

JLX128128G-33802-BN 使用说明书

目 录

序号	内 容 标 题	页码
1	概述	2
2	字符型模块的特点	2
3	外形及接口引脚功能	3~4
4	基本原理	4~5
5	技术参数	5~6
6	时序特性	6~8
7	指令功能及硬件接口与编程案例	9~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 JLX128128G-33802 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。

JLX128128G-33802 可以显示不大于 128×128 点阵单色或 4 灰度级的图片, 或显示 8 个 $\times 8$ 行=64 个的 16×16 点阵的汉字, 或显示 16 个 $\times 8$ 行=128 个的 8×16 点阵的英文、数字、符号。或显示 21 个 $\times 16$ 行的 5×8 点阵的英文、数字、符号。

2. JLX128128G-33802 图像型点阵液晶模块的特性

1.1 结构牢: 背光带有挡墙, 插接式 FPC。

1.2 IC 采用 UC1617S, 功能强大, 稳定性好。

1.3 功耗低: $1 - 100\text{mW}$ (不带背光 $1\text{mW} < 3.3\text{V}@0.3\text{mA}$), 带背光不大于 $100\text{mW} < 3.3\text{V}@30\text{mA}$);

1.4 显示内容:

- 128×128 点阵单色图片或 4 灰度级的图片,

- 或显示 8 个 $\times 8$ 行=64 个的 16×16 点阵的汉字, 按照 12×12 点阵汉字来计算可显示 10 字/行 $\times 10$ 行。

- 或显示 16 个 $\times 8$ 行=128 个的 8×16 点阵的英文、数字、符号。

- 或显示 21 个 $\times 16$ 行的 5×8 点阵的英文、数字、符号。;

- 可选用 16×16 点阵或其他点阵的图片来自编汉字也可配合晶联讯字库 IC (JLX-GB2312-1602) 来显示汉字。

1.5 指令功能强;

1.6 接口简单方便: 可选 I²C 总线、4 线 SPI 串口、6800 系列并口、8080 系列并口。

1.7 工作温度宽: $-20^\circ\text{C} - 70^\circ\text{C}$;

1.8 可靠性高。

3. 外形尺寸及接口引脚功能

3.1 外形尺寸图

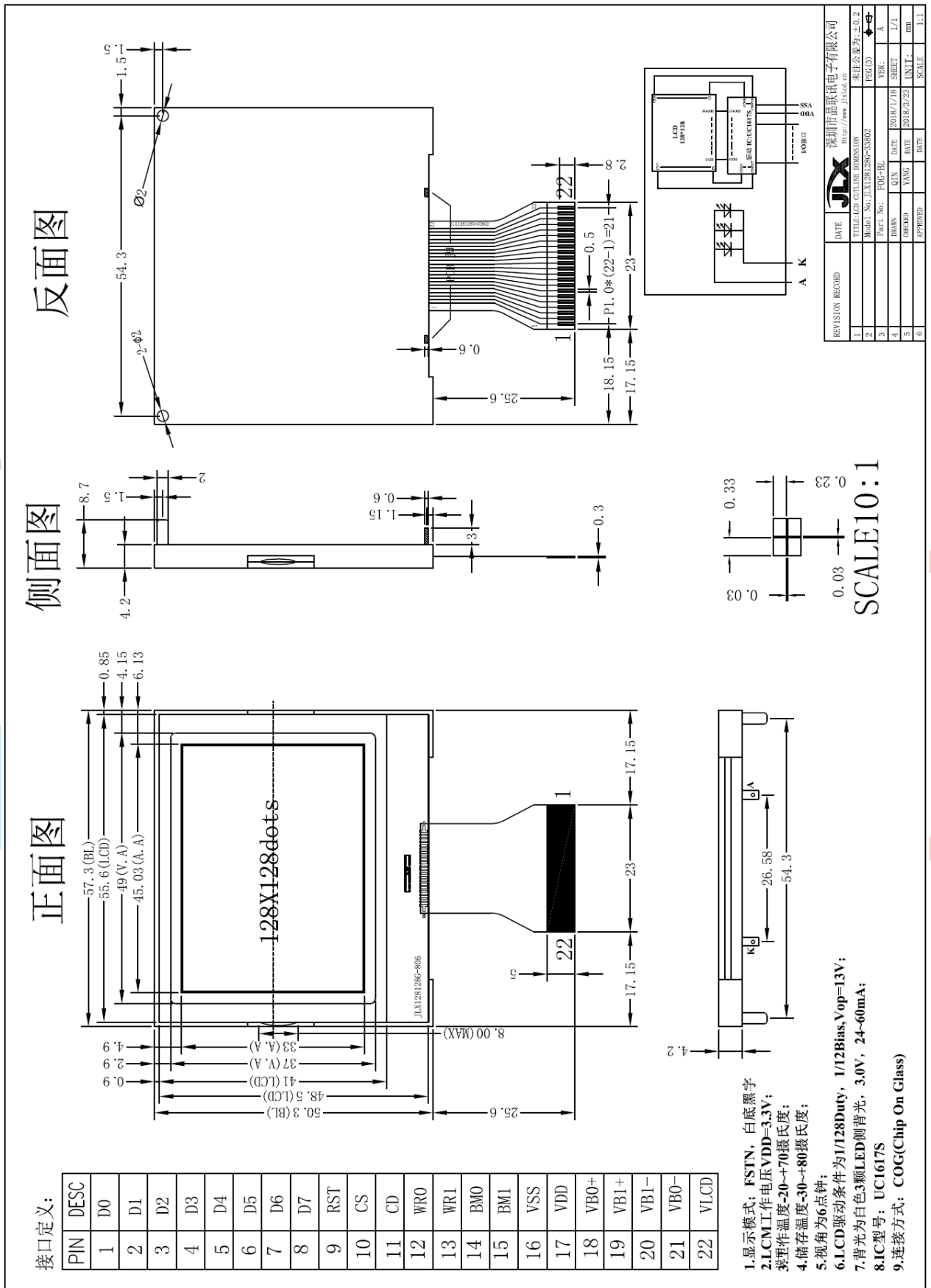


图 1. 外形尺寸

3.2 模块的接口引脚功能

3.2.1 接口引脚功能

引线号	符号	名称	功能																		
1	D0 (SCLK)	I/O	并行接口时: 数据总线 DB0 串行接口时: 串行时钟 (SCLK)																		
2	D1	I/O	数据总线																		
3	D2	I/O	数据总线																		
4	D3 (SDA)	I/O	并行接口时: 数据总线 DB3 串行接口时: 串行数据 (SDA)																		
5	D4	I/O	数据总线																		
6	D5	I/O	数据总线																		
7	D6	I/O	数据总线 (串行接口时: 接 VSS)																		
8	D7	I/O	数据总线 (串行接口时: 接 VDD)																		
9	RST	复位	低电平复位, 复位完成后, 回到高电平, 液晶模块开始工作																		
10	CS	片选	低电平片选																		
11	CD (RS)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器																		
12	WRO (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																		
13	WR1 (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																		
14	BM0	模式选择	Bus mode: The interface bus mode is determined by BM[1:0] and D[7:6] by the following relationship: <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>BM[1:0]</th> <th>D[7:6]</th> <th>Mode</th> </tr> </thead> <tbody> <tr> <td>11</td> <td>Data</td> <td>6800/8-bit</td> </tr> <tr> <td>10</td> <td>Data</td> <td>8080/8-bit</td> </tr> <tr> <td>01</td> <td>11</td> <td>2-wire I²C</td> </tr> <tr> <td>00</td> <td>10</td> <td>4-wire SPI w/ 8-bit token (S8: conventional)</td> </tr> <tr> <td>01</td> <td>10</td> <td>3-wire SPI w/ 9-bit token (S9: conventional)</td> </tr> </tbody> </table>	BM[1:0]	D[7:6]	Mode	11	Data	6800/8-bit	10	Data	8080/8-bit	01	11	2-wire I ² C	00	10	4-wire SPI w/ 8-bit token (S8: conventional)	01	10	3-wire SPI w/ 9-bit token (S9: conventional)
BM[1:0]	D[7:6]			Mode																	
11	Data	6800/8-bit																			
10	Data	8080/8-bit																			
01	11	2-wire I ² C																			
00	10	4-wire SPI w/ 8-bit token (S8: conventional)																			
01	10	3-wire SPI w/ 9-bit token (S9: conventional)																			
15	BM1																				
16	VSS	接地	0V																		
17	VDD	供电电源正极	供电电源正极																		
18	VB0+	倍压电路																			
19	VB1+	倍压电路																			
20	VB1-	倍压电路																			
21	VB0-	倍压电路																			
22	VLCD	倍压电路																			

4. 基本原理

4.1 液晶屏 (LCD)

在 LCD 上排列着 128×128 点阵, 128 个列信号与驱动 IC 相连, 128 个行信号也与驱动 IC 相连, IC 邦定在 LCD 玻璃上 (这种加工工艺叫 COG)。

4.2 工作电图:

