E 系列指令集

目　录

一.控制接口

1.1控制接口电气标准

模组的控制接口引脚定义（连接器型号：molex 0022057085）如表 2 所示。单片机等微控制器/微处理器通过此接口与模组进行通信。

管脚号	信号	说明
1	VCC	数字电源 5V 输入
2	VCC	数字电源 5V 输入
3	BUSY	串口缓冲区满信号标志
4	TX	串口输出 （RS232 电平或者 TTL 电平）注 1
5	RX	串口输入 （RS232 电平或者 TTL 电平）注 1
6	RX	串口输入 （RS232 电平或者 TTL 电平）注 1
7	GND	数字地输入
8	GND	数字地输入

注 1：通过跳线电阻转换

二.MIS-E V1.0 晶奥串口指令集

2.1 概述

用户使用的指令格式如表 6.1.1 所示:

完整指令帧	帧头	指令	指令所需参数	帧尾
	AA	XX	XX	CC 33 C3 3C

表 6.1.1

一条指令帧最多 256 字节（包括帧头帧尾）。

举例说明，要在坐标(100,50)处显示一张储存在 FLASH 中的 5 号位图片，显示方式为顶层显示，则需要向模组发送串口指令如下：

发送指令：**AA 71 00 05 00 64 00 32 CC 33 C3 3C**

指令说明：

AA	指令帧头	每帧指令的开头

71	指令	顶层显示图片指令
00 05	指令所需参数	此例中数据表示图片 5
00 64 00 32	指令所需参数	表示坐标（100,50）
CC 33 C3 3C	指令帧尾	每帧指令的结尾

表 6.1.2

显示效果如图 6.1

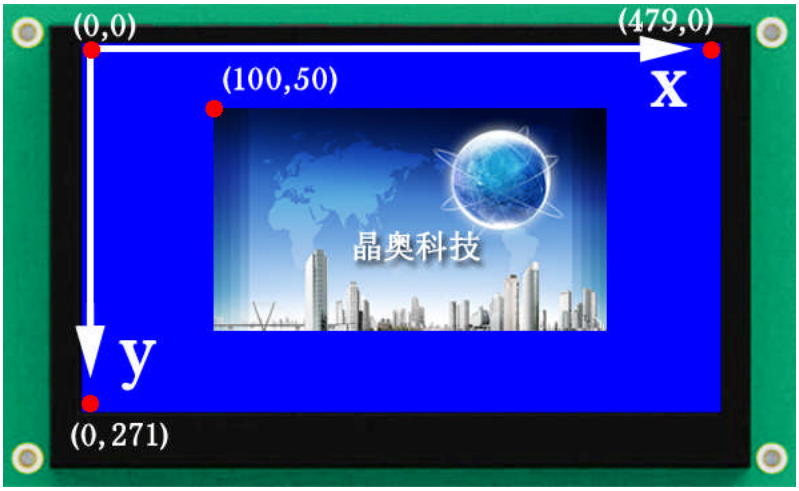

图 6.1

2.2 坐标及图层说明

以 M043S65-E 为例，屏幕分辨率为 480*272 ，即 X 方向坐标由左到右从 0 至 479，Y 方向坐标从 0 至 271。图片、字符的显示位置，以左上角坐标为准，如上图 6.1 中，图片坐标为(100，50)。

E 系列模组共有两个图层（顶层和底层），顶层色值为 0x0000 的区域，将透过顶层显示到底层。详细请参考图 6.4.9.1 实例。

2.3指令集列表（请点击超链接实现跳转）

分类	功能	指令	指令数据	说明
系统	握手指令	0x00	无	用于判断串口模组是否上电初始化完毕，通讯是否正常。 驱动函数：void HandShake( )
显示参数配置	设置调色板	0x40	F_col + B_col	<F_col>2 个字节：前景色； <B_col>2 个字节：背景色。 该指令用于设置前景色和背景色。颜色设定后，若无重新设定，就会一直保持下来。

				驱动函数：void Set_Color(uint F_col,uint B_col)
	设置字符显示行列间距	0x41	Xdis + Ydis	<Xdis>1 个字节：X 方向的字符间距，取值 0~7F； <Ydis>1 个字节：Y 方向的字符间距，取值 0~7F。
				驱动函数:void Set_Space(uchar Xdis,uchar Ydis)
文本显示	16X16 点阵	**0x54**	X+Y+String	<X>2 个字节：显示字符的起始位置 X 坐标； <Y>2 个字节：显示字符的起始位置 Y 坐标； <String>不大于 246 个字节 :要显示的字符串，汉字采用 GB2312 编码，遇到行末会自动换行。
	32X32 点阵	**0x55**		
	12X12 点阵	**0x6E**		驱动函数:void Write_Text_Normal(char Type,uint X, uint Y,uchar *Str) Type:字符大小(Type 有 4 种可选择的值：0x54,0x55, 0x6E,0x6F，对应不同大小的 4 种字符)
	24X24 点阵	**0x6F**		
	选择字库显示（包括自定义字库）	**0x98**	X+Y+Lib_ID+ C_Mode+ +F_col +B_col+ String	<X>2 个字节：显示字符的起始位置 X 坐标； <Y>2 个字节：显示字符的起始位置 Y 坐标； < Lib_ID > 1 个字节： 0x20：显示 12*12 点阵字符(ASCII 码为 6*12)； 0x21：显示 16*16 点阵字符(ASCII 码为 8*16)； 0x22：显示 24*24 点阵字符(ASCII 码为 12*24)； 0x23：显示 32*32 点阵字符(ASCII 码为 16*32)。 0x24：显示 20*40 大号 ASCII 码 0x25：显示 24*48 大号 ASCII 码 0x26：显示 28*56 大号 ASCII 码 0x27：显示 32*64 大号 ASCII 码 < C_Mode > 1 个字节： Bit7 取 0 文本前景色不显示； 取 1 文本前景色显示； Bit6 取 0 文本背景色不显示； 取 1 文本背景色显示； Bit5-0 保留； < F_col > 2 个字节：文本显示的前景色(不会改变系统设置的调色板)； < B_col>2 个字节 :文本显示的背景色(不会改变系统设置的调色板)； < String>不大于 239 个字节：要显示的字符串。采用 GB2312 编码，显示字符间距由 0x41 指令设定，遇到行末会自动换行。
				驱动函数: void Write_Text_Special(uint X,uint Y,uchar Lib_ID,uchar C_Mode, uint F_col,uint B_col,uchar *Str)
置点	背景色显示点	**0x50**	X+Y+(X1+Y1+X2+ Y2) ……	<X>2 个字节、<Y>2 个字节 分别是所显示点的<X> <Y>坐标值。如需一次绘制多个点，可在发帧尾前，发送多组坐标值。坐标总数据量不大于 250 字节。
	前景色显示点	**0x51**		驱动函数: void Draw_Dot(uchar Mode,uint X,uint Y) Mode 控制显示模式。如需一次绘制多个点，可自由修改

