JLX19264G-260-BN 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 JLX19264G-260 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。

JLX19264G-260 可以显示 192 列*64 行点阵单色图片，或显示 12 个/行*4 行 16*16 点阵的汉字，或显示 24 个/行*8 行 5*8 点阵的英文、数字、符号。

2．JLX19264G-260 图像型点阵液晶模块的特性

2.1 结构轻、薄、带背光、焊接式 FPC。

2.2 IC 采用 UC1604c,功能强大，稳定性好

2.3 功耗低:当电压为 3.3V 时，功耗低：不带背光 1mW（3.3V*0.3mA），带背光 150mW（3.3V*45mA）；

2.4 显示内容:

（1）192*64 点阵单色图片，或其它小于 192*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 12 字*4 行；

（3）按照 12*12 点阵汉字来计算可显示 16 字*4 行；

（4）按照 8*16 点阵汉字来计算可显示 24 字*4 行；

（5）按照 5*8 点阵汉字来计算可显示 32 字*8 行；

2.5 指令功能强:可软件调对比度、正显/反显转换、行列扫描方向可改（可旋转 180 度使用）。

2.6 接口简单方便:可选并行、串行、IIC 接口；

2.7 工作温度宽:-20℃ － +70℃；

3. 外形尺寸及接口引脚功能

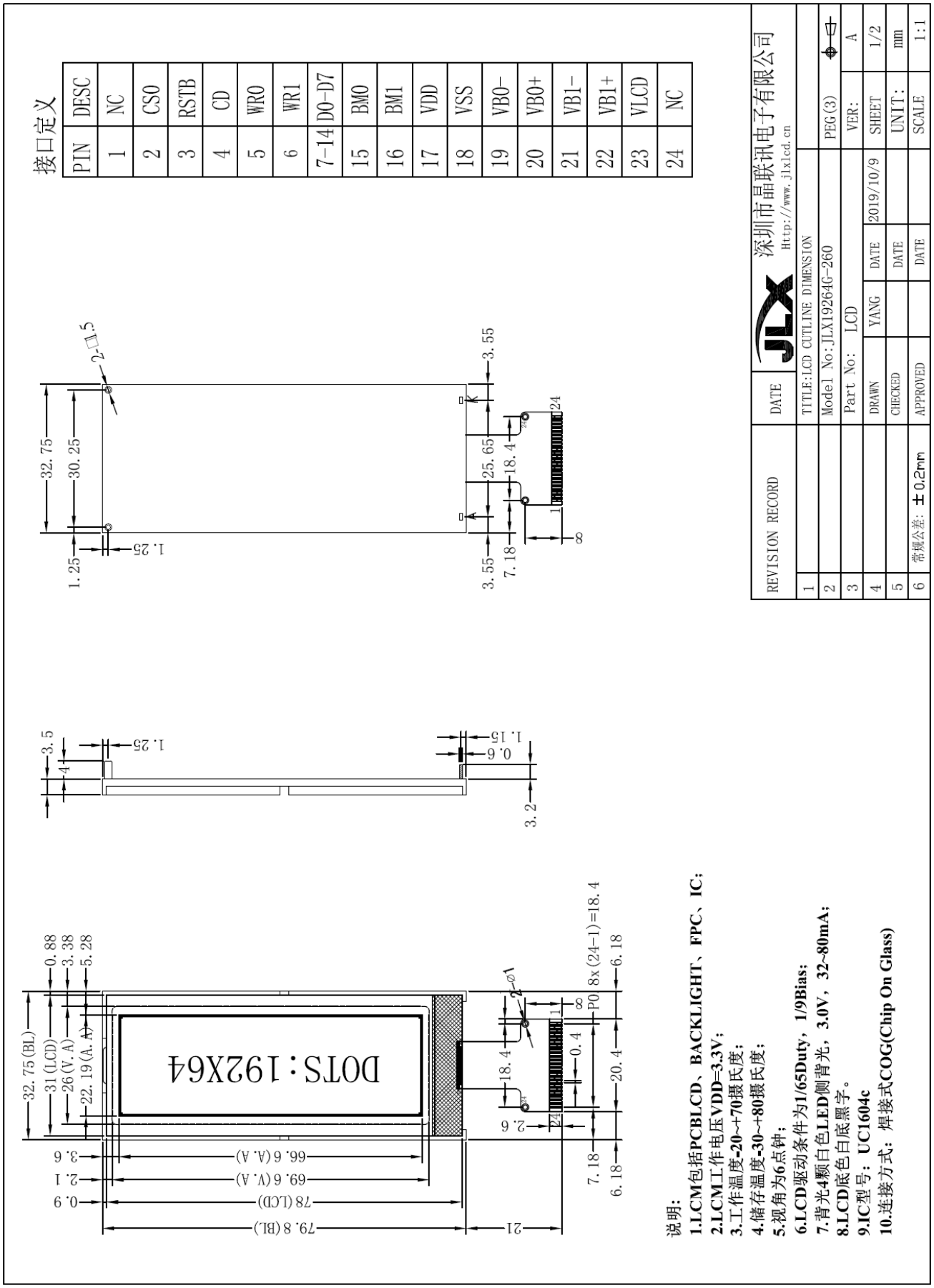


图 1. 外形尺寸

模块的接口引脚功能：

引 线 号	符 号	名 称	功 能
1	NC	NC	空脚
2	CS0(CS)	片选	低电平片选，IIC 接口时：接 VDD
3	RSTB	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
4	CD(RS)	寄存器选择信号	H:数据寄存器 0:指令寄存器 IIC 接口时：接 VSS
5	R/W(/WR)	6800 时序:读/写 8080 时序：写	并行接口时并且选择 6800 时序时：H:读数据 L:写数据 并行接口时并且选择 8080 时序时：写数据,低电平有效. IIC/串行接口时：接 VDD 或悬空
6	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. IIC/串行接口时：接 VDD 或悬空
7	D0(SCLK)	I/O	并行接口时：数据总线 DB6 IIC/串行接口时：串行时钟(SCLK)
8	D1	I/O	并行接口时：数据总线 DB1 IIC/串行接口时：悬空
9	D2	I/O	并行接口时：数据总线 DB2 IIC/串行接口时：悬空
10-12	D3-D5(SDA)	I/O	并行接口时：数据总线 DB3-DB5 IIC/串行接口时：串行数据 SDA（D3、D4、D5 连接一起做 SDA）
13	D6	I/O	并行接口时：数据总线 DB1 IIC/串行接口时：悬空
14	D7	I/O	并行接口时：数据总线 DB1 串行接口时：悬空 IIC 接口时：VDD
15	MB0	选择 6800 或 8080	并行接口时：H:6800 系统,L:8080 系统。 串行接口时：接 VSS IIC 接口时：接 VDD
16	MB1	选控制接口	接 VDD:选择并行接口。 串行/IIC 接口：接 VSS
17	VDD	供电电源正极	供电电源正极
18	VSS	接地	0V
19	VB0-	升压电容	倍压电路
20	VB0+	升压电容	倍压电路
21	VB1-	升压电容	倍压电路
22	VB1+	升压电容	倍压电路
23	VLCD	VLCD	与 VSS 之间串一个电容
24	NC	NC	空脚

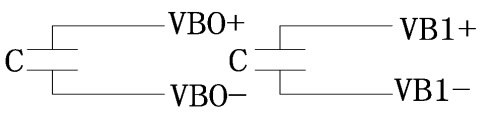


表 1：模块的接口引脚功能

4. 基本原理

4.1 液晶屏（LCD）

在 LCD 上排列着 192×64 点阵,192 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 工作电图：

图 2 是 JLX19264G-260 图像点阵型模块的电路框图,它由驱动 UC1604c 及几个电阻电容组成。

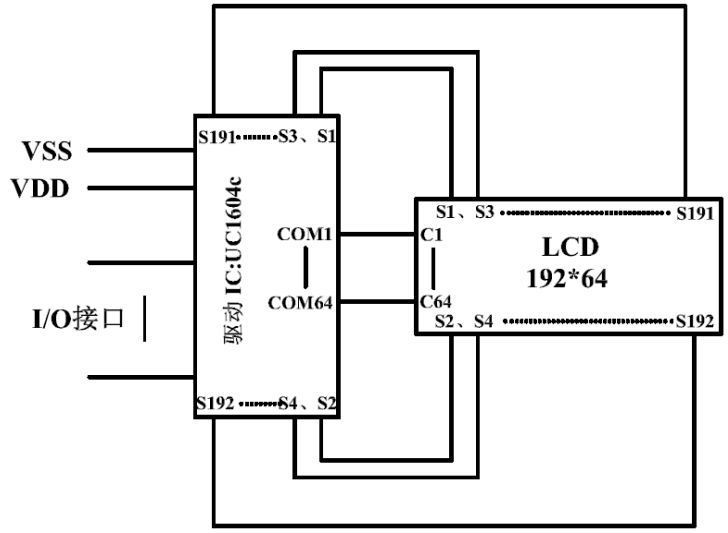


图 2:JLX19264G-260 图像点阵型液晶模块的电路框图

4.2 背光参数

该型号液晶模块带 LED 背光源。它的性能参数如下:
背光板可选择白色。
正常工作电流为：24∽60mA（LED 灯数共 3 颗）;
工作电压：3.0V;

5. 技术参数

5.1 最大极限参数（超过极限参数则会损坏液晶模块）

名称	符号	标准值			单位
		最小	典型	最大	
电路电源	VDD - VSS	-0.3		3.6	V
工作温度		-20		+70	℃
储存温度		-30		+80	℃

表 2：最大极限参数

5.2 直流（DC）参数

名 称	符 号	测 试 条 件	标 准 值			单位
			MIN	TYPE	MAX	
工作电压	VDD		2.4	3.3	3.6	V
背光工作电压	VLED		2.9	3.0	3.1	V
输入高电平	VIH	-	0.8xVDD		VDD	V

输入低电平	VIO	-	VSS		0.6	V
输出高电平	VOH	IOH = 0.2mA	0.8xVDD		VDD	V
输出低电平	VOO	IOO = 1.2mA	VSS		0.2xVDD	V
模块工作电流	IDD	VDD = 3.0V	-		0.3	mA
背光工作电流	ILED	VLED=3.0V	24	30	60	mA

表 3：直流（DC）参数

6. 读写时序特性

6.1 串行接口：

从 CPU 写到 UC1604c（Writing Data from CPU to UC1604c）

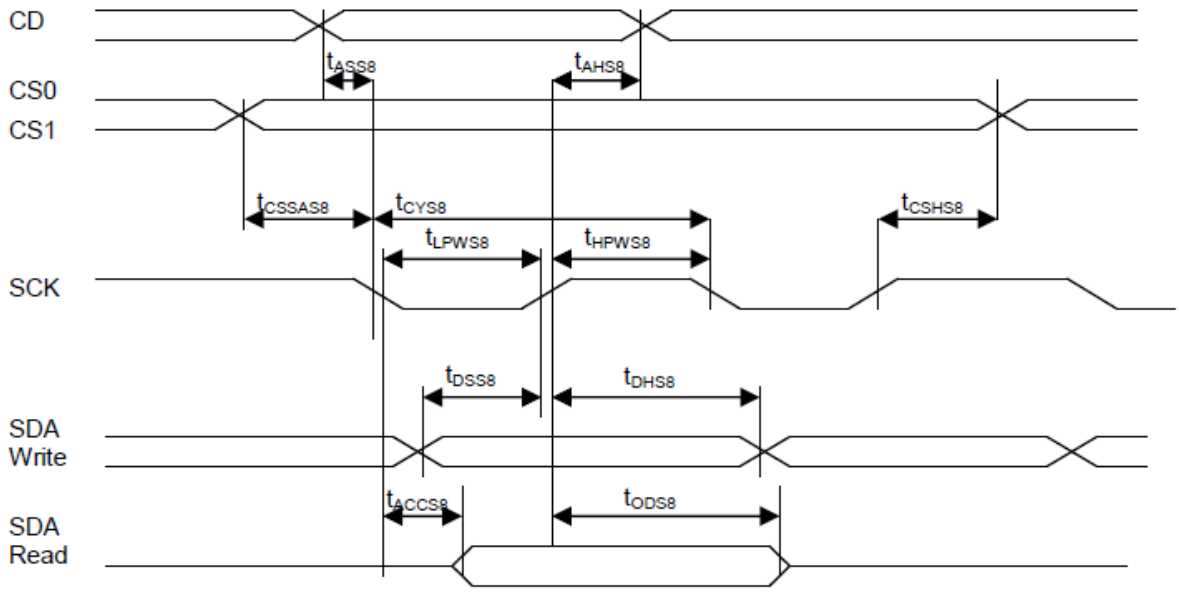


FIGURE 17: Serial Bus Timing Characteristics (for S8)