图 1 是 JLX128128G-33802 图像点阵型模块的电路框图, 它由驱动 IC UC1617S 及几个电阻电容组

成。

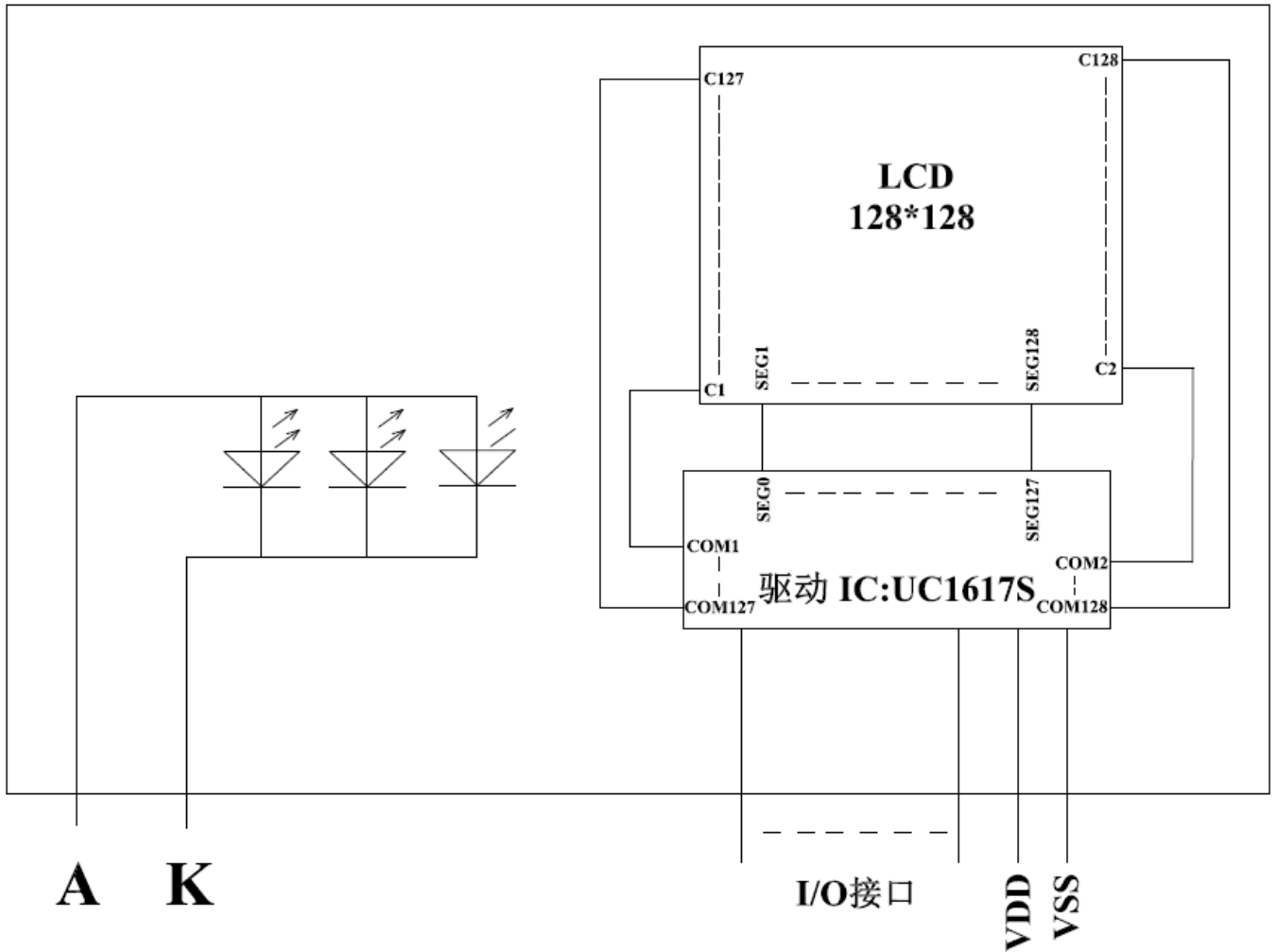


图 2: JLX128128G-33802 图像点阵型液晶模块的电路框图

4.3 背光参数

该型号液晶模块带 LED 背光源。它的性能参数如下:

工作温度: $-20\sim+70^{\circ}\text{C}$;

存储温度: $-30\sim+80^{\circ}\text{C}$;

背光灯颜色是白色。

正常工作电流为: $24\sim60\text{mA}$;

工作电压: 3.0V (用 3.3V 时, 请串限流电阻)。

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		7.0	V
LCD 驱动电压	VDD - V0	VDD - 13.5		VDD + 0.3	V
静电电压		-	-	100	V

工作温度		-20		+70	°C
储存温度		-30		+80	°C

表 5: 最大极限参数

5.2 直流 (DC) 参数

可以选择 3.3V 供电及 5.0V 供电两种方式:

名称	符号	测试条件	标准值			单位
			MIN	TYPE	MAX	
工作电压	VIN	3.3V 供电	2.7	3.3	3.5	V
输入高电平	VIH	-	2.2		VDD	V
输入低电平	VIO	-	-0.3		0.6	V
输出高电平	VOH	IOH = 0.2mA	2.4		-	V
输出低电平	VOO	IOO = 1.2mA	-		0.4	V
模块工作电流	IDD	VDD = 3.3V	-		0.3	mA
背光工作电流	ILED	VLED=3.0V	24	45	60	mA

表 6: 直流 (DC) 参数

6. 读写时序特性 (AC 参数)

6.1 4 线 SPI 串行接口写时序特性 (AC 参数)

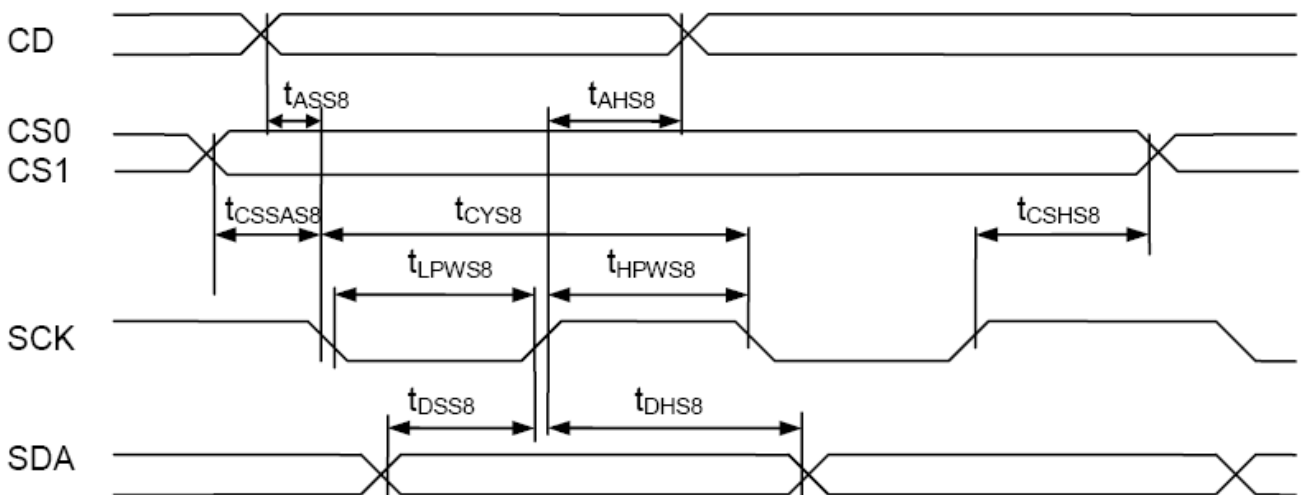


图 3. 从 CPU 写到 UC1617S (Writing Data from CPU to UC1617S)

表 7. 写数据到 UC1617S 的时序要求

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
读/写						
4线 SPI 串口时钟周期 (4-line SPI Clock Period)	T _{CYS8}	引脚: SCK	140/65	--		ns
保持SCK高电平脉宽 (SCLK "H" pulse width)	T _{HPWS8}		55/17			ns
保持SCLK低电平脉宽 (SCLK "L" pulse width)	T _{LPWS8}		55/17			ns

地址建立时间 (Address setup time)	T _{ASS8}	引脚: RS/CD	0	--	--	ns
地址保持时间 (Address hold time)	T _{AHS8}		0	--	--	ns
数据建立时间 (Data setup time)	T _{ACCS8}	引脚: SDA	15	--	--	ns
数据保持时间 (Data hold time)	T _{ODS8}		5	--	--	ns
片选信号建立时间 (CS-SCL time)	T _{CSSAS8}	引脚: CS	5			ns
片选信号保持时间 (CS-SCL time)	T _{Cshs8}		5			ns

VDD = 3.0V ± 5%, Ta = 25°C

6.2 6800 时序并行接口的时序特性 (AC 参数)

从 CPU 写到 UC1617S (Writing Data from CPU to UC1617S)

