

JLX6464G-952-BN 使用说明书

(焊接式 FPC)

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	4~5
5	技术参数	5
6	时序特性	6~7
7	指令功能及硬件接口与编程案例	7~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 JLX6464G-952-BN 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。

JLX6464G-952-BN 可以显示 64 列*64 行点阵单色图片，或显示 16*16 点阵的汉字 4 个*4 行，或显示 8*16 点阵的英文、数字、符号 8 个*4 行。或显示 5*8 点阵的英文、数字、符号 10 个*8 行。

2. JLX6464G-952-BN 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 ST7567A, 功能强大，稳定性好

2.3 功耗低：不带背光 1mW (3.3V*0.18mA)，带背光不大于 100mW (3.3V*30mA)；

2.4 显示内容：

(1) 64*64 点阵单色图片，或其它小于 64*64 点阵的单色图片；

(2) 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 4 字*4 行；

(3) 按照 8*16 点阵汉字来计算可显示 8 字*4 行；

(4) 按照 5*8 点阵汉字来计算可显示 10 字*8 行；

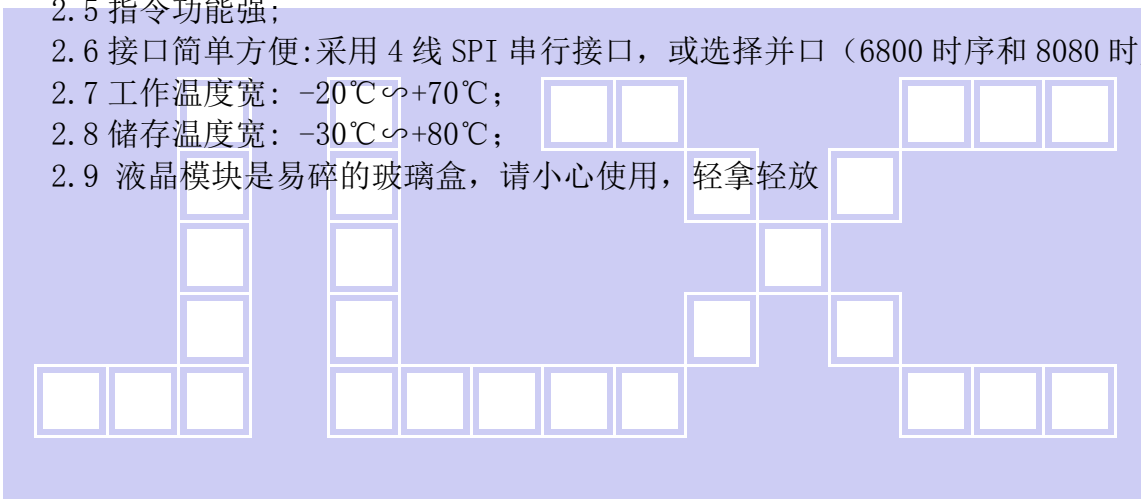
2.5 指令功能强；

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

2.7 工作温度宽：-20℃~+70℃；

2.8 储存温度宽：-30℃~+80℃；

2.9 液晶模块是易碎的玻璃盒，请小心使用，轻拿轻放



3. 外形尺寸及接口引脚功能

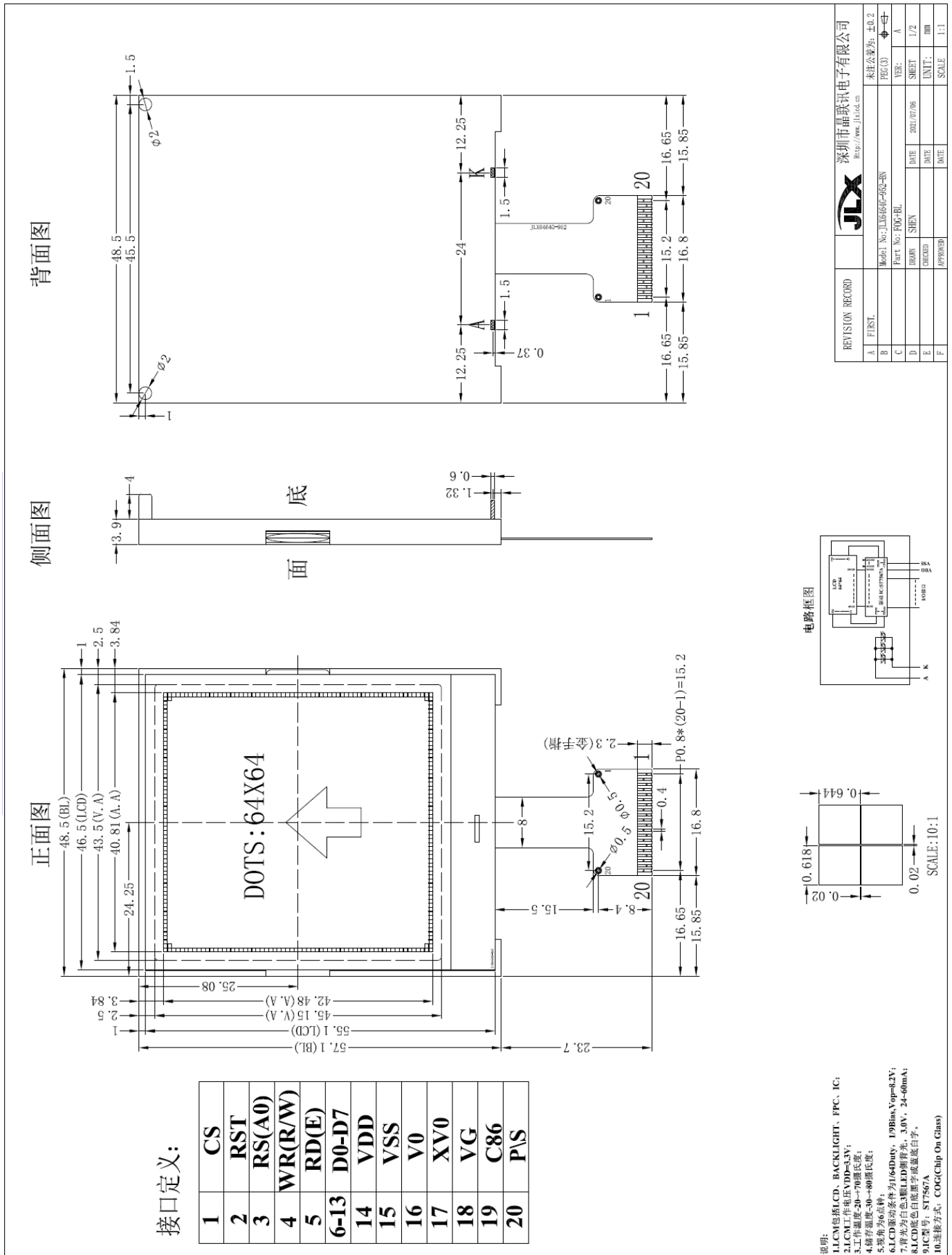


图 1. 外形尺寸

模块的接口引脚功能

引线号	符号	名称	功能
1	CS	片选	低电平片选
2	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
3	RS (即 CD)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器
4	RW (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: 悬空或接 3.3V
5	RD (E)	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: 悬空或接 3.3V
6	D0	I/O	并行接口时: 数据总线 DB0 串行接口时: 串行时钟 (SCLK)
7-9	D1-D3	I/O	并行接口时: 数据总线 DB1、DB2、DB3 串行接口时: 串行数据 (SDA) (D1、D2、D3 短接一起)
10-13	D4-D7	I/O	数据总线 DB4~DB7 串行接口时: 悬空或接 3.3V
14	VDD	供电电源正极	供电电源正极
15	VSS	供电电源负极	供电电源负极
16	V0	偏置电压	
17	XV0	偏置电压	
18	VG	LCD 倍压输出	
19	C86	选择 6800 或 8080	并行接口时: H: 6800 系统, L: 8080 系统。串行接口时: H
20	PS	选串并控制接口	接 VDD: 选择并行接口, 接 VSS: 选择串行接口

