

JLX19264G-26018-PN 使用说明书

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	3~5
4	基本原理	5~6
5	技术参数	6
6	时序特性	7~11
7	指令功能及硬件接口与编程案例	11~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 JLX19264G-26018 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-26018 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-26018 图像型点阵液晶模块的特性

2.1 结构轻、薄、带背光、铁框、PCB 板。

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 ($3.3V \times 0.3mA$)，带背光不大于 100mW ($3.3V \times 45mA$)；

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 15 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 31 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:并行、串行、IIC 接口。

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

3. 外形尺寸及接口引脚功能

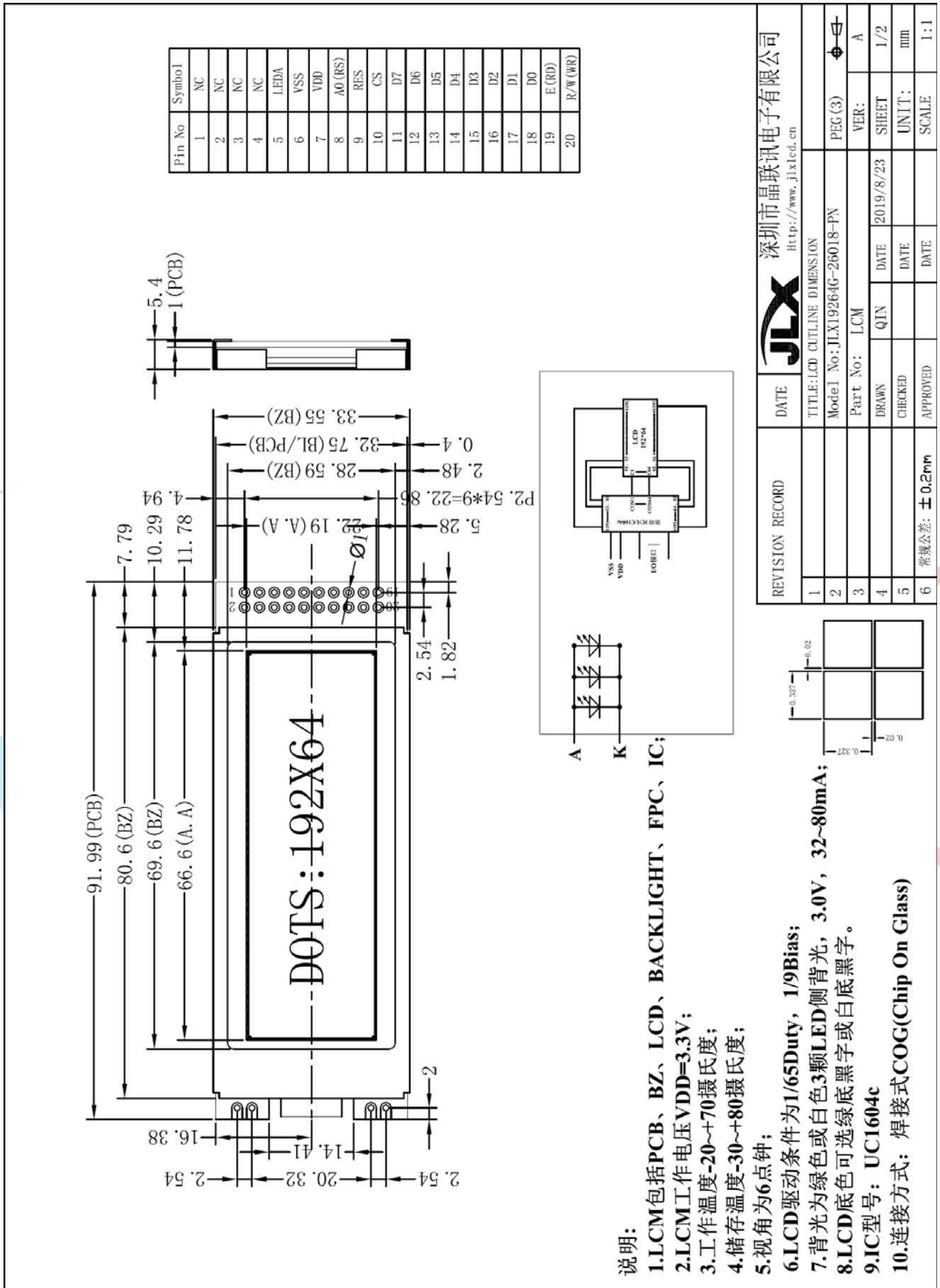


图 1. 外形尺寸

3.2 模块的接口引脚功能

3.2.1 并行时接口引脚功能

引 线 号	符 号	名 称	功 能
1	NC		空脚
2	NC		空脚
3	NC		空脚
4	NC		空脚
5	LEDA	背光电源	供电电源正极
6	VSS	接地	0V
7	VDD	电路电源	供电电源正极
8	A0 (RS)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器 (IC 资料上所写为“CD”)
9	RST	复位	低电平复位, 复位完成后, 回到高电平, 液晶模块开始工作
10	CS	片选	低电平片选
11	D7	I/O	数据总线 DB7
12	D6	I/O	数据总线 DB6
13	D5	I/O	数据总线 DB5
14	D4	I/O	数据总线 DB4
15	D3	I/O	数据总线 DB3
16	D2	I/O	数据总线 DB2
17	D1	I/O	数据总线 DB1
18	D0	I/O	数据总线 DB0
19	RD (E)	使能信号	6800 时序: 使能信号
20	WR	读/写	6800 时序: H: 读数据 L: 写数据

