JLX256128G-921-PL

带字库 IC 的编程说明书

目 录

1．概述

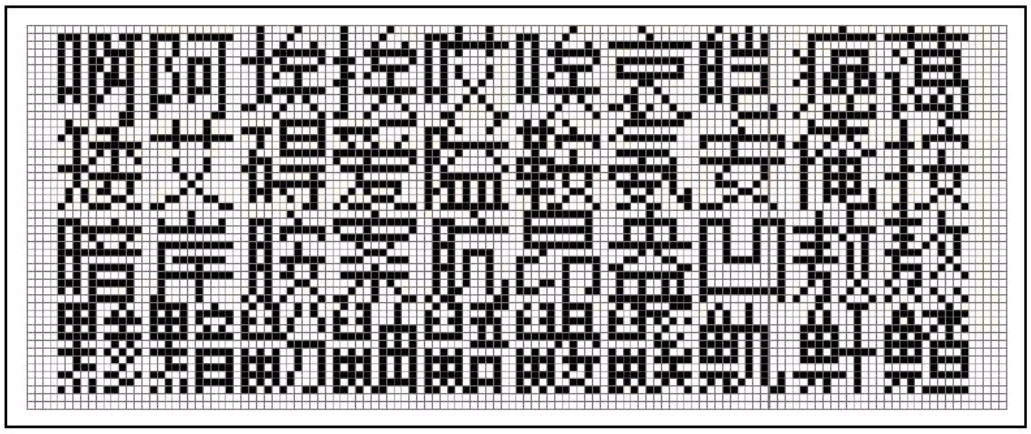
JLX256128G-921-PL 型液晶显示模块既可以当成普通的图像型液晶显示模块使用（即显示普通图像型的单色图片功能），又含有 JLX-GB2312 字库 IC，可以从字库 IC 中读出内置的字库的点阵数据写入到 LCD 驱动 IC 中，以达到显示汉字的目的。

此字库 IC 存储内容如下表所述： 表 1

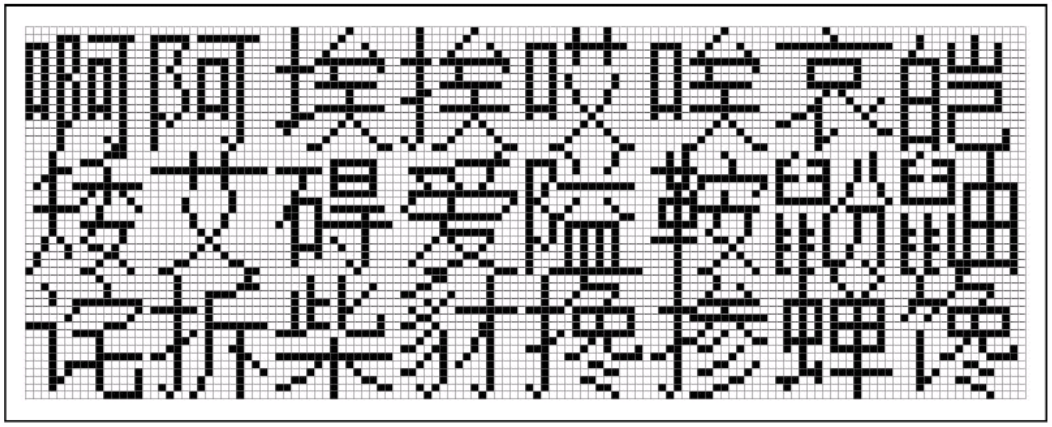
字符集	字库	字号	字符数	字体	排列方式
ASCII 字符集	ASCII	5x7	96	标准	Y-竖置横排
	ASCII	7x8	96	标准	Y-竖置横排
	ASCII	6x12	96	标准	Y-竖置横排
	ASCII	8x16	96	标准	Y-竖置横排
	ASCII	12x24	96	标准	Y-竖置横排
	ASCII	16x32	96	标准	Y-竖置横排
	ASCII	12 点阵不等宽	96	Arial（方头）	Y-竖置横排
	ASCII	12 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排
	ASCII	16 点阵不等宽	96	Arial（方头）	Y-竖置横排
	ASCII	16 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排
	ASCII	24 点阵不等宽	96	Arial（方头）	Y-竖置横排
	ASCII	24 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排
	ASCII	32 点阵不等宽	96	Arial（方头）	Y-竖置横排
	ASCII	32 点阵不等宽	96	Times new Roman（白正）	Y-竖置横排
汉字 字符集	GB2312 汉字	12x12	6763	宋体	Y-竖置横排
		16x16	6763	宋体	Y-竖置横排
		24x24	6763	宋体	Y-竖置横排
		32x32	6763	宋体	Y-竖置横排
	GB2312 字符	12x12	846	宋体	Y-竖置横排
		16x16	846	宋体	Y-竖置横排
		24x24	846	宋体	Y-竖置横排
		32x32	846	宋体	Y-竖置横排
	国标扩展字符	6x12	126	宋体	Y-竖置横排
		8x16	126	宋体	Y-竖置横排
		12x24	126	宋体	Y-竖置横排
		16x32	126	宋体	Y-竖置横排

