目　录

1 终端正常工作的硬件基础

1.1 选择合适的供电电源

1.1.1 迪文终端两种功耗标注方式的区别

标注方式 A 举例：DC7-28V　5VA

采用这种标注方式，说明：

a.对应的终端必须使用直流电源工作；

b.工作电压范围是 7-28V，即在这个范围内任何电压，终端均可以正常工作；

c.5VA 说明终端的功耗是 5VA，基本上是恒功率工作，选择电源**功率**一般比额定值大 20%就可以了。

对应这款终端，可以选择 9V 6W、12V 6W 或者 24V 6W 的电源给终端供电。注意供电电压不同时，电流会不同（9V 为 560mA，12V 为 420mA，24V 为 210mA）。

标注方式 B 举例：DC7-15V　150mA

采用这种标注方式，说明：

a.对应的终端必须使用直流电源工作；

b.工作电压范围是 7-15V，即在这个范围内任何电压，终端均可以正常工作；

c.150mA 说明终端的电流消耗是 150mA，基本上是恒电流工作，选择电源的**电流**一般比额定值大 20%就可以了。

对应这款终端，可以选择 9V 200mA 或者 12V 200mA 的电源给终端供电。注意供电电压不同时，功率会不同（9V 为 1.1W，12V 为 1.8W）。

对于 A 型终端，一般功耗比较大，我们选择靠近上限电压的电源供电，以降低供电电流，降低线路损耗；同时比较高的供电电压，抗电网电源波动的能力也会强一些（电网干扰往往是欠压形式）。

对于 B 型终端，我们一般选择靠近下限电压的稳压电源供电，以降低功耗，减少终端本身的发热，由于工作电压靠近下限，推荐采用开关电源供电以提高抗电网干扰能力。

1.1.2 抑制电源干扰

尽管迪文的终端产品在电源上已经做了大量处理，但是，在有些工业现场有强干扰的情况下（比如电源接地错误、特大工频炉辐射、感应雷干扰等），终端的抗干扰能力还是有限，需要客户自己选择以下推荐的几种处理方法之一或组合使用来提高抗电源干扰能力：

a.尽可能把整个电路系统和干扰源共地（等电位），但不要让干扰电流从本机地环路中流过；

b.供电时，尽可能把功率大、干扰大的电路放在前级；

c.如果有条件，使用独立的电源供电；

d.在市电侧，使用 1:1 的隔离变压器做电源隔离；

e.在线路上，组合使用气体放电管、压敏电阻和 TVS 管来吸收干扰；

f.尽可能使用电阻吸收的方式，而不是电容或者电感滤波的方式来滤除地环路上的干扰；

g.注意，在直流电源上串电感滤波时，如果电源波动很大，电感的自感（或和临近电感的互感）电动势会带来很大的干扰，尤其对 CPU、存储器等电源敏感器件。这种情况下，把电感换成一个电阻，滤波效果要更好。

1.1.3 电源选择不合适的常见故障

- 终端不断复位重启；
- 终端工作一会儿后黑屏，断电凉一会后上电又可以工作一段时间（多见于使用功率不够的铁心变压器降压、整流滤波的线性电源供电）；
- 使用 DC/DC 电源模块（比如 Vicor 公司相关产品）供电，由于这些模块一般都是直接对 220V 整流后开关稳压处理，如果接地处理不好，会使输出地线带电（220V 或 110V），对共地的设备没有影响，但一旦地线故障或者有真正的大地接上地线（比如用示波器探头的负极去夹持地线）时，就会放电对设备造成严重损坏（比如存储器内容丢失、CPU 烧毁等）；

1.2 串口的连接

1.2.1 PC 机 DB9 串口（针式连接器）针脚定义

针脚	定　义	数据方向	说　明	和迪文 232 终端连接
1	DCD	输入	载波检测	——
2	RXD	输入	接收数据	DOUT
3	TXD	输出	输出数据	DIN
4	DTR	输出	数据终端准备好	——
5	GND	GND	公共地	GND
6	DSR	输入	数据设备准备好	DTR（BUSY）
7	RTS	输出	请求传送	——
8	CTS	输入	清除以传送	——
9	RI	输入	振铃指示	——

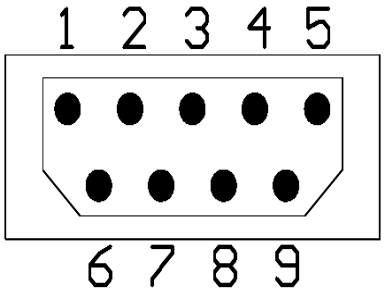


表 1-2-1

注：

- 表 1-2-1 中，“输入”表示数据输入到 PC，“输出”表示数据从 PC 输出。
- 当两个 RS232 串口设备数据线连接正确时，用万用表电压档测量，TXD 和 RXD 数据线应该都是负电压；
- RS232 串口空载时，用万用表电压档测量，数据发送数据线（TXD）应该是负电平。

1.2.2 串口电平的转换

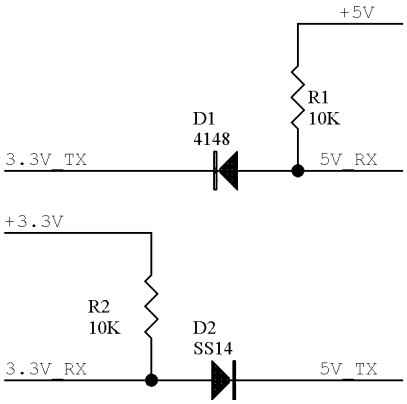


图 1-2-1

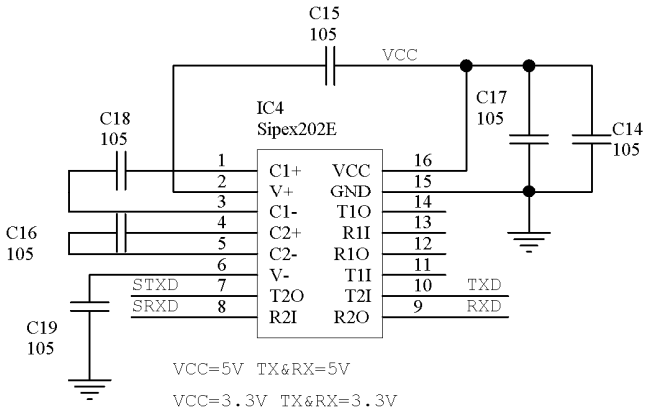


图 1-2-2

图 1-2-1 是 3.3V 和 5V 电平的 TTL 串口转换电路，其中 SS14 可用其它压降小于 0.3V 的肖特基二极管代替。

图 1-2-2 是 3.3V 或 5V 电平的 TTL 串口到 RS232 电平串口的转换电路。

1.2.3 RS485 接口的处理

迪文智能显示终端一般采用 RS232 接口，但是 PLC 等设备，或者信号需要远传时，往往需要使用抗干扰能力更好的 RS485 差分信号传输，这时就需要 RS232/RS485 转换电路。常见的无源 RS232/RS485 转换器和迪文显示终端的连接电路如下：

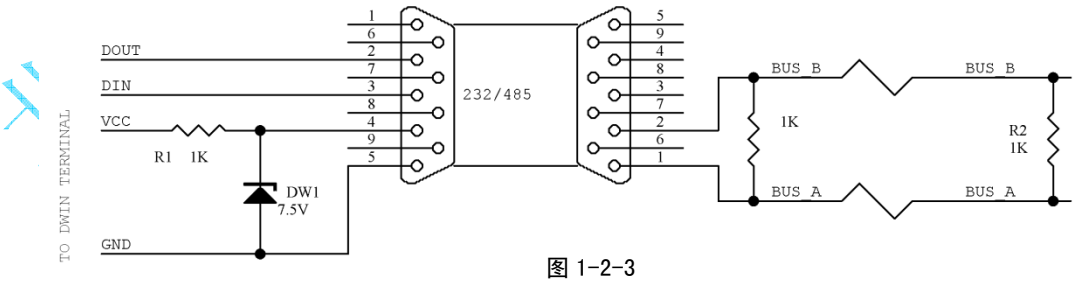


图 1-2-3

注：

- 由于迪文显示终端不允许串口窃电，所以要外供一个“窃电”电源；
- 有些 RS232/RS485 接口的 485 引脚定义可能和上图刚好相反，判别的方法是使用万用表电压档测量 RS485 接口 DB9 插针的 1、2 脚电位，高电位的是 A 线（485+），低电位的是 B 线（485-）。

2 基本约定

2.1 书写规范

在本文档中，约定以下书写规范：

- 用数据前加“0x”或数据后加“H”的方式表示 16 进制数据；
  比如，0xAA 或 AAH 都表示 16 进制数据 AA。
- 为了方便用户直接应用，串口指令都使用 16 进制格式书写，并不加任何标记；
  比如，AA 52 表示串口下发的两个字节 16 进制数据 0xAA 和 0x52，并且 AA 先发送；
- 用’ ’表示文本信息；
  比如’迪文 OK’表示字符串“迪文 OK”，其对应的 16 进制内码是 0xB5CF 0xCEC4 0x4F 0x4B；
- 用（x,y）表示显示屏上的坐标位置；
  比如（0,0）表示 x=0，y=0 的坐标原点。
- 迪文智能终端，字数据都采用 MSB 方式传送，所以本文档中也采用 MSB 方式，即高字节在前；
  比如，0x1234 表示串口传送时，0x12 先传送，0x34 后传送。

应用举例：

从（0，0）位置显示 32×32 点阵的汉字字符串“北京迪文科技”，指令可能会有以下几种方式表示。

方式 1：0xAA 0x55 (0,0) ‘北京迪文科技’

方式 2：

串口下发指令：AA 55 00 00 00 00 B1 B1 BE A9 B5 CF CE C4 BF C6 BC BC CC 33 C3 3C

指令	AA	55	00 00	00 00	B1 B1
含义	帧头	32 点阵文本显示指令	X 坐标 0	Y 坐标 0	北

指令	BE A9	B5 CF	CE C4	BF C6	BC BC	CC 33 C3 3C
含义	京	迪	文	科	技	帧尾

表 2-1-1

注：

- 上表中 0XB1B1 是汉字’北’的内码（GB2312 或 GBK 编码）；
- 指令中的显示坐标位置（x,y）指字符串第一个文字（“北”）的左上角位置；
  指令在终端执行后，显示结果如图 2-1-1 所示。

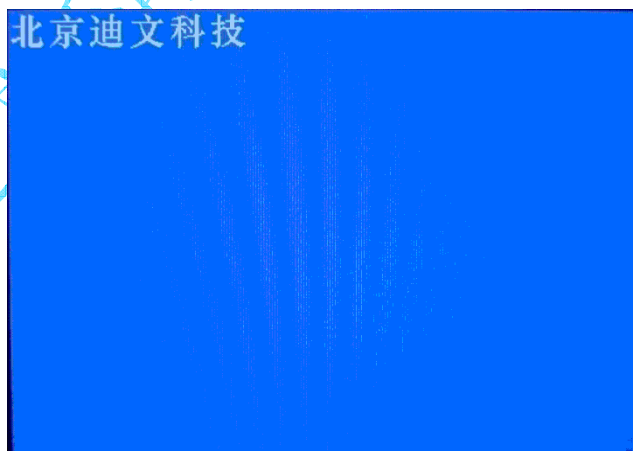


图 2-1-1

2.2 坐标系

迪文智能显示终端的坐标系与普通笛卡尔坐标系不同，而与通常的图像坐标系相同，即以可视区域的左上角作为坐标系原点，向右为 X 轴正向，向下为 Y 轴正向。如图 2-2-1 所示。

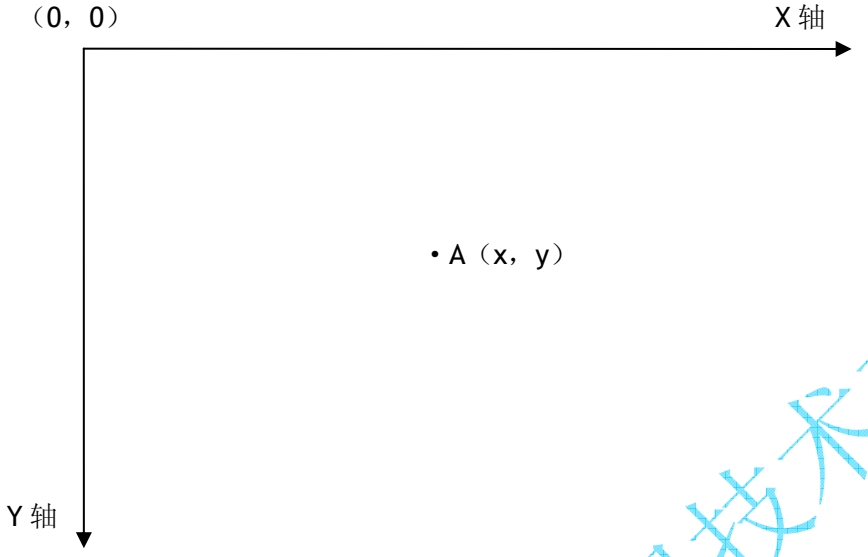


图 2-2-1　迪文显示终端的坐标系

在迪文显示终端指令中，统一以 4 个字节来表示坐标位置，X、Y 坐标分别以两个字节进行表示，坐标的表示范围为 0-65535。

例如，假设 A 点坐标为（275，412），则在迪文显示终端中用 16 进制数据表示为：

X=0x0113　Y=0x019C

2.3 颜色和调色板

2.3.1 调色板

迪文智能显示终端采用 16 位颜色模式，最多可表现出 2^{16}=65536 种颜色（65K 真彩色）。

16 位颜色的表示采用 5R6G5B 调色板模式。即每个像素点用 16 位（两个字节）来表示，其中红色分量（R）占 5 位、绿色分量（G）占 6 位、蓝色分量（B）占 5 位。

调色板高字节（高 8bit VD15-VD8）								调色板低字节（低 8bit VD7-VD0）							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R4	R3	R2	R1	R0	G5	G4	G3	G2	G1	G0	B4	B3	B2	B1	B0
红色调色板					绿色调色板						蓝色调色板				

表 2-3-1 调色板定义

比如：深红＝0xF800　深绿＝0x07E0　深蓝＝0x001F

2.3.2 前景色与背景色

前景色为进行图形操作时点、线、图形所显示的颜色，即指令集中的 COLOR，上电默认白色（0xFFFF）。

背景色即指令集中的 BKCOLOR，上电默认为蓝色（0x001F）。

图 2-3-1 中文字显示的白色即为前景色 COLOR，背景的蓝色即为背景色 BKCOLOR。

图 2-3-1

2.4 迪文 HMI 指令集

类　别	指令	说　明		T 系列	S 系列	K 系列
握　手	0x00	查看配置和版本信息		√	√	√
显示参数配置	0x40	设置调色板	调色板位数： T=16bit 65K 色 S=16bit 65K 色 K=18bit 262K 色	√	√	√
	0x41	设置字符显示间距		√	√	√
	0x42	取色到背景色调色板		√	√	√
	0x43	取色到前景色调色板		√	√	√
	0x44	设置光标显示模式		√	√	√
文本显示	0x53	8×8 点阵 ASCII 字符		√	√	√
	0x54	16×16 点阵 GBK 字符串显示		√	√	√
	0x55	32×32 点阵 GB2312 字符串显示		√	√	√
	0x6E	12×12 点阵 GBK 字符串显示		√	√	√
	0x6F	24×24 点阵 GB2312 字符串显示		√	√	√
	0x98	任意点阵，任意编码字符串显示		√	√	√
置　点	0x50	背景色置多个点（删除点）		√	√	√
	0x51	前景色置多个点		√	√	√
	0x74	动态曲线快速置点		√	√	√
	0x72	直接显存操作		√	√	√
线段和多边形	0x56	连接多条线（多边形）		√	√	√
	0x6D	删除多条线（多边形）		√	√	√
	0x75	快速显示连续的同底垂直线段（频谱）		×	√	√
	0x76	快速显示折线图		×	√	√
圆弧和圆域	0x57	反色/显示 多个圆弧或圆域		√	√	√
矩形框	0x59	显示多个矩形框		√	√	√
	0x69	删除多个矩形框		√	√	√
区域操作	0x52	清屏		√	√	√
	0x5A	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	显示保存在终端中的一幅全屏图像	存储器： T=96MB S=96MB K=992MB	√	√	√
	0x71	从保存在终端的一幅图片剪切一部分显示		√	√	√
	0xE2	将当前显示画面保存到终端中		√	√	√
	0x99	用户自定义图标显示		√	√	√
动画支持	0x9A	打开/关闭用户预先设置的动画显示		×	×	√
暂存缓冲区	0xC0	写数据到暂存缓冲区		×	√	√
	0xC1	01=显示暂存缓冲区预置的数据点		×	√	√
		02=显示暂存缓冲区预置的数据线		×	√	√
		03=使用暂存缓冲区的数据点连线（曲线动态缩放）		×	√	√
数据库操作	0xF2	修改字库		√	√	√
	0x90	写数据到用户数据库（32MB）		×	√	√
	0x91	从用户数据库读数据（32MB）		×	√	√
键盘操作	0x71	键码上传		×	√	×
	0xE5	配置键码接口		×	√	×
触摸屏操作	0x72	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	触摸屏校准		×	√	√
	0x78	触控界面自动切换模式下，预设键码自动上传。		×	√	√
蜂鸣器控制	0x79	蜂鸣器鸣叫一声（100mS）		×	×	√
背光控制	0x5E	关闭背光		√	√	√
	0x5F	打开背光或 PWM 方式调节背光亮度		√	√	√
时钟操作	0x9B	启用/关闭时钟自动叠加显示		×	×	√
	0xE7	设置时钟		×	×	√
参数配置	0xE0	配置用户串口速率、触摸屏数据上传格式		√	√	√
实用算法	0xB0	01=基于一级字库的拼音输入法		×	√	√
		02=计算(A×B+C)/D		×	√	√
		03=无符号整数（2 字节）数组排序		×	√	√
声音操作	0x30	播放指定存储位置的音乐		×	×	√
	0x32	实时音量调节		×	×	√
	0x33	立即停止播放		×	×	√
	0x3F	声音操作指令应答		×	×	√
配置文件操作（简易 OS）	触控界面自动切换（0x1E 字库文件）			×	√	√
	自动动画播放（0x1C 字库文件）			×	×	√
	图标字符定义（0x1D 字库文件）			×	√	√
	用户指令配置文件（0x1F 字库文件）			×	×	√
软件升级	串口在线升级内核软件			√	√	√

3 迪文终端演示开发助理使用说明

3.1 软件概述

3.1.1 软件用途

本软件的目的是提供使用迪文系列智能显示终端产品的用户提供一个有效的演示、评估平台，加快用户的研发进度。

3.1.2 软件运行

本软件可以运行在 PC 及其兼容机上，使用 WINDOWS XP 操作系统，需要有微软.net Framework 2.0 或以上版本基础类库的支持。本软件为绿色免安装类型，用户直接点击如图 3-1-1 所示的图标即可运行。