图 6. 从 CPU 写到 UC1604C（Writing Data from CPU to UC1604C）

6.2 串行接口：时序要求（AC 参数）：
写数据到 UC1604C 的时序要求：

Symbol	Signal	Description	Condition	Min.	Max.	Unit
(2.5V ≤ V_{DD} ≤ 3.6V, Ta= –30 to +85°C)				(Read / Write)		
t_{ASS8} t_{AHS8}	CD	Address setup time Address hold time		5 10	–	nS
t_{CSSAS8} t_{CSHS8}	CS1, CS0	Chip select setup time Chip select hold time		5 5	–	nS
t_{CYS8} t_{LPWS8} t_{HPWS8}	SCK	System Cycle time Low pulse width High pulse width		190 / 70 80 / 20 80 / 20	–	nS
t_{DSS8} t_{DHS8}	SDA (Write)	Data setup time Data hold time		20 10	–	nS
t_{ACC8} t_{OD8}	SDA (Read)	Read access time Output disable time	C_L = 100pF	– –	80 30	nS
(1.7V ≤ V_{DD} < 2.5V, Ta= –30 to +85°C)				(Read / Write)		
t_{ASS8} t_{AHS8}	CD	Address setup time Address hold time		5 10	–	nS
t_{CSSAS8} t_{CSHS8}	CS1, CS0	Chip select setup time Chip select hold time		10 10	–	nS
t_{CYS8} t_{LPWS8} t_{HPWS8}	SCK	System Cycle time Low pulse width High pulse width		230 / 110 100 / 40 100 / 40	– – –	nS nS nS
t_{DSS8} t_{DHS8}	SDA (Write)	Data setup time Data hold time		24 10	–	nS
t_{ACC8} t_{OD8}	SDA (Read)	Read access time Output disable time	C_L = 100pF	– –	100 60	nS

Note: tr (Rising time), tf (falling time) : ≤ 15nS

6.3 并行接口：（8080）

从 CPU 写到 UC1604c（Writing Data from CPU to UC1604c）

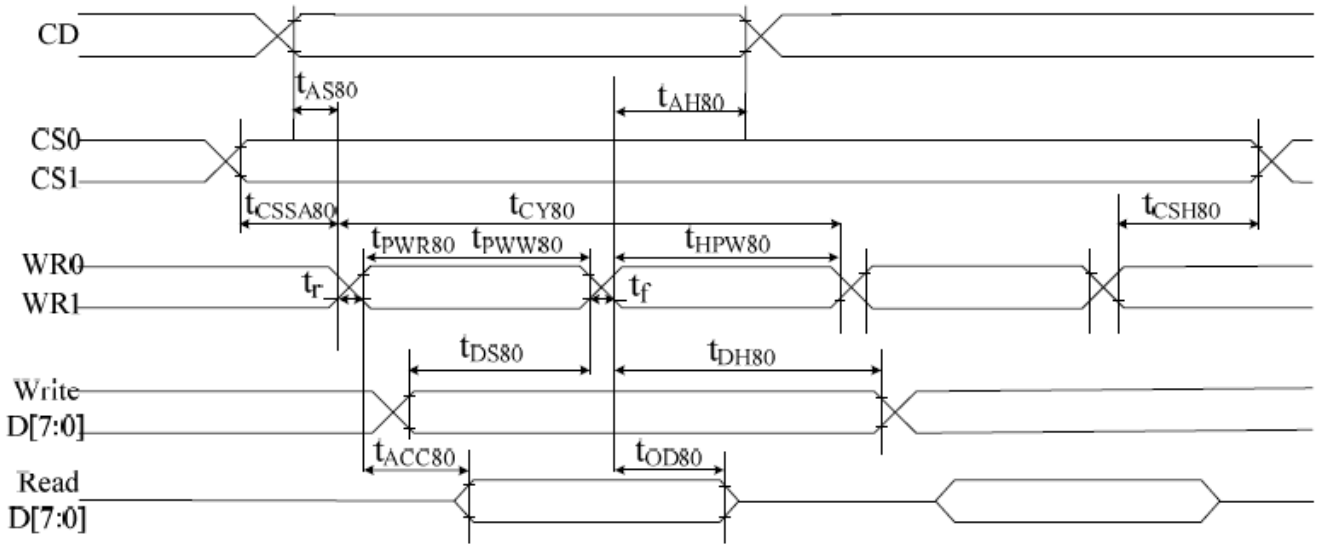


FIGURE 15: Parallel Bus Timing Characteristics (for 8080 MCU)

图 6. 从 CPU 写到 UC1604C（Writing Data from CPU to UC1604C）

6.4 并行接口：时序要求（AC 参数）：

写数据到 UC1604C 的时序要求：（8080 系列 MPU）

Symbol	Signal	Description	Condition	Min.	Max.	Unit
(2.5V ≤ V_{DD} ≤ 3.6V, Ta= –30 to +85°C)				(Read / Write)		
t_{AS80} t_{AH80}	CD	Address setup time Address hold time		5 10	–	nS
t_{CSSA80} t_{CSH80}	CS1, CS0	Chip select setup time Chip select hold time		5 5	–	nS
t_{CY80} t_{PWR80} / t_{PWW80} t_{HPW80}	WR0, WR1	System Cycle time Pulse width High pulse width		170 / 110 70 / 40 70 / 40	–	nS
t_{DS80} t_{DH80}	D7~D0 (Write)	Data setup time Data hold time		35 5	–	nS
t_{ACC80} t_{OD80}	D7~D0 (Read)	Read access time Output disable time	C_L = 100pF	– –	70 40	nS
(1.7V ≤ V_{DD} < 2.5V, Ta= –30 to +85°C)				(Read / Write)		
t_{AS80} t_{AH80}	CD	Address setup time Address hold time		5 10	–	nS
t_{CSSA80} t_{CSH80}	CS1, CS0	Chip select setup time Chip select hold time		5 5	–	nS
t_{CY80} t_{PWR80} / t_{PWW80} t_{HPW80}	WR0, WR1	System cycle time Pulse width High pulse width		270 / 190 120 / 80 120 / 80	–	nS
t_{DS80} t_{DH80}	D7~D0 (Write)	Data setup time Data hold time		60 5	–	nS
t_{ACC80} t_{OD80}	D7~D0 (Read)	Read access time Output disable time	C_L = 100pF	– –	115 80	nS

Note: tr (rising time), tf (falling time) : ≤ 15nS

6.5 并行接口：（6800）

从 CPU 写到 UC1604c（Writing Data from CPU to UC1604c）

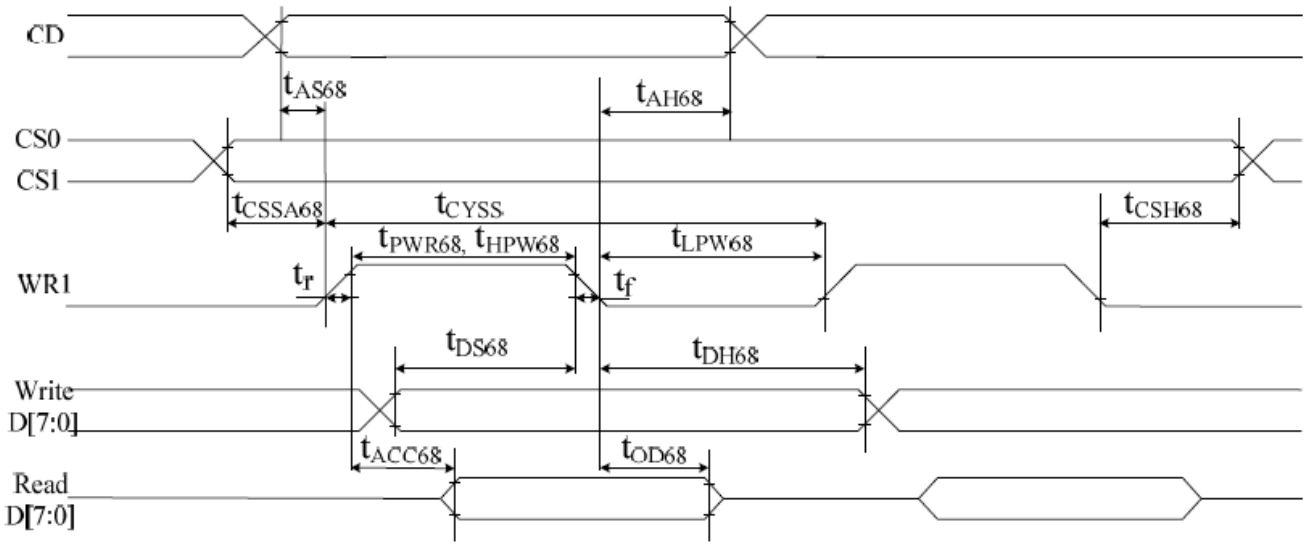


FIGURE 16: Parallel Bus Timing Characteristics (for 6800 MCU)

图 6. 从 CPU 写到 UC1604C（Writing Data from CPU to UC1604C）

6.6 并行接口：时序要求（AC 参数）：

写数据到 UC1604C 的时序要求：（6800 系列 MPU）

Symbol	Signal	Description	Condition	Min.	Max.	Unit
(2.5V ≤ V_{DD} ≤ 3.6V, Ta= –30 to +85°C)				(Read / Write)		
t_{AS68} t_{AH68}	CD	Address setup time Address hold time		5 10	–	nS
t_{CSSA68} t_{CSH68}	CS1, CS0	Chip select setup time Chip select hold time		5 5	–	nS
t_{CY68} t_{PWR68} / t_{PWW68} t_{HPW68}	WR1	System cycle time Pulse width High pulse width		170 / 110 70 / 40 70 / 40	–	nS
t_{DS68} t_{DH68}	D7~D0 (Write)	Data setup time Data hold time		35 5	–	nS
t_{ACC68} t_{OD68}	D7~D0 (Read)	Read access time Output disable time	C_L = 100pF	– –	70 40	nS
(1.7V ≤ V_{DD} < 2.5V, Ta= –30 to +85°C)				(Read / Write)		
t_{AS68} t_{AH68}	CD	Address setup time Address hold time		5 10	–	nS
t_{CSSA68} t_{CSH68}	CS1, CS0	Chip select setup time Chip select hold time		5 5	–	nS
t_{CY68} t_{PWR68} / t_{PWW68} t_{HPW68}	WR1	System cycle time Pulse width High pulse width		270 / 190 120 / 80 120 / 80	–	nS
t_{DS68} t_{DH68}	D7~D0 (Write)	Data setup time Data hold time		60 5	–	nS
t_{ACC68} t_{OD68}	D7~D0 (Read)	Read access time Output disable time	C_L = 100pF	– –	115 80	nS