2．字型样张：

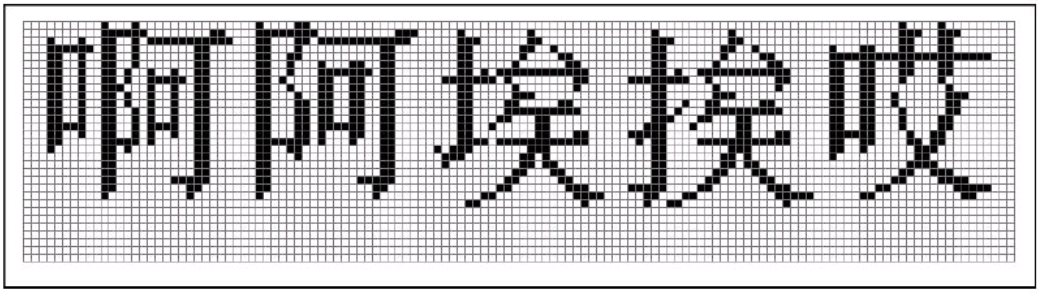
12x12 点阵 GB2312 汉字



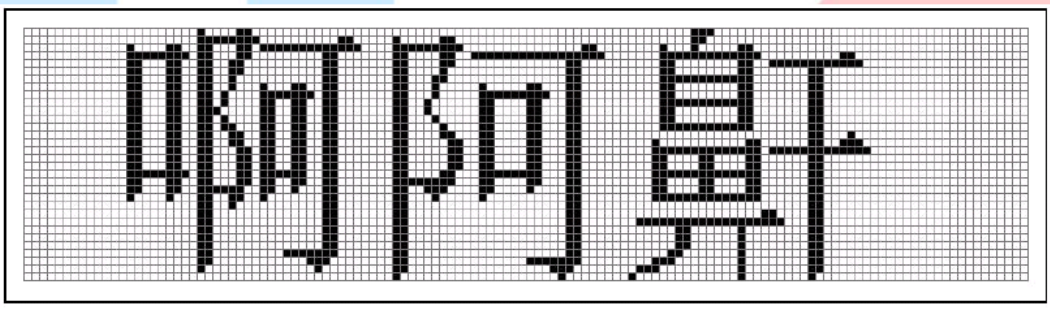
16x16 点阵 GB2312 汉字



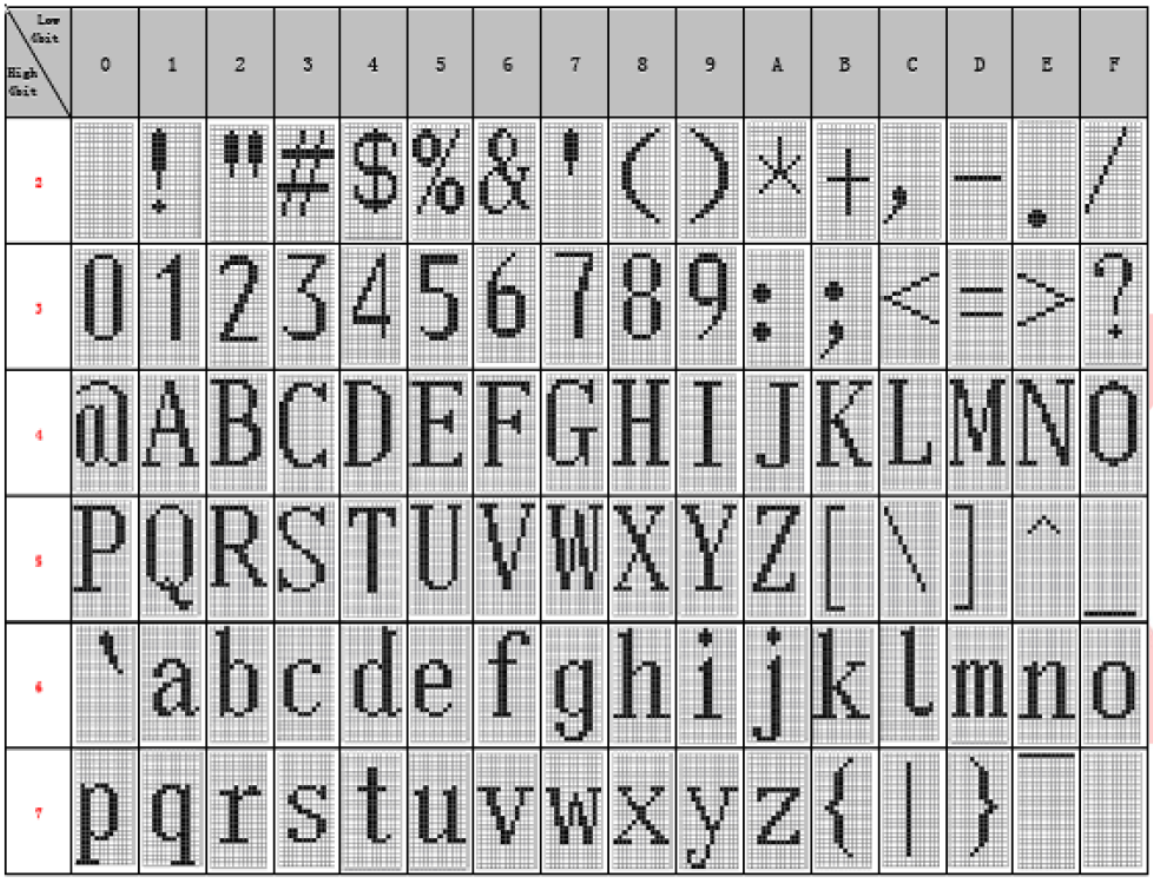
24x24 点阵 GB2312 汉字



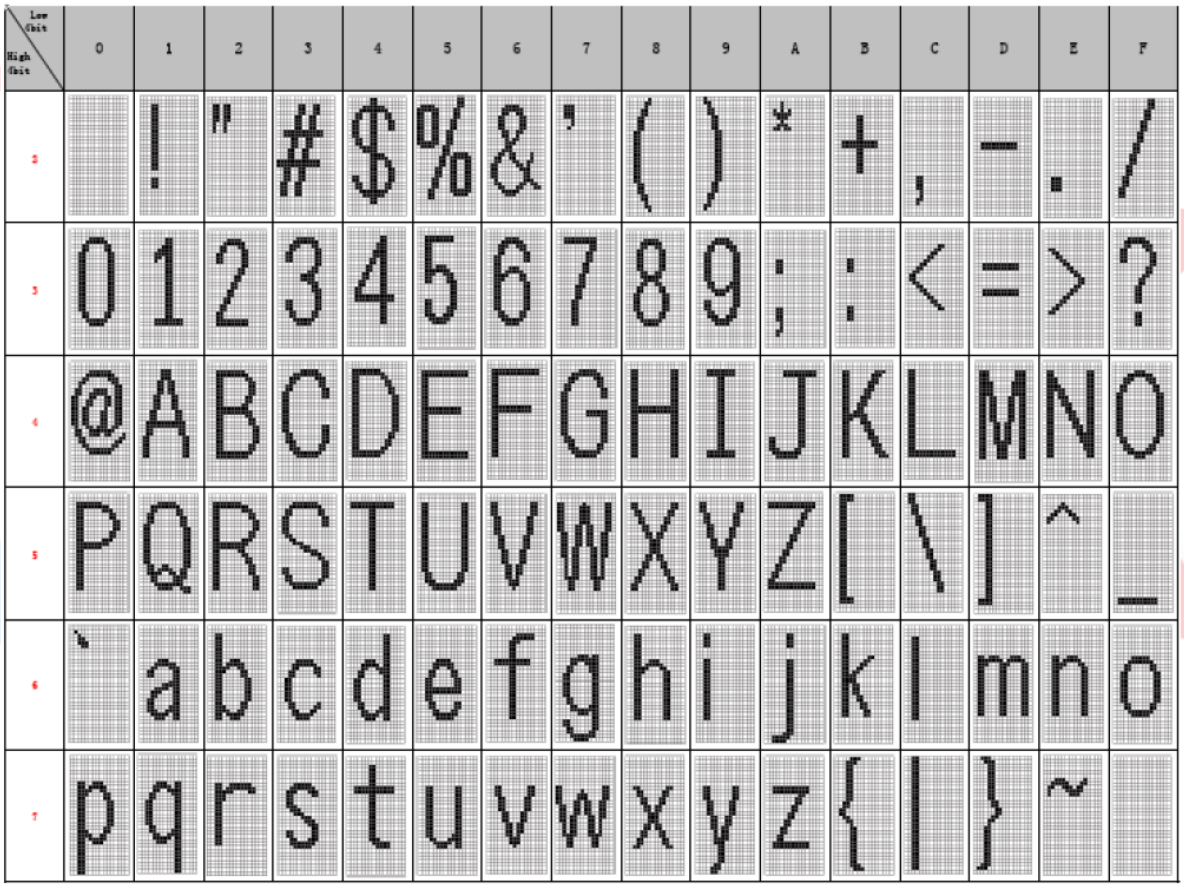
32x32 点阵 GB2312 汉字



12x24 点阵 ASCII 标准字符



16x32 点阵 ASCII 标准字符



3. 外形尺寸及接口引脚功能

3.1 外形图：

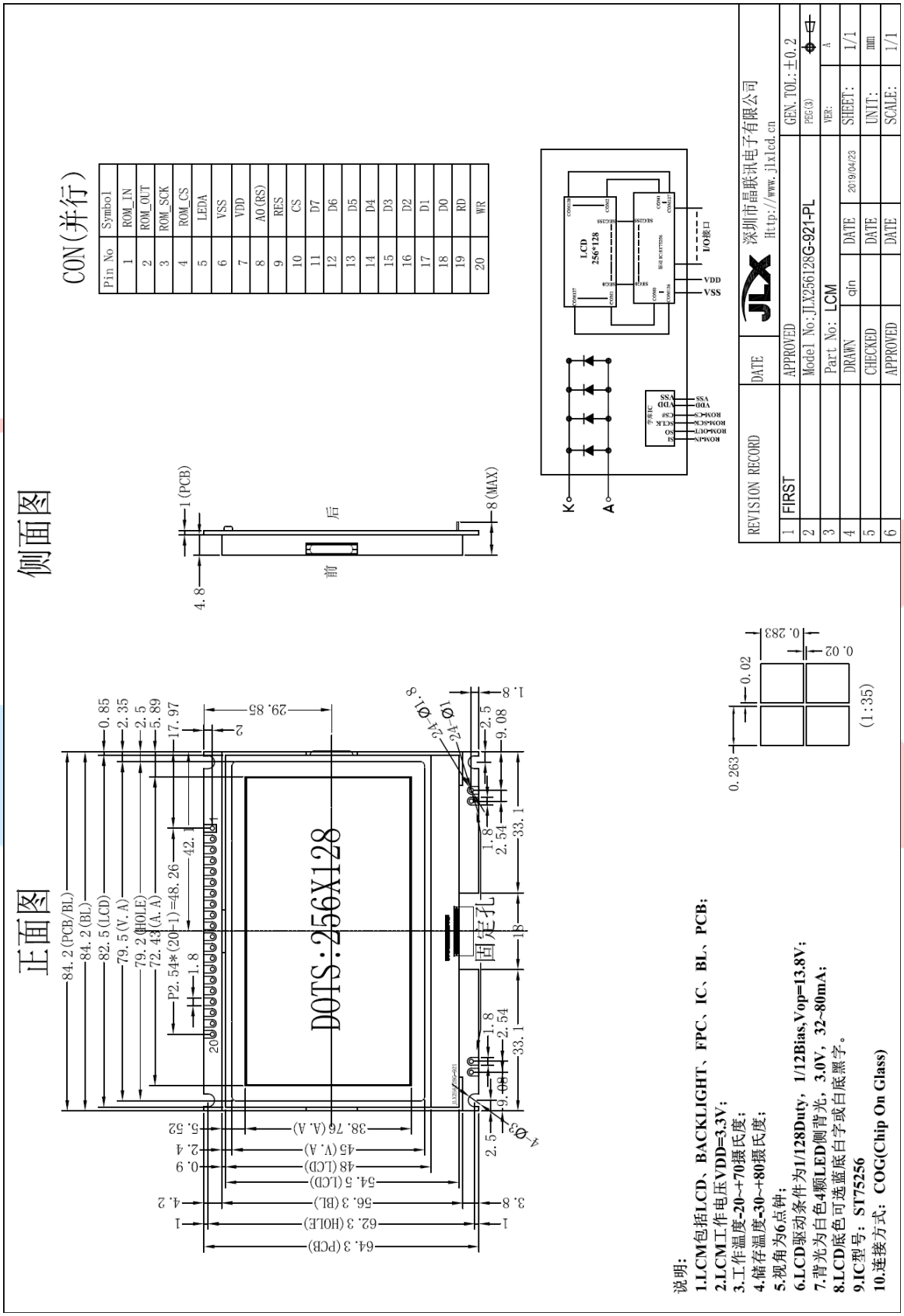


图 1. 外形尺寸

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	ROM-IN	字库 IC 接口 SI	串行数据输入	详见字库 IC:JLX-GB2312 说明书： ROM-IN 对应字库 IC 接口 SI，ROM-OUT 对应 SO， ROM-SCK 对应 SCLK， ROM-CS 对应 CS#
2	ROM-OUT	字库 IC 接口 SO	串行数据输出	
3	ROM-SCK	字库 IC 接口 SCLK	串行时钟输入	
4	ROM-CS	字库 IC 接口 CS#	片选输入	
5	LEDA	背光电源	背光电源正极，同 VDD 电压（5V 或 3.3V）	
6	VSS	接地	0V	
7	VDD	电路电源	5V,或 3.3V 可选	
8	RS(A0)	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”A0”）	
9	RES	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 串行时接口引脚功能

引 线 号	符 号	名 称	功 能	
1	ROM-IN	字库 IC 接口 SI	串行数据输入	详见字库 IC:JLX-GB2312 说明书： ROM-IN 对应字库 IC 接口 SI，ROM-OUT 对应 SO， ROM-SCK 对应 SCLK， ROM-CS 对应 CS#
2	ROM-OUT	字库 IC 接口 SO	串行数据输出	
3	ROM-SCK	字库 IC 接口 SCLK	串行时钟输入	
4	ROM-CS	字库 IC 接口 CS#	片选输入	
5	LEDA	背光电源	背光电源正极，同 VDD 电压（5V 或 3.3V）	
6	VSS	接地	0V	
7	VDD	电路电源	5V,或 3.3V 可选	

8	RS(A0)	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”A0”）
9	RES	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，此引脚接 VDD
12	D6	I/O	串行接口，此引脚接 VDD
13	D5	I/O	串行接口，此引脚接 VDD
14	D4	I/O	串行接口，此引脚接 VDD
15-17	D3-D1(SDA)	I/O	串行数据
18	D0(SCL)	I/O	串行时钟
19	RD(E)	使能信号	串行接口，此引脚接 VDD
20	WR	读/写	串行接口，此引脚接 VDD

表 2：模块串行接口引脚功能

3.2.3I2C 时接口引脚功能

引 线 号	符 号	名 称	功 能	
1	ROM-IN	字库 IC 接口 SI	串行数据输入	详见字库 IC:JLX-GB2312 说明书： ROM-IN 对应字库 IC 接口 SI，ROM-OUT 对应 SO， ROM-SCK 对应 SCLK， ROM-CS 对应 CS#
2	ROM-OUT	字库 IC 接口 SO	串行数据输出	
3	ROM-SCK	字 库 IC 接 口 SCLK	串行时钟输入	
4	ROM-CS	字库 IC 接口 CS#	片选输入	
5	LEDA	背光电源	背光电源正极，同 VDD 电压（5V 或 3.3V）	
6	VSS	接地	0V	
7	VDD	电路电源	3.3V	
8	RS(A0)	寄存器选择信号	I2C 接口，此引脚接 VDD	
9	RES	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 VSS	
11	D7	I/O	I²C 接口，引脚接从属地址 VSS	
12	D6	I/O	I²C 接口，引脚接从属地址 VSS	
13	D5	I/O	I²C 接口，引脚接 VDD	
14	D4	I/O	I²C 接口，引脚接 VDD	
15-17	D1 ～ D3 (SDA)	I/O	串行数据(D1、D2、D3 短接一起作为 SDA)	
18	D0(SCL)	I/O	串行时钟(SCL)	
19	E(RD)	使能信号	I2C 接口，此引脚接 VDD	
20	RW(WR)	读、写	I2C 接口，此引脚接 VDD	

表 2：模块 I2C 接口引脚功能

4. 工作电路框图：

见图 2，模块由 LCD 驱动 IC ST75256、字库 IC、背光组成。

电路框图

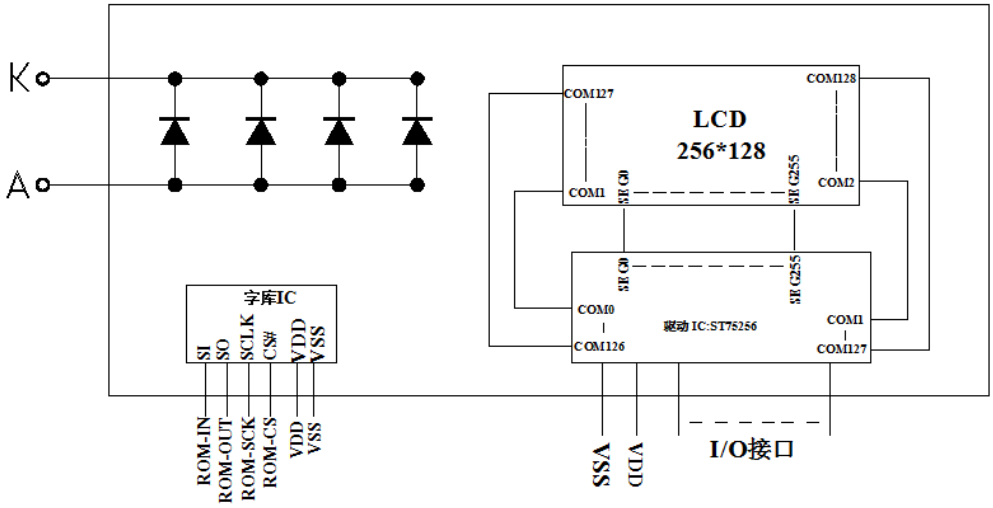


图 2:JLX256128G-921-PL 电路框图

5. 指令：

5.1 字库 IC（JLX-GB2312）指令表

Instruction	Description	Instruction Code(One-Byte)		Address Bytes	Dummy Bytes	Data Bytes
READ	Read Data Bytes	0000 0011	03 h	3	—	1 to ∞
FAST_READ	Read Data Bytes at Higher Speed	0000 1011	0B h	3	1	1 to ∞