图 3-1-1 迪文终端演示开发助理的安装和运行

3.1.3 系统配置要求

硬件：奔腾 IV 及以上 CPU，256M 以上内存的 PC 及其兼容机，。

操作系统： WINDOWS 2000/2003/XP

通讯方式：下位机为迪文系列显示终端产品，通过 RS-232 串口或 USB 与 PC 相连。

其他：软件配置文件为“Terminal.ini”，需与本软件可执行文件放在同一文件夹下，如图 3-1-1 所示。

3.1.4 软件安装

直接点击解压软件的压缩包，把可执行文件和配置文件放在同一文件夹下，然后点击“DWIN_DA_V2.exe”运行软件。

3.2 软件主界面

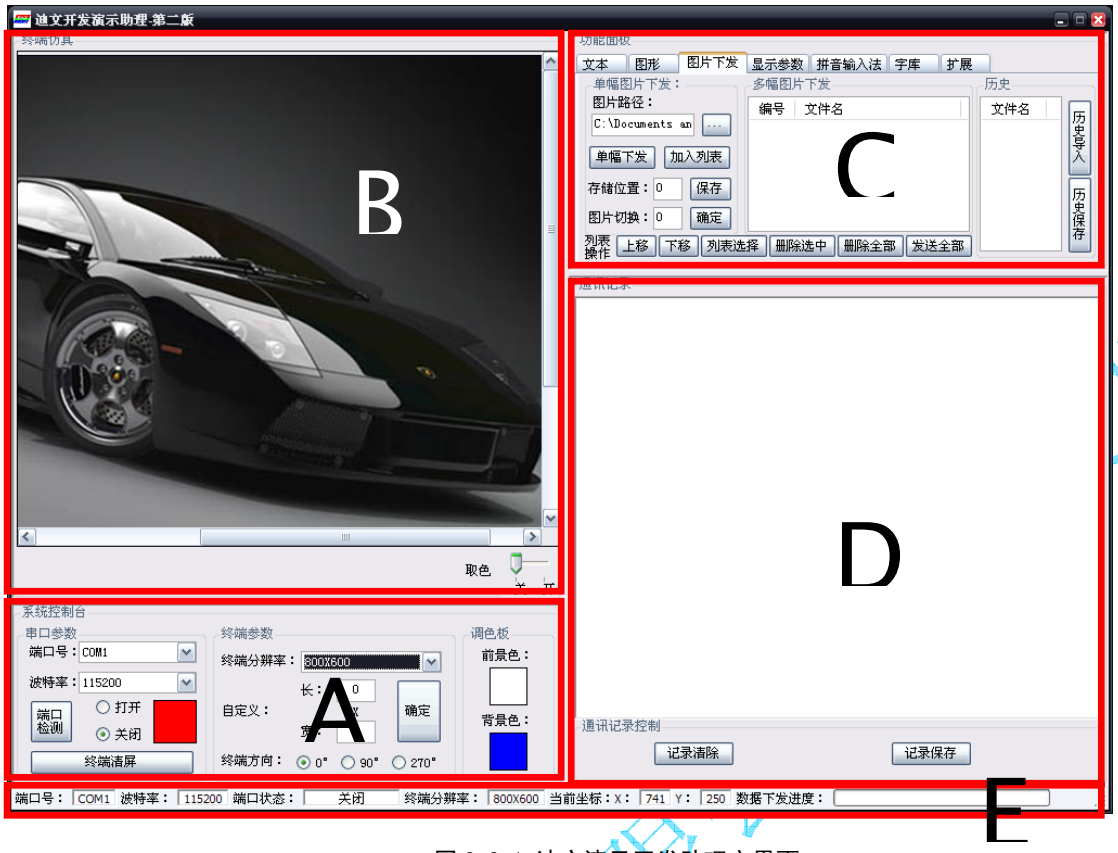


图 3-2-1 迪文演示开发助理主界面

如图 3-2-1 所示，主界面共分五个部分，分别是：

A：系统控制台；

B：终端仿真；

C：功能面板；

D：通讯记录；

E：状态栏。

下面详细介绍各部分功能和操作方法。

3.3 系统控制台

图 3-3-1 系统控制台

系统控制台负责控制软件与终端的通讯连接、设置全局参数等。

➢ **串口参数**

在串口参数中，端口号下拉列表列出了当前 PC 端可以打开的全部串口端口号。在打开串口与终端建立连接前请先选择正确的串口端口号。若不能确定当前所使用的串口，请点击 windows“开始”->“控制面板”->“性能和维护”->“系统”，在“硬件”选项卡的“设备管理器”的“端口（COM 和 LPT）”中查看，如图 3-3-2 所示。

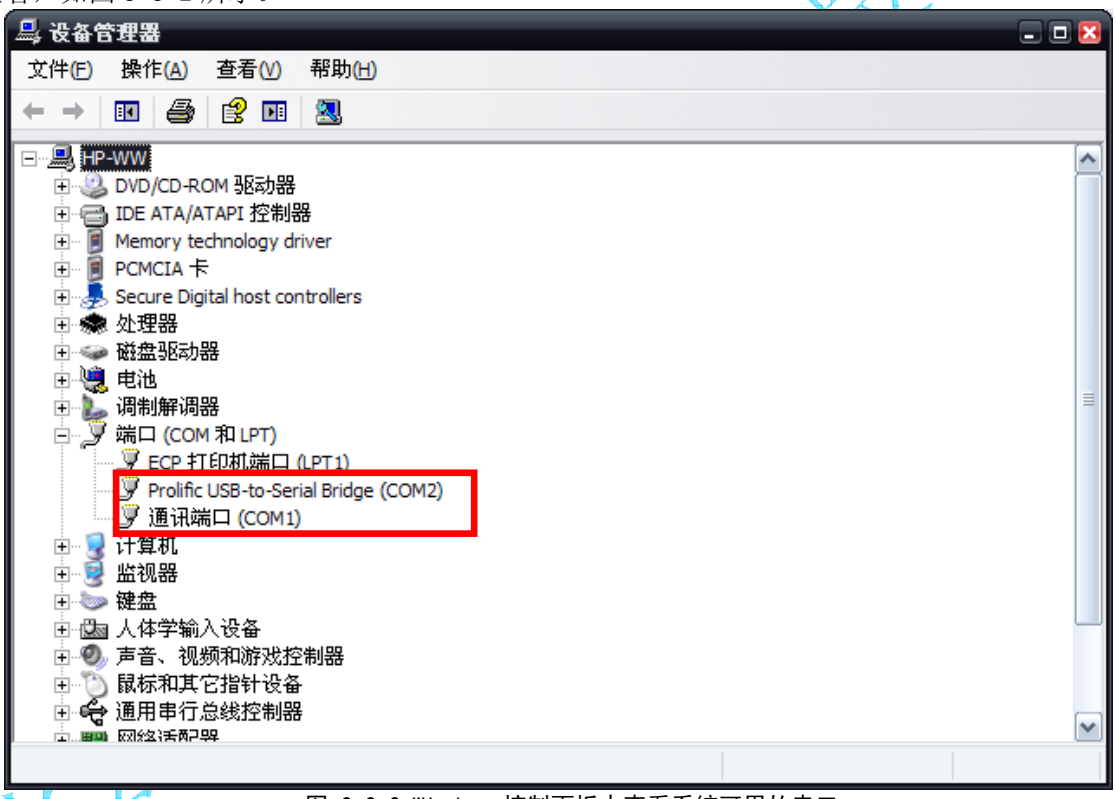


图 3-3-2 Windows 控制面板中查看系统可用的串口

波特率下拉列表列出了终端所支持的全部可能的波特率，在打开串口与终端建立连接前请先选择正确的串口波特率。（迪文终端出厂缺省波特率为 115200 bps）；

端口检测按钮用于在端口发生变化时（如插入 USB-RS232 转换接口），查找当前 PC 可打开的全部串口端口，并显示在串口号下拉列表。

迪文智能显示终端的 USB 接口使用了 Silicon_LABs 公司的 CP2102 接口 IC，相关驱动可以在 www.silabs.com 网站上下载。

在端口号和波特率设定完毕后，可打开串口。此时软件将主动向终端发送握手指令，若顺利收到终端的回复，表明通讯链路建立，此时串口状态指示灯将变为绿色，在状态栏相应位置显示“打开”。如图 3-3-3 所示。

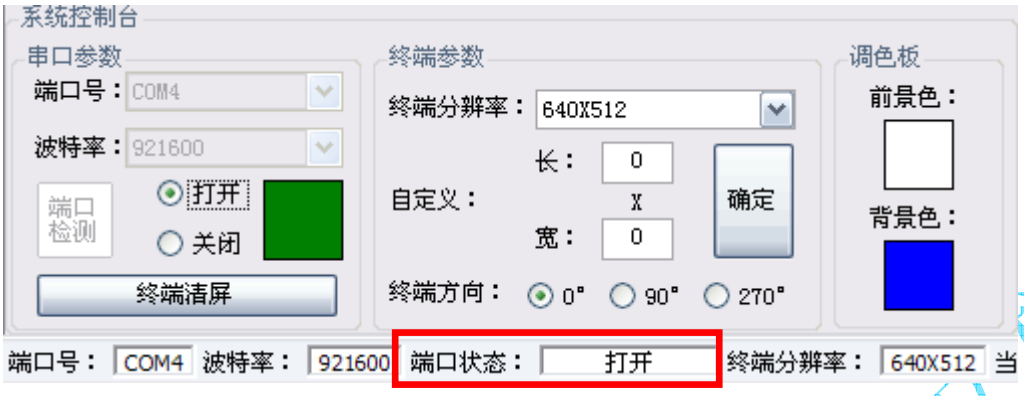


图 3-3-3 串口打开并和终端握手正常的界面

若未收到终端的回复，说明通讯链路有问题（如连接线故障、波特率不匹配等）。串口仍将打开，但串口状态指示灯显示为黄色，状态栏显示“未通过握手验证”。如图 3-3-4 所示。

图 3-3-4 串口打开正常但和终端握手失败的界面

- **终端参数**

迪文智能显示终端系列产品，因型号不同所采用的 TFT 屏的分辨率也不尽相同，这将直接影响到软件向终端下发图片以及图形操作的正确性，因此在使用本软件对终端进行操作前一定要选择与终端相匹配的分辨率参数。若下拉列表中未包含当前终端的分辨率，可自行添加。在自定义的输入框输入正确数值后，点击“确定”将自定义数值加入下拉列表。

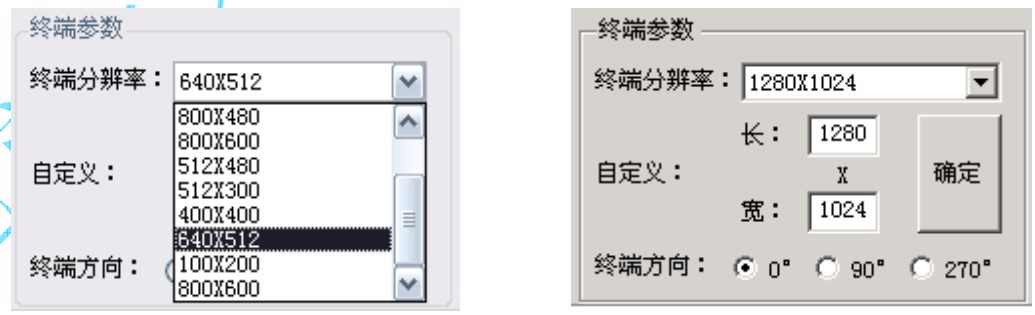


图 3-3-5 终端参数设置界面

终端方向是为满足不同用户对图片下发的特殊需求而设置。通过设置该项参数，可免去用户对原始图片的旋转操作，直接下载。

- **调色板**

调色板功能控制终端的前景色和背景色设置。两个色块分别表示终端的前景色和背景色，双击色块可弹出颜色选择窗口，用户可选择需要的颜色并确定，软件将自动向终端发送颜色设定指令。此外，在新设定背景色之后，软件主面板的终端仿真区若尚未加载图片，则会刷新为与背景色相同的颜色。

3.4 终端仿真

终端仿真用于在最大程度上模拟迪文终端的屏幕。用户可使用鼠标在其上进行操作，如绘制点、线、圆、矩形，拾取当前点颜色，获取当前坐标等。

➢ **获取坐标**

当鼠标在仿真区划过时，状态栏将显示对应的在终端上的坐标。在其后的文本、图形绘制等需要使用坐标定位的情况下，用鼠标点击仿真区可直接将坐标值填入。

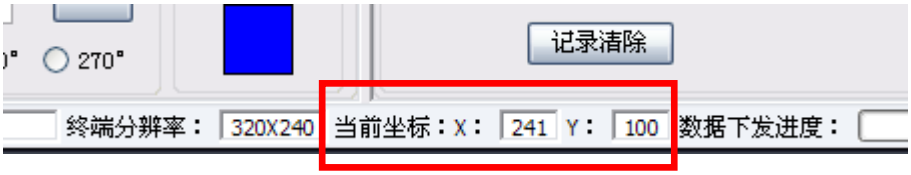


图 3-4-1 直接从屏幕上获取坐标值

➢ **获取颜色**

终端仿真区右下角有一个取色开关，当取色开关打开，且有图片加载到仿真区时取色操作才起作用。鼠标点击仿真区，将显示鼠标点击的颜色，该颜色值按迪文终端显存颜色格式为 2 字节数据（5R6G5B）。

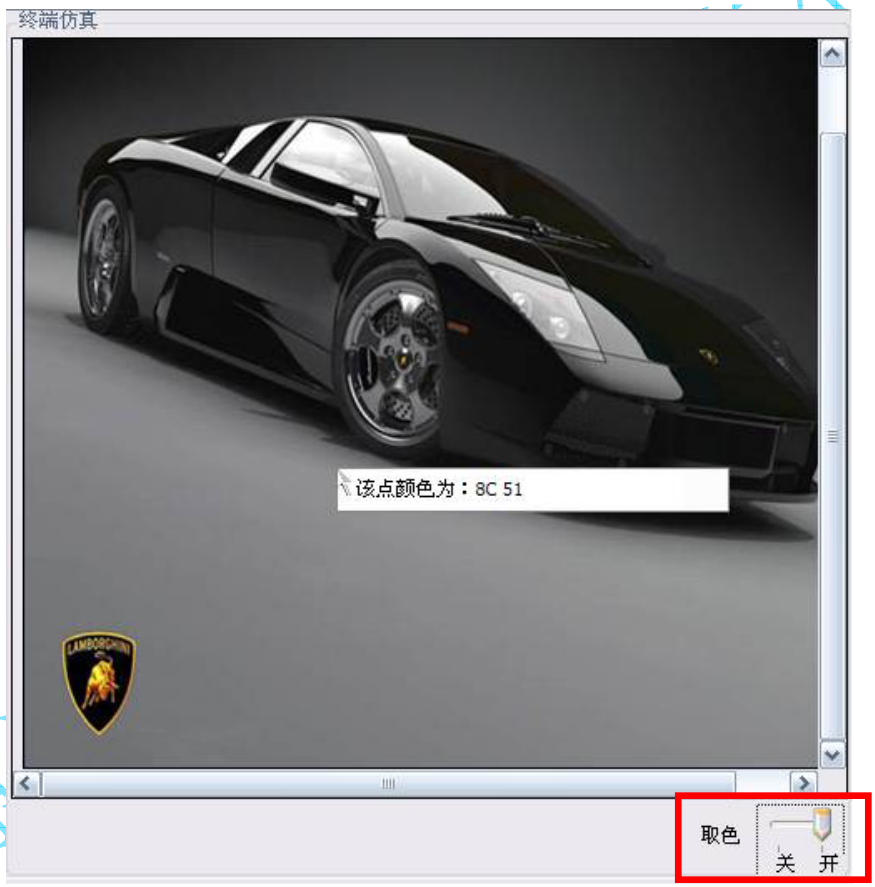


图 3-4-2 直接在图片上取色

3.5 功能面板（常用指令图示说明）

➢ 文本

“文本”功能负责对终端进行文本指令操作。迪文终端的文本指令分为标准和扩展两种模式。

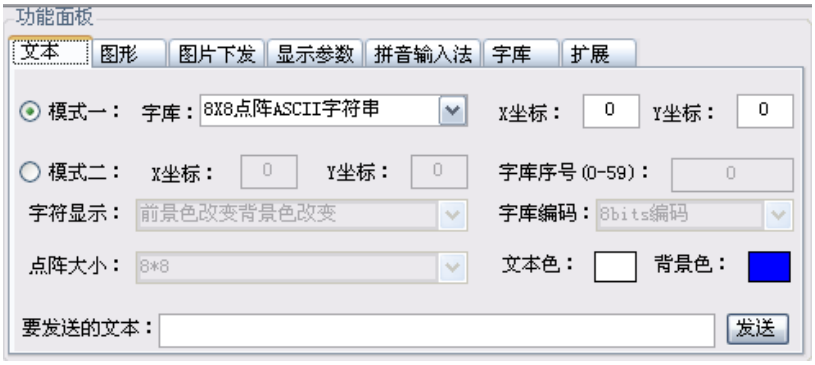


图 3-5-1

标准模式（模式一）

采用终端预置字库进行显示，用户只需选择需要的字库和坐标即可。字符显示位置的坐标可由用户自行输入也可在终端仿真区用鼠标点击获得；字符显示位置的坐标是指显示字符串第一个文字的左上角位置。

预置字库为：8×8 点阵 ASCII 字库、12 点阵 GBK 宋体、16 点阵 GBK 宋体、24 点阵 GB2312 宋体以及 32 点阵 GB2312 宋体。

标准模式使用终端调色板设置的前景色和背景色作为文本的前景色和背景色，并且前景色和背景色均显示出来。

扩展模式（模式二）

在该模式中，用户需自行指定坐标、字库编号、显示方式、编码方式、点阵大小以及文本的前景色和背景色（独立于终端调色板的前景色和背景色）。文本的前景色和背景色的设定方式同终端前景色和背景色的设定方式。

显示方式：包括“前景色改变背景色改变”、“前景色改变背景色不变”和“前景色不变背景色改变”三种方式。

以设置前景色为白色，背景色为蓝色，在迪文显示终端上显示“北京 DWIN1234”来说明 3 种方式的区别，如图 3-5-2 所示。

图 3-5-2A 前景色和背景色均显示

图 3-5-2B 前景色显示 背景色不显示

图 3-5-2A 前景色不显示 背景色显示

注意，文本输入框中，发送文本长度有限制，汉字最多 118 个，数字和英文字母最多 236 个；汉字和 ASCII 字符混排时，不要超过 236 个半角字符（1 个 ASCII 字符算 1 个半角字符，1 个汉字算 2 个半角字符）。

BIG5 码和 Shift-JIS（日文）显示

用户在使用 BIG5 编码字库显示繁体中文或使用 SJIS 编码字库显示日文片假名时，要注意以下两点：

a. 选择对应的编码字库和字符分辨率；

b. 要发送的文本应为繁体中文或日文片假名，可用输入法（如搜狗）的软键盘进行输入。

➢ 图形

“图形”功能负责对终端进行图形指令操作。包括点、线、区域操作三部分。

图 3-5-3

点操作

在指定坐标绘制一个像素大小的点，可选择以终端前景色显示或以终端背景色显示。在终端模拟区用鼠标点击即可在终端相应位置绘制（但在模拟区不会显示）。

线操作

在指定起始点和结束点之间绘制一条线段，可选择以终端前景色显示或以终端背景色显示。在终端模拟区用鼠标左键连续点击可在终端相应位置绘制连续的线段，鼠标右键单击结束绘制。

图 3-5-4 终端显示连线操作结果

注意，智能终端本身的线段显示指令，是把串口下发的多个坐标位置用线段连接（比如，下发多边形的顶点坐标即可自动连线成多边形），而演示软件仅仅只是演示了把鼠标点击的两点连线。

