>F2</td><td>F1</td><td>F0</td><td>定义开始列：开始最小：0X00</td></tr>
</table>

	0	G7	G6	G5	G4	G3	G2	G1	G0	**定义结束列**：结束最大：0X7f
(25) 停止滚动（Deactivate scroll）	0	0	0	1	0	1	1	1	0	**0x2E**：停止滚动 配合：**0x26、0x27、0x29、0x2A使用**
(26)开始滚动（Activate scroll）	0	0	0	1	0	1	1	1	1	**0x2F**：开始滚动
(27)设置垂直滚动区域（Set Vertical Scroll Area）	0	1	0	1	0	0	0	1	1	**0xA3**：设置垂直滚动区域
	0	0	0	A5	A4	A3	A2	A1	A0	滚动起始行，范围：00--63
	0	0	B6	B5	B4	B3	B2	B1	B0	滚动结束行，范围：01--64

请详细参考 IC 资料”SSD1309.PDF”。

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



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:

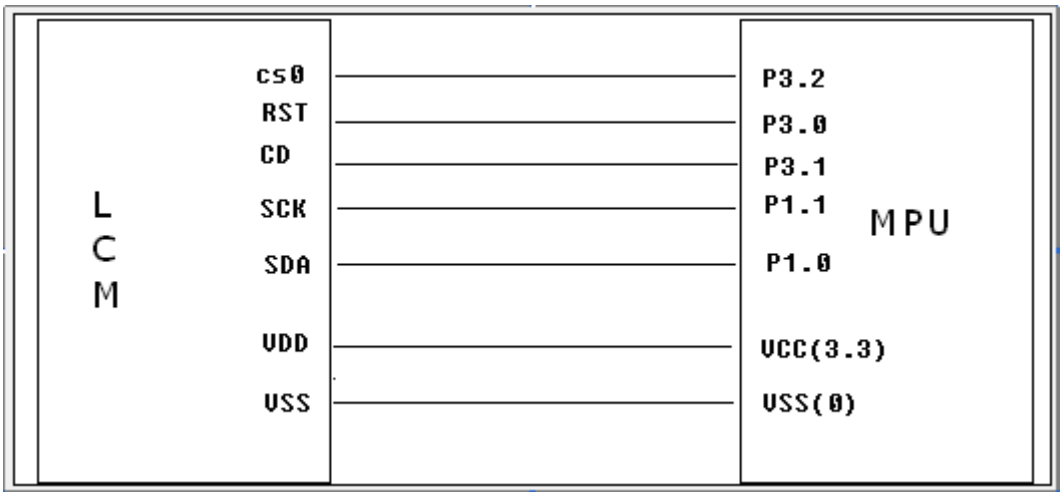
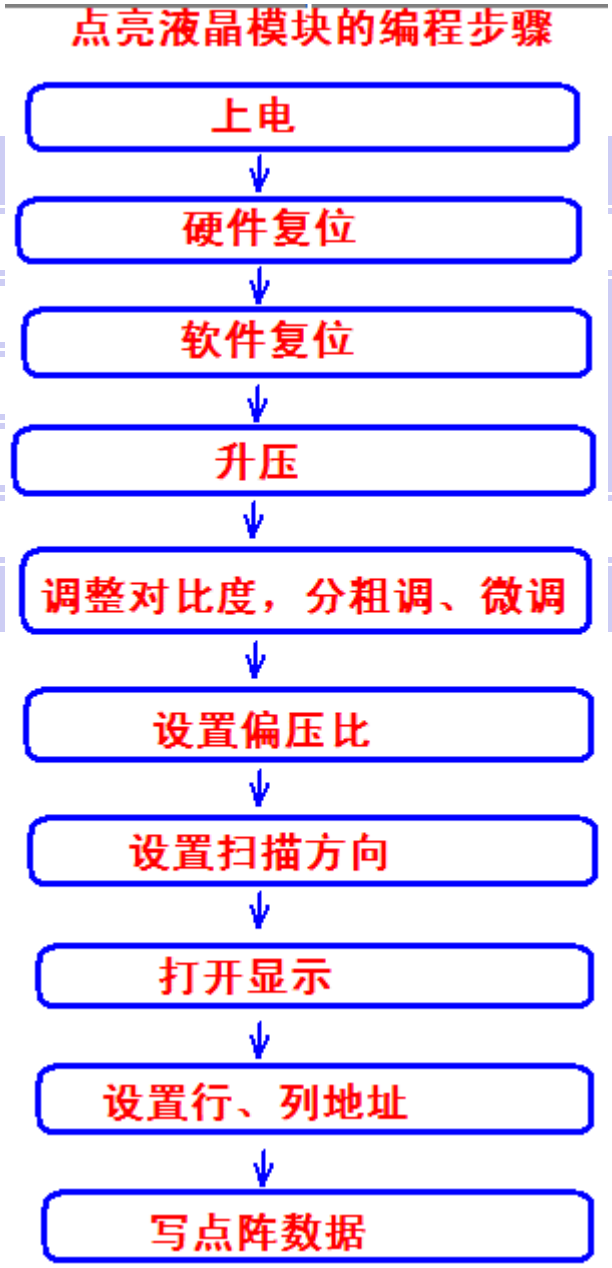
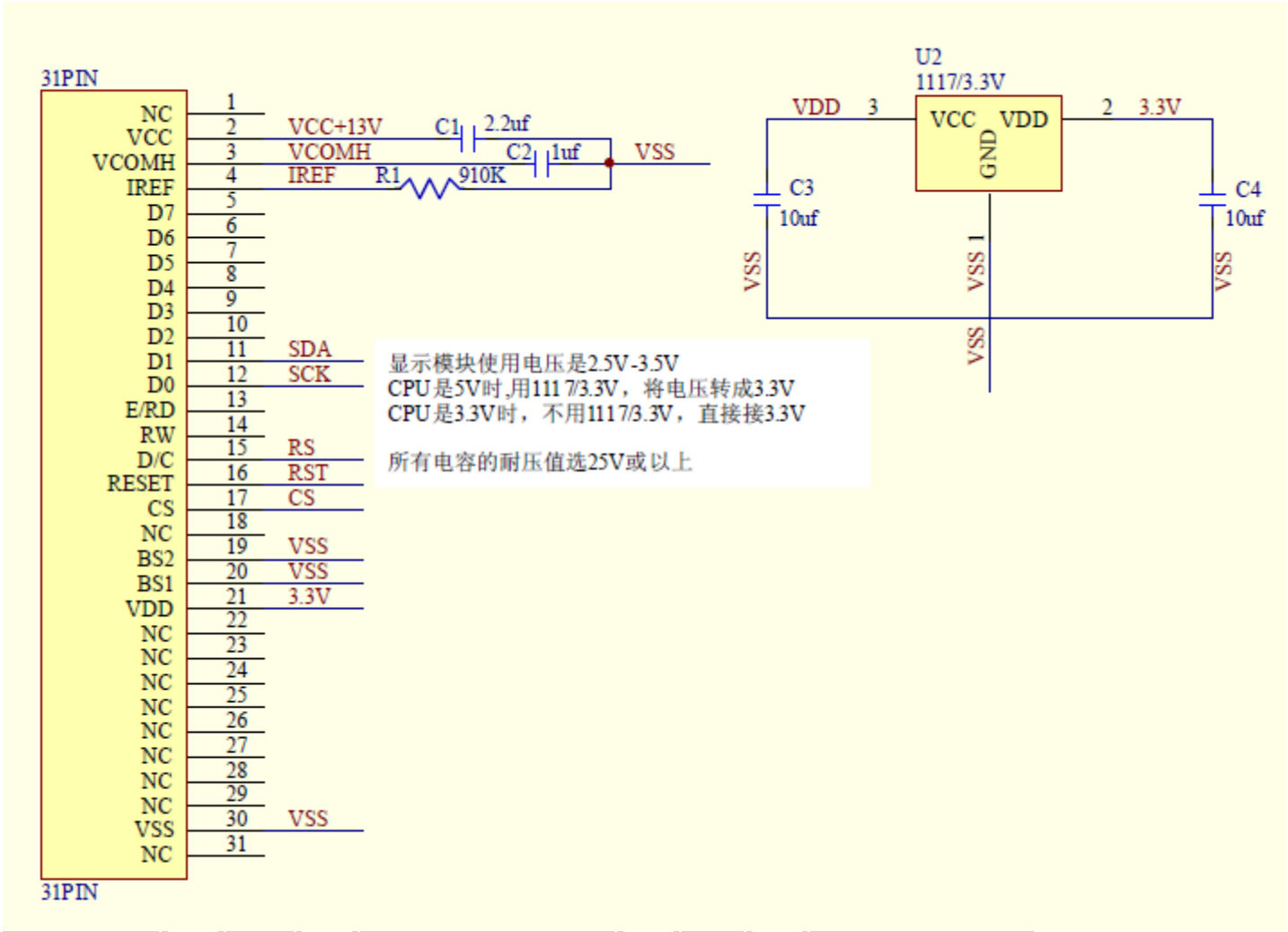