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 64×64 点阵, 64 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 工作电路框图:

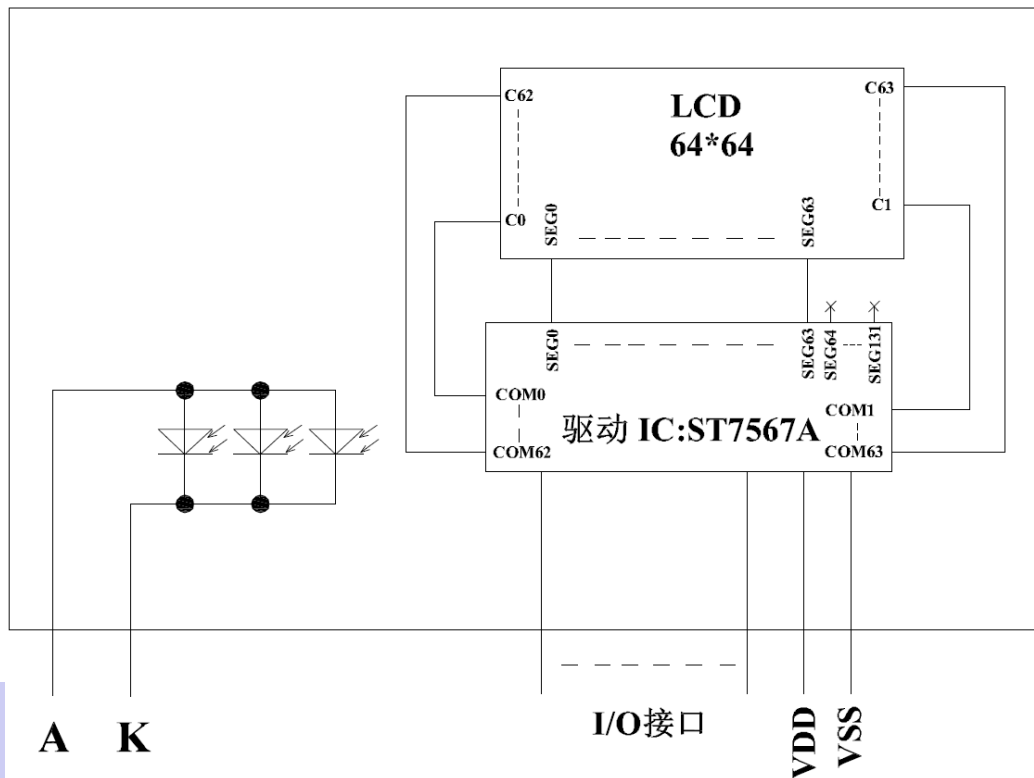


图 2: JLX6464G-952-BN 图像点阵型液晶模块的电路框图

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

存储温度: $-30^{\circ}\text{C} \sim +80^{\circ}\text{C}$;

背光白色;

正常工作电流为: $24 \sim 60\text{mA}$ (LED 灯数共 3 颗);

工作电压: 3.0V ; (串 20 欧电阻接 3.3V)

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	
电源	VDD - VSS	-0.3		3.6	V
工作温度		-20		+70	$^{\circ}\text{C}$
储存温度		-30		+80	$^{\circ}\text{C}$

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		2.6	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 _{IH}	-	0.8VDD	-	VDD	V
输入低电平	V _{I0}	-	0	-	0.2VDD	V
输出高电平	V _{OH}	I _{OH} = 0.2mA	0.8VDD		VDD	V

输出低电平	V ₀₀	I ₀₀ = 1.2mA	0	-	0.2V _{DD}	V
模块工作电流	I _{DD}	V _{DD} = 3.0V	-	0.3	1.0	mA
背光工作电流	I _{LED}	V _{LED} =3.0V (共 3 颗 LED 灯并联)	16	30	40	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 ST7567A (Writing Data from CPU to ST7567A)

System Bus Timing for 4-Line Serial Interface

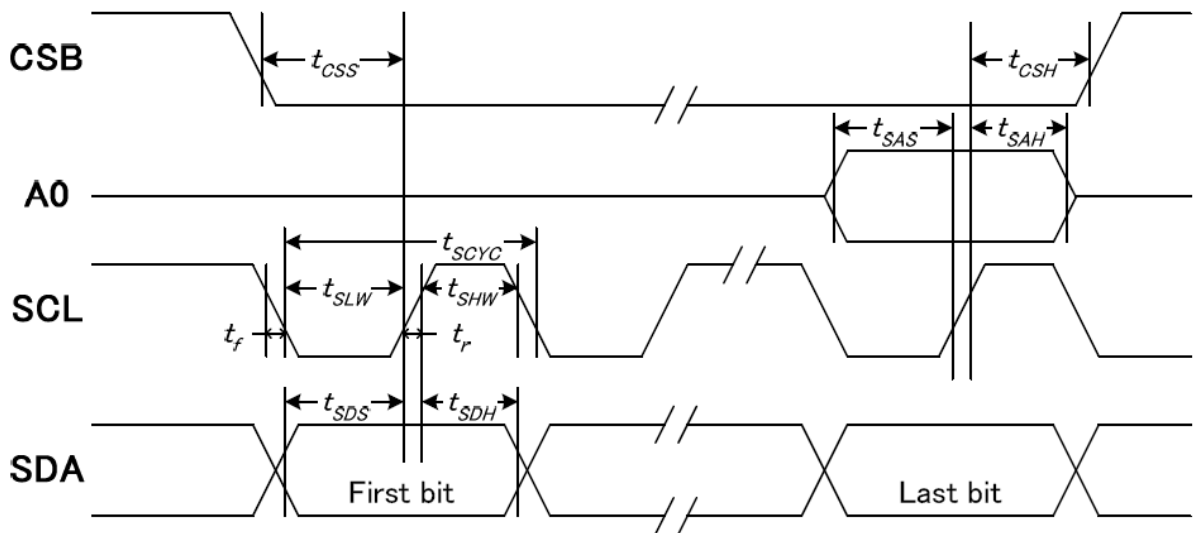


图 3. 从 CPU 写到 ST7567A (Writing Data from CPU to ST7567A)

6.2 串行接口: 时序要求 (AC 参数):

写数据到 ST7567A 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	—	—	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 = 3.3V, Ta = 25°C

6.3 并行接口:

从 CPU 写到 ST7567A (Writing Data from CPU to ST7567A)

