



目 录

- 1 串口说明 3
 - 1.1 串口工作模式..... 3
 - 1.2 数据帧架构..... 3
 - 1.3 通信帧缓冲区（FIFO） 3
 - 1.4 字节传送顺序..... 3
 - 1.5 传送方向..... 3
- 2 指令速查表..... 4
- 3 指令集说明..... 6
 - 3.1 握手指令(0x00)..... 6
 - 3.2 设置当前调色板(0x40)..... 6
 - 3.3 设置字符显示间距(0x41)..... 6
 - 3.4 取指定位置颜色(0x42, 0x43)..... 6
 - 3.5 光标显示(0x44)..... 7
 - 3.6 文本显示(0x53, 0x54, 0x55, 0x6E, 0x6F, 0x98, 0x45)..... 7
 - 3.6.1 标准字库显示（0x53, 0x54, 0x55, 0x6E, 0x6F） 7
 - 3.6.2 选择字库显示（0x98） 8
 - 3.6.3 设置/取消文本框限制（0x45） 10
 - 3.7 点显示(0x50, 0x51, 0x74, 0x72)..... 11
 - 3.7.1 置点（0x50, 0x51） 11
 - 3.7.2 动态曲线显示（0x74）..... 11
 - 3.7.3 直接显存操作(0x72)..... 11
 - 3.8 连线显示（0x56, 0x5D, 0x75, 0x76,0x78）..... 12
 - 3.8.1 指定点连线（0x56, 0x5D） 12
 - 3.8.2 频谱显示（0x75） 12
 - 3.8.3 折线图显示（0x76） 12
 - 3.8.4 按照偏移量连线（0x78） 13
 - 3.9 圆弧曲线显示(0x57)..... 13
 - 3.9.1 圆弧或圆域显示（0x57） 13
 - 3.9.2 圆弧段显示（0x5704） 13
 - 3.10 区域显示..... 14
 - 3.10.1 矩形框或矩形区域显示(0x59, 0x69, 0x5A, 0x5B, 0x5C)..... 14
 - 3.10.2 区域填充(0x64)..... 14
 - 3.10.3 双色位图填充(0x73)..... 14
 - 3.11 全屏清屏（0x52） 15
 - 3.12 指定区域平移(0x60, 0x61, 0x62, 0x63)..... 15
 - 3.13 图片或图标显示（0x70, 0x71, 0x99, 0xE2, 0X7B, 0x9E） 15
 - 3.13.1 图片显示（0x70） 15
 - 3.13.2 显示一幅图片并计算 CRC 校验（0x7B） 15
 - 3.13.3 剪切图标显示(0x71、0x9C、0x9D)..... 16
 - 3.13.4 自定义图标显示（0x99） 16
 - 3.13.5 保存当前屏幕显示图片到 HMI 中（0xE2） 16
 - 3.13.6 保存当前屏幕显示图片区域到暂存缓冲区中（0xE9） 17
 - 3.13.7 把保存暂存缓冲区中的图片区域还原（0x7F） 17
 - 3.13.8 剪切图标旋转角度后显示(0x9E, 仅 H600 支持)..... 17
 - 3.14 背光亮度控制（0x5E, 0x5F） 18
 - 3.14.1 背光关闭（0x5E） 18
 - 3.14.2 设定触控（键控）背光模式（0x5E） 18
 - 3.14.3 打开背光到最大亮度（0x5F） 18
 - 3.14.4 调节背光亮度（0x5F） 18
 - 3.15 触摸屏操作（0x72, 0x73, 0x78, 0x79, 0xE4）..... 19
 - 3.15.1 触摸位置自动上传(0x72, 0x73)..... 19
 - 3.15.2 触摸键码自动上传(0x78, 0x79)..... 19
 - 3.15.3 进入触摸屏校准模式（0xE4） 19
 - 3.16 工作模式配置（0xE0） 20



3.17 指令定时循环执行 (0x9A)	21
3.17.1 开启指令定时循环执行功能	21
3.17.2 关闭指令定时循环执行功能	21
3.18 暂存缓冲区操作 (0xC0,0xC1,0xC2)	21
3.18.1 写暂存缓冲区 (0xC0)	21
3.18.2 读取暂存缓冲区内容 (0xC2)	21
3.18.3 使用暂存缓冲区数据置点 (0xC101)	21
3.18.4 使用暂存缓冲区数据连线 (0xC102)	21
3.18.5 使用暂存缓冲区数据显示折线图 (0xC103)	21
3.18.6 使用暂存缓冲区数据高速显示折线图 (0xC104)	21
3.18.7 使用暂存缓冲区数据缩放显示折线图 (0xC105)	23
3.18.8 使用暂存缓冲区数据缩放显示窗口限制双向折线图 (0xC106)	23
3.18.9 使用暂存缓冲区作为置点缓冲区 (0xC107)	23
3.18.10 使用暂存缓冲区来显示多参数 (0xC108)	24
3.18.11 使用暂存缓冲区来缓冲指令实现同步显示 (0xC110)	25
3.19 键盘操作(0x71, 0xE5)	25
3.19.1 键码上传 (0x71)	25
3.19.2 键码设置(0xE5)	25
3.20 用户存储器读写 (0x90, 0x91)	26
3.20.1 写随机数据存储器 (0x90 64KB)	26
3.20.2 写顺序数据存储器 (0x90 30MB)	26
3.20.3 读数据存储器 (0x91)	26
3.21 字库或配置文件下载 (0xF2)	26
3.22 简单算法支持 (0xB0)	26
3.22.1 拼音输入法 (0xB001、0xB004)	26
3.22.2 MAC 计算 (0xB002)	26
3.22.3 数组排序 (0xB003)	26
3.23 蜂鸣器控制 (0x79)	26
3.24 时钟 (RTC) 显示和读取 (0x9B, 0xE7)	27
3.24.1 关闭时钟显示	27
3.24.2 打开时钟显示	27
3.24.3 时钟调整	27
3.24.4 读取当前时钟 (公历)	27
3.24.5 读取当前时钟 (农历)	27
3.25 音乐播放 (0x30, 0x32, 0x33)	28
3.25.1 播放指定位置的音乐 (0x30)	28
3.25.2 音量调节 (0x32)	28
3.25.3 停止播放 (0x33)	28
3.26 配置文件的使用 (触控界面, 键控界面, 动画, 图标库)	29
3.26.1 触控界面自动切换(0x1E、0x1A 配置文件)	29
3.26.2 键控界面自动切换(0x1B 配置文件)	31
3.26.3 自动循环执行指令组 (0x1C 配置文件)	31
3.26.4 图标显示 (0x1D 配置文件)	31
3.27 HMI 和视频功能的切换 (0x7A)	33
3.28 强制刷新 1 次全屏显示 (0xD0)	33
3.29 使用触摸屏输入参数或文本 (0x7C, 仅 H600 支持)	34
3.29.1 输入纯 ASCII 字符串 (0x7C01)	34
3.29.2 输入中英文混合字符串 (0x7C02)	35
3.29.3 强制退出输入法状态 (0x7C00)	35
4 HMI 软件升级方法	36
5 修订记录	37

1 串口说明

1.1 串口工作模式

迪文科技所有标准HMI产品均采用异步、全双工串口（UART），串口模式为8n1，即每个数据传送采用10个位：1个起始位，8个数据位（低位在前传送，LSB），1个停止位。

- 上电时，如果终端的I/O0引脚为高电平或者浮空状态，串口波特率由用户预先设置，范围为1200-115200bps，具体设置方法参考0xE0指令。
- 上电时，如果终端的I/O0引脚为低电平，串口波特率固定在921600bps。

注：I/O0引脚状态，在迪文不同HMI上有2种不同的设置方式：

跳线插针设置，短路则I/O0为低电平，开路为高电平；

由USB口直接控制，接上USB口，I/O0为低电平，不接为高电平。

1.2 数据帧架构

迪文 HMI 的串口数据帧由 4 个数据块组成，如下表所述：

数据块	1	2	3	4
举 例	0xAA	0x70	0x01	0xCC 0x33 0xC3 0x3C
说 明	帧头，固定为 0xAA	指令	数据，最多 248 字节	帧结束符（帧尾）

1.3 通信帧缓冲区（FIFO）

迪文 HMI 有一个 24 帧的通信帧缓冲区，通信帧缓冲区为 FIFO（先进先出存储器）结构，只要通信缓冲区不溢出，用户可以连续传送数据给 HMI。

迪文 HMI 有一个硬件引脚（用户接口中的“BUSY 引脚”）指示了 FIFO 缓冲区的状态，正常时，BUSY 引脚为高电平（RS232 电平为负电压），当 FIFO 缓冲区只剩下一个帧缓冲区时，BUSY 引脚会立即变成低电平（RS232 电平为正电压）。

对于一般的应用，由于迪文 HMI 的处理速度很快，用户用不着判断 BUSY 信号状态（比如迪文的图片下发软件就没有判断 BUSY 信号的状态）。

但对于短时间需要传送多个数据帧的应用，比如一次需要高速刷新上百个屏幕参数，建议客户使用 BUSY 信号来控制串口发送，当 BUSY 信号为低电平时，就不要发送数据给 HMI。

如果用户使用 HMI 过程中，出现“丢帧”现象，即某些数据没有显示出来，可能就是缓冲区溢出了，这时需要用示波器检查 BUSY 信号是否有跳变，如果有跳变，则需要减慢发送速度，或者增加硬件检测 BUSY 信号判忙处理。

1.4 字节传送顺序

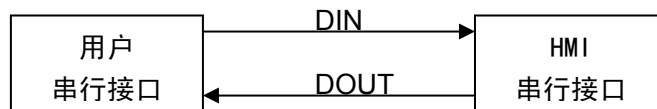
迪文 HMI 的所有指令或者数据都是 16 进制（HEX）格式；对于字型（2 字节）数据，总是采用高字节先传送（MSB）方式。

比如，x 坐标为 100，其 HEX 格式数据为 0x0064，传送给 HMI 时，传送顺序为 0x00 0x64。

1.5 传送方向

在迪文 HMI 上，传送方向按照下面的规则定义：

- 下行（Tx） 用户发送数据给 HMI，数据从 HMI 用户接口的“DIN 引脚”输入；
- 上行（Rx） HMI 发送数据给用户，数据从 HMI 用户接口的“DOUT 引脚”输出。



2 指令速查表

类别	指令	指令参数	说明
握手	0x00	无	查看配置和版本信息
显示参数配置	0x40	Fcolor+Bcolor	设置调色板
	0x41	D_X (0x00-0x7F)+D_Y (0x00-0x7F)	设置字符显示间距
	0x42	X+Y	取色到背景色调色板
	0x43	X+Y	取色到前景色调色板
	0x44	Mode+X+Y+Wide (0x01-0x1F)+Height (0x01-0x1F)	设置光标显示模式
文本显示	0x53	X+Y+String	8×8 点阵 ASCII 字符
	0x54		16×16 点阵 GBK 扩展码字符串显示
	0x55		32×32 点阵 GB2312 内码字符串显示
	0x6E		12×12 点阵 GBK 扩展码字符串显示
	0x6F		24×24 点阵 GB2312 内码字符串显示
	0x98	X+Y+Lib_ID+C_mode+C_dot+Fcolor+Bcolor+String	任意点阵, 任意编码字符串显示
0x45	(Xs, Ys, Xe, Ye) /0x00	开启/关闭文本框限制	
置点	0x50	(x, y) ₀ +(x, y) ₁ +……+(x, y) _n	背景色置多个点 (删除点)
	0x51		前景色置多个点
	0x74	X+Ys+Ye+Bcolor+(y, Fcolor) ₀ +……+(y, Fcolor) _n	动态曲线快速置点
	0x72	Address(H:M:L)+Data_word ₀ +……+Data_word _n	直接显存操作
线段和多边形	0x56	(x, y) ₀ +(x, y) ₁ +……+(x, y) _n	把指定点用前景色线段连接 (显示多边形)
	0x5D		把指定点用背景色线段连接 (删除多边形)
	0x75	X+Y+Height_max+Height ₀ +Height ₁ +……+Height _n	快速显示连续的同底垂直线段 (频谱)
	0x76	X+X_dis (0x00-0xFF)+Y ₀ +Y ₁ +……+Y _n	快速显示折线图 (X _i =X+i*X_dis, Y _i = Y _i)
	0x78	X+Y+(D_X0, D_Y0)+……+(D_Xi, D_Yi)	偏移量连线
圆弧和圆域	0x57	(Type, x, y, r) ₀ +(Type, x, y, r) ₁ +……+(Type, x, y, r) _n	反色/显示 多个圆弧或圆域
矩形框	0x59	(Xs, Ys, Xe, Ye) ₀ +(Xs, Ys, Xe, Ye) ₁ +……+(Xs, Ys, Xe, Ye) _n	前景色显示多个矩形框 (显示矩形框)
	0x69		背景色显示多个矩形框 (删除矩形框)
区域操作	0x73	Color0+Color1+(Xs, Ys, Xe, Ye)+(X, Y)+Data	双色位图填充指定区域
	0x64	X+Y+Color	指定区域填充
	0x52	无	清屏
	0x5A	(Xs, Ys, Xe, Ye) ₀ +(Xs, Ys, Xe, Ye) ₁ +……+(Xs, Ys, Xe, Ye) _n	多个指定区域清除
	0x5B		多个指定区域填充
	0x5C		多个指定区域反色
	0x60		多个指定区域左环移
	0x61		多个指定区域右环移
	0x62		多个指定区域左移
0x63	多个指定区域右移		
图片/图标显示	0x70		Picture_ID
	0x7B	Picture_ID	显示一幅全屏图像并计算 CRC 校验和
	0x71	Picture_ID+Xs+Ys+Xe+Ye+X+Y	从保存在终端的一幅图片剪切一部分显示 (背景显示)
	0x9C	Picture_ID+Xs+Ys+Xe+Ye+X+Y	从保存在终端的一幅图片剪切一部分显示 (背景不显示), 自动恢复当前图片背景。
	0x9D	Picture_ID+Xs+Ys+Xe+Ye+X+Y	从保存在终端的一幅图片剪切一部分显示 (背景不显示)
	0x9E	Mode+Pic_ID+Xs+Ys+Xe+Ye+Xc+Yc+AL+Xc1+Yc1	从保存在终端的一幅图片剪切一部分, 旋转一个指定角度后粘贴到当前页面显示。
	0xE2	Picture_ID	将当前显示画面保存到终端中
	0x99	(x, y, Icon_ID) ₀ +……+(x, y, Icon_ID) _n /无	用户自定义图标显示
	0xD0	无	强制刷新一次全屏显示
动画支持	0x9A	0xFF/Pack_ID	关闭/打开自动执行用户预先设置的指令组
暂存缓冲区操作	0xC0	Address(H:L)+Data_word ₀ +……+Data_word _n	写数据到暂存缓冲区
	0xC1	0x01+Address+Pixel_Number(H:L)	显示暂存缓冲区预置的数据点
		0x02+Address+Line_Number(H:L)	显示暂存缓冲区预置的数据线
		0x03+Address+X+Y+Line_Number+D_x+Dis_x+K_y+Color	使用暂存缓冲区的数据点连线 (曲线动态缩放)
		0x04+Adr1+X+Y+Line_Number+0x01+Dis_x+Color1+Adr0+Color0	使用暂存缓冲区的数据点高速无闪烁连线 (示波器)
		0x05+Address+x+y+Line_Number+D_x+Dis_x+M_y+D_y+Color	使用暂存缓冲区数据缩放显示折线图