图 7.串行接口

7.5.1 程序:






```
//    液晶演示程序
//    液晶模块型号：JLX12864OLED-242，串行接口！
//    驱动 IC 是:SSD1309
//    版权所有：晶联讯电子：网址  http://www.jlxlcd.cn;
#include <reg52.H>
#include <intrins.h>
#include <string.h>
#include <stdio.h>
//=========================================================================
sbit SCK   = P1^1;    //SCK 接到“D0”脚
sbit SDA   = P1^0;    //SDA 接到“D1”脚
sbit RESET = P3^0;
sbit RS    = P3^1;
sbit CS    = P3^2;
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 (2000);
}

//写指令到 OLED 显示模块
void transfer_command(int data1)
{
    uchar i;
    CS=0;
    RS= 0;
     for(i=0;i<8;i++)
     {
        SCK= 0;
        if (data1 & 0x80)   SDA = 1;
        else                SDA = 0;
        SCK= 1;
        data1 <<= 1;
    }
    CS=1;
}

//写数据到 OLED 显示模块
void transfer_data(int data1)
{
    uchar i;
    CS=0;
    RS= 1;
     for(i=0;i<8;i++)
     {
        SCK= 0;
        if (data1 & 0x80)   SDA = 1;
        else                SDA = 0;
        SCK= 1;
        data1 <<= 1;
    }
```

```
   CS=1;
}

//OLED 显示模块初始化
void initial_lcd()
{
      RESET =0;              //低电平复位
      delay_ms (200);
      RESET =1;              //复位完毕
      delay_ms (200);
      transfer_command(0xfd);//设置指令锁
      transfer_command(0x12); //开锁
      transfer_command(0xae);//关显示
      transfer_command(0xd5);//晶振频率
      transfer_command(0xa0); //默认参数
      transfer_command(0xa8);//duty 设置
      transfer_command(0x3f);//duty=1/64
      transfer_command(0xd3);//显示偏移
      transfer_command(0x00);//范围：00--3F
      transfer_command(0x40);//起始行，默认 0x40
      transfer_command(0xc8); //行扫描顺序：从上到下
      transfer_command(0xa1); //列扫描顺序：从左到右
      transfer_command(0xda); //设置 COM 硬件配置
      transfer_command(0x12); //0X12:正常显示
      transfer_command(0x81); //微调对比度,本指令的 0x81 不要改动，改下面的值
      transfer_command(0xdf); //微调对比度的值，可设置范围 0x00～0xff
      transfer_command(0xd9); //预充电周期模式设置：
      transfer_command(0x82); //0X82:默认值，范围：00--ff
      transfer_command(0xdb); //0XDB:设置 VCOMH
      transfer_command(0x34); //0x34：0.78xVCC
      transfer_command(0xa4); //常规显示
      transfer_command(0xa6); //常规：正显
      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(j=0;j<8;j++)
    {
        lcd_address(1+j,1);
        for(i=0;i<128;i++)
        {
            transfer_data(0x00);
        }
    }
}

//full display test
void full_display(uchar data1,uchar data2)
{
    int i,j;
    for(i=0;i<8;i++)
    {
        lcd_address(i+1,1);
        for(j=0;j<64;j++)
        {
            transfer_data(data1);
            transfer_data(data2);
        }
    }
}
//测试外框是否缺划（少行、少列）
void test_box()
{
    int i,j;

//第 1 页：
    lcd_address(1,1);
    transfer_data(0xff);
    for(i=1;i<127;i++)
    {
        transfer_data(0x01);
    }
    transfer_data(0xff);
//第 2 页：
    lcd_address(2,1);
    transfer_data(0xff);
    for(i=1;i<127;i++)
    {
        transfer_data(0x80);
    }
    transfer_data(0xff);
```

```
//第 3 页：
	lcd_address(3,1);
	transfer_data(0xff);
	for(i=1;i<127;i++)
	{
		transfer_data(0x01);
	}
	transfer_data(0xff);

//第 4 页～第 7 页：
	for(j=4;j<=7;j++)
	{
		lcd_address(j,1);
		transfer_data(0xff);
		for(i=1;i<127;i++)
		{
			transfer_data(0x00);
		}
		transfer_data(0xff);
	}

//第 8 页：
	lcd_address(8,1);
	transfer_data(0xff);
	for(i=1;i<127;i++)
	{
		transfer_data(0x80);
	}
	transfer_data(0xff);
}

//测试
void test()
{
	full_display(0xff,0xff);
	waitkey();
	full_display(0x55,0x55);
	waitkey();
	full_display(0xaa,0xaa);
	waitkey();
	full_display(0xff,0x00);
	waitkey();
	full_display(0x00,0xff);
	waitkey();
	full_display(0x55,0xaa);
	waitkey();
```

```
    full_display(0xaa,0x55);
    waitkey();
    test_box();
    waitkey();
}

//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 128x16 点阵图像
void display_128x16(uchar page,uchar column,uchar *dp)
{
    uint i,j;
    for(j=0;j<2;j++)
    {
        lcd_address(page+j,column);
        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,"01234567890123456789");
          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();
     }
}
```