14-1 System Bus Timing for 6800 Series MPU

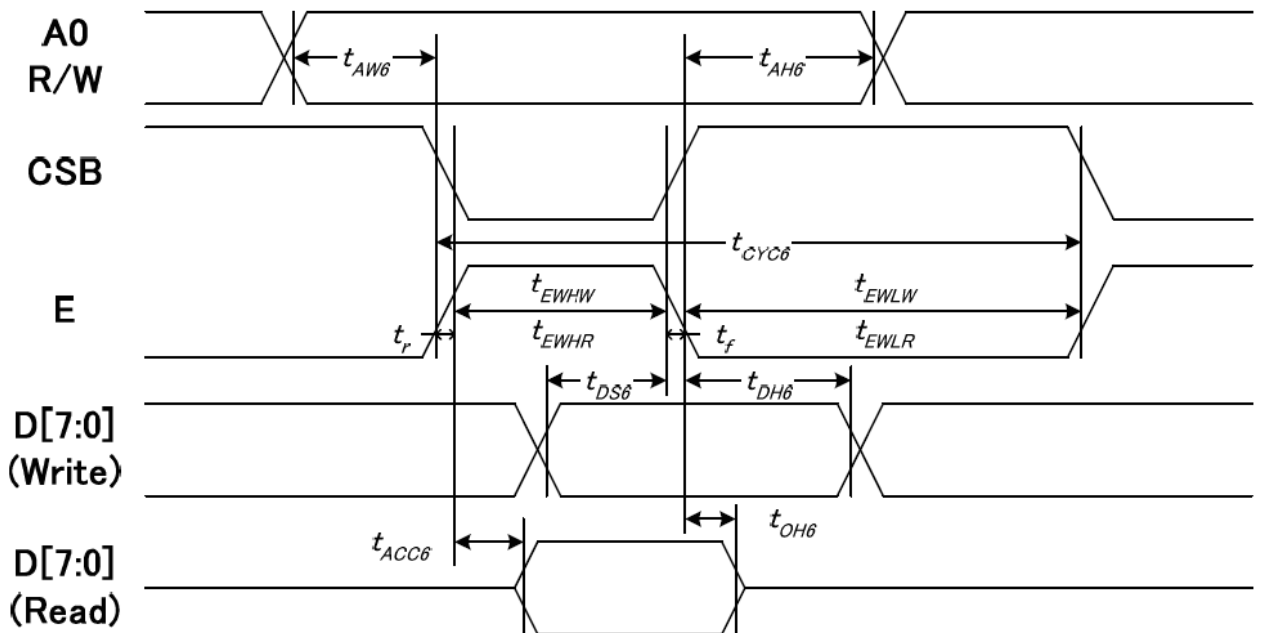


图 4. 从 CPU 写到 ST7567A (Writing Data from CPU to ST7567A)

写数据到 ST7567A 的时序要求: (6800 系列 MPU)

表 5.

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	10	—	—	
系统循环时间		tCYC8	240	—	—	
使能“低”脉冲(写)	WR	tCCLW	80	—	—	
使能“高”脉冲(写)		tCCHW	80	—	—	
使能“低”脉冲(读)	RD	tCCLR	140	—	—	
使能“高”脉冲(读)		tCCHR	80	—	—	
写数据建立时间	D7-D0	tDS8	40	—	—	
写数据保持时间		tDH8	10	—	—	
读时间		tACC8	—	—	70	
读输出允许时间		tOH8	5	—	50	

VDD=3.3V, Ta=25°C

System Bus Timing for 8080 Series MPU

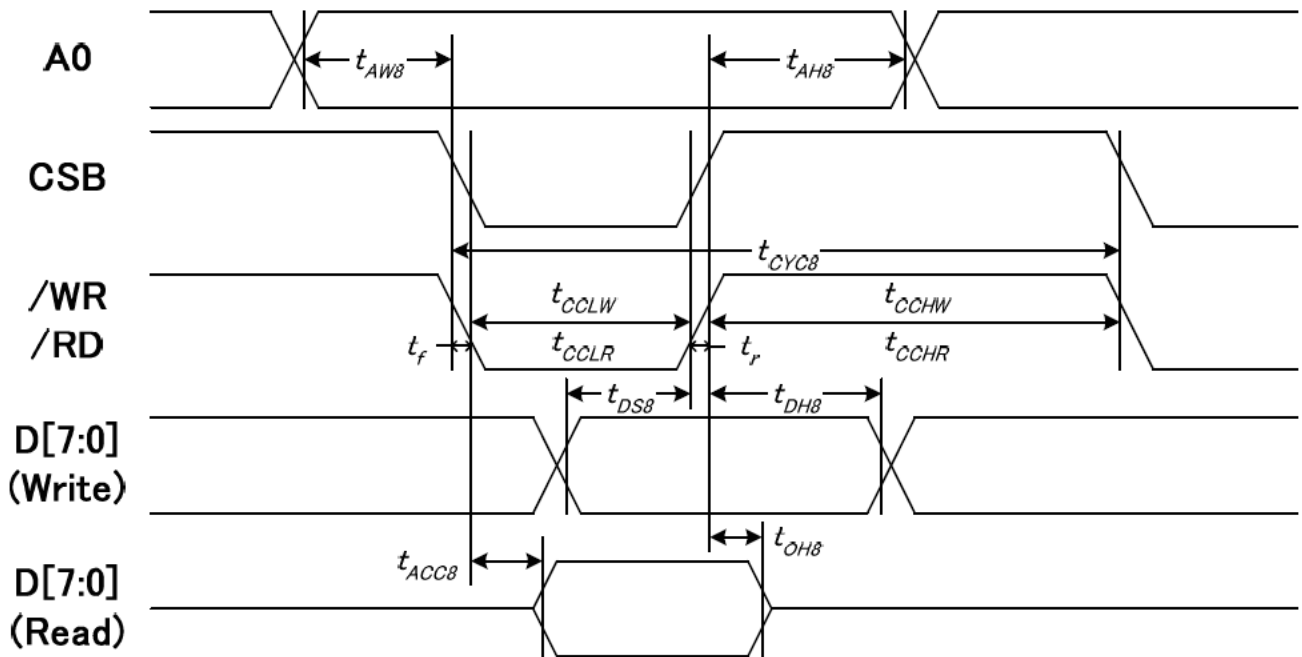


图 5. 从 CPU 写到 ST7567A (Writing Data from CPU to ST7567A)

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)

表 6.

项目	符号	测试条件	极限值			单位
			MIN	TYPE	MAX	
地址保持时间	A0	tAH6	10	—	—	ns
地址建立时间		tAW6	0	—	—	
系统循环时间		tCYC6	240	—	—	
使能“低”脉冲(写)	WR	tEWLW	80	—	—	
使能“高”脉冲(写)		tEWHW	80	—	—	
使能“低”脉冲(读)	RD	tEWL R	140	—	—	
使能“高”脉冲(读)		tEWH R	80	—	—	
写数据建立时间	D7-D0	tDS6	40	—	—	
写数据保持时间		tDH6	20	—	—	
读时间		tACC6	—	—	70	
读输出允许时间		tOH6	5	—	50	

VDD=3.3V, Ta=25°C

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

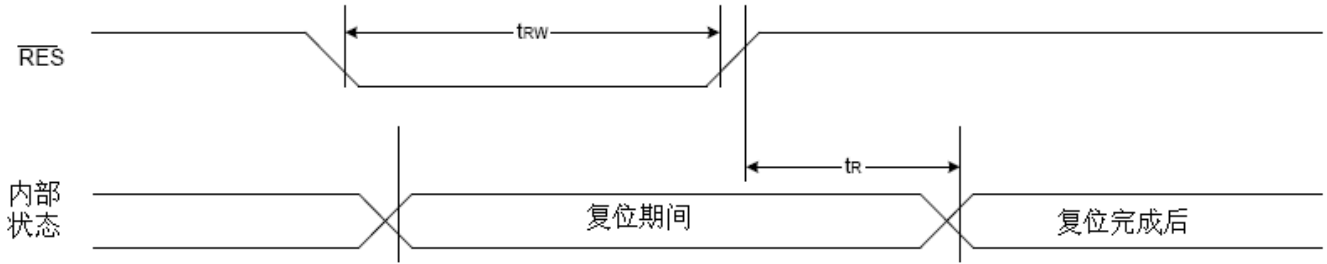


图 6: 电源启动后复位的时序

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	tr		100	—	200	ms
复位保持低电平的时间	trw	引脚: RES	200	—	—	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) 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 0x40
(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	并口时: 读驱动IC的当前状态, 串口时不能用此指令。 本液晶模块使用串行接口, 不具备此功能。
(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								并口时: 读已经显示到液晶屏上的点阵数据。串口时不能用此指令。 本液晶模块使用串行接口, 不具备此功能。
(8) 显示列地址增减 (ADC select)			1	0	1	0	0	0	0	0 1	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	0	显示正显/反显: 1 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	0	显示全部点阵: 1 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	0	设置偏压比: 1 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	0	0XE0 : “读-改-写” 开始。 本液晶模块使用串行接口, 不具备此功能。 详情请参考IC资料
(13) 退出上述“读-改-写”指令(End)	0	1	1	1	0	1	1	1	0	0xEE : 上述“读-改-写”指令结束 本液晶模块使用串行接口, 不具备此功能。 详情请参考 IC 资料
(14) 软件复位 (Reset)	0	1	1	1	0	0	0	1	0	0XE2 : 软件复位。
(15) 行扫描顺序选择 (Common output mode select)		1	1	0	0	0	0	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				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	静态图标的开关设置: 1 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	0	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