				驱动函数，在发帧尾前，发送多组坐标值。
多线段连线	多个指定点用线段进行连接（前景色）	0x56	X1+Y1+X2+Y2+（X3+Y3 +……）	指令 0x56 使用前景色把指定点用线段连接； 指令 0x5D 使用背景色把指定点用线段连接； <X1>2 个字节：第一个点 X 坐标； <Y1>2 个字节：第一个点 Y 坐标； <X2>2 个字节：第二个点 X 坐标； <Y2>2 个字节：第二个点 Y 坐标。
	多个指定点用线段进行连接（背景色）	0x5D		驱动函数: void Draw_Line(uchar Mode,uint X1,uint Y1,uint X2,uint Y2) Mode 控制显示模式。如需一次绘制多条线，可自由修改驱动函数，在发帧尾前，发送多组坐标值。
画圆	画圆	0x57	Mode+X+Y+R	<Mode> 1 个字节：Mode=0x00：背景色画圆；Mode=0x01：前景色画圆（由 0x40 指令设置）。 <（X，Y）> 共 4 个字节：圆心坐标。 <R>1 个字节：圆的半径。
				驱动函数: void Draw_Circle(uchar Mode,uint X,uint Y,uchar R)
区域操作	前景色显示多个矩形框	0x59	X_sta+Y_sta+X_end+Y_end+……	<X_sta>2 个字节：矩形区域左上角 X 坐标； <Y_sta>2 个字节：矩形区域左上角 Y 坐标； <X_end>2 个字节：矩形区域右下角 X 坐标； <Y_end>2 个字节：矩形区域右下角 Y 坐标。
	背景色显示多个矩形框	0x69		驱动函数: void Draw_Rectangle(uchar Mode,uint X_sta,uint Y_sta,uint X_end,uint Y_end) Mode 控制 前景色/背景色 显示模式。如需一次绘制多个框，可自由修改驱动函数，在发帧尾前，发送多组坐标值。
	前景色填充多个矩形框	0x5B	X_sta+Y_sta+X_end+Y_end+……	<X_sta>2 个字节：矩形区域左上角 X 坐标； <Y_sta>2 个字节：矩形区域左上角 Y 坐标； <X_end>2 个字节：矩形区域右下角 X 坐标； <Y_end>2 个字节：矩形区域右下角 Y 坐标。
				驱动函数: void Fill_Rectangle(uchar Mode,uint X_sta,uint Y_sta,uint X_end,uint Y_end) Mode 控制 前景色/背景色 显示模式。如需一次填充多个区域，可自由修改驱动函数，在发帧尾前，发送多组坐标值。
	背景色填充多个矩形框	0x5A		
	全屏清屏	0x52	无	用当前背景色全屏清屏（背景色用 0x40 指令设置）。
				驱动函数：void Clear_Screen();
图片显示	顶层显示图片（位号 0-359）	0x70	Image_num+X+Y	< Image_num >2 个字节 :保存在智能终端 Flash 存储器的图片位号。 <X><Y>各 2 个字节:图片显示位置对应的 X,Y 坐标。
	底层显示图片（位号 0-359）	0x71		驱动函数: void Display_Image(uchar Layer, uint Image_num,uint X,uint Y)

	从保存在终端的一幅图片剪切一部分显示（位号 0-359）	0x9B	Layer+Image_num+X_sta+Y_sta+X_end+Y_end+Xdis+Ydis	<Layer>1 个字节 :取 0xFF 顶层显示 ,取 0x00 底层显示； < Image_num >2 个字节：保存在智能终端 Flash 存储器的图片位号； <X_sta> <Y_sta>共 4 个字节：表示要剪切区域在原图的左上角坐标； <X_end> <Y_end>共 4 个字节：表示要剪切区域在原图的右下角坐标； <Xdis> <Ydis>共 4 个字节：表示剪切下来的图片在当前屏幕显示位置的左上角坐标。
				驱动函数: void Display_CutImage(uchar Layer, uint Image_num ,uint X_sta,uint Y_sta,uint X_end,uint Y_end,uint X_dis,uint Y_dis)
	图片连续播放显示（位号 0-359）	0x9C	Layer+Mode+Time+X+Y+Image_1+Image_2+ ……(Image_n)	<Layer> 1 个字节：图片显示层设置，取 0x00 底层显示，取 0xff 顶层显示。 <Mode> 1 个字节：连续循环次数，其中 0x00 表示停止当前循环（该值下仍可显示 Image_1 的图片），0xff 表示无限循环显示。 <Timer>1 个字节：每次切换图片的时间间隔为 Timer*5ms，最小值为 40ms。 <xadd><yadd> 4 个字节 :(xadd,yadd)为显示图片的起始位置。 Photo_num_x 为依次显示的图片位号。 < Image _x>2 个字节： Photo_num_x 为依次要显示的保存在智能终端 Flash 存储器的图片位号。(x=1,2,3…,n)
				驱动函数:void Continuous_Image(uchar Layer,uchar Mode,uchar Time,uint X,uint Y,uint Image_1,uint Image_2)如需动画显示更多图片，可自由修改驱动函数，在发帧尾前，发送多个图片位号。
背光控制	背光关闭	0x5E	无	将液晶屏背光关闭。
				驱动函数:void Backlight_OFF()
	打开背光	0x5F	无	将液晶屏背光打开，默认显示最高亮度。 无背光调节功能的模块，打开背光直接发送指令“AA 5F CC 33 C3 3C”即可。
				驱动函数:void Backlight_ON()
	背光 PWM 调节		Pwm	<Pwm>1 个字节：调节背光亮度，取值 0x01-0x3F,0x3F 时为最大亮度。
				驱动函数: void Set_Backlight(uchar Pwm)
参数配置	波特率设置	0xE0	0x55+0xAA+0x5A+0xA5+0x00+Bps +0x00	<Bps>1 个字节：设置串口波特率，上电默认波特率为 115200bps。下电后会保留当前波特率设置。
				驱动函数: void Set_ComBps(uchar Bps)
触摸屏控制	校准模式	0xE4	0x55+0xAA+0x5A+0xA5	进入触摸屏校准模式。
				驱动函数: void Set_Touch()

	开关控制	0xE4	0x66+0x99+0x69+0xFF	开启触摸屏控制。
				驱动函数: void Touch_ON()
		0xE4	0x66+0x99+0x69+0x00	关闭触摸屏控制。
				驱动函数: void Touch_OFF()
	触摸屏按下后位置上传	0x73	Xpos + Ypos	（Xpos，Ypos）表示被触发处的坐标值，X、Y 均为 2 个字节表示，且高字节在前传送。
				驱动函数:无
蜂鸣器控制	发送该指令后蜂鸣器发声	0x79	Time	< Btime>1 个字节：蜂鸣器发声长度，Time*10ms。发送该指令后蜂鸣器发声。
				驱动函数: void ON_Buzzer(uchar Time)
键盘控制	4*4 矩阵键盘	0x71	KeyValue	< KeyValue >1 个字节:表示 4*4 键盘的键值 ,0x00-0x0F 当键盘被按下后，回传键值。键值编码对应 4*4 矩阵键盘从左到右、从上到下从 0x00-0x0F。
				驱动函数:无

2.4 指令集详解（请点击超链接实现跳转）

2.4.1 握手指令 (0x00)

用户发送：**AA 00 CC 33 C3 3C**
设备返回：**AA version**
指令说明：握手指令主要用于判断串口模组是否上电初始化完毕，通讯是否正常。用户发送握手指令后，设备返回数据 **AA version** ，则表示设备通讯正常，连机成功。其中 **version** 是当前串口彩色液晶模块终端版本号。例如：AA 00（00 即为版本号）。

2.4.2 设置当前调色板 (0x40)

操作指令：**AA 40 Fcor Bcor CC 33 C3 3C**
参数说明： **Fcor (2** 字节**)** ：前景色的色值（2 个字节，即有 2^16=65536 种颜色）；
Bcor (2 字节**)** ：背景色的色值（2 个字节，即有 2^16=65536 种颜色）；

16bit 调色板定义 5R6G5B 模式（5bit 红色 6bit 绿色 5bit 蓝色），如下表所示：

位数	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
颜色	R					G						B				

表 6.4.2

每一色的色值越大，其颜色的成分越多，如：
纯红色色值：**F800** H（对应表 6.4.2 中 16 位：**1111 1000 0000 0000**）；

纯绿色色值：**07E0** H（对应表 6.4.2 中 16 位：**0000 0111 1110 0000**）；
纯蓝色色值：**001F** H（对应表 6.4.2 中 16 位：**0000 0000 0001 1111**）；