并行接口

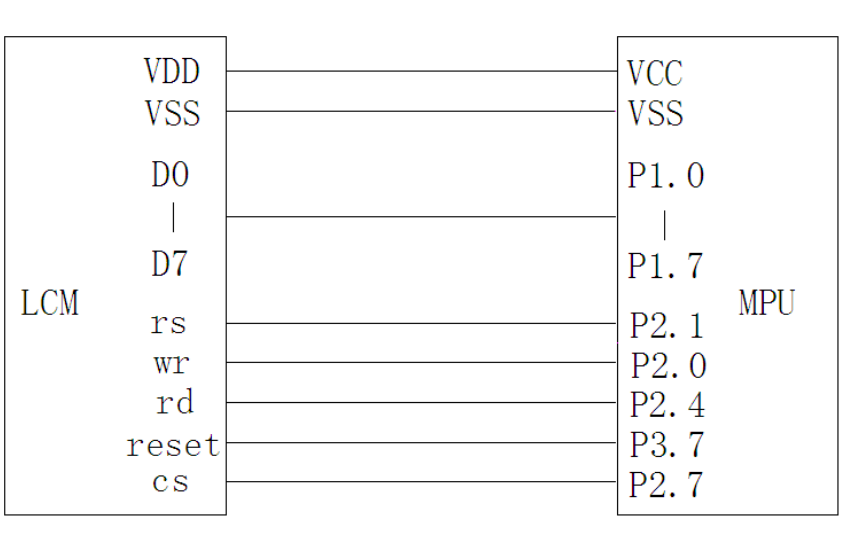
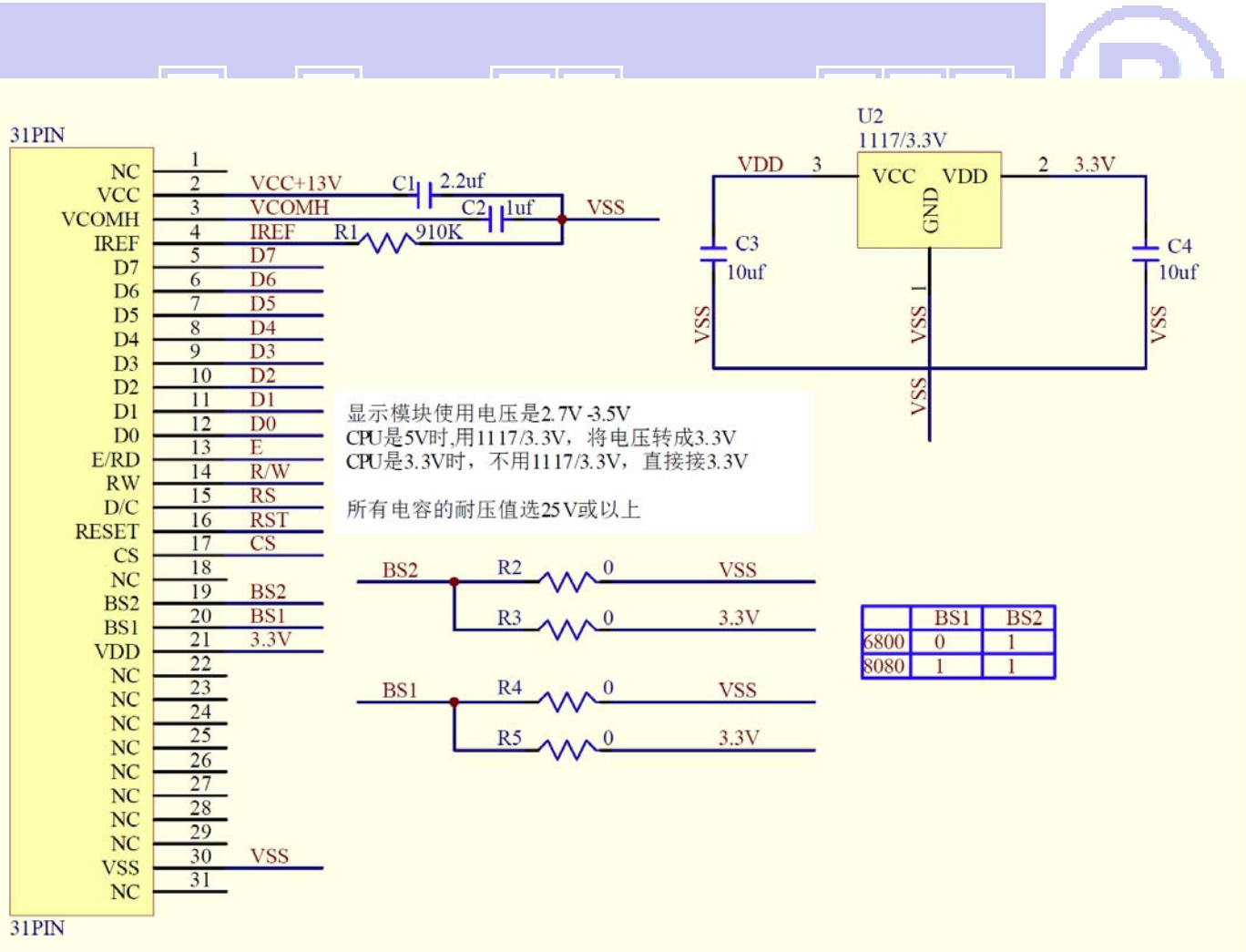