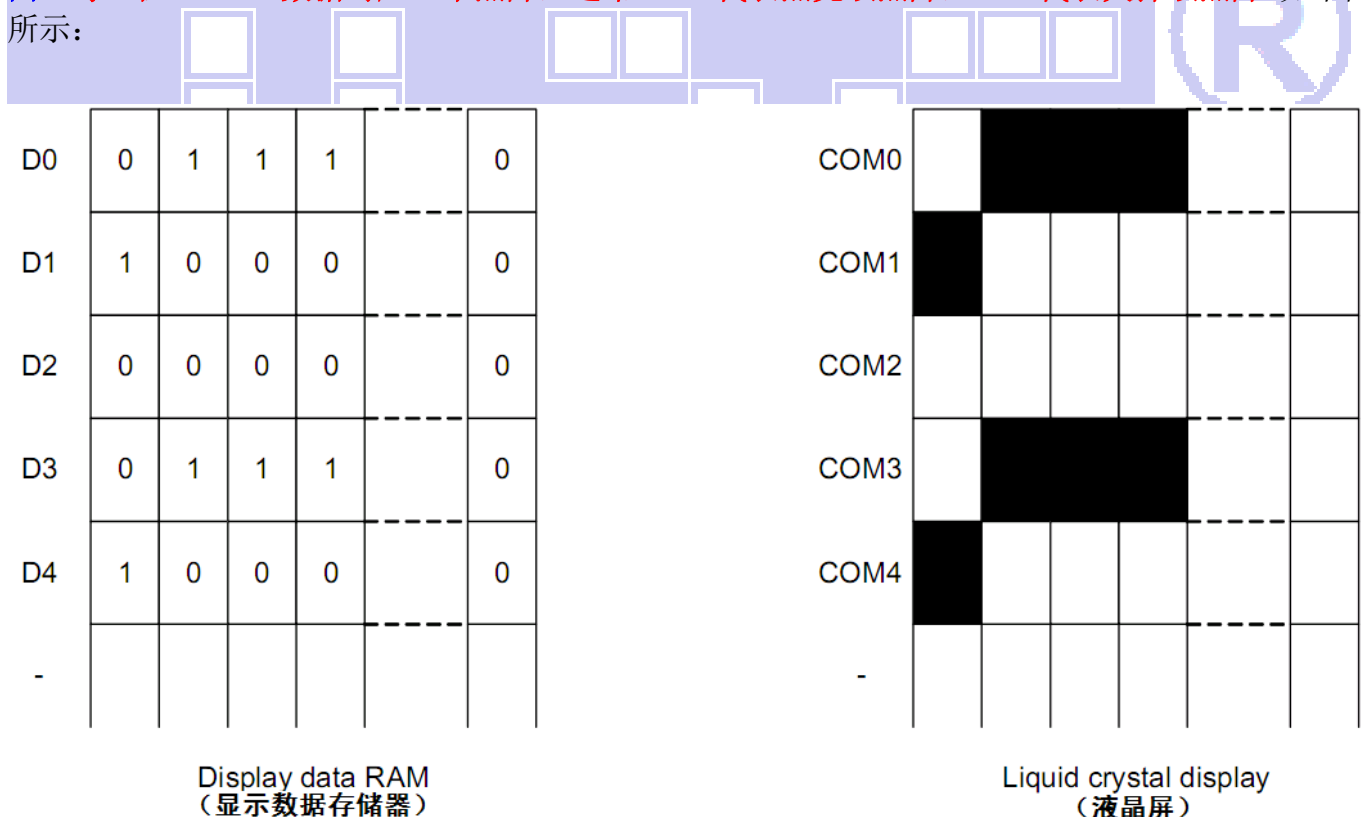
(21) 省电模式 (Power save)										省电模式, 此非一条指令, 是由“(10)显示全部点阵”、(19)静态图标显示: 开/关等指令合成一个“省电功能”。详细看 IC 规格书 “POWER SAVE”部分
(22) 空指令 (NOP)	0	1	1	1	0	0	0	1	1	空操作
(23) 测试 (Test)	0	1	1	1	1	*	*	*	*	内部测试用, 千万别用!

请详细参考 IC 资料”ST7567A_V1.1a.PDF”的第 37~49 页。

7.2 点阵与 DD RAM(显示数据存储器)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 64*64 点阵的屏分为 8 个“页”, 从第 0“页”到第 7“页”。

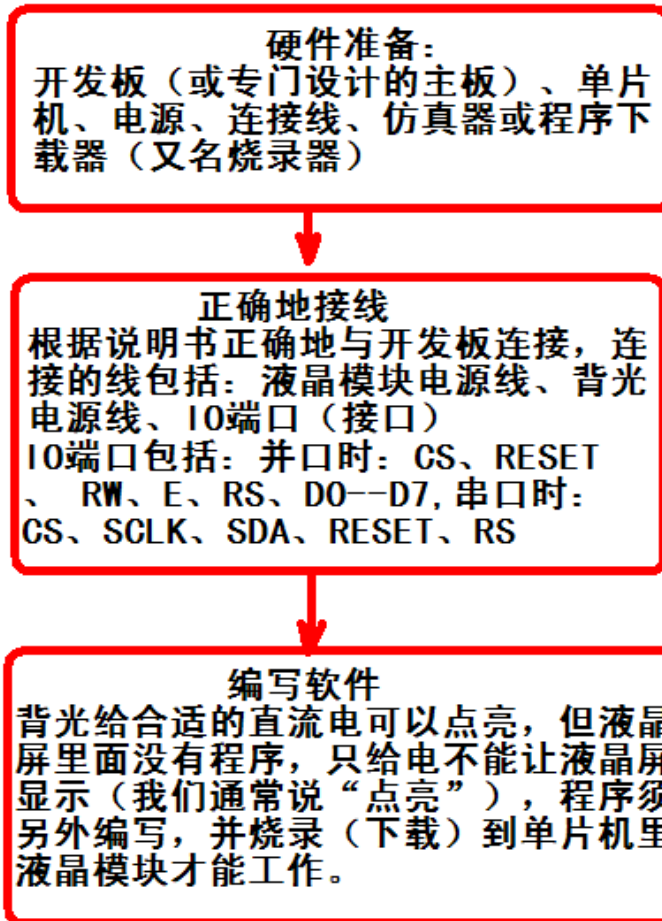
DB7--DB0 的排列方向: 数据是从下向上排列的。最低位 D0 是在最上面, 最高位 D7 是在最下面。每一位 (bit) 数据对应一个点阵, 通常“1”代表点亮该点阵, “0”代表关掉该点阵。如下图所示:



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。

点亮液晶模块的步骤

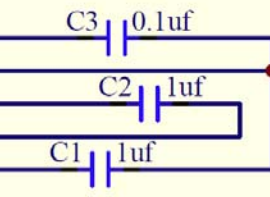
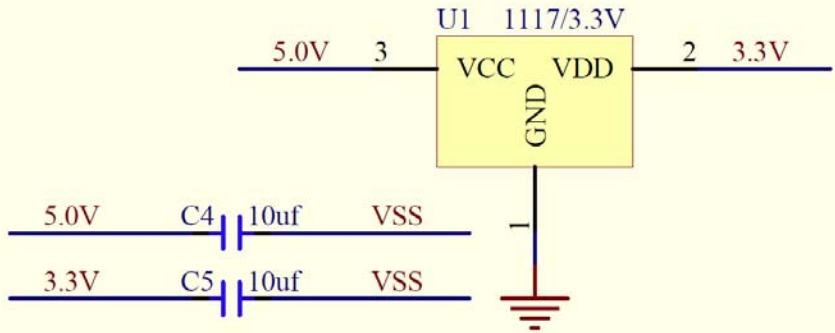


7.4 程序举例:

液晶模块与 MPU (以 8051 系列单片机为例) 接口图如下:

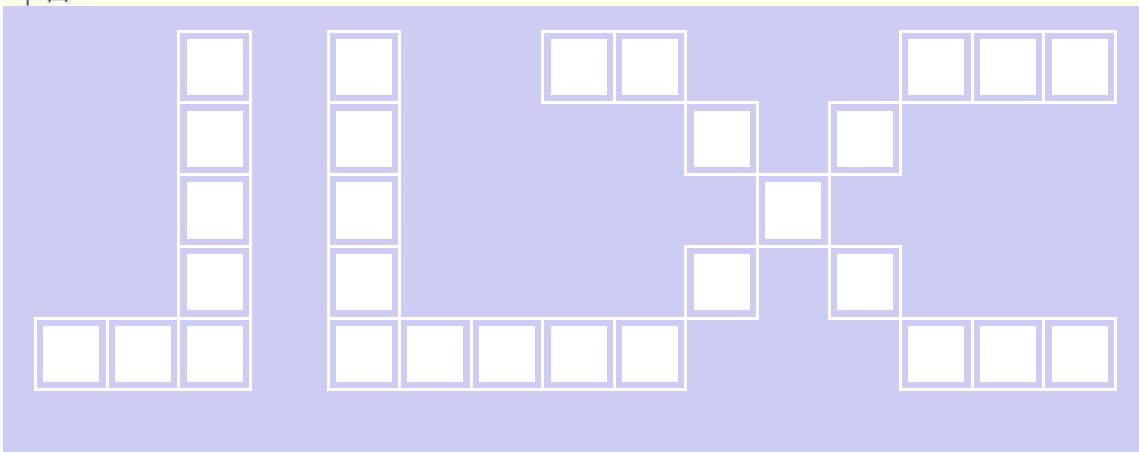
20PIN

CS	1	CS
RST	2	RST
RS	3	RS
WR	4	3.3V
E	5	3.3V
D0	6	SCK
D1	7	SDA
D2	8	SDA
D3	9	SDA
D4	10	3.3V
D5	11	3.3V
D6	12	3.3V
D7	13	3.3V
VDD	14	3.3V
VSS	15	VSS
V0	16	V0
XV0	17	XV0
VG	18	VG
C86	19	3.3V
PS	20	VSS



所有电容的耐压值均选耐压为25V以上
 液晶模块使用电压: 2.7V-3.5V
 CPU是5V时, 用1117/3.3把电压转为3.3V
 CPU是3.3V时, 直接使用

串口



点亮液晶模块的编程步骤

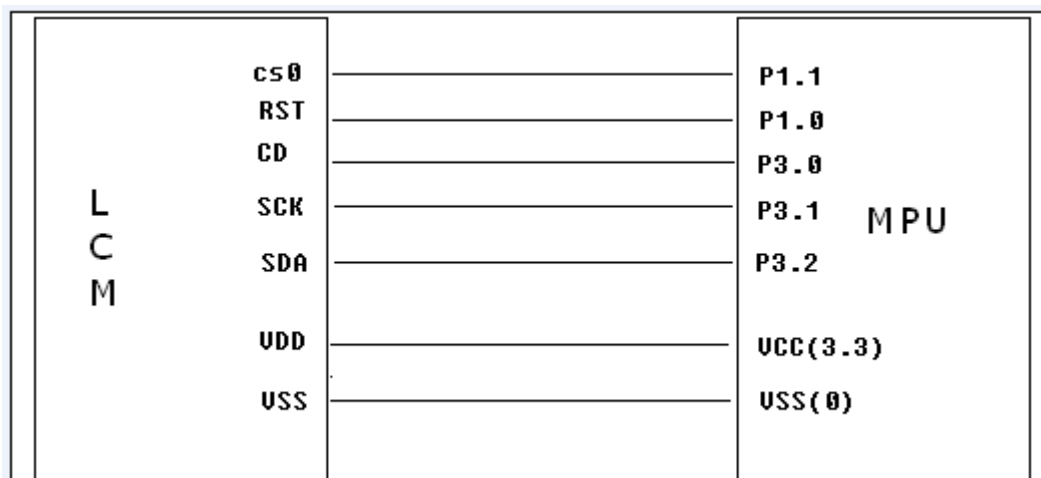


图 7. 串行接口


```
0x00, 0x00, 0x00, 0xFE, 0xFC, 0x0C, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x0C, 0x04, 0x04,
0x04, 0x84, 0xFF, 0xFF, 0x04, 0x04, 0x04, 0x04, 0x04, 0xFE, 0xFE, 0x04, 0x00, 0x00, 0x00, 0x00,
0xC0, 0xC0, 0xC0, 0xC0, 0xE0, 0x60, 0x7F, 0x3F, 0x30, 0x30, 0x10, 0x18, 0x18, 0x88, 0xC0, 0xF8,
0x7F, 0x1F, 0x01, 0x00, 0x00, 0x00, 0x00, 0xF8, 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x01,
0x01, 0x00, 0x00, 0x00, 0x20, 0x20, 0x30, 0x18, 0x0C, 0x0C, 0x07, 0x03, 0x01, 0x00, 0x04,
0x04, 0x0C, 0x0C, 0x1C, 0x38, 0x3C, 0x1F, 0x0F, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};
```

```
uchar 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, 0xFF, 0x20, 0xE1, 0x26, 0x2C, 0x20, 0x20, 0x00,
0x04, 0x02, 0x01, 0xFF, 0x40, 0x20, 0x18, 0x07, 0x00, 0x00, 0x03, 0x0C, 0x30, 0x60, 0x20, 0x00};
```

```
uchar 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};
```

```
uchar code shi[]={
/*-- 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};
```

```
uchar 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, 0xFE, 0x22, 0x22, 0x22, 0x22, 0xFE, 0x00, 0x00,
0x80, 0x40, 0x30, 0x0F, 0x02, 0x02, 0x02, 0xFF, 0x02, 0x02, 0x42, 0x82, 0x7F, 0x00, 0x00};
```

```
uchar code mao_hao[]={
/*-- 文字: : (冒号) --*/
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
0x00, 0x00, 0x00, 0xC0, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 0x00, 0x00};
```

```
char code num0[]={
/*-- 文字: 0 --*/
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x0F, 0x10, 0x20, 0x20, 0x10, 0x0F, 0x00
};
```

```
char code num1[]={
/*-- 文字: 1 --*/
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```





```
0x00, 0x10, 0x10, 0x08, 0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x3F, 0x20, 0x20, 0x00, 0x00
```

```
};
```

```
char code num2[]={
```

```
/*-- 文字: 2 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0x70, 0x08, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x30, 0x28, 0x24, 0x22, 0x21, 0x30, 0x00
```