Note: tr (Rising time), tf (falling time) : ≤ 15nS

6.7 IIC 接口：

从 CPU 写到 UC1604c（Writing Data from CPU to UC1604c）

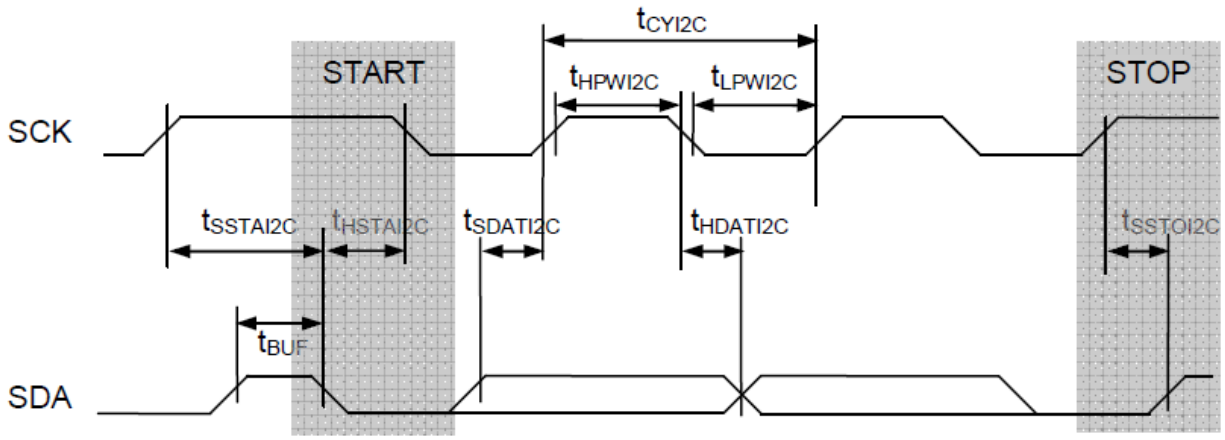


FIGURE 19: Serial bus timing characteristics (for I^2C)

图 6. 从 CPU 写到 UC1604C（Writing Data from CPU to UC1604C）

6.8 IIC 接口：时序要求（AC 参数）：

写数据到 UC1604C 的时序要求：

Symbol	Signal	Description	Condition	Min.	Max.	Unit
(2.5V ≤ V_{DD} ≤ 3.6V, Ta= –30 to +85°C)				(Read / Write)		
t_{CYI2C}	SCK	SCK cycle time		610 / 305	–	nS
t_{HPWI2C}		High pulse width		290 / 110		
t_{LPWI2C}		Low pulse width		290 / 165		
$t_{SSTAI2C}$	SCK SDA	Setup time – START		28	–	nS
$t_{HSTAI2C}$		Hold time – START		55		
t_{SDAI2C}		Setup time – Data		40		
t_{HDAI2C}		Hold time – Data		11		
$t_{SSTOI2C}$		Setup time – STOP		28		
t_{BUF}	SDA	Bus Free time between STOP and START		165	–	nS
(1.7V ≤ V_{DD} < 2.5V, Ta= –30 to +85°C)				(Read / Write)		
t_{CYI2C}	SCK	SCK cycle time		780 / 360	–	nS
t_{HPWI2C}		High pulse width		375 / 130		
t_{LPWI2C}		Low pulse width		375 / 200		
$t_{SSTAI2C}$	SCK SDA	Setup time – START		33	–	nS
$t_{HSTAI2C}$		Hold time – START		80		
t_{SDAI2C}		Setup time – Data		80		
t_{HDAI2C}		Hold time – Data		11		
$t_{SSTOI2C}$		Setup time – STOP		33		
t_{BUF}	SDA	Bus Free Time between STOP and START		220	–	nS

Note: tr (Rising time), tf (falling time) : ≤ 15nS

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

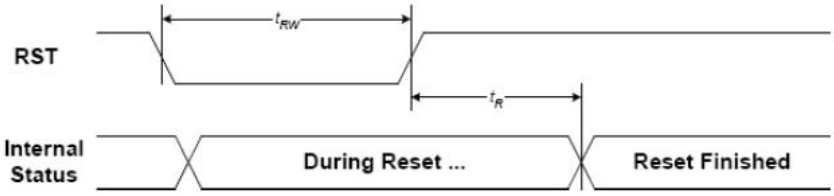


FIGURE 20: Reset Characteristics

(1.7V ≤ V_{DD} ≤ 3.6V, Ta= –30 to +85°C)

Symbol	Signal	Description	Condition	Min.	Max.	Unit
t_{RW}	RST	Reset low pulse width		3	–	μS
t_R	RST, Internal Status	Reset to Internal Status pulse delay		6	–	mS

图 7：**电源启动后复位的时序**

7. 指令功能：

7.1 指令表

下表是“UC1604C”IC 支持的指令：

CD:0:指令； 1:数据 W/R: 0:写； 1:读 D7~D0:有用的数据位； -：不必理会的

表 8.

指令名称		指令/数据	读/写	指 令 码								说 明
		CD(RS)	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
1.写数据字节 (Write Data Byte)		1	0	#	#	#	#	#	#	#	#	写1个字节
2.读数据字节 (Read Data Byte)		1	1	#	#	#	#	#	#	#	#	读1个字节的数据
3.读取状态 (Get Status)		0	1	ID	MX	MY	WA	DE	WS	MD	MS	从液晶驱动IC(UC1604C)里读取状态.具体“ID”、“MX”、“MY”这些字代表什么意思，请查阅“UC1604C”IC资料（找客服人员获取IC资料）。
				VER	POR	PM5	PM4	PM3	PM2	PM1	PM0	
4	设置列地址低4位 Set Column Address LSB	0	0	0	0	0	0	CA3	CA2	CA1	CA0	高4位与低4位共同组成列地址，指定192列中的其中一列。比如液晶模块的第1列地址十六进制为0x00，那么此指令由2个字节来表达：0x10, 0x00. 第100列地址十六进制为0x63，那么此指令由2个字节来表达：0x16, 0x03
	设置列地址高 4 位 Set Column Address MSB	0	0	0	0	0	1	CA7	CA6	CA5	CA4	

5.设置温度补偿系数 (Set Temp. compensation)	0	0	0	0	1	0	0	1	TC1	TC0	设置温度补偿系数TC1~0：温度升高每一度的液晶电压值升高的百分比： 0x24：-0.00%/°C, 0x25:-0.05%/ °C, 0x26:-0.10% °C 0x27:-0.15% °C
6.设置电源控制 (Set Power control)	0	0	0	0	1	0	1	PC2	PC1	PC0	设置电源控制PC2~PC0, PC[1:0]:选择升压的电流： 00b:0.6mA； 01b:1.0mA； 10b:1.4mA； 11b:2.3mA； PC2:选择升压方式： 0b：外部供电给VLCD， 1b：内部升压给VLCD(7倍升压)
7.设置高级的程序控制（双字节指令） Set Adv. Program Control. (double-byte command)	0	0	0	0	1	1	0	0	R	R	设置APC[R]7~0,R=0~3，此指令是IC原厂使用的，我们用不着。
			APC[R]7	APC[R]6	APC[R]5	APC[R]4	APC[R]3	APC[R]2	APC[R]1	APC[R]0	
8.设置起始行 (Set Scroll Line)	0	0	0	1	SL5	SL4	SL3	SL2	SL1	SL0	设置起始行，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
9. 页地址设置 (Page address set)	0	0	1	0	1	1	PA3	PA2	PA1	PA0	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1页到第8页.
10.对比度电位器设置（双字节指令） Set Vbias Potentiometer (double-byte command)	0	0	1	0	0	0	0	0	0	1	设置内部电位器微调，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.
			#	#	#	#	#	#	#	#	
11.设置部分显示控制（set partial display control）	0	0	1	0	0	0	0	1	0	LC5	设部分显示： 当LC5=0时，不允许部分显示,DUTY正常。(0X84) 当LC5=1时，允许部分显示， DUTY=DEN-DST+1,（DEN即显示结束行，DST即显示开始行）.(0X85)
12.设置存储器（RAM）地址控制 （ set RAM address control）	0	0	1	0	0	0	1	AC2	AC1	AC0	AC[2]=0:页地址自动+1; AC[2]=1:页地址自动-1; AC[1]=0:列地址自动+1直到LCD边缘为止，然后页地址将+/-1; AC[1]=1:页地址自动+/-1直到LCD边缘为止，然后列地址将+1; AC[0]=0:列地址或页地址（取决于AC[1]=0还是1）在到达LCD边缘后会停止; AC[0]=1:列地址或页地址（取决于AC[1]=0还是1）在到达LCD边缘后会