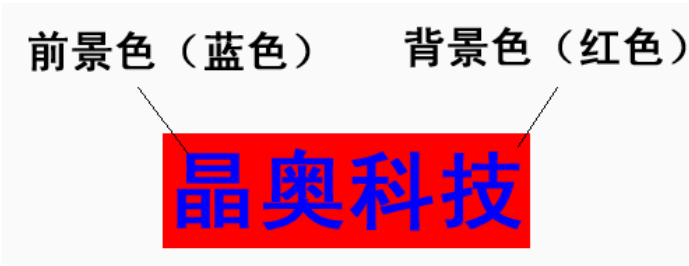

图 6.4.2

指令说明：该指令用于设置前景色和背景色。发送该指令后液晶屏不会立刻有显示颜色变化，只有发送其它涉及前景色/背景色显示的指令时，屏才会有色彩显示变化。颜色设定后，若无重新设定，就会一直保持下来。

2.4.3 设置字符间距 (0x41)

操作指令：**AA 41 Xdis Ydis CC 33 C3 3C**
参数说明：**Xdis (1 字节) ：X 方向的字符间距（列间距），取值范围 0x00-0x7F，默认值为 0x00；**
Ydis (1 字节) ：Y 方向的字符间距（列间距），取值范围 0x00-0x7F，默认值为 0x00 .
指令说明：该指令用于设置字符间距，间距设定后，若无重新设定，就会一直保持下来。

图 6.4.3

2.4.4 文本显示 (0x54,0x55,0x6E,0x6F,0x98)

2.4.4.1 标准文本显示 (0x54,0x55,0x6E,0x6F)

操作指令：**AA CMD X Y String CC 33 C3 3C**
参数说明：**CMD(1 字节) ：**
0x54：显示 16*16 点阵字符（AscII 码字符以半角 8*16 点阵显示）；
0x55：显示 32*32 点阵字符（AscII 码字符以半角 16*32 点阵显示）；
0x6E：显示 12*12 点阵字符（AscII 码字符以半角 6*12 点阵显示）；
0x6F：显示 24*24 点阵字符（AscII 码字符以半角 12*24 点阵显示）；
X (2 字节) ：显示字符的起始位置 **X** 坐标（即第一个字符左上角 **X** 坐标值）；
Y (2 字节) ：显示字符的起始位置 **y** 坐标（即第一个字符左上角 **y** 坐标值）；
String (不大于 246 字节) ：要显示的字符串。

指令说明：在屏幕的任意位置显示指定字库的文本。汉字采用 GB2312 编码。用户设定完文字的前景色、背景色及字符间距后，通过这条指令，即可直接显示字符。显示颜色由 **0x40** 指令设定，显示字符间距由 **0x41** 指令设定，遇到行末会自动换行。

参考实例：见下实例。

2.4.4.2 选择字库显示 (0x98)

操作指令：**AA 98 X Y Lib_ID C_Mode Fcor Bcor String CC 33 C3 3C**

参数说明：**X (2** 字节**)** ：显示字符的起始位置 **X** 坐标（即第一个字符左上角 **X** 坐标值）；

Y (2 字节**)** ：显示字符的起始位置 **y** 坐标（即第一个字符左上角 **y** 坐标值）；

Lib_ID (1 字节**)** ：选择所显示的字库。（包括自定义字库）

0x20：显示 12*12 点阵字符（AscII 码字符以半角 6*12 点阵显示）；

0x21：显示 16*16 点阵字符（AscII 码字符以半角 8*16 点阵显示）；

0x22：显示 24*24 点阵字符（AscII 码字符以半角 12*24 点阵显示）；

0x23：显示 32*32 点阵字符（AscII 码字符以半角 16*32 点阵显示）；

0x24：显示 20*40 大号 ASCII 码；

0x25：显示 24*48 大号 ASCII 码；

0x26：显示 28*56 大号 ASCII 码；

0x27：显示 32*64 大号 ASCII 码。

C_Mode (1 字节**)** ：

位	7	6	5	4	3	2	1	0
功能	前景色 1：显示 0：不显示	背景色 1：显示 0：不显示	保留（一般取 0）					

Fcor (2 字节**)** ：字符的前景色（此指令不改变设置调色板）；

Bcor (2 字节**)** ：字符的背景色（此指令不改变设置调色板）；

String (不大于 **239** 字节**)** ：要显示的字符串。

指令说明：该指令用于显示字符。可适用与多种字库，并可调用自定义字库。

参考实例：以 Flash 中存储的 1 号图为背景，在坐标(50,50) 处显示间距为 18，大小为 32*32 的文字“晶奥科技”（白色字体无背景色），在坐标(50,100)处显示间距为 0，大小为 32*32 的文字“晶奥科技”（蓝底红字），在坐标(50,150)处显示间距为 34，大小为 16*32 的 ASCII 字符“12ABC”（绿底蓝字）。

实例效果：

图 6.4.4

实例程序：

```
{
    Display_Image(1,1,0,0);                    //顶层显示 Flash 中 1 号满屏图片
    Set_Color(0xF800,0x001F);                  //前景色红色，背景色蓝色
    Write_Text_Normal(0x55,50,100,"晶奥科技");  //显示蓝底红字“晶奥科技”
    Set_Space(18,0);                           //设置字符 X 方向间距为 18
    Write_Text_Special(50,50,0x23,0x80, 0xFFFF,0x0000,"晶奥科技");
                                               //显示白色“晶奥科技”
    Set_Space(34,0);                           //设置字符 X 方向间距为 34
    Write_Text_Special(50,150,0x23,0xC0, 0x001F,0x07E0,"12ABC");
                                               //显示字符“12ABC”
}
```

2.4.5 显示点 (0x50,0x51)

操作指令：**AA CMD X1 Y1 (X2 Y2 ...Xn Yn) CC 33 C3 3C**

参数说明：**CMD (1 字节)** ：

0x51：前景色显示点**(0x40** 指令设置颜色**)**

0x50：背景色显示点**(0x40** 指令设置颜色**)**

X1 (2 字节) ：第一个点的 **X** 坐标值；

Y1 (2 字节) ：第一个点的 **Y** 坐标值；

. .

. .

. .

Xn (2 字节) ：第 **n** 个点的 **X** 坐标值；

Yn (2 字节) ：第 **n** 个点的 **Y** 坐标值。

指令说明：该指令主要实现在屏幕的任意位置画点。用户选择使用前景色画点**(0x51)**或背景色画点**(0x50)**，坐标数据总数不大于 250 字节，即一帧串口指令最多显示 62 个点）。

参考实例：在屏幕坐标(50,50)处画一白色点，在坐标(100,100)和坐标(100,150)处各画一个红色点。

实例效果：

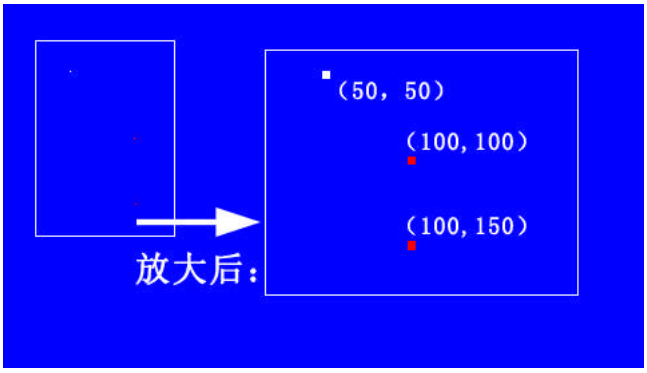

图 6.4.5

实例程序：

```
{
    Set_Color(0x0000,0x001F);  //前景色任意（此处为黑色），背景色蓝色
    Clear_Screen();            //背景色清屏
    Set_Color(0xF800,0xFFFF);  //前景色红色，背景色白色
    Draw_Dot(0,50,50);          //用背景色在（50，50）处画点
    Draw_Dot(1,100,100);        //用前景色在（100,100）处画点
    Draw_Dot(1,100,150);       //用前景色在（100,150）处画点
}
```