区域操作

包括“圆形区域显示”、“矩形区域显示”和“矩形区域移动”3 部分。

圆形区域显示

圆形区域位置确定方法

第一步：在仿真区点击第一次，确定圆形区域圆心位置；

第二步：点击第二次，确定圆形区域的边界，即确定圆的半径。

4 种圆形区域显示的不同效果（终端调色板前景色为白色，背景色为蓝色）：

图 3-5-5A 显示圆弧

图 3-5-5B 反色圆弧

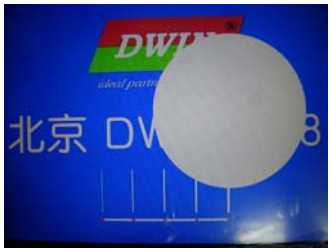


图 3-5-5C 显示圆域

图 3-5-5D 反色圆域

圆弧的边界宽度仅为一个像素，若希望加宽圆弧的边界宽度（厚度），可采用半径递增（递减）的方法，用多个空心圆组成宽边界的圆弧。

矩形区域显示

以下示例中设置终端调色板前景色为白色，背景色为红色。

显示矩形框：以终端前景色在终端上显示一个空心矩形，如图 3-5-6A 所示。

删除矩形框：以终端背景色在终端上显示一个空心矩形，如图 3-5-6B 所示。

图 3-5-6A 显示矩形框

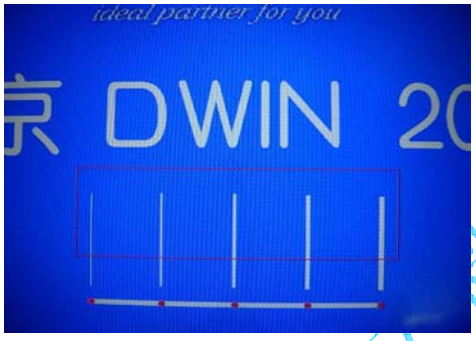


图 3-5-6B 删除矩形框

矩形框线宽为 1 个像素，若想显示宽边（粗线）矩形框，可用多个矩形框组成宽边界的空心矩形。

显示矩形域：以终端前景色显示一个实心矩形区域，如图 3-5-7A 所示。

清除矩形域：以终端背景色显示一个实心矩形区域，如图 3-5-7B 所示。

反色矩形域：在终端上反色显示一个实心矩形区域；如图 3-5-7C 所示。

图 3-5-7A 显示（填充）矩形域

图 3-5-7B 清除矩形域

图 3-5-7C 反色矩形域

矩形区域移动

左环移：指定区域进行向左环移操作（移动像素数由用户设定，取值范围为 1-16），如图 3-5-8A 所示；

右环移：指定区域进行向右环移操作，如图 3-5-8B 所示；

左移右清除：指定区域进行向左移动操作，右边空出部分由背景色填充，如图 3-5-8C 所示；

右移左清除：指定区域进行向右移动操作，左边空出部分由背景色填充，如图 3-5-8D 所示；

图 3-5-8A 左环移

图 3-5-8B 右环移图

3-5-8C 左移右清除

图 3-5-8D 右移左清除

小结：

区域移动指令可与线段绘制指令相配合用于显示动态曲线。

具体思路为：

使用左移右清除将旧曲线左移，右边清除出新的区域，在新区域中绘制新的曲线。

反复操作，可令曲线区域最右端始终为新的数据，并且曲线在“移动”。

➢ 图片下发

“图片下发”功能是本软件中最重要的组成部分，负责对智能终端进行图片下发操作。支持单幅图片下发、多幅图片下发和历史记录下发 3 种方式。

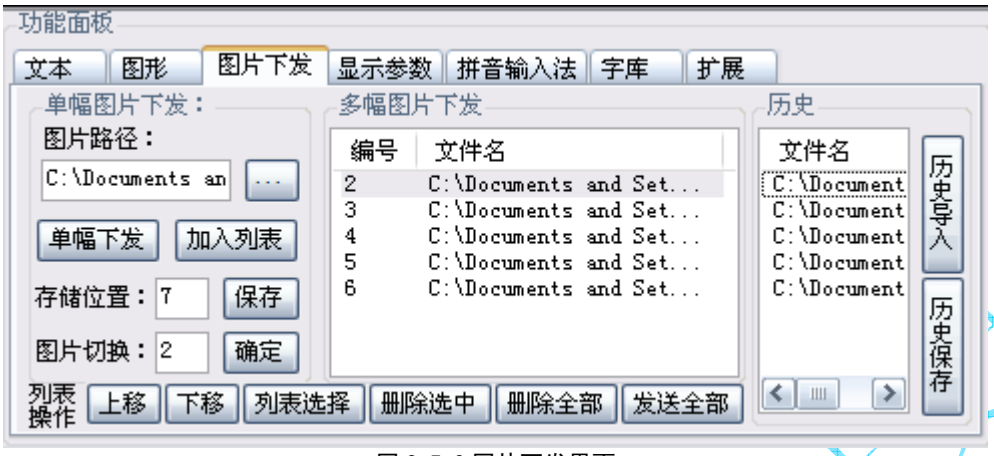


图 3-5-9 图片下发界面

单幅图片下发

在图片路径输入要下发的图片路径，或点击旁边的按钮选择图片。

点击“单幅下发”按钮，完成图片下发。注意，此时要确保串口打开并与终端正常通讯。

完成下发后，在“存储位置”输入目标存储位置号，点击“保存”按钮将刚刚下发的图片保存至智能终端。注意，单幅下发如不执行保存操作，图片将无法保存至终端。

3 种图片下发模式，如图 3-5-10 所示。

图 3-5-10A 0°（正常）下发模式

图 3-5-10B 90° 下发模式图

3-5-10C 270° 下发模式

多幅图片下发

在单幅图片下发中，选择图片后，可点击“加入列表”将图片路径加入多幅列表；也可点击“列表选择”按钮选择多幅图片，并将路径加入列表。

对于多幅列表中的列表项，可使用列表操作中各按钮对其进行操作。

多幅图片下发时图片存储位置预先指定，下发时自动保存；存储位置显示在列表中，可通过双击字段进行修改；

点击“发送全部”按钮将列表中的图片依次下发，在下发过程中软件将出现假死现象（鼠标无响应），属正常，请耐心等待，发送结束后会恢复正常。

历史记录下发

在多幅下发完成后，将自动把列表中内容更新至历史列表；

用户可点击“历史保存”按钮将本次下发内容保存至历史记录文件（*.log）；

用户如想重复以前的下发，可点击“历史导入”按钮导入以前的发送记录。前提是要保证记录的路径下图片是存在的；

单幅图片下发历史记录不保存。

➢ 显示参数

“显示参数”功能负责对终端进行显示参数设定操作。包括字符间距、触摸板校准、光标显示和背光亮度控制四部分。

图 3-5-11 显示参数界面

字符间距

设置终端上显示文本的行间距和列间距（0-127 的非负整数），输入数值后点击“设定”按钮完成设定。字符间距的设定，只改变后续文本显示的字符间距。不同行业间距的显示效果如图 3-5-12 所示。

图 3-5-12A 列间距＝0

图 3-5-12B 列间距＝10

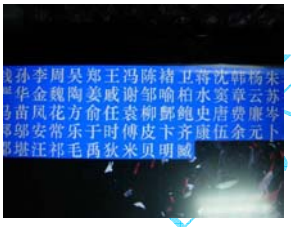


图 3-5-12C 行间距＝0

图 3-5-12D 行间距＝10

触摸板校准

点击“校准”按钮将向终端发送触摸板校准指令，然后根据智能终端提示点击不同屏幕位置对触摸板进行校准。

光标显示

对终端上显示的光标进行各参数设置，包括光标的显示和关闭，光标出现的位置，以及光标的宽和高（1-31 的非负整数）。点击“设定”按钮完成对光标的设定。光标启用后，会自动在智能显示终端上以 1Hz 的频率闪烁。

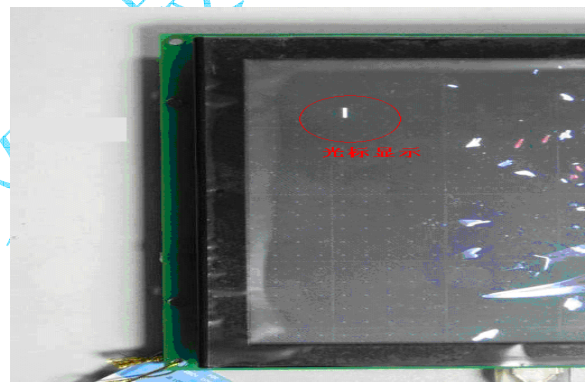


图 3-5-13 在（50，50）位置显示宽为 2，高为 10 的光标

背光亮度控制

使用“开”和“关”按钮控制智能显示终端背光的开闭或关闭。

LED 背光的智能显示终端支持 64 级背光亮度调节，可用鼠标拖动滑块进行调整。

图 3-5-14A 背光亮度为 64（满亮度）

图 3-5-14B 背光亮度为 16（25%亮度）

➢ **拼音输入法**

“拼音输入法”功能负责对终端进行拼音查询演示操作。

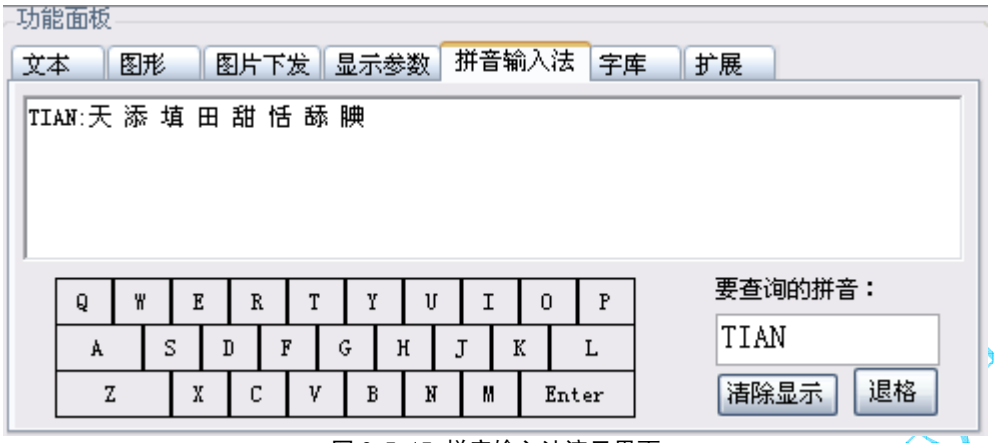


图 3-5-15 拼音输入法演示界面

在该页面板上设置了 26 个字母的模拟键盘，用户可用鼠标点击字母进行输入，也可在文本框处直接输入（不可超过 6 个字母，不支持模糊查询）。

输入完成后，点击“Enter”按钮发送查询指令，终端返回的该拼音下的汉字将在上方显示。

点击“清除显示”按钮清除显示的拼音查询结果。

点击“退格”按钮，可对文本框内待查询的拼音进行退格操作。

如图 3-5-15，输入拼音“TIAN”进行查询

串口将发送

AA B0 01 54 49 41 4E CC 33 C3 3C

串口将接收到

AA B0 01 08 CC EC CC ED CC EE CC EF CC F0 CC F1 CC F2 CC F3 CC 33 C3 3C

软件显示

TIAN:天 添 填 田 甜 恬 舔 腆

➢ **字库下载**

“字库”功能负责对终端进行字库下发操作。

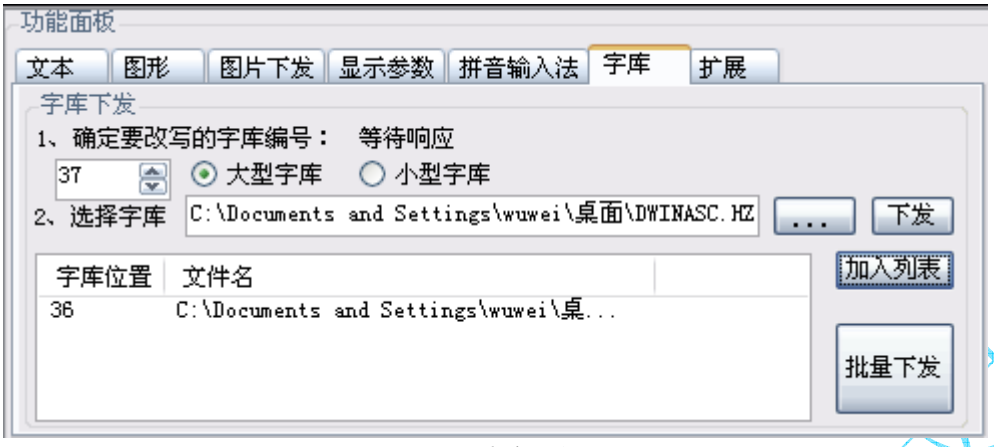


图 3-5-16 字库下发界面

迪文终端支持的字库分为大小两种：

32 个最大 128KB 容量的小字库，存储位置为 0-31（0x00-0x1F）。0 号位置为默认 8×8 ASCII 字符库，不允许修改。

28 个最大 1MB 的字库，单个字库可装下 16 点阵内的 GBK 扩展字库或 32 点阵内的 GB2312 二级字库。字库允许组合使用，最大可拼接为 1 个 28MB 的特大字库。

同图片类似，字库文件可进行单个下发或批量下发。

首先要为欲下发的字库文件指定位置，位置号根据字库文件的大小不同而不同。然后选择字库文件，点击“下发”按钮进行单个下发。或者点击“加入列表”将字库文件路径放入列表，点击“批量下发”按钮下发。

➢ **扩展**

“扩展”功能为用户提供了两项扩展的功能：

a. 发送 16 进制自写指令；

b. 查询任意颜色的 16 位（5R6G5B）编码。

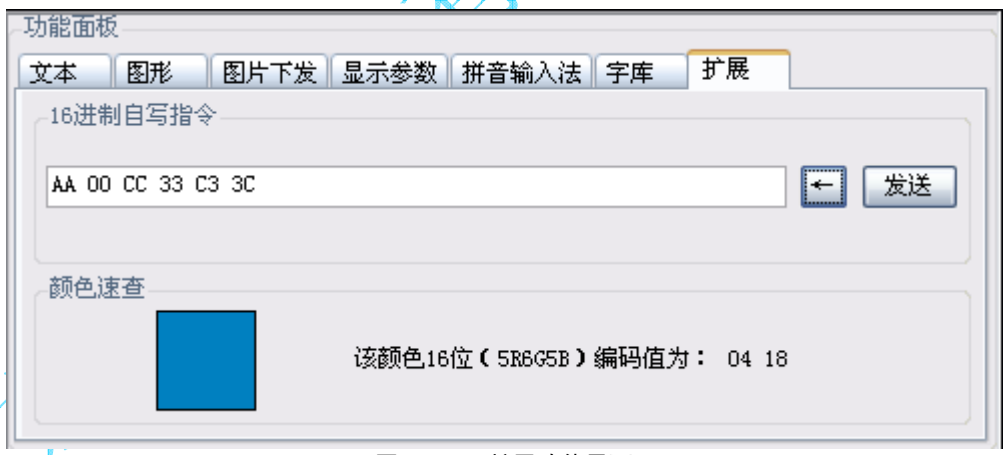


图 3-5-17 扩展功能界面

自写指令

自写指令功能用于向终端发送 16 进制指令串。

用户在文本框中填入要发送的指令串，点击“发送”按钮向终端发送（“←”按钮进行退格操作）。

颜色速查

颜色查找功能可帮助用户快速得到所需颜色的 16 位（5R6G5B）编码。

用户点击颜色框，弹出颜色选择对话框；选择所需颜色，点击确定；在右边将显示对应颜色的 16 位（5R6G5B）编码。

3.6 通讯记录

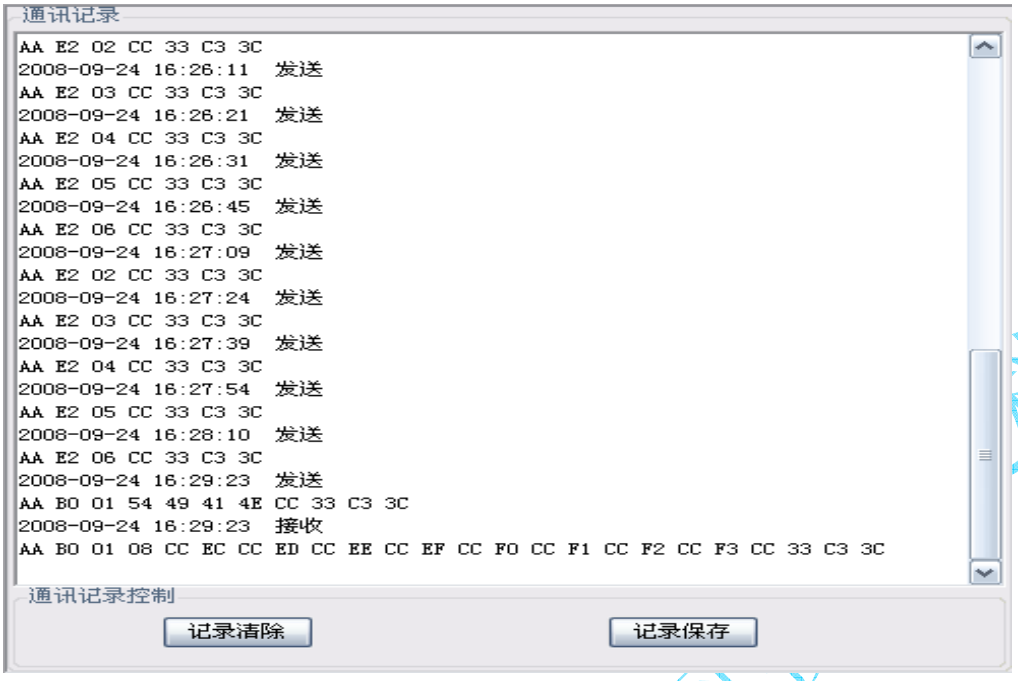


图 3-6-1 通讯记录界面

如图 3-6-1 所示，在通讯记录区中显示了软件与智能显示终端的通讯记录，包括通讯的日期、时间、发送/接收以及通讯数据。此处向用户提供了迪文终端指令的标准，用户可将指令复制到如串口调试助手中直接发送，或者作为单片机等设备开发时的参考。

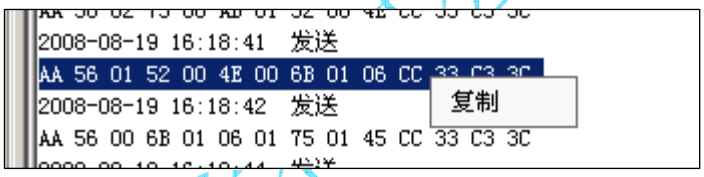


图 3-6-2 通讯记录的复制

点击记录清除按钮可清除记录区的全部通讯记录，不可撤销；点击记录保存按钮可将记录区的全部记录保存为*.log 日志文件（路径为本软件应用程序所在文件夹），以备日后查找参考使用。

3.7 状态栏

状态栏显示当前软件的串口参数、坐标、数据下发进度等信息。

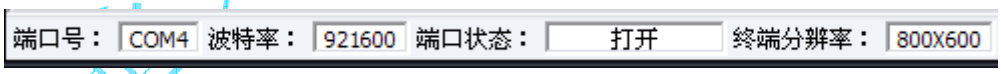


图 3-7-1 状态栏 A

图 3-7-2 状态栏 B

4 文本功能

4.1 字符编码

字符编码就是以二进制的数字来对应字符集的字符，目前用得最普遍的字符集是 ANSI，对应 ANSI 字符集的二进制编码就称为 ANSI 码（更通俗的叫做 ASCII 码），DOS 和 Windows 系统都使用了 ANSI 码，但在系统中使用的字符编码要经过二进制转换，称为系统内码。

由于 ANSI 码是单一字节（8 位二进制数）的编码集，最多只能表示 256 个字符，不能表示众多的汉字字符，各个国家和地区在 ANSI 码的基础上又设计了各种不同的汉字编码集，以能够处理大数量的汉字字符。这些编码使用单字节来表示 ANSI 的英文字符（即兼容 ANSI 码），使用双字节来表示汉字字符。在微软的主页对部分的编码有比较详细的列表，大家可以参考一下，地址：http://www.microsoft.com/globaldev/reference/WinCP.mspx。

- **GB2312 汉字编码标准**

GB2312 码是中华人民共和国国家汉字信息交换用编码，全称《信息交换用汉字编码字符集--基本集》，由国家标准总局发布，1981 年 5 月 1 日实施，通行于大陆，新加坡等地也使用此编码。

GB2312 收录简化汉字及符号、字母、日文假名等共 7445 个图形字符，其中汉字占 6763 个。GB2312 规定“对任意一个图形字符都采用两个字节表示，每个字节均采用七位编码表示”，习惯上称第一个字节为“区码（高字节）”，第二个字节为“位码（低字节）”。GB2312-80 包含了大部分常用的一、二级汉字和 9 区的符号。该字符集是几乎所有的中文系统和国际化的软件都支持的中文字符集，这也是最基本的中文字符集。其编码范围是区码 0xA1－0xFE，位码 0xA1-0xFE；汉字从 0xB0A1 开始，结束于 0xF7FE。

- **GBK 汉字编码标准**

由于 GB2312 标准表示的汉字有限，所以对其进行扩展，形成了 GBK 汉字编码标准，称为扩展码。GBK 标准高字节为 0x81－0xFE，低字节分两部分，一是 0x40－0x7E，二是 0x80－0xFE。其中和 GB2312 相同的区域，字完全相同。扩展部分大概是按部件(部首)和笔顺(笔画)从 GB13000 中取出再排列入 GBK 中。GBK 基本上把所有的汉字都包含进来了。