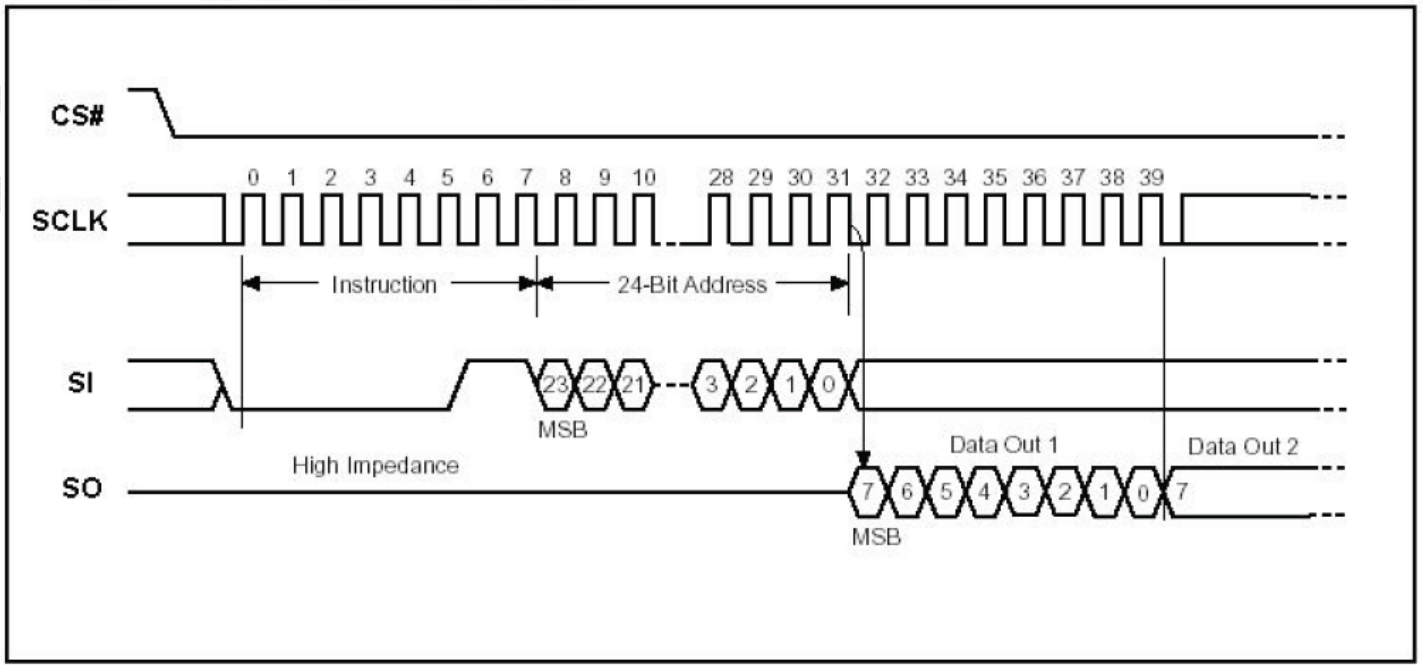
所有对本芯片 SPI 接口的操作只有 2 个，那就是 Read Data Bytes (READ “一般读取”)和 Read Data Bytes at Higher Speed (FAST_READ “快速读取点阵数据”)。.

Read Data Bytes（一般读取）：

Read Data Bytes 需要用指令码来执行每一次操作。READ 指令的时序如下(图)：

- 首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（03 h）和 3 个字节的地址和通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。
- 然后该地址的字节数据通过串行数据输出引脚（SO）移位输出，每一位在串行时钟（SCLK）下降沿被移出。
- 读取字节数据后，则把片选信号（CS#）变为高，结束本次操作。
  如果片选信号（CS#）继续保持为底，则下一个地址的字节数据继续通过串行数据输出引脚（SO）移位输出。

图：Read Data Bytes (READ) Instruction Sequence and Data-out sequence:

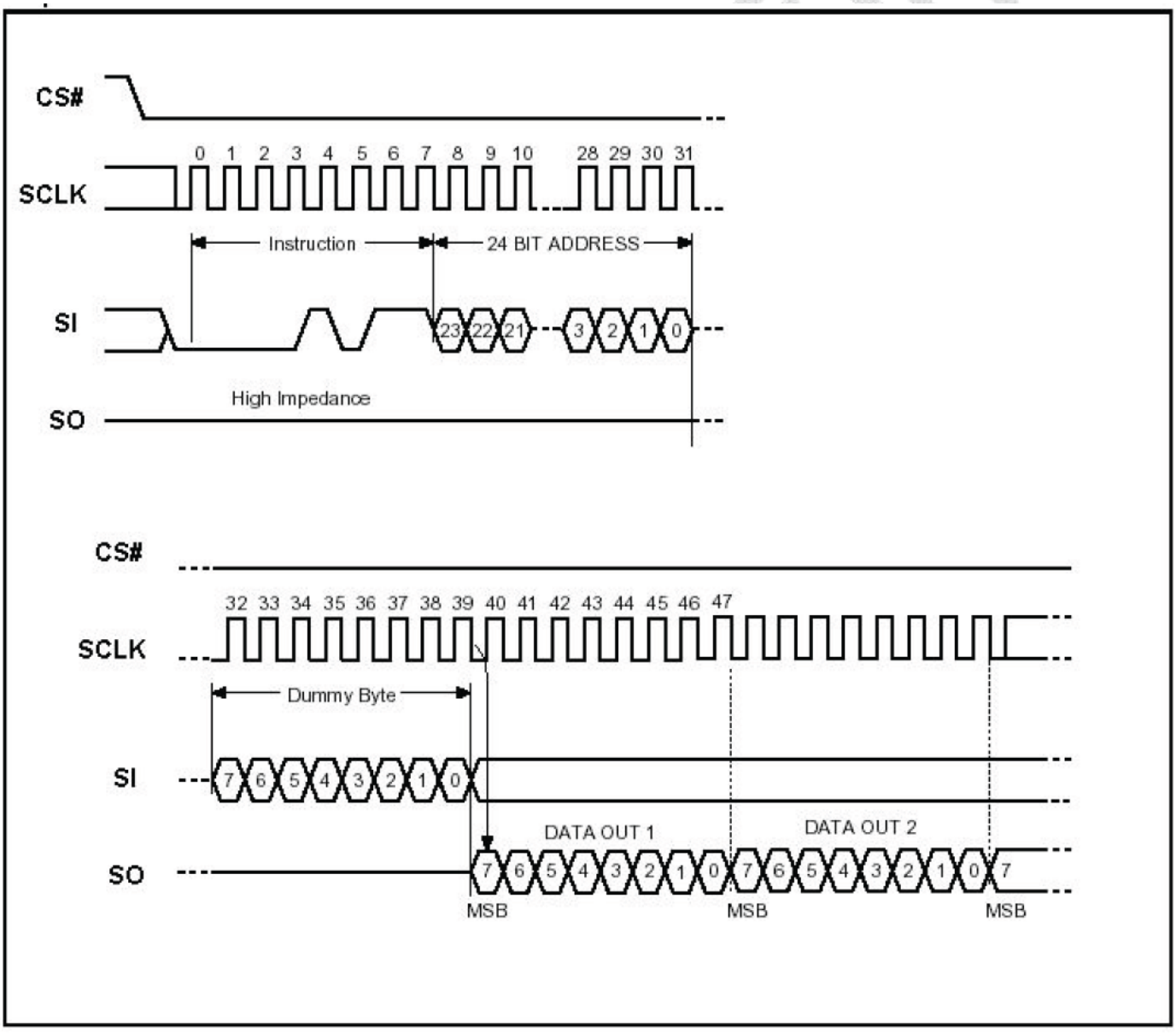


Read Data Bytes at Higher speed（快速读取）：

Read Data Bytes at Higher Speed 需要用指令码来执行操作。READ_FAST 指令的时序如下(图)：

- 首先把片选信号（CS#）变为低，紧跟着的是 1 个字节的命令字（0B h）和 3 个字节的地址以及一个字节 Dummy Byte 通过串行数据输入引脚（SI）移位输入，每一位在串行时钟（SCLK）上升沿被锁存。
- 然后该地址的字节数据通过串行数据输出引脚（SO）移位输出，每一位在串行时钟（SCLK）下降沿被移出。
- 如果片选信号（CS#）继续保持为底，则下一个地址的字节数据继续通过串行数据输出引脚（SO）移位输出。例：读取一个 15x16 点阵汉字需要 32Byte，则连续 32 个字节读取后结束一个汉字的点阵数据读取操作。
  如果不需要继续读取数据，则把片选信号（CS#）变为高，结束本次操作。

图：Read Data Bytes at Higher Speed (READ_FAST) Instruction Sequence and Data-out sequence:



5.2 LCD 驱动 IC 指令表

指令名称	指 令 码										
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	
(1)扩展指令1	0	0	0	0	1	1	EXT1	0	0	EXT0	扩展指令 1、2、3、4 0X30:扩展指令 1
Ext[1:0]=0,0(Extension Command1/扩展指令 1) 0X30 扩屏指令 1 一定要调用 0X30 才能用扩展指令 1											
(2)显示开/关 （display on/off）	0	0	1	0	1	0	1	1	1	0 1	显示开/关: 0XAE:关，0XAF：开
(3)正显/反显 （Inverse Display）	0	0	1	0	1	0	0	1	1	0 1	显示正显/反显 0XA6:正显，正常 0XA7：反显
(4)所有点阵开/关 (All Pixel ON/OFF)	0	0	0	0	1	0	0	0	1	0 1	0X22：所有点阵关 0X23：所有点阵开
(5) 控制液晶屏显示 (Display Control)	0	0	1	1	0	0	1	0	1	0	0XCA:显示控制
	1	0	0	0	0	0	0	CLD	0	0	0X00:设置 CL 驱动频率：CLD=0
	1	0	DT7	DT6	DT5	DT4	DT3	DT2	DT1	DT0	0X7F:点空比：Duty=128
	1	0	0	0	LF4	F1	LF3	LF2	LF1	LF0	0X20:帧周期
(6)省电模式 (Power save)	0	0	1	0	0	1	0	1	0	SLP	0X94：SLP=0，退出睡眠模式 0X95：SLP=1，进入睡眠模式
(7)页地址设置 (Set Page Address)	0	0	0	1	1	1	0	1	0	1	0X75：页地址设置
	1	0	YS7	YS6	YS5	YS4	YS3	YS2	YS1	YS0	0X00：起始页地址
	1	0	YE7	YE6	YE5	YE4	YE3	YE2	YE2	YE0	0X1F：结束页地址,每 4 行为 1 页
(8)列地址设置 (Set Column Address)	0	0	0	0	0	1	0	1	0	1	0X15：列地址设置
	1	0	XS7	XS6	XS5	XS4	XS3	XS2	XS1	XS0	0X00：起始列地址
	1	0	XE7	XE6	XE5	XE4	XE3	XE2	XE1	XE0	0XFF：结束列地址 XE=256
(9)行列扫描方向 (Data Scan Direction)	0	0	1	0	1	1	1	1	0	0	0XBC：行列扫描方向
	1	0	0	0	0	0	0	MV	MX	MY	0X00：MX、MY=Normal
(10)写数据到晶液屏 (Write Data)	0	0	0	1	0	1	1	1	0	0	0X5C：写数据
	1	0	D7	D6	D5	D4	D3	D2	D1	D0	8 位显示数据
(11)读液晶屏显示数据 (Read Data)	0	0	0	1	0	1	1	1	0	1	0X5D：读数据
	1	1	D7	D6	D5	D4	D3	D2	D1	D0	8 位显示数据
(12)指定区域显示数据 (Partial In)	0	0	1	0	1	0	1	0	0	0	0XA8：指定显示区域
	1	0	PTS7	PTS6	PTS5	PTS4	PTS3	PTS2	PTS1	PTS0	起始区域地址：00h≤PTS≥A1h
	1	0	PTE7	PTE6	PTE5	PTE4	PTE3	PTE2	PTE1	PTE0	结束区域地址：00h≤PTE≥A1h
(13) 退出指定区域显示 (Partial Out)	0	0	1	0	1	0	1	0	0	1	0XA9：退出指定区域显示
(14)读/改/写	0	0	1	1	1	0	0	0	0	0	0XE0：进入读/改/写
(15)退出读/改/写	0	0	1	1	1	0	1	1	1	0	0XEE：退出读/改/写
(16)指定显示滚动区域 (Scroll Area)	0	0	1	0	1	0	1	0	1	0	0XAA：滚动区域设置
	1	0	TL7	TL6	TL5	TL4	TL3	TL2	TL1	TL0	TL[7:0]:起始区域地址
	1	0	BL7	BL6	BL5	BL4	BL3	BL2	BL1	BL0	BL[7:0]:结束区域地址
	1	0	NSL7	NLS6	NSL5	NSL4	NSL3	NSL2	NSL1	NSL0	NSL[7:0]:指定行数
	1	0	0	0	0	0	0	0	SCM1	SCM0	SCM[1:0]:显示模式
(17)显示初始行设置	0	0	1	0	1	0	1	0	1	1	0XAB：滚动开始初始行设置