表 1: 模块并行接口引脚功能

3.2.2 四线串行时接口引脚功能

引 线 号	符 号	名 称	功 能
1	NC		空脚
2	NC		空脚
3	NC		空脚
4	NC		空脚
5	LEDA	背光电源	供电电源正极
6	VSS	接地	0V
7	VDD	电路电源	供电电源正极
8	A0 (RS)	寄存器选择信号	H: 数据寄存器 0: 指令寄存器 (IC 资料上所写为“CD”)
9	RST	复位	低电平复位, 复位完成后, 回到高电平, 液晶模块开始工作
10	CS	片选	低电平片选
11	D7	I/O	串行接口, 此引脚不用, 建议接 VDD
12	D6	I/O	串行接口, 此引脚不用, 建议接 VDD
13-15	D3-D5 (SDA)	I/O	串行数据
16	D2	I/O	串行接口, 此引脚不用, 建议接 VDD
17	D1	I/O	串行接口, 此引脚不用, 建议接 VDD
18	D0 (SCK)	I/O	串行时钟
19	RD (E)	使能信号	串行接口, 此引脚不用, 建议接 VDD
20	WR	读/写	串行接口, 此引脚不用, 建议接 VDD

表 2：4 线 SPI 串行接口引脚功能

3.2.3 I²C 总线时接口引脚功能

引 线 号	符 号	名 称	功 能
1	NC		空脚
2	NC		空脚
3	NC		空脚
4	NC		空脚
5	LEDA	背光电源	供电电源正极
6	VSS	接地	0V
7	VDD	电路电源	供电电源正极
8	A0 (RS)	寄存器选择信号	I2C 接口，此引脚接 VSS
9	RST	复位	低电平复位，复位完成后，回到高电平，液晶模块开始工作
10	CS	片选	I2C 接口，此引脚接 VDD
11	D7	I/O	I2C 接口，此引脚接 VDD
12	D6	I/O	I2C 接口，此引脚不用，建议接 VDD
13-15	D5-D3 (SDA)	I/O	串行数据
16	D2	I/O	I2C 接口，此引脚不用，建议接 VDD
17	D1	I/O	I2C 接口，此引脚不用，建议接 VDD
18	D0 (SCK)	I/O	串行时钟
19	RD (E)	使能信号	I2C 接口，此引脚不用，建议接 VDD
20	WR	读/写	I2C 接口，此引脚不用，建议接 VDD

表 3：I²C 总线接口引脚功能

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 绑定在 LCD 玻璃上 (这种加工工艺叫 COG)。

4.2 工作电图:

图 2 是 JLX19264G-26018 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。

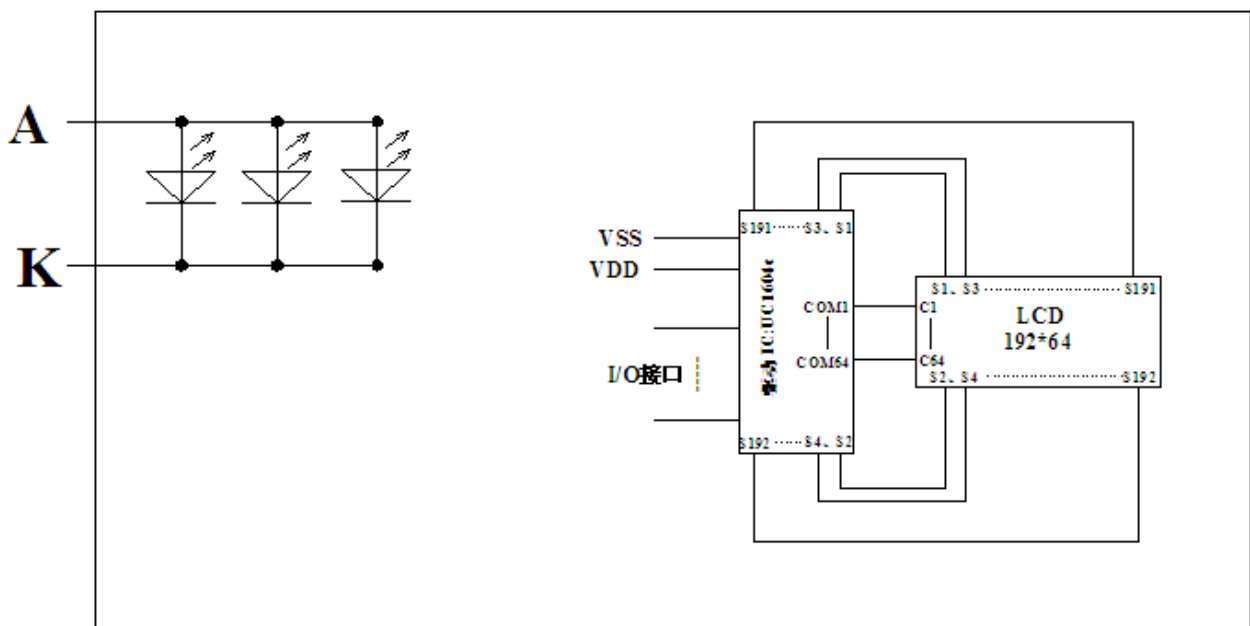


图 2: JLX19264G-26018 图像点阵型液晶模块的电路框图

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	°C
储存温度		-30		+80	°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		2.4	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	V00	I00 = 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	45	60	mA