		0x06+Address+x+y+Line_Number+D_x+Dis_x+M_y+D_y+Color+Ymin+Ymax	使用暂存缓冲区数据缩放显示窗口限制双向折线图
		0x0700+Address+Xlen+Ylen+Xs+Ys	清空置点缓冲区
		0x0701+Address+Xlen+Ylen+Xs+Ys+Color+Mode+(Xi, Yi)	在置点缓冲区置点
		0x0702+Address+Xlen+Ylen+Xs+Ys+Color	恢复置点缓冲区显示到当前显示页面
		0x10+Address+Frame_Number	使用暂存缓冲区缓冲指令实现同步显示
		0x08+Address+<Parameter_N>	使用暂存缓冲区进行多参数显示
	0xC2	<Address> +<Data_Length>	从暂存缓冲区回读数据
数据库操作	0xF2	0xF2+0xF2+0x5A+0xA5+Lib_ID	修改字库
	0x90	0x55+0xAA+0x5A+0xA5+Address(H:MH:ML:L)+Data	写数据到用户数据库 (32MB)
	0x91	Address+Read_Length(H:L)	从用户数据库读数据 (32MB)
键盘操作	0x71	K_code	键码上传
	0xE5	0x55+0xAA+0x5A+0xA5+K_Code ₀ +.....+K_code ₆₃	配置键码接口
触摸屏操作	0x72	Touch_X+Touch_Y	触摸屏松开后, 最后一次数据上传 (可 0xE0 指令设置关闭)
	0x73		触摸屏按下时, 数据上传 (可 0xE0 指令设置只传 1 次)
	0xE4	0x55+0xAA+0x5A+0xA5	触摸屏校准
	0x78	Touch_Code	触控界面自动切换模式下, 预设键码自动上传。
	0x79		
	0x7C	01 R_ID VP_ID X Y Str_Max_Num Str_Scale Str_Color	触摸屏输入 ASCII 字符串
		02 R_ID VP_ID X Y Str_Max_Num Str_Scale Str_Color T_Color Tx Ty	触摸屏输入中英文混合字符串
蜂鸣器控制	0x79	BZ_time	蜂鸣器鸣叫一声 (10×Bz_time mS)
视频切换	0x7A	Work_Mode+Video_Mode+Video_CH	切换视频和 HMI 功能
背光控制	0x5E	无或 0x55+0xAA+0x5A+0xA5+V_ON+V_OFF+ON_TIME	关闭背光或设置触控 (键控) 背光模式
	0x5F	无或 PWM_T (0x00-0x3F)	打开背光或 PWM 方式调节背光亮度的
时钟操作	0x9B	0x5A、0x5B (读取) /0x00(关闭)/0xFF+M+TM+Color+X+Y (打开)	启用/关闭时钟自动叠加显示; 读取当前时钟
	0xE7	0x55+0xAA+0x5A+0xA5+YY:MM:DD:HH:MM:SS	设置时钟
参数配置	0xE0	0x55+0xAA+0x5A+0xA5+Panel_Set+Bode_Set+Para1 或 0x55+0xAA+0x5A+0xA5+Panel_Set+Bode_Set+Para1+Para2	配置用户串口速率、触摸屏数据上传格式、背光控制模式, 显示模式。 V6.0 以上版本, 0xE0 指令修改参数掉电不保存。
实用算法	0xB0	下发:0x01+PY_Code 应答:0x01+HZ_num+String	基于一级字库的拼音输入法
		下发:0x02+A+B+C+D 应答:0x02+E+F	计算(A×B+C)/D, E 是 4 字节商, F 是 2 字节余数
		下发:0x03+Data_Pack0 应答:0x03+Data_Pack1	无符号整数 (2 字节) 数组排序
		下发:0x04+PY_Code 应答:0x04+HZ_num+String	基于 GBK 字库的拼音输入法
声音操作	0x30	Start_Seg+Play_number+Play_time	播放指定存储位置的音乐
	0x32	Volume_L+Volume_R+0x00	实时音量调节
	0x33	0x55+0xAA+0x5A	立即停止播放
	0x3F	'OK'	声音操作指令应答
配置文件操作 (简易 OS)		Pic_Now+(x _s , y _s , x _e , y _e)+P_next+P_cut+Touch_Code	触控界面自动切换 (0x1E 字库文件)
		Pic_Now+0x00:K_Code+Pnext+(Pcut, x _s , y _s , x _e , y _e , x, y)+Touch_Code	键控界面的自动切换 (0x1B 字库文件)
		Delay+Length+Command	自动指令播放 (0x1C 字库文件)
		Pic_ID+(x _s , y _s , x _e , y _e)	图标字符定义 (0x1D 字库文件)
		Command_Length+Command_String	用户预设的触控上传指令 (0x1A 字库文件)
软件升级		DWIN_M600_BOOT!	串口在线升级内核软件

3 指令集说明

3.1 握手指令 (0x00)

Tx: AA 00 CC 33 C3 3C

Rx: AA 00 'OK_V*.*' P1 P2 P3 Pic_ID CC 33 C3 3C

或者

Tx: AA 00 00 CC 33 C3 3C

Rx: AA 00 'OK_V*.*' P1 P2 P3 P4 Pic_ID CC 33 C3 3C

- OK_V*.*, *.*是 HMI 的当前软件版本;
- P1 是当前 HMI 所使用的显示屏配置模式 (具体参数请参考 0xE0 指令);
- P2 是当前用户所设置的串口波特率;
- P3 是触摸屏、蜂鸣器、显示配置模式 1;
- P4 是显示配置模式 2;
- Pic_ID 是当前显示图片的 ID;

迪文 HMI 上电初始化需要 0.5-2 秒左右的时间 (取决于用户的电源容量和上电速率), 在上电初始化未完成之前, 不会响应用户指令。用户可以通过发送握手指令来确认 HMI 是否已经上电初始化完成。

3.2 设置当前调色板 (0x40)

TX: AA 40 <FC> <BC> CC 33 C3 3C

Rx: 无

- <FC> 前景色调色板, 2 字节 (16 bit, 65K color), 复位默认值是 0xFFFF (白色)。
- <BC> 背景色调色板, 2 字节 (16 bit, 65K color), 复位默认值是 0x001F (蓝色)。
- 16bit 调色板定义是 5R6G5B 模式, 如下表所叙:

16bit 调色板位定义																	
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Define	R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0	
	红色 0xF800					绿色 0x07E0						蓝色 0x001F					



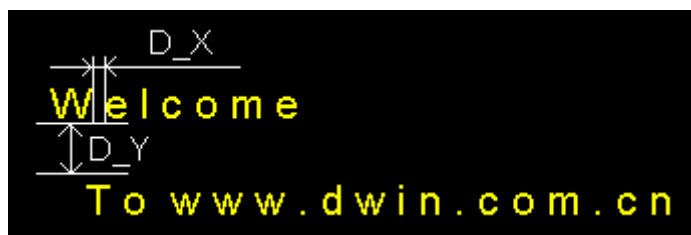
一旦设定好, 除非重新设定, 就会一直保持下来, 直到 HMI 硬件复位后恢复默认值。

3.3 设置字符显示间距 (0x41)

TX: AA 41 <D_X> <D_Y> CC 33 C3 3C

Rx: 无

- <D_X> X 方向的字符间距 (列间距), 取值范围 0x00-0x7F, 复位默认值是 0x00。
- <D_Y> Y 方向的字符间距 (行间距), 取值范围 0x00-0x7F, 复位默认值是 0x00。



一旦设定好, 除非重新设定, 就会一直保持下来, 直到 HMI 硬件复位后恢复默认值。

3.4 取指定位置颜色 (0x42, 0x43)

Tx: AA <CMD> <X> <Y> CC 33 C3 3C

Rx: 无

- <CMD> 0x42 为取指定位置颜色到背景色调色板; 0x43 为取指定位置颜色到前景色调色板。
- <X> <Y> 指定位置的坐标(迪文 HMI, 坐标均为 2 字节表示)。

举例:

```
AA 42 00 10 01 00 CC 33 C3 3C
```

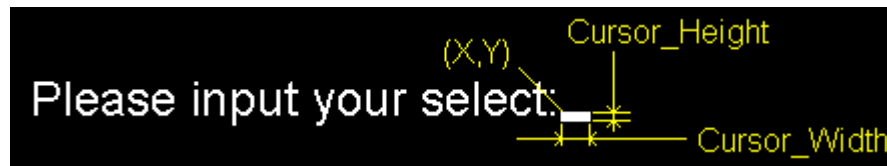
取 x=16 (0x0010) y=256 (0x0100) 位置的颜色到背景色调色板。

3.5 光标显示(0x44)

Tx: AA 44 <Cursor_EN> <X> <Y> <Cursor_Width> <Cursor_Height> CC 33 C3 3C

Rx: 无

- <Cursor_EN>
 - 0x01 光标显示打开, 光标将在(x, y)位置显示;
 - 0x00 光标显示关闭。
- <X>, <Y> 是光标左上角的坐标位置;
- <Cursor_Width> 是显示光标的宽度, 取值范围 0x01-0x1F;
- <Cursor_Height> 是显示光标的高度, 取值范围 0x01-0x01F。



当禁止光标显示时 (*Cursor_EN=0x00*), 指令中的其它参数没有意义。

举例:

```
AA 44 01 00 80 00 60 10 03 CC 33 C3 3C
```

在(128,96)位置, 打开一个 16×3 点阵的光标显示。

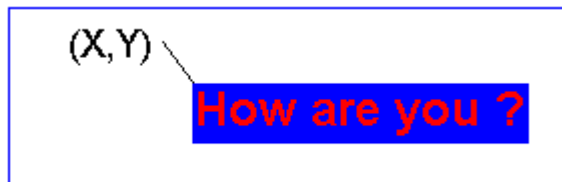
3.6 文本显示(0x53, 0x54, 0x55, 0x6E, 0x6F, 0x98, 0x45)

3.6.1 标准字库显示 (0x53, 0x54, 0x55, 0x6E, 0x6F)

Tx: AA <CMD> <X> <Y> <String> CC 33 C3 3C

Rx: 无

- <CMD>
 - 0x53: 显示 8*8 点阵 ASCII 字符串;
 - 0x54: 显示 16*16 点阵的扩展码汉字字符串 (ASCII 字符以半角 8*16 点阵显示);
 - 0x55: 显示 32*32 点阵的内码汉字字符串 (ASCII 字符以半角 16*32 点阵显示);
 - 0x6E: 显示 12*12 点阵的扩展码汉字字符串 (ASCII 字符以半角 6*12 点阵显示);
 - 0x6F: 显示 24*24 点阵的内码汉字字符串 (ASCII 字符以半角 12*24 点阵显示);
- <X> <Y> 显示字符串的起始位置 (第一个字符左上角坐标位置)
- <String> 要显示的字符串, 汉字采用 GB2312 (0x55、0x6F; 内码) 或者 GBK (0x54、0x6E, 内码扩展) 编码, 显示颜色由 0x40 指令设定, 显示字符间距由 0x41 指令设置, 遇到行末会自动换行。0x0D、0x0A 被处理成“回车和换行”。



举例:

```
AA 55 00 80 00 30 48 6F 77 20 61 72 65 20 79 6F 75 20 3F CC 33 C3 3C
```

从 (128, 48) 位置开始显示字符串 “How are you?”。

3.6.2 选择字库显示 (0x98)

Tx: AA 98 <X> <Y> <Lib_ID> <C_Mode> <C_dots> <Fcolor> <Bcolor> <String> CC 33 C3 3C

Rx: 无

- > <X> <Y> 显示字符串第一个字符的左上角指标;
- > <Lib_ID> 字库选择, 取值范围0x00-0x3B, 对应0xF2指令保存的字库位置;

HMI 内核有 32MB 字库存储器, 被分割成 60 个大小不同的字库, Lib_ID 定义如下:

Lib_ID	容量	说明	出厂默认值
0x00-0x1F	128KB	32 个最大 128KB 容量的小字库, 一般用来设计用户需要的特殊图标或不同字体的 ASCII 字符显示。	0x00=ASCII 字符库, 请不要修改 0x02-0x1F: 空
0x20-0x3B	1MB	28 个最大 1MB 容量的字库。 <ul style="list-style-type: none"> ● 单个字库可以装下 16 点阵以内的 GBK 扩展字库 (12×12 或 16×16 点阵), 或者 32 点阵以内的 GB2312 二级字库(12×12、16×16、24×24、32×32); ● 字库允许组合使用, 最大可以拼接成一个 28MB 的特大点阵字库。字库组合使用时, 0x98 或 0xF2 指令中的 Lib_ID 是指首字库地址; 比如一个 32 点阵的 UNICODE 编码字库, 将占用 8MB 的字库空间, 我们可以把 Lib_ID=0x20-0x27 的空间分配给它, 下一个字库将从 0x28 开始; 使用 0x98 指令显示时, Lib_ID=0x20。 	0x20=12 点阵 GBK 宋体 0x21=16 点阵 GBK 宋体 0x22=24 点阵 GB2312 宋体 0x23=32 点阵 GB2312 宋体 0x24-0x3B: 空

- > <C_Mode> 选择文本显示模式以及编码方式, 如下表所述:

位	.7-.4	.3-.0
定义	显示模式	字符编码方式
说明	.7=1 文本前景色显示 =0 文本前景色不显示	0x00 8bit 编码 (字库中最多只有 256 个字符)
	.6=1 文本背景色显示 =0 文本背景色不显示	0x01 GB2312 中文内码
	.5=1 文本纵向显示 =0 文本横向显示	0x02 GBK 中文扩展内码或韩文 HANGUL 编码
	.4=1 自动调整字符 X 间距 =0 按照设定点阵大小显示	0x03 BIG5 繁体中文编码
		0x04 SJIS 日文编码
		0x05 UNICODE UNICODE 编码 (UTF16)
		0x06-0x0F: 未定义