(Set Start Line)	1	0	SL7	SL6	SL5	SL4	SL3	SL2	SL1	SL0	00h≤SL≥A1h
(18)开振荡电路	0	0	1	1	0	1	0	0	0	1	0XD1：开内部振荡电路
(19)关振荡电路	0	0	1	1	0	1	0	0	1	0	0XD2：关内部振荡电路
(20)电源控制	0	0	0	0	1	0	0	0	0	0	0X20：电源控制
(Power Control)	1	0	0	0	0	0	VB	0	VF	VR	0X0B：VB、VF、VR=1
(21)液晶内部电压设置	0	0	1	0	0	0	0	0	0	1	0X81:设置对比度
(Set Vop)	1	0	0	0	Vop5	Vop4	Vop3	Vop2	Vop1	Vop0	0X26:微调对比度,范围 0X00-0XFF
	1	0	0	0	0	0	0	Vop7	Vop6	Vop5	0X04:粗调对比度,范围 0X00-0X07
											先微调再粗调，顺序不能变
(22)液晶内部电压控制	0	0	1	1	0	1	0	1	1	VOL	0XD6:VOP 每格增加 0.04V
(Vop Control)											0XD7:VOP 每格减少 0.04V
(23)读寄存器模式	0	0	0	1	1	1	1	1	0	REG	0X7C：读寄存器值 Vop[5:0]
											0X7D：读寄存器值 Vop[8:6]
(24)空操作	0	0	0	0	1	0	0	1	0	1	0X25：空操作
(25)读状态 (并行、IIC)	0	1	D7	D6	D5	D4	D3	D2	D1	D0	读状态字节
(26)读状态（串行接口）	0	0	1	1	1	1	1	1	1	0	读状态字节
	0	1	D7	D6	D5	D4	D3	D2	D1	D0	
(27)数据格式选择	0	0	0	0	0	0	1	D0	0	0	0X80：数据 D7→D0
(Data Format Select)											0XC0：数据 D0→D7
(28)显示模式	0	0	1	1	1	1	0	0	0	0	0XF0：显示模式设置
(Display Mode)	1	0	0	0	0	1	0	0	0	DM	0X10：黑白模式
											0X11：4 灰级度模式
(29)ICON设置	0	0	0	1	1	1	0	1	1	ICON	0X77：使能 ICON RAM
											0X76：禁用 ICON RAM
(30)设置主/从模式	0	0	0	1	1	0	1	1	1	MS	0X6E：主模式(使用主模式)
											0X6F：从模式
Ext[1:0]=0,1(Extension Command 2)						0X31 扩屏指令 2 一定要调用 0X31 才能用扩展指令 2					
(31)灰度设置	0	0	0	0	1	0	0	0	0	0	0X20:灰度级设置
Set Gray Level	1	0	0	0	0	0	0	0	0	0	GL[4:0]: 浅灰度级设置
	1	0	0	0	0	0	0	0	0	0	GD[4:0]: 深灰度级设置
	1	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	GL4	GL3	GL2	GL1	GL0	
	1	0	0	0	0	GL4	GL3	GL2	GL1	GL0	
	1	0	0	0	0	GL4	GL3	GL2	GL1	GL0	
	1	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	GD4	GD3	GD2	GD1	GD0	
	1	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	GD4	GD3	GD2	GD1	GD0	
	1	0	0	0	0	GD4	GD3	GD2	GD1	GD0	
	1	0	0	0	0	GD4	GD3	GD2	GD1	GD0	
	1	0	0	0	0	0	0	0	0	0	
	1	0	0	0	0	0	0	0	0	0	
(32)LCD偏压比设置	0	0	0	0	1	1	0	0	1	0	0X32：偏压比设置
	1	0	0	0	0	0	0	0	0	0	

	1	0	0	0	0	0	0	0	BE1	BE0	0X01：升压电容频率
	1	0	0	0	0	0	0	BS2	BS1	BS0	0X02：偏压比，BIAS=1/12
(33)升压倍数	0	0	0	1	0	1	0	0	0	1	0X51:内建升压倍数设置
(Booster Level)	1	0	0	1	1	1	1	0	1	BST	0X7B:10 倍
(34)电压驱动选择	0	0	0	1	0	0	0	0	0	DS	0X41：LCD 内部升压
(35)自动读取控制	0	0	1	1	0	1	0	1	1	1	XARD=0：使能自动读
	1	0	1	0	0	XARD	1	1	1	1	XARD=0：不使能自动读
(36)控制OTP读写	0	0	1	1	1	0	0	0	0	0	0xe0：OTP 读写
	1	0	0	0	ER/RD	0	0	0	0	0	WR/RD=0；0x00,使能 OTP 读 ER/RD=1；0x20,使能 OTP 写
(37)控制OTP出	0	0	1	1	1	0	0	0	0	1	控制 OTP 出
(38)写OTP	0	0	1	1	1	0	0	0	1	0	写 OTP
(39)读OTP	0	0	1	1	1	0	0	0	1	1	读 OTP
(40)OTP选择控制	0	0	1	1	1	0	0	1	0	0	0xe4：OTP 选择控制
	1	0	1	Ctrl	0	0	1	0	0	1	Ctrl=1：0xc9，不使能 OTP Ctrl=0：0x89，使能 OTP
(41)OTP程序设置	0	0	1	1	1	0	0	1	0	1	OTP 程序设置
	1	0	0	0	0	0	1	1	1	1	
(42)帧速率	0	0	1	1	1	1	0	0	0	0	0xf0：帧速率设置在不同的温度范围
	1	0	0	0	0	FRA4	FRA3	FRA2	FRA1	FRA0	
	1	0	0	0	0	FRB4	FRB3	FRB2	FRB1	FRB0	
	1	0	0	0	0	FRC4	FRC3	FRC2	FRC1	FRC0	
	1	0	0	0	0	FRD4	FRD3	FRD2	FRD1	FRD0	
(43)温度范围	0	0	1	1	1	1	0	0	1	0	0xf2：温度范围设置
	1	0	0	TA6	TA5	TA4	TA3	TA2	TA1	TA0	
	1	0	0	TB6	TB5	TB4	TB3	TB2	TB1	TB0	
	1	0	0	TC6	TC5	TC4	TC3	TC2	TC1	TC0	
(44)温度梯度补偿	0	0	1	1	1	1	0	1	0	0	0xf4：温度补偿系数设置
	1	0	MT13	MT12	MT11	MT10	MT03	MT02	MT01	MT00	
	1	0	MT33	MT32	MT31	MT30	MT23	MT22	MT21	MT20	
	1	0	MT53	MT52	MT51	MT50	MT43	MT42	MT41	MT40	
	1	0	MT73	MT72	MT71	MT70	MT63	MT62	MT61	MT60	
	1	0	MT93	MT92	MT91	MT90	MT83	MT82	MT81	MT80	
	1	0	MTB3	MTB2	MTB1	MTB0	MTA3	MTA2	MTA1	MTA0	
	1	0	MTD3	MTD2	MTD1	MTD0	MTC3	MTC2	MTC1	MTC0	
	1	0	MTF3	MTF2	MTF1	MTF0	MTE3	MTE2	MTE1	MTE0	
Ext[1:0]=1,0(Extension Command 3)	**0x38 扩屏指令 3 一定要调用 0X38 才能用扩展指令 3**										
(45) ID 设置	0	0	1	1	0	1	0	1	0	1	0xd5：ID 设置
	1	0	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	
(46)读 ID	0	0	0	1	1	1	1	1	1	RID	RID=1：0x7f，使能
Ext[1:0]=1,1(Extension Command 4)	**0x39 扩屏指令 4 一定要调用 0X39 才能用扩展指令 4**										
(47) 使能 OTP	0	0	1	1	0	1	0	1	1	0	0xd6：使能 OTP EOTP=1；不使能 EOTP，一般不使能 EOTP EOTP=0；使能 EOTP

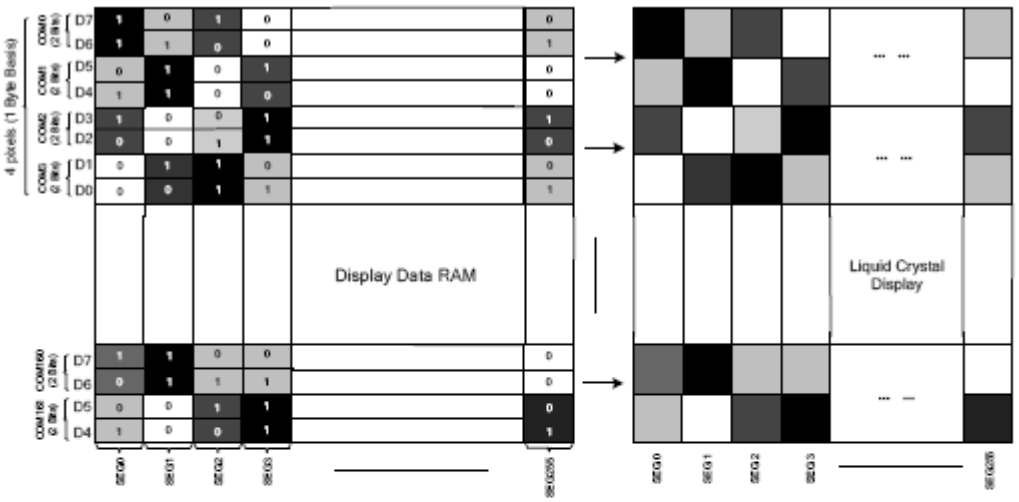
表 5. 指令表

请详细参考 IC 资料”ST75256.PDF”。

5.3 点阵与 DD RAM 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 256*128 点阵的屏分为 16 个“页”，从第 0“页”到第 15“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：



2 Bits Data N=0~3		DDRAM		LCD
D2N+1	D2N			
1	1	1	1	
0	0	0	0	
1	0	1	0	
0	1	0	1	

Figure 21 DDRAM Mapping (4-Level Gray Scale Mode)