2.4.6 画圆指令 (0x57)

操作指令：**AA 57 Mode X Y R CC 33 C3 3C**

参数说明：**Mode** (**1** 字节) ：

0x00: 背景色画圆；

0x01: 前景色画圆；

X (**2** 字节) ：圆心 **X** 坐标值；

Y (**2** 字节) ：圆心 **Y** 坐标值；

R (**1** 字节) ：圆的半径。

指令说明：该指令实现以指定的坐标为圆心，画一个半径为 **R** 的圆。圆的颜色由调色板的前景色、背景色，以及本条指令的 **Mode** 决定。

参考实例：以(100,100)为圆心，50 为半径画一个红色圆，70 为半径画一个蓝色圆。

实例效果：

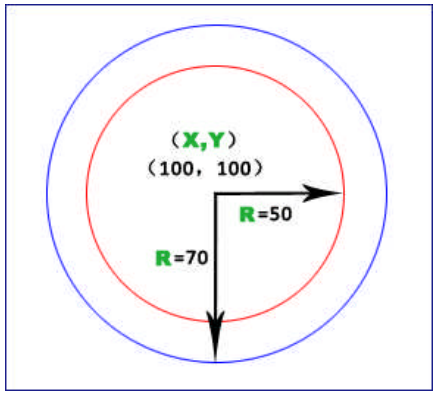

图 6.4.6

参考程序：

```
{
    Set_Color(0x0000,0xFFFF);    //前景色任意（此处为黑色），背景色白色
    Clear_Screen();              //背景色清屏
    Set_Color(0xF800,0x001F);    //前景色红色，背景色蓝色
    Draw_Circle(1,100,100,50);   //用前景色以（100,100）为圆心，50 为半径画圆
    Draw_Circle(0,100,100,70);   //用背景色以（100,100）为圆心，70 为半径画圆
}
```

2.4.7 区域显示指令 (0x59,0x69,0x5B,0x5A,0x52)

2.4.7.1 矩形区域操作 (0x59,0x69,0x5B,0x5A)

操作指令：**AA CMD X1_sta Y1_sta X1_end Y1_end (X2_sta Y2_sta X2_end Y2_end ... Xn_sta Yn_sta Xn_end Yn_end) CC 33 C3 3C**

参数说明：**CMD (1 字节)** ：

0x59：使用前景色显示矩形框，线宽为 **1** 像素**(0x40** 指令设置颜色**)**；

0x69：使用背景色显示矩形框，线宽为 **1** 像素**(0x40** 指令设置颜色**)**；

0x5B：使用前景色填充矩形区域 **(0x40** 指令设置颜色**)**；

0x5A：使用背景色填充矩形区域 **(0x40** 指令设置颜色**)**；

X1_sta (2 字节**)** ：第一个矩形区域左上角 **X** 坐标值；

Y1_sta (2 字节**)** ：第一个矩形区域左上角 **Y** 坐标值；

X1_end (2 字节**)** ：第一个矩形区域右下角 **X** 坐标值；

Y1_end (2 字节**)** ：第一个矩形区域右下角 **Y** 坐标值；

. .

. .

. .

Xn_sta (2 字节**)** ：第 **n** 个矩形区域左上角 **X** 坐标值；

Yn_sta (2 字节**)** ：第 **n** 个矩形区域左上角 **Y** 坐标值；

Xn_end (2 字节**)** ：第 **n** 个矩形区域右下角 **X** 坐标值；

Yn_end (2 字节**)** ：第 **n** 个矩形区域右下角 **Y** 坐标值；

指令说明：该指令实现在屏幕指定矩形区域画矩形或填充矩形。

参考实例：以 Flash 中存储的 2 号图片为背景，将屏幕坐标(50,50)到(100,100)的方形区域用红色清屏；并在坐标(200,20)到(350,150)的区域画一个蓝色方框。

实例效果：

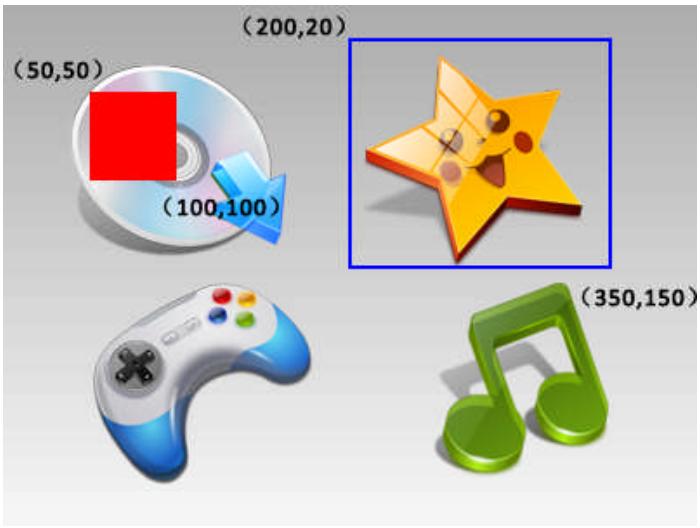

图 6.4.7

实例程序：

```
{
    Display_Image(1,2,0,0);              //顶层显示 FLASH 中存储的 2 号位图片
    Set_Color(0xF800,0x001F);            //前景色红色，背景色蓝色
    Fill_Rectangle(1,50,50,100,100);     //用前景色填充红色矩形
    Draw_Rectangle(0,200,20,350,150);    //用背景色画蓝色方框
}
```

2.4.7.2 全屏清屏 (0x52)

操作指令：**AA 52 CC 33 C3 3C**
指令说明：该指令使用背景色清屏。

2.4.8 连线显示 (0x56,0x5D)

操作指令：**AA CMD X1 Y1 (X2 Y2 ...Xn Yn) CC 33 C3 3C**
参数说明：**CMD (1 字节)** ：
0x56：使用前景色连接指定点**(0x40** 指令设置颜色**)**；
0x5D：使用背景色连接指定点**(0x40** 指令设置颜色**)**；
X1 (2 字节) ：第一个点的 **X** 坐标值；
Y1 (2 字节) ：第一个点的 **Y** 坐标值；
. .
. .
. .
Xn (2 字节) ：第 **n** 个点的 **X** 坐标值；
Yn (2 字节) ：第 **n** 个点的 **Y** 坐标值。
指令说明：该指令实现将指定的多个坐标点连接起来。
参考实例：用连线功能画一个简易表格。