说明: C_Mode.4=1 启用时, 字符必须是 8bit 编码, 并且下载的字库需要旋转 90° (TS3, 纵向模式 1)



显示模式设置举例:

前景色	ON	ON	OFF
背景色	ON	OFF	ON
<C_mode>高 2bit 值	0xC*	0x8*	0x4*
字符显示效果	上图 B 区域	上图 A 区域	上图 C 区域

➤ <C_dots> 显示字符大小设置

C_Dots	字库类型 (C_Mode 设置字节的低 4bit)		
	0x00 或 0x05	0x01-0x04	
		ASCII 字符	非 ASCII 字符
0x00	8*8	6*12	12*12
0x01	6*12	8*16	16*16
0x02	8*16	12*24	24*24
0x03	12*24	16*32	32*32
0x04	16*32	20*40	40*40
0x05	20*40	24*48	48*48
0x06	24*48	28*56	56*56
0x07	28*58	32*64	64*64
0x08	32*64	—	40*80
0x09	12*12	—	48*96
0x0A	16*16	—	56*112
0x0B	24*24	—	64*128
0x0C	32*32	—	80*80
0x0D	40*40	—	96*96
0x0E	48*48	—	112*112
0x0F	56*56	—	128*128
0x10	64*64	—	—
0x11	40*80	—	—
0x12	48*96	—	—
0x13	56*112	—	—
0x14	64*128	—	—
0x15	80*80	—	—
0x16	96*96	—	—
0x17	112*112	—	—
0x18	128*128	—	—
0x19	6*8	—	—
0x1A	8*10	—	—
0x1B	8*12	—	—
0x1C	100*200	—	—
0x1D	200*200	—	—
0x1E	64*48	—	—

比如, 如果C_Mode=0x*1 (内码编码), 则C_dots=0x07将显示64*64的中文字符和32*64的ASCII字符。

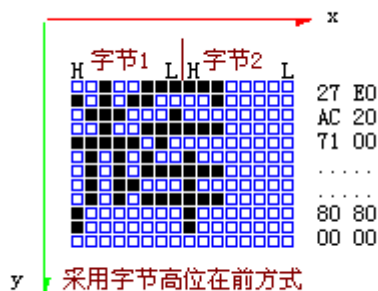
- <Fcolor> 字符显示的前景色;
- <Bcolor> 字符显示的背景色;
- <String> 字符串数据, 显示间隔由 0x41 指令设置, 遇到行末会自动换行。

举例:

AA 98 00 80 00 30 23 C1 03 F8 00 00 1F B1 B1 BE A9 B5 CF CE C4 BF C6 BC BC D3 D0 CF DE B9 AB CB BE CC 33 C3 3C

如果用户使用 0x98 指令, 要使用自己设计的字库来显示 ASCII 字符 (或其它 8bit 编码的字库), 不能使用 0x01-0x04 的编码方法 (GB2312、GBK、BIG5、SJIS), 在 0x01-0x04 编码方式下, HMI 会自动使用 0x00 字库来显示 ASCII 字符。关于 0x00 字库的修改方式, 请咨询迪文技术支持。

字库中, 所有文件均采用 x 方向, 高位在先 (MSB) 的扫描存储方式, 如下图所示:



推荐使用点阵字库提取软件 TS3 来生成字库。



3.6.3 设置/取消文本框限制（0x45）

设置文本框限制

Tx: AA 45 <Xs> <Ys> <Xe> <Ye> CC 33 C3 3C

Rx: 无

➤ <Xs> <Ys> <Xe> <Ye> 文本框位置;

设置文本框位置后，文本显示时将在文本框限制范围内自动换行。

取消文本框限制

Tx: AA 45 00 CC 33 C3 3C

Rx: 无

取消文本框限制后，文本显示将在全屏范围内自动换行。

3.7 点显示 (0x50, 0x51, 0x74, 0x72)

3.7.1 置点 (0x50, 0x51)

Tx: AA <CMD> <(x0,y0) (x1,y1)(xi, yi)> CC 33 C3 3C

Rx: 无

➤ <CMD>

0x50: 背景色显示点 (删除点);

0x51: 前景色显示点 (置点)

➤ <(x0,y0) (x1,y1)(xi, yi)> 要显示的点坐标, 一帧串口数据最多显示 62 个点。

举例:

AA 51 00 00 00 00 00 03 00 06 00 05 00 20 CC 33 C3 3C

以前景色显示 3 个点, 坐标位置为 (0, 0), (3, 6), (5, 32), 显示结果如下:



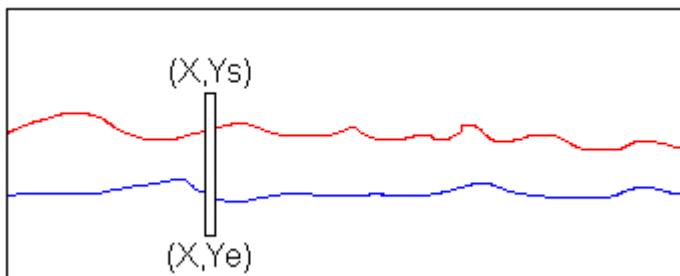
3.7.2 动态曲线显示 (0x74)

Tx: AA 74 <X> <Ys> <Ye> <Bcolor> <(Y0,Fcolor0), (Y1,Fcolor1).....(Yi, Fcolori)> CC 33 C3 3C

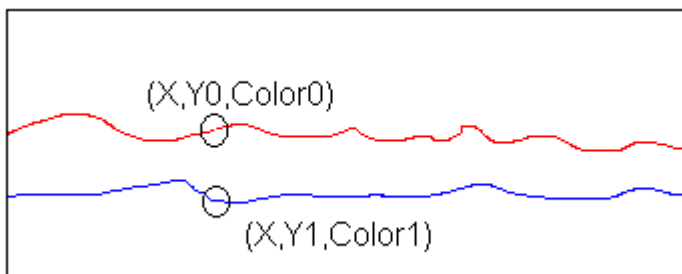
Rx: 无

本条指令主要用来方便用户在一个视窗中快速显示多条变化 (动态) 的曲线, 终端按照下面的顺序来处理指令:

第 1 步: 用 <Bcolor> 颜色擦除从 (X, Ys) 到 (X, Ye) 的垂直线, 把原来的显示内容清空;



第 2 步: 在 (X, Yi) 位置用 <Fcolori> 颜色置点。



本指令并不会改变用户预设的调色板属性!

3.7.3 直接显存操作 (0x72)

Tx: AA 72 <Address_H:M:L> <Pixel_data0.....Pixel_datan> CC 33 C3 3C

Rx: 无

主要用来下载图片到 HMI, 用户一般不需要使用。

3.8 连线显示 (0x56, 0x5D, 0x75, 0x76, 0x78)

3.8.1 指定点连线 (0x56, 0x5D)

Tx: AA <CMD> <(x0,y0) (x1,y1)(xi, yi)> CC 33 C3 3C

Rx: 无

➤ <CMD>

0x56: 用前景色 (0x40 指令设置) 把指定点用线段连接;

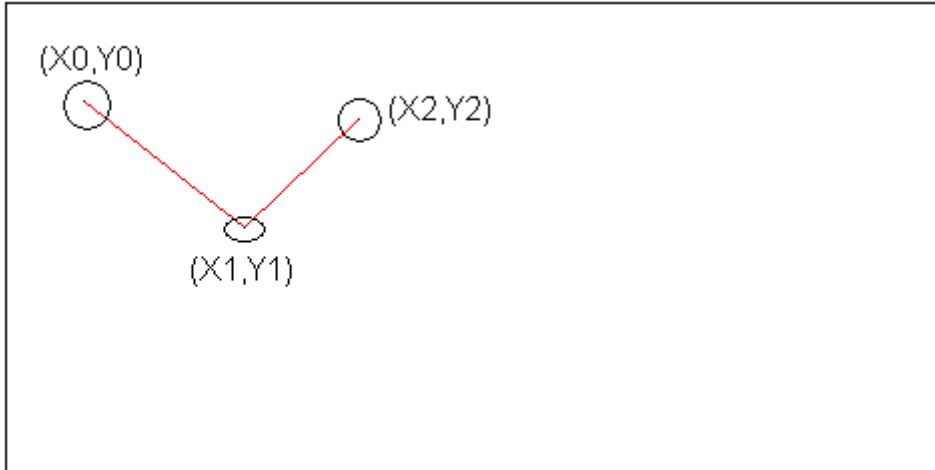
0x5D: 用背景色 (0x40 指令设置) 把指定点用线段连接;

➤ <(x0,y0) (x1,y1)(xi, yi)> 是连线点的坐标。

举例:

AA 56 00 28 00 32 00 78 00 70 00 B1 00 3A CC 33 C3 3C

用前景色把 3 个点 (40, 50), (120, 112), (177, 58) 连线, 显示结果如下。



3.8.2 频谱显示 (0x75)

Tx: AA 75 <(x, y)>, <H_max>, <H0.....Hi> CC 33 C3 3C

Rx: 无

➤ <(x, y)> x 为频谱的 X 轴起点坐标, 每显示一根谱线后, x=x+1; y 为频谱的水平基准位置, 每根谱线的 Y 轴起始和终止坐标分别为 y 和 (y-Hi)。

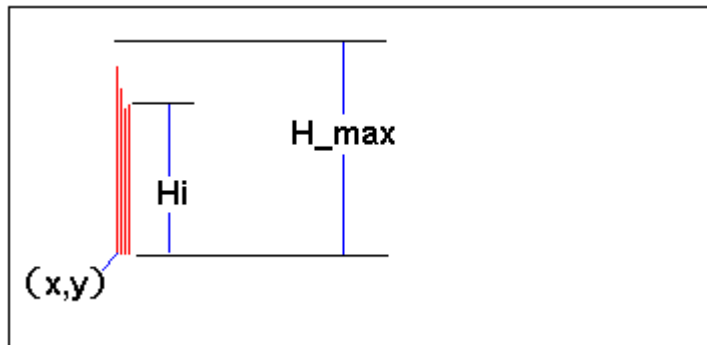
➤ <H_max> 谱线的最大高度,

如果 H_max=0x01-0xFF, 则谱线高度 Hi 也是 1 字节的变量;

如果 H_max=0x00, 则后续两字节为 Hmax, Hi 为两字节的变量。

➤ <H0.....Hi> 是单根谱线的高度, 1 字节或者两字节。

显示谱线颜色由 0x40 调色板设定, 显示谱线时, 谱线 (Hi 高度) 会以前景色显示, 空余谱线 (H_max-Hi) 会以背景色 (擦除) 显示。



3.8.3 折线图显示 (0x76)

Tx: AA 76 <x>, <x_dis>, <Y0.....Yi> CC 33 C3 3C

Rx: 无

➤ <x> 折线图的 X 轴起点坐标, 每连线一点后, x=x+x_dis;

➤ <x_dis> x 坐标的增量;

➤ <Y0.....Yi> 折线图的顶点坐标, 使用前景色连线显示。

本指令的功能同 0x56 基本相似, 只是 X 坐标为 HMI 自动计算, 提高了连线速度。

3.8.4 按照偏移量连线 (0x78)

Tx: AA 78 <x, y>, <dx0, dy0>, <dx1, dy1>, …… , <dxn, dyn> CC 33 C3 3C

Rx: 无

- <x, y> 连线的起点坐标;
- <dxn, dyn> 1字节的 x, y 偏移量, 最高位 (.7) 为符号位, “1” 表示负;

3.9 圆弧曲线显示 (0x57)

3.9.1 圆弧或圆域显示 (0x57)

Tx: AA 57 (<Type_0> <X_0> <Y_0> <R_0>) …… (<Type_i> <X_i> <Y_i> <R_i>) CC 33 C3 3C

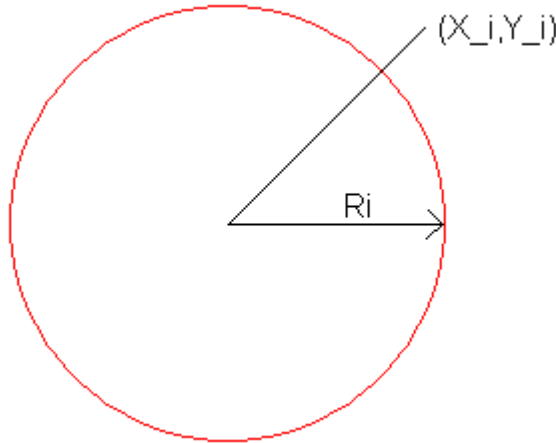
Rx: 无

- <Type_i> 格式控制
 - 0x00 把指定的圆弧反色显示;
 - 0x01 前景色显示 (0x40 指令设定) 指定的圆弧;
 - 0x02 把指定的圆域反色显示;
 - 0x03 用前景色 (0x40 指令设定) 填充指定的圆域。
- <X_i> <Y_i> 圆弧或圆域的圆心坐标;
- <R_i> 圆弧或圆心的半径, 0x01-0xFF

举例:

AA 57 01 00 60 00 60 40 CC 33 C3 3C

用前景色显示一个圆弧, 圆心是 (96, 96), 半径是 64, 显示结果如下图所示:



3.9.2 圆弧段显示 (0x5704)

Tx: AA 57 04 <X> <Y> <R> <A_S> <A_E> CC 33 C3 3C

Rx: 无

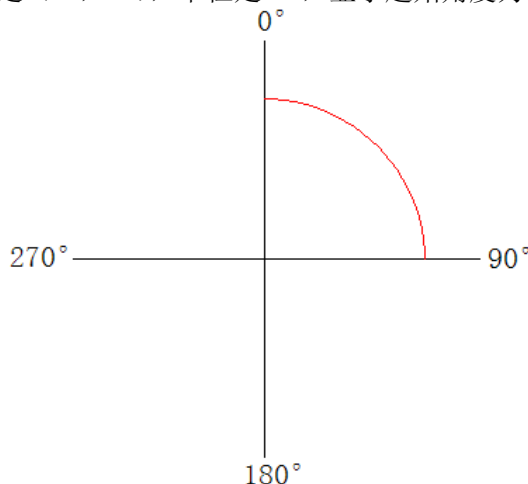
- <X> <Y> 圆弧的圆心坐标;
- <R> 圆弧半径, 0x0001-0x03FF
- <A_S> 圆弧起始角度, 0x00-0x02D0 (0-720), 单位为 “0.5° ”
- <A_E> 圆弧结束角度, 0x00-0x02D0 (0-720), 单位为 “0.5° ”

显示颜色为 0x40 指令设定的前景色。

举例:

AA 57 04 00 60 00 60 00 40 00 00 00 B4 CC 33 C3 3C

用前景色显示一个圆弧, 圆心是 (96, 96), 半径是 64, 显示起始角度为 0°, 结束角度是 90°, 显示结果如下图所示:



3.10 区域显示

3.10.1 矩形框或矩形区域显示(0x59, 0x69, 0x5A, 0x5B, 0x5C)

Tx: AA <CMD> (<Xs_0> <Ys_0> <Xe_0><Ye_0>)(<Xs_i> <Ys_i> <Xe_i><Ye_i>) CC 33 C3 3C

Rx: 无

➤ <CMD >

0x59 以前景色 (0x40 指令设置) 显示矩形框, 显示线宽是 1 个点阵;

0x69 以背景色 (0x40 指令设置) 显示矩形框, 显示线宽是 1 个点阵;

0x5A 以背景色 (0x40 指令设置) 填充矩形区域;

0x5B 以前景色 (0x40 指令设置) 填充矩形区域;

0x5C 把指定的矩形区域反色显示 (XOR 0xFF 操作), 再次反色将复原。

➤ <Xs_i> <Ys_i> <Xe_i> <Ye_i> (Xs_i, Ys_i) 是矩形框或矩形域的左上角坐标, (Xe_i, Ye_i) 是矩形框或矩形域的右下角坐标。

举例:

AA 5C 00 40 00 40 00 80 00 80 CC 33 C3 3C

把左上角坐标 (64, 64) 和右下角坐标 (128, 128) 定义的矩形区域反色, 指令执行后效果如下:



3.10.2 区域填充(0x64)

Tx: AA 64 < (X, Y) > <Color> CC 33 C3 3C

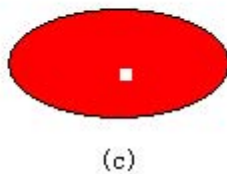
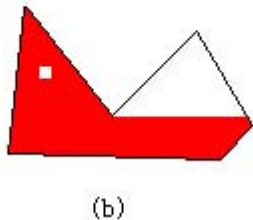
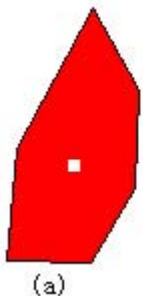
Rx: 无

➤ <(X, Y)> (X, Y) 是区域填充的种子点位置;

➤ <Color> 填充颜色;

注意:

- 填充区域的初始颜色要和种子点位置颜色一致; 否则只会填充和种子点颜色一致的区域 (其他颜色区域作为边界处理);
- 只适用于“凸多边形”区域填充, 对于“凹多边形区域”会有一些区域填充不到 (如下图 b 所示), 可以通过设置不同的种子点位置来实现“凹多边形区域”的完全填充;



区域填充的效果 (白点是种子点位置, 实际上也被填充了)

c. 不会改变调色板的属性。

举例:

AA 64 00 64 00 64 F8 00 CC 33 C3 3C

3.10.3 双色位图填充(0x73)

Tx: AA 73 Color0 Color1 (Xs, Ys) (Xe, Ye) (X, Y) Data CC 33 C3 3C

Rx: 无

- Color0: 0 bit 代表的填充颜色, 16bit;
- Color1: 1 bit 代表的填充颜色, 16bit;
- (Xs, Ys): 位图显示区域边界的左上角坐标;
- (Xe, Ye): 位图显示区域边界的右下角坐标;
- (X, Y): 位图填充的起始坐标;
- Data: 双色位图信息

3.11 全屏清屏 (0x52)

Tx: AA 52 CC 33 C3 3C

Rx: 无

使用背景色 (0x40 指令设定) 把全屏填充 (清屏)。

3.12 指定区域平移 (0x60, 0x61, 0x62, 0x63)

Tx: AA <CMD> (<Xs_0> <Ys_0> <Xe_0><Ye_0> <N_0>)(<Xs_i> <Ys_i> <Xe_i><Ye_i> <N_i>) CC 33 C3 3C

Rx: 无

➤ <CMD>

0x60 选定区域右环移: 从右向左移, 最左边区域移到最右边。

0x61 选定区域左环移: 从左向右移, 最右边区域移到最左边。

0x62 选定区域右移: 从右向左移, 最左边区域丢失, 最右边区域用背景色填充 (0x40 指令设定)。

0x63 选定区域左移: 从左向右移, 最右边区域丢失, 最左边区域用背景色填充 (0x40 指令设定)。

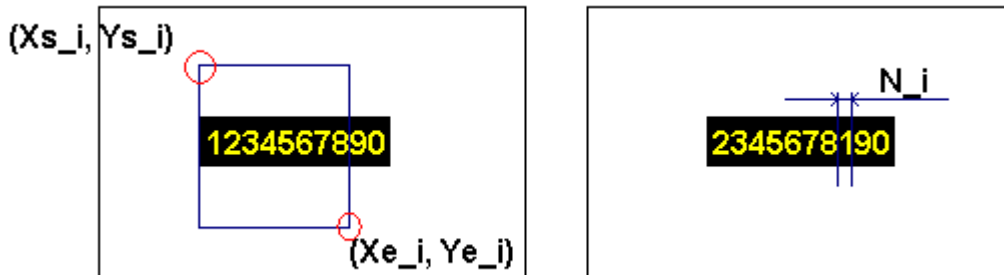
➤ <Xs_i> <Ys_i> <Xe_i><Ye_i> 选择区域的左上角和右下角坐标。

➤ <N_i> 移动区域点阵数, 0x01-0x0F。

举例:

AA 60 00 40 00 40 00 80 00 80 08 CC 33 C3 3C

把左上角坐标为 (64, 64), 右下角坐标为 (128, 128) 的区域从左向右平移 8 个点阵, 如下图所示:



注: 本指令在偏转 90° 显示版本中不支持。

3.13 图片或图标显示 (0x70, 0x71, 0x99, 0xE2, 0x7B, 0x9E)

3.13.1 图片显示 (0x70)

Tx: AA 70 <Pic_ID> CC 33 C3 3C

Rx: 无

<Pic_ID> 保存在 HMI Flash 存储器的图片索引 ID (对应 0xE2 指令)。

举例:

AA 70 00 CC 33 C3 3C 显示保存在 HMI 中的第 0 幅图片。

AA 70 01 02 CC 33 C3 3C 显示保存在 HMI 中的第 258 幅图片。

如果图片存储数量超过 256 幅, 图片 ID 会超过 255, 此时直接用两字节表示即可。

3.13.2 显示一幅图片并计算 CRC 校验 (0x7B)

Tx: AA 7B <Pic_ID> CC 33 C3 3C

Rx: AA 7B <CheckSum_H:L> CC 33 C3 3C

<Pic_ID> 保存在 HMI Flash 存储器的图片索引 ID (对应 0xE2 指令)。

<CheckSum_H:L> 当前图片内容的 CRC-16 校验值。

本指令用来对用户下载到 Flash 的图片进行校验, 以确保下载正确。

3.13.3 剪切图标显示 (0x71、0x9C、0x9D)

Tx: AA 71 <Pic_ID> <Xs> <Ys> <Xe> <Ye> <X> <Y> CC 33 C3 3C

或: AA 9C <Pic_ID> <Xs> <Ys> <Xe> <Ye> <X> <Y> CC 33 C3 3C

或: AA 9D <Pic_ID> <Xs> <Ys> <Xe> <Ye> <X> <Y> CC 33 C3 3C

Rx: 无

➤ <Pic_ID> 保存在 HMI Flash 存储器的图片索引 ID (对应 0xE2 指令)。

➤ <Xs> <Ys> <Xe> <Ye> (Xs, Ys) 是要剪切区域在原来图片的左上角坐标, (Xe, Ye) 是右下角坐标。

➤ <X> <Y> (X, Y) 是剪切下来的图片在当前屏幕显示位置的左上角坐标。

举例:

AA 71 08 01 90 00 00 03 1F 01 90 00 C8 00 14 CC 33 C3 3C

把第 8 幅图片的 (400, 0) (799, 400) 的区域剪切下来, 并显示到当前屏幕的 (200, 20) 位置, 结果如下:



0x9C、0x9D 指令与 0x71 指令的差异在于, 0x9C、0x9D 指令不会显示剪切图片的背景色, 实现“透明”的效果; 0x9C 指令执行前, 会自动先恢复背景; 0x9D 指令不会自动恢复背景。

3.13.4 自定义图标显示 (0x99)

Tx: AA 99 <X0, Y0, Icon_ID0>.....<Xn, Yn, Icon_IDn> CC 33 C3 3C

Rx: 无

➤ <Xn, Yn> 用户图标显示的位置

➤ <Icon_IDn> 用户图标的索引 ID, 两字节。

用户图标需要预先在 0x1D 配置文件中定义, 详见 3.26.4 节。

3.13.5 保存当前屏幕显示图片到 HMI 中 (0xE2)

Tx: AA E2 <Pic_ID> CC 33 C3 3C

Rx: AA E2 4F 4B CC 33 C3 3C

➤ <Pic_ID> 要保存的图片索引号。

3.13.6 保存当前屏幕显示图片区域到暂存缓冲区中 (0xE9)

Tx: AA E9 Xs Ys Xe Ye CC 33 C3 3C

Rx: 无

➤ Xs,Ys,Xe,Ye, 当前屏幕显示区域的左上角和右下角坐标。

仅 K600_XGA 支持。

3.13.7 把保存暂存缓冲区中的图片区域还原 (0x7F)

Tx: AA 7F Xs Ys Xe Ye CC 33 C3 3C

Rx: 无

➤ Xs, Ys, Xe, Ye, 还原到当前屏幕显示区域的左上角和右下角坐标。

仅 K600_XGA 支持。

3.13.8 剪切图标旋转角度后显示 (0x9E, 仅 H600 支持)

Tx: AA 9E <Mode> <Pic_ID> <Xs> <Ys> <Xe> <Ye> <Xc> <Yc> <AL> <Xc1> <Yc1> CC 33 C3 3C

Rx: 无

➤ <Mode> 0x00=透明剪切, (Xs, Ys) 位置为背景;

0x01=不透明剪切;

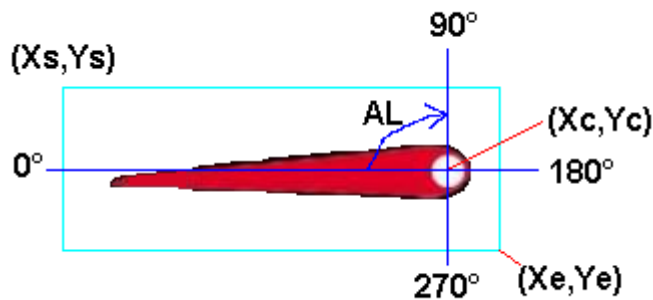
➤ <Pic_ID> 保存在 HMI Flash 存储器的图片索引 ID (对应 0xE2 指令), 两个字节 0x0000-0xFFFF;

➤ <Xs> <Ys> <Xe> <Ye> (Xs, Ys) 是要剪切区域在原来图片的左上角坐标, (Xe, Ye) 是右下角坐标;

➤ <Xc> <Yc> 图标旋转中心在原来图片的坐标位置;

➤ <AL> 图标旋转的角度, 两个字节 (0-719), 单位 0.5°;

➤ <Xc1> <Yc1> 剪切下来的图标在当前屏幕显示位置的旋转中心坐标。



提示: 图片设计时把原始指针图标放在 0 刻度位置, 在表盘刻度是顺时针方向均匀增大时, 剪切的旋转角度和实际参数将是线性对应的。



3.14 背光亮度控制（0x5E, 0x5F）

3.14.1 背光关闭（0x5E）

Tx: AA 5E CC 33 C3 3C

Rx: 无

3.14.2 设定触控（键控）背光模式（0x5E）

Tx: AA 5E 55 AA 5A A5 <V_ON> <V_OFF> <ON_TIME> CC 33 C3 3C

Rx: 无

- <V_ON> 点击触摸屏（或键盘）后背光自动点亮亮度，0x00-0x3F
- <V_OFF> 一段时间触摸屏（或键盘）没有点击后，背光自动关闭的亮度，0x00-0x3F
- <ON_TIME> 背光点亮的时间，单位为 0.5 秒，0x00-0xFF（最大 127.5 秒）

*设定的背光模式会保存下来。背光熄灭时，第一次点击将只会点亮背光而不会处理。
背光亮度触控（键控）功能，须通过 0xE0 指令来启用。*

3.14.3 打开背光到最大亮度（0x5F）

Tx: AA 5F CC 33 C3 3C

Rx: 无

3.14.4 调节背光亮度（0x5F）

Tx: AA 5F <PWM_T> CC 33 C3 3C

Rx: 无

- <PWM_T> 背光亮度 PWM 控制设定值，取值 0x00-0x3F，0x00 将关闭背光，0x3F 背光最亮。

调节的背光亮度不会保存，开机默认为最大亮度。

对于使用 CCFL 背光方式的 HMI，其背光亮度不能 PWM 调整，只能“开或关”控制。

3.15 触摸屏操作 (0x72, 0x73, 0x78, 0x79, 0xE4)

3.15.1 触摸位置自动上传 (0x72, 0x73)

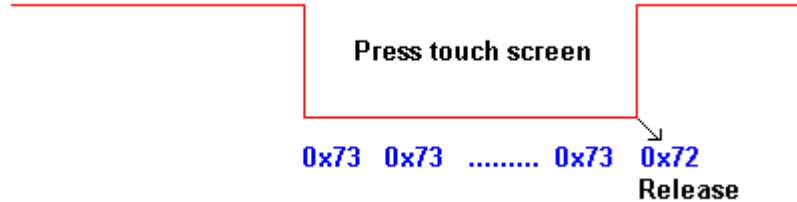
当按压触摸屏时，HMI 将自动以如下格式上传触摸位置坐标。

Tx: 无

Rx: AA 73 <X> <Y> CC 33 C3 3C 当按压触摸屏时上传，1 次或多次（可由 0xE0 指令设置）。

AA 72 <X> <Y> CC 33 C3 3C 当离开触摸屏时上传，仅 1 次（可由 0xE0 指令设置）。

➤ <X> <Y> 触摸位置坐标，与屏幕分辨率对应。



如果用户启用了触控界面处理功能（0xE0 指令设置），并同时启用了点击无效区域不上传坐标位置，则点击触摸屏不会上传 0x72、0x73 指令。

3.15.2 触摸键码自动上传 (0x78, 0x79)

如果用户启用了触控、键控界面处理功能（0xE0 指令设置），并启用了触控、键控键码回传功能，则当点击有效的触控区域或按键时，HMI 会自动上传用户预先设置的 2 字节触控、键控键码（0x1E、0x1B 配置文件定义）。

Tx: 无

Rx: AA 78 <Touch_Code> CC 33 C3 3C

注：0x79 对应触摸屏被按压时（0x73），0x78 对应触摸屏抬起时（0x72）；

3.15.3 进入触摸屏校准模式 (0xE4)