下图摘自 ST75256 IC 资料，可通过“ST75256.PDF”之第 37 页获取最佳效果。

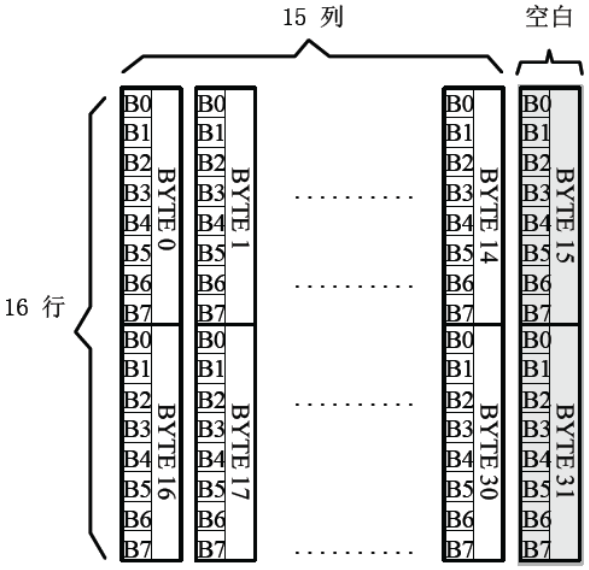
6 字库排置（竖置横排）

6.1 点阵排列格式

每个汉字在芯片中是以汉字点阵字模的形式存储的，每个点用一个二进制位表示，存 1 的点，当显示时可以在屏幕上显示亮点，存 0 的点，则在屏幕上不显示。点阵排列格式为竖置横排：即一个字节的高位表示下面的点，低位表示上面的点（如果用户按 16bit 总线宽度读取点阵数据，请注意高低字节的顺序），排满一行后再排下一行。这样把点阵信息用来直接在显示器上按上述规则显示，则将出现对应的汉字。

6.2 15X16 点汉字排列格式

15X16 点汉字的信息需要 32 个字节（BYTE 0 – BYTE 31）来表示。该 15X16 点汉字的点阵数据是竖置横排的，其具体排列结构如下图：

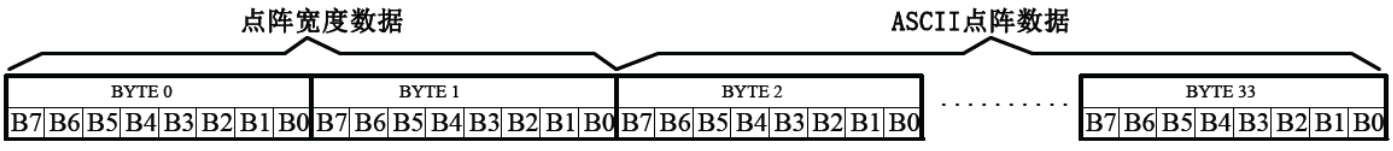


6.3 16 点阵不等宽 ASCII 方头（Arial）字符排列格式

16 点阵不等宽字符的信息需要 34 个字节（BYTE 0 – BYTE33）来表示。

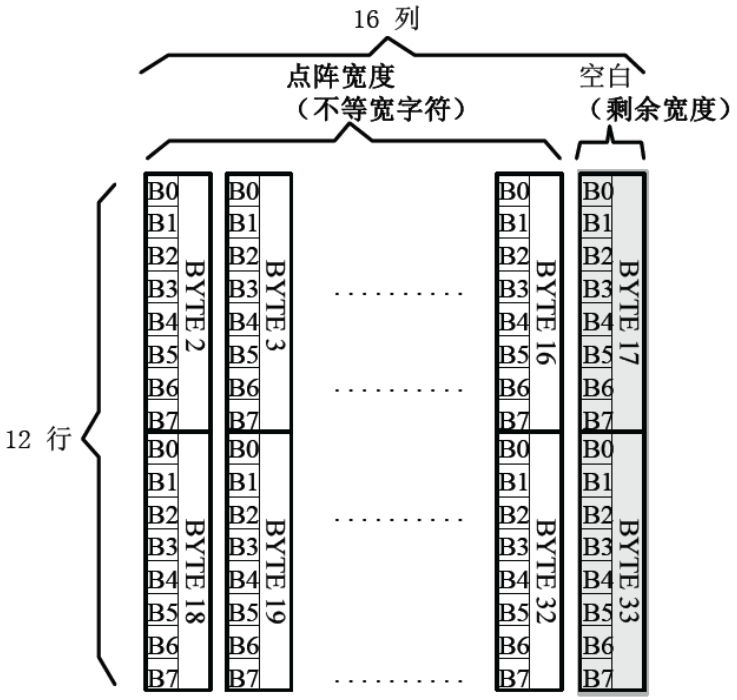
■ 存储格式

由于字符是不等宽的，因此在存储格式中 BYTE0~ BYTE1 存放点阵宽度数据，BYTE2-33 存放竖置横排点阵数据。具体格式见下图：



■ 存储结构

点阵存储宽度固定为 16，根据不同字符，其实际点阵宽度会小于 16，并会出现相应的空白区。根据 BYTE0~ BYTE1 所存放点阵的宽度数据，可以对还原下一个字的显示或排版留作参考。



例如：ASCII 方头字符 B

0-33BYTE 的点阵数据是:　00 0C 00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 63 63 63 63 67 3E 1C 00 00 00 00 00

其中：

BYTE0~ BYTE1:　　00 0C　　为 ASCII 方头字符 B 的点阵宽度数据，即：12 位宽度。字符后面有 4 位空白区，可以在排版下一个字时考虑到这一点，将下一个字的起始位置前移。（见下图）

BYTE2-33:　00 F8 F8 18 18 18 18 18 F8 F0 00 00 00 00 00 00 00 7F 7F 63 63 63 63 63 67 3E 1C 00 00 00 00 00　　为 ASCII 方头字符 B 的点阵数据。

7 点阵数据验证（客户参考用）

客户将芯片内“A”的数据调出与以下进行对比。若一致，表示 SPI 驱动正常工作；若不一致，请重新编写驱动。

排置：Y（竖置横排）点阵大小 8X16
字母"A"
点阵数据：00 E0 9C 82 9C E0 00 00 0F 00 00 00 00 00 0F 00

排置：W（横置横排）点阵大小 8X16
字母"A"
点阵数据：00 10 28 28 28 44 44 7C 82 82 82 82 00 00 00 00

8 附录

8.1 GB23121 区字符（846 字符）

GB2312 标准点阵字符 1 区对应码位的 A1A1~A9EF 共计 846 个字符；

GB2312　1 区

A1	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A			、	。	·	ˉ	ˇ	¨	〃	々	—	～	‖	…	‘	’
B	“	”	〔	〕	〈	〉	《	》	「	」	『	』	〖	〗	【	】
C	±	×	÷	∶	∧	∨	∑	∏	∪	∩	∈	∷	√	⊥	∥	∠
D	⌒	⊙	∫	∮	≡	≌	≈	∽	∝	≠	≮	≯	≤	≥	∞	∵
E	∴	♂	♀	°	′	″	℃	$	¤	￠	£	‰	§	№	☆	★
F	○	●	◎	◇	◆	□	■	△	▲	※	→	←	↑	↓	〓	

A2	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ⅰ	ⅱ	ⅲ	ⅳ	ⅴ	ⅵ	ⅶ	ⅷ	ⅸ	ⅹ					
B		1.	2.	3.	4.	5.	6.	7.	8.	9.	10.	11.	12.	13.	14.	15.
C	16.	17.	18.	19.	20.	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
D	(12)	(13)	(14)	(15)	(16)	(17)	(18)	(19)	(20)	①	②	③	④	⑤	⑥	⑦
E	⑧	⑨	⑩	€		㈠	㈡	㈢	㈣	㈤	㈥	㈦	㈧	㈨	㈩	
F		Ⅰ	Ⅱ	Ⅲ	Ⅳ	Ⅴ	Ⅵ	Ⅶ	Ⅷ	Ⅸ	Ⅹ	Ⅺ	Ⅻ			

A3	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		！	＂	＃	￥	％	＆	＇	（	）	＊	＋	，	－	．	／
B	０	１	２	３	４	５	６	７	８	９	：	；	＜	＝	＞	？
C	＠	Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ	Ｇ	Ｈ	Ｉ	Ｊ	Ｋ	Ｌ	Ｍ	Ｎ	Ｏ
D	Ｐ	Ｑ	Ｒ	Ｓ	Ｔ	Ｕ	Ｖ	Ｗ	Ｘ	Ｙ	Ｚ	［	＼	］	＾	＿
E	｀	ａ	ｂ	ｃ	ｄ	ｅ	ｆ	ｇ	ｈ	ｉ	ｊ	ｋ	ｌ	ｍ	ｎ	ｏ
F	ｐ	ｑ	ｒ	ｓ	ｔ	ｕ	ｖ	ｗ	ｘ	ｙ	ｚ	｛	｜	｝	￣	

GB2312　1区

A4	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		あ	あ	い	い	う	う	え	え	お	お	か	が	き	ぎ	く
B	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
C	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
D	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
E	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
F	ゐ	ゑ	を	ん												

A5	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
B	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
C	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
D	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
E	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
F	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									

A6	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
B	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
C		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
D	π	ρ	σ	τ	υ	φ	χ	ψ	ω	'	°	`	:	;	!	?
E	ˆ	ˇ	ˉ	ˇ	ˆ	ˇ	ˊ	ˋ	ˍ	˩	˨	˧	˦	˥	▁	▔
F	ˆ	ˇ	∣	⋮	∣	⁞										

GB2312 1区

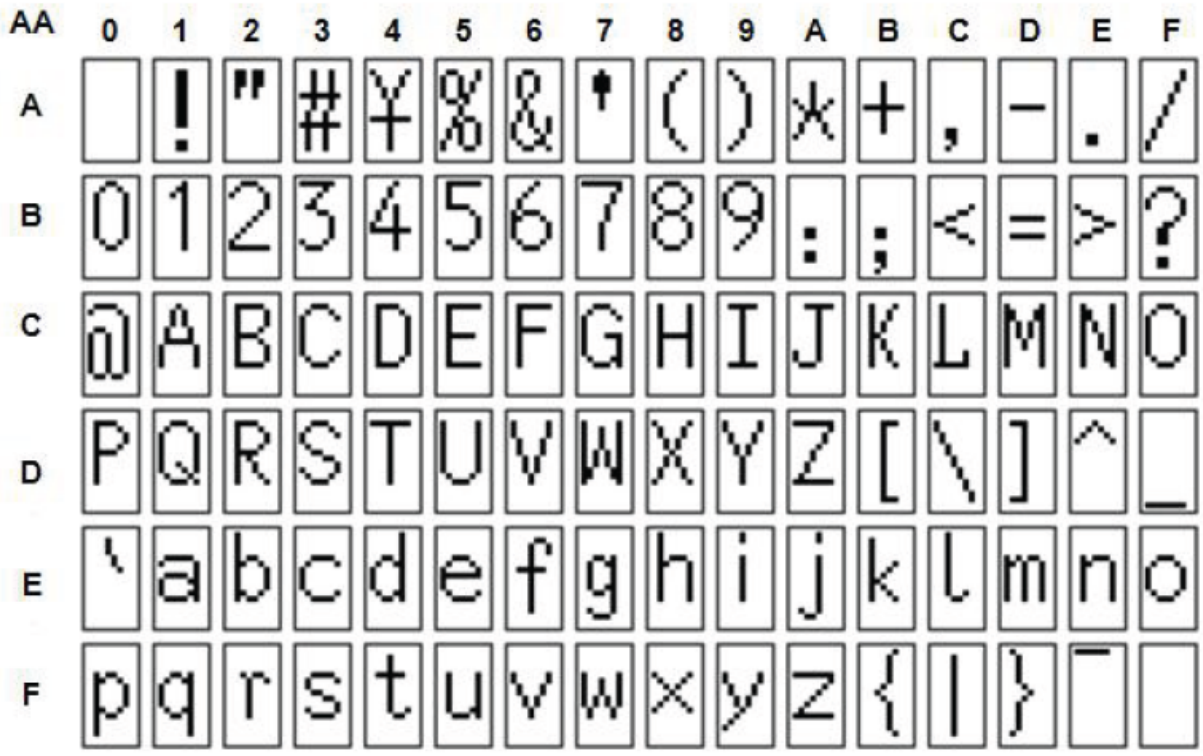
A7	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
B	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
C	Ю	Я														
D		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
E	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
F	ю	я														