表 3：直流（DC）参数

6. 读写时序特性

6.1 并行接口：(8080)

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

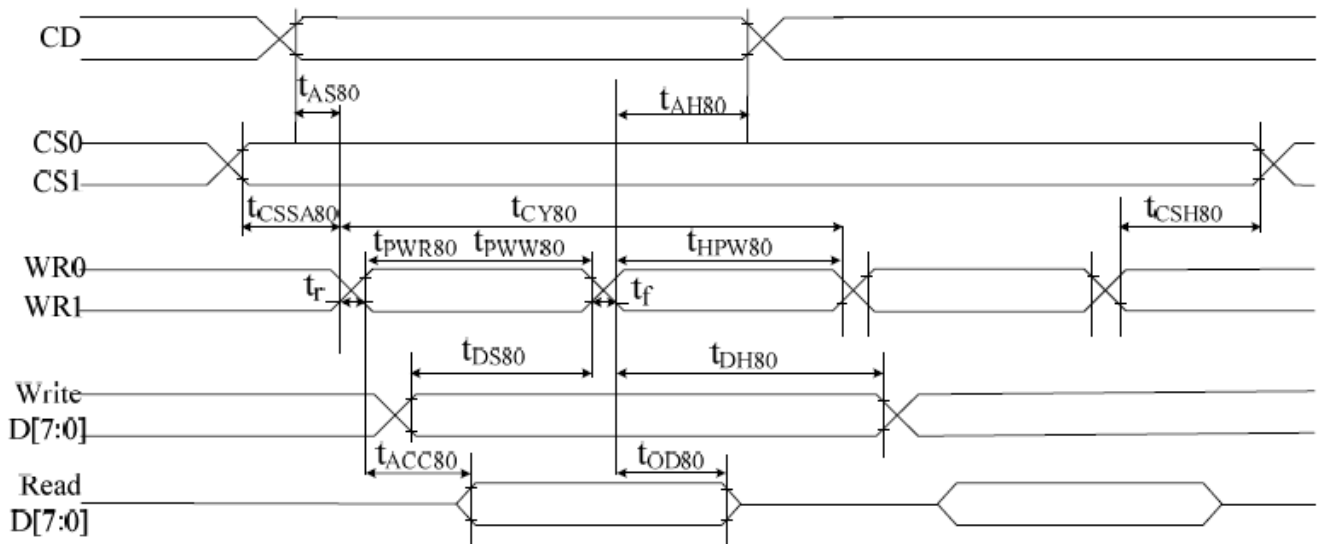


FIGURE 15: Parallel Bus Timing Characteristics (for 8080 MCU)

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

6.2 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}	WR0, WR1	System Cycle time Pulse width		170 / 110 70 / 40	—	nS
t _{HPW80}		High pulse width		70 / 40		
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}	WR0, WR1	System cycle time Pulse width		270 / 190 120 / 80	—	nS
t _{HPW80}		High pulse width		120 / 80		
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

6.3 并行接口：(6800)

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

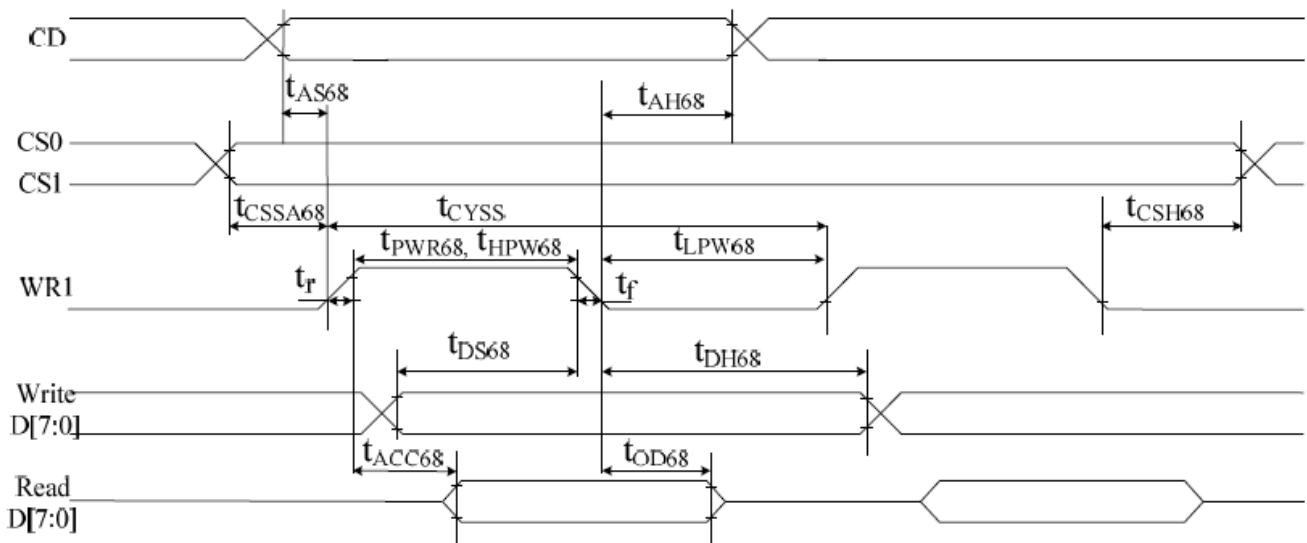


FIGURE 16: Parallel Bus Timing Characteristics (for 6800 MCU)