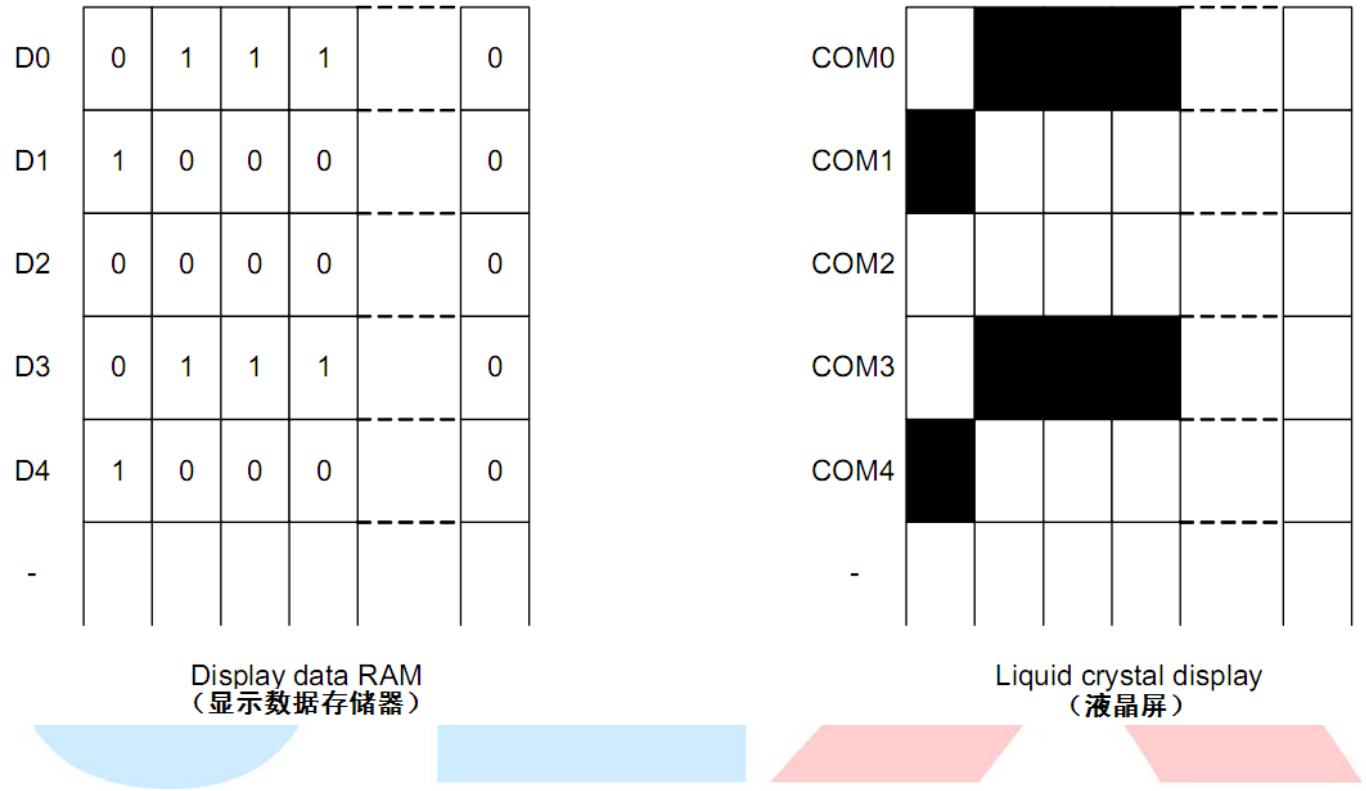
											重新开始；且列地址或页地址将+1。
13.设置帧频 (set Frame Rate)	0	0	1	0	1	0	0	0	LC4	LC3	LC[4:3]=00:76帧/秒(0XA0) LC[4:3]=00:95帧/秒(0XA1) LC[4:3]=00:132帧/秒(0XA2) LC[4:3]=00:168帧/秒(0XA3)
14.设置打开全部点阵	0	0	1	0	1	0	0	1	0	DC1	DC1=1:打开所有点阵(0XA5) DC1=0:正常显示（默认值=0）(0XA4)
15.设置反显	0	0	1	0	1	0	0	1	1	DC0	DC0=1:反显(0XA7) DC0=0:正常显示（默认值=0）(0XA6)
16.显示开/关	0	0	1	0	1	0	1	1	1	DC2	DC2=1:打开显示（0xAF） DC2=0:关显示（0xAE）
17.设置 LCD 映射控制（set LCD Mapping control）	0	0	1	1	0	0	0	MY	MX	0	MY=0:显示顺序为从上到下； MY=1:显示顺序为从下到上； MX=0:显示顺序为从左到右； MX=1:显示顺序为从右到左。
18.系统复位	0	0	1	1	1	0	0	0	1	0	系统复位（0xe2）
19.空	0	0	1	1	1	0	0	0	1	1	空指令（0xe3）
20.内部检测用（IC 厂）											IC厂使用，我们不管
21. 设置 Bias 比例	0	0	1	1	1	0	1	0	BR1	BR0	BR[1:0]=00:BIAS=1/6; (0XE8) BR[1:0]=01:BIAS=1/7; (0XE9) BR[1:0]=10:BIAS=1/8; (0XEA) BR[1:0]=11:BIAS=1/9; (0XEB) （针对本液晶屏请设置为1/9,以获得最佳效果）
22.设置 LCD 的总行数（双字节指令）	0	0	1	1	1	1	0	0	0	1	设置LCD的总行数，不设置表示默认为64。本液晶屏为64行，所以不用设置这一条指令。
	0	0	-	-	CEN5	CEN4	CEN3	CEN2	CEN1	CEN0	
23.设置部分显示的开始行。（双字节指令）	0	0	1	1	1	1	0	0	1	0	设置部分显示的开始行。双指令： 1.0xf2 2.0x00~0x3f
	0	0	-	-	部分显示的开始行						
24.设置部分显示的结束行。（双字节指令）	0	0	1	1	1	1	0	0	1	1	设置部分显示的结束行。双指令： 1.0xf3 2.0x00~0x3f
	0	0	-	-	部分显示的结束行						
25~30.MTP 方面的指令，只与液晶模块厂家及 IC 厂家有用。											
在S8及S9接口（两种SPI串行接口）方式时，用下列指令可以读状态及显示数据：											
31.读 IC 的状态	0	0	1	1	1	1	1	1	1	0	1、0xfe 2.读状态1 3.读状态2
	0	1	ID	MX	MY	WA	DE	WS	MD	MS	
	0	1	VER	POR	PM5	PM4	PM3	PM2	PM1	PM0	

32 读数据	0	0	1	1	1	1	1	1	1	1	1.0xff 2. 数据
	1	1	#	#	#	#	#	#	#	#	

7.3 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 192*64 点阵的屏分为 8 个“页”，从第 0“页”到第 7“页”。

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



下图摘自 UC1604C IC 资料，可通过“UC1604c_a1.3.pdf”获取最佳效果。

PA[3:0]	0	Line Address	Page
0000	D0	R0	Page 0
	D1	R1	
	D2	R2	
	D3	R3	
	D4	R4	
	D5	R5	
	D6	R6	
	D7	R7	
0001	D0	R8	Page 1
	D1	R9	
	D2	R10	
	D3	R11	
	D4	R12	
	D5	R13	
	D6	R14	
	D7	R15	
0010	D0	R16	Page 2
	D1	R17	
	D2	R18	
	D3	R19	
	D4	R20	
	D5	R21	
	D6	R22	
	D7	R23	
0011	D0	R24	Page 3
	D1	R25	
	D2	R26	
	D3	R27	
	D4	R28	
	D5	R29	
	D6	R30	
	D7	R31	
0100	D0	R32	Page 4
	D1	R33	
	D2	R34	
	D3	R35	
	D4	R36	
	D5	R37	
	D6	R38	
	D7	R39	
0101	D0	R40	Page 5
	D1	R41	
	D2	R42	
	D3	R43	
	D4	R44	
	D5	R45	
	D6	R46	
	D7	R47	
0110	D0	R48	Page 6
	D1	R49	
	D2	R50	
	D3	R51	
	D4	R52	
	D5	R53	
	D6	R54	
	D7	R55	
0111	D0	R56	Page 7
	D1	R57	
	D2	R58	
	D3	R59	
	D4	R60	
	D5	R61	
	D6	R62	
	D7	R63	
1000	D0	R64	Page 8

Panel Location	MY=0 SL=0	MY=0 SL=16	MY=1 SL=0	MY=1 SL=16
COM1	R0	R16	R63	R15
COM2	R1	R17	R62	R14
COM3	R2	R18	R61	R13
COM4	R3	R19	R60	R12
COM5	R4	R20	R59	R11
COM6	R5	R21	R58	R10
COM7	R6	R22	R57	R9
COM8	R7	R23	R56	R8
COM9	R8	R24	R55	R7
COM10	R9	R25	R54	R6
COM11	R10	R26	R53	R5
COM12	R11	R27	R52	R4
COM13	R12	R28	R51	R3
COM14	R13	R29	R50	R2
COM15	R14	R30	R49	R1
COM16	R15	R31	R48	R0
COM17	R16	R32	R47	R63
COM18	R17	R33	R46	R62
COM19	R18	R34	R45	R61
COM20	R19	R35	R44	R60
COM21	R20	R36	R43	R59
COM22	R21	R37	R42	R58
COM23	R22	R38	R41	R57
COM24	R23	R39	R40	R56
COM25	R24	R40	R39	R55
COM26	R25	R41	R38	R54
COM27	R26	R42	R37	R53
COM28	R27	R43	R36	R52
COM29	R28	R44	R35	R51
COM30	R29	R45	R34	R50
COM31	R30	R46	R33	R49
COM32	R31	R47	R32	R48
COM33	R32	R48	R31	R47
COM34	R33	R49	R30	R46
COM35	R34	R50	R29	R45
COM36	R35	R51	R28	R44
COM37	R36	R52	R27	R43
COM38	R37	R53	R26	R42
COM39	R38	R54	R25	R41
COM40	R39	R55	R24	R40
COM41	R40	R56	R23	R39
COM42	R41	R57	R22	R38
COM43	R42	R58	R21	R37
COM44	R43	R59	R20	R36
COM45	R44	R60	R19	R35
COM46	R45	R61	R18	R34
COM47	R46	R62	R17	R33
COM48	R47	R63	R16	R32
COM49	R48	R0	R15	R31
COM50	R49	R1	R14	R30
COM51	R50	R2	R13	R29
COM52	R51	R3	R12	R28
COM53	R52	R4	R11	R27
COM54	R53	R5	R10	R26
COM55	R54	R6	R9	R25
COM56	R55	R7	R8	R24
COM57	R56	R8	R7	R23
COM58	R57	R9	R6	R22
COM59	R58	R10	R5	R21
COM60	R59	R11	R4	R20
COM61	R60	R12	R3	R19
COM62	R61	R13	R2	R18
COM63	R62	R14	R1	R17
COM64	R63	R15	R0	R16
CIC	R64	R64	R64	R64

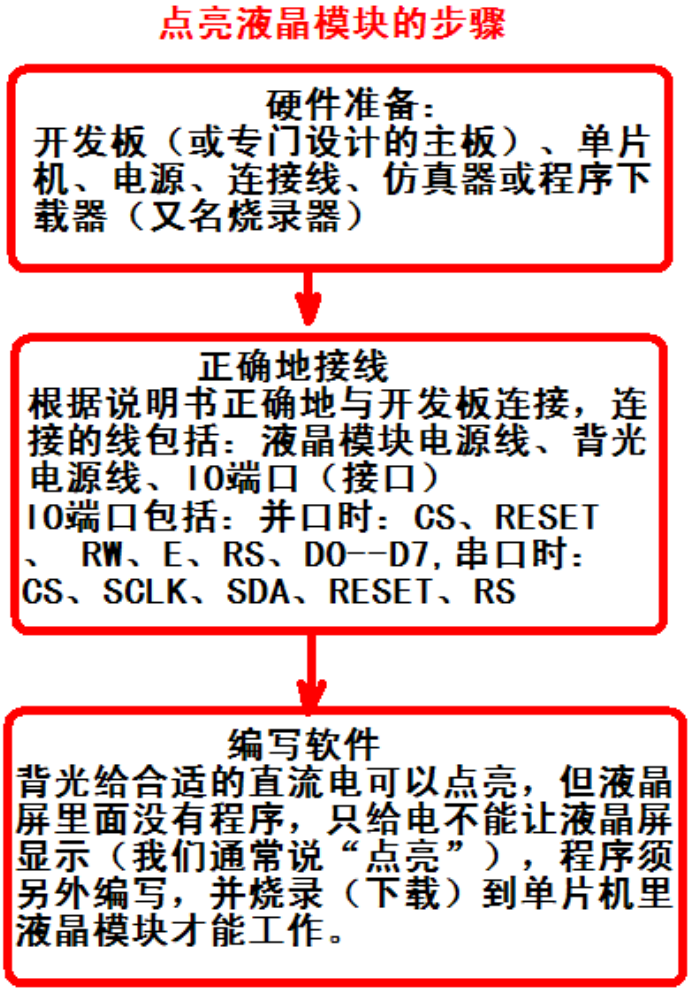
MX=0	SEG1	SEG2	SEG3	SEG4	SEG5	SEG6	SEG7	SEG8		SEG128	SEG129	SEG130	SEG131	SEG132
MX=1	SEG132	SEG131	SEG130	SEG129	SEG128	SEG127	SEG126	SEG125		SEG5	SEG4	SEG3	SEG2	SEG1

Example for memory mapping: let MX = 0, MY = 0, SL = 0, according to the data shown in the above table:

⇒ Page 0 SEG 1 (D7-D0) : 0001 1111b
⇒ Page 0 SEG 2 (D7-D0) : 1100 1100b

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:

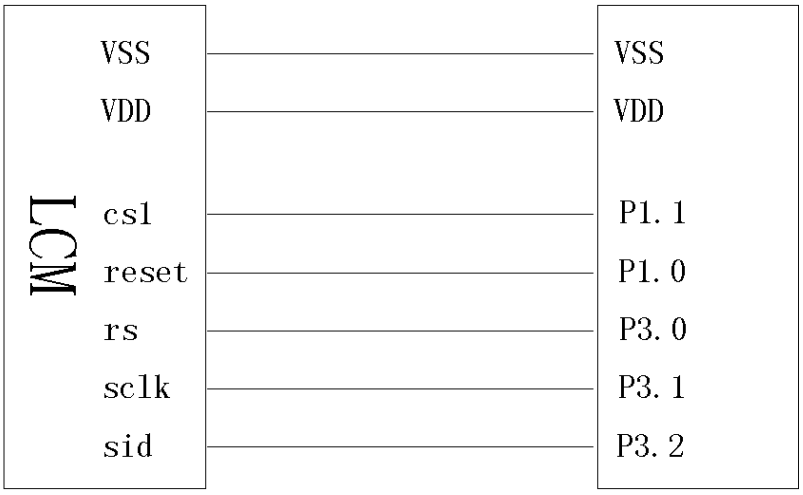
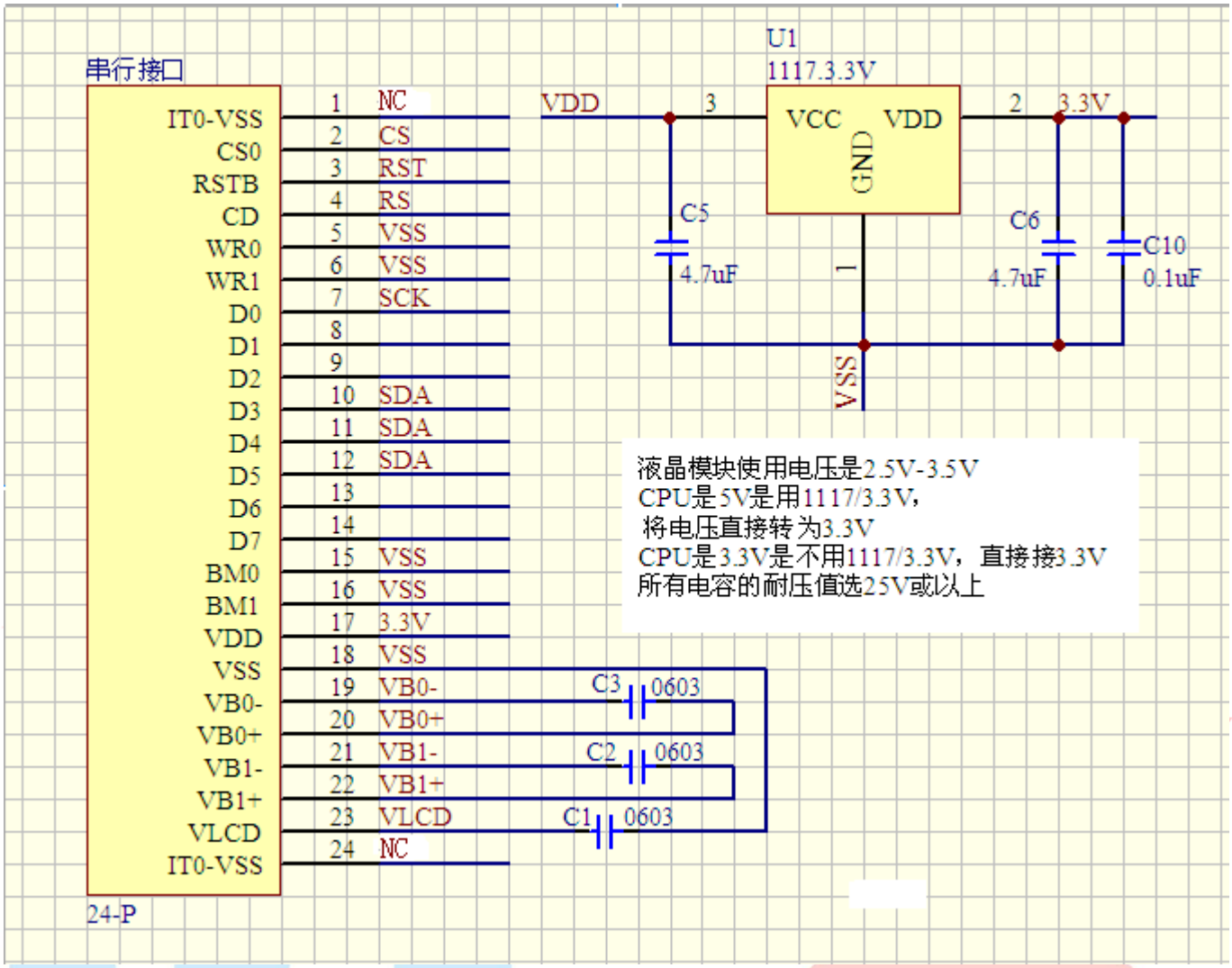


图 8. 串行接口

串行电路图



7.5.1 程序

```
//    液晶演示程序 JLX19264G-260，串行接口！
//    驱动 IC 是:UC1604c


#include <reg52.h>
#include <intrins.h>
#include <Ctype.h>

sbit cs1=P3^2;
sbit reset=P3^1;
sbit rs=P3^0;
sbit sclk=P1^0;
sbit sid=P1^1;
sbit key=P2^0;

#define uchar unsigned char
```

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


uchar code ascii_table_8x16[95][16];
uchar code ascii_table_5x8[95][5];
uchar code bmp1[];
uchar code bmp2[];
uchar code bmp3[];
void delay_us(int i);
void delay(int i);

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

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,0x20,0x20,0x20,0x20,0xFF,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 tai1[]={
//--  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 shi1[]={
//--  文字:  使  --
//--  宋体 12;  此字体下对应的点阵为：宽 x 高=16x16   --
0x40,0x20,0xF0,0x1C,0x07,0xF2,0x94,0x94,0x94,0xFF,0x94,0x94,0x94,0xF4,0x04,0x00,
0x00,0x00,0x7F,0x00,0x40,0x41,0x22,0x14,0x0C,0x13,0x10,0x30,0x20,0x61,0x20,0x00};

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


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    --
0x00,0x10,0x10,0xF8,0x00,0x00,0x00,0x00,0x00,0x20,0x20,0x3F,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    --
0x00,0x00,0xC0,0x20,0x10,0xF8,0x00,0x00,0x00,0x07,0x04,0x24,0x24,0x3F,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    --
0x00,0x38,0x08,0x08,0xC8,0x38,0x08,0x00,0x00,0x00,0x00,0x3F,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;
        if(data1&0x80) sid=1;
        else sid=0;
        sclk=1;
        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;
        if(data1&0x80) sid=1;
```

```
        else sid=0;
        sclk=1;
        data1=data1<<=1;
    }
    cs1=1;
}



//延时 1
void delay(int i)
{
 int j,k;
 for(j=0;j<i;j++)
 for(k=0;k<110;k++);
}

//延时 2
void delay_us(int i)
{
 int j,k;
 for(j=0;j<i;j++)
 for(k=0;k<10;k++);
}

void waitkey()
{
repeat:
    if(key==1)goto repeat;
    else delay(400);
}

//LCD 模块初始化
void initial_lcd()
{
    reset=0;        //低电平复位
    delay(800);
    reset=1;          //复位完毕
    delay(800);
    transfer_command(0xe2); //软复位
    delay(500);
    transfer_command(0x2f);  //打开内部升压
    delay(500);

    transfer_command(0x81);  //微调对比度
```

```
    transfer_command(0x56);  //微调对比度的值，可设置范围 0x00～0xFF
    transfer_command(0xeb);  //1/9 偏压比（bias）
    transfer_command(0xc2);  //行扫描顺序：从上到下 oxc2
//  transfer_command(0xa0);  //列扫描顺序：从左到右
    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<8;i++)
    {
        lcd_address(1+i,1);
        for(j=0;j<192;j++)
        {
            transfer_data(0x00);
        }
    }
}

void display_graphic_192x64(uchar *dp)
{
    uchar i,j;
    for(i=0;i<8;i++)
    {
        lcd_address(i+1,1);
        for(j=0;j<192;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}
```

```
//==================display a piture of 128*64 dots================
void full_display(uchar data_left,uchar data_right)
{
     int i,j;
     for(i=0;i<8;i++)
     {
          lcd_address(i+1,1);
          for(j=0;j<96;j++)
          {
               transfer_data(data_left);
               transfer_data(data_right);
          }
     }
}

//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 *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++;
        }
    }
}

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++;
    }
}

//显示一串 5x8 点阵的字符串
//括号里的参数分别为（页，列，是否反显，数据指针）
/*
void display_string_5x8(uint page,uint column,uchar reverse,uchar *text)
{
    uchar i=0,j,k,data1;
```

```
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(page,column);
            for(k=0;k<5;k++)
            {
                if(reverse==1)     data1=~ascii_table_5x8[j][k];
                else data1=ascii_table_5x8[j][k];
                transfer_data(data1);
            }
            if(reverse==1)     transfer_data(0xff);
            else transfer_data(0x00);
            i++;
            column+=6;
        }
        else
        i++;
    }
}
*/

//显示一串 5x8 点阵的字符串
//括号里的参数分别为（页，列，是否反显，数据指针）
void display_string_5x8(uint page,uint column,uchar reverse,uchar *text)
{
    uchar i=0,j,k,data1;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(page,column);
            for(k=0;k<5;k++)
            {
                if(reverse==1)     data1=~ascii_table_5x8[j][k];
                else data1=ascii_table_5x8[j][k];
                transfer_data(data1);
            }
            if(reverse==1)     transfer_data(0xff);
            else transfer_data(0x00);
            i++;
            column+=6;
        }
        else
```

```
        i++;
    }
}

void display_string_5x8_1(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_5x8[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++;
    }
}

void main(void)
{
    while(1)
    {
        initial_lcd();
        clear_screen();
        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(3,100,1,"1.Graphic      ");
        display_string_5x8(4,100,0,"2.Chinese       " );
        display_string_5x8(5,100,0,"3.Movie         ");
        display_string_5x8(6,100,0,"4.Contrast      ");
        display_string_5x8(7,100,0,"5.Mirror        ");
        display_string_5x8(8,1,1,"    PRE    USER     DEL     NEW     ");
        display_string_5x8(8,59,0," ");
        display_string_5x8(8,94,0," ");
        display_string_5x8(8,97+48,0," ");
```

```
waitkey();

clear_screen();                                    //clear all dots
display_graphic_192x64(bmp1);
delay(2000);
waitkey();
clear_screen();
display_graphic_192x64(bmp2);
delay(2000);
waitkey();
clear_screen();
display_graphic_192x64(bmp3);
delay(2000);
waitkey();
clear_screen();
display_graphic_32x32(1,1,cheng1);              //在第 1 页，第 49 列显示单个汉字"成"
delay(2000);
waitkey();
clear_screen();                                    //clear all dots
display_graphic_16x16(5,1,zhuang1);         //在第 5 页，第 1 列显示单个汉字"状"
display_graphic_16x16(5,(1+16),tai1);           //在第 5 页，第 17 列显示单个汉字"态"
display_graphic_8x16(5,(1+16*2),mao_hao);       //在第 5 页，第 25 列显示单个字符":"
display_graphic_16x16(5,(1+16*2+8),shi1);       //在第 5 页，第 41 列显示单个汉字"使"
display_graphic_16x16(5,(1+16*3+8),yong1);      //在第 5 页，第 49 列显示单个汉字"用"
display_graphic_8x16(5,(89),num0);              //在第 5 页，第 89 列显示单个数字"0"
display_graphic_8x16(5,(89+8*1),num0);          //在第 5 页，第 97 列显示单个数字"0"
display_graphic_8x16(5,(89+8*2),mao_hao);       //在第 5 页，第 105 列显示单个字符":"
display_graphic_8x16(5,(89+8*3),num0);          //在第 5 页，第 113 列显示单个数字"0"
display_graphic_8x16(5,(89+8*4),num0);          //在第 5 页，第 121 列显示单个数字"0"
waitkey();
clear_screen();                                     //clear all dots
display_string_8x16(1,1,"(<\"0123456abt~`!@#$%^\">)");//在第 1 页，第 1 列显示字符串
display_string_8x16(3,1,"{[(<\" '&*|\\@#_-+=' \">)]}");//在第*页，第*列显示字符串
display_string_5x8_1(5,1,"[!#$%&'()*+,-./0123456789:;<=>?]");
display_string_5x8_1(6,1,"[ABCDEFGHIJKLMNOPQRSTUVWXYZabcd]");
display_string_5x8_1(7,1,"(abcdefghijklmnopqrstuvwxyzabcd)");
display_string_5x8_1(8,1,"{[(<\" '&*|\\@abcde012#_-+=' \">)]}");
waitkey();
delay(2000);
full_display(0xff,0xff);
waitkey();
delay(2000);
full_display(0x55,0xaa);
waitkey();
delay(2000);
full_display(0xaa,0x55);
```