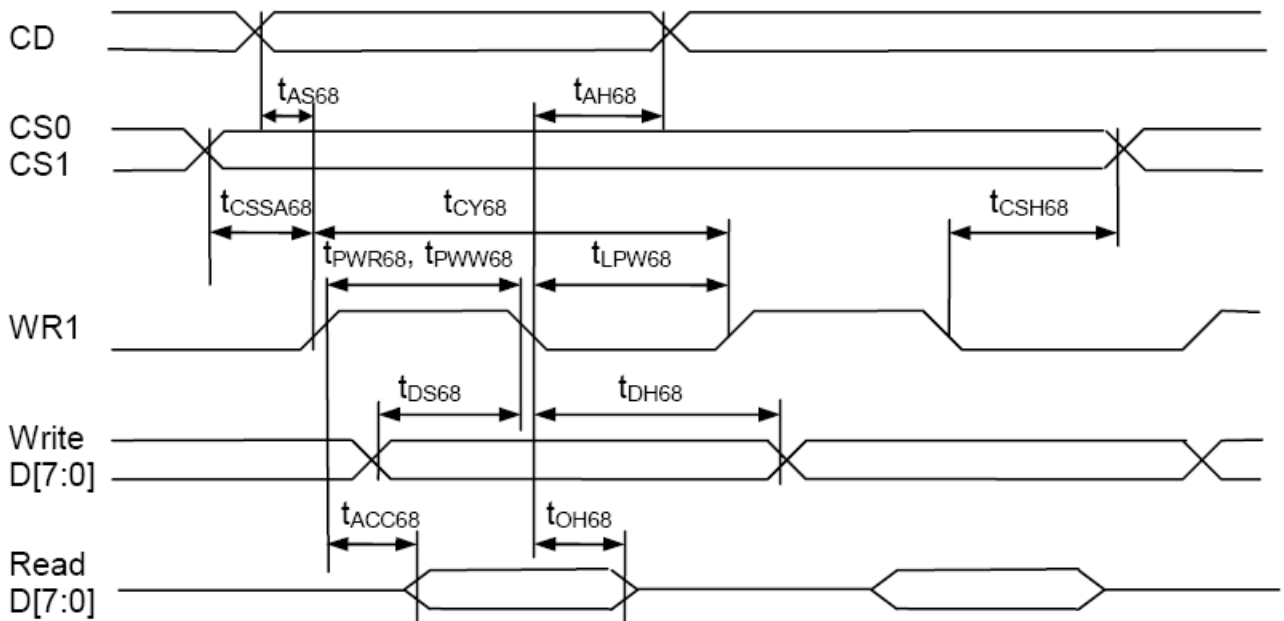


图 4. 写数据到 UC1617S 的时序要求 (6800 系列 MPU)

表 8. 读写数据的时序要求

项目	符号	名称	极限值			单位
			MIN	TYPE	MAX	
			读/写	读/写	读/写	
地址保持时间	A0/CD	T _{AS68}	0	--	--	ns
地址建立时间		T _{AH68}	0		--	ns
系统循环时间		T _{CY68}	170		--	ns
使能脉冲周期	WR1	T _{CY68}	200/160			
使能脉冲宽度		T _{PWR68/TPWW68}	85/65	--	--	ns
使能“低”脉冲宽度		T _{LPW68}	85/65	--	--	ns

写数据建立时间	D0-D7	TDS68	-/30		--	ns
写数据保持时间		TDH68	-/0		--	
读时间		TACC68	-/-		70/-	
读输出允许时间		TOH68	-/-		30/-	ns
片选信号建立时间	CS	TCSSA68	5			ns
片选信号保持时间		TCSH68	5			ns

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

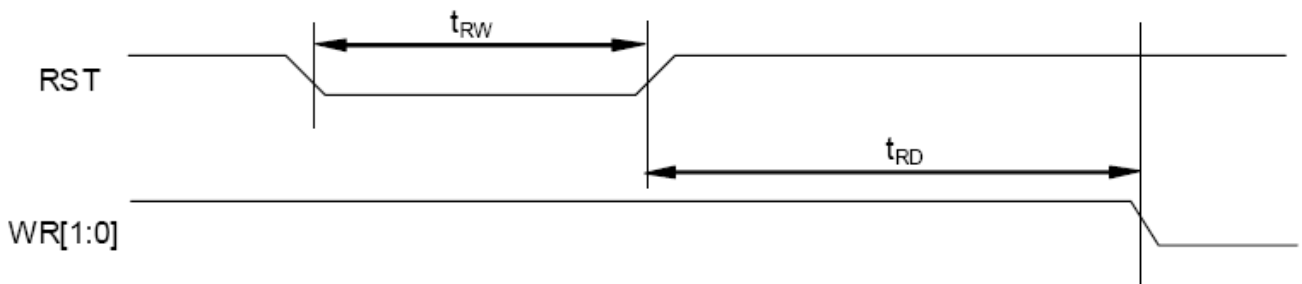


图 5: 电源启动后复位的时序

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_{RW}		3	--		us
复位保持低电平的时间	T_{RD}	引脚: RESET, WR	10	--	--	ms

7. 指令功能:

7.1 指令表

COMMAND SUMMARY

The following is a list of host commands supported by UC1617s

- C/D**: 0: Control, 1: Data
- W/R**: 0: Write Cycle, 1: Read Cycle
- D7-D0**: # Useful Data bits - Don't Care

	Command	C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0	Action	Default
1.	Write Data Byte	1	0	#	#	#	#	#	#	#	#	Write 1 byte	N/A
2.	Read Data Byte	1	1	#	#	#	#	#	#	#	#	Read 1 byte	N/A
3.	Get Status	0	1	-	MX	MY	WA	DE	WS	MD	MS	Get {Status, Ver, PMO, Prod, PID}	N/A
				Ver[1:0]			PMO[5:0]			Prod[3:0]			
4.	Set Page_C Address	0	0	0	0	0	#	#	#	#	#	Set CA[4:0]	00H
5.	Set Temp. Compensation	0	0	0	0	1	0	0	1	#	#	Set TC[1:0]	00b
6.	Set Panel Loading	0	0	0	0	1	0	1	0	#	#	Set PC[1:0]	10b
7.	Set Pump Control	0	0	0	0	1	0	1	1	#	#	Set PC[3:2]	11b
8.	Set Adv. Program Control (double-byte command)	0	0	0	0	1	1	0	0	R	R	Set R, R = 0~2	N/A
		0	0	#	#	#	#	#	#	#	#	#	
9.	Set Scroll Line LSB	0	0	0	1	0	0	#	#	#	#	Set SL[3:0]	0H
	Set Scroll Line MSB	0	0	0	1	0	1	-	#	#	#	Set SL[6:4]	0H
10.	Set Row Address LSB	0	0	0	1	1	0	#	#	#	#	Set RA[3:0]	0H
	Set Row Address MSB	0	0	0	1	1	1	-	#	#	#	Set RA[6:4]	0H
11.	Set V _{BIAS} Potentiometer (double-byte command)	0	0	1	0	0	0	0	0	0	1	Set PM[7:0]	4EH
		0	0	#	#	#	#	#	#	#	#		
12.	Set Partial Display Control	0	0	1	0	0	0	0	1	#	#	Set LC[10:9]	00b: Disable
13.	Set RAM Address Control	0	0	1	0	0	0	1	#	#	#	Set AC[2:0]	001b
14.	Set Fixed Lines	0	0	1	0	0	1	0	0	0	0	Set {FLT, FLB}	00H
		0	0	#	#	#	#	#	#	#	#		
15.	Set Line Rate	0	0	1	0	1	0	0	0	#	#	Set LC[4:3]	00b
16.	Set All-Pixel-ON	0	0	1	0	1	0	0	1	0	#	Set DC[1]	0b
17.	Set Inverse Display	0	0	1	0	1	0	0	1	1	#	Set DC[0]	0b
18.	Set Display Enable	0	0	1	0	1	0	1	1	#	#	Set DC[3:2]	10b
19.	Set LCD Mapping Control	0	0	1	1	0	0	0	#	#	#	Set LC[2:0]	000b
20.	Set N-Line Inversion	0	0	1	1	0	0	1	0	0	0	Set NIV[3:0]	6H
		0	0	-	-	-	-	#	#	#	#		
21.	Set LCD Gray Shade 1	0	0	1	1	0	1	0	0	#	#	Set LC[6:5]	01b
22.	Set LCD Gray Shade 2	0	0	1	1	0	1	0	1	#	#	Set LC[8:7]	10b
23.	System Reset	0	0	1	1	1	0	0	0	1	0	System Reset	N/A
24.	NOP	0	0	1	1	1	0	0	0	1	1	No operation	N/A
25.	Set Test Control (double-byte command)	0	0	1	1	1	0	0	1	TT		For testing only. Do not use.	N/A
		0	0	#	#	#	#	#	#	#	#		
26.	Set LCD Bias Ratio	0	0	1	1	1	0	1	0	#	#	Set BR[1:0]	11b: 11
27.	Set COM End	0	0	1	1	1	1	0	0	0	1	Set CEN[6:0]	127
		0	0	-	#	#	#	#	#	#	#		
28.	Set Partial Display Start	0	0	1	1	1	1	0	0	1	0	Set DST[6:0]	0
		0	0	-	#	#	#	#	#	#	#		
29.	Set Partial Display End	0	0	1	1	1	1	0	0	1	1	Set DEN[6:0]	127
		0	0	-	#	#	#	#	#	#	#		