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

6.4 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}	CD	Address setup time		5	—	nS
t _{AH68}	CD	Address hold time		10	—	nS
t _{CSSA68}	CS1, CS0	Chip select setup time		5	—	nS
t _{CSH68}	CS1, CS0	Chip select hold time		5	—	nS
t _{CY68}		System cycle time		170 / 110	—	nS
t _{PWR68} / t _{PWW68}	WR1	Pulse width		70 / 40	—	nS
t _{HPW68}	WR1	High pulse width		70 / 40	—	nS
t _{DS68}	D7~D0	Data setup time		35	—	nS
t _{DH68}	(Write)	Data hold time		5	—	nS
t _{ACC68}	D7~D0	Read access time	C _L = 100pF	—	70	nS
t _{OD68}	(Read)	Output disable time		—	40	nS
(1.7V ≤ V _{DD} < 2.5V, Ta = -30 to +85°C)						
(Read / Write)						
t _{AS68}	CD	Address setup time		5	—	nS
t _{AH68}	CD	Address hold time		10	—	nS
t _{CSSA68}	CS1, CS0	Chip select setup time		5	—	nS
t _{CSH68}	CS1, CS0	Chip select hold time		5	—	nS
t _{CY68}		System cycle time		270 / 190	—	nS
t _{PWR68} / t _{PWW68}	WR1	Pulse width		120 / 80	—	nS
t _{HPW68}	WR1	High pulse width		120 / 80	—	nS
t _{DS68}	D7~D0	Data setup time		60	—	nS
t _{DH68}	(Write)	Data hold time		5	—	nS
t _{ACC68}	D7~D0	Read access time	C _L = 100pF	—	115	nS
t _{OD68}	(Read)	Output disable time		—	80	nS

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

6.5 串行接口：

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

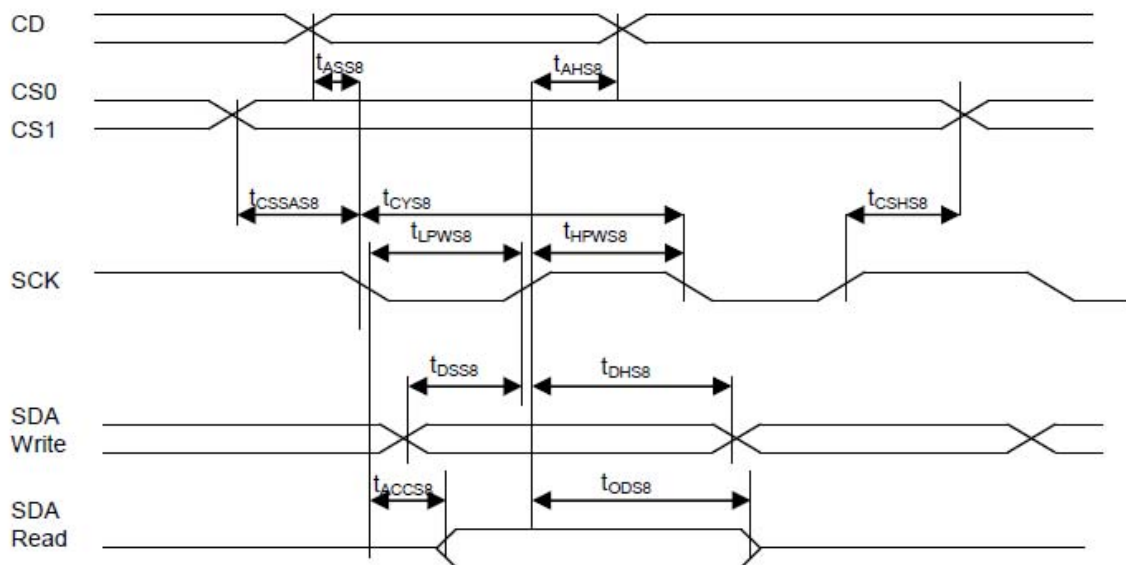


FIGURE 17: Serial Bus Timing Characteristics (for S8)

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

6.6 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)						
t_{ASS8}	CD	Address setup time		5	—	nS
t_{AHS8}	CD	Address hold time		10	—	nS
t_{CSSAS8}	CS1, CS0	Chip select setup time		5	—	nS
t_{CSHS8}	CS1, CS0	Chip select hold time		5	—	nS
t_{CYS8}	SCK	System Cycle time		190 / 70	—	nS
t_{LPWS8}	SCK	Low pulse width		80 / 20	—	nS
t_{HPWS8}	SCK	High pulse width		80 / 20	—	nS
t_{DSS8}	SDA (Write)	Data setup time		20	—	nS
t_{DHS8}	SDA (Write)	Data hold time		10	—	nS
t_{ACC8}	SDA (Read)	Read access time	C _L = 100pF	—	80	nS
t_{ODS8}	SDA (Read)	Output disable time	C _L = 100pF	—	30	nS
(1.7V ≤ V _{DD} < 2.5V, Ta = -30 to +85°C)						
t_{ASS8}	CD	Address setup time		5	—	nS
t_{AHS8}	CD	Address hold time		10	—	nS
t_{CSSAS8}	CS1, CS0	Chip select setup time		10	—	nS
t_{CSHS8}	CS1, CS0	Chip select hold time		10	—	nS
t_{CYS8}	SCK	System Cycle time		230 / 110	—	nS
t_{LPWS8}	SCK	Low pulse width		100 / 40	—	nS
t_{HPWS8}	SCK	High pulse width		100 / 40	—	nS
t_{DSS8}	SDA (Write)	Data setup time		24	—	nS
t_{DHS8}	SDA (Write)	Data hold time		10	—	nS
t_{ACC8}	SDA (Read)	Read access time	C _L = 100pF	—	100	nS
t_{ODS8}	SDA (Read)	Output disable time	C _L = 100pF	—	60	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)