```
            waitkey();
            delay(2000);
            full_display(0xff,0x00);
            waitkey();
            delay(2000);
            full_display(0x00,0xff);
            waitkey();
            delay(2000);
        }
}

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,  0x00,0x00,0x00,0x0D,  0x0D,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,  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,  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,  0x00,0x00,0x00,0x06,  0x06,0x00,0x00,0x00,  //-:-
0x00,0x00,0x00,0x30,  0x30,0x00,0x00,0x00,  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,  0x20,0x20,0x20,0x00,  0x00,0x01,0x01,0x01,  0x01,0x01,0x01,0x00,  //-=-

0x00,0x08,0x18,0x30,  0x60,0xC0,0x80,0x00,  0x00,0x08,0x0C,0x06,  0x03,0x01,0x00,0x00,  //->-
    ASCII 码：0X3E
0x18,0x1C,0x04,0xC4,  0xE4,0x3C,0x18,0x00,  0x00,0x00,0x00,0x0D,  0x0D,0x00,0x00,0x00,  //-?-
0xF0,0xF0,0x08,0xC8,  0xC8,0xF8,0xF0,0x00,  0x07,0x0F,0x08,0x0B,  0x0B,0x0B,0x01,0x00,  //-@-
0xE0,0xF0,0x98,0x8C,  0x98,0xF0,0xE0,0x00,  0x0F,0x0F,0x00,0x00,  0x00,0x0F,0x0F,0x00,  //-A-
    ASCII 码：0X41
0x04,0xFC,0xFC,0x44,  0x44,0xFC,0xB8,0x00,  0x08,0x0F,0x0F,0x08,  0x08,0x0F,0x07,0x00,  //-B-
0xF0,0xF8,0x0C,0x04,  0x04,0x0C,0x18,0x00,  0x03,0x07,0x0C,0x08,  0x08,0x0C,0x06,0x00,  //-C-
0x04,0xFC,0xFC,0x04,  0x0C,0xF8,0xF0,0x00,  0x08,0x0F,0x0F,0x08,  0x0C,0x07,0x03,0x00,  //-D-
0x04,0xFC,0xFC,0x44,  0xE4,0x0C,0x1C,0x00,  0x08,0x0F,0x0F,0x08,  0x08,0x0C,0x0E,0x00,  //-E-
0x04,0xFC,0xFC,0x44,  0xE4,0x0C,0x1C,0x00,  0x08,0x0F,0x0F,0x08,  0x00,0x00,0x00,0x00,  //-F-
0xF0,0xF8,0x0C,0x84,  0x84,0x8C,0x98,0x00,  0x03,0x07,0x0C,0x08,  0x08,0x07,0x0F,0x00,  //-G-

0xFC,0xFC,0x40,0x40,  0x40,0xFC,0xFC,0x00,  0x0F,0x0F,0x00,0x00,  0x00,0x0F,0x0F,0x00,  //-H-
    ASCII 码：0X48
0x00,0x00,0x04,0xFC,  0xFC,0x04,0x00,0x00,  0x00,0x00,0x08,0x0F,  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,  0x00,0x00,0x00,0x00,  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,  0x00,0x00,0x00,0x01,  0x03,0x07,0x0E,0x00,  //-\-
0x00,0x00,0x04,0x04,  0xFC,0xFC,0x00,0x00,  0x00,0x00,0x08,0x08,  0x0F,0x0F,0x00,0x00,  //-]-
0x08,0x0C,0x06,0x03,  0x06,0x0C,0x08,0x00,  0x00,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,  //-^-
0x00,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,  0x20,0x20,0x20,0x20,  0x20,0x20,0x20,0x20,  //-_-
```

```
0x00,0x00,0x03,0x07,  0x04,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,  //-`-
0x00,0xA0,0xA0,0xA0,  0xE0,0xC0,0x00,0x00,  0x07,0x0F,0x08,0x08,  0x07,0x0F,0x08,0x00,  //-a-
    ASCII 码：0X61
0x04,0xFC,0xFC,0x20,  0x60,0xC0,0x80,0x00,  0x00,0x0F,0x0F,0x08,  0x08,0x0F,0x07,0x00,  //-b-
0xC0,0xE0,0x20,0x20,  0x20,0x60,0x40,0x00,  0x07,0x0F,0x08,0x08,  0x08,0x0C,0x04,0x00,  //-c-
0x80,0xC0,0x60,0x24,  0xFC,0xFC,0x00,0x00,  0x07,0x0F,0x08,0x08,  0x07,0x0F,0x08,0x00,  //-d-
0xC0,0xE0,0xA0,0xA0,  0xA0,0xE0,0xC0,0x00,  0x07,0x0F,0x08,0x08,  0x08,0x0C,0x04,0x00,  //-e-

0x40,0xF8,0xFC,0x44,  0x0C,0x18,0x00,0x00,  0x08,0x0F,0x0F,0x08,  0x00,0x00,0x00,0x00,  //-f-
0xC0,0xE0,0x20,0x20,  0xC0,0xE0,0x20,0x00,  0x27,0x6F,0x48,0x48,  0x7F,0x3F,0x00,0x00,  //-g-
0x04,0xFC,0xFC,0x40,  0x20,0xE0,0xC0,0x00,  0x08,0x0F,0x0F,0x00,  0x00,0x0F,0x0F,0x00,  //-h-
0x00,0x00,0x20,0xEC,  0xEC,0x00,0x00,0x00,  0x00,0x00,0x08,0x0F,  0x0F,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,0x08,0x00,0x00,  //-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,  0x00,0x00,0x00,0x0F,  0x0F,0x00,0x00,0x00,  //-|-
0x00,0x04,0x04,0xBC,  0xF8,0x40,0x40,0x00,  0x00,0x08,0x08,0x0F,  0x07,0x00,0x00,0x00,  //-}-
0x08,0x0C,0x04,0x0C,  0x08,0x0C,0x04,0x00,  0x00,0x00,0x00,0x00,  0x00,0x00,0x00,0x00,  //-~-
ASCII 码：0X7E

};



uchar code ascii_table_5x8[95][5]={
/*全体 ASCII 列表:5x8 点阵*/
```

```
0x00, 0x00, 0x00, 0x00, 0x00, //-  -    //space
0x00, 0x00, 0x4f, 0x00, 0x00, //-!-
0x00, 0x07, 0x00, 0x07, 0x00, //-"-
0x14, 0x7f, 0x14, 0x7f, 0x14, //-#-
0x24, 0x2a, 0x7f, 0x2a, 0x12, //-$-
0x23, 0x13, 0x08, 0x64, 0x62, //-%-
0x36, 0x49, 0x55, 0x22, 0x50, //-&-
0x00, 0x05, 0x07, 0x00, 0x00, //-'-
0x00, 0x1c, 0x22, 0x41, 0x00, //-(-
0x00, 0x41, 0x22, 0x1c, 0x00, //-)-
0x14, 0x08, 0x3e, 0x08, 0x14, //-*-
0x08, 0x08, 0x3e, 0x08, 0x08, //-+-
0x00, 0x50, 0x30, 0x00, 0x00, //-,-
0x08, 0x08, 0x08, 0x08, 0x08, //---
0x00, 0x60, 0x60, 0x00, 0x00, //-.-
0x20, 0x10, 0x08, 0x04, 0x02, //-/-
0x3e, 0x51, 0x49, 0x45, 0x3e, //-0-
0x00, 0x42, 0x7f, 0x40, 0x00, //-1-
0x42, 0x61, 0x51, 0x49, 0x46, //-2-
0x21, 0x41, 0x45, 0x4b, 0x31, //-3-
0x18, 0x14, 0x12, 0x7f, 0x10, //-4-
0x27, 0x45, 0x45, 0x45, 0x39, //-5-
0x3c, 0x4a, 0x49, 0x49, 0x30, //-6-
0x01, 0x71, 0x09, 0x05, 0x03, //-7-
0x36, 0x49, 0x49, 0x49, 0x36, //-8-
0x06, 0x49, 0x49, 0x29, 0x1e, //-9-
0x00, 0x36, 0x36, 0x00, 0x00, //-:-
0x00, 0x56, 0x36, 0x00, 0x00, //-;-
0x08, 0x14, 0x22, 0x41, 0x00, //-<-
0x14, 0x14, 0x14, 0x14, 0x14, //-=-
0x00, 0x41, 0x22, 0x14, 0x08, //->-
0x02, 0x01, 0x51, 0x09, 0x06, //-?-
0x32, 0x49, 0x79, 0x41, 0x3e, //-@-
0x7e, 0x11, 0x11, 0x11, 0x7e, //-A-
0x7f, 0x49, 0x49, 0x49, 0x36, //-B-
0x3e, 0x41, 0x41, 0x41, 0x22, //-C-
0x7f, 0x41, 0x41, 0x22, 0x1c, //-D-
0x7f, 0x49, 0x49, 0x49, 0x41, //-E-
0x7f, 0x09, 0x09, 0x09, 0x01, //-F-
0x3e, 0x41, 0x49, 0x49, 0x7a, //-G-
0x7f, 0x08, 0x08, 0x08, 0x7f, //-H-
0x00, 0x41, 0x7f, 0x41, 0x00, //-I-
0x20, 0x40, 0x41, 0x3f, 0x01, //-J-
0x7f, 0x08, 0x14, 0x22, 0x41, //-K-
0x7f, 0x40, 0x40, 0x40, 0x40, //-L-
0x7f, 0x02, 0x0c, 0x02, 0x7f, //-M-
```