Command		C/D	W/R	D7	D6	D5	D4	D3	D2	D1	D0	Action	Default		
30.	Set Window Program Starting Page_C Address	0	0	1	1	1	1	0	1	0	0	Note (3)	Set WPC0	0	
		0	0	-	-	-	#	#	#	#	#				
31.	Set Window Programming Starting Row Address	0	0	1	1	1	1	0	1	0	1			Set WPP0	0
		0	0	-	#	#	#	#	#	#	#				
32.	Set Window Programming Ending Page_C Address	0	0	1	1	1	1	0	1	1	0		Set WPC1	31	
		0	0	-	-	-	#	#	#	#	#				
33.	Set Window Programming Ending Row Address	0	0	1	1	1	1	0	1	1	1		Set WPP1	127	
		0	0	-	#	#	#	#	#	#	#				
34.	Enable window program	0	0	1	1	1	1	1	0	0	#	Set AC[3]	0: Disable		
35.	Set MTP Operation control	0	0	1	0	1	1	1	0	0	0	Set MTPC[5:0]	10H		
		0	0	-	-	#	#	#	#	#	#				
36.	Set MTP Write Mask	0	0	1	0	1	1	1	0	0	1	Set MTPM[5:0]	0		
		0	0	-	-	#	#	#	#	#	#				
37.	Set V _{MTP1} Potentiometer	0	0	1	1	1	1	0	1	0	0	Note (3)	Set MTP1	N/A	
		0	0	#	#	#	#	#	#	#	#				
38.	Set V _{MTP2} Potentiometer	0	0	1	1	1	1	0	1	0	1				Set MTP2
		0	0	#	#	#	#	#	#	#	#				
39.	Set MTP Write Timer	0	0	1	1	1	1	0	1	1	0		Set MTP3		
		0	0	#	#	#	#	#	#	#	#				
40.	Set MTP Read Timer	0	0	1	1	1	1	0	1	1	1		Set MTP4		
		0	0	#	#	#	#	#	#	#	#				
SERIAL READ COMMAND (ENABLED ONLY IN S8/S9 MODE)															
41.	Get Status	0	0	1	1	1	1	1	1	1	0	Get status until chip disabled	N/A		
		0	1	-	MX	MY	WA	DE	WS	MD	MS				
		0	1	Ver[1:0]		PMO[5:0]									
		0	1	Prod[3:0]			0	PID	0	0					

表 8. 指令表

请详细参考 IC 资料”UC1617S.PDF”。

7.3 点阵与 DD RAM 地址的对应关系

DB0—DB7 的排列方向：数据是从左向右排列的。

RAM

Line Addresss	Data								D1/0	D3/2	D5/4	D7/6	D1/0	D3/2	D5/4	D7/6	MY=0		MY=1						
	D1/0	D3/2	D5/4	D7/6	SL=0	SL=16	SL=0	SL=16																	
00H	11	10	01	00													R1	R113	R128	R16					
01H	00	11	10	01													R2	R114	R127	R15					
02H																	R3	R115	R126	R14					
03H																	R4	R116	R125	R13					
04H																	R5	R117	R124	R12					
05H																	R6	R118	R123	R11					
06H																	R7	R19	R122	R10					
07H																	R8	R120	R121	R9					
08H																	R9	R121	R120	R8					
09H																	R10	R122	R119	R7					
0AH																	R11	R123	R118	R6					
0BH																	R12	R124	R117	R5					
0CH																	R13	R125	R116	R4					
0DH																	R14	R126	R115	R3					
0EH																	R15	R127	R114	R2					
0FH																	R16	R128	R113	R1					
10H																	R17	R1	R112	R128					
11H																	R18	R2	R111	R127					
12H																	R19	R3	R110	R126					
13H																	R20	R4	R109	R125					
14H																	R21	R5	R108	R124					
15H																	R22	R6	R107	R123					
16H																	R23	R7	R106	R122					
17H																	R24	R8	R105	R121					
18H																	R25	R9	R104	R120					
19H																	R26	R10	R103	R119					
1AH																	R27	R11	R102	R118					
1BH																	R28	R12	R101	R117					
		Page_C0								Page_C1								Page_C31							
6CH																	R109	R93	R20	R36					
6DH																	R110	R94	R19	R35					
6EH																	R111	R95	R18	R34					
6FH																	R112	R96	R17	R33					
70H																	R113	R97	R16	R32					
71H																	R114	R98	R15	R31					
72H																	R115	R99	R14	R30					
73H																	R116	R100	R13	R29					
74H																	R117	R101	R12	R28					
75H																	R118	R102	R11	R27					
76H																	R119	R103	R10	R26					
77H																	R120	R104	R9	R25					
78H																	R121	R105	R8	R24					
79H																	R122	R106	R7	R23					
7AH																	R123	R107	R6	R22					
7BH																	R124	R108	R5	R21					
7CH																	R125	R109	R4	R20					
7DH																	R126	R110	R3	R19					
7EH																	R127	R111	R2	R18					
7FH																	R128	R112	R1	R17					
																		MUX							
MX		0		C1 C2 C3 C4 C5 C6 C7 C8								C125		C126		C127		C128							
		1		C128 C127 C126 C125 C124 C123 C122 C121								C4		C3		C2		C1							

Example: when MX=0, MY=0, SL=0, the corresponding data in SRAM as the pixels shown is:

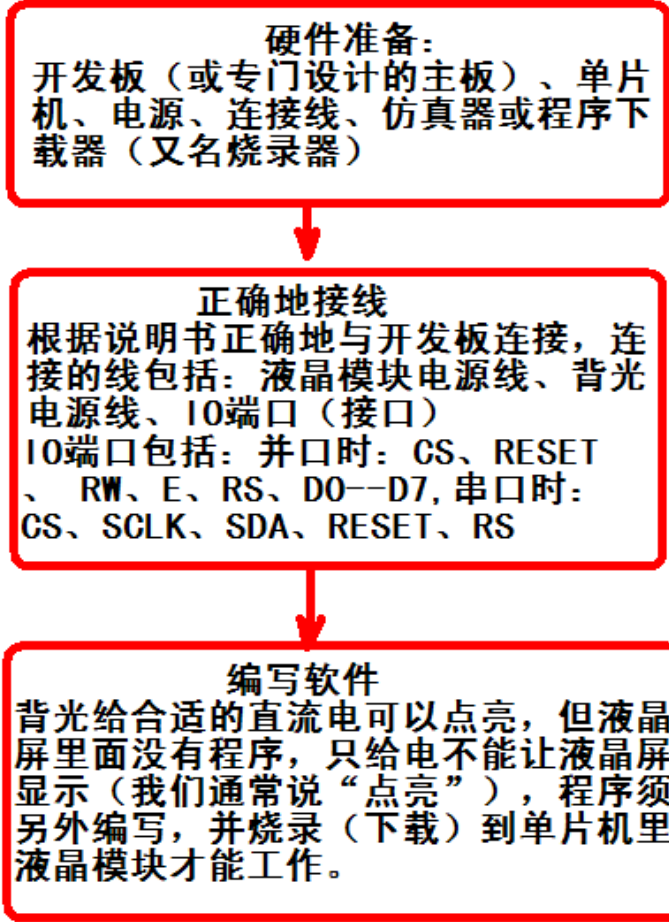
Row1 Page_C0 ⇒ D[7:0] : 00011011b

Row2 Page_C0 ⇒ D[7:0] : 01101100b

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：

7.5.1 并行接口

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下：

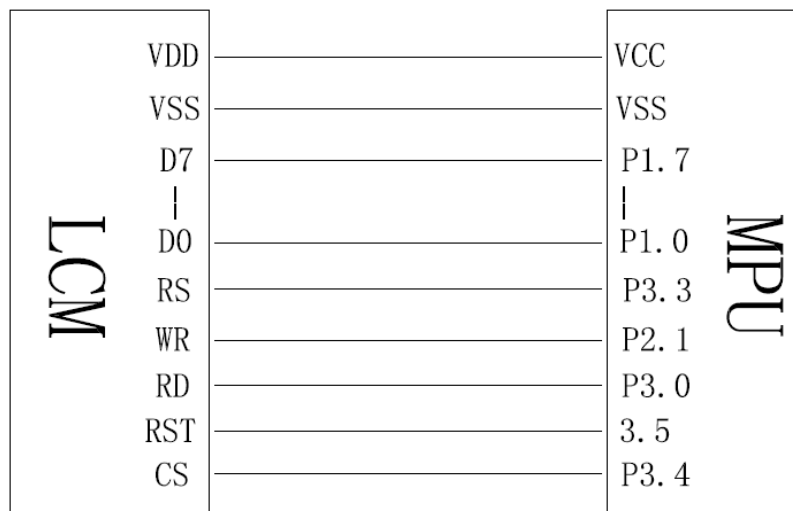
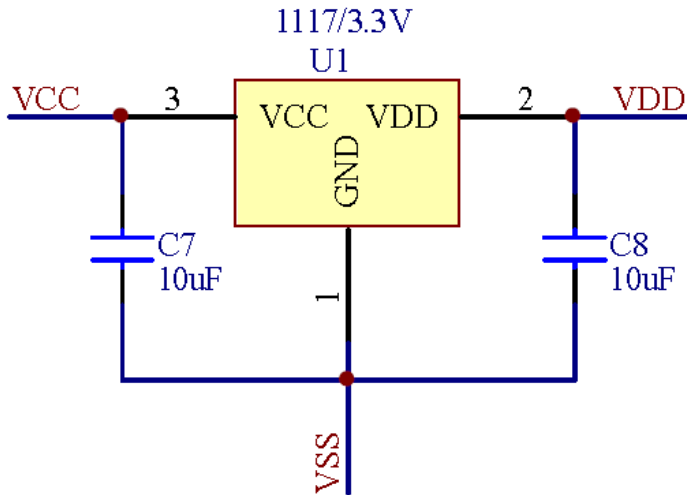
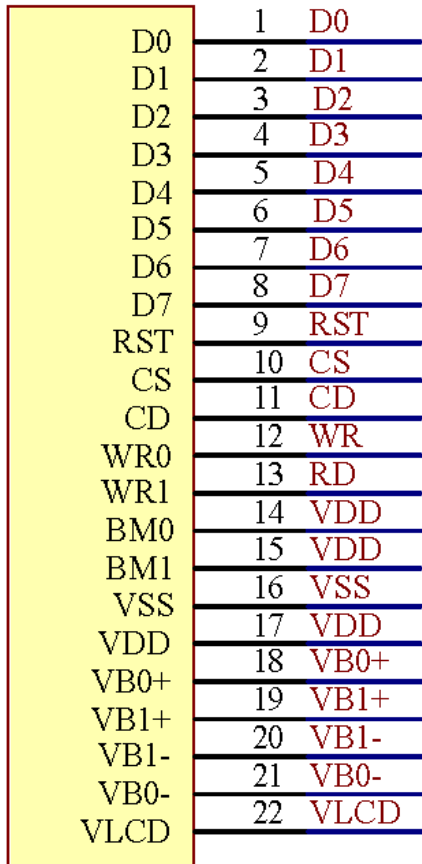