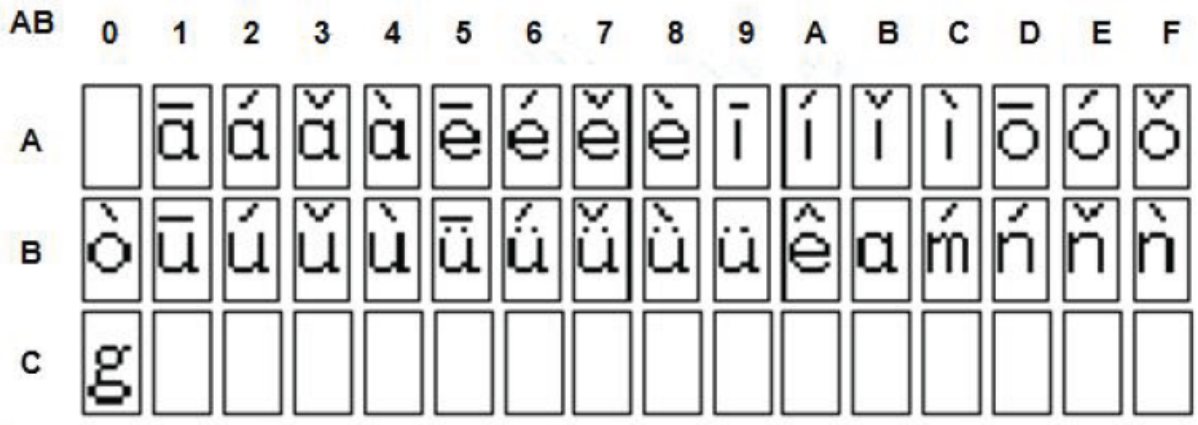
A8	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A		ā	á	ǎ	à	ē	é	ě	è	ī	í	ǐ	ì	ō	ó	ǒ
B	ò	ū	ú	ǔ	ù	ǖ	ǘ	ǚ	ǜ	ü	ê	ɑ	ḿ	ń	ň	ǹ
C	ɡ					ㄅ	ㄆ	ㄇ	ㄈ	ㄉ	ㄊ	ㄋ	ㄌ	ㄍ	ㄎ	ㄏ
D	ㄐ	ㄑ	ㄒ	ㄓ	ㄔ	ㄕ	ㄖ	ㄗ	ㄘ	ㄙ	ㄚ	ㄛ	ㄜ	ㄝ	ㄞ	ㄟ
E	ㄠ	ㄡ	ㄢ	ㄣ	ㄤ	ㄥ	ㄦ	ㄧ	ㄨ	ㄩ						
F																

A9	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A					—	━	∣	┃	┄	┅	┆	┇	┈	┉	┊	┋
B	┌	┍	┎	┏	┐	┑	┒	┓	└	┕	┖	┗	┘	┙	┚	┛
C	├	┝	┞	┟	┠	┡	┢	┣	┤	┥	┦	┧	┨	┩	┪	┫
D	┬	┭	┮	┯	┰	┱	┲	┳	┴	┵	┶	┷	┸	┹	┺	┻
E	┼	┽	┾	┿	╀	╁	╂	╃	╄	╅	╆	╇	╈	╉	╊	╋
F																