Tx: AA E4 55 AA 5A A5 CC 33 C3 3C

发送指令后，按照屏幕提示操作，依次点击屏幕“左上角”，“右上角”“左下角”十字交叉线白点提示的触摸位置；当校准完成时，HMI 会上传下面的指令：

Rx: AA E4 4F 4B CC 33 C3 3C

除非用户重新装配过触摸屏，迪文独有的漂移补偿技术保证触摸屏只需要在产品装配完成后校准 1 次，即可保证产品寿命周期内无需要再校准！



3.16 工作模式配置 (0xE0)

Tx: AA E0 55 AA 5A A5 <TFT_ID> <Bode_Set> <Para1> CC 33 C3 3C

或者 AA E0 55 AA 5A A5 <TFT_ID> <Bode_Set> <Para1> <Para2> CC 33 C3 3C

Rx: AA E0 <TFT_ID> <Bode_Set> <Para1> CC 33 C3 3C

或者 AA E0 <TFT_ID> <Bode_Set> <Para1> <Para2> CC 33 C3 3C

- <TFT_ID> 配置 TFT 屏参数; (V5.3 以后版本不开放给用户, 写 0x00 即可)
- <Bode_Set> 设置串口通信波特率, 设置如下:

Bode_set	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
波特率	1200	2400	4800	9600	19200	38400	57600	115200
以下波特率由于 PC 串口不支持, 请慎重设置								
Bode_set	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
波特率	28800	76800	62500	125000	250000	230400	345600	691200

- <Para1>配置触摸屏和键盘的处理模式, 设置如下:

Para1	Bit description
.7	0= 点击触摸屏后, 松开触摸屏时, 自动上传 0x72 指令; 触控模式下必须设置为 0。 1= 点击触摸屏后, 离开触摸屏时, 不上传 0x72 指令。
.6	坐标回传模式下: 0= 点击触摸屏后, 会以 100mS 的间隔定时自动上传 0x73 指令, 直到触摸屏松开; 1= 点击触摸屏后, 只会在按下时自动上传 1 次 0x73 指令。 触控模式下, 并且 Para1.0=1: 0= 点击触摸屏后, 会以 100mS 的间隔定时自动上传 0x79 指令, 直到触摸屏松开; 1= 点击触摸屏后, 只会在按下时自动上传 1 次 0x79 指令。
.5	0= 点击触摸屏后, HMI 不进行触控界面的切换; 1= 点击触摸屏后, HMI 自动按照 0x1E 配置文件的要求进行触控界面的切换。
.4	0= 背光不受触摸屏或键盘控制; 1= 背光由触摸屏或键盘控制, 同时用户也可以通过 0x5E/0x5F 指令强制开关。
.3	0= 触摸屏或按键有蜂鸣器伴音; 1= 触摸屏或按键无蜂鸣器伴音。
.2	0= 0° 显示 1= 偏转 90° 显示
.1	0= 触控模式下, 蜂鸣器伴音一直开启; 1= 触控模式下, 蜂鸣器只有在点击有效位置时鸣叫一次; Para1.1=1 时, 同时要设置 Para1.3=1。
.0	触控模式下: 0= 不上传 0x79 指令; 1= 上传 0x79 指令。

- <Para2>配置显示模式, 设置如下: (V6.0 以前版本不开放给用户, 不写本字节即可)

Para2	Bit description
.7	0=V5.0 以前版本刷新方式 (自动高速更新, 0x71+0x98 配合会抖动); 1=200mS 自动刷新显示, 指令执行后会后延刷新时间, 确保连续的指令可以同步显示。
.6	触摸屏坐标回传模式下: 0=触摸屏坐标回传不跟随 Para1.2 变化; 1=触摸屏坐标回传跟随 Para1.2 (偏转 90° 显示) 变化。
.5	0=显示偏转 180° (反视角显示); 1=正常视角显示。
.4	0=文本显示 (0x53、0x54、0x55、0x6E、0x6F、0x98), 自动恢复图片背景, 并且忽略文本的背景色; 1=文本显示时, 不自动恢复图片背景。
.3	保留, 写 1
.2	保留, 写 1
.1	保留, 写 1
.0	保留, 写 1

V6.0 以上版本, 0xE0 指令配置参数掉电不保存; 如果需要保存请使用迪文 PC 配置工具软件配置。重新上电会恢复用户预先通过配置工具软件设置的参数值。

3.17 指令定时循环执行 (0x9A)

本指令的主要作用是方便用户对定时使用的指令 (比如使用 0x71 指令显示动画) 预先保存在 HMI, 使用时无须用户干预, 节省用户代码。

3.17.1 开启指令定时循环执行功能

Tx: AA 9A Pack_ID CC 33 C3 3C

Rx: 无

- <Pack_ID> 自动循环指令组 ID, 0x00-0x0F, 每组 8KB, 最多包括 64 条 HMI 指令, 每条 HMI 指令区间 128 字节。

HMI 只能执行 1 组循环指令, 指令组定义在 0x1C 配置文件中, 最多 16 个指令组。

3.17.2 关闭指令定时循环执行功能

Tx: AA 9A FF CC 33 C3 3C

Rx: 无

3.18 暂存缓冲区操作 (0xC0, 0xC1, 0xC2)

暂存缓冲区的内容会被 0x52 (清屏) 指令改变。

3.18.1 写暂存缓冲区 (0xC0)

Tx: AA C0 <Address> <Word0……Wordn> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF。

- <Word0……Wordn> 要写入的数据

3.18.2 读取暂存缓冲区内容 (0xC2)

Tx: AA C2 <Address> <Data_Length> CC 33 C3 3C

Rx: AA C2 <Data_Pack> CC 33 C3 3C

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF。

- <Data_Length> 读取的数据 (Word) 长度, 2 字节。

- <Data_Pack> 读取的数据。

3.18.3 使用暂存缓冲区数据置点 (0xC101)

Tx: AA C1 01 <Address> <Pixel_Number> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF。

- <Pixel_Number> 置点数目, 每点 3 个字, 最多 13653 个点。

暂存缓冲区的点数据格式定义为: (X, Y, Color)

3.18.4 使用暂存缓冲区数据连线 (0xC102)

Tx: AA C1 02 <Address> <Line_Number> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;

- <Line_Number> 连线数目, 每条线 5 个字, 最多 8191 条线;

暂存缓冲区的连线数据格式定义为: (Xs, Ys, Xe, Ye, Color)。

3.18.5 使用暂存缓冲区数据显示折线图 (0xC103)

Tx: AA C1 03 <Address> <x> <y> <Line_Number> <D_x> <Dis_x> <K_y> <Color> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;

- <x> 显示起始位置的 x 坐标;

- <y> 显示 Y 坐标的 0 点 (最低点) 位置, 实际连线点位置 = y - Ly;

- <Line_Number> 连线数目, 0x0000-0x9FFF, 每条线 1 个字, 最多 40960 条线;

- <D_X> 读缓冲区的点间隔, 0x01-0xFF, 即每连 1 条线后, Address = Address + D_x;

- <Dis_X> 显示的 x 坐标增量, 0x01-0x0F, 即每连 1 条线后, x = x + Dis_x;

- <K_y> y 轴放大倍数, 0x00-0xFF, 单位为 1/16, K_y = 32 对应 y 轴放大 2 倍;

- <Color> 为显示线条的颜色, 不改变系统调色板属性;

暂存缓冲区的连线数据格式定义为: Ly (2 字节), Ly 为点的高度。

3.18.6 使用暂存缓冲区数据高速显示折线图 (0xC104)

Tx: AA C1 04 <Adr1> <x> <y> <Line_Number> <D_x> <Dis_x> <Color1> <Addr0> <Color0> CC 33 C3 3C

Rx: 无



- <x> 显示起始位置的 x 坐标;
- <y> 显示 Y 坐标的 0 点（最低点）位置, 实际连线点位置 = y - Ly;
- <Line_Number> 连线数目, 每条线 1 个字, 最多 40960 条线;
- <D_x> 固定写 0x01;
- <Dis_X> 显示的 x 坐标增量, 0x01-0x0F, 即每连 1 条线后, $x = x + \text{Dis_x}$;
- <Addr0> <Addr1> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
<Addr0> 是本窗口要擦除的历史曲线的首地址;
<Addr1> 是本窗口将要显示的曲线的首地址;
假设窗口连线数据点为 100 个, 暂存缓冲区只有 1 条曲线, 那么 Addr1 和 Addr0 的绝对值相差为 100。
- <Color0> <Color1> 为显示线条的颜色, 不改变系统调色板属性;
<Color0> 应该设置成窗口的背景颜色
<Color1> 应该设置成将要显示曲线的颜色

暂存缓冲区的连线数据格式定义为: Ly (2 字节), Ly 为点的高度。

本指令和 0xC103 指令基本类似, 不同在于:

- A. 读缓冲区的点间隔固定为 1;
- b. 避免了整个窗口清除带来的闪烁, 每连一个点 1 (由 Addr1 和 Color1 确定) 之前, 先把对应的原来的点 0 (由 Addr0 和 Color0 确定) 擦除, 实现**无闪烁显示**。

采用 115200bps 通信速率时, 本指令可以达到的折线显示速度极限大约是 5500 点/秒。

3.18.7 使用暂存缓冲区数据缩放显示折线图 (0xC105)

Tx: AA C1 05 <Address> <x> <y> <Line_Number> <D_x> <Dis_x> <M_y> <D_y> <Color> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
- <x> 显示起始位置的 x 坐标;
- <y> 显示 Y 坐标的 0 点 (最低点) 位置, 实际连线点位置 = $y - Ly$;
- <Line_Number> 连线数目, 0x0000-0x9FFF, 每条线 1 个字, 最多 40960 条线;
- <D_x> 读缓冲区的点间隔, 0x01-0xFF, 即每连 1 条线后, $Address = Address + D_x$;
- <Dis_x> 显示的 x 坐标增量, 0x01-0x0F, 即每连 1 条线后, $x = x + Dis_x$;
- <M_y> <D_y> y 轴放大倍数, 0x00-0xFF, 显示的高度 = $Y \times M_y / D_y$, 比如 $M_y = 4$, $D_y = 2$ 对应 Y 轴放大 2 倍;
- <Color> 为显示线条的颜色, 不改变系统调色板属性;

暂存缓冲区的连线数据格式定义为: Ly (2 字节), Ly 为点的高度。

3.18.8 使用暂存缓冲区数据缩放显示窗口限制双向折线图 (0xC106)

Tx: AA C1 06 <Address> <x> <y> <Line_Number> <D_x> <Dis_x> <M_y> <D_y> <Color> <Ymin> <Ymax> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
- <x> 显示起始位置的 x 坐标;
- <y> 显示 Y 坐标的 0 点位置, 实际连线点位置 = $y + Ly$ 或 $y - Ly$;
- <Line_Number> 连线数目, 0x0000-0x9FFF, 每条线 1 个字, 最多 40960 条线;
- <D_x> 读缓冲区的点间隔, 0x01-0xFF, 即每连 1 条线后, $Address = Address + D_x$;
- <Dis_x> 显示的 x 坐标增量, 0x01-0x0F, 即每连 1 条线后, $x = x + Dis_x$;
- <M_y> <D_y> y 轴放大倍数, 0x00-0xFF, 显示的高度 = $Y \times M_y / D_y$, 比如 $M_y = 4$, $D_y = 2$ 对应 Y 轴放大 2 倍;
- <Color> 为显示线条的颜色, 不改变系统调色板属性;
- <Ymin> Y 轴下限坐标 (窗口下限);
- <Ymax> Y 轴上限坐标 (窗口上限);

暂存缓冲区的连线数据格式定义为: Ly (2 字节), 说明如下:

Ly .15 为连线方向, 0=正向, 1=负向; 比如 $Ly = 0x8010$ 表示负向高度 16;

Ly .14- Ly .0 为连线高度。

3.18.9 使用暂存缓冲区作为置点缓冲区 (0xC107)

清空置点缓冲区:

Tx: AA C1 07 00 <Address> <X_Len> <Y_Len> CC 33 C3 3C

Rx: 无

- <Address> 置点缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
- <X_Len> 置点缓冲区对应屏幕窗口的 X 方向宽度 (点阵数), 0x0000-0x0FFF;
- <Y_Len> 置点缓冲区对应屏幕窗口的 Y 方向宽度 (点阵数), 0x0000-0x0FFF;

注意, 置点缓冲区占据的 RAM 地址大小 = $(X \times Y) / 16$, 比如用户从暂存缓冲区的 0x0000 位置开始开辟了一个 64×64 的置点缓冲区, 那么占据暂存缓冲区的地址空间为: 0x0000-0x00FF。

在置点缓冲区置点:

Tx: AA C1 07 01 <Address> <X_Len> <Y_Len> <Xs, Ys> <Color> <Mode> <(X0, Y0) ... (Xi, Yi)> CC 33 C3 3C

Rx: 无

- <Address> 置点缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
- <X_Len> 置点缓冲区对应屏幕窗口的 X 方向宽度 (点阵数), 0x0000-0x0FFF;
- <Y_Len> 置点缓冲区对应屏幕窗口的 Y 方向宽度 (点阵数), 0x0000-0x0FFF;
- <Xs, Ys> 置点缓冲区对应屏幕窗口的坐标位置;
- <Color> 置点/删除点颜色, 颜色是在当前页面上显示的颜色, 不影响 0x40 指令的调色板设置;
- <(X0, Y0) ... (Xi, Yi)> 置点位置, 越界不置点, 对应当前页面实际显示位置 $X = Xs + Xi$ $Y = Ys + Yi$;
- <Mode> 置点模式:
 - 0x00=仅在置点缓冲区删除点, 0x10=在置点缓冲区和当前页面同时删除点;
 - 0x01=仅在置点缓冲区置点, 0x11=在置点缓冲区和当前页面同时置点。

恢复置点缓冲区到当前页面:

Tx: AA C1 07 02 <Address> <X_Len> <Y_Len> <Xs, Ys> <Color> CC 33 C3 3C

Rx: 无

- <Address> 置点缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
- <X_Len> 置点缓冲区对应屏幕窗口的 X 方向宽度 (点阵数), 0x0000-0x0FFFF;
- <Y_Len> 置点缓冲区对应屏幕窗口的 Y 方向宽度 (点阵数), 0x0000-0x0FFF;
- <Xs, Ys> 置点缓冲区对应屏幕窗口的坐标位置;
- <Color> 置点颜色, 颜色是在当前页面上显示的颜色;