实例效果：

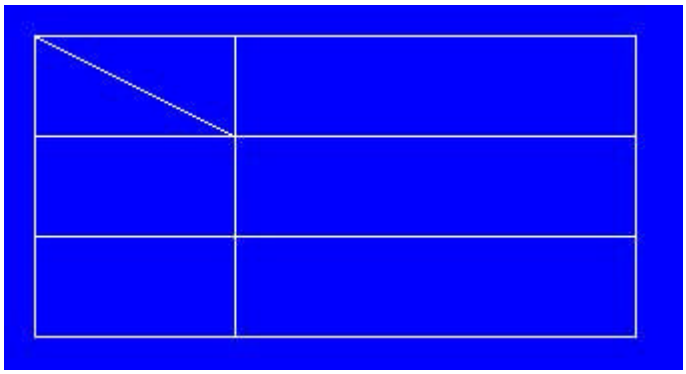

图 6.4.8

实例程序：

```
{
    Set_Color(0xFFFF,0x001F);          //前景色白色，背景色蓝色
    Clear_Screen();                    //背景色清屏
    Draw_Rectangle(1,50,50,350,200);   //用前景色画矩形外框
    Draw_Line(1,50,100,350,100);       //框内横线 1
    Draw_Line(1,50,150,350,150);       //框内横线 2
    Draw_Line(1,150,50,150,200);       //框内纵线
    Draw_Line(1,50,50,150,100);        //框内斜线
}
```

2.4.9 图片和图标显示 (0x70,0x71,0x9B,0x9C)

2.4.9.1 图片显示 (0x70,0x71)

操作指令：**AA CMD Image_num X Y CC 33 C3 3C**

参数说明：**CMD (1 字节)** ：

0x70：顶层显示图片；

0x71：底层显示图片；

Image _num (2 字节) ：保存在 FLASH 存储器的图片位号；

X (2 字节) ：图片显示位置的 **X** 坐标；

Y (2 字节) ：图片显示位置的 **Y** 坐标。

指令说明：该指令实现在指定坐标处，显示一张保存在 FLASH 中的图片。

参考实例：在底层显示 FLASH 中的 1 号图作为背景图，在顶层(10,10)坐标处显示一张 10 号位小图片(小图片黑色区域色值为 0x0000，即纯黑色)，两个图层的图像将进行拼合，可参看图层说明。

实例效果：

FLASH 中 1 号图

FLASH 中 10 号图

合成效果（图 6.4.9.1）

实例程序：

```
{
    Display_Image(0,1,0,0);          //底层(0,0)坐标处显示 1 号图
    Display_Image(1,10,10,10);       //顶层(10,10)坐标处显示 10 号图
}
```

2.4.9.2 剪切图片显示 (0x9B)

操作指令：**AA 9B Layer Image _num X_sta Y_sta X_end Y_end X_dis Y_dis CC 33 C3 3C**

参数说明：
- **Layer** (1 字节)：取 **0xFF** 顶层显示，取 **0x00** 底层显示；
- **Image _num** (2 字节)：被剪切图片的图位号；
- **X_sta** (2 字节)：图片剪切区域的左上角 **X** 坐标值；
- **Y_sta** (2 字节)：图片剪切区域的左上角 **Y** 坐标值；
- **X_end** (2 字节)：图片剪切区域的右下角 **X** 坐标值；
- **Y_end** (2 字节)：图片剪切区域的右下角 **Y** 坐标值；
- **X_dis** (2 字节)：剪切下的图片在屏幕上放置位置的左上角 **X** 坐标值；
- **Y_dis** (2 字节)：剪切下的图片在屏幕上放置位置的左上角 **Y** 坐标值。

指令说明：该指令用于实现将指定图片的某一指定区域裁切，并显示到屏幕的指定位置。

参考实例：全屏显示 FLASH 中的 1 号位图片，并将 2 号图片的(200,20)至(350,150)区域剪切至屏幕(20,20)

处显示。

实例效果：

Flash 中 1 号图

Flash 中 2 号图

效果图（图 6.4.9.2）

实例程序：

```
{
    Display_Image(1,1,0,0);                    //顶层显示 1 号位满屏图片
    Display_Cut_Image(1,2,200,20,350,150,20,20);
        //将 2 号位图片(200,20 )至(350,150)区域剪切至屏幕(20,20)处显示(顶层显示)
}
```

2.4.9.3 图片连续播放显示 (0x9C)

操作指令：**AA 9C Layer Mode Time X Y Image_1 Image_2 …… (Image_n) CC 33 C3 3C**

参数说明：**Layer (1 字节)** ：图片显示层标志位**(0xFF** 顶层显示 **0x00** 底层显示**)**；

Mode (1 字节) ：连续循环的次数。其中 **0x00** 表示停止当前循环（该值下仍可显示 Image_1 的图片），**0xFF** 表示无限循环显示；

Time (1 字节) ：每次切换图片的时间间隔为 Timer*5ms，最小值为 40ms；

X (2 字节) ：图片显示位置的 **X** 坐标；

Y (2 字节) ：图片显示位置的 **Y** 坐标；

Image_1 (2 字节) ：第 **1** 张图的图位号；

Image_2 (2 字节) ：第 **2** 张图的图位号；

. .

. .

Image_n(2 字节) ：第 **n** 张图的图位号。

指令说明：该指令实现多张指定的图片，以指令所设定的时间按顺序切换。

参考实例：底层显示 FLASH 中的 1 号图作为背景，顶层(10,10)坐标处无限循环连续显示 10~13 号图。(10~13 号图为不同旋转角度的齿轮)

实例效果：如图 6.4.9.1，不同点是，此处的齿轮是转动的。在实际设计中，也可用 0x70 指令和 0x71 指令配合实现各种灵活的界面设计。

实例程序：

首先修改 void Continuous_Image 函数：

```
Void Continuous_Image(uchar Layer,uchar Mode,uchar Time,uint X,uint Y,uint Image_1,uint Image_2,uint Image_3,uint Image_4)
{
    SendChar(0xAA);
    ……(省略，与原函数相同)
    SendChar((Image_1>>8)&0xFF);
    SendChar(Image_1);
    SendChar((Image_2>>8)&0xFF);
    SendChar(Image_2);
    SendChar((Image_3>>8)&0xFF);        //添加第 3 张图的图位号
    SendChar(Image_3);
    SendChar((Image_4>>8)&0xFF);        //添加第 4 张图的图位号
    SendChar(Image_4)；
    ……(省略，与原函数相同)
}
实现功能：
{
    Display_Image(0,1,0,0);                    //底层显示 FLASH 中的 1 号图作为背景
    Continuous_Image(1,0xFF,20,10,10,10,11,12,13)；    //顶层(10,10)坐标处无限循环连续显示 10~13 号图
}
```

2.4.10 背光控制 (0x5E,0x5F)

2.4.10.1 关闭背光 (0x5E)

操作指令：**AA 5E CC 33 C3 3C**

指令说明：该指令用于关闭背光。

2.4.10.2 打开背光 (0x5F)

操作指令：**AA 5F (Pwm) CC 33 C3 3C**

参数说明：**Pwm (1 字节)** ：调节背光亮度，取值 **0~63 (** 即 **0x00~0x3F)**。**63** 时为最大亮度，上电默认为 **63**。如只需打开背光，不需要调节亮度，则指令中无需包含此字节。

指令说明：该指令用于开启背光或调节背光亮度。（部分模组含此功能）

2.4.11 参数配置 (0xE0)

操作指令：**AA E0 55 AA 5A A5 00 Bps 00 CC 33 C3 3C**
参数说明：**55 AA 5A A5 00** ：固定值；
Bps (1 字节**)** ：设置串口波特率，首次上电默认波特率为 **115200bps**。

Bps 值	0x03	0x04	0x05	0x06	0x07
波特率	9600	19200	38400	57600	115200

表 6.4.11

00 ：固定值。
指令说明：该指令用于波特率的配置，范围是 **9600~115200 bps** 。配置的参数保存在串口模块的存储 器中，掉电不会丢失。

2.4.12 触摸屏控制 (0xE4,0x73)

2.4.12.1 触摸屏开关 (0xE4)