- **BIG5 繁体中文编码标准**

BIG5 是台湾地区的繁体中文编码标准，简称“大五码”。如不去考虑特殊符号，及后来的七个扩充字，BIG5 的排序方式如下。将所有的字分成两大群：常用字区与次常用字区，每一个字区分别用笔画来排序，同一个笔画的字，依部首来排。BIG5 码把代码表分为 89 个区， 每个字由两个字节组成，其高字节编码范围为 0xA1－0xF9,低字节编码范围为 0x40－0x7E 与 0xA1－0xFE,总计收入 13868 个字（包括 5401 个常用字、7652 个次常用字、7 个扩充字、以及 808 个各式符号)。

- **HANGUL 韩文编码标准**

HANGUL 码是韩国文字常用的一种编码，他的编码规则与我们现行的 GBK 编码规则是一样的。

- **Shift－JIS 日文编码标准**

Shift-JIS 码（即 S-JIS 码）是在 Windows 系统中比较常用的一个日文编码。它也是由两个字节组成。高字节是从 0x81-0x84，0x87-0x9F，0xE0-0xEA，0xED-0xEE，0xFA-0xFC，低字节是从 0x40-0xFC。

- **Unicode 通用字符编码标准**

因为世界各国语言文字的不同，导致了编码的混乱，给信息交互和传递带来了很大的不便（比如中国发一条短消息到美国的手机，如果编码标准不同，显示就会是一堆乱码）。国际标准组织于1984 年4 月成立ISO/IECJTC1/SC2/WG2 工作组，针对各国文字、符号进行统一性编码。1991年美国跨国公司成立Unicode Consortium，并于1991 年10 月与WG2达成协议，采用同一编码字集。目前Unicode 是采用16 位编码体系，其字符集内容与ISO10646 的BMP（Basic Multilingual Plane）相同。Unicode 于1992 年6 月通过DIS（Draf International Standard），目前版本V2.0 于1996 公布，内容包含符号6811 个，汉字20902 个，韩文拼音11172 个，造字区6400 个，保留20249 个，共计65534 个字符。UNICODE的第一个区（高字节＝0x00），与ANSI编码完全相同，即我们常说的ASCII字符表。

4.2 字库的生成和使用

迪文的智能显示终端不仅支持 GB2312/GBK 中文编码标准，也支持 BIG5、HANGUL、S-JIS 和 Unicode 字符编码标准，使文本的显示非常方便。

迪文的智能显示终端采用点阵字库，一共有 32MB 的字库空间，被分割成 32 个 128KB 的小字库和 28 个 1MB 的大字库，并且字库允许合并、组合使用。要显示出不同大小、不同字体、不同编码方式的文本，就需要涉及到字库的提取问题。在互联网上有很多基于 Windows 平台的字模提取软件，迪文智能显示终端支持最常见的字库格式，推荐使用 TS3 点阵字库生成器来生成用户需要的字库。

图 4-2-1 TS3 点阵字库生成器的界面

使用 TS3 提取 ASCII 字符时，请选择 Unicode 编码方式，并在自定义范围中选择 0000-007F 即可，如图 4-2-2 所示。

图 4-2-2 TS3 点阵字库生成器提取 ASCII 字符的界面

对于用户的特殊要求字库，比如要求数码管效果的数字符号，手指状的光标等，用户可以做成指定点阵的、单色 BMP 图片 mail 到 dwinhmi@263.net，迪文的工程师会帮你生成智能显示终端支持的字库格式以方便调用。

4.3 文本显示（printf()函数的实现）

迪文智能显示终端的文本显示指令有 6 条，如表 4-3-1 所示。

指令	显示数据	显示文本	
0x53	x+y+String	显示 8×8 点阵 ASCII 字符串	显示颜色由调色板指定，显示模式为前景色和背景色均显示。
0x54		显示 16×16 点阵汉字串（GBK）	
0x55		显示 32×32 点阵汉字串 （GB2312）	
0x6E		显示 12×12 点阵汉字串（GBK）	
0x6F		显示 24×24 点阵汉字串（GB2312）	
0x98	x+y+Lib_ID+C_Mode+C_dots+Color+Bcolor+String	用指定颜色和显示模式，显示任意编码方式、任意大小的文本。	
● 显示坐标位置(x,y)指字符串第一个文字的左上角坐标； ● 显示字符间距由 0x41 指令设置，遇到行末自动换行显示； ● String 指要显示字符的内码字符串，符合 GBK 标准； ● ASCII 字符将自动采用半角显示，0x0D、0x0A 将被处理成“回车（x=0）”和“换行（y=y+行间距）”；			

表 4-3-1 迪文智能显示终端的文本显示指令

4.3.1 固定内容文本的显示

固定内容的文本一般用来做界面提示信息，推荐直接在软件中写成完整的指令，直接用一个发送数据串的子程序发送即可,相关程序的 C 和 ASM51 参考代码如下：

```
//发送数据串的 C 程序，0xCC 33 C3 3C 为迪文智能显示
//终端的帧结束符
void String(*Str)
{unsigned char d1,d2,d3,d4;
 d1=0x00;
for(;;)
{Txbyte(*Str);  //串口发送一个字节数据
 d1=d2;
 d2=d3;
 d3=d4;
 d4=*Str;
 Str++;
 if(d1==0xcc&&d2==0x33&&d3==0xc3&&d4=0x3c)
{break;}}
}
```

```
;发送数据串的 ASM51 程序
STRING: CLR     A
        MOVC    A,@A+DPTR
        MOV     SBUF,A          ;发送到串口
        JNB     TI,$
        CLR     TI
        INC     DPTR
        MOV     D1,D2
        MOV     D2,D3
        MOV     D3,D4
        MOV     D4,A
        MOV     A,D1
        CJNE    A,#0CCH,STRING
        MOV     A,D2
        CJNE    A,#33H,STRING
        MOV     A,D3
        CJNE    A,#0C3H,STRING
        MOV     A,D4
        CJNE    A,#3CH,STRING
        RET
```

使用上面的 String()函数来显示固定内容的文本是非常方便的，比如要在屏幕的（20,100）位置显示 32×32 点阵的文本’Wecome to 迪文科技’，即使以汇编代码，也可以这样简单的实现。

```
        MOV     DPTR,#STR1
        LCALL   STRING

STR1:   DW 0AA55H,20,100
        DB 'Wecome to 迪文科技',0CCH,33H,0C3H,3CH
```

4.3.2 背景图片上叠加文本显示

有些时候，我们需要在背景图片上叠加文本显示，而不要改变原来的背景，这时有两种情况：

1.要显示的文本位置，背景是纯色的。

使用 0x42 指令（取指定位置颜色到背景色调色板），然后再显示文本，这样显示出来的文本的背景色就同背景图片的底色相同，如图 4-3-1A 所示的“12:33:40”时钟显示。使用 0x42 指令取色时要注意，取色的位置要离文本显示的坐标位置向外偏一些，以避免错把文本的显示颜色取上。

2.要显示的文本位置，背景不是纯色的。

使用 0x71 指令和 0x98 指令配合来实现，先使用 0x71 指令把要显示部分的内容用原始图片覆盖，再用 0x98 指令（设置“前景色显示，背景色不显示”模式）把文本显示出来。如图 4-3-1B 所示。

图 4-3-1A 在纯色背景上叠加文本显示（12:33:40）

图 4-3-1B 在非纯色的背景上叠加文本显示（DEMO MODE）

4.3.3 变量的显示

➢ **数据变量**

迪文智能显示终端不支持对数据类型的处理，用户需要把要显示的数据转变成 ASCII 文本才能在终端上正确显示出来。比如，要把数据 127(0x7F)在终端上显示，必须要把 127 转变成 0x31 0x32 0x37 发送给终端才可以。相关程序的 C 和 ASM51 参考代码如下：

```
//数据变量的显示
 void Printn(int x,y,char n)
{ unsigned char a,b;
  Txbyte(0xAA);
  Txbyte(0x54); //显示 8*16 的 ASCII 字符
  Txword(x);    //显示位置的 x,y 坐标
  Txword(y);
  a=n/100;      //把数据转变为 ASCII 码
  Txbyte(a+0x30);
  b=(n-a*100)/10;
  Txbyte(b+0x30);
  a=n-100*a-10*b;
  Txbyte(a+0x30);
  TxEOF();      //发送 CC 33 C3 3C
}
```

```
;数据变量的显示
PRINTN:PUSH   ACC            ;要显示的数据
       MOV    DPTR,#0AA54H   ;8×16 点阵的 ASCII 字符
       LCALL  TXWORD
       MOV    DPH,PSXH        ;x 坐标
       MOV    DPL,PSXL
       LCALL  TXWORD
       MOV    DPH,PSYH        ;y 坐标
       MOV    DPL,PSYL
       LCALL  TXWORD
       POP    ACC
       MOV    B,#100
       DIV    AB
       ADD    A,#30H
       LCALL  TXBYTE
       MOV    A,B
       MOV    B,#10
       DIV    AB
       ADD    A,#30H
       LCALL  TXBYTE
       MOV    A,B
       ADD    A,#30H
       LCALL  TXBYTE
       LCALL  TXEOF           ;发送 CC 33 C3 3C
       RET
```

➢ **文本变量**

在汇编程序设计中，用前面的 STRING()函数可以很方便的实现文本变量的显示。而在 C 语言中，我们可以考虑构造一个类似标准 C 的屏幕打印函数 Prints()来实现。

```
//调用举例   Prints(0,0,"北京迪文科技有限公司欢迎您使用真彩色智能显示终端!")
void Prints(int x,int y,unsigned char *s)
{Txbyte(0xAA);     //帧头 0xAA
 Txbyte(0x54);     //0x54=16 点阵字符串，0x55=32 点阵 0x6E=12 点阵 0x6F=24 点阵 0x98=任意点阵
 Txword(x);   //发送 x 坐标
 Txword(y);   //发送 y 坐标
 while(*s)    //发送字符串内容
    {Txbyte(*s);
    s++;}
 TxEOF();}    //发送帧结束符 cc 33 c3 3c
```

4.4 文本输入（scanf()函数的实现）

在很多情况下，我们需要通过键盘或者触摸屏进行数据等文本的输入，使用迪文智能显示终端，可以很方便的实现这些操作。

下面的例子，我们利用触摸屏键盘进行温度设定，“OK”键确认，“CE”键退出。构造一个类似标准C的输入函数Scanfn()来实现，相关的C参考代码如下：

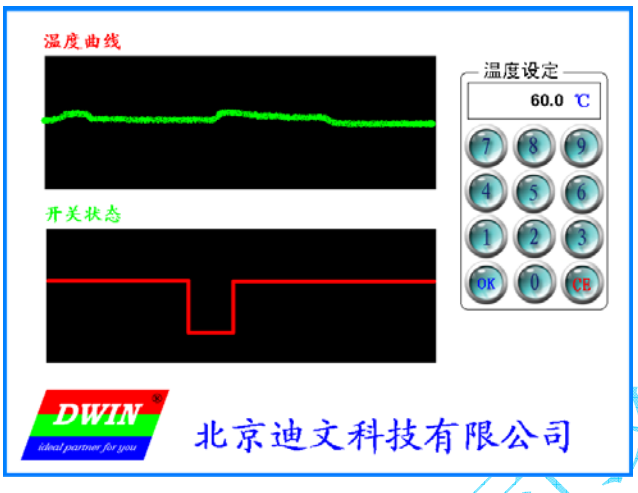


图 4-4-1 文本输入界面示例

```
//键盘输入函数，输入格式为XX.X
Scanfn()
{int n;
 char d[4],keycode;
 d[0]=0x30;
 d[1]=0x30;
 d[2]='.';
 d[3]=0x30;
 for(;;)
{if(kbhit())
{if(keycode=='E')          //确认键
{break;}
if(keycode=='C')           //取消键
{continue;}
d[0]=d[1];        //每输入一个数就左移一位，修改为n=(n*10+(keycode-0x30))就是标准输入法
d[1]=d[3];
d[3]=keycode;
prints(682,110,*d);        //显示数据
}}
n=d[0]*100+d[1]*10+d[3]; //返回数据
return(n);}
```

上面的代码示例是按照工业中应用比较广泛的“防错误输入法”设计，即当输入错误时，不用退出，重新输入一遍即可。算法通过每输入一个新数据，整个数据窗口就左移一位来实现，这种方法在人机交互中应用非常广泛，我们在后面的很多例程中会多次用到。

5 图形功能

5.1 实时动态曲线图显示

在迪文智能显示终端的应用中，我们经常会需要实时描出变量的曲线趋势图，比如温度随时间变化的曲线，心电监护仪的呼吸波曲线等。

这里，我们用显示时△y/△x 的值来定义曲线的变化幅度，把曲线按照其变化幅度的大小，分成两大类：一类是呼吸波、瞬间电流等变化幅度很大的曲线，我们称之为“大动态曲线”；另一类是温度等变化幅度不大的曲线，我们称之为“小动态曲线”。这两类曲线，在迪文智能显示终端上可以用不同的办法来完美实现，下面分别说明。

5.1.1 小动态曲线图实时显示的实现

迪文智能显示终端有一条 0x74 指令是专门为用户方便的实现小动态曲线的显示而设置的。

指令	数　据	说　明
0x74	（X+Y_S+Y_E+Bkcolor）+（Y1+Color1）+…+（Yi+Colori）	1、 Y_S 为 Y 坐标起点，Y_E 为 Y 坐标终点。 2、 以指定的颜色（Bkcolor）擦除从（X，Y_S）至（X，Y_E）的垂直线； 3、 在（X，Yi）位置置颜色为 COLORi 的点，可以同时在不同位置置多个点。 注意：并不会改变预先设置的调色板属性

表 5-1-1 0x74 指令说明

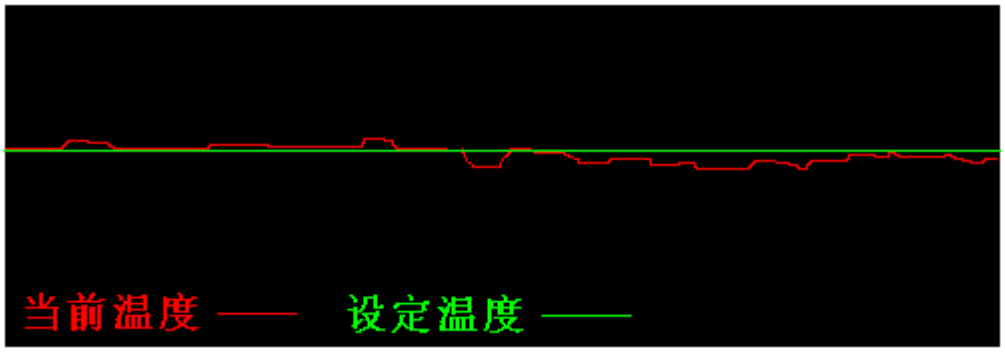


图 5-1-1 小动态曲线的实现

使用 0x74 指令实现实时小动态曲线图显示，就是把采样点当成 x 轴，不同变量采样结果作为 y 轴的不同点，通过不停的采样输出刷新（x++）来实现。下面是图 5-1-1 例程的 C 参考代码。

```
//使用 0x74 指令在迪文智能显示终端上显示温度曲线的例程
int x,t_now,t_set;
x=10;              //把位置初始化在显示窗口的左侧起始边界；显示窗口为（10,10)到（790,310）
for(;;)
{rdtmp();          //读当前温度值到 t_now
Txword(x);         //曲线的当前（采样）位置
Txword(10);        //（曲线显示窗口）y 轴起始坐标位置
Txword(310);       //（曲线显示窗口）y 轴结束坐标位置
Txword(0x0000);    //（曲线显示窗口）背景颜色为黑色
Txword(t_now);     //描当前温度位置
Txword(0xf800);    //当前温度用红色显示
Txword(t_set);     //描设定温度位置
Txword(0x07e0);    //设定温度用绿色显示
TxEOF();           //发送 0xCC 33 C3 3C 帧结束符
x++;               //下一个位置
if(x>790)
{x=10;}            //如果到达窗口的右侧边界，就把位置回复到左侧起始边界
delay(1);}          //采样延时 1mS
```

上面的例程中，通过在 TxEOF()函数之前增加其它的采样点数据（位置和颜色），可以很方便的实现多条曲线的同时实时显示。

5.1.2 大动态曲线图实时显示的实现

由于变化幅度大，大动态曲线不能再用动态描点来实现，否则这些过于离散的点很难让人看出真正的曲线模样，尤其是在多条曲线叠在一起同时显示时，情况会更加严重。

大动态曲线可以通过组合使用迪文智能显示终端的 0x56（连线）和 0x5A（区域清除）指令来实现。

指令	数　据	说　明
0x56	（X0+Y0）+…+（Xi+Yi）	把指定的点用线段连接。
0x5A	（X_S+Y_S+X_E+Y_E）$_K$	清除（背景色填充）矩形区域。（X_S, Y_S）为矩形域左上角，（X_E+Y_E）为右下角坐标。

表 5-1-2 0x56 和 0x5A 指令说明

其基本思路就是先把将要显示曲线的区域用 0x5A 指令清除，然后把最近的两个采样点用线段连接；不停的重复上面的过程，就实现了“动态”的曲线。如图 5-1-2A 和图 5-1-2B 所示。

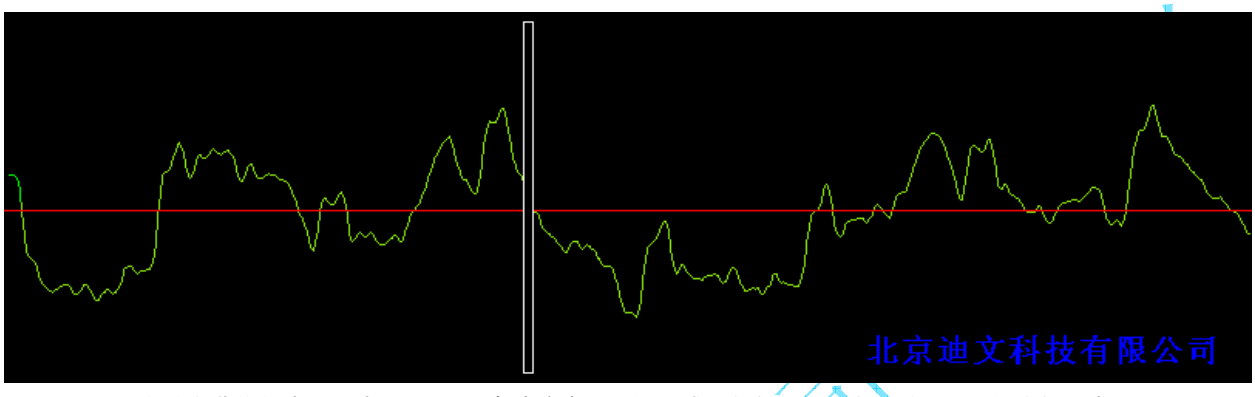


图 5-1-2A 大动态曲线的实现：先用 0x5A 指令清除将显示的区域（白色框只是为了说明问题的方便，实际看不到）

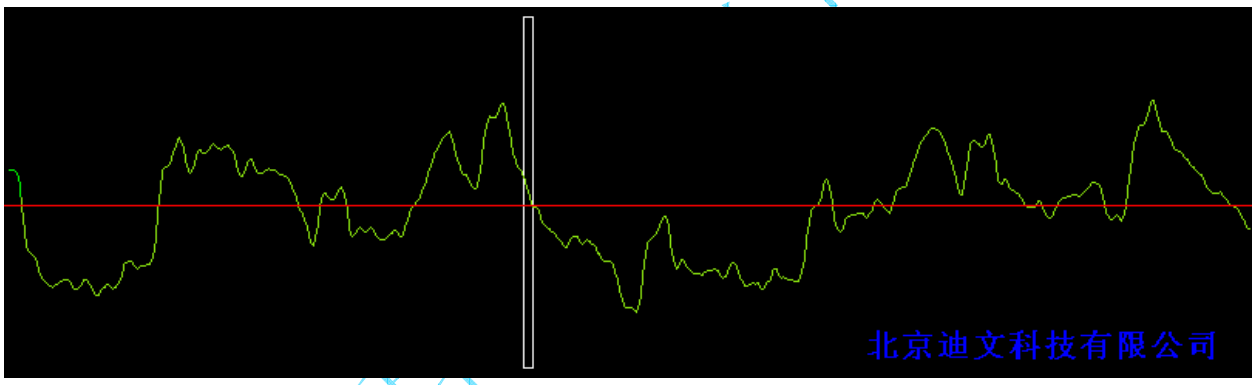


图 5-1-2B 大动态曲线的实现：用 0x56 指令连线变量的最近两个位置（白色框只是为了说明问题的方便，实际看不到）

下面是图 5-1-2 例程的 C 参考代码。

```
//在迪文 DMT80480S070_02WN 800×480 真彩色 TFT 智能显示终端上评估大动态曲线显示
int x,v1_now,v1_old,v2_now,v2_old;
x=10;                    //显示窗口为（10,10)到（790,470）
v1_old=0;                //v1 赋初值
v2_old=0;                //v2 赋初值
for(;;)
{adpro();                          //读取 A/D 采样结果到 v1_now,v2_now
clrw(x,10,x+3,470);                //使用 0x5A 指令清除将要连线的窗口;如果背景不变,使用 0x71 指令。
setcolor(0x07e0);                  //设置显示颜色为绿色
line（x,v1_old,x+3,v1_now);        //连线参数 1 的最近两个结果
setcolor(0xf800);                  //设置显示颜色为红色
line（x,v2_old,x+3,v2_now);        //连线参数 2 的最近两个结果
x=x+3;                             //下一个坐标位置
if(x>787)                          //判断 x 坐标是否越界 787=790-3
{x=10;}
v1_old=v1_now;
v2_old=v2_now;
delay(10);}          //A/D 延时 10mS
```