仅在置点缓冲区置点 (01 模式) 并恢复到当前页面之前必须要清空置点缓冲区。

本指令主要提供用户“图层”来灵活、方便的实现一些纯色图标的操作, 比如十字光标的移动。

3.18.10 使用暂存缓冲区来显示多参数 (0xC108)

Tx: AA C1 08 <Address> <Parameter_N> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
- <Parameter_N> 要显示的参数数量, 0x01-0xFF;

注意: 显示为右对齐方式

暂存缓冲区的参数数据格式定义如下:

暂存缓冲区 相对地址	定义	说明
0x00	Mode	Mode. 15:.12 显示的最多整数位数; Mode. 11:.8 显示的小数位数, 0 表示无小数位; Mode. 7 0=无效的整数位零不显示 1=无效的整数位零显示 比如 Mode. 15:.12=3, Mode. 7=1, 则 0.3 会显示成 000.3。 Mode. 6 0=文本使用 0xC108 指令定义的背景色 1=文本自动取背景色显示, 取色位置为 (X-1, Y), 需要背景是纯色。 Mode. 5-Mode. 4 未定义, 写 0 Mode. 3-Mode. 0 显示字体大小 (基于 Lib_ID=0 默认 ASCII 字库) 0=8*8 1=8*12 2=6*12 3=8*16 4=12*24 5=16*32 6=20*40 7=24*48 8=28*56 9=32*64 0x0A-0x0F 未定义
0x01	X	参数显示的起始位置 X 坐标
0x02	Y	参数显示的起始位置 Y 坐标
0x03	F_Color	参数显示的前景色, 不会改变调色板设置;
0x04	B_Color	参数显示的背景色, 不会改变调色板设置;
0x05	Parameter	参数数据, 4 个字节的带符号整数, Parameter. 31 是符号位。

参数设置举例

显示数据	Parameter	Mode (显示字体为 16*32 点阵, 6.3 模式, 无效的整数位 0 不显示)
12345	0x00 00 30 39	0x6005
123.450	0x00 01 E2 3A	0x6305
-123.450	0xFF FE 1D C6	0x6305

举例:

AA C0 00 00 63 05 00 0A 00 0A F8 00 00 1F 00 01 E2 3A CC 33 C3 3C AA C1 08 00 00 01 CC 33 C3 3C

将在 (10, 10) 位置显示 123.450。



3.18.11 使用暂存缓冲区来缓冲指令实现同步显示 (0xC110)

Tx: AA C1 10 <Address> <Frame_Number> CC 33 C3 3C

Rx: 无

- <Address> 暂存缓冲区 (RAM) 的首地址 (字地址), 一共 40KWord, 0x0000-0x9FFF;
- <Frame_Number> 要连续显示的指令帧数目, 0x01-0xFF;

暂存缓冲区的指令帧定义如下:

首地址	定义	帧结构
<Address>	第 1 个指令帧	帧长度+指令+数据
<Address>+0x80	第 2 个指令帧	比如, 显示第 0 幅图片, 定义为 03 70 00 00
.....		注意: 由于暂存缓冲区数据是以字的形式存储的, 所以写入的字节数必须是偶数个。
<Address>+0x80×(K-1)	第 K 个指令帧	

举例:

在缓冲区写入指令;

```
AA C0 00 00 09 55 00 00 00 00 30 31 32 33 CC 33 C3 3C
AA C0 00 80 09 53 00 64 00 64 34 35 36 37 CC 33 C3 3C
```

实现同步显示:

```
AA C1 10 00 00 02 CC 33 C3 3C
```

说明:

由于暂存缓冲区是按照字 (Word) 存储, 所以一个指令帧占据的地址是 0x80, 但实际存储空间长度是 256 字节。

本指令主要用来解决页面有很多参数需要同时刷新显示, 为了避免通信引起的延时 (尤其是低波特率时) 造成参数刷新过程不同步的问题。

3.19 键盘操作 (0x71, 0xE5)

3.19.1 键码上传 (0x71)

Tx: 无

Rx: AA 71 <K_code> CC 33 C3 3C

- <K_code> 用户预先设置的键码, 一旦 HMI 的键盘接口扫描到按键按下, 将自动上传键码, 键码速度为 5 键/秒。

3.19.2 键码设置 (0xE5)

Tx: AA E5 55 AA 5A A5 <K_Code0.....K_Code63> CC 33 C3 3C

Rx: 无

- <K_Code0.....K_Code63> 要设置的键码, 固定为 64 个, 对于 4×4 的键盘接口, 只有 16 个键码有效。



3.20 用户存储器读写 (0x90, 0x91)

数据库的物理介质是 NAND Flash, 可擦写次数是 100000 次, 寿命为 10 年。

3.20.1 写随机数据存储器 (0x90 64KB)

Tx: AA 90 55 AA 5A A5 01 DE <Address> <Data0.....Data*i*> CC 33 C3 3C

Rx: AA 90 4F 4B CC 33 C3 3C

- <Address> 写入数据存储器的首地址, 2 字节, 0x0000-0xFFFF;
- <Data0.....Data*i*> 要写入的数据串, 字节的形式。

3.20.2 写顺序数据存储器 (0x90 30MB)

Tx: AA 90 55 AA 5A A5 <Address> <Data0.....Data*i*> CC 33 C3 3C

Rx: AA 90 4F 4B CC 33 C3 3C

- <Address> 写入数据存储器的首地址, 4 字节, 0x00000000-0x01DDFFFF;
- <Data0.....Data*i*> 要写入的数据串, 字节的形式。

与随机数据存储器不同, 顺序数据存储器只能顺序写入, 不能随机地址写。

整个顺序存储器分成 239 个 128KB 数据页, 每遇到页首 (地址=* *****0 00 00) 会自动擦除当前要写的页, 擦除前不会做数据的备份, 其它页数据不影响。适合做无纸记录、音频录音等连续、大数据量的数据存储。

3.20.3 读数据存储器 (0x91)

Tx: AA 91 <Address> <Length> CC 33 C3 3C

Rx: AA 91 <Address> <Length> <Data0.....Data*i*> CC 33 C3 3C

- <Address> 读数据存储器的首地址, 4 字节, 0x00000000-0x01DEFFFF; 随机数据存储器地址范围为 0x01 DE 00 00-0x01DE FF FF;
- <Length> 读数据存储器的长度, 2 字节, 一次最多读取 64KB;
- <Data0.....Data*i*> 读出的数据串, 字节的形式。

3.21 字库或配置文件下载 (0xF2)

Tx: AA F2 F2 F2 5A A5 <Lib_ID> CC 33 C3 3C

Rx: Please Tx Text_Lib !

然后用户下发相应字库;

字库保存完成后, HMI 再次应答: *****One Text_Lib Saved OK !*****

- <Lib_ID> 字库的存储位置, 一共 60 个字库 (0x00-0x3B), 其中 0x00-0x1F 为 32 个 128KB 的小字库 (配置文件), 0x20-0x3B 为 28 个 1MB 的大字库。

除非用户需要自己设计汉字库, 请不要修改 Lib_ID=0x00、0x20、0x21、0x22、0x23 位置的字库, 否则会引起 0x53、0x54、0x55、0x6E、0x6F 指令显示不正确。

3.22 简单算法支持 (0xB0)

3.22.1 拼音输入法 (0xB001、0xB004)

Tx: AA B0 01 <PY_Code> CC 33 C3 3C 或 AA B0 04 <PY_Code> CC 33 C3 3C

Rx: AA B0 01 <HZ_Num> <HZ_String> CC 33 C3 3C 或 AA B0 04 <HZ_Num> <HZ_String> CC 33 C3 3C

- <PY_Code> 汉字拼音, 大写表示, 最多 6 个字节;
- <HZ_Num> 该拼音下的汉字数目, 0 表示拼音错误; B001 指令数目为 1 字节, B004 指令为 2 字节。
- <HZ_String> 该拼音下的所有汉字, 内码编码。

0xB001 是针对 GB2312-80 1 级字库; 0xB004 是针对 GBK 扩展字库。

使用 0xB004 指令, 则 Lib_ID=0x01 位置的字库用户不能使用。

3.22.2 MAC 计算 (0xB002)

Tx: AA B0 02 <A,B,C,D> CC 33 C3 3C

Rx: AA B0 02 <E,F> CC 33 C3 3C

- <A,B,C,D> 计算(A×B+C)/D, A、B、C、D 均为 2 字节无符号整数;
- <E,F> 计算结果, E 是商 (4 字节), F 是余数 (2 字节)

3.22.3 数组排序 (0xB003)

Tx: AA B0 03 <Pack0> CC 33 C3 3C

Rx: AA B0 03 <Pack1> CC 33 C3 3C

- <Pack0> 要排序的数组, 数组数据为 2 字节;
- <Pack1> 排序后的数组, 数组数据是 2 字节, 升序排列。

3.23 蜂鸣器控制 (0x79)

Tx: AA 79 <On_Time> CC 33 C3 3C



Rx: 无

- <On_Time> 0x01-0xFF, 蜂鸣器鸣叫时间长度, 单位为 10mS。

蜂鸣器鸣叫指定时间。

3.24 时钟 (RTC) 显示和读取 (0x9B, 0xE7)

3.24.1 关闭时钟显示

Tx: AA 9B 00 CC 33 C3 3C

Rx: 无

3.24.2 打开时钟显示

Tx: AA 9B FF <RTC_Mode> <Text_Mode> <Color> <X> <Y> CC 33 C3 3C

Rx: 无

- <RTC_Mode> 时钟显示模式

0x00: HH:MM:SS

0x01: 20YY-MM-DD HH:MM:SS

- <Text_Mode> 时钟显示的字体

0x00: 8*8

0x01: 6*12

0x02: 8*16

0x03: 12*24

0x04: 16*32

0x05: 20*40

0x06: 24*48

0x07: 28*56

- <Color> 时钟显示字体颜色

- <X> <Y> 时钟显示位置;

3.24.3 时钟调整

Tx: AA E7 55 AA 5A A5 <YY:MM:DD:HH:MM:SS> CC 33 C3 3C

Rx: 无

- <YY:MM:DD:HH:MM:SS> 为要设置的时间, 年:月:日:时:分:秒, BCD 码表示。

举例:

AA E7 55 AA 5A A5 08 11 28 12 57 00 CC 33 C3 3C

设置当前时间为 2008 年 11 月 28 日, 12 时 57 分 00 秒。

3.24.4 读取当前时钟 (公历)

Tx: AA 9B 5A CC 33 C3 3C

Rx: AA 9B 5A <YY:MM:DD:WW:HH:MM:SS> CC 33 C3 3C

- <YY:MM:DD:WW:HH:MM:SS> 当前时钟数据

比如 08 12 25 04 09 58 00 表示 2008 年 12 月 25 日星期四, 时间是 09:58:00。

如果 HMI 不支持时钟功能, 将返回未知的结果!

3.24.5 读取当前时钟 (农历)

Tx: AA 9B 5B CC 33 C3 3C

Rx: AA 9B 5B <YY:MM:DD:生肖:天干:地支> CC 33 C3 3C

- <YY:MM:DD:生肖:天干:地支> 当前农历年月日和生肖、天干地支纪年, 生肖和纪年均为内码表示。

比如 09 02 03 '牛己丑' 表示当前农历日期是 2009 年 02 月 03 日, 2009 年是“牛”年, 纪年为“己丑”

如果 HMI 不支持时钟功能, 将返回未知的结果!



3.25 音乐播放（0x30, 0x32, 0x33）

本指令需要相应的硬件（迪文 DMA5601 立体声音录放模组）支持。

3.25.1 播放指定位置的音乐（0x30）

Tx: AA 30 <Start_SEG> <SEG_Number> <Play_Time> CC 33 C3 3C

Rx: AA 3F 4F 4B 开始播放

AA 3F 4F 4B 播放结束

- <Start_SEG> 播放起始音乐段地址，0x00-0xFF；
- <SEG_Number> 播放的音乐段数目，0x00-0xFF；
- <Play_Time> 重复播放次数，0x00-0xFF

3.25.2 音量调节（0x32）

Tx: AA 32 <Volume_L> <Volume_R> 00 CC 33 C3 3C

Rx: AA 3F 4F 4B

- <Volume_L> 左声道音量，0x00-0x3F；
- <Volume_R> 右声道音量，0x00-0x3F；

3.25.3 停止播放（0x33）

Tx: AA 33 55 AA 5A CC 33 C3 3C

Rx: AA 3F 4F 4B

3.26 配置文件的使用 (触控界面, 键控界面, 动画, 图标库)

迪文 HMI 通过以下配置文件来实现简单的 OS 功能, 大大降低用户代码工作量。

3.26.1 触控界面自动切换 (0x1E、0x1A 配置文件)

带触摸屏的迪文 HMI, 为了减少用户的代码量, 可以通过预先下载配置文件到 HMI 中, 并把 HMI 配置为触控界面自动切换模式来实现触控界面的用户“免干预”。

其开发过程如下:

第 1 步: 先设计好和 HMI 物理分辨率相同的用户界面, 并下载到 HMI (终端) 中;

比如使用 DMT64480S057_12WT, 就把界面分辨率设计成 640×480 像素点阵。

第 2 步: 生成配置文件

配置文件是由最多 8192 条触控指令组成的二进制文件, 每条触控指令 16 个字节, 定义如表 3-26-1:

首地址	数据长度 (Byte)	定义	说明
0x00	2	Pic_Now	当前显示屏幕的图片编号; 如果 Pic_Now 的高字节为 0xFF 表示触控指令结束。
0x02	4	x_s, y_s	有效触控区域的左上角坐标。
0x06	4	x_e, y_e	有效触控区域的右下角坐标。
0x0A	2	Pic_Next	点击有效触控区域后切换到下一个界面的图片编号; 如果 Pic_Next 的高字节为 0xFF 表示不进行界面切换。
0x0C	2	Pic_Cut	触控动画图片编号; 如果 Pic_Cut 的高字节为 0xFF 表示没有触控动画图片。
0x0E	2	Touch_Code	点击有效触控区域后, 上传的触控键码 (作为触发用户软件的消息); 如果 Touch_Code 的高字节为 0xFF 表示不上传触控键码。 如果 Touch_Code 的高字节为 00xF0-0xF3, 表示上传的数据串索引到 0x1A 配置文件, 此时 (Touch_Code-0xF000) 为索引 ID。0x1A 配置文件中, 每条索引长度固定为 128 字节, 第一个字节为本条索引的有效长度。

表 3-26-1 触控界面配置指令的定义

生成配置文件的过程, 其实就是用户安排界面切换流程, 设计界面的过程, 一般美工即可完成。配置文件可以直接用 UltraEdit 编写, 也可以借助一些编译系统 (比如 C51, ASM51) 来实现。