图 8.并行接口



以下为并行接口方式范例程序

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 rs=P2^1;     //OLED 的 RS
sbit rd=P2^4;     //OLED 的 E
sbit wr=P2^0;     //OLED 的 R/W
sbit reset=P3^7;
sbit cs=P2^7;
D0—D7 对应单片机的 P1.0—P1.7;

//写指令到 LCD 模块
void transfer_command_lcd(int data1)
{
    cs=0;
    rs =0;
    rd =0;
    wr =0;
    P1=data1;
    rd =1;
    cs=1;
    rd =0;
}
```

```
//写数据到 LCD 模块
void transfer_data_lcd(int data1)
{
    cs=0;
    rs =1;
    rd =0;
    wr =0;
    P1=data1;
    rd =1;
    cs=1;
    rd =0;
}
```

IIC 接口

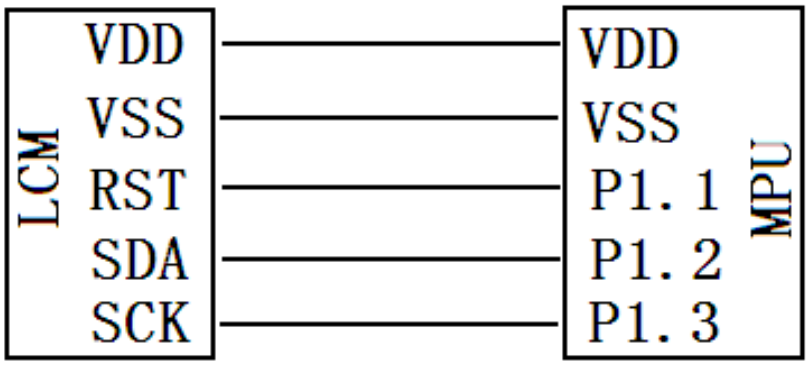
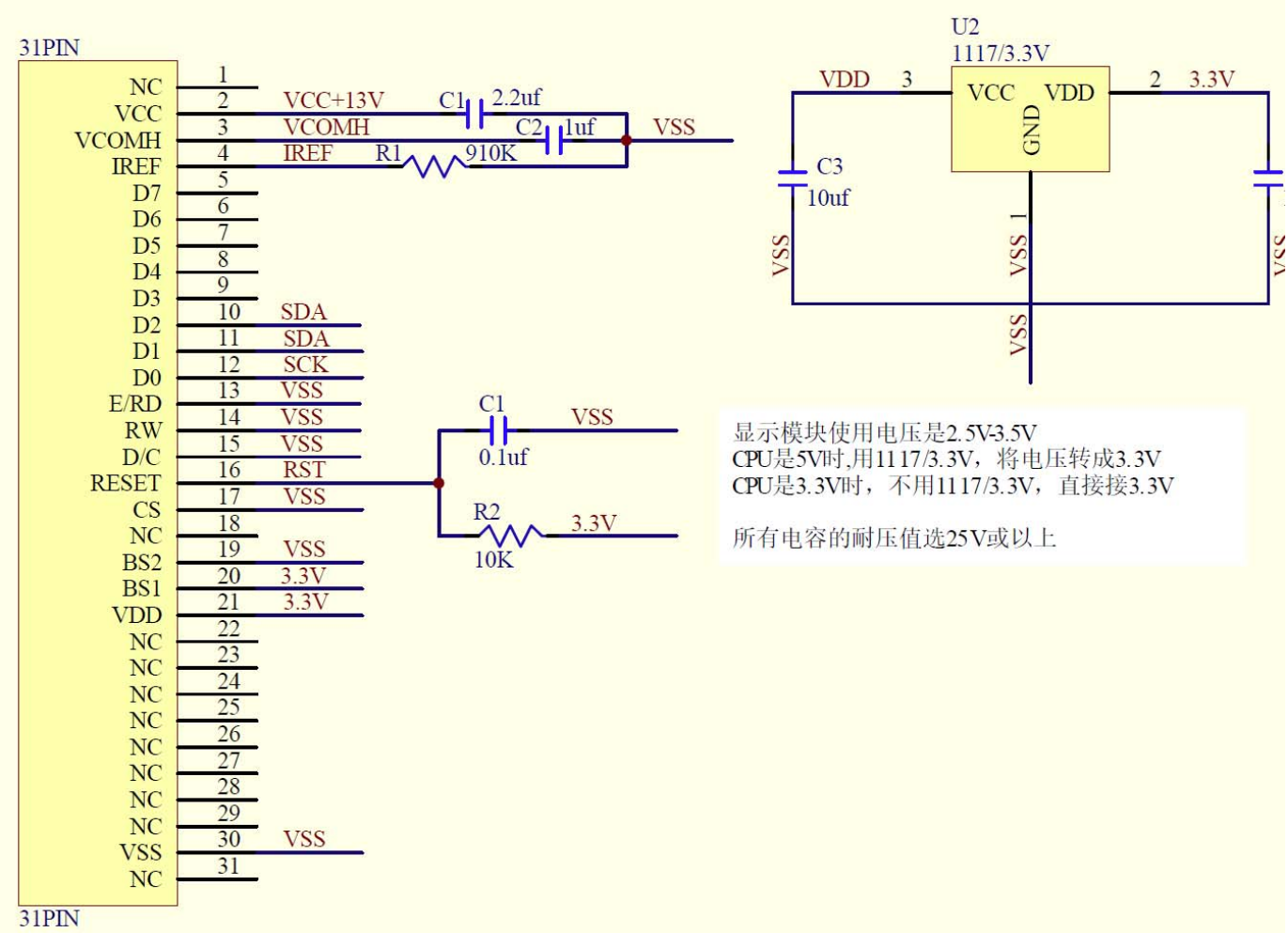


图 9. IIC



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：

IIC 接口

```
sbit   SCK=P1^3;     //OLED 的 SCLK
sbit   SDA=P1^2;     //OLED 的 SDA
```

```
void transfer(int data1)
{
      int i;
      for(i=0;i<8;i++)
     {
            SCK=0;
            if(data1&0x80) SDA=1;
            else SDA=0;
            SCK=1;
            SCK=0;
            data1=data1<<1;
     }
            lcd_SDA=0;
            SCK=0;
            SCK=1;
}

void start_flag()
{
      SCK=1;         /*START FLAG*/
      SDA=1;         /*START FLAG*/
      SDA=0;         /*START FLAG*/
```

```
}

void stop_flag()
{
    SCK=1;          /*STOP FLAG*/
    SDA=0;          /*STOP FLAG*/
    SDA=1;          /*STOP FLAG*/
}

//写命令到液晶显示模块
void transfer_command(uchar com)
{
    start_flag();
    transfer(0x78);
    transfer(0x80);
    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-