操作指令：**AA E4 66 99 69 Touch CC 33 C3 3C**
参数说明：**66 99 69** ：固定值；
Touch (1 字节**)** ：
0xFF：开启触摸屏控制；
0x00：关闭触摸屏控制。
指令说明：该指令用于开启或关闭触摸屏控制。

2.4.12.2 触摸屏校准 (0xE4)

操作指令：**AA E4 55 AA 5A A5 CC 33 C3 3C**
设备返回：**AA F0**
参数说明：**55 AA 5A A5** ：固定值；
F0 ：固定值。
指令说明：串口智能模块接收到操作指令后，液晶屏四个角上依次出现蓝色十字标记，用户需要点击相应位置并保持几十 ms。一共需要有效点击四次。当四次点击完成后，模块返回 **0xAA+ 0xF0** 表示响应。发送该指令前需要先开启触摸屏控制功能。

2.4.12.3 触摸屏返回数据 (0x73)

用户操作：点击触摸屏
设备返回：**AA 73 X_pos Y_pos CC 33 C3 3C**
参数说明：**X_pos (2** 字节**)** ：被触发处的 **X** 坐标值；
Y_pos (2 字节**)** ：被触发处的 **Y** 坐标值。
指令说明：串口模块的触摸屏被有效按下后，模块将发送被触发处的坐标值。**X**、**Y** 坐 标值均为 **2** 字节，且高字节在前传送。

2.4.13 蜂鸣器控制 (0x79)

操作指令：**AA 79 Time CC 33 C3 3C**
参数说明：**Time (1 字节)** ：蜂鸣器发声长度=**Time*10ms** 。
指令说明：该指令用于控制蜂鸣器发声及发声时间。

2.4.14 键盘控制 (0x71)

用户操作：按下键盘按键。
设备返回：**AA 71 Key CC 33 C3 3C**
参数说明：**Key (1 字节)** ：表示 **4*4** 键盘的键值，**0x00~0xFF**。
指令说明：当键盘被按下后，串口模组回传键值。键值编码对应 **4*4** 键盘从左到右、从上到下从 **0x00 ~0xFF**。

附录 A：8051 串口驱动函数库（请点击超链接实现跳转）

```
/*******************************************************************************
*****************************      8051 串口驱动函数库      ************************
*******************************************************************************/
/*==============================================================================
函数名称: void MCU51_Init()
函数功能:初始化串口
入口参数:无
出口参数:无
==============================================================================*/
void MCU51_Init()
{
    ES = 1;         //开串口中断（带触摸屏需用）
    EA=1;    //开总中断（带触摸屏需用）
    TMOD|=0x20;      //定时器 T1 使用工作方式 2
    TH1=0xFA;        //设置初值
    TL1=0xFA;
    PCON|=0x80;        //SMOD=1；
    SCON|=0x50;        //工作方式 1，波特率 9600bit/s,允许接收
    TR1=1;          //开始计时
}
/*==============================================================================
函数名称:void SendChar(char Data)
函数功能:发送数据程序(一次发送一个字节)
入口参数:Data  要发送的数据
出口参数:无
==============================================================================*/
void SendChar(uchar Data)
```

```
{
    SBUF=Data;
    while(TI==0);
    TI=0;
}
/*==========================================================================================
函数名称:void SendAscIIStr(uchar *Str)
函数功能:ASCII 形式发送一个字符串（AscII 码形式）
入口参数:pet  要发送的数据
出口参数:无
==========================================================================================*/
void SendAscIIStr(uchar *Str)
{
    int i=0;
    while(Str[i]!='\0')
    {
        SendChar(Str[i]);
        i++;
    }
}
/*==========================================================================================
函数名:char HexChar(char c)
功能:ASCII 码转换成十六进制形式
入口参数:c   要转换的字符串
出口参数:无
==========================================================================================*/
char HexChar(char c)
{
    if((c>='0')&&(c<='9'))
```

```
        return c-0x30;
        else if((c>='A')&&(c<='F'))
        return c-'A'+10;
        else if((c>='a')&&(c<='f'))
        return c-'a'+10;
        else return 0x10;
}
```