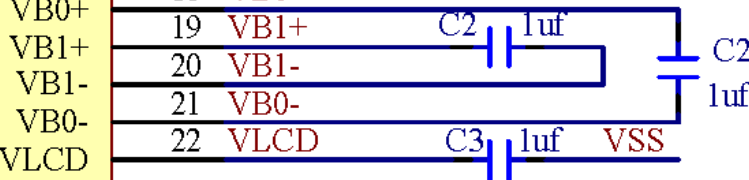
图 8. 并行接口

LCD



5V的时候用1117/3.3V降压到3.3V给LCD供电

LCD



并行接口

7.5.2 以下是并行接口例程序

```
//液晶模块: JLX128128G-338-BN
//驱动 IC:UC1617S. 4 灰阶 (也叫灰度级) 驱动 IC
//点阵: 128x128
//说明: 每个点阵是有 4 灰阶的 (2 的平方=4), 所以每个点阵是由 2 比特来代表的。
//为了应用一般的汉字库及普通的单色无灰阶, 所以增设了: "write_mono_data(uchar mono_data)" 这个函数
//如果要显示 4 灰阶的图像, 可以通过相关 4 灰阶的取模软件来取数据。
```

```
#include <REG52.H>

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

sbit LCD_CS=P3^4; /*3.4 接口定义*/
sbit LCD_RST=P3^5; /*3.3 接口定义*/
sbit LCD_RS=P3^3; /*接口定义*/
sbit rd=P3^0; /*接口定义*/
sbit wr=P2^1; /*接口定义。另外 P1.0~1.7 对应 DB0~DB7*/
sbit key=P2^0; /*按键接口, P2.0 口与 GND 之间接一个按键*/
```

```
uchar code bmp1[];

uchar code jiong1[]={//横向取模
/*- 文字: 囧 -*/
/*- 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00,0x00,0x7F,0xFC,0x44,0x84,0x46,0x44,0x44,0x24,0x48,0x34,0x50,0x14,0x6F,0xEB,
0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x7F,0xFC,0x00,0x00,
```

```

};

uchar code lei1[]={//横向取模
/*- 文字: 晶 -*/
/*- 宋体 12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x1F,0xF0,0x11,0x10,0x1F,0xF0,0x11,0x10,0x11,0x10,0x1F,0xF0,0x00,0x00,0xFE,0xFE,
0x92,0x92,0x92,0x92,0xFE,0xFE,0x92,0x92,0x92,0x92,0xFE,0xFE,0x82,0x82,0x00,0x00,
};

uchar code text[]={//横向取模
/*- 文字: 晶 -*/
/*- MS UI Gothic12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x00,0x00,0x0F,0xF0,0x08,0x10,0x0F,0xF0,0x08,0x10,0x0F,0xF0,0x08,0x10,0x00,0x00,
0x7E,0x7E,0x42,0x42,0x7E,0x7E,0x42,0x42,0x42,0x42,0x7E,0x7E,0x42,0x42,0x00,0x00,

/*- 文字: 联 -*/
/*- MS UI Gothic12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x01,0x08,0xFE,0x8C,0x44,0x48,0x44,0x50,0x7F,0xFE,0x44,0x20,0x44,0x20,0x7C,0x20,
0x47,0xFE,0x44,0x20,0x4E,0x20,0xF4,0x20,0x44,0x50,0x04,0x48,0x04,0x86,0x05,0x04,

/*- 文字: 讯 -*/
/*- MS UI Gothic12; 此字体下对应的点阵为: 宽 x 高=16x16 -*/
0x40,0x00,0x27,0xF8,0x31,0x08,0x21,0x08,0x01,0x08,0xF1,0x08,0x17,0xE8,0x11,0x08,
0x11,0x08,0x11,0x08,0x11,0x08,0x11,0x0A,0x15,0x0A,0x19,0x0A,0x11,0x04,0x00,0x00,

};

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

//等待一个按键
void waitkey()
{
repeat:
    if (key==1) goto repeat;
    else;
        delay(600);
}

//=====transfer command to LCM=====
void transfer_command_lcd(int data1)
{
    LCD_CS=0;
    LCD_RS=0;
    rd=0;
    wr=0;
    P1=data1;
    rd=1;
    LCD_CS=1;
    rd=0;
}

//-----transfer data to LCM-----

```

```

void transfer_data_lcd(int data1)
{
    LCD_CS=0;
    LCD_RS=1;
    rd=0;
    wr=0;
    P1=data1;
    rd=1;
    LCD_CS=1;
    rd=0;
}

/*传送一个字节（8 比特）黑白图像的数据*/
void write_mono_data(uchar mono_data)
{
    char i, j;
    uchar four_gray_data=0;    //定义 4 灰度级的数据
    for(j=0;j<2;j++)
    {
        four_gray_data=0;
        for(i=0;i<4;i++)
        {
            four_gray_data>>=2;    //4 灰度级的数据右移 2 位
            if(mono_data&0x80)    //单色黑白数据与 0x80（二进制 10000000）进行“与”运算
            {
                four_gray_data+=0xc0; //4 灰度级的数据+0xc0(二进制 11000000)
            }
            else;
            mono_data<<=1;    //单色黑白数据左移一位
        }
        transfer_data_lcd(four_gray_data);    //写进一个 8bits 的数据，驱动了 4 个像素点，因为每个像素点用了 2bits 数据
    }
}

void clear()
{
    int i, j;
    LCD_CS=0;
    transfer_command_lcd(0x70);    //行地址高 3 位
    transfer_command_lcd(0x60);    //行地址低 4 位
    transfer_command_lcd(0x00);    //列地址
    for(i=0;i<128;i++)
    {
        for(j=0;j<16;j++)
        {
            write_mono_data(0x00);
        }
    }
}

void lcd_address(uchar row, uchar column)
{
    transfer_command_lcd(0x00+column);    //列地址，每个地址管 4 列
    transfer_command_lcd(0x70+(row>>4));    //行地址的高 3 位
    transfer_command_lcd(0x60+(row&0x0f));    //行地址的低 4 位
}

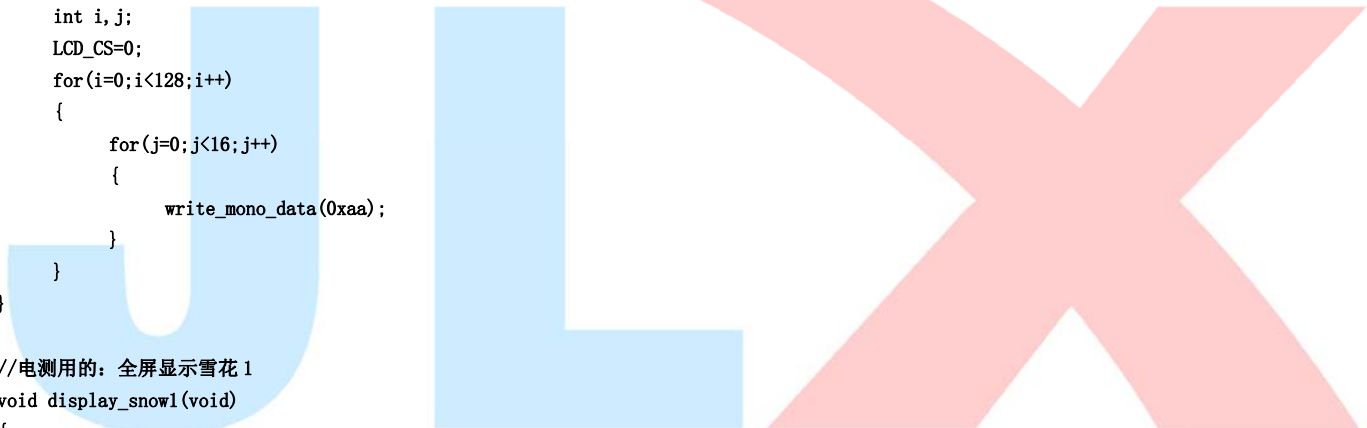
//电测用的：全屏显示黑
void display_black(void)
{

```

```

int i, j;
LCD_CS=0;
for(i=0;i<128;i++)
{
    for(j=0;j<16;j++)
    {
        write_mono_data(0xff);
    }
}
}
//电测用的：全屏显示偶数列
void display_even_column(void)
{
    int i, j;
    LCD_CS=0;
    for(i=0;i<128;i++)
    {
        for(j=0;j<16;j++)
        {
            write_mono_data(0x55);
        }
    }
}
//电测用的：全屏显示奇数列
void display_odd_column(void)
{
    int i, j;
    LCD_CS=0;
    for(i=0;i<128;i++)
    {
        for(j=0;j<16;j++)
        {
            write_mono_data(0xaa);
        }
    }
}
//电测用的：全屏显示雪花 1
void display_snow1(void)
{
    int i, j;
    LCD_CS=0;
    for(i=0;i<64;i++)
    {
        for(j=0;j<16;j++)
            write_mono_data(0x55);
        for(j=0;j<16;j++)
            write_mono_data(0xaa);
    }
}
//电测用的：全屏显示雪花 2
void display_snow2(void)
{
    int i, j;
    LCD_CS=0;
    for(i=0;i<64;i++)
    {
        for(j=0;j<16;j++)
            write_mono_data(0xaa);
        for(j=0;j<16;j++)