8.2 8x16 点国际扩展字符（126 字符）

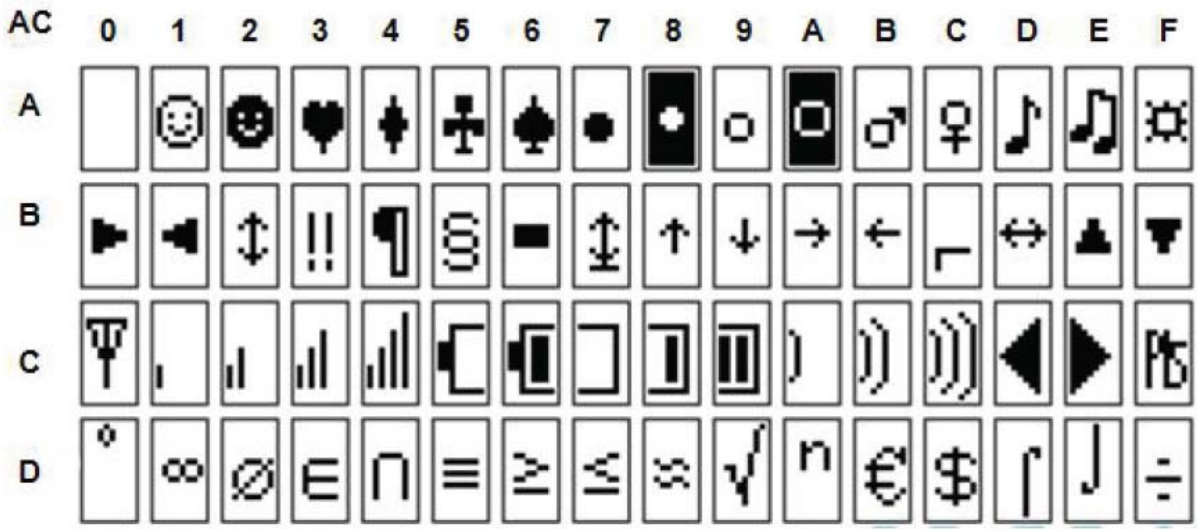
内码组成为 AAA1~ABC0 共计 126 个字符





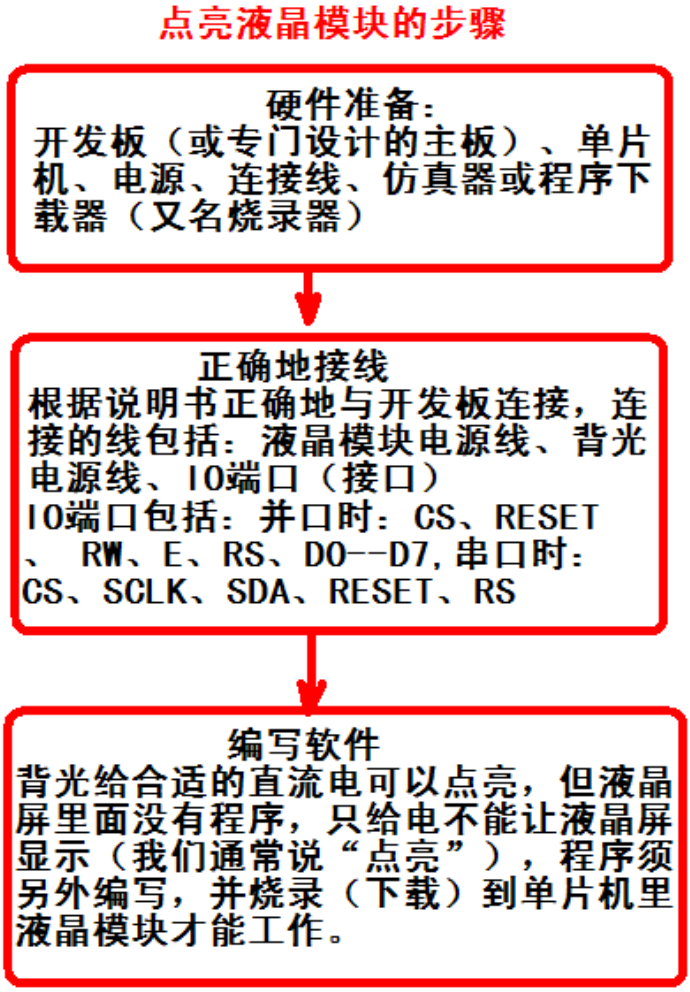
8.3 8x16 点特殊字符（64 字符）

内码组成为 ACA1~ACDF 共计 64 个字符

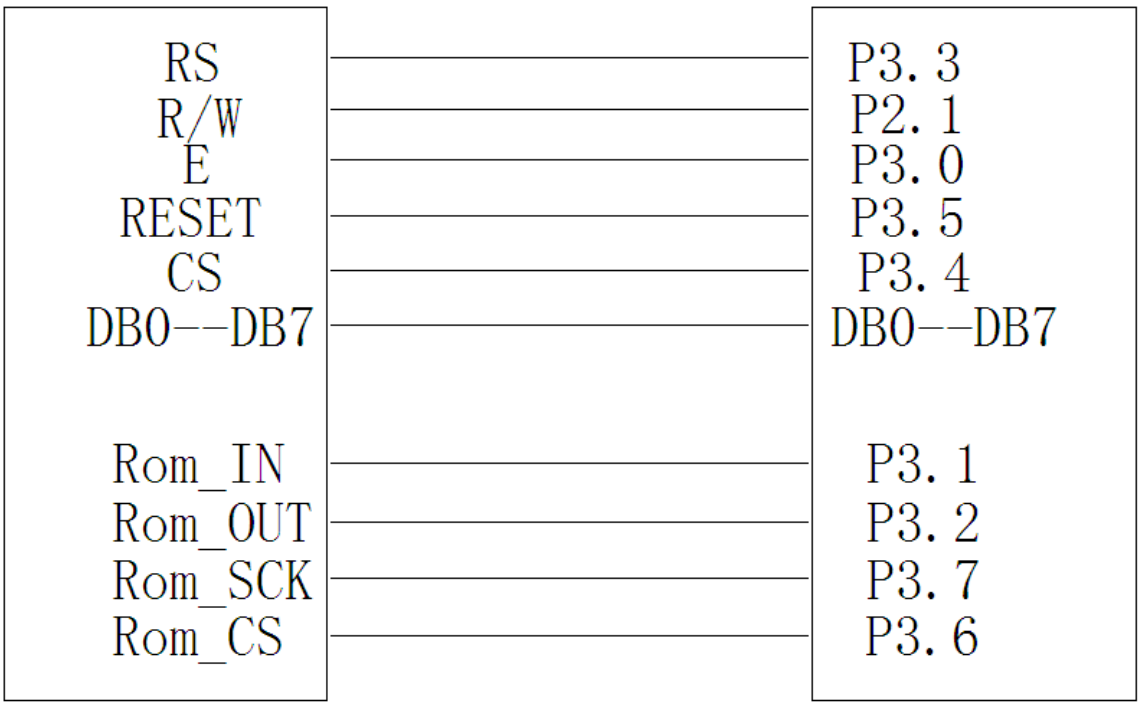


7. 硬件设计及例程：

7.1 当 LCD 驱动 IC 采用并行接口方式时的硬件设计及例程：



7.2 硬件接口： 下图为并行方式的硬件接口：



7.2 并行接口

```
/*    液晶模块型号：JLX256128G-921
      并行接口
      驱动 IC 是:ST75256
      版权所有：晶联讯电子：网址  http://www.jlxlcd.cn;
*/
#include <reg52.H>
#include <intrins.h>
#include <chinese_code.h>


sbit cs1=P3^4;      /*3.4 接口定义*/
sbit lcd_reset=P3^5;/*3.3 接口定义*/
sbit rs=P3^3;           /*接口定义*/
sbit rd=P3^0;           /*接口定义*/
sbit wr=P2^1;           /*接口定义。另外 P1.0~1.7 对应 DB0~DB7*/

sbit Rom_IN=P3^1;    /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI*/
sbit Rom_OUT=P3^2;   /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 SO*/
sbit Rom_CS=P3^6;    /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#*/
sbit Rom_SCK=P3^7;   /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/

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

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


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


/*延时：1us 的 i 倍*/
void delay_us(int i)
{
  int j,k;
  for(j=0;j<i;j++)
       for(k=0;k<1;k++);
}


/*等待一个按键，我的主板是用 P2.0 与 GND 之间接一个按键*/
void waitkey()
{
 repeat:
       if (key==1) goto repeat;
  else delay(3500);
}

//=============transfer command to LCM===============
void transfer_command_lcd(int data1)
{
  cs1=0;
  rs=0;
  rd=0;
  delay_us(1);
  wr=0;
  P1=data1;
  rd=1;
  delay_us(1);
  cs1=1;
  rd=0;
```

```
}

//-----------transfer data to LCM---------------
void transfer_data_lcd(int data1)
{
  cs1=0;
  rs=1;
  rd=0;
  delay_us(1);
  wr=0;
  P1=data1;
  rd=1;
  delay_us(1);
  cs1=1;
  rd=0;
}



void initial_lcd()
{
  lcd_reset=0;
  delay(100);
  lcd_reset=1;
  delay(100);

  transfer_command_lcd(0x30);   //EXT=0
  transfer_command_lcd(0x94);   //Sleep out
  transfer_command_lcd(0x31);   //EXT=1
  transfer_command_lcd(0xD7);   //Autoread disable
  transfer_data_lcd(0X9F);      //

  transfer_command_lcd(0x32);   //Analog SET
  transfer_data_lcd(0x00);        //OSC Frequency adjustment
  transfer_data_lcd(0x01);        //Frequency on booster capacitors->6KHz
  transfer_data_lcd(0x02);        //Bias=1/12

  transfer_command_lcd(0x20);   // Gray Level
  transfer_data_lcd(0x01);
  transfer_data_lcd(0x03);
  transfer_data_lcd(0x05);
  transfer_data_lcd(0x07);
  transfer_data_lcd(0x09);
  transfer_data_lcd(0x0b);
```

```
  transfer_data_lcd(0x0d);
  transfer_data_lcd(0x10);
  transfer_data_lcd(0x11);
  transfer_data_lcd(0x13);
  transfer_data_lcd(0x15);
  transfer_data_lcd(0x17);
  transfer_data_lcd(0x19);
  transfer_data_lcd(0x1b);
  transfer_data_lcd(0x1d);
  transfer_data_lcd(0x1f);

  transfer_command_lcd(0x30);   //EXT=0
  transfer_command_lcd(0x75);   //Page Address setting
  transfer_data_lcd(0X00);      // XS=0
  transfer_data_lcd(0X14);      // XE=159 0x28
  transfer_command_lcd(0x15);   //Clumn Address setting
  transfer_data_lcd(0X00);      // XS=0
  transfer_data_lcd(0Xff);      // XE=256

  transfer_command_lcd(0xBC);    //Data scan direction
  transfer_data_lcd(0x00);         //MX.MY=Normal
  transfer_data_lcd(0xA6);

  transfer_command_lcd(0xCA);    //Display Control
  transfer_data_lcd(0X00);         //
  transfer_data_lcd(0X9F);         //Duty=1/128
  transfer_data_lcd(0X20);         //Nline=off

  transfer_command_lcd(0xF0);    //Display Mode
  transfer_data_lcd(0X10);         //10=Monochrome Mode,11=4Gray

  transfer_command_lcd(0x81);    //EV control
  transfer_data_lcd(0x39);         //VPR[5-0]
  transfer_data_lcd(0x04);         //VPR[8-6]
  transfer_command_lcd(0x20);    //Power control
  transfer_data_lcd(0x0B);         //D0=regulator ; D1=follower ; D3=booste,  on:1 off:0
  delay_us(100);
  transfer_command_lcd(0xAF);    //Display on

}


/*写 LCD 行列地址：X 为起始的列地址，Y 为起始的行地址，x_total,y_total 分别为列地址及行地址的起点到终点的差值 */
void lcd_address(int x,int y,x_total,y_total)
{
  x=x-1;
```

```
  y=y-1;

  transfer_command_lcd(0x15); //Set Column Address
  transfer_data_lcd(x);
  transfer_data_lcd(x+x_total-1);

  transfer_command_lcd(0x75); //Set Page Address
  transfer_data_lcd(y);
  transfer_data_lcd(y+y_total-1);
  transfer_command_lcd(0x30);
  transfer_command_lcd(0x5c);

}


/*清屏*/
void clear_screen()
{
  int i,j;
  lcd_address(0,0,256,17);
  for(i=0;i<17;i++)
  {
       for(j=0;j<256;j++)
       {
            transfer_data_lcd(0x00);
       }
  }
}


void test(int x,int y)
{
  int i,j;
  lcd_address(x,y,256,16);

  for(i=0;i<16;i++)
  {
       for(j=0;j<256;j++)
       {
            transfer_data_lcd(0xff);
       }
  }
}

void display_string_16x16(uchar column, uchar page,uchar *text)
{
```

```
uchar i,j,k;
uint address;
j=0;
while(text[j]!= '\0')
{
    i=0;
    address=1;
    while(Chinese_text_16x16[i]> 0x7e)
    {
        if(Chinese_text_16x16[i] == text[j])
        {
            if(Chinese_text_16x16[i+1] == text[j+1])
            {
                address=i*16;
                break;
            }
        }
        i +=2;
    }
    if(column>255)
    {
        column=0;
        page+=2;
    }
    if(address !=1)
    {
        lcd_address(column,page,16,2);
        for(k=0;k<2;k++)
        {
            for(i=0;i<16;i++)
            {
                transfer_data_lcd(Chinese_code_16x16[address]);
                address++;
            }
        }
        j +=2;
    }
    else
    {
        for(k=0;k<2;k++)
        {
            lcd_address(column,page,16,2);
            for(i=0;i<16;i++)
            {
                transfer_data_lcd(0x00);
            }
```

```
            }
            j++;
        }
        column+=16;
    }
}


/*显示 32*32 点阵的汉字或等同于 32*32 点阵的图像*/
void disp_32x32(int x,int y,uchar *dp)
{
  int i,j;
  lcd_address(x,y,32,4);
  for(i=0;i<4;i++)
  {
        for(j=0;j<32;j++)
        {
            transfer_data_lcd(*dp);
            dp++;
        }
  }
}


/*显示 256*128 点阵的图像*/
void disp_256x128(int x,int y,char *dp)
{
  int i,j;
  lcd_address(x,y,256,16);
  for(i=0;i<16;i++)
  {
        for(j=0;j<256;j++)
        {
            transfer_data_lcd(*dp);
            dp++;
        }
  }
}
/****送指令到晶联讯字库 IC***/
void send_command_to_ROM( uchar datu )
{
  uchar i;
  for(i=0;i<8;i++ )
  {
        if(datu&0x80)
            Rom_IN = 1;
```

```
        else
            Rom_IN = 0;
            datu = datu<<1;
            Rom_SCK=0;
            Rom_SCK=1;
            delay_us(1);
    }
}

/****从晶联讯字库 IC 中取汉字或字符数据（1 个字节）***/
static uchar get_data_from_ROM( )
{
  uchar i;
  uchar ret_data=0;
  Rom_SCK=1;
  for(i=0;i<8;i++)
  {
        Rom_OUT=1;
        Rom_SCK=0;
        ret_data>>=1;
        if( Rom_OUT )
            ret_data+=0x80;
        else
            ret_data=ret_data+0;
        Rom_SCK=1;
        delay_us(1);
  }

  return(ret_data);
}


//从指定地址读出数据写到液晶屏指定 (page,column)座标中
void get_and_write_8x16(ulong fontaddr,uchar column,uchar page)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                          //地址的低 8 位,共 24 位
  lcd_address(column,page,8,2);
  for(j=0;j<2;j++)
  {
      for(i=0; i<8; i++ )
    {
```

```
            disp_data=get_data_from_ROM();
            transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
        }
  }
  Rom_CS=1;
}

void get_and_write_12x24(ulong fontaddr,uchar column,uchar page)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                          //地址的低 8 位,共 24 位
  lcd_address(column,page,16,3);
  for(j=0;j<3;j++)
  {
      for(i=0; i<16; i++ )
    {
            disp_data=get_data_from_ROM();
            transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
        }
  }
  Rom_CS=1;
}


void get_and_write_16x32(ulong fontaddr,uchar column,uchar page)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                  //地址的低 8 位,共 24 位
  lcd_address(column,page,16,4);
  for(j=0;j<4;j++)
  {
      for(i=0; i<16; i++ )
    {
            disp_data=get_data_from_ROM();
            transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
        }
  }
```

```
  Rom_CS=1;
}


//从指定地址读出数据写到液晶屏指定（page,column)座标中
void get_and_write_16x16(ulong fontaddr,uchar column,uchar page)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
  lcd_address(column,page,16,2);
  for(j=0;j<2;j++)
  {
      for(i=0; i<16; i++ )
    {
          disp_data=get_data_from_ROM();
          transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
      }
  }
  Rom_CS=1;
}

//从指定地址读出数据写到液晶屏指定（page,column)座标中
void get_and_write_24x24(ulong fontaddr,uchar column,uchar page)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
  lcd_address(column,page,24,3);
  for(j=0;j<3;j++)
  {
      for(i=0; i<24; i++ )
    {
          disp_data=get_data_from_ROM();
          transfer_data_lcd(disp_data);    //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1

      }
  }
  Rom_CS=1;
}
```

```
//从指定地址读出数据写到液晶屏指定（page,column)座标中
void get_and_write_32x32(ulong fontaddr,uchar column,uchar page)
{
  uchar i,j,disp_data;
  Rom_CS = 0;
  send_command_to_ROM(0x03);
  send_command_to_ROM((fontaddr&0xff0000)>>16);  //地址的高 8 位,共 24 位
  send_command_to_ROM((fontaddr&0xff00)>>8);          //地址的中 8 位,共 24 位
  send_command_to_ROM(fontaddr&0xff);                 //地址的低 8 位,共 24 位
  lcd_address(column,page,32,4);
  for(j=0;j<4;j++)
  {
      for(i=0; i<32; i++ )
    {
          disp_data=get_data_from_ROM();
          transfer_data_lcd(disp_data);   //写数据到 LCD,每写完 1 字节的数据后列地址自动加 1
      }
  }
  Rom_CS=1;
}


//*****************************************************************

ulong  fontaddr=0;
void display_GB2312_16x16_string(uchar column,uchar page,uchar *text)
{
  uchar i= 0,temp1,temp2;
  temp1=column;
  temp2=page;

  while((text[i]>0x00))
  {
      if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
      {
          //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
          //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
          //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
          fontaddr = (text[i]- 0xb0)*94;
          fontaddr += (text[i+1]-0xa1)+846;
          fontaddr = (ulong)(fontaddr*32);
          fontaddr = (ulong)(fontaddr+0x2c9d0);

          get_and_write_16x16(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
```

```
        i+=2;
        column+=16;

        if ((temp2<=15&&temp1<=256)&&column>248)
        {
            //自动换行，当遇到奇数个字母或符号就提前 8 个点
            //设成符>256 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
            column=1;
            page+=2;
            if (page>15)column=1;
        }
    }

    else if(((text[i]>=0xa1) &&(text[i]<=0xa3))&&(text[i+1]>=0xa1))
    {
        //国标简体（GB2312）15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算：
        //Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0
        //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
        fontaddr = (text[i]- 0xa1)*94;
        fontaddr += (text[i+1]-0xa1);
        fontaddr = (ulong)(fontaddr*32);
        fontaddr = (ulong)(fontaddr+0x2c9d0);

        get_and_write_16x16(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
        i+=2;
        column+=16;

        if ((temp2<=15&&temp1<=256)&&column>248)
        {
            //自动换行，当遇到奇数个字母或符号就提前 8 个点
            //设成符>128 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
            column=1;
            page+=2;
            if (page>15)column=1;
        }
    }

    else if((text[i]>=0x20) &&(text[i]<=0x7e))
    {
        fontaddr = (text[i]- 0x20);
        fontaddr = (unsigned long)(fontaddr*16);
        fontaddr = (unsigned long)(fontaddr+0x1dd780);

        get_and_write_8x16(fontaddr,column,page); //从指定地址读出数据写到液晶屏指定（page,column)座标中
        i+=1;
        column+=8;
```

```
            if ((temp1<=15&&temp2<=256)&&column>248)
            {
                //自动换行，当遇到奇数个字母或符号就提前 8 个点
                //设成符>128 时当有奇数个字符时就会显半个汉字,因为一个字符只占 8 个点（一个字节）
                column=1;
                page+=2;
                if (page>15)column=1;
            }
        }

        else
            i++;
    }
}


//*****************************************************************
void display_GB2312_24x24_string(uchar column,uchar page,uchar *text)
{
  uchar i= 0,temp1,temp2;
  temp1=column;
  temp2=page;

  while((text[i]>0x00))
  {
        if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
        {
            //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
            fontaddr = (text[i]- 0xb0)*94;
            fontaddr += (text[i+1]-0xa1)+846;
            fontaddr = (ulong)(fontaddr*72);
            fontaddr = (ulong)(fontaddr+0X068190);

            get_and_write_24x24(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=24;
        }

        else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
        {
            //国标简体（GB2312）15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
```