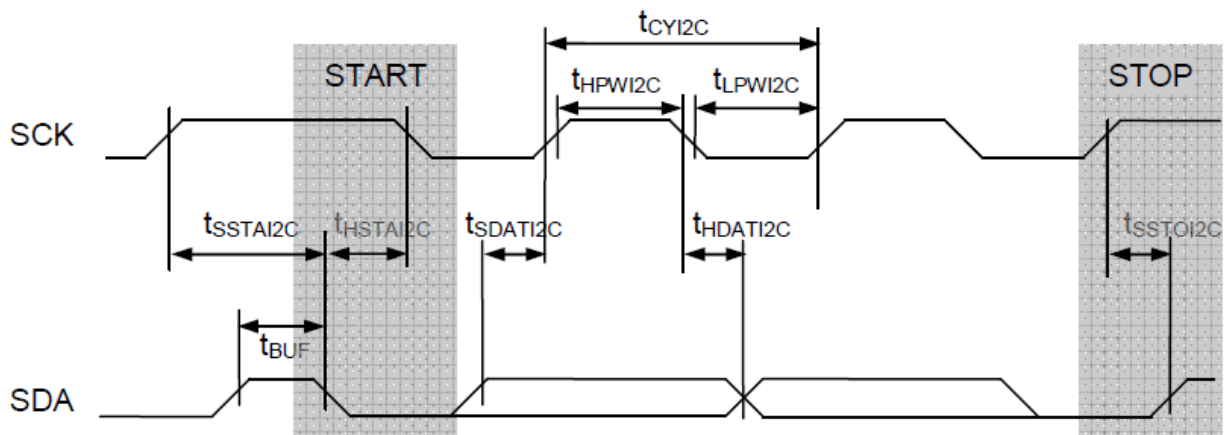
FIGURE 19: Serial bus timing characteristics (for I²C)

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):

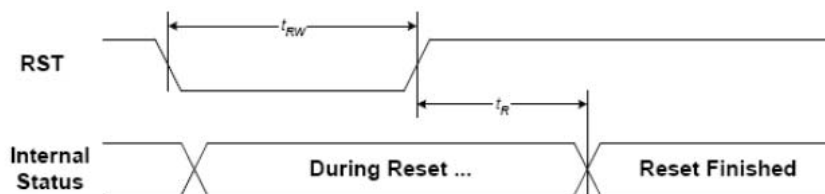


FIGURE 20: Reset Characteristics

(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
	设置列地址高4位 Set Column Address MSB	0	0	0	0	0	1	CA7	CA6	CA5	CA4	

												达：0x16,0x03
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, 可以理解为微调用对比度值, 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
	0	1	ID	MX	MY	WA	DE	WS	MD	MS	2. 读状态1 3. 读状态2

	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
	1	1	#	#	#	#	#	#	#	#	2. 数据

7.3 点阵与 DD RAM(显示数据存储单元)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：

D0	0	1	1	1		0
D1	1	0	0	0		0
D2	0	0	0	0		0
D3	0	1	1	1		0
D4	1	0	0	0		0
-						

Display data RAM
(显示数据存储单元)

COM0						
COM1						
COM2						
COM3						
COM4						
-						

Liquid crystal display
(液晶屏)

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

⇒ 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

点亮液晶模块的步骤

硬件准备:
开发板 (或专门设计的主板)、单片机、电源、连接线、仿真器或程序下载器 (又名烧录器)

正确地接线
根据说明书正确地与开发板连接, 连接的线包括: 液晶模块电源线、背光源电源线、IO端口 (接口)
IO端口包括: 并口时: CS、RESET、RW、E、RS、D0--D7, 串口时: CS、SCLK、SDA、RESET、RS

编写软件
背光给合适的直流电可以点亮, 但液晶屏里面没有程序, 只给电不能让液晶屏显示 (我们通常说“点亮”), 程序须另外编写, 并烧录 (下载) 到单片机里液晶模块才能工作。

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:

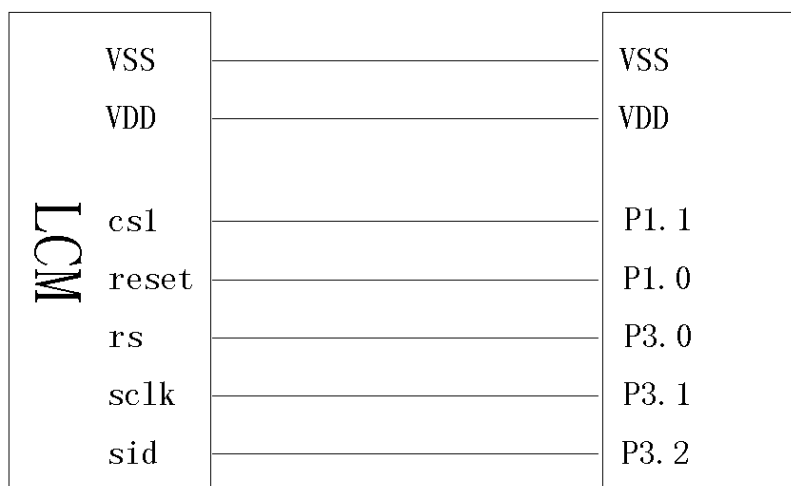


图 8. 串行接口

7.5.1 程序

```
// 液晶演示程序 JLX19264G-26018，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 scl=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[];  
void delay_us(int i);  
void delay(int i);  
  
uchar code chengl[]={  
//-- 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 zhuangl[]={  
//-- 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 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 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, 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(0xc4); //行扫描顺序：从上到下 0xc2  
    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 picture 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]);
            }
            i++;
            column+=6;
        }
        else
            i++;
    }
}
```

列地址, 最后为数据

```
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_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),tail); //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,"[ABCDEFGHJKLMNOPQRSTUVWXYZabcd]");
    display_string_5x8_1(7,1,"(abcdefghijklmnopqrstuvwxyza)");
    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, 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, //~
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: C:\Documents and Settings\Administrator\桌面\12832J.bmp --*/
/*-- 宽度 x 高度=128x32 --*/
/*
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x4C, 0x54, 0x54, 0x64, 0x00,
0x28, 0x54, 0x54, 0x78, 0x40, 0x00, 0x00, 0x44, 0x7C, 0x48, 0x04, 0x00, 0x28, 0x54, 0x54, 0x78,
0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x08, 0xFA, 0x00, 0x00, 0x50, 0xA8, 0xA8, 0xF0, 0x80, 0x00, 0x00,
0x98, 0xA8, 0xA8, 0xC8, 0x00, 0x70, 0x88, 0x88, 0x88, 0x70, 0x00, 0x88, 0xF8, 0x88, 0x08, 0xF0,
0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x80, 0x60, 0x10, 0x88, 0x48,
0x44, 0x44, 0x42, 0x82, 0x02, 0x02, 0xC2, 0x42, 0x44, 0x44, 0x48, 0x48, 0x10, 0x60, 0x80, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x01, 0x80, 0x60, 0x10, 0xC8, 0x48, 0x44, 0x44, 0x42, 0x42,
0x02, 0x02, 0x02, 0x82, 0x44, 0x44, 0x88, 0x08, 0x10, 0x60, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0xC1, 0x00, 0x00, 0xC1, 0x00,
0x04, 0x04, 0x8A, 0x71, 0x00, 0x00, 0xCF, 0x08, 0x04, 0x04, 0x88, 0x70, 0x00, 0x00, 0xC1, 0x3E,
0x00, 0x00, 0x00, 0x00, 0x80, 0x40, 0x40, 0x00, 0x00, 0x00, 0x40, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x3E, 0xC1, 0x00, 0x00, 0xCF, 0x08, 0x04, 0x04, 0x88, 0x70,
```

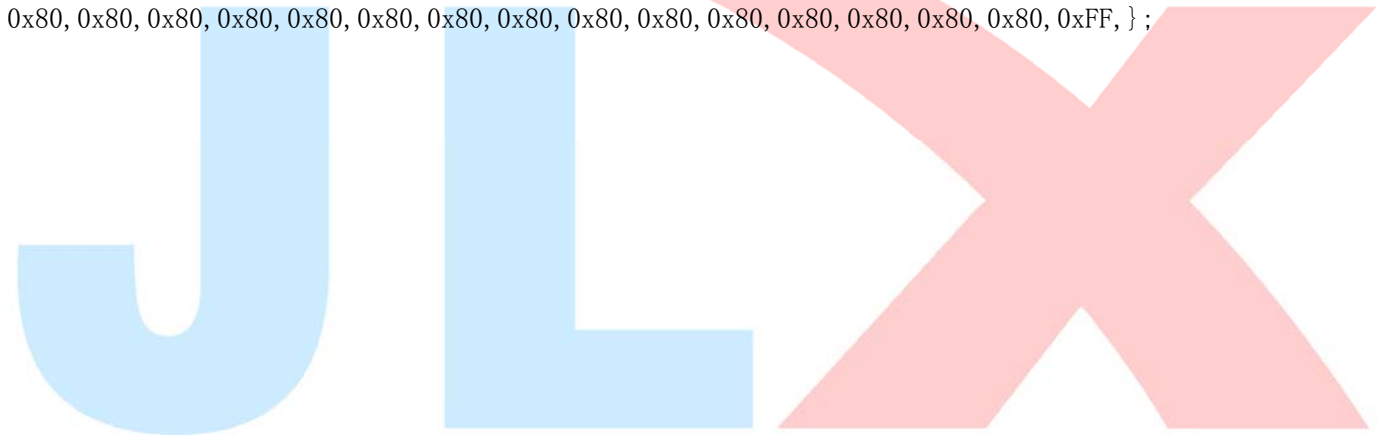
0x00, 0x00, 0x7F, 0x80, 0x00, 0x00, 0x80, 0x7F, 0x00, 0x00, 0xC1, 0x3E, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x04, 0x08, 0x09,
0x11, 0x11, 0x10, 0x20, 0x20, 0x20, 0x20, 0x11, 0x11, 0x11, 0x08, 0x08, 0x04, 0x03, 0x00, 0x00,
0x00, 0x00, 0x00, 0x11, 0x1F, 0x11, 0x01, 0x00, 0x00, 0x11, 0x1F, 0x10, 0x00, 0x00, 0x00, 0x11,
0x1F, 0x12, 0x01, 0x00, 0x1F, 0x01, 0x1F, 0x01, 0x1E, 0x00, 0x11, 0x1F, 0x11, 0x01, 0x1E, 0x10,
0x00, 0x0E, 0x15, 0x15, 0x16, 0x00, 0x00, 0x13, 0x15, 0x15, 0x19, 0x00, 0x00, 0x13, 0x15, 0x15,
0x19, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x04, 0x08, 0x09, 0x11, 0x11, 0x10, 0x20,
0x20, 0x20, 0x20, 0x10, 0x11, 0x11, 0x08, 0x08, 0x04, 0x03, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
*/