附录 B：晶奥液晶显示模组驱动函数库（请点击超链接实现跳转）

```
/*****************************************************************************
*************************        晶奥液晶显示模组驱动函数库        ***********************
*****************************************************************************/
/*============================================================================
函数名称：void HandShake
函数功能：握手
入口参数：无
出口参数：AA version          version 为版本号
============================================================================*/
void HandShake()
{
    SendChar(0xAA);
    SendChar(0x00);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*============================================================================
函数名称：void Set_Color
函数功能：设置前景色和背景色
入口参数：F_col       前景色色值
          B_col       背景色色值
出口参数：无
============================================================================*/
void Set_Color(uint F_col,uint B_col)
{
    SendChar(0xAA);
    SendChar(0x40);
    SendChar((F_col>>8)&0xFF);
    SendChar(F_col&0xFF);
    SendChar((B_col>>8)&0xFF);
    SendChar(B_col&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
```

```
}
/*==========================================================================================
函数名称：void Set_Space
函数功能：设置字符间距
入口参数：Xdis          X 方向字符间距
          Ydis          Y 方向字符间距
出口参数：无
==========================================================================================*/
void Set_Space(uchar Xdis,uchar Ydis)
{
    SendChar(0xAA);
    SendChar(0x41);
    SendChar(Xdis);
    SendChar(Ydis);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================================
函数名称：void Write_Text_Normal
函数功能：标准方式字库显示
入口参数：Type          字库选择
            X           字符左上角 X 坐标
            Y           字符左上角 Y 坐标
          Str           需要显示的字符串
出口参数：  无
==========================================================================================*/
void Write_Text_Normal(uchar Type,uint X,uint Y,uchar *Str)
{
    SendChar(0xAA);
    SendChar(Type);
    SendChar((X>>8)&0xFF);
    SendChar(X&0xFF);
    SendChar((Y>>8)&0xFF);
    SendChar(Y&0xFF);
    SendAscIIStr(Str);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================================
函数名称：void Write_Text_Special
```

```
函数功能：特殊方式字库      显示,显示标准字库、大尺寸 ASCII 码字库和自定义字库
入口参数：    X             字符左上角 X 坐标
              Y             字符左上角 Y 坐标
          Lib_ID            字库选择
          C_Mode            前景色/背景色   显示控制位
           F_col            字符前景色
           B_col          字符背景色
             Str          需要显示的字符串
出口参数：无
================================================================================*/
void Write_Text_Special(uint X,uint Y,uchar Lib_ID,uchar C_Mode, uint F_col,uint B_col,uchar *Str)
{
    SendChar(0xAA);
    SendChar(0x98);
    SendChar((X>>8)&0xFF);
    SendChar(X&0xFF);
    SendChar((Y>>8)&0xFF);
    SendChar(Y&0xFF);
    SendChar(Lib_ID);
    SendChar(C_Mode);
    SendChar((F_col>>8)&0xFF);
    SendChar(F_col&0xFF);
    SendChar((B_col>>8)&0xFF);
    SendChar(B_col&0xFF);
    SendAscIIStr(Str);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*================================================================================
函数名称：void Draw_Dot
函数功能：画点
入口参数：Mode      前景色/背景色   画点方式选择（Mode 非 0 用前景色画点，Mode 为 0 用背景色画
点）
          X         所画点的 X 坐标
          Y         所画点的 Y 坐标
出口参数：无
================================================================================*/
void Draw_Dot(uchar Mode,uint X,uint Y)
{
    SendChar(0xAA);
    if(Mode)
        SendChar(0x51);              //Mode 非 0 用前景色画点
```

```
    else
        SendChar(0x50);                 //Mode 为 0 用背景色画点
    SendChar((X>>8)&0xFF);
    SendChar(X&0xFF);
    SendChar((Y>>8)&0xFF);
    SendChar(Y&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================
函数名称：void Draw_Line
函数功能：画线
入口参数：Mode      前景色/背景色   画线方式选择（Mode 非 0 用前景色画线，Mode 为 0 用背景色画线）
          X1        第 1 个点 X 坐标
          Y1        第 1 个点 Y 坐标
          X2        第 2 个点 X 坐标
          Y2        第 2 个点 Y 坐标
出口参数：无
==========================================================================*/
void Draw_Line(uchar Mode,uint X1,uint Y1,uint X2,uint Y2)
{
    SendChar(0xAA);
    if(Mode)
        SendChar(0x56);                 //Mode 非 0 用前景色画线
    else
        SendChar(0x5D);                 //Mode 为 0 用背景色画线
    SendChar((X1>>8)&0xFF);
    SendChar(X1&0xFF);
    SendChar((Y1>>8)&0xFF);
    SendChar(Y1&0xFF);
    SendChar((X2>>8)&0xFF);
    SendChar(X2&0xFF);
    SendChar((Y2>>8)&0xFF);
    SendChar(Y2&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================
函数名称：void Draw_Circle
```

```
函数功能：画圆
入口参数：Mode      前景色/背景色  画圆方式选择（Mode 非 0 用前景色画圆，Mode 为 0 用背景色画圆）
          X         圆心 X 坐标
          Y         圆心 Y 坐标
          R         圆的半径
出口参数：无
================================================================================*/
void Draw_Circle(uchar Mode,uint X,uint Y,uchar R)
{
    SendChar(0xAA);
    SendChar(0x57);
    if(Mode)
        SendChar(0x01);                 //Mode 非 0 用前景色画圆
    else
        SendChar(0x00);                 //Mode 为 0 用背景色画圆
    SendChar((X>>8)&0xFF);
    SendChar(X&0xFF);
    SendChar((Y>>8)&0xFF);
    SendChar(Y&0xFF);
    SendChar(R);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*================================================================================
函数名称：void Draw_Rectangle
函数功能：画矩形框
入口参数：Mode      前景色/背景色  画矩形框方式选择（Mode 非 0 用前景色画框，Mode 为 0 用背景色画框）
          X_sta     矩形区域左上角 X 坐标
          Y_sta     矩形区域左上角 Y 坐标
          X_end     矩形区域右下角 X 坐标
          Y_end     矩形区域右下角 Y 坐标
出口参数：无
================================================================================*/
void Draw_Rectangle(uchar Mode,uint X_sta,uint Y_sta,uint X_end,uint Y_end)
{
    SendChar(0xAA);
    if(Mode)
        SendChar(0x59);                 //Mode 非 0 用前景色画框
    else
        SendChar(0x69);                 //Mode 为 0 用背景色画框
```

```
    SendChar((X_sta>>8)&0xFF);
    SendChar(X_sta&0xFF);
    SendChar((Y_sta>>8)&0xFF);
    SendChar(Y_sta&0xFF);
    SendChar((X_end>>8)&0xFF);
    SendChar(X_end&0xFF);
    SendChar((Y_end>>8)&0xFF);
    SendChar(Y_end&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================
函数名称：void Fill_Rectangle
函数功能：填充矩形
入口参数：Mode      前景色/背景色   填充矩形方式选择（Mode 非 0 用前景色填充，Mode 为 0 用背景色
                    填充）
          X_sta     矩形区域左上角 X 坐标
          Y_sta     矩形区域左上角 Y 坐标
          X_end     矩形区域右下角 X 坐标
          Y_end     矩形区域右下角 Y 坐标
出口参数：无
==========================================================================*/
void Fill_Rectangle(uchar Mode,uint X_sta,uint Y_sta,uint X_end,uint Y_end)
{
    SendChar(0xAA);
    if(Mode)
        SendChar(0x5B);             //Mode 非 0 用前景色填充
    else
        SendChar(0x5A);             //Mode 为 0 用背景色填充
    SendChar((X_sta>>8)&0xFF);
    SendChar(X_sta&0xFF);
    SendChar((Y_sta>>8)&0xFF);
    SendChar(Y_sta&0xFF);
    SendChar((X_end>>8)&0xFF);
    SendChar(X_end&0xFF);
    SendChar((Y_end>>8)&0xFF);
    SendChar(Y_end&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
```

```
}
/*============================================================================
函数名称：void Clear_Screen
函数功能：全屏清屏（背景色）
入口参数：无
出口参数：无
============================================================================*/
void Clear_Screen()
{
    SendChar(0xAA);
    SendChar(0x52);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*============================================================================
函数名称：void Display_Image
函数功能：显示图片
入口参数：    Layer        顶层/底层   显示模式标识位(非 0 为顶层显示，0 为底层显示)
          Image_num      所显示图片的图位号
              X          图片显示位置的 X 坐标
              Y          图片显示位置的 Y 坐标
出口参数：无
============================================================================*/
void Display_Image(uchar Layer,uint Image_num,uint X,uint Y)
{
    SendChar(0xAA);
    if(Layer)
        SendChar(0x71);         //Layer 非 0 顶层显示图片
    else
        SendChar(0x70);         //Layer 为 0 底层显示图片
    SendChar((Image_num>>8)&0xFF);
    SendChar(Image_num&0xFF);
    SendChar((X>>8)&0xFF);
    SendChar(X&0xFF);
    SendChar((Y>>8)&0xFF);
    SendChar(Y&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*============================================================================
```

```
函数名称：void Display_CutImage
函数功能：显示剪切图片
入口参数：    Layer          显示层标识位(非 0 为顶层显示，0 为底层显示)
            Image_num      被剪切图片的图位号
              X_sta        剪切区域的左上角 X 坐标
              Y_sta        剪切区域的左上角 Y 坐标
              X_end        剪切区域的右下角 X 坐标
              Y_end        剪切区域的右下角 Y 坐标
              X_dis        剪切图片在屏幕上显示位置的左上角 X 坐标
              Y_dis        剪切图片在屏幕上显示位置的左上角 Y 坐标
出口参数：无
================================================================================*/
void Display_CutImage(uchar Layer,uint Image_num,uint X_sta,uint Y_sta,uint X_end,uint Y_end,uint X_dis,uint
Y_dis)
{
    SendChar(0xAA);
    SendChar(0x9B);
    if(Layer)
        SendChar(0xFF);         //Layer 非 0 顶层显示图片
    else
        SendChar(0x00);         //Layer 为 0 底层显示图片
    SendChar((Image_num>>8)&0xFF);
    SendChar(Image_num&0xFF);
    SendChar((X_sta>>8)&0xFF);
    SendChar(X_sta&0xFF);
    SendChar((Y_sta>>8)&0xFF);
    SendChar(Y_sta&0xFF);
    SendChar((X_end>>8)&0xFF);
    SendChar(X_end&0xFF);
    SendChar((Y_end>>8)&0xFF);
    SendChar(Y_end&0xFF);
    SendChar((X_dis>>8)&0xFF);
    SendChar(X_dis&0xFF);
    SendChar((Y_dis>>8)&0xFF);
    SendChar(Y_dis&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*================================================================================
函数名称：void Continuous_Image
函数功能：图片连续播放显示
入口参数：    Layer          显示层标识位(非 0 为顶层显示，0 为底层显示)
```

```
                Mode        连续循环次数，其中 0x00 表示停止当前循环 0xFF 表示无限循环显示
                Time        每次切换图片的时间间隔为 Timer*5ms，最小值为 40ms
                  X         图片显示位置的左上角 X 坐标
                  Y         图片显示位置的右下角 Y 坐标
              Image_1       第 1 张图片的图位号
              Image_2       第 2 张图片的图位号
出口参数：无
================================================================================================*/
void Continuous_Image(uchar Layer,uchar Mode,uchar Time,uint X,uint Y,uint Image_1,uint Image_2)
{
    SendChar(0xAA);
    SendChar(0x9C);
    if(Layer)
        SendChar(0xFF);         //Layer 非 0 顶层显示图片
    else
        SendChar(0x00);         //Layer 为 0 底层显示图片
    SendChar(Mode);
    SendChar(Time);
    SendChar((X >>8)&0xFF);
    SendChar(X&0xFF);
    SendChar((Y>>8)&0xFF);
    SendChar(Y&0xFF);
    SendChar((Image_1>>8)&0xFF);
    SendChar(Image_1&0xFF);
    SendChar((Image_2>>8)&0xFF);
    SendChar(Image_2&0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}

/*================================================================================================
函数名称：void Backlight_OFF
函数功能：关闭背光
入口参数：无
出口参数：无
================================================================================================*/
void Backlight_OFF()
{
    SendChar(0xAA);
    SendChar(0x5E);
    SendChar(0xCC);
    SendChar(0x33);
```

```
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================================
函数名称：void Backlight_ON
函数功能：打开背光
入口参数：无
出口参数：无
==========================================================================================*/
void Backlight_ON()
{
    SendChar(0xAA);
    SendChar(0x5F);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================================
函数名称：void Set_Backlight
函数功能：设置背光亮度
入口参数：Pwm        背光亮度值
出口参数：无
==========================================================================================*/
void Set_Backlight(uchar Pwm)
{
    SendChar(0xAA);
    SendChar(0x5F);
    SendChar(Pwm);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================================
函数名称：void Set_ComBps
函数功能：设置模组串口波特率
入口参数：Bps        波特率参数
出口参数：无
==========================================================================================*/
void Set_ComBps(uchar Bps)
{
    SendChar(0xAA);
    SendChar(0xE0);
```

```
    SendChar(0x55);
    SendChar(0xAA);
    SendChar(0x5A);
    SendChar(0xA5);
    SendChar(0x00);
    SendChar(Bps);
    SendChar(0x00);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================
函数名称：void Set_Touch
函数功能：触摸屏校准
入口参数：无
出口参数：AA   F0
==========================================================================*/
void Set_Touch()
{
    SendChar(0xAA);
    SendChar(0xE4);
    SendChar(0x55);
    SendChar(0xAA);
    SendChar(0x5A);
    SendChar(0xA5);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================
函数名称：void Touch_ON
函数功能：开启触摸屏
入口参数：无
出口参数：无
==========================================================================*/
void Touch_ON()
{
    SendChar(0xAA);
    SendChar(0xE4);
    SendChar(0x66);
    SendChar(0x99);
    SendChar(0x69);
```

```
    SendChar(0xFF);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================================
函数名称：void Touch_OFF
函数功能：关闭触摸屏
入口参数：无
出口参数：无
==========================================================================================*/
void Touch_OFF()
{
    SendChar(0xAA);
    SendChar(0xE4);
    SendChar(0x66);
    SendChar(0x99);
    SendChar(0x69);
    SendChar(0x00);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
/*==========================================================================================
函数名称：void ON_Buzzer
函数功能：控制蜂鸣器响
入口参数：Time        蜂鸣器发声时间（Time*10ms）
出口参数：无
==========================================================================================*/
void ON_Buzzer(uchar Time)
{
    SendChar(0xAA);
    SendChar(0x79);
    SendChar(Time);
    SendChar(0xCC);
    SendChar(0x33);
    SendChar(0xC3);
    SendChar(0x3C);
}
```