5.2 进度条的实现

进度条（如图 5-2-1），顾名思义，就是反应一个事件进程的图标，通过该图形能让人们简单明了的了解事物的进程情况。当然，工业自动化现场显示的棒状图也可以看作是“竖立”起来的进度条。

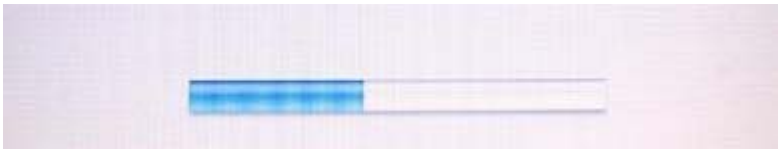


图 5-2-1 进度条

简单的进度条，可以通过组合使用迪文智能显示终端的 0x59、0x5A 和 0x5B 指令来实现。基本思路是先用 0x59 指令勾画出进度条的外廓线，然后依据进度状态，使用 0x5B 指令填充进度部分，使用 0x5A 指令清除剩余的进度部分；当进度变化时，重复 0x5B 和 0x5A 指令即可。如图 5-2-2 所示。

指令	数　据	说　明
0x59	$(X_S+Y_S+X_E+Y_E)_K$	显示矩形框。(X_S, Y_S) 为矩形框左上角，(X_E+Y_E) 为右下角坐标。
0x5A	$(X_S+Y_S+X_E+Y_E)_K$	清除（背景色填充）矩形区域。(X_S, Y_S) 为矩形域左上角，(X_E+Y_E) 为右下角坐标。
0x5B	$(X_S+Y_S+X_E+Y_E)_K$	填充（前景色填充）矩形区域。(X_S, Y_S) 为矩形域左上角，(X_E+Y_E) 为右下角坐标。

表 5-2-1 进度条相关指令说明

图 5-2-2A 使用 0x59 勾画轮廓

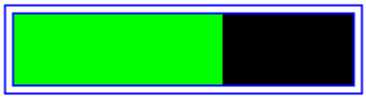


图 5-2-2B 使用 0x5B 填充进度区域

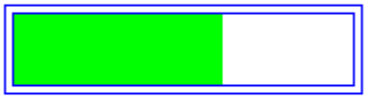


图 5-2-2C 使用 0x5A 清除非进度区域

```
//下面的代码实现了简单进度条的显示，进度条左上角坐标是（x,y），宽度为 100*20
void status（int x,int y,unsigned char step）//（x,y）为进度条内框左上角，step 为进度
{setcolor(0x001f）；                    //设置进度条外框颜色为蓝色
rectan(x-2,y-2,x+104,y+22)；            //用 0x59 指令画矩形框，作为外框,外框的厚度为 4 个像素
rectan(x,y,x+102,y+20)；
setcolor2(0x07e0,0xffff)；              //设置前景色为绿色，背景色为白色
fillw(x+1,y+1,x+1+step,y+19)；          //使用 0x5B 指令，前景色填充进度区域
clrw(x+2+step,y+1,x+101,y+19)；}        //使用 0x5A 指令，背景色清除非进度区域
```

5.3 模拟仪表板的实现

工业现场，为了显示的直观生动，常常模拟“仪表板”的显示效果，由于涉及到不规则形状区域的处理，不能直接使用进度条填充的办法实现。我们可以把仪表板所有的可能显示状态做成小图片，把这些小图片拼接成一幅或多幅图片，保存到智能显示终端中，然后使用 0x71 图片剪切指令来根据显示状态要求从这些图片上“剪切”指定的仪表状态到指定的仪表板显示位置显示来实现。

指令	数　据	说　明
0x71	PICNUM+ X_S+Y_S+X_E+Y_E+X+Y	将存储在显示终端 FLASH 中，索引号为 PICNUM 的图片中，（X_S，Y_S）为左上角，（X_E，Y_E）为右下角坐标的显示区域剪切到当前屏幕的（X,Y）位置显示出来。

表 5-3-1 0x71 图片剪切指令说明

下面，通过一个 C 程序具体说明一个模拟油压表的实现过程。示例的模拟油压表共有 10 个不同的指针状态，拼接在一幅 640×480 的图片上，如图 5-3-1 所示。

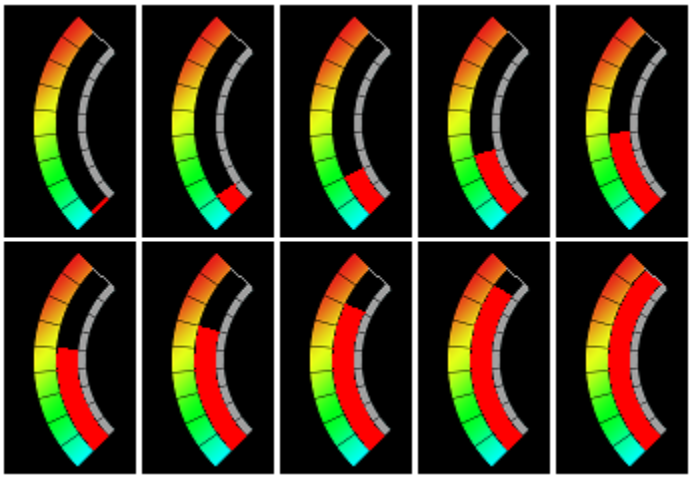


图 5-3-1 拼接了 10 个不同指针状态仪表板的图片

```
//在（x,y)位置显示模拟的油压表
oilmeter(int x,int y,unsigned char p)
{int x0,y0;
 Txword(0xAA71);          //图片剪切指令 0x71
 Txbyte(0x0A);            //拼接的油压表图片保存在 Flash 第 10 幅图片位置
 x0=(p%5)*108;            //发送被剪切小图标的左上角 x 坐标
 Txword(x0);
 y0=(p/5)*240;            //发送被剪切小图标的左上角 y 坐标
 Txword(y0);
 Txword(x0+107);          //发送被剪切小图标的右上角 x 坐标
 Txword(y0+239);          //发送被剪切小图标的右上角 y 坐标
 Txword(x);               //发送油压表显示位置的 x 坐标
 Txword(y);               //发送油压表显示位置的 y 坐标
 TxEOF();}                //发送 0xCC 33 C3 3C 帧结束符
```

使用 0x71 指令，配合界面分解和图片设计，可以实现非常复杂的图形界面。
更方便的调用，可以使用配置文件来实现，请参考本文档 7.2 节。

5.4 使用暂存缓冲区方便的实现历史曲线回放（M100 内核终端不支持）

在应用中，经常需要把历史记录的曲线调出来查看，而在查看中，往往需要对历史数据进行快速回放，典型的，比如：

- 把任意一个指定时间段的数据波形快速调出来回放；
- 窗口滑动显示：类似透过开动的火车窗口看窗外景色的效果（屏幕显示区域类似火车窗户）；
- 配合按键或触摸屏的快速的翻页切换。

要完美的实现上面的功能，采用“5.1 动态曲线图实时显示”的办法是行不通的。主要原因是因为历史回访对数据刷新的实时性要求很高，回放中用户可能会频繁的来回切换显示记录页，而迪文智能显示终端采用串口通信，用户系统的 CPU 一般最多就支持 115200bps 的波特率（2000 点/秒的显示速度，正常显示已经非常快了），如果每次记录页改变，都需要通过串口来重新更新数据（置点或连线），串口的数据传送速度低就成了致命的问题。

比如，以 115200bps 波特率在 DMT80480S070_02WT 终端上刷新 1 条满屏（x 方向 800 点）曲线，需要对 400 个点（连线间距至少为 2，否则会连在一起看不清楚）连线或对 800 个点置点，需要的时间大约＝800×4/（11520×0.9）＝310mS 。这是单条曲线的结果，如果是多条曲线，时间会成比例增加。对于历史回放来说，帧切换速度要控制到 100ms 以内，300mS 实在是太慢了（以每次按键，图像窗口右移 50 个数据点为例，上面的通信速度，单条曲线 1 秒最多移动 3 次（150 个数据点），已经明显让人感觉不流畅了）。如果用户系统的串口速率达不到 115200，比如只有 9600，那么就只好使用一个小窗口（放大镜效果，减少了每次刷新数据量）来做历史回放了！

为了解决上面的问题，就需要使用迪文智能显示终端的暂存缓冲区高速置点（连线）功能。暂存缓冲区是在智能显示终端里面开辟的一个 40KW（80KB）的 RAM 区，用户可以事先把需要显示的数据“暂存”到里面，然后等数据准备好后，通过简单的指令调用让终端直接本地高速访问内部 RAM 来实现指令的高速同步执行（比如置点速度可以达到 0.1uS/点，对应 10^7 点/秒）。

暂存缓冲区指令除了执行速度快，显示“同步”外；由于数据保存在存储器里面，用户可以极其方便的通过修改显示指令访问的存储器首地址来方便的实现平滑的平移显示（0xC101、0xC102）或曲线缩放（0xC103 指令）效果。

暂存缓冲区相关的指令说明如表 5-4-1 所叙。

指令	数　据	说　明
0xC0	ADRH+ADRL+Data0+…+Datan	写数据到暂存缓冲区，ADRH:L 为首地址，范围 0x0000-0x9FFF，共 40KWord；每个地址两个字节数据。
0xC1	0x01＋ADRH+ADRL+Pn_H+Pn_L	使用暂存缓冲区的数据置点，ADRH:L 为置点数据的起始存储地址，Pn_H:L 为置点数目，每点 3Word 数据；最多 13653 个点。 缓冲区数据格式：Psx+Psy+Pixel_Color（颜色，MSB）
	0x02+ADRH+ADRL＋Ln_H+Ln_L	使用暂存缓冲区的数据连线，ADRH:L 为置点数据的起始存储地址，Ln_H:L 为连线数目，每条线 5Word 数据；最多 8191 条线。 缓冲区数据格式：Xs+Ys+Xe+Ye+Line_Color（颜色，MSB）
	0x03+Address+X+Y+Line_Number +D_x+Dis_x+Color	Address：暂存缓冲区的首地址； 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：读缓冲区的点间隔，0x01-0xFF，即每连 1 条线后，缓冲区地址自动增加 D_X，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； Color：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。

表 5-4-1 暂存缓冲区指令说明

下面以实现图 5-4-1 的 3 条曲线回放为例来说明暂存缓冲区指令的使用，相关的 C 参考代码如下。

```
//使用迪文智能显示终端的暂存缓冲区功能来实现历史记录的快速、方便回放
int pa[1000],pb[1000],pc[1000]; //历史记录数据
int i,j,x;
long adr;
x=40;
for(i=0;i<25;i++)        //发送 pa 数据到暂存数据区
{Txword(0xaac0);         //写暂存缓冲区指令
Txword(i*120);           //缓冲区首地址，pa 数据区 0x0000-0x1FFF，每个点 6 字节
for(j=0;j<40;j++)
{Txword(x);              //x 坐标
Txword(pa[i*40+j]);      //点数据
Txword(0xf800);          //颜色
x++;
if(x>200)                //历史记录窗口位置 x＝40-200
{(x=40;}}
TxEOF();}                //发送帧结束符
x=40;
for(i=0;i<25;i++)        //发送 pb 数据到暂存数据区
{Txword(0xaac0);
Txword(i*120+0x2000);
for(j=0;j<40;j++)
{Txword(x);
Txword(pb[i*40+j]);
Txword(0x07e0);
x++;
if(x>200)
{(x=40;}}
TxEOF();}
//发送 pc 数据到暂存数据区
x=40;
for(i=0;i<25;i++)
{Txword(0xaac0);
Txword(i*120+0x4000);
for(j=0;j<40;j++)
{Txword(x);
Txword(pc[i*40+j]);
Txword(0x001f);
x++;
if(x>200)
{(x=40;}}
TxEOF();}
//下面开始通过 0xC1 指令来快速查看历史记录
adr=0;              //缓冲区首地址
while(KEYOK)        //如果检测到按键，就判断是否要翻页
{if(Kcode==1)       //下翻键
{adr=adr+480;       //每次翻一个窗口，480＝160×3
if(adr>3000)
{adr=0;}}
if(Kcode==0)        //上翻键
{if(adr<480)
{adr=480;}
```

图 5-4-1 使用暂存缓冲区来方便的实现历史记录曲线回放

```
adr=adr-480;}
piccut(10,40,300,200,400,40,300,200,400); //把历史记录曲线窗口用 0x71 指令剪切、粘贴，同时清除了原来的曲线
for(i=0;i<3;i++)
{Txword(0xaac1);          //显示历史记录
Txbyte(0x01);
Txword(adr+0x2000*i);     //暂存缓冲区首地址 pa(i=0) pb(i=1) pc(i=2)
Txword(160);              //窗口宽度是 160 个点
TXEOF();}
KEYOK=FALSE;}
```

5.5 如何设计类似 Windows 风格的图形界面

由于迪文智能显示终端的以下 3 大特点，我们强烈建议客户设计“时尚、直观”的“图形”界面来实现人机交互，如图 5-5-1 所示。

- **拥有标准 128MB，并可以轻松扩充到 1GB 的不压缩图片存储器；**
- **和显示屏等分辨率，精度极高，稳定可靠的模拟触摸屏处理技术；**
- **灵活、简洁的串口操作指令集实现 65K 色的完美演绎。**

基于以上特点，采用迪文智能显示终端来做人机交互界面（HMI），界面的设计过程变成了一个纯粹“美工”或者“艺术”的设计和体验过程。只要你可以想得出来，基本上迪文终端都可以帮你演绎出来。由于迪文的触摸屏没有传统 HMI 的数字触摸屏“放置格子”限制，触控区域可随意放置；配合超大、不压缩的图片存储空间可以让您把界面真正设计成一组交互页，并且可以用很简单的代码为这些显示页之间建立“链接”，简化程序的设计，大大降低开发的难度，如下所示。

```
//链接表的定义  {当前图片,x0,y0,x1,y1,切换的图片}
int xdata mlink[][6]={{0,0,0,639,479,2},{0xffff,0,0,0,0,0}};
int i,x,y,n,;    //x,y 为触摸位置
i=0;
while(TCHOK)
{TCHOK=FALSE;    //清除触摸标记
n=mlink[i][0];
if(n==0xffff)    //0xffff 为链接表的结束符
{break;}
if(n==Pic_n)&&(x>mlink[i][1])&&(y>mlink[i][2])&&(x<mlink[i][3])&&(x<mlink[i][4]))
{Pic_n=mlink[i][5];    //取出下一个显示图片界面位置
Picdisp(Pic_n);        //显示图片 AA 70 Pic_n CC 33 C3 3C
break;}
i++;}
```

上面的 C 程序例子，完成了触摸“链接”的处理，mlink[][]数组的举例是对一个 640×480 的终端，在显示开机界面（第 0 页）时，点击屏幕的任何一个位置（x0=0,y0=0,x1=639,y1=479）时，将切换到第 2 幅图片显示。用户可以方便的在数组中添加其它的“链接”并把相应的界面保存到智能显示终端中，即可完成复杂的触控界面设计。可以使用配置文件来让 HMI 实现自动触控切换，请参考本文档 7.1 节。

对于一般的触控产品设计，上面这几行简单的代码基本上已经把你的触控软件设计任务完成了一多半，并且将来的产品升级或者界面切换顺序调整，只需要修改 mlink[][]数组的定义即可，一般的美工或者文员即可完成修改而不需要“劳驾”专业的研发工程师。

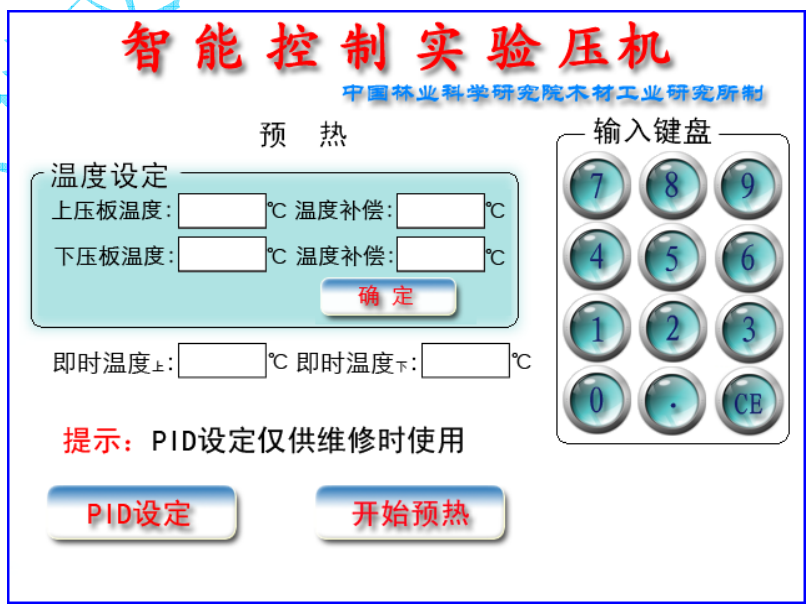


图 5-5-1 典型的触控界面设计

5.6 区域图片（照片）实时刷新

如图 5-6-1 所示，在某些应用场合，我们可能需要在屏幕一些小区域实时（及时）刷新显示内容，比如图示的员工照片区域。如果照片是有限并且固定的（比如一个餐厅的电子菜谱），我们当然可以先把照片做好保存到终端中，使用智能显示终端的 0x71 区域剪切、粘贴指令来实现（详见“5.3 模拟仪表板的实现”）。如果图像需要实时刷新（比如网络传过来的照片）或者图像数目过于庞大（比如有上万名员工，并且每天都有员工入职和离职），使用 0x71 指令就达不到要求了，此时需要使用迪文智能显示终端的 0x72 直接写显存指令来实现区域图片实时刷新。

图 5-6-1 区域图片实时显示示例

指令	数　据	说　明
0x72	ADRH+ADRM+ADRL＋DATA0＋……＋DATAn	将数据（DATA0－DATAn）直接写入显存，（ADRH:ADRM:ADRL）为显存首地址，每个地址两个字节数据。数据串长度不能超过 248（n<＝248）。

表 5-6-1 0x72 指令说明

- **显存地址计算**

假设智能显示终端的横向（x 方向）分辨率为 H，纵向（y 方向）分辨率为 V，则屏幕上（x,y）位置点的显存地址计算方法是：　Adr＝（y×H＋x）　。

比如，DMT64480S057_01WT 屏幕上(150,100)位置，其显存地址＝100×640＋150＝64150（0xFA96）。

- **像素数据**

迪文的 65K 彩色智能显示终端，每个像素点，由两个字节（16bit）的显示数据组成。显示数据按照 5R6G5B 的调色板组织，发送时高字节在前（MSB）发送。比如，要在上面例子的位置显示一个红色的点，使用 0x72 指令发送：　AA 72 00 FA 96 F8 00 CC 33 C3 3C