/*-- 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,
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,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01,
0x01, 0x01, 0x81, 0x81, 0xF1, 0x79, 0x07, 0xC3, 0x81, 0x81, 0x81, 0x03, 0x07, 0x3D, 0xC1, 0x01,
0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x81, 0x81, 0x81,
0x81, 0xC1, 0x41, 0x41, 0xC1, 0xC1, 0x81, 0x81, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFE, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x1F,
0x0F, 0x07, 0x07, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x07, 0x07, 0x0F, 0x1F, 0x7F, 0xFF, 0xFF, 0xFF,
0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0x80, 0xF0, 0xF0, 0x60, 0x20, 0x10, 0x10, 0x08, 0x08,
0x0C, 0x04, 0x04, 0x04, 0x04, 0x06, 0x06, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x07, 0x0F, 0x0F, 0x0F,
0x1F, 0x3F, 0x3F, 0x7F, 0xFF, 0xFF, 0xFF, 0xFF, 0x7E, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x01, 0x01, 0x83, 0xC3, 0x67, 0x27, 0x17, 0x1F, 0x0F, 0x0F, 0x0F, 0x07, 0x07, 0x07,
0x07, 0x07, 0x07, 0x07, 0x07, 0x06, 0x04, 0x0C, 0x0C, 0x0C, 0x09, 0x09, 0x1F, 0x3E, 0x3F, 0x7C,
0xF8, 0xF8, 0xF8, 0xE0, 0xF0, 0x78, 0x38, 0x1C, 0xCC, 0xEE, 0x7F, 0x3F, 0x1F, 0x0F, 0x03, 0x03,
0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x01, 0x03, 0x03, 0x04, 0xF8, 0xC0, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0xFF, 0x3F, 0x07, 0xC1, 0xE0, 0xF0, 0xF8,
0xF8, 0xF8, 0xFC, 0xFC, 0xFC, 0xFC, 0xF8, 0xF8, 0xF8, 0xF0, 0xE0, 0xC0, 0x00, 0x00, 0x80, 0xC0,
0xE0, 0xF0, 0xF0, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF8, 0xF0, 0xF0, 0xE0, 0xC0, 0x83, 0x0F,
0x7F, 0xDF, 0xEF, 0xFF, 0xF7, 0x19, 0x04, 0x03, 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xE0, 0xF0, 0xF8, 0xFC, 0xFC, 0xFC, 0xFE, 0xFE, 0xFE,

[illegible]

0xFF, 0x3F, 0x3F, 0x1F, 0x1F, 0x67, 0xF8, 0xFF, 0xFF, 0xFF, 0xFF, 0x3F, 0x1F, 0x07, 0x01,
0x01, 0x01, 0x01, 0x0F, 0x1F, 0x3F, 0x3F, 0x3F, 0x3F, 0x1F, 0x0F, 0x0F, 0x0F, 0x0F, 0x0F,
0x0F, 0x0F, 0x0F, 0x07, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x03, 0x07, 0x1F, 0xFF, 0x87, 0x03,
0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x01, 0x03,
0x0C, 0x18, 0x70, 0xE0, 0x80, 0x00, 0x00, 0x00, 0x01, 0x03, 0x06, 0x98, 0xA1, 0xEF, 0xFF, 0x1C,
0x7E, 0x2F, 0x37, 0x03, 0x03, 0x01, 0x01, 0x01, 0x01, 0x10, 0x18, 0x0C, 0x0C, 0x07, 0x83, 0xC1,
0xF0, 0x1F, 0x07, 0x00, 0x00, 0x00, 0x00, 0x80, 0xE3, 0x7F, 0x11, 0x01, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF,
0xFF, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xBF, 0x81, 0x80, 0x81, 0x81, 0x83,
0x83, 0x83, 0x83, 0x81, 0x81, 0x80, 0x80, 0x80, 0x80, 0x81, 0x83, 0x87, 0x87, 0x87, 0x87, 0x83,
0x81, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0xBF, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x81, 0xB1, 0xBF, 0x81, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xBF, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x83, 0x86, 0x8C, 0xB8, 0xA0, 0xB8, 0xBE, 0xA7, 0xB3, 0xA0, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xA0, 0xB0, 0x88, 0x87, 0xB0, 0x9E,
0x83, 0x80, 0x80, 0x80, 0xB0, 0x98, 0x87, 0x83, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80,
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xFF, } ;