```
};
```

```
char code num3[]={
```

```
/*-- 文字: 3 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0x30, 0x08, 0x88, 0x88, 0x48, 0x30, 0x00, 0x00, 0x18, 0x20, 0x20, 0x20, 0x11, 0x0E, 0x00
```

```
};
```

```
char code num4[]={
```

```
/*-- 文字: 4 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0x00, 0x0C, 0x20, 0x10, 0x08, 0x00, 0x00, 0x00, 0x07, 0x04, 0x24, 0x24, 0x3F, 0x24, 0x00
```

```
};
```

```
char code num5[]={
```

```
/*-- 文字: 5 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0xF8, 0x08, 0x88, 0x88, 0x08, 0x08, 0x00, 0x00, 0x19, 0x21, 0x20, 0x20, 0x11, 0x0E, 0x00
```

```
};
```

```
char code num6[]={
```

```
/*-- 文字: 6 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0xE0, 0x10, 0x88, 0x88, 0x18, 0x00, 0x00, 0x00, 0x0F, 0x11, 0x20, 0x20, 0x11, 0x0E, 0x00
```

```
};
```

```
char code num7[]={
```

```
/*-- 文字: 7 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0x38, 0x08, 0x08, 0x38, 0x08, 0x00, 0x00, 0x00, 0x00, 0x3F, 0x00, 0x00, 0x00, 0x00
```

```
};
```

```
char code num8[]={
```

```
/*-- 文字: 8 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0x70, 0x88, 0x08, 0x08, 0x88, 0x70, 0x00, 0x00, 0x1C, 0x22, 0x21, 0x21, 0x22, 0x1C, 0x00
```

```
};
```

```
char code num9[]={
```

```
/*-- 文字: 9 --*/
```

```
/*-- 宋体 12: 此字体下对应的点阵为: 宽 x 高=8x16 --*/
```

```
0x00, 0xE0, 0x10, 0x08, 0x08, 0x10, 0xE0, 0x00, 0x00, 0x00, 0x31, 0x22, 0x22, 0x11, 0x0F, 0x00
```

```
};
```

```
/*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;
//        delay_us(2);
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        delay_us(2);
        data1=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;
//        delay_us(1);
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
//        delay_us(1);
        data1=data1<<=1;
    }
    cs1=1;
}
```

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

```
/*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(3500);
}

/*LCD 模块初始化*/
void initial_lcd()
{
    reset=0;          /*低电平复位*/
    delay(100);       //100ms
    reset=1;          /*复位完毕*/
    delay(200);       //200ms
    transfer_command(0xe2); /*软复位*/
    delay(5);         //5ms
    transfer_command(0xae); /*开显示*/
    transfer_command(0x2c); /*升压步聚 1*/
    delay(5);         //5ms
    transfer_command(0x2e); /*升压步聚 2*/
    delay(5);         //5ms
    transfer_command(0x2f); /*升压步聚 3*/
    delay(5);         //5ms
    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(0x19); /*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 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++;
        }
    }
}

```

```
void display_string_8x16(uint page, uint column, uchar *text)
```

```

{
    uint i=0, j, k, n;
    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]); /*显示 5x7 的 ASCII 字到 LCD 上, y 为页地址, x 为列地址, 最后为数据*/
                }
            }
            i++;
            column+=8;
        }
        else
            i++;
    }
}

```

```
void display_string_5x7(uint page, uint column, uchar *text)
```

```
{
```

```

uint i=0, j, k;
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++)
        {
            transfer_data(ascii_table_5x7[j][k]);/*显示 5x7 的 ASCII 字到 LCD 上, y 为页地址, x 为列地址, 最后为数据*/
        }
        i++;
        column+=6;
    }
    else
        i++;
}
}

```

```

//=====display a picture of 128*64 dots=====

```

```

void disp_grap(uchar page, uchar column, uchar *dp)

```

```

{
    int i, j;
    for(i=0;i<8;i++)
    {
        lcd_address(page+i, column);
        for(j=0;j<64;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}

```

```

void main(void)

```

```

{
    while(1)
    {
        initial_lcd();
        clear_screen();
        disp_grap(1, 1, graphic2); //display a picture of 128*64 dots
        waitkey();
        disp_grap(1, 1, graphic3); //display a picture of 128*64 dots
        waitkey();
        clear_screen(); //clear all dots
        display_graphic_32x32(3, 32*0, cheng1); /*在第 1 页, 第 49 列显示单个汉字“成”*/
    }
}

```



```

display_graphic_32x32(3, 32*1, gong1);          /*在第 1 页，第 49 列显示单个汉字“功”*/
waitkey();
clear_screen();
display_graphic_16x16(1, 1, 0, zhuang1);        /*在第 5 页，第 1 列显示单个汉字“状”*/
display_graphic_16x16(1, (1+16), 0, tail);      /*在第 5 页，第 17 列显示单个汉字“态”*/
display_graphic_8x16(1, (1+16*2), mao_hao);    /*在第 5 页，第 25 列显示单个字符“:”*/
display_graphic_16x16(4, (1+16*1), 1, shi1);   /*在第 5 页，第 41 列显示单个汉字“使”*/
display_graphic_16x16(4, (1+16*2), 1, yong1);  /*在第 5 页，第 49 列显示单个汉字“用”*/
display_graphic_8x16(7, (24), num0);           /*在第 5 页，第 89 列显示单个数字“0”*/
display_graphic_8x16(7, (24+8*1), num0);      /*在第 5 页，第 97 列显示单个数字“0”*/
display_graphic_8x16(7, (24+8*2), mao_hao);   /*在第 5 页，第 105 列显示单个字符“:”*/
display_graphic_8x16(7, (24+8*3), num0);      /*在第 5 页，第 113 列显示单个数字“0”*/
display_graphic_8x16(7, (24+8*4), num0);      /*在第 5 页，第 121 列显示单个数字“0”*/
waitkey();
clear_screen();
disp_grap(1, 1, graphic1); //display a picture of 128*64 dots
waitkey();

```

```

}
}

char code graphic1[]={
/*--- ??????:E: ?????\????\JLX6464G-952. bmp ---*/
/*--- ??x??=64x64 ---*/
0xFF, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0xFD, 0x25, 0x25, 0x25,
0x25, 0x25, 0xFD, 0x01, 0x01, 0x01, 0x01, 0x01, 0x09, 0xF9, 0x49, 0x49, 0xF9, 0x09, 0x01, 0x41,
0x45, 0x59, 0xC1, 0x51, 0x4D, 0x41, 0x01, 0x01, 0x01, 0x01, 0x09, 0x31, 0x01, 0x09, 0x09, 0xF9,
0x09, 0x09, 0x09, 0xF9, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFC, 0x24, 0x24, 0x25, 0x25, 0xFD, 0x01,
0xFD, 0x25, 0x25, 0x24, 0x24, 0xFC, 0x00, 0x00, 0x40, 0x7F, 0x22, 0x22, 0xFF, 0x10, 0x04, 0x04,
0xC4, 0x34, 0x0F, 0x34, 0xC4, 0x04, 0x04, 0x00, 0x01, 0x01, 0x01, 0xFF, 0x80, 0x42, 0x02, 0xFF,
0x02, 0x02, 0x00, 0x3F, 0xC0, 0x00, 0xE0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x01, 0x01, 0x01, 0xC1, 0x23, 0x20,
0x23, 0xE1, 0x01, 0x01, 0x01, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0xE0, 0x22, 0x21,
0x20, 0xE0, 0x20, 0x00, 0x00, 0xC1, 0x22, 0x20, 0x20, 0x20, 0xC0, 0x01, 0x00, 0x80, 0x40, 0x23,
0x20, 0x40, 0x80, 0x00, 0x00, 0x01, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x60, 0x80, 0x80,
0x88, 0x78, 0x08, 0x00, 0x00, 0x04, 0x04, 0x04, 0x04, 0x04, 0x04, 0x00, 0x00, 0x00, 0xFC,
0x03, 0x00, 0x00, 0x00, 0x00, 0xC1, 0xA0, 0x90, 0x88, 0x86, 0xC1, 0x00, 0x00, 0x3F, 0x40, 0x80,
0x80, 0x40, 0x3F, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xC0, 0x20, 0x10, 0x10,
0x30, 0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20, 0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0xC0, 0x20, 0x10, 0x10, 0x30, 0x00, 0x00, 0x00, 0x80, 0x40, 0x20,
0xF0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x1F, 0x22, 0x41, 0x41,
0x22, 0x1C, 0x00, 0x00, 0x0E, 0x09, 0x48, 0x48, 0x7F, 0x48, 0x00, 0x00, 0x41, 0x63, 0x5C, 0x1D,
0x63, 0x41, 0x00, 0x00, 0x1F, 0x22, 0x41, 0x41, 0x22, 0x1C, 0x00, 0x00, 0x0E, 0x09, 0x48, 0x48,
0x7F, 0x48, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,

```