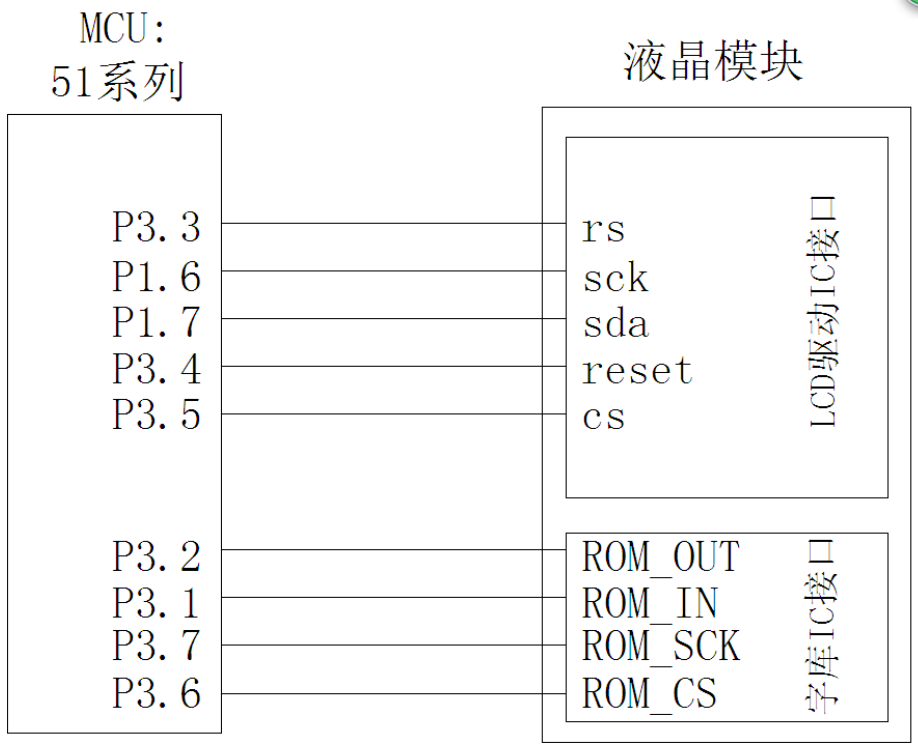
```
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*72);
            fontaddr = (ulong)(fontaddr+0X068190);

            get_and_write_24x24(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=24;
        }

        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long)(fontaddr*48);
            fontaddr = (unsigned long)(fontaddr+0x1dff00);
            get_and_write_12x24(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=1;
            column+=12;
        }
        else
            i++;
    }
}

//*************************************************************
void display_GB2312_32x32_string(uchar column,uchar page,uchar *text)
{
  uchar i= 0,temp1,temp2;
  temp1=column;
  temp2=page;

  while((text[i]>0x00))
  {
      if(((text[i]>=0xb0) &&(text[i]<=0xf7))&&(text[i+1]>=0xa1))
      {
          //国标简体（GB2312）汉字在晶联讯字库 IC 中的地址由以下公式来计算：
          //Address = ((MSB - 0xB0) * 94 + (LSB - 0xA1)+ 846)*32+ BaseAdd;BaseAdd=0
          //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
          fontaddr = (text[i]- 0xb0)*94;
          fontaddr += (text[i+1]-0xa1)+846;
          fontaddr = (ulong)(fontaddr*128);
          fontaddr = (ulong)(fontaddr+0xedf00);

          get_and_write_32x32(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
          i+=2;
          column+=32;
```

```
        }

        else if(((text[i]>=0xa1) &&(text[i]<=0xa9))&&(text[i+1]>=0xa1))
        {
            //国标简体（GB2312）15x16 点的字符在晶联讯字库 IC 中的地址由以下公式来计算：
            //Address = ((MSB - 0xa1) * 94 + (LSB - 0xA1))*32+ BaseAdd;BaseAdd=0
            //由于担心 8 位单片机有乘法溢出问题，所以分三部取地址
            fontaddr = (text[i]- 0xa1)*94;
            fontaddr += (text[i+1]-0xa1);
            fontaddr = (ulong)(fontaddr*128);
            fontaddr = (ulong)(fontaddr+0xedf00);

            get_and_write_32x32(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=2;
            column+=32;
        }

        else if((text[i]>=0x20) &&(text[i]<=0x7e))
        {
            fontaddr = (text[i]- 0x20);
            fontaddr = (unsigned long)(fontaddr*64);
            fontaddr = (unsigned long)(fontaddr+0x1e5a50);

            get_and_write_16x32(fontaddr,column,page);      //从指定地址读出数据写到液晶屏指定（page,column)座标中
            i+=1;
            column+=16;
        }
        else
            i++;
    }
}


//------------------------------
void main ()
{
  initial_lcd();                                          //对液晶模块进行初始化设置
  while(1)
  {

        clear_screen();
        display_GB2312_16x16_string(1,1,"GB2312 简体字 16X16、24X24、32X32");
        display_GB2312_16x16_string(1,3,"简体汉字库及 8X16、12X24、16X32");
        display_GB2312_16x16_string(1,5,"的 ASCII 码⑴①〇●◎◇◆ 1.2.‖…");
        display_GB2312_24x24_string(8,7,"啊阿埃挨哎唉哀皑癌蔼");
        display_GB2312_24x24_string(1,10,"ABCDEFGHIJKLMNOPQRSTU");
```

```
        display_GB2312_32x32_string(1,13,"深圳市晶联讯电子");
        waitkey();
        clear_screen();
        display_GB2312_32x32_string(1,1,"鑫森淼焱晶磊众品");
        display_GB2312_32x32_string(1,5,"鬟鬣麽麾縻麅麋麈");
        display_GB2312_32x32_string(1,9,"麋麒麈麝麟黛黜黝");
        display_GB2312_32x32_string(1,13,"點黟黢黩黧黡黲黯黯");
        waitkey();

        clear_screen();                                    //清屏
        disp_256x128(1,1,bmp8);                  //显示一幅 256*128 点阵的黑白图。
        waitkey();
        clear_screen();
        disp_32x32(49,1,jing2);
        disp_32x32((32*1+49),1,lian2);
        disp_32x32((32*2+49),1,xun2);
        disp_32x32((32*3+49),1,dian2);
        disp_32x32((32*4+49),1,zi2);
        display_string_16x16(33,7,"深圳市晶联讯电子有限公司");
        waitkey();
    }
}
```

7.3 当 LCD 驱动 IC 采用串行接口方式时的硬件设计及例程：

7.3.1 硬件接口：下图为串行方式的硬件接口：



7.3.2、以下为串行接口方式范例程序

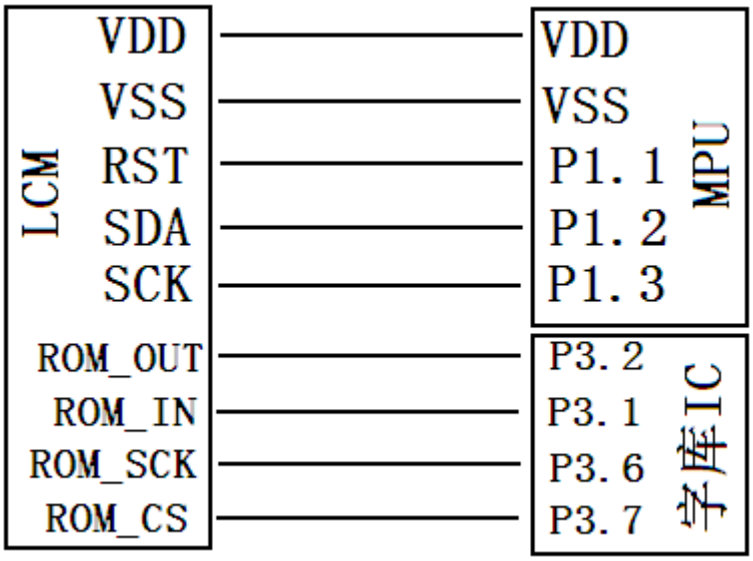
与并行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

```
sbit lcd_cs1  =  P3^5;//CS
sbit lcd_reset=  P3^4;//RST
sbit lcd_sclk =  P1^6;//串行时钟
sbit lcd_rs   =  P3^3;//RS
sbit lcd_sid  =  P1^7;//串行数据
sbit key=P2^0;          /*按键接口，P2.0 口与 GND 之间接一个按键*/

sbit Rom_IN=P3^1;    /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI*/
sbit Rom_OUT=P3^2;   /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 SO*/
sbit Rom_SCK=P3^7;   /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
sbit Rom_CS=P3^6;    /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#*/


//写指令到 LCD 模块
void transfer_command_lcd(int data1)
{
     char i;
     lcd_cs1=0;
     lcd_rs=0;
     for(i=0;i<8;i++)
```

```
    {
        lcd_sclk=0;
        if(data1&0x80) lcd_sid=1;
        else lcd_sid=0;
        lcd_sclk=1;
        data1<<=1;
    }
    lcd_cs1=1;
}

//写数据到 LCD 模块
void transfer_data_lcd(int data1)
{
    char i;
    lcd_cs1=0;
    lcd_rs=1;
    for(i=0;i<8;i++)
    {
        lcd_sclk=0;
        if(data1&0x80) lcd_sid=1;
        else lcd_sid=0;
        lcd_sclk=1;
        data1<<=1;
    }
    lcd_cs1=1;
}
```

7.4 当 LCD 驱动 IC 采用 IIC 接口方式时的硬件设计及例程：

7.4.1 硬件接口：下图为 IIC 方式的硬件接口：



7.4.2、以下为 IIC 接口方式范例程序

与串行方式相比较，只需改变接口顺序以及传送数据、传送命令这两个函数即可：

```
sbit  reset=P1^1;
sbit scl=P1^3;
sbit sda=P1^2;
sbit  key=P2^0;

sbit Rom_IN=P3^1;       /*字库 IC 接口定义:Rom_IN 就是字库 IC 的 SI*/
sbit Rom_OUT=P3^2;      /*字库 IC 接口定义:Rom_OUT 就是字库 IC 的 SO*/
sbit Rom_SCK=P3^7;      /*字库 IC 接口定义:Rom_SCK 就是字库 IC 的 SCK*/
sbit Rom_CS=P3^6;       /*字库 IC 接口定义 Rom_CS 就是字库 IC 的 CS#*/


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(0x78);
    transfer(0x80);
    transfer(com);
    stop_flag();
}

//写数据到液晶显示模块
void transfer_data(uchar dat)
{
    start_flag();
    transfer(0x78);
    transfer(0xC0);
    transfer(dat);
    stop_flag();
}
```

-EDN-