```
0x7f, 0x04, 0x08, 0x10, 0x7f, //-N-
0x3e, 0x41, 0x41, 0x41, 0x3e, //-O-
0x7f, 0x09, 0x09, 0x09, 0x06, //-P-
0x3e, 0x41, 0x51, 0x21, 0x5e, //-Q-
0x7f, 0x09, 0x19, 0x29, 0x46, //-R-
0x46, 0x49, 0x49, 0x49, 0x31, //-S-
0x01, 0x01, 0x7f, 0x01, 0x01, //-T-
0x3f, 0x40, 0x40, 0x40, 0x3f, //-U-
0x1f, 0x20, 0x40, 0x20, 0x1f, //-V-
0x3f, 0x40, 0x38, 0x40, 0x3f, //-W-
0x63, 0x14, 0x08, 0x14, 0x63, //-X-
0x07, 0x08, 0x70, 0x08, 0x07, //-Y-
0x61, 0x51, 0x49, 0x45, 0x43, //-Z-
0x00, 0x7f, 0x41, 0x41, 0x00, //-[-
0x02, 0x04, 0x08, 0x10, 0x20, //-\-
0x00, 0x41, 0x41, 0x7f, 0x00, //-]-
0x04, 0x02, 0x01, 0x02, 0x04, //-^-
0x40, 0x40, 0x40, 0x40, 0x40, //-_-
0x01, 0x02, 0x04, 0x00, 0x00, //-`-
0x20, 0x54, 0x54, 0x54, 0x78, //-a-
0x7f, 0x48, 0x48, 0x48, 0x30, //-b-
0x38, 0x44, 0x44, 0x44, 0x44, //-c-
0x30, 0x48, 0x48, 0x48, 0x7f, //-d-
0x38, 0x54, 0x54, 0x54, 0x58, //-e-
0x00, 0x08, 0x7e, 0x09, 0x02, //-f-
0x48, 0x54, 0x54, 0x54, 0x3c, //-g-
0x7f, 0x08, 0x08, 0x08, 0x70, //-h-
// 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-
// 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,//-~-
};



uchar code bmp1[]={
/*--  调入了一幅图像：D:\e\新开发部\显示图案收藏\19264G-329 小熊及 JERRY.bmp  --*/
/*--  宽度 x 高度=192x64  --*/

0xFF,0x01,0x01,0x01,0x01,0x01,0x01,0x81,0xC1,0xE1,0xF1,0xF9,0xF9,0xF9,0xFD,0xFD,
0xFD,0xFD,0xF9,0xF9,0xF1,0xF1,0xE1,0xC1,0xC1,0x41,0x41,0x41,0x41,0x41,0x41,0x41,
0x41,0x41,0xC1,0x81,0x81,0x81,0xC1,0xE1,0xF1,0xF9,0xF9,0xFD,0xFD,0xFD,0xFD,0xFD,
0xFD,0xF9,0xF9,0xF1,0xE1,0xC1,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,
0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0xC1,0xE1,0xE1,0xF1,0xF1,0xF9,0xF9,0xF9,
0xF9,0xF9,0xF9,0xF1,0xF1,0xE1,0xC1,0x81,0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01,
                                        .
                                        .
                                        .
};
```

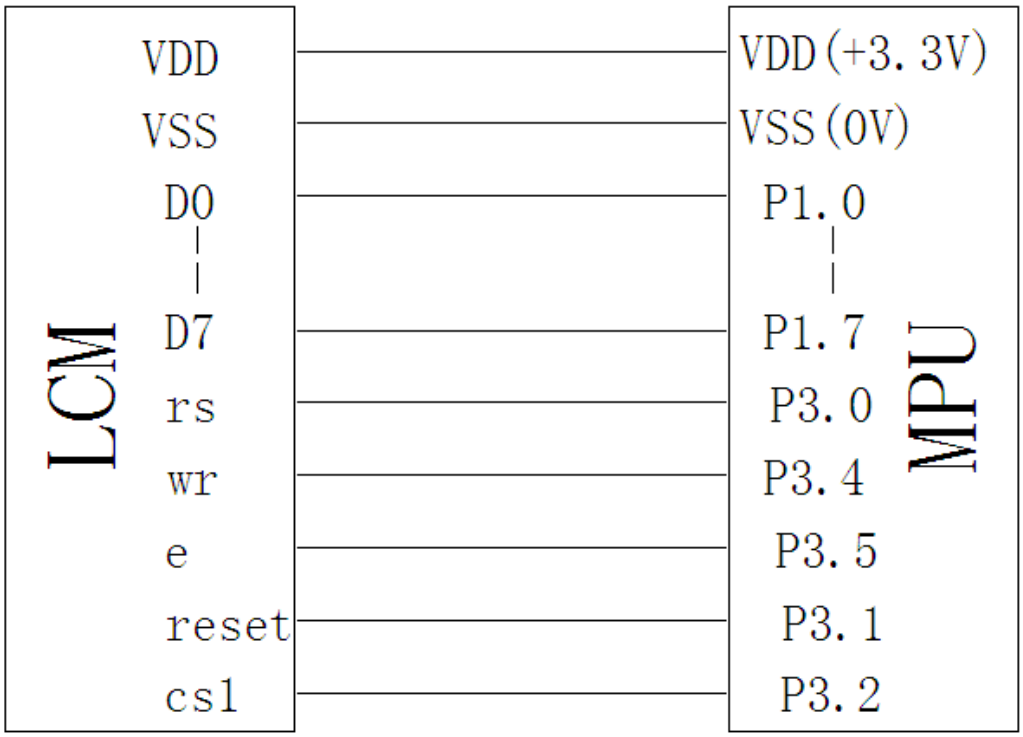
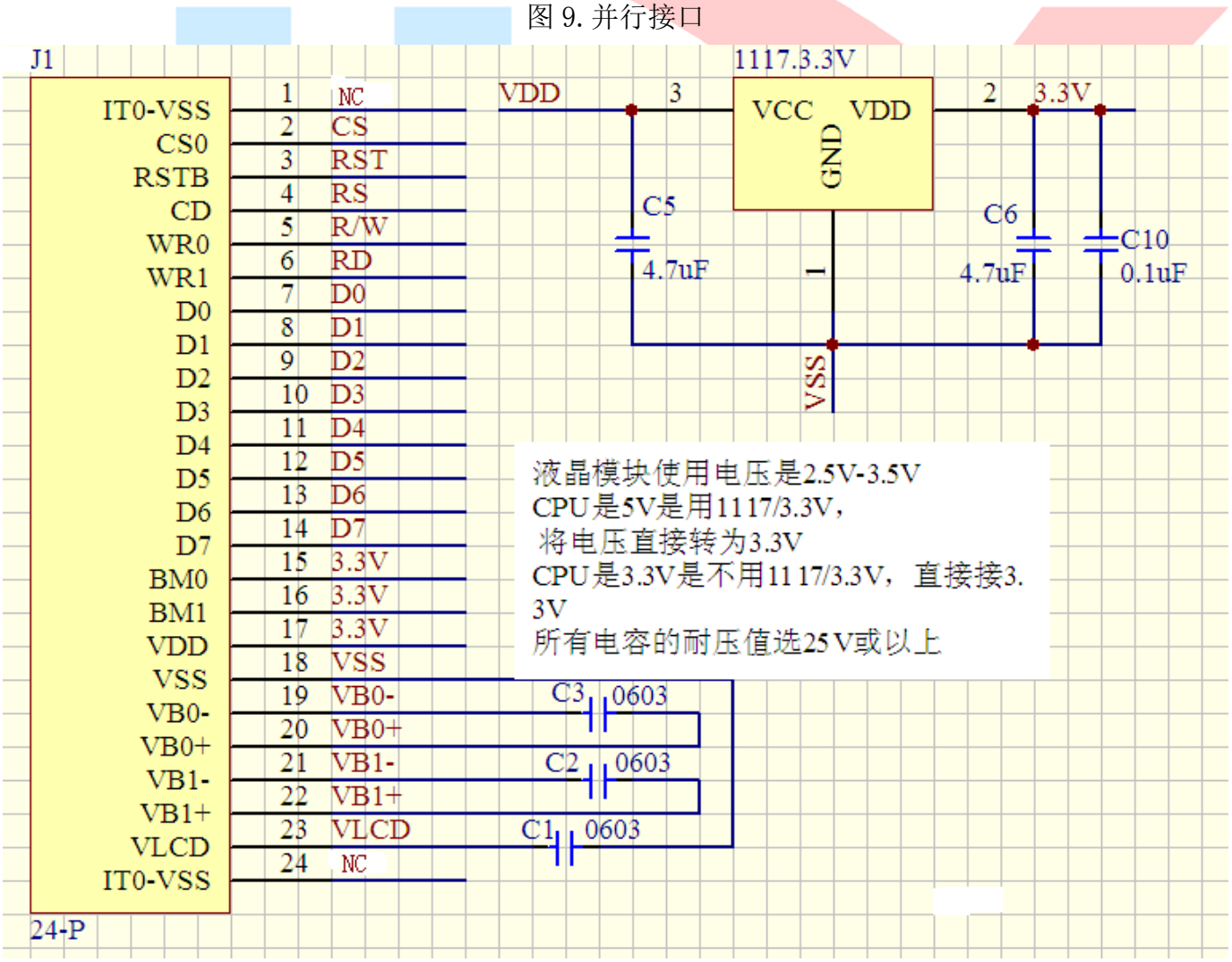

LCM
VDD
VSS
D0
D7
rs
wr
e
reset
csl
MPU
VDD(+3.3V)
VSS(0V)
P1.0
P1.7
P3.0
P3.4
P3.5
P3.1
P3.2

图 9. 并行接口


J1
IT0-VSS
CS0
RSTB
CD
WR0
WR1
D0
D1
D2
D3
D4
D5
D6
D7
BM0
BM1
VDD
VSS
VB0-
VB0+
VB1-
VB1+
VLCD
IT0-VSS
24-P
1 NC
2 CS
3 RST
4 RS
5 R/W
6 RD
7 D0
8 D1
9 D2
10 D3
11 D4
12 D5
13 D6
14 D7
15 3.3V
16 3.3V
17 3.3V
18 VSS
19 VB0-
20 VB0+
21 VB1-
22 VB1+
23 VLCD
24 NC
1117.3.3V
VDD
3
VCC
VDD
GND
2
3.3V
C5
4.7uF
C6
4.7uF
C10
0.1uF
VSS
C3 0603
C2 0603
C1 0603
液晶模块使用电压是2.5V-3.5V
CPU是5V是用1117/3.3V，
将电压直接转为3.3V
CPU是3.3V是不用1117/3.3V，直接接3.3V
所有电容的耐压值选25V或以上

并行原理图

并行程序与串行只是接口定义、写数据和命令不一样，其它都一样

并行程序：

```
#include <reg52.h>
#include <intrins.h>
#include <Chinese_code.h>

sbit cs1=P3^2;      /*接口定义*/
sbit reset=P3^1;    /*接口定义*/
sbit rs=P3^0;       /*接口定义*/
sbit e=P3^5;        /*接口定义*/
sbit wr=P3^4;       /*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*/