```

0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X80, 0XC0, 0X40,
0X20, 0X10, 0X10, 0X10, 0X08, 0X08, 0X04, 0X04, 0X04, 0X02, 0X02, 0X02, 0X42, 0XE2, 0X02,
0XA2, 0XA2, 0XE2, 0X02, 0X02, 0X02, 0X04, 0X04, 0X04, 0X04, 0X08, 0X08, 0X10, 0X10, 0X20, 0X20,
0X40, 0XC0, 0X80, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0XC0, 0X20, 0X10, 0X08, 0X04, 0X02, 0X01, 0X00, 0X00, 0X00,
0X02, 0X1F, 0X00, 0X00, 0X02, 0X1F, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X43, 0X60,
0XF3, 0X62, 0X42, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X02, 0X1F, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X01, 0X02, 0X04, 0X08, 0X10, 0X60, 0X80, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X80, 0X70, 0X0C, 0X03, 0X00, 0X00, 0X04, 0X3E, 0X00, 0X3E, 0X22, 0X3E, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0XFF, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X3A, 0X2A, 0X2E, 0X00, 0X00, 0X00, 0X03, 0X0C, 0X70, 0X80, 0X00, 0X00,
0X00, 0XFC, 0X03, 0X00, 0X00, 0XE0, 0XA0, 0XE0, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X80, 0XC0,
0X7F, 0XC0, 0X80, 0X80, 0X80, 0X80, 0X80, 0X80, 0X80, 0X80, 0X80, 0X80, 0X80, 0X80, 0XE0, 0XC0,
0X80, 0X00, 0X00, 0X00, 0X00, 0X00, 0X20, 0XA0, 0XE0, 0X00, 0X00, 0X00, 0X07, 0XF8, 0X00,
0X00, 0X1F, 0XE0, 0X00, 0X00, 0X02, 0X02, 0X03, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X01,
0X03, 0X01, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X03, 0X01,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X02, 0X02, 0X03, 0X00, 0X00, 0X00, 0XE0, 0X1F, 0X00,
0X00, 0X00, 0X01, 0X0E, 0X30, 0XC0, 0X00, 0X00, 0X00, 0X7C, 0X54, 0X7C, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X02, 0X1F, 0X00, 0X17, 0X15, 0X1D, 0X00, 0X0A,
0X00, 0X1F, 0X11, 0X1F, 0X00, 0X1F, 0X11, 0X1F, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X18, 0X14, 0X3E, 0X10, 0X00, 0X00, 0X00, 0XC0, 0X30, 0X0E, 0X01, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X01, 0X06, 0X08, 0X10, 0X20, 0X40, 0X80, 0X00, 0X00, 0X00,
0X00, 0X00, 0X08, 0XE8, 0X18, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0XC0,
0X40, 0X40, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0XB8, 0XA8, 0XE8, 0X00, 0X00,
0X00, 0X00, 0X00, 0X80, 0X40, 0X20, 0X10, 0X08, 0X06, 0X01, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X01, 0X01, 0X02,
0X04, 0X04, 0X08, 0X08, 0X10, 0X10, 0X10, 0X20, 0X20, 0X20, 0X20, 0X40, 0X40, 0X40, 0X40, 0X47,
0X45, 0X47, 0X40, 0X40, 0X40, 0X20, 0X20, 0X20, 0X20, 0X10, 0X10, 0X10, 0X08, 0X08, 0X04, 0X04,
0X02, 0X01, 0X01, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00, 0X00,
};

```

```

uchar 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, 0x00, // !- ASCII 码: 0X21
0x00, 0x0E, 0x1E, 0x00, 0x00, 0x1E, 0x0E, 0x00, 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, 0xCC, 0x98, 0x00, 0x06, 0x0C, 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, 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, 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, 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, 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, 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, 0x00, 0x06, 0x06, 0x00, 0x00, 0x00, //:-:-	
0x00, 0x00, 0x00, 0x30, 0x30, 0x00, 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, 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, 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, 0xC0, 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, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, //_-
 0x00, 0x00, 0x03, 0x07, 0x04, 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, 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, 0x00, 0x00, 0x08, 0x0F, 0x0F, 0x08, 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, 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, //~-



};

```
uchar code ascii_table_5x7[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-
```

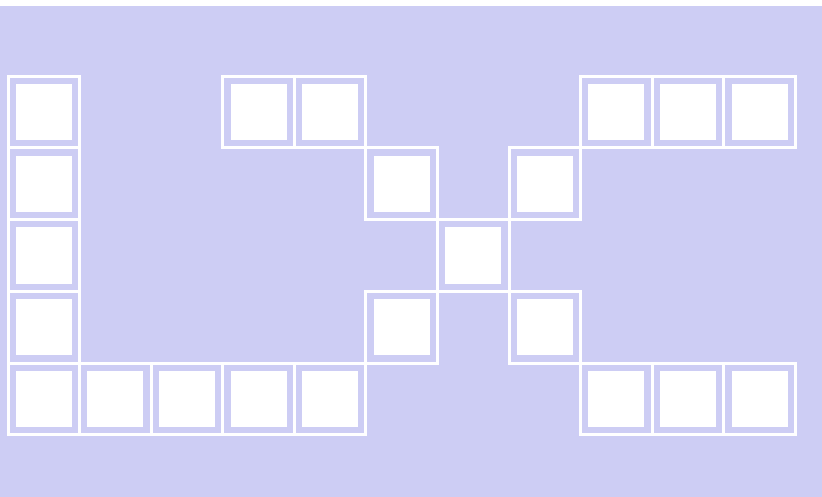
```
0x00, 0x41, 0x7f, 0x41, 0x00, //-I-
```

```
0x00, 0x00, 0x00, 0x00, 0x00, //-J-
```