```




```
        write_mono_data(0x55);
    }
}
//电测用的: 全屏显示奇数行
void display_odd_row(void)
{
    int i, j;
    LCD_CS=0;
    for(i=0;i<64;i++)
    {
        for(j=0;j<16;j++)
            write_mono_data(0xFF);
        for(j=0;j<16;j++)
            write_mono_data(0x00);
    }
}
//电测用的: 全屏显示偶数行
void display_even_row(void)
{
    int i, j;
    LCD_CS=0;
    for(i=0;i<64;i++)
    {
        for(j=0;j<16;j++)
            write_mono_data(0x00);
        for(j=0;j<16;j++)
            write_mono_data(0xff);
    }
}

//定义显示窗口大小
void window_program()
{
    transfer_command_lcd(0x70);
    transfer_command_lcd(0x60);
    transfer_command_lcd(0x10);
    transfer_command_lcd(0x00);

    transfer_command_lcd(0xF4); //set colum address start
    transfer_command_lcd(0x00);

    transfer_command_lcd(0xF6);
    transfer_command_lcd(0x1f); //set colum address end

    transfer_command_lcd(0xF5); //set row address start
    transfer_command_lcd(0x00);

    transfer_command_lcd(0xF7); //set row address end
    transfer_command_lcd(0x7f);
    transfer_command_lcd(0xF9); //set window outside mode enable
}

void display_graphic_128x128(uchar *dp)
{
    int i, j, y=0, x=0, row;
    LCD_CS=0;
    for(i=0;i<128;i++)
    {
        transfer_command_lcd(0x00+x); //列地址, 每个地址管 4 列
```

```

row=y+i;
transfer_command_lcd(0x70+(row>>4)); //行地址的高3位
transfer_command_lcd(0x60+(row&0x0f)); //行地址的低4位
for(j=0;j<16;j++)
{
    write_mono_data(*dp);
    dp++;
}
}
}

```

```

void display_graphic_16x16(uchar row,uchar column,uchar *dp)
{
    int i,j;
    LCD_CS=0;
    for(i=0;i<16;i++)
    {
        lcd_address(row+i,column);
        for(j=0;j<2;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
    }
}

```

```

void display_N_16x16(uchar row,uchar column,uchar n,uchar *dp)
{
    int i,j,k,m;
    m=column;
    LCD_CS=0;
    for(k=0;k<n;k++)
    {
        for(i=0;i<16;i++)
        {
            lcd_address(row+i,column);
            for(j=0;j<2;j++)
            {
                write_mono_data(*dp);
                dp++;
            }
        }
        column+=4;
        m+=4;
        if(m>=29)
        {
            column=0;
            m=0;
            row+=16;
        }
        else;
    }
}

```

```

display_graphic_8x16(y,x,uchar *dp)
{
    int i,j,row;
    LCD_CS=0;

```

```
for(i=0;i<16;i++)
{
    transfer_command_lcd(0x00+x);          //列地址, 每个地址管 4 列
    row=y+i;
    transfer_command_lcd(0x70+(row>>4)); //行地址的高 3 位
    transfer_command_lcd(0x60+(row&0x0f)); //行地址的低 4 位
    for(j=0;j<1;j++)
    {
        write_mono_data(*dp);
        dp++;
    }
}
}
display_graphic_8x8(y, x, uchar *dp)
{
    int i, j, row;

    for(i=0;i<8;i++)
    {
        transfer_command_lcd(0x00+x);
        row=y+i;
        transfer_command_lcd(0x70+(row>>4));
        transfer_command_lcd(0x60+(row&0x0f));
        for(j=0;j<1;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
    }
}

void LCD_INITIAL()
{
    LCD_RST=0;
    delay(200);
    LCD_RST=1;
    delay(50);

    transfer_command_lcd(0xE2); //system reset
    delay(10);
    transfer_command_lcd(0x27);
    transfer_command_lcd(0x2b);
    transfer_command_lcd(0x2f); //set pump control
    transfer_command_lcd(0xeb); //set bias=1/11
    transfer_command_lcd(0x81); //set
    transfer_command_lcd(0x36); //set PM=12, vop=12.8v, 4c
    transfer_command_lcd(0xa9); //set linerate mux, a2
    transfer_command_lcd(0xc8);
    transfer_command_lcd(0x0b);
    transfer_command_lcd(0x89);
    transfer_command_lcd(0xc4); //MY=1, MX=0: 从左到右, 再从上到下。
    transfer_command_lcd(0xf1); //f1
    transfer_command_lcd(0x7f);
    transfer_command_lcd(0xd3); //gray shade set
    transfer_command_lcd(0xd7); //gray shade set
    transfer_command_lcd(0xaf); //set display enable
}

void main(void)
{
```

```

LCD_INITIAL();
while(1)
{
    clear();
    // display_graphic_16x16(0, 1, lei1);           /*在第 yy 行, 第 xx 列显示单个自编生僻汉字“囧”*/
    // display_graphic_16x16(0, 8, jin);           /*显示单个自编生僻汉字“晶”*/
    display_N_16x16(0, 0, 64, text);
    waitkey();
    clear();
    display_graphic_128x128 bmp1;
    waitkey();
    display_black();
    waitkey();
    display_odd_column();
    waitkey();
    display_even_column();
    waitkey();
    display_odd_row();
    waitkey();
    display_even_row();
    waitkey();
    display_snow1();
    waitkey();
    display_snow2();
    waitkey();
}
}

uchar code bmp1[]={//横向取模
/*- 调入了一幅图像: E:\work\图片收藏夹\黑白屏图片\JLX128128G-338. bmp --*/
/*- 宽度 x 高度=128x128 --*/
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01,

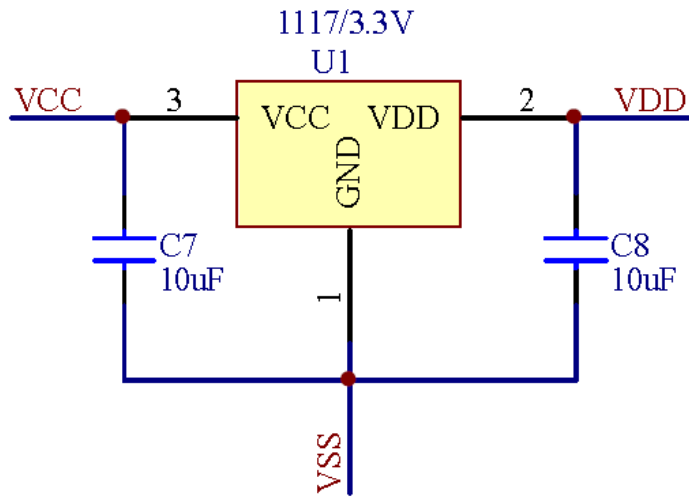
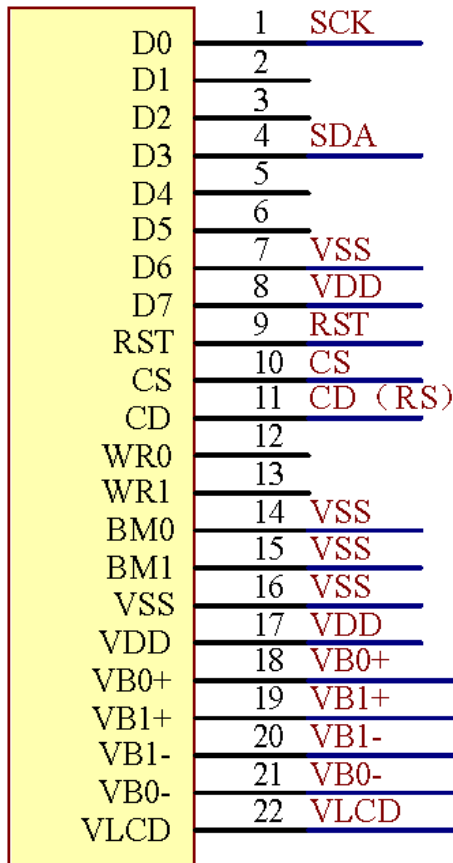
```

7.5.4 串行接口

液晶模块与 MPU(以 8051 系列单片机为例)接口图如下:

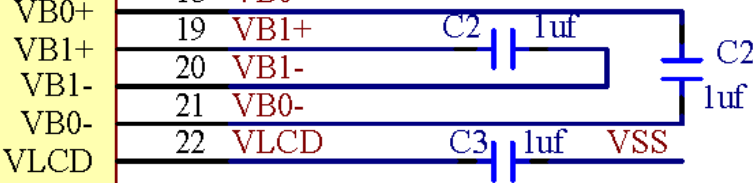


LCD



5V的时候用1117/3.3V降压到3.3V给LCD供电

LCD



串行接口

7.5.5 以下是串行接口例程序

与并行程序相比, 只需改变传送数据和命令子程序即可

//传送指令

```
void transfer_command_lcd(unsigned char cmd)
{
    int k;

    LCD_CS = 0;
    LCD_RS = 0;

    for (k=0; k<8; k++)
    {
        cmd = cmd<<1;
        LCD_SCL = 0;
        LCD_SDA = CY;
        LCD_SCL = 1;
    }

    LCD_CS=1;
}
```