图 3-26-1 以一般硬件工程师最熟悉的, 也是最“古老”的 DOS 系统下的 ASM51 编译器为例, 来说明如何编写配置文件。



姓名: 肖钦 (Xiao Qin)
称号: 鞍马王
性别: 男
籍贯: 南京
生日: 1985.1.12 (654,195)
身高: 1.64米
体重: 55公斤
奥运会报名项目: 体操

有效触控区域 (792,292)

Pic_Now=19



姓名: 何雯娜 (He Wenna)
国家/地区: 中国
籍贯: 福建 龙岩
民族: 汉族
性别: 女
出生日期: 1989-01-19
身高: 160cm
体重: 48kg
大项: 蹦床
小项: 女子网上个人

Pic_Next=20



Pic_Cut=0x02

;DMT80480S070_02WT HMI 触控界面切换示例
;点击第 19 幅图片的“下一位”按钮, HMI 自动
;切换到第 20 幅图片, 并上传键码 1 (0x0031)
;点击时, 使用第 2 幅图片上的按钮按下效果。
;蓝色字体为有效触控区域坐标
ORG 0000H
DW 19, 654, 195, 792, 292, 20, 2, 1 ;1 条触控指令
DW 0FFFFH ;所有触控指令定义结束
END

图 3-26-1 使用 ASM51 编译器来编写配置文件的举例

上面的图例中, 只写了一个触控按键, 切换一个界面的例子, 更多的界面切换, 用户增加指令即可。把编写好的配置文件 (*.ASM), 使用 ASM51 编译器 (ASM51.EXE) 编译生成 HEX 文件, 再使用 HEX 转 BIN



工具（HEXBIN.EXE）转换成 BIN 文件，就获得了我们需要的配置文件。

第 3 步：把配置文件下载到 HMI（终端）中

使用 0xF2 字库下载指令，把生成的二进制配置文件下载到 HMI（终端）的 0x1E 字库位置即可。

第 4 步：配置 HMI 为触控界面自动切换模式

使用 0xE0 指令，把 Para1 参数的第 5 位（Para1.5, 0x20）置 1，点击触摸屏时，HMI（终端）将不再上传坐标位置，而是自动进行触控界面的切换，上传用户预定义的触控键码。

第 5 步：测试界面切换是否准确，可能会需要多次重复第 2 和第 3 步工作。

0x1A 配置文件的定义			
0x1A 文件由最多 1024 条指令构成，每条指令最多 127 字节，固定占 128 字节存储器空间，单条指令定义如下。			
首地址	数据长度(Byte)	定义	说明
0x00	1	Length_Command	本条指令长度。 如果 Length_Command=0x00，表示本条指令是一条组合指令（批处理指令）。
0x01	不定	指令	如果 Length_Command 不是 0x00，0x01 开始是要发送的指令。
		指令指针 (Cmd_EN, Cmd_ID, Tx_Delay)	如果 Length_Command=0x00，后面是组合指令的指针，每个指针由 4 个字节组成，最多 31 个指令指针（组合指令）。 指针定义如下： Cmd_EN: 0x00 表示指令组发送结束，其它表示指令发送； Cmd_ID: 要发送的指令 ID, 0x000-0x3FF, 对应 0x1A 文件的非组合指令编号，不支持组合指令的嵌套； Tx_Delay: 每条指令的发送间隔，单位为 0.1 秒。 发送组合指令期间，HMI 不响应任何用户指令和外设操作。

以上开发过程，也可以借助迪文提供的“触控界面开发软件”在计算机上完成触控文件的生成并进行仿真测试。软件具体使用方法请咨询迪文技术支持工程师。

使用配置文件来设计触控界面，不仅大大降低了二次开发的代码量，降低了开发难度；更重要的，我们希望这能够改变我们产品设计和市场开拓的思路：

- 把产品的“**算法**”和“**界面**”设计两部分彻底分开；算法是企业的核心竞争力，而不用让宝贵的研发资源浪费在大量冗长的界面代码设计上；
- **产品研发可以并行进行**，不仅界面和算法可以并行同时设计；而且可以多个美工来负责不同的界面设计；由于触控键码起到了“触发消息”的作用，负责算法不同部分的工程师也可以进行并行设计和调试；
- **提高了产品的可靠性**；原则上来说，所有的用户程序处于同一个并行的级别上，功能模块之间相互独立，简化了测试流程。
- 让产品的**升级换代非常容易**。产品稳定后，产品的升级换代，基本上都是“界面”的升级换代，“算法”很少改进。
- 采用配置文件的方式，可以很轻松的实现**客制化或者多风格**（很多“皮肤”）的界面，因为只要在配置文件跳转位置上插入不同界面即可，而上传的键码是相同的。
- **大大缩短新产品的开发时间**，提高了市场竞争力，以迪文帮助客户开发一款产品为例：
 - a. 用 1—3 个工作日和客户谈妥基本需求；
 - b. 用 1—2 个工作日我们的美工就会在标准 HMI 的基础上，提供客户产品的最终界面；同时我们的结构工程师也准备好了最终产品的三维效果图请客户确认；
 - c. 客户确认后，我们的硬件工程师开始 PCB 设计，结构工程师把图纸交到深圳的工厂做快速成型，美工开始修改界面并提交软件工程师流程文件；
 - d. 1 个星期左右，我们就可以提交客户完整的成品样机进行验收。

3.26.2 键控界面自动切换 (0x1B 配置文件)

与触控界面的控制类似，键控界面是通过按键来触发的。其配置文件保存在 0x1B 配置文件中。0x1B 配置文件由最多 5957 条键控指令组成，每条键控指令 22 个字节，定义如表 3-26-2：

首地址	数据长度 (Byte)	定义	说明
0x00	2	Pic_Now	当前显示屏幕的图片编号； 如果 Pic_Now 的高字节为 0xFF 表示键控指令结束。
0x02	2	0x00:Key_Code	按键键码
0x04	2	Pic_Next	按键切换到下一个界面的图片编号； 如果 Pic_Next 的高字节为 0xFF 表示不进行界面切换。 如果 Pic_Next 的高字节为 0xFE，表示进行区域图片的切换。
0x06	14	Pic_Cut, Xs, Ys, Xe, Ye, X, Y	Pic_next 高字节=0xFE 区域图片切换区域定义 Pic_next 高字节=其它 无定义
0x14	2	Touch_Code	按键后，上传的键码(作为触发用户软件的消息)；如果 Touch_Code 的高字节为 0xFF 表示不上传键码。 如果 Touch_Code 的高字节为 00xF0-0xF3，表示上传的数据串索引到 0x1A 配置文件，此时 (Touch_Code-0xF000) 为索引 ID。0x1A 配置文件中，每条索引长度固定为 128 字节，第一个字节为本条索引的有效长度。

表 3-26-2 键控界面配置指令的定义

3.26.3 自动循环执行指令组 (0x1C 配置文件)

由 16 个 8KB 的指令组组成，每个指令组固定 8KB，包含最多 64 条 HMI 指令，每条 HMI 指令固定占据 128 字节存储器空间。

每条 HMI 指令格式定义如下：

首地址	定义	说明
0x00	Command_Delay	指令执行后延时时间，单位为 8mS，0x00 表示不延时
0x01	Command_Length	本条指令长度 0x00：本条指令无效 其它：本条指令长度（0x02 开始计算长度）
0x02-0x7F	指令	去掉 0xAA 帧头和 0xCC 33 C3 3C 帧结束符后的标准 HMI 指令

例如：

要实现 3 幅图片循环切换的效果，可编写配置文件如下：

```

ORG 0000H                ;第一组，PADID=00H
DB 125,2,70H,00H        ;显示第 0 幅图片
ORG 0080H
DB 125,2,70H,01H        ;显示第 1 幅图片
ORG 0100H
DB 125,2,70H,02H        ;显示第 2 幅图片

ORG 2000H                ;第二组，PADID=01H
DB 125,2,70H,03H        ;显示第 3 幅图片
ORG 2080H
DB 125,2,70H,04H        ;显示第 4 幅图片
    
```

用 9A 指令调用：

```

AA 9A 00 CC 33 C3 3C;第 0、1、2 幅图片循环切换
AA 9A 01 CC 33 C3 3C;第 3、4 幅图片循环切换
    
```

说明：

1. 每条指令固定 128 字节存储器空间，所以，每条指令的首地址分别为 0000H、0080H、0100H...
2. 每组指令固定 8KB 的存储器空间，所以，每组指令的首地址分别为 0000H、2000H、4000H...

3.26.4 图标显示 (0x1D 配置文件)

迪文 HMI 有 1 条 0x71 图片剪切指令，可以让用户把保存在 HMI 中一幅图片上的一个区域剪切下来，粘贴到当前显示界面的指定位置（详见本文档 3.13.2 节）。但在实际使用时，客户使用起来还是不方便，所以增加了 0x99 指令和 0x1D 图标定义库文件，来让客户以文本的方式调用图标显示。

0x99 指令的格式如表 3-26-3 所示。

指令	数据	说明
0x99	$(x, y, Icon_ID)_0 + \dots + (x, y, Icon_ID)_n$	(x, y) 是图标显示目标位置的左上角坐标, Icon_ID 是图标在图标库文件的索引 ID。

表 3-26-3 0x99 指令的格式

图标库配置文件是由最多 13107 条 (对应 Icon_ID=0x0000-0x3332) 图标定义组成的二进制文件, 每条图标定义包含 10 个字节, 定义如表 3-26-4:

首地址	数据长度 (Byte)	定义	说明
0x00	2	Pic_ID	图标保存的图片编号。
0x02	4	X_s, Y_s	图标区域的左上角坐标。
0x06	4	X_e, Y_e	图标区域的右下角坐标。

表 3-26-4 图标的定义

举例:

```
ORG 0000H
DW 10,0,0,100,100; Icon_ID is 0000H
DW 11,0,0,100,100; Icon_ID is 0001H
DW 12,0,0,100,100; Icon_ID is 0002H
```

用 99 指令调用:

```
AA 99 00 00 00 00 00 00 00 64 00 64 00 01 00 00 00 64 00 02 CC 33 C3 3C
```

分别在 (0, 0)、(100, 100)、(0, 100) 的位置显示保存的图标。

对迪文 HMI 而言, 当收到 0x99 指令时, 会按照如下的步骤处理:

- 根据 Icon_ID, 到 0x1D 配置文件的 Icon_ID×10 的位置开始, 取出 Pic_ID, (X_s,Y_s), (X_e,Y_e);
- 与 0x71 指令一样, 进行图片剪切: 0x71 Pic_ID, (X_s,Y_s), (X_e,Y_e) (X,Y)
- 进行下一个图标显示的处理。

配置文件的生成可以参考本文档 3.26.1 节的方法。

配合配置文件, 使用 0x99 指令, 可以很方便的解决以下问题:

- 对于模拟表盘, 可以把不同刻度显示做成图标, 然后根据程序变量进行方便、直接的调用, 而不用用户在代码中二次查表;
- 对于特殊字符, 甚至是 UNICODE 编码也没有的字符, 可以做成图标的方式来调用; 免去了做字库的麻烦。
- 可以把 Windows 的一些界面“元素”, 比如光标、鼠标指针等, 很方便的“拿来”使用。

3.27 HMI 和视频功能的切换（0x7A）

对于模拟视频播放

Tx: AA 7A <Work_Mode> <Video_Mode> <Video_CH> CC 33 C3 3C

Rx: 无

- <Work_Mode> 0x00=HMI 0x01=Video
- <Video_Mode> 0x00=PAL 0x01=NTSC
- <Video_CH> 0x00=CVBS 接口输入视频信号 0x01=S 端子输入视频信号

对于基于 SD 卡的数字视频播放

Tx: AA 7A <Work_Mode> <KEY_VALUE> CC 33 C3 3C

Rx: 无

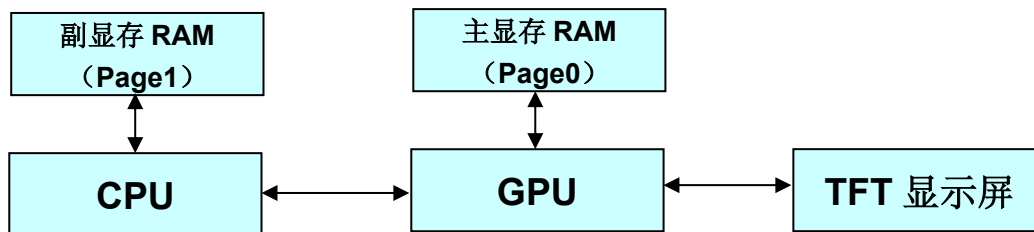
- <Work_Mode> 0x00=HMI 0x01=Video 播放
- <KEY_VALUE> 视频播放模式下，功能键值，定义如下

KEY_VALUE	功能键值说明
0x00	按键抬起
0x01	播放/暂停键 按下
0x02	声音开启
0x03	声音关闭
0x04	菜单操作：确认键 按下
0x05	菜单操作：向后键 按下
0x06	菜单操作：向前键 按下
0x07	菜单操作：退出键 按下
0x08	菜单操作：音量增加键 按下
0x09	菜单操作：音量减小键 按下
0x0A	菜单操作：中文/英文选择键 按下

3.28 强制刷新 1 次全屏显示（0xD0）

Tx: AA D0 CC 33 C3 3C

Rx: 无



K600、H600 在 V5.5 以上版本采用双显存结构（如上图所示）以避免以下问题：

- 使用背景恢复（比如 0x71+0x98 指令实现无背景色文本显示）时出现闪烁或者抖动；
- 一个页面有很多参数显示时，显示不同步（波特率越低越明显）的问题。

使用双显存架构时，用户看到的界面是主显存（Page0），而指令操作的界面是副显存（Page1），CPU 会定时（100ms）把副显存的显示内容刷新到主显存上，以确保显示的同步。但下面两种情况例外：

- 0x70 指令显示完图片后，CPU 立即自动把副显存内容刷新到主显存上；
- CPU 处理其它涉及显示的指令后，即使刷新显示定时到，也会自动把刷新周期后延 30ms，如果 30ms 不再有新的显示指令收到，才会刷新显示。

0xD0 指令的功能就是不管刷新定时，立即把副显存的内容刷新到主显存上。

在实时高速曲线显示中，需要定时（比如 100ms）发送 0xD0 指令，否则显示内容可能不会更新（没有显示出来）。

3.29 使用触摸屏输入参数或文本 (0x7C, 仅 H600 支持)

使用触摸屏输入参数, 必须满足以下条件:

1. 使用 0x1E 触控配置文件;
2. 键码的低字节必须是有效的键码并符合以下规范:

键码 (HEX 格式)	定义	说明
0xF0	ESC	退出
0xF1	Enter	确认
0xF2	BackSpace	退格
0xF3	Page up	上翻页
0xF4	Page down	下翻页
0xF5	CapsLock	大/小写切换, 仅限 A-Z (a-z), 有效时按钮会突出显示 如果预设键码为 A-Z: 则点击 CapsLock 后为 a-z; 如果预设键码为 a-z: 则点击 CapsLock 后为 A-Z。 如果使用, 在触控文件中不能设置按钮动画。
0xFD	Return	返回

3. 用来做输入显示的文本框背景必须为纯色;

4. 使用 0x7C02 中文输入时, 必须在 0x01 字库放置迪文 GBK 输入法词库 (DWIN_PY_GBK_01.BIN)。

3.29.1 输入纯 ASCII 字符串 (0x7C01)

Tx: AA 7C 01 R_ID VP_ID X Y Str_Max_Num Str_Scale Str_Color [Init_V] CC 33 C3 3C

Rx: AA 7C 01 R_ID Str_Number String CC 33 C3 3C

- <R_ID> 用户预设的一个两字节返回值 (0x0000-0xFFFF), 用于用户软件进行返回参数识别;
 - <VP_ID> 输入法使用的虚拟触控页面编号, VP_ID=0xFFFF 表示使用当前页面;
 - <X, Y> 文本输入显示的左上角地址, 由于文本显示背景色是从 (X, Y) 地址读取, 所以实际显示地址是从 (X+1, Y) 开始的, 文本显示时使用右对齐方式;
 - <Str_Max_Num> 输入字符串的最大长度, 0x01-0x40, 最多 64 个 ASCII 字符;
 - <Str_Scale> 输入字符串的显示格式控制
- <Str_Scale.7> 0=输入的字符正常显示; 1=输入的字符显示为"*", 用来做密码等输入的显示。
 <Str_Scale.6> 0=输入无初始值; 1=输入有初始值。
 <Str_Scale.5-.4> 保留, 写 0。
 <Str_Scale.3-.0> 输入文本显示大小, 0x00-0x07, 最多 8 种字体。

Str_Scale	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07
字体大小	8*8	6*12	8*16	12*24	16*32	20*40	24*48	28*56

- <Str_Color> 输入字符串的显示颜色, 使用 65K 颜色模式。
- <[Init_V]> 预设的初始值, ASCII 编码; 设定初始值时必须同时设定 <Str_Scale.6>=1。
- <Str_Number> 实际返回的 ASCII 字符数目, 0x00 表示空值
- <String> 返回的 ASCII 字符

结束返回条件: ESC (0xF0, 返回空值)、Enter (0xF1)、Return (0xFD)



3.29.2 输入中英文混合字符串 (0x7C02)

Tx: AA 7C 02 R_ID VP_ID X Y Str_Max_Num Str_Scale Str_Color T_Color Tx Ty [Init_V] CC 33 C3 3C

Rx: AA 7C 02 R_ID String_Number String CC 33 C3 3C

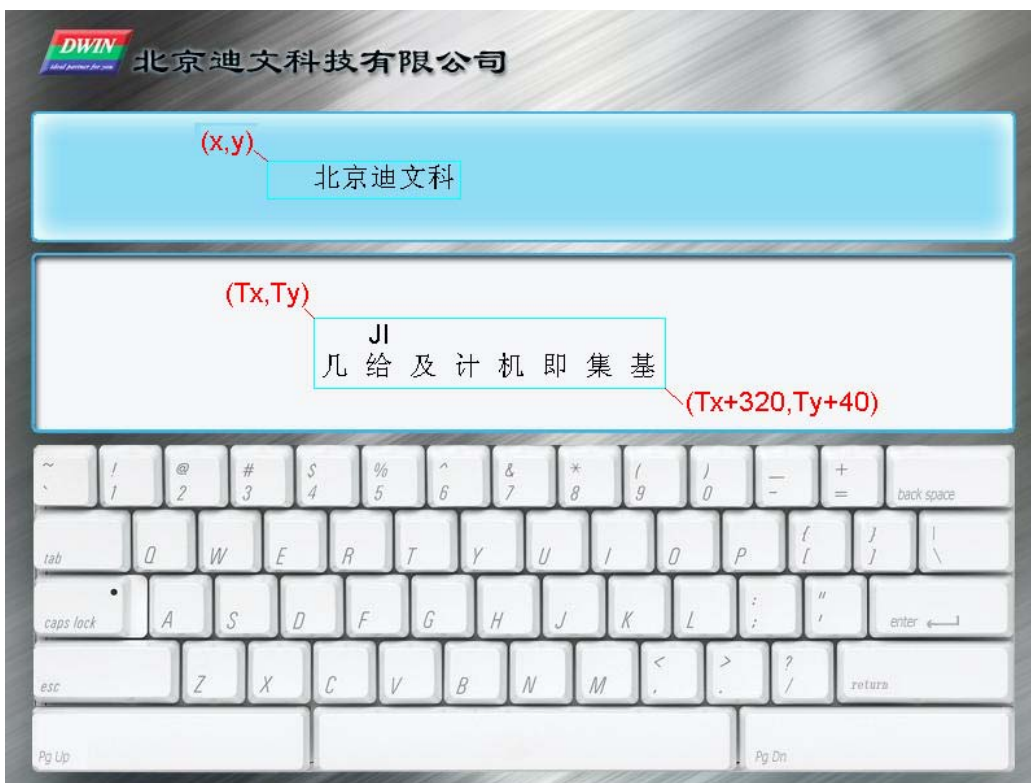
- <R_ID> 用户预设的一个两字节返回值 (0x0000-0xFFFF), 用于用户软件进行返回参数识别;
- <VP_ID> 输入法使用的虚拟触控页面编号, VP_ID=0xFFFF 表示使用当前页面;
- <X, Y> 文本输入显示的左上角地址, 由于文本显示背景色是从 (X, Y) 地址读取, 所以实际显示地址是从 (X+1, Y) 开始的, 文本显示时使用右对齐方式;
- <Str_Max_Num> 输入字符串的最大长度, 0x01-0x40, 最多 64 个英文字符或者 32 个中文字符;
- <Str_Scale> 输入字符串的格式控制
 <Str_Scale.7> 保留, 写 0。
 <Str_Scale.6> 0=输入无初始值; 1=输入有初始值。
 <Str_Scale.5-.0> 保留, 写 0。

中英文混合输入方式下, ASCII 码显示为 8*16 点阵, 汉字显示为 16*16 点阵。

- <Str_Color> 输入字符串的显示颜色, 使用 65K 颜色模式;
- <T_Color> 中文列表显示的颜色, 使用 65K 颜色模式;
- <Tx, Ty> 中文列表显示区域的左上角地址, 由于显示背景色是从 (Tx, Ty) 地址读取, 所以实际显示地址是从 (Tx+1, Ty) 开始的, 中文列表显示固定使用 16×16 点阵, 每列 8 个, 为了触摸选择方便, 每个汉字实际占用 40×40 点阵的显示空间。
- <[Init_V]> 预设的初始值, GBK 编码; 设定初始值时必须同时设定 <Str_Scale.6>=1。
- <Str_Number> 实际返回的字节数目
- <String> 返回的字符串, 中文使用 GBK 编码方式

此输入方式下, Enter (0xF1) 将直接输入 ASCII 字符;特殊字符”BD”将选择输入全角标点。

结束返回条件: ESC (0xF0, 返回空值)、Return (0xFD)



3.29.3 强制退出输入法状态 (0x7C00)

Tx: AA 7C 00 CC 33 C3 3C

Rx: 无

注意:

- 在使用 0x7C 指令期间, 触摸屏将不再传给用户数据, 直到输入结束; 但触摸屏输入期间, HMI 可以执行其它不涉及触摸屏操作的指令。
- 如果虚拟页面不是当前页面, 那么不能设置按钮动画效果。

4 HMI 软件升级方法

工具

- 直流稳压电源壹台，输出电压调整到合适值；
- 串口线壹条；
- 带硬件串口，安装有串口调试助手 SSCOM3.2 软件的计算机壹台；

更新步骤说明

- a. HMI **关电**，把串口跟计算机 COM1 连接；
- b. 打开 SSCOM3.2 软件，点击 **打开文件** **选择 HMI 的程序**，比如 M600_V40.BIN；
- c. 在发送栏写上“**DWIN_M600_BOOT!**”，设置**定时发送**时间为“10”；
- d. 勾上“**发送新行**”和“**定时发送**”，然后给 HMI 上电；
- e. 串口会收到“Erase”，收不到检查串口是否连接好，或者 HMI 损坏；
- f. 大约 1 秒，收到“Please Tx File!”后，勾掉“自动发送”，然后点击 **发送文件**；
- g. 等待 3-10 秒，串口收到“*****END*****”表示下载完成；
- h. 给 HMI **掉电**，软件升级成功；



给 HMI 上电前的串口界面



5 修订记录

日期	修订记录	内核版本	文档版本
2008.12.01	首次发布。	--	Ver1.0
2008.12.09	修订了配置文件的使用，增加了键控界面的说明。	--	Ver1.1
2008.12.11	在 0x98 指令增加了几款特殊字体。	--	Ver1.1
2008.12.15	增加了 0xC104 指令；修改了 0xC103 指令，增加了 Y 轴比例放大功能。	--	Ver1.1
2008.12.25	统一了 T 和 S 系列指令集；增加了 0x9B 指令读取时钟功能。	--	Ver1.2
2009.01.19	增加了 0x5E 指令对背光的触控或键控功能； 增加了 0xE0 指令 PARAL.4 的定义：背光控制开启。	--	Ver1.3
2009.02.02	增加了 0x9B 指令对农历的支持。	--	Ver1.3
2009.02.12	增加了触控界面操作中对批处理指令的支持 (0x1A 配置文件)。	--	Ver1.3
2009.03.13	增加了 0x64 区域填充指令。	--	Ver1.4
2009.04.07	增加了 0x5704 圆弧显示指令。	--	Ver1.4
2009.06.29	增加了 0x9C 指令，实现了纯色背景的透明图标叠加； 增加了 0xC106 指令，是对 0xC105 指令的补充：正负双向曲线，窗口限制。	--	Ver1.5
2009.07.16	修改了 0x9A 指令和 0x1C 配置文件说明，M600 实现定时自动循环执行指令组，方便用户设计动画等。	--	Ver1.5
2009.07.23	修改了 0x1A 和 0x1E 配置文件的定义，以使用户指令扩展到 1024 条	--	Ver1.5
2009.08.07	增加了 0xC110 指令，方便低波特率下实现同步显示	--	Ver1.5
2009.08.08	修改了 0x79、0xE0 指令，增加了蜂鸣器的触摸伴音控制和蜂鸣器软件控制	--	Ver1.6
2009.09.03	增加了 0x7A 指令来切换 HMI 和 Video 模式	--	Ver1.6
2009.09.17	增加了 0xC2 指令来回读暂缓冲区数据，可以提供用户 RMA 的 RAM 功能。	--	Ver1.6
2009.12.27	增加了 0x7B 指令来对用户图片进行校验，确保下载成功。	--	Ver1.6
2010.01.04	由于产品使用内核的调整，修改了 0xE0 指令中<TFT_ID>对应的产品列表； 修改了触摸屏校准指令 (0xE4) 的 3 个校准点位置。	Ver4.5	Ver1.7
2010.02.21	增加了 0x98 指令中对 C_Mode.5 的定义，以支持文本纵向自动换行显示。	Ver4.5	Ver1.7
对于 128MB 版本的 HMI 内核，升级到 Ver5.0 以上版本 HMI 软件后，需要重新下载字库和图片！			
2010.03.06	在 0x00 指令应答中，增加了上传当前显示图片的 ID； 增加了 0xE0 指令中对 Paral.2 的定义，用来整合 0° 和偏转 90° 内核； 串口不再支持帧超时方式，只支持帧结束符。	Ver5.0	Ver1.8
2010.06.08	在 0x98 指令中，增加了 C_Mode.4 的定义以支持 ASCII 字符的间距自动调整	Ver5.1	Ver1.8
2010.07.01	增加了 0x9D 指令，功能和 0x9C 相同，只是不自动恢复当前图片背景； 增加了 0xD0 指令，强制刷新一次全屏显示； 增加了 0xB004 指令，扩展拼音输入法到 GBK 字库。	Ver5.2	Ver1.9
2010.07.15	增加了触控上传指令 0x79，增加了对 Paral.1、Paral.0 位的定义，以支持触控模式下按住触摸屏时自动上传触控代码 (0x79) 和点击有效触控区域蜂鸣器才有伴音。	Ver5.3	Ver1.9
2010.08.02	取消了 0xE0 指令修改 TFT_ID 的功能，以避免用户设置错误导致显示异常； 增加了 0xC107 置点缓冲区操作指令，用于图层操作。	Ver5.3	Ver1.9
2010.08.09	增加了 0xC108 指令，用于方便的进行多参数显示。	Ver5.3	Ver1.9
2010.08.23	增加了 0x78 指令，用于快速增量连线；增加了 0x98 指令的 64×48 字体。	Ver5.3	Ver1.9
2010.10.20	增加了文本框限制/取消指令 0x45，以方便用户在指定区域显示文本段落； 修改触摸屏校准提示点为十字交叉线； 0x9B 时钟显示指令增加了 24×48，32×64 两种字体。	Ver5.3	Ver2.0
2011.01.27	0x7A 指令增加了对基于 SD 卡的视频播放 HMI 的指令支持；	Ver5.5	Ver2.0
2011.05.11	增加了 0x73 双色位图填充指令；0x98 指令增加了对大点阵字符的支持。	Ver5.5	Ver2.0
2011.07.01	修改了 0xE0 指令：不能修改 TFT_ID，掉电后参数不保存，增加了背景自动恢复； 增加了高速 (6.225 或 12Mbps) 图片和字库下载功能；	Ver6.0	Ver2.1
2011.07.15	增加了 0x7C 触摸屏输入法指令，方便用户通过触摸屏输入中英文文本或参数。	Ver6.1	Ver2.1
2011.08.01	增加了 0x9E 图标旋转剪切指令，方便用户透明旋转指针来实现模拟表盘显示的效果。	Ver6.1	Ver2.2
2011.10.18	对 0x7C 触摸屏字符输入指令，修改 Str_Scale 参数定义以支持密码输入和输入初始值设置。	Ver6.2	Ver2.2



对于本文档或迪文 HMI 使用的任何疑问, 欢迎 mail 给 HMI 产品线技术支持专用邮箱:

dwinhmi@263.net

或致电免费服务电话 400 018 9008

有关迪文 HMI（智能显示终端）的最新资料, 欢迎访问我们的网站:

www.dwin.com.cn

感谢大家一直以来对迪文的支持, 您的支持是我们进步的动力!
谢谢大家!