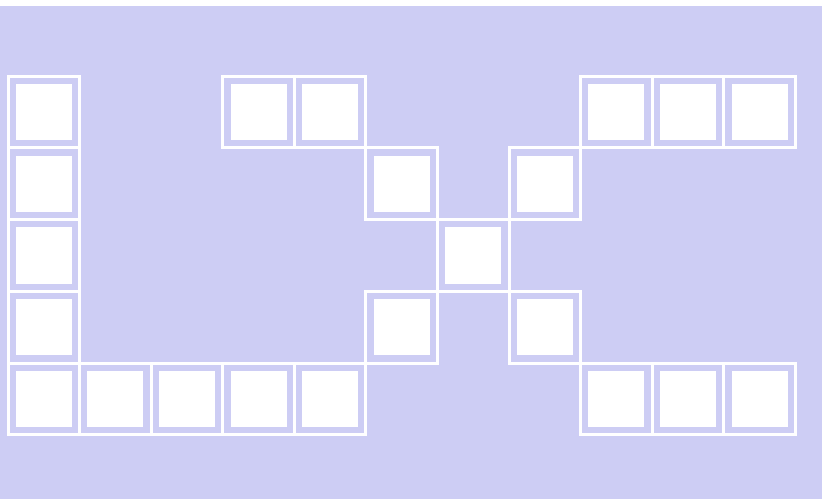
```
0x00, 0x00, 0x00, 0x00, 0x00, //-K-
```

```
0x00, 0x00, 0x00, 0x00, 0x00, //-L-
```

```
0x00, 0x00, 0x00, 0x00, 0x00, //-M-
```



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, //-~-
};
    
```

并行接口

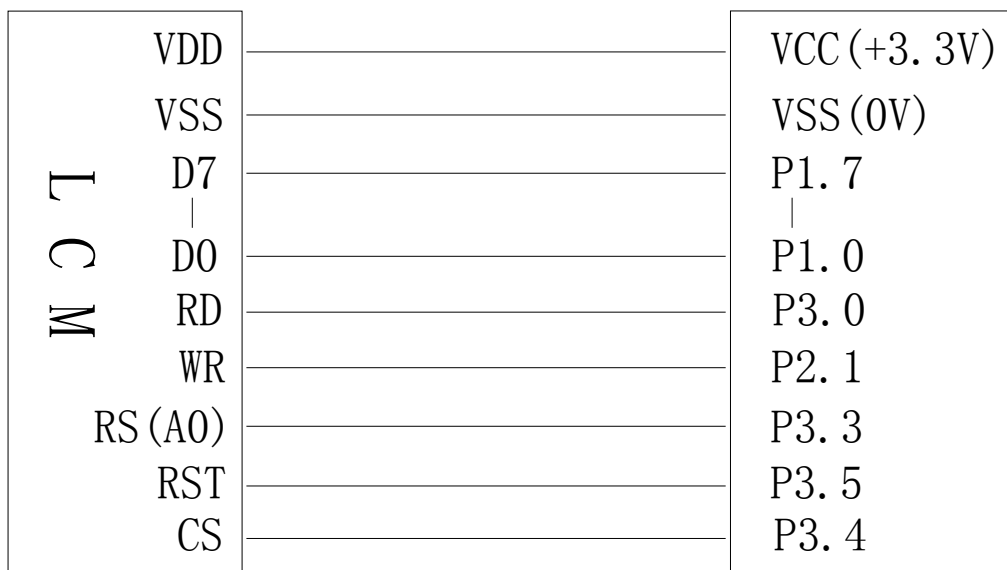
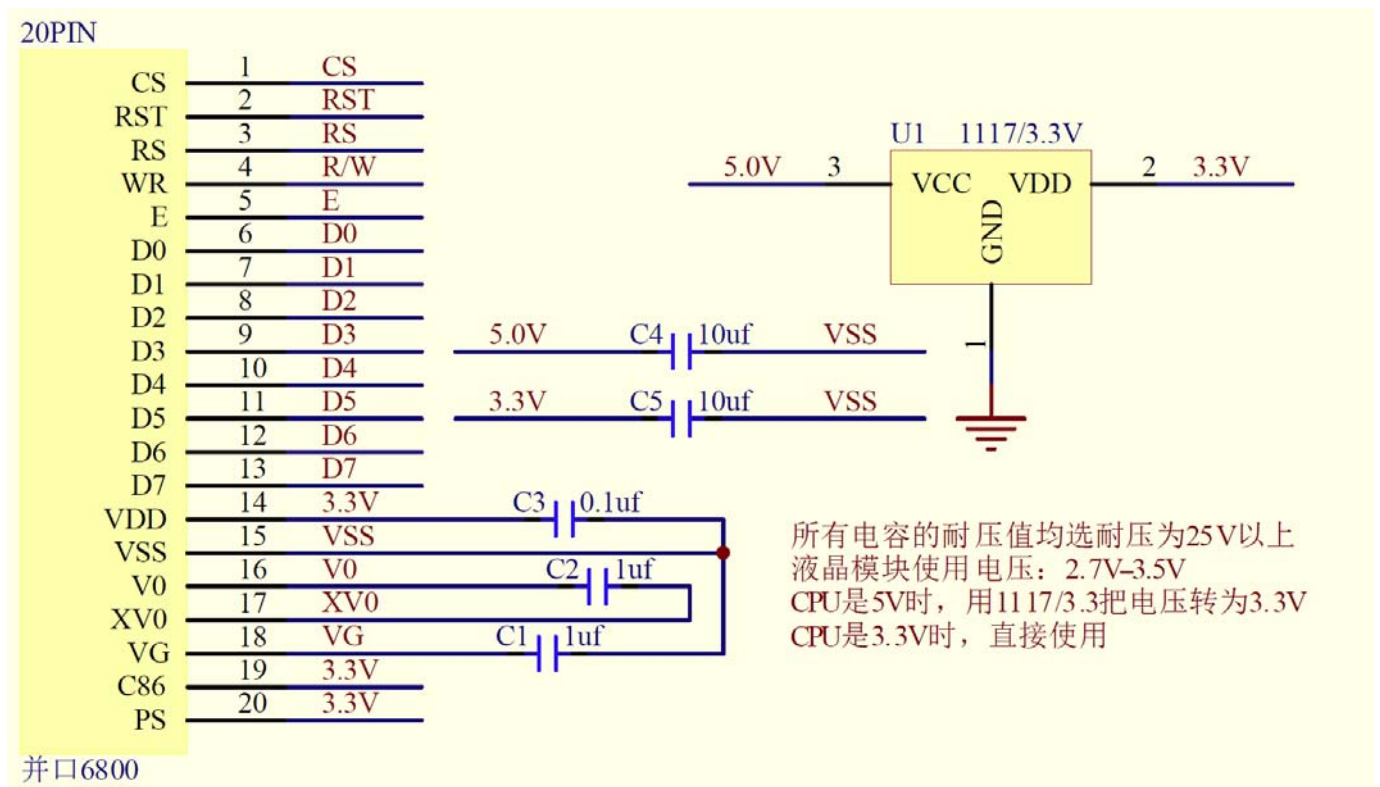


图 8. 并行接口

并程序与串程序基本相同，唯一不同之处是程序接口定义和数据传输子函数不一样，用下面并程序接口定义和数据传输子函数代替串程序原程序位置即可

```
#include <reg51.H>

sbit CS=P3^4;          //对应LCD的CS引脚
sbit RST=P3^5;        //对应LCD的RESET复位引脚
sbit RS=P3^3;         //对应LCD的RS引脚
sbit E=P3^0;          //对应LCD的E/RD引脚
sbit WR=P2^1;         //对应LCD的WR(R/W)引脚 另外: P1.0—P1.7对应LCD的D0—D7引脚
sbit key=P2^0;        /*按键接口, P2.0口与GND之间接一个按键*/
```

```
/*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++);
}
```

```
//写指令到LCD模块
void transfer_command(int data1)
{
```

```
    CS=0;
    RS=0;
    WR=0;
    P1=data1;
    E=1;
    //delay_us(1);
    E=0;
    CS=1;
    E=0;
    P1=0;
}
```

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

```
    CS=0;
    RS=1;
```




```
WR=0;  
P1=data1;  
E=1;  
//delay_us(3);  
E=0;  
CS=1;  
E=0;  
P1=0;  
}
```

-END-