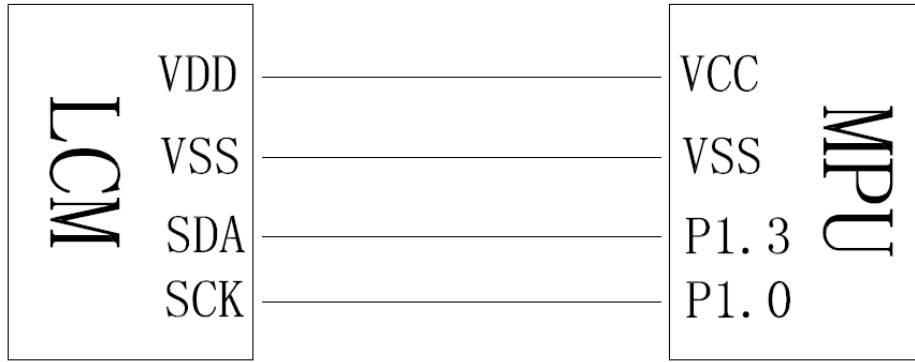
//传送数据

```
void transfer_data_lcd(unsigned char dat)
{
    unsigned char k;
    LCD_CS = 0;
    LCD_RS = 1;
    for (k=0; k<8; k++)
    {
```

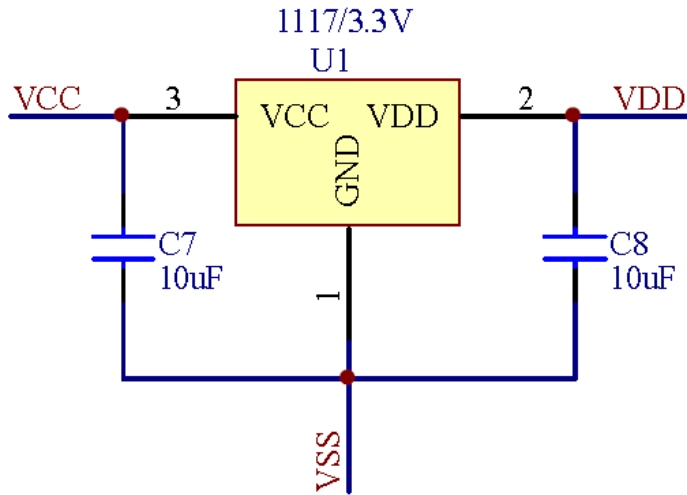
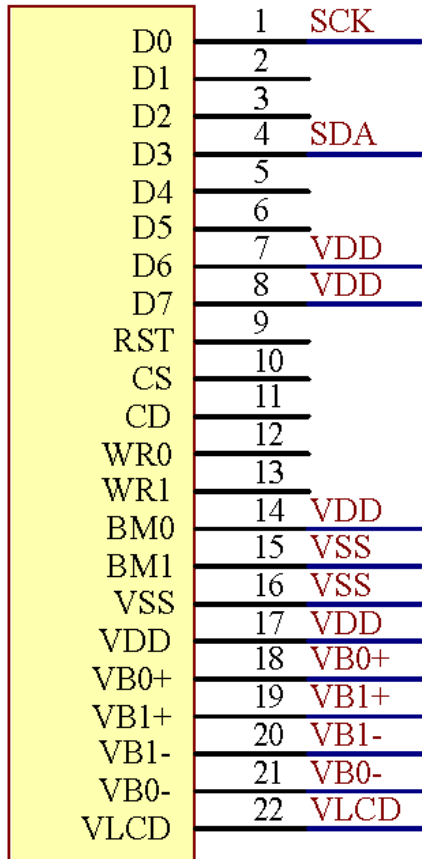
```

dat = dat<<1;
LCD_SDA = CY;
LCD_SCL = 0;
LCD_SCL = 1;
}
LCD_CS=1;
}
    
```

7.5.4 I²C 接口



LCD



5V的时候用1117/3.3V降压到3.3V给LCD供电

LCD

I²C 接口

7.5.5 以下是 I²C 接口例程序
7.5.5 以下是 I²C 接口例程序

//液晶模块: JLX128128G-328-PN

//驱动 IC:UC1617S. 4 灰阶 (也叫灰度级) 驱动 IC

//点阵：128x128

//说明：每个点阵是有4灰阶的（2的平方=4），所以每个点阵是由2比特来代表的。

//为了应用一般的汉字库及普通的单色无灰阶,所以增设了：“write_mono_data(uchar mono_data)”这个函数

//如果要显示4灰阶的图像，可以通过相关4灰阶的取模软件来取数据。

```
#include <REG52.H>
```

```
#define uchar unsigned char
```

```
#define uint unsigned int
```

```
#define ulong unsigned long int
```

```
sbit sda = P1^3;
```

```
sbit scl = P1^0;
```

```
sbit key=P2^0; /*按键接口，P2.0口与GND之间接一个按键*/
```

```
uchar code bmp1[];
```

```
uchar code jiong1[]={//横向取模
```

```
/*-- 文字： 囧 --*/
```

```
/*-- 宋体 12; 此字体下对应的点阵为：宽 x 高=16x16 --*/
```

```
0x00,0x00,0x7F,0xFC,0x44,0x84,0x46,0x44,0x44,0x24,0x48,0x34,0x50,0x14,0x6F,0xE4,
0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x48,0x24,0x7F,0xFC,0x00,0x00,
};
```

```
uchar code lei1[]={//横向取模
```

```
/*-- 文字： 磊 --*/
```

```
/*-- 宋体 12; 此字体下对应的点阵为：宽 x 高=16x16 --*/
```

```
0x1F,0xF0,0x11,0x10,0x1F,0xF0,0x11,0x10,0x11,0x10,0x1F,0xF0,0x00,0x00,0xFE,0xFE,
0x92,0x92,0x92,0x92,0xFE,0xFE,0x92,0x92,0x92,0x92,0xFE,0xFE,0x82,0x82,0x00,0x00,
};
```

```
uchar code text[]={//横向取模
```

```
/*-- 文字： 晶 --*/
```

```
/*-- MS UI Gothic12; 此字体下对应的点阵为：宽 x 高=16x16 --*/
```

```
0x00,0x00,0x0F,0xF0,0x08,0x10,0x0F,0xF0,0x08,0x10,0x0F,0xF0,0x08,0x10,0x00,0x00,
0x7E,0x7E,0x42,0x42,0x7E,0x7E,0x42,0x42,0x42,0x42,0x7E,0x7E,0x42,0x42,0x00,0x00,
```

```
/*-- 文字： 联 --*/
```

```
/*-- MS UI Gothic12; 此字体下对应的点阵为：宽 x 高=16x16 --*/
```

```
0x01,0x08,0xFE,0x8C,0x44,0x48,0x44,0x50,0x7F,0xFE,0x44,0x20,0x44,0x20,0x7C,0x20,
0x47,0xFE,0x44,0x20,0x4E,0x20,0xF4,0x20,0x44,0x50,0x04,0x48,0x04,0x86,0x05,0x04,
```

```
/*-- 文字： 讯 --*/
```

```
/*-- MS UI Gothic12; 此字体下对应的点阵为：宽 x 高=16x16 --*/
```

```
0x40,0x00,0x27,0xF8,0x31,0x08,0x21,0x08,0x01,0x08,0xF1,0x08,0x17,0xE8,0x11,0x08,
```

0x11,0x08,0x11,0x08,0x11,0x08,0x11,0x0A,0x15,0x0A,0x19,0x0A,0x11,0x04,0x00,0x00,

/*};

//延时

```
void delay(int n_ms)
{
    int i,j;
    for(i=0;i<n_ms;i++)
        for(j=0;j<110;j++);
}
```

//等待一个按键

```
void waitkey()
{
    repeat:
    if (key==1) goto repeat;
    else;
    delay(600);
}
```

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

```
void start_flag()
{
    scl=1;    /*START FLAG*/
    sda=1;    /*START FLAG*/
    sda=0;    /*START FLAG*/
}
```



```
void stop_flag()
```

```
{
    scl=1;      /*STOP FLAG*/
    sda=0;      /*STOP FLAG*/
    sda=1;      /*STOP FLAG*/
}
```

```
/*传送一个字节（8 比特）黑白图像的数据*/
```

```
void write_mono_data(uchar mono_data)
```

```
{
    char i,j;
    uchar four_gray_data=0; //定义 4 灰度级的数据
    for(j=0;j<2;j++)
    {
        four_gray_data=0;
        for(i=0;i<4;i++)
        {
            four_gray_data>>=2; //4 灰度级的数据右移 2 位
            if(mono_data&0x80) //单色黑白数据与 0x80（二进制 10000000）进行“与”运算
            {
                four_gray_data+=0xc0; //4 灰度级的数据+0xc0(二进制 11000000)
            }
            else;
            mono_data<<=1; //单色黑白数据左移一位
        }
        transfer(four_gray_data); //写进一个 8bits 的数据，驱动了 4 个像素点，因为每个像素点用了 2bits 数据
    }
}
```

```
void clear()
```

```
{
    int i,j;
    start_flag();
    transfer(0x7e);
    for(i=0;i<128;i++)
    {
        for(j=0;j<16;j++)
        {
            write_mono_data(0x00);
        }
    }
    stop_flag();
}
```

```
void lcd_address(uchar row,uchar column)
```

```
{
```

```

start_flag();
transfer(0x7c);
transfer(0x00+column); //列地址，每个地址管 4 列
transfer(0x70+(row>>4)); //行地址的高 3 位
transfer(0x60+(row&0x0f)); //行地址的低 4 位
stop_flag();
}