- **宽行和换行的处理**

0x72 指令下发数据长度最多为 248 字节（124 个连续的点），对于像素点超过 124 点阵的宽行，需要把数据分割成几帧下发；换行时，由于显存地址不连续，也必须重新计算显存地址然后下发。

- **下载速度和时间**

迪文智能显示终端的串口速度可以用户设定，最高为 921600bps，对于一个 H×V 的小区域图片，以波特率 B（bps）下发，需要的时间大约为：T=（H×V×2）/（(B×0.1）×0.9）　秒。

比如，128×160 的照片，115200bps 下发到终端显示，大约需要 4 秒，采用 921600bps 下传，则只需要 0.5 秒。相关的 C 参考代码如下：

```
//把一幅 128×160 的照片，下发到迪文 640×480 终端（x,y）为照片左上角位置显示
unsigned char bmp[160][256];     //位图数据
int i,j,k,n;
long adr;                        //显存地址
for(i=0;i<160;i++)               //每个循环发送一行
{adr=y*480+x;                    //计算显存首地址，640×480 分辨率终端
k=0;                             //位图数据的指针位置
for(;;)
{n=256-k;                        //本帧要发送的数据数目，256 是一行 128 像素的数据字节数
if(n==0)
{break;}                         //本行发送完毕，跳出发送下一行
if(n>248)
{n=248;}                         //每帧最多 248 字节
Txword(0xaa72);                  //0x72，直接显存操作
Txadr(adr+k);                    //发送显存首地址
for(j=0;j<n;j++)
{Txbyte(bmp[i][k]);              //发送位图数据
k++;}}}
```

6 外设和附加功能

为了用户使用方便，降低用户系统的复杂程度，迪文科技生产的智能显示终端集成了很多外设和附加功能。其中最基本的外设是 4×4（某些型号为 8×8）矩阵键盘接口、4 线触摸屏、背光亮度调节、用户数据库以及一些方便单片机使用的辅助算法，下面逐一介绍。

6.1 键盘接口

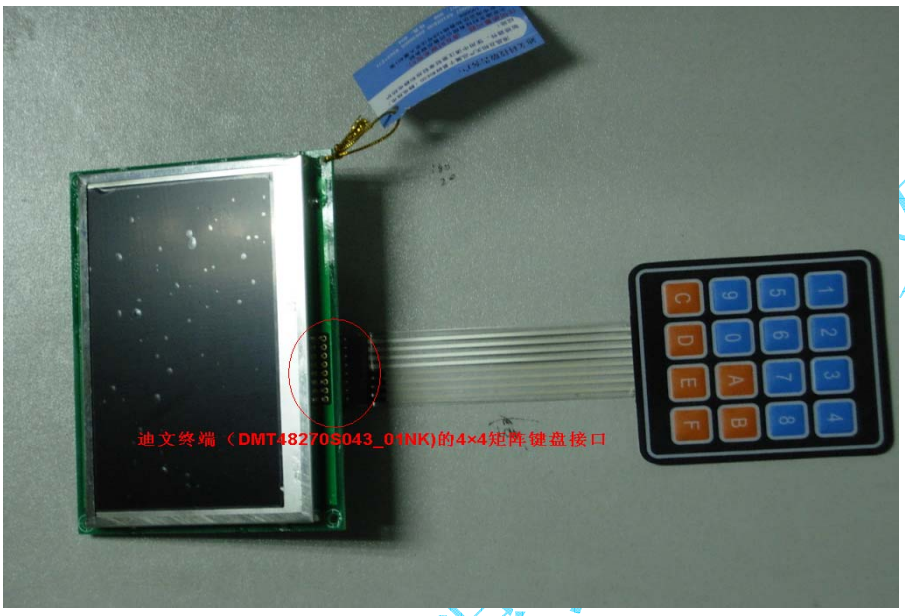


图 6-1-1 迪文终端（DMT48270S043_01NK）的键盘接口示例图

迪文智能显示终端中，和键盘接口相关的指令有两条：0x71（键码上传）和 0xE5（键码配置），相关说明如表 6-1-1 所示。

指令	数　据	说　明
0x71	K_Code	键盘接口有键按下时，键盘接口键码自动上传。
0xE5	0x55+0xAA+0x5A+0xA5+K0+ … +K63	配置键盘接口，K0-K63 对应一个 8×矩阵键盘的 64 个单键值；对于 4×4 键盘，只有部分按键有映射。

表 6-1-1 键盘接口相关指令说明

键盘的扫描由智能显示终端自动完成，当按键按下时，终端会自动从串口上传对应的键码，推荐用户使用串口中断方式来接收键码，相关 C 和 ASM51 参考代码如下：

```
//C 串口中断服务程序，定义了数组 Rx_key[3]，正确的键码放在 K_code 并置位标记 Key_ok
    for(i=0;i<2;i++)
    {Rx_key[i]=Rx_key[i+1];} //移动接收窗口，以方便判断
    Rx_key[2]=SBUF;          //把接收到的串口数据放在最后
    if((Rx_key[0]==0xAA)&&(Rx_key[1]==0x71)&&(Key_ok==FALSE))
    //接收完毕并且没有键码未处理
    {Key_ok=TRUE;            //置位接收成功标记
    K_code=Rx_key[2];}       //保存键码
```

```
;ASM51 串口中断服务程序
        MOV     RXKEY0,RXKEY1    //移动接收窗口
        MOV     RXKEY1,RXKRY2
        MOV     RXKEY2,SBUF
        CLR     RI
        JB      KEYOK,RXKEYE     //如果有键码没有处理就不接收新键码
        MOV     A,RXKEY0
        CJNE    A,#0AAH,RXKEYE   //判断是否接收完毕
        MOV     A,RXKEY1
        CJNE    A,#71H,RXKEYE
        MOV     KEYCODE,RXKEY2   //置位键码接收成功标记
        SETB    KEYOK
RXKEYE: NOP
```

直接使用 16 进制的键码对用户程序的处理是非常不方便的，所以终端设置了 0xE5 指令，通过 0xE5 指令，用户可以方便的把键盘上的按键和上传的键码对应起来，提高程序的可读性。使用 0xE5 指令来配置键码，需要 3 个步骤，下面以把图 6-1-1 键盘的键码对应成 ASCII 字符（即按 1 键上传 0x31 键码）为例来说明。

- **键码配置第 1 步：先把整个键码顺序排列。**

串口发送下面的指令：

AA E5 55 AA 5A A5 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F CC 33 C3 3C

- **键码配置第 2 步：测试键码。**

按压键盘所有的按键，把串口上传的键码填到对应按键位置，获得一个键码表，见表 6-1-2 的“顺序键码”列。

按　键	0	1	2	3	4	5	6	7
顺序键码	0x00	0x01	0x02	0x03	0x08	0x09	0x0A	0x0B
需要的键码	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37
按　键	8	9	A	B	C	D	E	F
顺序键码	0x10	0x11	0x12	0x13	0x18	0x19	0x1A	0x1B
需要的键码	0x38	0x39	0x41	0x42	0x43	0x44	0x45	0x46

表 6-1-2 键码测试表

- **键码配置第 3 步：获得新的键码配置表并下发到智能显示终端。**

根据第 2 步测试的键码，我们用“需要的键码”去取代原来“顺序键码”位置，并把不需要的键码修改为 0xFF(或其它不使用的键码，比如 0x00 以提高抗干扰能力)，获得了我们需要的键码配置表，用 0xE5 指令下发到终端即可。之后，再按压“1”键，终端上传的键码将是 0x31。

AA E5 55 AA 5A A5 30 31 32 33 FF FF FF FF 34 35 36 37 FF FF FF FF 38 39 41 42 FF FF FF FF 43 44 45 46 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF CC 33 C3 3C

6.2 触摸屏

6.2.1 触摸屏原理

触摸屏作为液晶显示屏的附件，附着在显示屏的表面，与显示屏相配合使用。迪文智能显示终端采用的是四线电阻式模拟触摸屏。触摸屏的屏体部分是一块与显示器表面非常配合的多层复合薄膜，由一层玻璃或有机玻璃作为基层，表面涂有一层透明的导电层（ITO 膜），上面再盖有一层外表面硬化处理、光滑防刮的塑料层，它的内表面也涂有一层透明导电层，在两层导电层之间有许多细小(小于千分之一英寸)的透明隔离点把它们隔开绝缘，如图 6-2-1 所示。

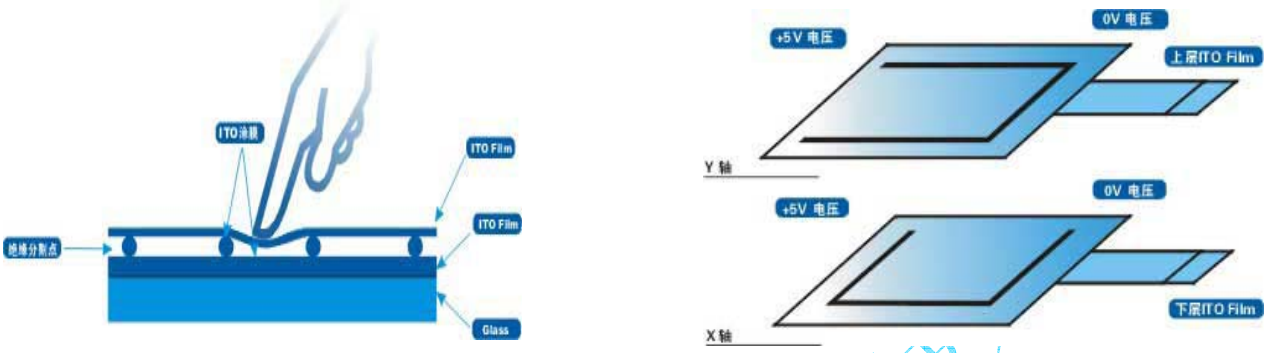


图 6-2-1 触摸屏工作原理

四线式触摸屏的 X 工作面和 Y 工作面分别加在两个导电层上，共有四根引出线，分别连到触摸屏的 X 电极对和 Y 电极对上。触摸屏的两个金属导电层是触摸屏的两个工作面，在每个工作面的两端各涂有一条银胶，称为该工作面的一对电极，若在一个工作面的电极对上施加电压，则在该工作面上就会形成均匀连续的平行电压分布。当在 X 方向的电极对上施加一确定的电压，而 Y 方向电极对上不加电压时，在 X 平行电压场中，触点处的电压值可以在 Y+(或 Y-)电极上反映出来，通过测量 Y+电极对地的电压大小，便可得知触点的 X 坐标值。同理，当在 Y 电极对上加电压，而 X 电极对上不加电压时，通过测量 X+电极的电压，便可得知触点的 Y 坐标。

典型的触摸屏处理 IC 是 BB 公司（TI）的 ADS7843 和 ADS7846，但由于其不带漂移补偿、ESD 能力弱，误差较大，在实际应用中的效果，尤其是处理 8 寸以上触摸屏的效果不是很好，容易出现以下问题：

- 触摸屏四个角的误差比较大，并且误差方向不确定；
- 触摸效果受温度、尤其是湿度影响大；
- 工作一段时间后，需要用户软件重新校准并做漂移补偿；
- 对静电敏感，现场操作员身上静电比较多（比如冬天穿毛衣）时，点击的第一下会漂移很大，容易造成误动作。
- 不带漂移补偿，对大尺寸的廉价触摸屏（ITO 导电膜不均匀）基本上不可用。

这些问题，对于工业产品是无法容忍的，自 2008 年 06 月开始，在迪文的智能显示终端中不再使用 ADS7843 作为触摸屏处理 IC，而使用迪文自行研制的 TPS01 触摸屏控制器。使用 TPS01 控制器，不仅完全杜绝了 ADS7843 的上述问题，使迪文的模拟触摸屏可以达到标准 HMI 数字触摸屏的稳定可靠，最重要的，由于稳定性的提高，使用户触控界面的设计更加灵活、方便，不再拘束于数字触摸屏的“方格”位置限制。

6.2.2 触摸屏上传数据格式

迪文智能显示终端中，和触摸屏相关的指令如表 6-2-1 所示。

指令	数　据	说　明
0x72	X+Y	触摸屏停止点击（松开）时的触摸位置数据上传。
0x73	X+Y	点击中，触摸数据上传，连续按下时，每秒上传 10 次左右。
0xE4	0x55 0xAA 0x5A 0xA5	触摸屏校准
0xE0	55 AA 5A A5＋TFT_ID＋Bode_Set＋Para1	设置终端的 TFT 面板显示驱动模式（TFT_ID）、接口波特率（Bode_set）、系统配置参数 Para1；设置的参数掉电后不会丢失。 Para1 定义如下： Para1.7　点击触摸屏后 0x72 指令是否上传 0=上传 1=不上传 Para1.6　点击触摸屏后 0x73 指令传送方式 0=连续传送 1=只在按下时传送一次。 Para1.5　是否由 HMI 处理触控界面切换，　1＝处理　0=不处理 Para1.4-Para1.0　未定义，写 0

表 6-2-1 和触摸屏相关的指令

6.2.3 触摸按键的识别和处理

图 6-2-2 触摸按键的识别

触摸按键就是在屏幕上显示一个按钮的图标，当用户点击该图标时，用户软件可以正确识别出是点击该图标并执行正确的操作。当用户点击触摸屏时，智能显示终端会自动上传触摸位置的 x、y 坐标，用户软件通过判断坐标位置是否在预先设定的按钮区域内来识别，识别算法是：

如果 x0<x<x1，并且 y0<y<y1，则表明点击区域有效，其中(x0,y0)是触摸区域的左上角坐标，(x1,y1)是触摸区域的右下角坐标。

如图 6-2-2 所示，有效按钮的坐标位置须满足条件：31<x<231　191<y<388。

在用户软件设计中，通常是通过串口中断捕获触摸位置，然后查表来确定触摸位置。

更方便的应用，可以使用配置文件，让迪文 HMI 自动完成，请参考本文档 7.1 节。

以识别图 6-2-2 的触摸按键为例，相关 C 和 ASM51 参考代码如下：

```
//串口中断服务程序，正确的触摸键码放在 K_code 并置位标记 Key_ok
int xdata kpos[1][5]={{31,191,231,388,1}}; //按键区域，对应的键码是 1；更多区域增加行
int x,y,i,K_code;
unsigned char Rx_tch[5];
for(i=0;i<5;i++)
{Rx_tch[i]=Rx_tch[i+1];} //移动接收窗口，以方便判断
Rx_tch[5]=SBUF;          //把接收到的串口数据放在最后
if((Rx_tch[0]==0xAA)&&(Rx_tch[1]==0x72)&&(Key_ok==FALSE)) //接收完毕且无键码未处理
{x=Rx_tch[2]*256+Rx_tch[3];//把接收的坐标数据换算成整数
y=Rx_tch[4]*256+Rx_tch[5];
for(i=0;i<1;i++)
{if((x>kpos[i][0])&&(y>kpos[i][1])&&(x<kpos[i][2])&&(y>kpos[i][3]))//判断点击是否有效
{K_code=kpos[4];          //取出对应的键码
Key_ok=TRUE;              //置位接收成功标记
break;}}}

;点击迪文智能显示终端触摸屏，串口中断服务程序，读取触摸位置并完成查表键码识别
    MOV     RXAA,RX72    ;移动接收窗口
    MOV     RX72,RXXH
    MOV     RXXH,RXXL
    MOV     RXXL,RXYH
    MOV     RXYH,RXYL
    MOV     RXYL,SBUF
    CLR     RI
    JB      KEYOK,RXTCHE     ;如果有键码没有处理就不接收新键码
    MOV     A,RXAA
    CJNE    A,#0AAH,RXTCHE   ;判断是否接收完毕
    MOV     A,RX72
    CJNE    A,#72H,RXTCHE
    MOV     DPTR,#KEYPOS     ;接收完毕，就取出坐标并查表识别
```

```
RXTCH1: MOV     A,#00H
        MOVC    A,@A+DPTR
        CJNE    A,#0FFH,RXTCH2
        LJMP    RXTCHE          ;遇到 0xFF 表示没有搜索到有效按键
RXTCH2: MOV     B,A
        MOV     A,#01H
        MOVC    A,@A+DPTR
        CLR     C
        SUBB    A,RXXL
        MOV     A,B
        SUBB    A,RXXH
        JNC     RXTCH3          ;x>x0
        MOV     A,#02H
        MOVC    A,@A+DPTR
        MOV     B,A
        MOV     A,#03H
        MOVC    A,@A+DPTR
        CLR     C
        SUBB    A,RXYL
        MOV     A,B
        SUBB    A,RXYH
        JNC     RXTCH3          ;y>y0
        MOV     A,#04H
        MOVC    A,@A+DPTR
        MOV     B,A
        MOV     A,#05H
        MOVC    A,@A+DPTR
        CLR     C
        SUBB    A,RXXL
        MOV     A,B
        SUBB    A,RXXH
        JC      RXTCH3          ;x<x1
        MOV     A,#06H
        MOVC    A,@A+DPTR
        MOV     B,A
        MOV     A,#07H
        MOVC    A,@A+DPTR
        CLR     C
        SUBB    A,RXYL
        MOV     A,B
        SUBB    A,RXYH
        JC      RXTCH3          ;x<y1
        MOV     A,#09H
        MOVC    A,@A+DPTR
        MOV     KCODE,A         ;取出键码，只要低位
        SETB    KEYOK
        LJMP    RXTCHE
RXTCH3: MOV     A,DPL           ;指向下一个识别位置
        ADD     A,#10
        MOV     DPL,A
        CLR     A
        ADDC    A,DPH
        MOV     DPH,A
        LJMP    RXTCH1
RXTCHE: NOP

KEYPOS: DW 31,191,231,388,1
        DB 0FFH                 ;触摸屏区域定义结束
```

如果用户希望触摸按键被识别后，能够“动作”（比如改变按钮颜色或把按钮突出）提示用户，可以把 0x73、0x72 指令和图片剪切的 0x71 指令组合起来使用，即：

- 当用户按下触摸屏时（收到 0x73 指令），用 0x71 指令切换一个按下的按钮到当前按钮位置提示；
- 当用户松开触摸屏时（收到 0x72 指令），用 0x71 指令切换一个未按下的按钮到当前按钮位置提示。

如图 6-2-3 所示。

触摸屏未按下时　　触摸屏按下时(收到 0x73)　　触摸屏松开后回复正常(收到 0x72)

图 6-2-3 配合 0x72、0x73 指令，借助 0x71 指令实现触摸按键的“动作”过程

6.2.4 触摸屏校准

当用户发现触摸屏出现精确度下降现象时或在迪文智能显示终端第一次安装好触摸屏后，需要对触摸屏进行校准操作。

触摸屏校准由串口向终端发送触摸屏校准指令“AA E4 55 AA 5A A5 CC 33 C3 3C”来实现，发送后，用户按照屏幕提示依次点击触摸屏“左上角”，“左上角”，“右下角”白色校准点位置即可。

校准“左上角”白点的智能显示终端提示界面如图 6-2-4 所示，红色圆圈内的白点就是提示用户点击的校准点。

图 6-2-4 校准触摸屏“左上角”的智能显示终端提示界面

6.3 访问 32MB 用户存储器

在单片机系统设计中，经常会涉及到一些变量、参数或者历史记录的保存，但是使用非易失存储器，尤其是大容量的非易失存储器是一件比较麻烦的事情，因为：

a. 大多数 MB 以上容量的存储器都是并口的，需要用户单片机要有并行总线接口或者比较多的 I/O 口来连接，而一般单片机的总线就 64KB 的寻址空间，扩展总线 PCB 设计很麻烦，生产工艺控制不好故障率就比较高；

b. 小容量的 I^2C 或 SPI 串行接口存储器，其读写时序有严格的要求，用户需要写很多代码来作接口，占用 CPU 资源，这样往往会带来和 CPU 其它任务的冲突；

c. 除非是以备份电池来维持 RAM 数据掉电不丢失的 NvRAM 存储器，一般的 Flash 非易失性存储器的擦除都是按照扇区进行的；除非是整个扇区的写，否则在写之前要使有一个比较大的 RAM 来做备份；不仅增加了系统的复杂度（外扩 RAM，并且 RAM 基本都是并口的），软件处理起来也比较麻烦；

d. NAND Flash“天生”就有坏扇区，使用之前必须进行格式化操作以“跳过”损坏的扇区；由于使用电荷存储来记录信息，几乎所有的 Flash 存储器在使用过程中都会受到 ESD 放电、宇宙射线等干扰而引起“位反转”（0 变 1 或者 1 变 0），造成记录信息的改变，必须使用纠错技术（一般多使用 BCH 编码等 FEC 技术）来恢复信息。