附录 C：自定义字库参照表（请点击超链接实现跳转）

LibID 文字大小 / 存储位置	8*16	12*24	16*32	20*40	24*48	28*56	32*64
0 存储区，1M大小	0x02	0x03	0x04	0x05	0x06	0x07	0x08
1 存储区，1M大小	0x12	0x13	0x14	0x15	0x16	0x17	0x18
2存储区（无）	无	无	无	无	无	无	无
3 存储区，128K大小	0x31	0x32	0x33	0x34	0x35	0x36	0x37
4 存储区，128K大小	0x41	0x42	0x43	0x44	0x45	0x46	0x47
5 存储区，128K大小	0x51	0x52	0x53	0x54	0x55	0x56	0x57
6 存储区，128K大小	0x61	0x62	0x63	0x64	0x65	0x66	0x67
7 存储区，128K大小	0x71	0x72	0x73	0x74	0x75	0x76	0x77
8 存储区，128K大小	0x81	0x82	0x83	0x84	0x85	0x86	0x87
9 存储区，128K大小	0x91	0x92	0x93	0x94	0x95	0x96	0x97
A 存储区，128K大小	0xA1	0xA2	0xA3	0xA4	0xA5	0xA6	0xA7
B 存储区，128K大小	0xB1	0xB2	0xB3	0xB4	0xB5	0xB6	0xB7
C 存储区，128K大小	0xC1	0xC2	0xC3	0xC4	0xC5	0xC6	0xC7
D 存储区，128K大小	0xD1	0xD2	0xD3	0xD4	0xD5	0xD6	0xD7
E 存储区，128K大小	0xE1	0xE2	0xE3	0xE4	0xE5	0xE6	0xE7
F 存储区，128K大小	0xF1	0xF2	0xF3	0xF4	0xF5	0xF6	0xF7
说明：LibID的高位表示不同存储区，低位表示字符大小。注意:同等大小的字库，1M大小存储区的低字节标志位比128K大小存储区的低字节大1。							

例：AA 98 00 64 00 64 37 C1 00 EF E8 00 30 31 32 33 34 35 36 37 38 39 CC 33 C3 3C。本条指令的 0x37 就表示显示 3 存储区的字体，该存储区的字体大小为 32*64。

另外，当以上存储区不足时，还可使用备用存储区，3E~FE 区，参考下表：

LibID 文字大小 / 存储位置	8*16	12*24	16*32	20*40	24*48	28*56	32*64
3E 存储区，128K大小	0x39	0x3A	0x3B	0x3C	0x3D	0x3E	0x3F
4E 存储区，128K大小	0x49	0x4A	0x4B	0x4C	0x4D	0x4E	0x4F
5E 存储区，128K大小	0x59	0x5A	0x5B	0x5C	0x5D	0x5E	0x5F
6E 存储区，128K大小	0x69	0x6A	0x6B	0x6C	0x6D	0x6E	0x6F
7E 存储区，128K大小	0x79	0x7A	0x7B	0x7C	0x7D	0x7E	0x7F
8E 存储区，128K大小	0x89	0x8A	0x8B	0x8C	0x8D	0x8E	0x8F
9E 存储区，128K大小	0x99	0x9A	0x9B	0x9C	0x9D	0x9E	0x9F
AE 存储区，128K大小	0xA9	0xAA	0xAB	0xAC	0xAD	0xAE	0xAF
BE 存储区，128K大小	0xB9	0xBA	0xBB	0xBC	0xBD	0xBE	0xBF
CE 存储区，128K大小	0xC9	0xCA	0xCB	0xCC	0xCD	0xCE	0xCF
DE 存储区，128K大小	0xD9	0xDA	0xDB	0xDC	0xDD	0xDE	0xDF
EE 存储区，128K大小	0xE9	0xEA	0xEB	0xEC	0xED	0xEE	0xEF
FE 存储区，128K大小	0xF9	0xFA	0xFB	0xFC	0xFD	0xFE	0xFF