//写指令到 LCD 模块
void transfer_command(int data1)
{
    cs1=0;
    rs=0;
    wr=0;
    e=0;
    P1=data1;
    e=1;
    e=0;
    P1=0x00;
    cs1=1;
}
//写数据到 LCD 模块
void transfer_data(int data1)
{
    cs1=0;
    rs=1;
    wr=0;
    e=0;
    P1=data1;
    e=1;
    e=0;
    P1=0x00;
    cs1=1;
}
```

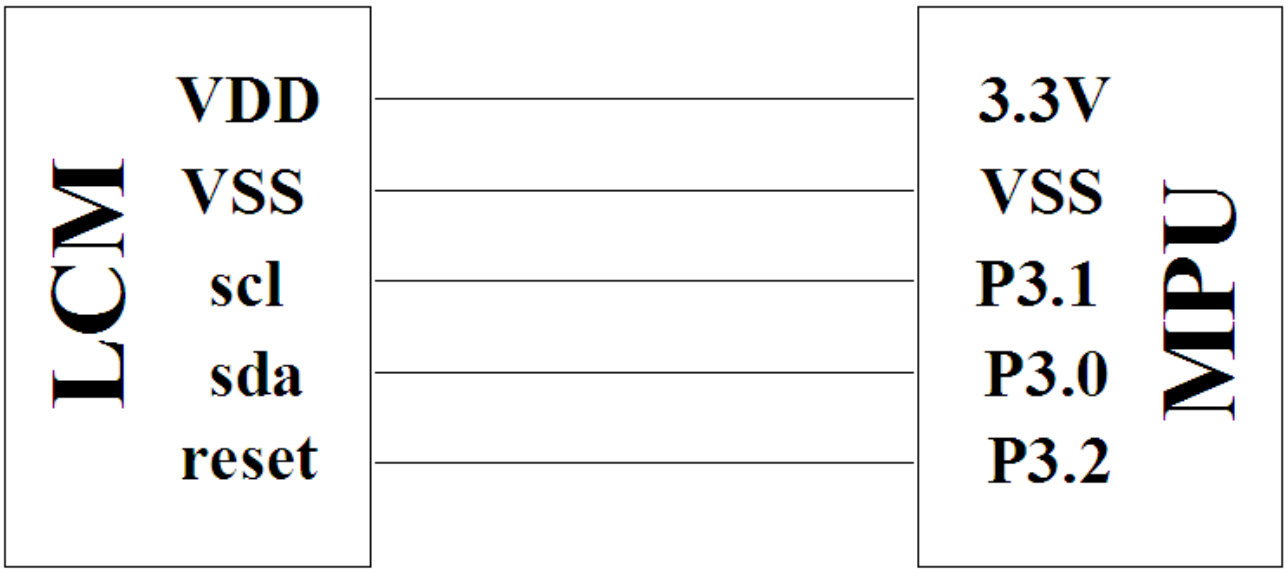
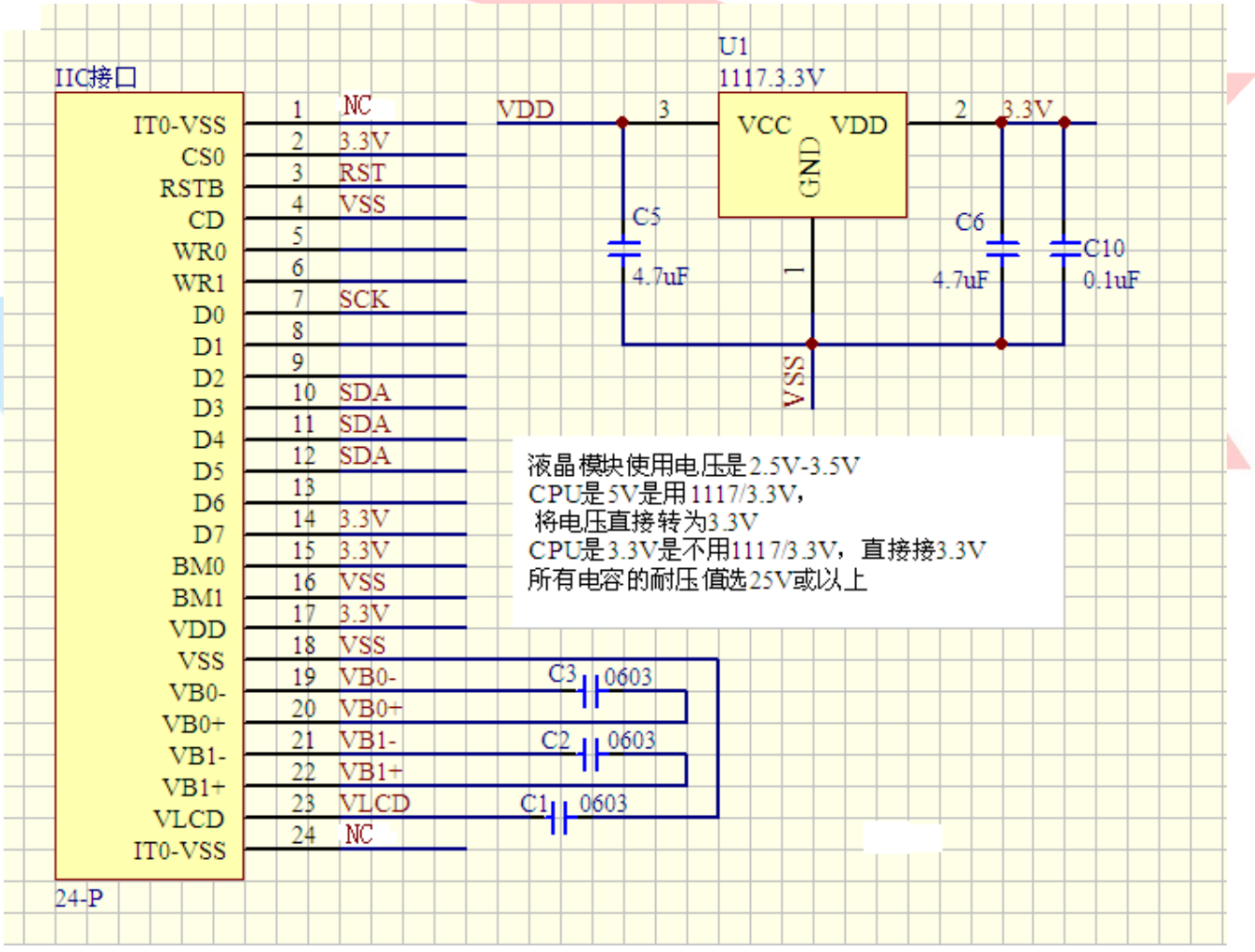


图 9. IIC 接口



IIC 原理图

IIC 程序与串、并行接口定义、写数据和命令不一样，取模代码是一样的

IIC 程序：

```
//   液晶演示程序 JLX19264G-260，IIC 接口！
//   驱动 IC 是:UC1604c

#include <reg52.h>
#include <intrins.h>
#include <Chinese_code.h>

sbit  reset=P3^2;
sbit scl=P3^1;
sbit sda=P3^0;
sbit  key=P2^0;


void delay_us(int i);
void delay(int i);


//延时 1
void delay(int i)
{
 int j,k;
 for(j=0;j<i;j++)
 for(k=0;k<110;k++);
}

//延时 2
void delay_us(int i)
{
 int j,k;
 for(j=0;j<i;j++)
 for(k=0;k<10;k++);
}

void waitkey()
{
repeat:
     if(key==1)goto repeat;
     else delay(400);
}


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*/
}

//写命令到液晶显示模块
void transfer_command(uchar com)
{
     start_flag();
     transfer(0x7c);
     transfer(com);
     stop_flag();
}

//写数据到液晶显示模块
void transfer_data(uchar dat)
{
     start_flag();
     transfer(0x7e);
     transfer(dat);
     stop_flag();
}


//LCD 模块初始化
void initial_lcd()
{
     reset=0;        //低电平复位
     delay(100);
     reset=1;          //复位完毕
     delay(800);
     transfer_command(0xe2); //软复位
     delay(200);
     transfer_command(0x2f);  //打开内部升压
     delay(200);

     transfer_command(0x81);  //微调对比度
     transfer_command(0x56);  //微调对比度的值，可设置范围 0x00～0xFF
     transfer_command(0xeb);  //1/9 偏压比（bias）
     transfer_command(0xc4);  //行列扫描顺序：从上到下 oxc2
     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<8;i++)
    {
        lcd_address(1+i,1);
        for(j=0;j<192;j++)
        {
            transfer_data(0x00);
        }
    }
}

void display_graphic_192x64(uchar *dp)
{
    uchar i,j;
    for(i=0;i<8;i++)
    {
        lcd_address(i+1,1);
        for(j=0;j<192;j++)
        {
            transfer_data(*dp);
            dp++;
        }
    }
}

//==================display a piture of 128*64 dots================
void full_display(uchar data_left,uchar data_right)
{
    int i,j;
    for(i=0;i<8;i++)
    {
        lcd_address(i+1,1);
        for(j=0;j<96;j++)
        {
            transfer_data(data_left);
            transfer_data(data_right);
        }
    }
}

//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 *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++;
        }
    }
}

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++;
    }
}
```

```
//显示一串 5x8 点阵的字符串
//括号里的参数分别为（页，列，是否反显，数据指针）
void display_string_5x8(uint page,uint column,uchar reverse,uchar *text)
{
    uchar i=0,j,k,data1;
    while(text[i]>0x00)
    {
        if((text[i]>=0x20)&&(text[i]<=0x7e))
        {
            j=text[i]-0x20;
            lcd_address(page,column);
            for(k=0;k<5;k++)
            {
                if(reverse==1)    data1=~ascii_table_5x8[j][k];
                else data1=ascii_table_5x8[j][k];
                transfer_data(data1);
            }
            if(reverse==1)    transfer_data(0xff);
            else transfer_data(0x00);
            i++;
            column+=6;
        }
        else
        i++;
    }
}

void display_string_5x8_1(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_5x8[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++;
    }
}



void main(void)
{
    while(1)
    {
        initial_lcd();
        clear_screen();
        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(3,100,1,"1.Graphic      ");
        display_string_5x8(4,100,0,"2.Chinese       " );
        display_string_5x8(5,100,0,"3.Movie         ");
        display_string_5x8(6,100,0,"4.Contrast      ");
        display_string_5x8(7,100,0,"5.Mirror        ");
        display_string_5x8(8,1,1,"    PRE    USER     DEL     NEW     ");
        display_string_5x8(8,59,0," ");
        display_string_5x8(8,94,0," ");
        display_string_5x8(8,97+48,0," ");
        waitkey();

        clear_screen();                                 //clear all dots
        display_graphic_192x64(bmp1);
        waitkey();
        clear_screen();
        display_graphic_32x32(1,1,cheng1);            //在第 1 页，第 49 列显示单个汉字"成"
        waitkey();
        clear_screen();                                 //clear all dots
        display_graphic_16x16(5,1,zhuang1);       //在第 5 页，第 1 列显示单个汉字"状"
        display_graphic_16x16(5,(1+16),tai1);         //在第 5 页，第 17 列显示单个汉字"态"
        display_graphic_8x16(5,(1+16*2),mao_hao);     //在第 5 页，第 25 列显示单个字符":"
        display_graphic_16x16(5,(1+16*2+8),shi1);     //在第 5 页，第 41 列显示单个汉字"使"
        display_graphic_16x16(5,(1+16*3+8),yong1);    //在第 5 页，第 49 列显示单个汉字"用"
        display_graphic_8x16(5,(89),num0);            //在第 5 页，第 89 列显示单个数字"0"
        display_graphic_8x16(5,(89+8*1),num0);        //在第 5 页，第 97 列显示单个数字"0"
        display_graphic_8x16(5,(89+8*2),mao_hao);     //在第 5 页，第 105 列显示单个字符":"
        display_graphic_8x16(5,(89+8*3),num0);        //在第 5 页，第 113 列显示单个数字"0"
        display_graphic_8x16(5,(89+8*4),num0);        //在第 5 页，第 121 列显示单个数字"0"
        waitkey();
        clear_screen();                                   //clear all dots
        display_string_8x16(1,1,"(<\"0123456abt~`!@#$%^\">)");//在第 1 页，第 1 列显示字符串
        display_string_8x16(3,1,"{[(<\" '&*|\\@#_-+=' \">)]}");//在第*页，第*列显示字符串
        display_string_5x8_1(5,1,"[!#$%&'()*+,-./0123456789:;<=>?]");
        display_string_5x8_1(6,1,"[ABCDEFGHIJKLMNOPQRSTUVWXYZabcd]");
        display_string_5x8_1(7,1,"(abcdefghijklmnopqrstuvwxyzabcd)");
        display_string_5x8_1(8,1,"{[(<\" '&*|\\@abcde012#_-+=' \">)]}");
        waitkey();
        full_display(0xff,0xff);
        waitkey();
        full_display(0x55,0xaa);
        waitkey();
        full_display(0xaa,0x55);
        waitkey();
        full_display(0xff,0x00);
        waitkey();
        full_display(0x00,0xff);
        waitkey();
    }
}
```