对于 PLC 系统，要增加“黑匣子”功能（大容量存储器），更不是一件容易的事情。

为了简化、方便用户系统的设计，让用户真正专注于其“专业”的“核心竞争力”（**毕竟绝大多数的产品不是纯粹的电子产品，客户的核心竞争力也不是电路或者软件研发水平高低，电路或者软件更多是起一个添彩或者加花的作用**），降低研发难度，在迪文智能显示终端中，在图片存储器中单独开辟了一个最大 32MB 的用户存储器区，可以很方便的通过串口进行存储器的读写操作，而具体的操作过程（比如格式化和纠错）用户则不必关心。这个存储器采用 NAND 型 Flash，擦写次数 10 万次，寿命 10 年，配合迪文的差错控制技术，完全可以满足一般测控系统对非易失数据存储的要求。

智能显示终端种和用户存储器相关的指令如表 6-3-1 所示。

指令	数　据	说　明
0x90	下发： 0x55+0xAA+0x5A+0xA5+ADRH:MH:ML:L+Data 应答：‘OK’	写数据到用户数据存储器，ADRH:MH:ML:L 是首地址，Datas 是要存储的数据，数据库空间最大约为 30MB（29.9375MB,00000-01:DE:FF:FF），和图片存储器的后 32MB 空间（另外 2MB 被系统保留）重叠； 内部存储器分成两个空间： a. 地址范围 0x01:DE:00:00－0x01:DE:FF:FF 的 64KB 随机存储空间，每次写操作，总是执行“回读－修改－回写”，不修改的数据会被保护。 仅 M600 模组支持 64KB 随机数据存储空间，M100 模组不支持； 由于使用暂存缓冲区备份，写随机数据存储器会修改暂存缓冲区后 30KW 的内容。 b. 地址范围 0x00:00:00:00－0x01:DD:FF:FF 的 29.875MB 顺序数据存储数据库，分成 239 个 128KB 数据页，每遇到页首（地址＝********1 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。 数据库的物理介质是 NAND Flash，可擦写次数是 100000 次，寿命为 10 年。
0x91	下发：ADRH:MH:ML:L+LENH:L 应答：ADRH:MH:ML:L＋LENH:L＋Data	从指定地址读数据存储器数据，Len_H:L 是读数据长度（0x0000 表示 65536），Data 是读回的数据，每次最多读取 64KB。 执行 0x90 和 0x91 指令期间，终端不响应用户指令。

表 6-3-1 和用户存储器相关的指令说明

➢ 迪文智能显示终端 128MB 数据空间的结构划分

物理地址范围	0x00000000-0x01FFFFFF	0x02000000-0x07FFFFFF
空间大小	32MB	96MB
说　　明	字库存储器 （32 个 128KB 的小字库， 28 个 1MB 的大字库）	图片存储器（为了纠错需要，每幅图片存储空间会比计算值略大），可以存储 153 幅 640×480 的全屏图片。

表 6-3-2 未使用用户存储器的 128MB 数据空间划分

物理地址 范　　围	0x00000000 -0x01FFFFFF	0x02000000 -0x05FFFFFF	0x06000000 -0x07FDFFFF	0x07FE0000 -0x07FFFFFF
空间大小	32MB	64MB	31.875MB	128KB
说　　明	字库存储器	图片存储器，可以存储 102 幅 640×480 的全屏图片。	0x00000000-0x01DDFFFF 29.875MB 顺序数据存储器	0x01DE0000-0x01DEFFFF 64KB 随机存储器
			用户存储器（为了纠错等需要，用户存储器的实际物理存储空间比用户可用空间要大）	

表 6-3-3 使用用户存储器的 128MB 数据空间划分

➢ 迪文智能显示终端读用户存储器的过程

当从串口收到读用户存储器的指令时，终端首先会根据地址计算出实际的物理存储器地址，然后从存储器中读出数据，进行差错处理，然后发送到串口。注意，用户存储器数据读取时，一次最多读取 64KB（对应读取长度=0x0000）的数据。

➢ 迪文智能显示终端写用户存储器的过程

当从串口收到写用户存储器的指令时，终端首先会根据地址区分是写随机存储器还是顺序存储器，

对于随机存储器，终端按照以下步骤进行写操作：

a. 先把 64KB 的随机存储器数据全部读回到暂存缓冲区（RAM 回读备份）；

b. 擦除随机数据存储器；

c. 把要写的数据写入暂存缓冲区的对应位置（修改）；

d. 把暂存缓冲区的数据进行纠错编码后写入随机数据存储器（回写）；

e. 串口应答‘OK’，写操作完成。

以上步骤，由迪文智能显示终端自动完成，用户无须干预。

注意，M100 内核的低分辨率版本终端，由于没有暂存缓冲区，也就不支持 64KB 的随机数据库；对 M100 内核的智能显示终端写随机数据库将引起错误。

写随机数据存储器，终端每次都对原来的数据进行了备份处理，用户可以方便、“随机”的修改任何位置的存储内容。

对于顺序存储器，终端按照以下步骤进行写操作：

a. 判断当前要写的地址是不是处于 1 个 128KB 扇区的首地址（地址＝** *0 00 00），如果是，执行步骤 b，不是则跳到步骤 c；

b. 擦除一个 128KB 扇区；

c. 写入一个字节数据到指定地址；

d. 地址指针加 1，判断写入数据是否写完，写完跳到步骤 e，反之跳到步骤 a；

e. 串口应答‘OK’，写操作完成。

以上步骤，由迪文智能显示终端自动完成，用户无须干预。

写顺序数据存储器，终端每次不会对原来的数据进行备份处理，后面的数据不会覆盖前面的数据，但前面已经写过的数据，除非擦除，否则就不能改写了；这种方式只适合保存有时间先后“顺序”的历史记录数据。

6.4 使用终端的“拼音输入法”实现中文输入

中文的输入对单片机软件设计来说，是一件很麻烦的事情。但是迪文智能显示终端通过内嵌拼音输入法，使得用户软件的处理变得异常简单，相关的指令如表 6-4-1 所示。

指令	数　据	说　明
0xB0	下发：0x01+PY_Code	基于 1 级汉字库的拼音输入法，PY_Code 是用户下发的拼音，大写表示，最多 6 字节；终端应答，HZ_Num 是该拼音下的汉字数目，0x00 表示拼音错误；Strings 是该拼音下的所有汉字，内码编码。
	应答：0x01+HZ_Num+Strings	

表 6-4-1 拼音输入法指令说明

拼音输入法实质上是一个触摸键码识别和数据库检索的过程，借助迪文终端的拼音输入法检索指令，可以很轻松的实现常用汉字录入。

图 6-4-1 基于迪文终端拼音输入法的触摸键盘中文输入

以图 6-4-1 所示的触摸键盘中文输入为例，通过 C 参考程序来说明基于迪文智能显示终端实现拼音输入法的思路：

```
//用户通过字母键盘输入拼音
//拼音实时在图 6-4-1 的小红框区域显示
//拼音对应的汉字实时在 6-4-1 的大红框区域显示
//用户通过直接点击 6-4-1 大红框区域显示的汉字而选择所要输入的汉字
//在迪文科技 DMT64480S057_01WT 终端上演示

unsigned char py[6]={0x20,0x20,0x20,0x20,0x20,0x20}; //触摸键盘输入的拼音，初始化成空格
unsigned char *str;      //保存终端传来的对应拼音内码
unsigned char i,hz_h,hz_l;       //hz_h:1 是返回的输入汉字内码，hz_h=0x00 表示无效

while(TCHOK)     //有触摸按键按下，位置保存在（x,y）
{Keytch();       //把触摸位置转换成按键
```

```
if(KEYOK)
{for(i=0;i<5;i++)          //是字母键被按下，就把字母赋给拼音数组
{py[i]=py[i+1];}           //窗口左移，比如原来是“ABCDEF”,输入是 K，变成“BCDEFK”
py[5]=K_code;
Txstr(0xaab001,*py);       //把拼音发送给终端查表 AA B0 01 Py CC 33 C3 3C
Rxstr(*str);               //接收终端返回的拼音内码数组 AA B0 01 n String
setcolor(0x07e0,0x001f); //设置颜色为绿色/蓝色
prints(8,235,*py);         //把拼音显示出来（图上的小红框区域）
setcolor(0x0000,0x001f); //设置颜色为黑色/蓝色
clrw(0,184,639,216);       //清除将要显示汉字的区域（图上的大红框区域）
if(str[3]==0x00)
 {break;}                        //n=0，拼音错误，直接跳出
prints(0,184,*(str+3))} ;          //拼音正确，把该拼音下的汉字显示出来
else
 {if((x>0)&&(x<640)&&(y>184)&&(y<216))  //判断是否点击的汉字显示区（也是输入汉字选择区）
 {i=40*((y-184)/16)+x/16;          //计算点击的是哪个汉字
hz_h=0x00;                 //把 hz_h 设置成 0x00，如果有正确的汉字输入就会为非零
if(i<n)                    //点击位置不能超过该拼音下的汉字数目
{hz_h=*(str+2+i);          //内码高字节
hz_l=*(str+3+i);}}          //内码低字节
TCHOK=FALSE;}
```

6.5 使用终端的“数据排序算法”对测量数据进行处理

在很多应用场合下，需要对数据进行滤波处理以去掉干扰，“平均值滤波”是应用比较广泛，而又简单有效的算法。采用平均值滤波，基本有以下 3 个步骤：

a.采集多组数据；

b.去掉数据里面的最大和最小值；

c.对剩余的数据进行累加平均。

对一些简单的 8 位单片机来说，对字型数据（比如 12bit A/D 结果）进行平均值滤波处理就不是特别方便处理的了，其中最麻烦的问题就是去掉数据中的最大和最小值过程，尤其涉及到排除多个最大、最小值时，由于涉及排序处理，运算量比较大，响应会比较慢。

在迪文智能显示终端中，内嵌了对字型数据的排序处理，用户通过串口把数据发给终端，终端会按照升序排列后回传给用户，起到用户 CPU 的“算术协处理器”作用。相关指令见表 6-5-1。

指令	数　据	说　明
0xB0	下发：0x03+Data_Pack0	排序，Data_Pack0 是要排序的两字节数组，MSB 方式传送，Data_Pack1 是排序后的数组，Data_Pack1 是升序排列。
	应答：0x03+Data_Pack1	

表 6-5-1 迪文终端的数据排序指令说明

举例：

假设原始数据序列：　123,100,127,128,119,122,125,130,147,133

通过迪文终端排序后的序列：100,119,122,123,125,127,128,130,133,147

用户把序列两边的各 3 个最值扔掉：100,119,122,123,125,127,128,130,133,147

用户对剩下的 4 个数据取平均值，获得滤波结果：125

当然，更简单的滤波做法是取序列中间的数据（125 或 127）直接作为滤波结果。

下面给出一个基于迪文终端对实时 A/D 结果进行滤波处理的 C 参考代码，这个例子也可以当成“通过过采样提高 A/D 精度”的解决办法（基于统计平均值的结果，采样速度提高 4 倍，A/D 精度提高 1bit）。

```
//每 32 次 A/D 结果进行排序处理，取中间 16 个数据的累加和作为 A/D 结果
//12bit A/D，输出为 16bit 结果（16 次累加和，实际精度为 14bit）
unsigned int ad_result，ad_out;  //A/D 返回结果
unsigned char i,*adin;

Txword(0xaab0);                  //数据排序 AA B0 03 Data_Pack CC 33 C3 3C
Txbyte(0x03);
for(i=0;i<32;i++)
{Rd_ad();                          //A/D 转换，结果保存在 ad_result
Txword(ad_result);}                //把 A/D 结果发给终端排序
TxEOF();}                    //帧结束符
Rxstr(*adin);                //接收终端上传的排序结果
ad_out=0;
for(i=16;i<32;i++)           //只取序列的中间 16 个结果进行累加和计算
{ad_out=ad_out+(*(adin+2*i+3))*256+*(adin+2*i+4);}
```

7 使用配置文件来简化设计

7.1 让 HMI 自动进行触控界面切换

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_11WT，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

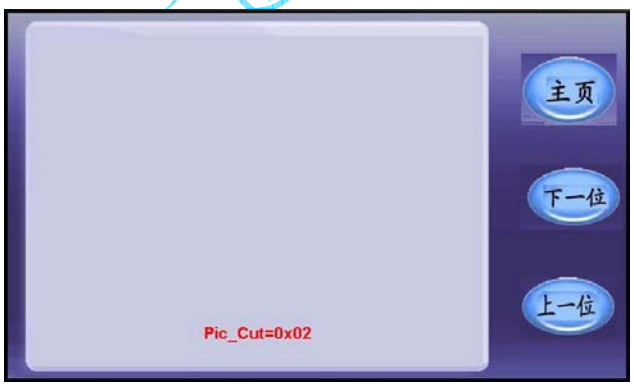
配置文件是由最多 8192 条触控指令组成的二进制文件，每条触控指令 16 个字节，定义如表 7-1：

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。

表 7-1 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 7-1 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。




```
;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 7-1 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。

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。

7.2 方便的调用不同图标显示

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 5.3 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来调用图标显示。

0x99 指令的格式如表 7-2 所示。

指令	数　据	说　明
0x99	$(x,y,Icon\_ID)_0$+……+$(x,y,Icon\_ID)_n$	（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 7-2 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 7-3：

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 7-3 图标的定义

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,(Xs,Ys),(Xe,Ye);
- 与 0x71 指令一样，进行图片剪切：0x71 Pic_ID,（Xs,Ys），（Xe,Ye）(X,Y)
- 进行下一个图标显示的处理。

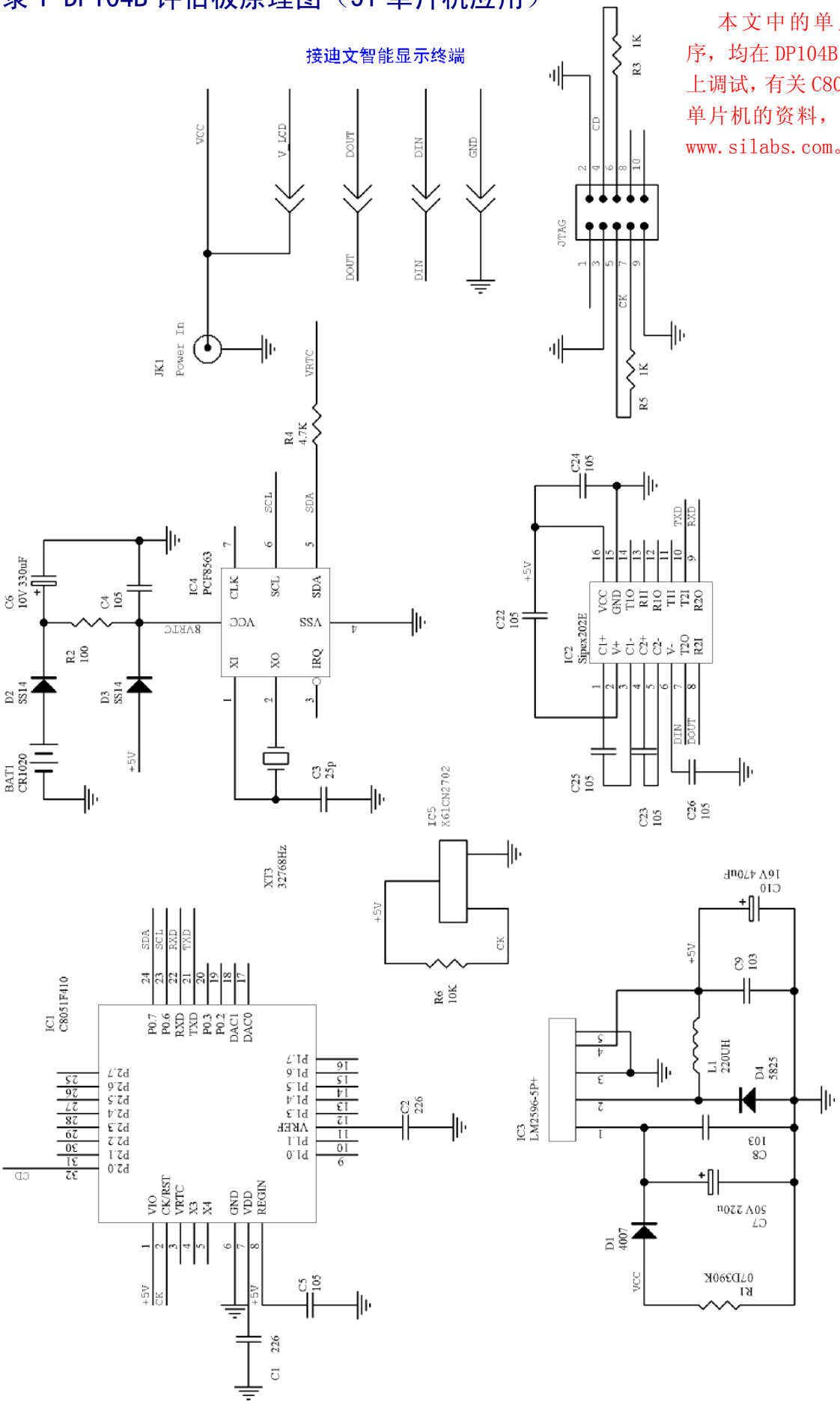
配置文件的生成可以参考本文档 7.1 节的方法。

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。

附录 1 DP104B 评估板原理图（51 单片机应用）

本文中的单片机程序，均在 DP104B 评估板上调试，有关 C8051F410 单片机的资料，请访问 www.silabs.com。



附录 2 51 单片机汇编语言（ASM51）程序设计概要

➢ 基本约定

代码空间：64KB　0000-0FFFFH，复位后从 0000H 位置开始执行程序
RAM 空间：256B　0000-0FFH
　其中：00-1F 为 4 个寄存器区（R0-R7，RS0、RS1 来切换），20-2F 为可以位寻址区，80-FF 只能使用间接寻址。
　堆栈指针 SP，习惯上把 0x80-0xFF 的 128 字节 RAM 作为堆栈，使用 MOV　SP,#80H 来设置。

➢ 常用的伪指令，用来增加汇编的可读性

EQU　替换　编译时，会把 EQU 的项目做替代，一般用来定义变量，注意要先定义后才能使用。
　　SYSFLG　EQU　20H　　定义了一个变量，位于 20H 存储单元
BIT　位定义
　　PWMOUT　BIT　P3.0　　定义 I/O 口 P3.0 为 PWM 输出接口，当然也可以使用　PWMOUT　EQU　P3.0　。
DB 或 DW　在代码空间定义字节或字常量数据
　　ASC_TAB：DB　‘0123456789ABCDEF’
ORG　告诉编译器后面的程序从 ORG 指定的地址开始存放
ORG　　0023H
LJMP　UARTPRO　　串口中断入口地址是 0023H，当串口中断产生时，跳转到 UARTPRO 子程序.
MACRO/ENDM　宏定义一个程序块　类似于 EQU 了一段程序，用来提高程序的兼容性和可读性

```
RSTWDT   MACRO
         MOV      WDTCN,#0A5H          ;如果同时使用了硬件 WDT，则只需在宏中加一行代码即可
         ENDM
```

➢ 数据传送指令 MOV（寄存器之间）　MOVC（片内 ROM）　MOVX（片外存储器）

```
MOV  A,B  ;把 B 的内容放到 A
MOV  A，R7
;利用 MOVC 来查表
MOV DPTR,#TABLE ； 表的首地址
MOV A,#01H　　； 要查表的位置
MOVC　A,@A+DPTR　； 查表的返回值装载在 A 中（‘1’）

TABLE：DB　‘0123456789’

MOV  A,@R0  间指寻址，以 R0（或 R1）作为指针
MOV R0,#30H
MOV  A，@R0
MOV  R1,#31H
MOV  @R1,A
```

上面的 4 行指令等效　MOV　31H,30H

➢ 跳转指令

JB/JNB　如果标志位为 1/0 则跳转　　JB　KEYIN,KEYST
JC/JNC　如果 C 标志为 1/0 则跳转　　加减运算和 CJNE 指令会影响 C 标志
JZ/JNZ　如果 A＝0 或 A 不为零则跳转　JZ　TESTEND
CJNE A/R?，*，K　如果 A（或 R?，R0-R7）不等于*则跳转到 K
DJNZ *,K　先把*减 1，再判断*,如果*不为零则跳转到 K