```

//电测用的：全屏显示黑

```
void display_black(void)
```

```

{
    int i,j;
    for(i=0;i<128;i++)
    {
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)
        {
            write_mono_data(0xff);
        }
        stop_flag();
    }
}

```

//电测用的：全屏显示偶数列

```
void display_even_column(void)
```

```

{
    int i,j;
    for(i=0;i<128;i++)
    {
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)
        {
            write_mono_data(0x55);
        }
        stop_flag();
    }
}

```

//电测用的：全屏显示奇数列

```
void display_odd_column(void)
```

```

{
    int i,j;
    for(i=0;i<128;i++)
    {
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)

```

```

    {
        write_mono_data(0xaa);
    }
    stop_flag();
}
}

```

//电测用的: 全屏显示雪花 1

```

void display_snow1(void)
{
    int i,j;
    for(i=0;i<64;i++)
    {
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)
            write_mono_data(0x55);
        for(j=0;j<16;j++)
            write_mono_data(0xaa);
        stop_flag();
    }
}

```

//电测用的: 全屏显示雪花 2

```

void display_snow2(void)
{
    int i,j;
    for(i=0;i<64;i++)
    {
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)
            write_mono_data(0xaa);
        for(j=0;j<16;j++)
            write_mono_data(0x55);
        stop_flag();
    }
}

```

//电测用的: 全屏显示奇数行

```

void display_odd_row(void)
{
    int i,j;
    for(i=0;i<64;i++)
    {
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)
            write_mono_data(0xFF);
    }
}

```

```

        for(j=0;j<16;j++)
            write_mono_data(0x00);
        stop_flag();
    }
}

```

//电测用的：全屏显示偶数行

```

void display_even_row(void)
{
    int i,j;
    for(i=0;i<64;i++)
    {
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)
            write_mono_data(0x00);
        for(j=0;j<16;j++)
            write_mono_data(0xff);
        stop_flag();
    }
}

```

//定义显示窗口大小

```

void window_program()
{
    start_flag();
    transfer(0x7c);
    transfer(0x60);
    transfer(0x10);
    transfer(0x00);

    transfer(0xF4);//set colum address start
    transfer(0x00);

    transfer(0xF6);
    transfer(0x1f);//set colum address end

    transfer(0xF5);//set row address start
    transfer(0x00);

    transfer(0xF7);//set row address end
    transfer(0x7f);
    transfer(0xF9);//set window outside mode enable
    stop_flag();
}

```



```
void display_graphic_128x128(uchar row,uchar column,uchar *dp)
```

```
{
    int i,j;
    for(i=0;i<128;i++)
    {
        start_flag();
        transfer(0x7c);
        lcd_address(row+i,column);
        stop_flag();
        start_flag();
        transfer(0x7e);
        for(j=0;j<16;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
        stop_flag();
    }
}
```

```
void display_graphic_16x16(uchar row,uchar column,uchar *dp)
```

```
{
    int i,j;
    for(i=0;i<16;i++)
    {
        start_flag();
        transfer(0x7c);
        lcd_address(row+i,column);
        stop_flag();
        start_flag();
        transfer(0x7e);
        for(j=0;j<2;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
        stop_flag();
    }
}
```

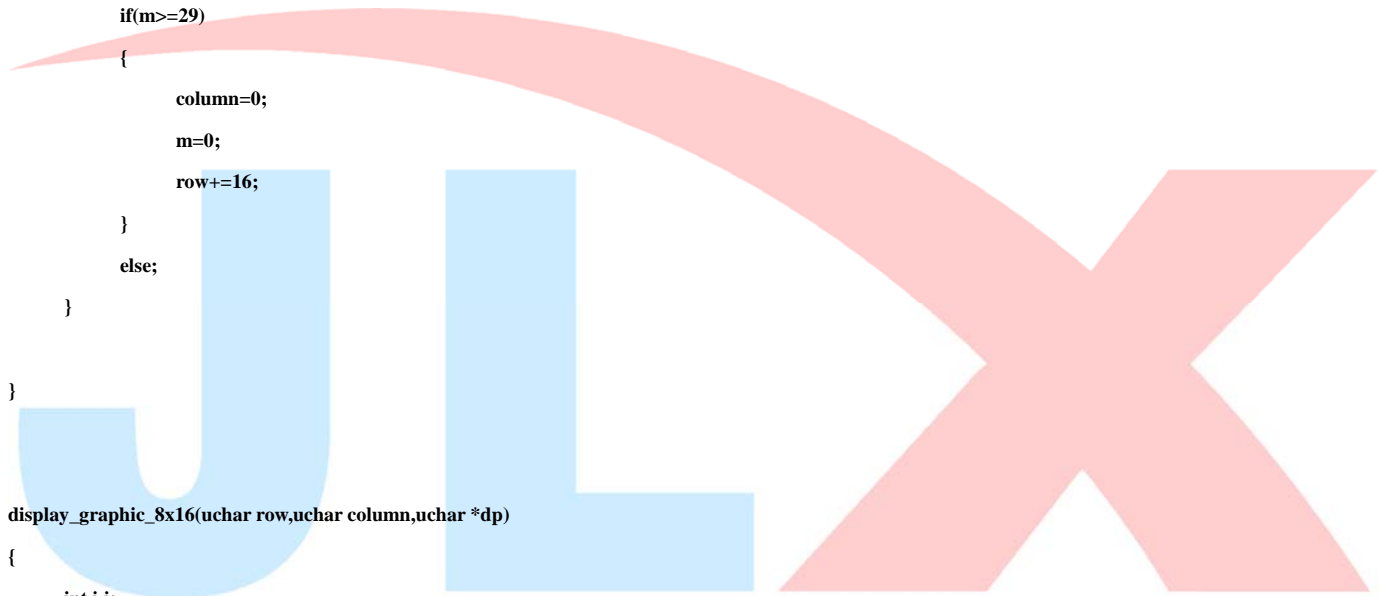
```
void display_N_16x16(uchar row,uchar column,uchar n,uchar *dp)
```

```
{
    int i,j,k,m;
    m=column;
    for(k=0;k<n;k++)
    {
        for(i=0;i<16;i++)
```

```

{
    start_flag();
    transfer(0x7c); //选择 SLAVE ADDRESS
    lcd_address(row+i,column);
    stop_flag();
    start_flag();
    transfer(0x7e);
    for(j=0;j<2;j++)
    {
        write_mono_data(*dp);
        dp++;
    }
    stop_flag();
}
column+=4;
m+=4;
if(m>=29)
{
    column=0;
    m=0;
    row+=16;
}
else;
}
}
display_graphic_8x16(uchar row,uchar column,uchar *dp)
{
    int i,j;
    for(i=0;i<16;i++)
    {
        start_flag();
        transfer(0x7c);
        lcd_address(row+i,column);
        stop_flag();
        start_flag();
        transfer(0x7e);
        for(j=0;j<1;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
        stop_flag();
    }
}
}

```



```
display_graphic_8x8(uchar row,uchar column,uchar *dp)
```

```
{
    int i,j;

    for(i=0;i<8;i++)
    {
        start_flag();
        transfer(0x7c);
        lcd_address(row+i,column);
        stop_flag();
        start_flag();
        transfer(0x7e);
        for(j=0;j<1;j++)
        {
            write_mono_data(*dp);
            dp++;
        }
        stop_flag();
    }
}
```

```
void LCD_INITIAL()
```

```
{
    start_flag();
    transfer(0x7c); //选择 SLAVE ADDRESS
    transfer(0x00); //控制字：表示以下传输的 N 个字节是指令
    transfer(0xE2); //system reset
    delay(10);
    transfer(0x27);
    transfer(0x2b);
    transfer(0x2f); //set pump control
    transfer(0xeb); //set bias=1/11
    transfer(0x81); //set
    transfer(0x36); //set PM=12,vop=12.8v,4c
    transfer(0xa9); //set linerate mux,a2
    transfer(0xc8);
    transfer(0x0b);
    transfer(0x89);
    transfer(0xc4); //MY=1,MX=0:从左到右，再从上到下。
    transfer(0xf1); //f1
    transfer(0x7f);
    transfer(0xd3); //gray shade set
    transfer(0xd7); //gray shade set
    transfer(0xAf); //set display enable
    stop_flag();
}
```

```

void main(void)
{

    LCD_INITIAL();
    while(1)
    {
        clear();
//        display_graphic_16x16(0,1,le1);           /*在第 yy 行，第 xx 列显示单个自编生僻汉字“囧”*/
//        display_graphic_16x16(0,8,jin);         /*显示单个自编生僻汉字"晶"*/
        display_N_16x16(0,0,64,text);
        waitkey();
        clear();
        display_graphic_128x128(0,0,bmp1);
        waitkey();
        display_black();
        waitkey();
        display_odd_column();
        waitkey();
        display_even_column();
        waitkey();
        display_odd_row();
        waitkey();
        display_even_row();
        waitkey();
        display_snow1();
        waitkey();
        display_snow2();
        waitkey();
    }
}

uchar code bmp1[]={//横向取模
/*-- 调入了一幅图像： E:\work\图片收藏夹\黑白屏图片\JLX128128G-328.bmp --*/
/*-- 宽度 x 高度=128x128 --*/
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
0x80,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x01,
,
}

```