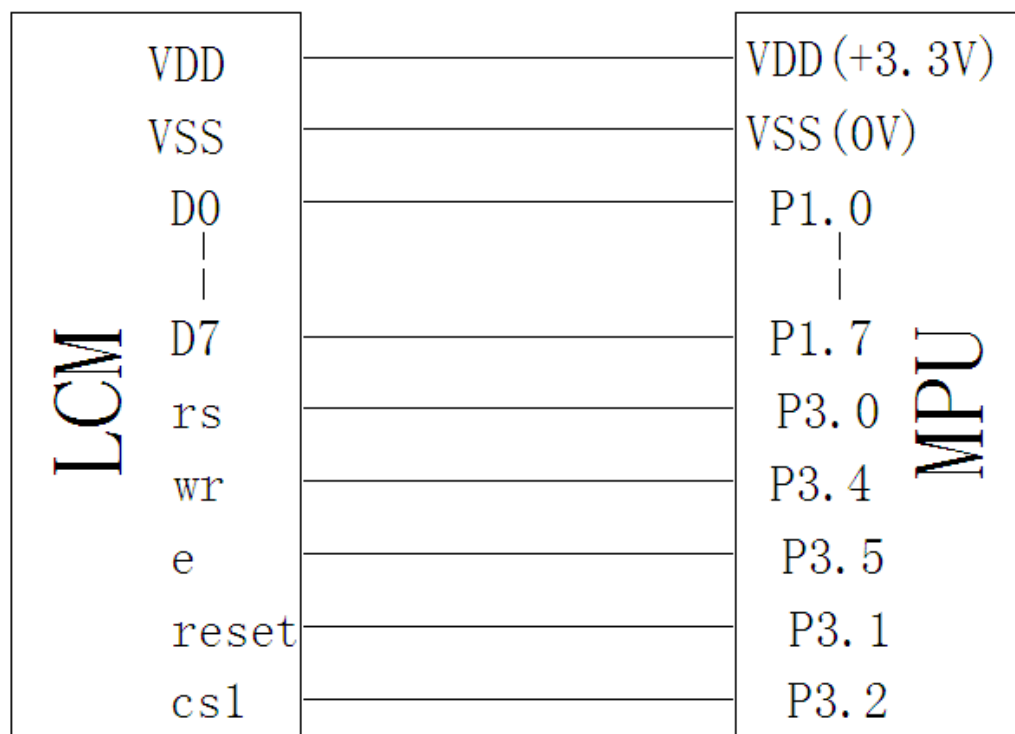


图 9. 并行接口

并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;
}
```

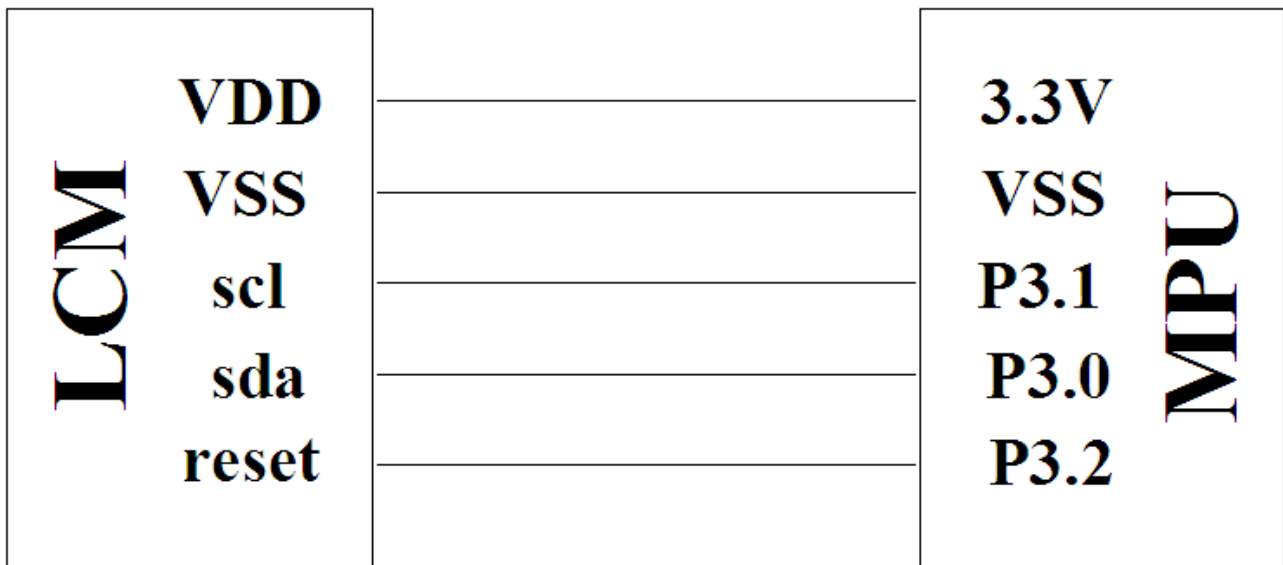


图 9. IIC 接口

IIC 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-26018, 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 0xc2
    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 picture 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, chengl);           //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), tail);        //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, "[ABCDEFGH IJKLMNOPQRSTUVWXYZabcd]") ;
display_string_5x8_1(7, 1, "(abcdefghijklmnopqrstu vwxyzabcd)") ;
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();
}
}

```