```
      MOV R7,#10
Delay: NOP
     DJNZ R7,    delay
```

SJMP/LJMP　在 256 字节和 64KB 代码空间内跳转

➢ 位操作指令

ORL　或，用来把特定位置 1
ANL　与，用来把特定位清 0
XRL　异或，用来把特定位取反　　XRL　A,#0FFH　把 A 的内容取反
CPL　取反，用来把指定的位取反　CPL　P3.0
SETB 置位　　CLR　清零位

```
MOV   C,B1
MOV   B2,C        ；把 B1 位的内容传给 B2 位
```

➢ 算术运算

```
ADD   A,B         ；A＝A+B
ADDC  A,B         ；A=A+B+C
SUBB  A,B         ；A=A-B-C
DA    A           ；把 A 做 10 进制调整，注意会影响 C 标志；比如 A=0AH，执行 DA A 后，A 会变成 10H。
MUL   AB          ；B:A=A*B     B 为结果高字节，A 为结果低字节
DIV   AB          ；A＝A/B 的商   B＝A/B 的余数
```

比如：计算 R7*R6＋R5－R4，结果放到 R3:R2

```
MOV A,R7      ;R7*R6
MOV B,R6
MUL AB
ADD A,R5      ;R7*R6+R5,注意进位
MOV R2,A
CLR  A
ADDC A,B
MOV  R3,A
CLR  C        ;计算 R3:R2-R4
MOV  A,R2
SUBB A,R4
MOV  R2,A
MOV  A,R3
SUBB A,#00H
MOV  R3,A
```

➢ LCALL/RET/RETI/PUSH/POP/JMP　程序进程控制

LCALL　程序调用
RET　　子程序返回
RETI　 中断程序返回
PUSH　 把参数压入堆栈
POP　　把参数弹出堆栈
JMP　@A+DPTR　跳转到 A+DPTR 指定的程序位置

```
PUSH    DPH     ;假设 DPTR＝1001
PUSH    DPL
MOV     DPTR,#2000
……
POP     DPL     ;DPTR=2001
POP     DPH     ;DPTR=1001
;注意，堆栈是后进先出模式。
```

➢ 几个常用的 SFR

- 串口

SBUF、TI、RI、SCON、TMOD、TH1、TL1、ES

- 定时器

TMOD、TCON、TF*、TR*、ET*

定时器的 AutoReload 模式（以 T0 为例）：

设置为自动重装载模式后，TH0 为一个 8 位的定时器，装载值就是 TL0 的值，当 TH0 运行到溢出时，会置位 TF0（中断打开则会响应中断），同时会把 TL0 的值自动再装载到 TH0。

自动重装载模式一般用来产生精确的系统定时（定时器中断服务程序不会改变定时时间间隔）

- DPTR

这是 51 单片机唯一的一个 16 位寄存器，用来作为数据指针，可以直接赋 16 位的值。

➢ 程序的典型结构

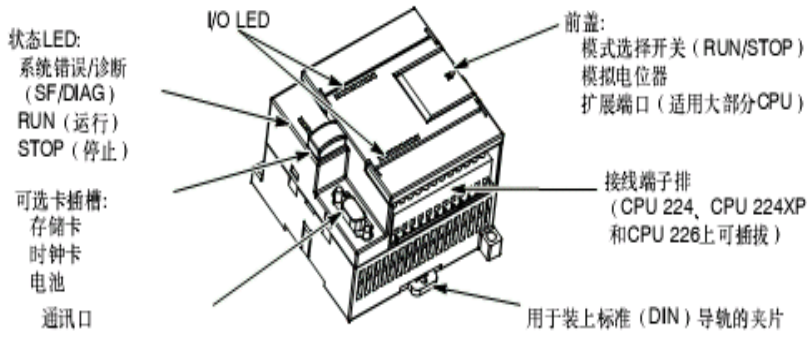
```
;程序的最前面定义变量
SQROUT    BIT     P3.5              ;方波输出的引脚

;指定复位和中断的程序入口地址
ORG     0000H
LJMP    POWERON

;复位后程序入口
ORG     0100H
POWERON: CLR    EA                ;关闭中断
         MOV    SP,#80H           ;给堆栈指针赋初值
         LCALL  INIT              ;初始化子程序
START:   NOP                      ;主程序（循环）
         CPL    SQROUT            ;不停取反 I/O 输出方波
         LJMP   START
```

附录 3 PLC 开发迪文终端指南（S7-200）

➢ S7-200 相关知识

S7-200 CPU 由微处理器、集成电源和数字量 I/O 点组成。

在下载了程序之后，S7--200 将保留所需的逻辑，用于监控应用程序中的输入输出设备。



Siemens 公司提供多种类型的 CPU 以适应各种应用， 表中给出各种 CPU 的特性比较， 详细信息参见 Siemens 公司技术资料

特性	CPU 221	CPU 222	CPU 224	CPU 224XP	CPU 226
外形尺寸（mm）	90 x 80 x 62	90 x 80 x 62	120.5 x 80 x 62	140 x 80 x 62	190 x 80 x 62
程序存储器: 可在运行模式下编辑 不可在运行模式下编辑	4096字节 4096字节	4096字节 4096字节	8192字节 12288字节	12288字节 16384字节	16384字节 24576字节
数据存储区	2048字节	2048字节	8192字节	10240字节	10240字节
掉电保持时间	50小时	50小时	100小时	100小时	100小时
本机I/O 数字量 模拟量	6入/4出 -	8入/6出 -	14入/10出 -	14入/10出 2入/1出	24入/16出 -
扩展模块数量	0个模块	2个模块[1]	7个模块[1]	7个模块[1]	7个模块[1]
高速计数器 单相 双相	4路30KHz 2路20KHz	4路30KHz 2路20KHz	6路30KHz 4路20KHz	4路 30 kHz 2路200 kHz 3路20 kHz 1路100 kHz	6路30KHz 4路20KHz
脉冲输出（DC）	2路20KHz	2路20KHz	2路20KHz	2路100 kHz	2路20KHz
模拟电位器	1	1	2	2	2
实时时钟	配时钟卡	配时钟卡	内置	内置	内置
通讯口	1 RS-485	1 RS-485	1 RS-485	2 RS-485	2 RS-485
浮点数运算	有				
I/O映象区	256（128入/128出）				
布尔指令执行速度	0.22μs /指令				

为了更好地满足应用要求，S7-200 系列提供多种类型的扩展模块。可以利用这些扩展模块完善 CPU 的功能。

扩展模块数量		型号		
数字量模块	输入	8 x DC输入	8 x AC输入	16 x DC输入
	输出	4 x DC输出 8 x DC输出	4 x 继电器输出 8 x AC输出	8 x 继电器输出
	混合	4 x DC输入/4 x DC输出 4 x DC输入/4 x 继电器输出	8 x DC输入/8 x DC输出 8 x DC输入/8 x 继电器输出	16 x DC输入/16 x DC输出 16 x DC输入/16 x继电器输出
模拟量模块	输入	4输入	4热电偶输入	2热电阻输入
	输出	2输出		
	混合	4输入/1输出		
智能模块		定位 以太网	调制解调器 互联网	PROFIBUS-DP
其它模块		ASI		

➢ 迪文智能显示终端和 S7-200 的连接和使用

STEP 7--Micro/WIN 编程软件为用户开发、编辑和监控自己的应用程序提供了良好的编程环境。为了能快捷高效地开发应用程序，STEP 7-Micro/WIN 软件提供了三种程序编辑器。

S7-200 通过自由口和迪文智能显示终端进行通讯。

自由口通信模式的通信协议可自由定义，通讯所需要的信息存放在特殊字节 SMB30 中，用户需要做如下说明：

1：奇偶检验
2：每一个字符的位数
3：波特率
自由口通讯模式可以接受和发送数据
代码范例如下：

```
//假设终端在(50,50)(150,150) 和（170，170）(200,200)两个位置有两个按键，分别为加减。
//按动按键会改变（0，0）位置处显示的数字。

TITLE=程序注释

Network 1 // 网络标题
// 网络注释
LD      SM0.1
CALL    SBR0

Network 2
LDB=    VB50,  16#CC
MOVB    16#0,  VB50
LPS
AB<=    VB104,  150
AB>=    VB104,  50
AB<=    VB106,  150
AB>=    VB106,  50
INCB    VB84
AENO
=       L63.7
LPP
AB<=    VB104,  200
AB>=    VB104,  170
AB<=    VB106,  200
AB>=    VB106,  170
DECB    VB84
AENO
0       L63.7
XMT     VB70,  0

TITLE=子程序注释
Network 1 // 网络标题
// 115200,n,8,1,      命令头检测，命令尾检测，超时检测。
MOVB    16#30,  VB84
MOVB    16#0,  VB50
MOVB    16#19,  SMB30
MOVB    16#F0,  SMB87
MOVB    16#AA,  SMB88
MOVB    16#3C,  SMB89
```

```
MOVB    5,  SMB90
MOVB    20,  SMB94
ATCH    INT0,  23
ATCH    INT1,  9
ENI
RCV     VB100,  0

TITLE=中断程序注释
Network 1 // 网络标题
// 网络注释
LDB=    SMB86,  16#20
AB=     VB102,  16#72
LPS
MOVB    16#CC,  VB50
AENO
CRETI
LPP
NOT
RCV     VB100,  0

TITLE=中断程序注释
Network 1 // 网络标题
// 网络注释
LD      SM0.0
RCV     VB100,  0
```

迪文智能显示终端产品在如下方面区别于 S7-200 标配的 TD200 文本显示终端：

1. 迪文显示终端不支持网络配置，不支持一带多的操作模式，只能实现一个 PLC 控制一台显示终端。
2. 迪文显示终端不会自主的去获取 PLC 的相关信息，PLC 编程人员需要编写相关代码提供给显示终端进行显示。
3. 迪文显示终端采用 232 接口，S7-200 通讯口输出需要通过一个 485 转 232 的转接器来连接。
4. 迪文显示终端是 TFT 真彩图形显示，而 TD200 是文本显示终端。

附录 4 软件模拟串口（ASM51）

由于大多数单片机（MCU）只有一个硬件串口（UART 口），如果硬件串口目前已经被其它设备使用，那么就需要用软件模拟串口来和迪文智能显示终端连接。

➢ I/O 口软件模拟串口发送（TXD）

下面的代码在标准 51 单片机上，使用任意的 I/O 口来模拟一个软件发送串口（TXD）和迪文智能显示终端连接，这段实用的代码由于不使用额外的软件或硬件资源，特别适合用户对老产品的升级改造，因为只要单片机上有一个空闲的 I/O 口，基本不改硬件，简单修改代码就可以完成产品显示的升级换代！

```
;软件模拟串口发送 1 个字节数据，发送数据在 ACC，采用延时方式
;晶体=11.0292MHz 标准 51 单片机，模拟串口速率 115200bps，n81
    UTXD     BIT     P3.4          ;软件串口发送口 TXD，可以是任意一个 I/O
    TXBYTE:  PUSH    IE            ;关闭中断，防止终端程序干扰
             PUSH    PSW
             PUSH    B
             MOV     B,R7
             PUSH    B
             CLR     EA
             CLR     UTXD          ;起始位
             CLR     UTXD
             MOV     R7,#8         ;8 个数据位
    TXBYTE1:RRC      A
             NOP
             MOV     UTXD,C
             NOP
             NOP
             DJNZ    R7,TXBYTE1
             NOP
             NOP
             SETB    UTXD          ;停止位
             POP     B
             MOV     R7,B
             POP     B
             POP     PSW
             POP     IE            ;中断恢复
             RET
;调用方式
             MOV     A,#0AAH
             LCALL   TXBYTE
```

➢ I/O 口软件模拟串口接收（RXD）

下面的代码，使用软件模拟串口接收迪文智能终端的触摸屏信息；RXD 可以使用任何一个 I/O 口，软件资源上使用 T0 定时器作为串口接收时钟。

```
;标准 51 单片机，22.1184MHz 晶体
;19200bps N81
;接收触摸屏信息 AA 72 XH XL YH YL

    URXD BIT  P1.5 ;软件串口接收 RXD，可以使用任意一个 I/O
    COMFLG   EQU  20H
             TCHOK     BIT  COMFLG.7
             SYNCOK    BIT  COMFLG.6

        RXAA     EQU      30H      ;触摸屏接收缓冲区
```

```
RX72    EQU     31H
RXXH    EQU     32H
RXXL    EQU     33H
RXYH    EQU     34H
RXYL    EQU     35H

RXLEN   EQU     40H
RXBUF   EQU     41H     ;软件串口接收缓冲寄存器

  ORG  000BH
  LJMP UART1PRO  ;软件串口接收中断

;软件串口初始化，T0 工作在 Autoload 模式
      MOV   TMOD,#22H
      MOV   TH0,#238  ;10uS 定时器中断，软件串口波特率为 19200
      MOV   TL0,#238
      CLR   TF0
      CLR   SYNCOK
      SETB  TR0
      SETB ET0
      CLR   TCHOK

;10uS T0 中断，软件模拟串口接收
UART1PRO:PUSH   ACC
         PUSH   PSW
         CLR    TF0
         JB     SYNCOK,URXDPR1
         JB     URXD,URXDPRE
         SETB   SYNCOK
         MOV    TH0,#160
         MOV    TL0,#112
         MOV    RXLEN,#8
         SJMP   URXDPRE
URXDPR1: MOV    C,URXD
         MOV    A,RXBUF
         RRC    A
         MOV    RXBUF,A
         DJNZ   RXLEN,URXDPRE
         MOV    TH0,#238        ;接收完成一个完整的字节
         MOV    TL0,#160
         CLR    SYNCOK
         JB     TCHOK,URXDPRE   ;还有触摸屏数据没有处理就不接收新的数据
         MOV    RXAA,RX72
         MOV    RX72,RXXH
         MOV    RXXH,RXXL
         MOV    RXXL,RXYH
         MOV    RXYH,RXYL
         MOV    RXYL,RXBUF
         MOV    A,RXAA
         CJNE   A,#0AAH,URXDPRE
         MOV    A,RX72
         CJNE   A,#72H,URXDPRE
         SETB   TCHOK           ;接收到触摸屏数据
URXDPRE:POP     PSW
         POP    ACC
         RETI
```

附录 5 迪文 HMI（串口智能显示终端）选型指南

迪文 HMI（串口智能显示终端）产品编号规则

DM	T	32240	S	035	_	01	W	T
迪文科技智能显示终端	B=单色 D=带灰度或伪彩色 T=真彩色	显示分辨率	M=MCU 核 T=商规 HMI S=工规 HMI K=增强型 HMI	显示区域对角线尺寸	_	硬件序列号	W=宽温（-20/+65℃以外） N=常温（-20/+65℃以内）	N=普通 T=配触摸屏 K=配键盘接口 A=配语音模组 J=老产品升级兼容 Z=ODM 产品

➢ MCU 核的单色 HMI 产品型录

显示尺寸（英寸）	产品编号	分辨率	颜　色	产品特征
2.2	DMB12232M022_01WK	122×32	单色	STN
2.8	DMB12864M028_01WK	128×64	单色	STN，黄绿膜
	DMB12864M028_11WK	128×64	单色	STN，蓝膜

➢ T 系列商规 HMI 产品型录

显示尺寸（英寸）	产品编号	分辨率	颜　色	产品特征
3.5	DMT32240T035_01WN	320×240	65K 彩色	TFT，全新 A 级屏，LED 背光。
5.6	DMT64480T056_01WN	640×480	65K 彩色	TFT，全新 A 级屏，LED 背光。
7.0	DMT80480T070_01WN	800×480	65K 彩色	TFT，全新 A 级屏，LED 背光。
10.4	DMT64480T104_01WN	640×480	65K 彩色	TFT，旧屏，CCFL 背光，双灯。

T 系列 HMI 采用和 S 系列 HMI 相同的系统架构，主要适合于一般民用或商用场合，做简单的“文本”和“图片”界面的显示（多用来对原来单色界面进行显示升级）。对于工业自动化现场、安防、电力、交通、医疗等高可靠性应用需求，请使用 S 系列（工规）或 K 系列（工规增强型）HMI。

T 系列 HMI 采用含铅工艺生产，批量订货可以提供满足 RoHS 要求的产品，单价约上浮 15%-20%。

➢ S 系列工规 HMI 产品型录

显示尺寸（英寸）	产品编号	分辨率	颜　色	产品特征	
3.5	**DMT32240S035_01W N/T/K**	**320×240**	**65K 彩色**	**LED 背光**	
	DMT32240S035_02W N/T/K	320×240	65K 彩色	LED 背光，AD-TFT，半透半反，阳光下清晰可视	
4.3	**DMT48270S043_01N N/T/K**	**480×272**	**65K 彩色**	**LED 背光**	
5.7	DMT32240S057_01N N/T/K	320×240	65K 彩色	CCFL 背光	
	DMT32240S057_11N N/T	320×240	65K 彩色	DMT32240S057_01 配 ABS 塑胶面板	
	DMT32240S057_02W N/T/K	320×240	65K 彩色	LED 背光	
	DMT32240S057_12W N/T	320×240	65K 彩色	DMT32240S057_02 配 ABS 塑胶面板	
	DMT64480S057_01W N/T/K	**640×480**	**65K 彩色**	**LED 背光，超宽温（-30-85℃），支持 USB**	
	DMT64480S057_11W N/T	640×480	65K 彩色	DMT64480S057_01 配 ABS 塑胶面板	
7.0	**DMT80480S070_02W N/T/K**	**800×480**	**65K 彩色**	**LED 背光，支持 USB**	
8.0	**DMT80600S080_01W N/T/K**	**800×600**	**65K 彩色**	**LED 背光，超宽温（-30-85℃），支持 USB**	
8.4	DMT64480S084_01W N/T	640×480	65K 彩色	托架安装，支持 USB	CCFL 背光
	DMT64480S084_21W N/T	640×480	65K 彩色	ABS 塑胶面板，支持 USB	
10.4	DMT64480S104_11W N/T/A	640×480	65K 彩色	ABS 塑胶面板，支持 USB	
12.1	DMT80600S121_01W N/T	800×600	65K 彩色	硬铝合金面板，支持 USB	
19.0	DMT64512S190_01N N	640×512	65K 彩色	LTPS 屏，配显示器壳后不支持 USB	

➢ K 系列增强型 HMI 产品型录

显示尺寸（英寸）	产品编号	分辨率	颜　色	产品特征
5.7	DMT64480K057_01WTA	640×480	262K 彩色	400cd/m²亮度，IP65 外壳，USB、RTC 和音乐播放
10.4	DMT80600K104_01WTA	800×600	262K 彩色	400cd/m²亮度，IP65 外壳，USB、RTC 和音乐播放

➢ 特殊规格 HMI 产品型录

显示尺寸（英寸）	产品编号	分辨率	颜　色	产品特征
5.7	DMT32240S057_03WKZ	640×480	65K 彩色	8×8 键盘，矿用防爆（低压，低功耗，小电容）
8.0	DMT80600S080_02WKZ	800×600	65K 彩色	4×4 键盘，矿用防爆（低压，低功耗，小电容）

附录 6 修订记录和联系方式

本文档的主要目的是提供第一次使用迪文 HMI 产品（智能显示终端）的用户一本入门参考书，更多的问题，欢迎 mail 给 HMI 产品线技术支持专用邮箱：

dwinhmi@263.net

或致电（86）10-62102630 62105007 62621271 62636805

有关迪文智能显示终端的最新资料，欢迎访问我们的网站：

www.dwin.com.cn

www.tftdriver.com

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！

日 期	修订记录	修订后版本
2008.10.08	首次发布	Ver1.0
2008.12.01	1.更新了“附录 5 选型指南”的内容，增加了 T 系列和 K 系列 HMI 的选型参考； 2.修改了 2.4 节的内容，增加了 HMI 指令集； 3.修改了 5.4 节内容，使用 0xC103 来实现曲线的缩放； 4.增加了第 7 章：“7 使用配置文件来简化设计”；	Ver2.0