专业素养. 诚实守信. 追求卓越

迪文 HMI 用户软件开发指南

（Ver3.2　2012.02）

北京迪文科技有限公司产品部

目 录

1 文本 . . . 2
1.1 在程序中直观的引用文本 . . . 2
1.2 文本框格式控制 . . . 2
1.3 ASCII 字符间距自动调整 . . . 3
1.4 文本背景色不显示 . . . 3
1.5 艺术字（特殊字符）显示 . . . 4
1.6 竖排文本显示 . . . 4
1.7 数值参数显示 . . . 5
1.8 文本滚动显示 . . . 6
2 曲线 . . . 8
2.1 动态曲线：通过曲线移动实现 . . . 8
2.2 动态曲线：通过窗口移动实现并有缩放和历史回放功能 . . . 9
3 图片、图标和动画 . . . 10
3.1 进度条显示 . . . 10
3.2 模拟表盘显示 . . . 11
3.3 图标叠加显示 . . . 11
3.4 备份和恢复当前界面 . . . 12
3.5 指针表盘显示 . . . 13
3.6 流程图动画（指令定时自动执行） . . . 14
4 触摸屏界面 . . . 15
4.1 触控界面（无须用户代码干涉）设计 . . . 15
4.2 触摸屏参数录入 . . . 16
4.3 触摸屏中文录入（GBK 字库） . . . 17
5 系统配置和外设 . . . 18
5.1 系统配置参数说明 . . . 18
5.2 RTC 时钟 . . . 19
5.3 背光亮度调整和屏保亮度设置 . . . 20
5.4 视角调整 . . . 20
5.5 内置数据库读写 . . . 21
5.6 视频播放 . . . 22
5.7 蜂鸣器 . . . 22
6 用户程序设计建议 . . . 23
7 软件相关参数说明 . . . 24
7.1 字库说明 . . . 24
7.2 图片存储数量 . . . 24
7.3 典型指令执行时间 . . . 24
附录 1 迪文 HMI 指令一览表 . . . 25
附录 2 ASM51 应用实例：转动的时钟 . . . 27
附录 3 ASM51 应用实例：最简单的示波器 . . . 33
附录 4 C51 串口通信程序参考 . . . 41
附录 5 文档涉及的 C51 函数原型 . . . 42

1 文本

指令	说 明
0x40	设置调色板。
0x42	取指定位置颜色到背景色调色板。
0x41	设置字符显示间距。
0x45	设置/取消文本框限制。
0xE0	配置显示方式： PARA1.2： 0=正常显示 1=偏转 90° 显示； PARA2.4（仅 H600、K600+支持）：0=文本显示时自动恢复背景 1=按照指定的背景色显示文本。
0x53	显示 8×8 点阵的 ASCII 字符串，对应 0x00 字库。
0x6E	显示 12×12 点阵的 GBK 中文字符串，对应 0x20 字库（中文）和 0x00 字库（ASCII）。
0x54	显示 16×16 点阵的 GBK 中文字符串，对应 0x21 字库（中文）和 0x00 字库（ASCII）。
0x6F	显示 24×24 点阵的 GB2312 中文字符串，对应 0x22 字库（中文）和 0x00 字库（ASCII）。
0x55	显示 32×32 点阵的 GB2312 中文字符串，对应 0x23 字库（中文）和 0x00 字库（ASCII）。
0x98	显示任意点阵、任意编码方式的字符串，对应客户指定的字库；但是 GB2312 和 GBK 模式下的 ASCII 字符使用 0x00 字库。
0x9C	使用图标来显示，图标背景色自动滤除。
0x9D	使用图标来显示，图标背景色会自动滤除，并且显示前先用当前页面恢复背景。
0xC108	基于暂存缓冲区的参数显示，HMI 自动完成参数的识别和显示。

1.1 在程序中直观的引用文本

C 语言

```
uchar *HZSTR="汉字显示 OK";
uint x,y,color;
prints（0x54,x,y,HZSTR）      //16 点阵显示，（x,y）是坐标位置
```

对于固定的文本，也可以直接写成

```
prints(0x54,x,y,"汉字显示 OK")
```

ASM51

```
HZSTR: DB "汉字显示测试 OK",0xFE        ;0xFE 为字符串结束符
MOV   DPTR,#HZSTR
LCALL TXROMS
```

在 GBK 编码的中文操作系统上，C 或 ASM 编译器会自动把"汉字显示测试 OK"翻译成 GBK 编码，中文 GBK 编码（扩展字库）向下兼容 GB2312（一级、二级字库）。

1.2 文本框格式控制

如右图效果的实现步骤：

- 0x41 指令设置字符间距
  dx=0 dy=12
- 0x45 指令设置文本框
  (Xs,Ys) (Xe,Ye)
- 使用 0x98 指令显示文本
  用 0x0D、0x0A 做换行控制
  C_Mode.6=0 不显示背景色

1.3 ASCII 字符间距自动调整

由于点阵字库每个字符占据固定的大小，对于类似“i”、“l”、“.”的 ASCII 字符，显示将会很松散，可以通过使用 0x98 指令显示，并设置 0x98 指令参数 C_Mode.4=1 来解决。

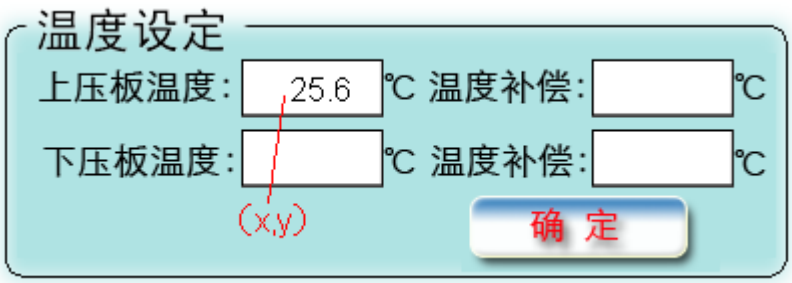
以显示 WiFi 为例：

C_Mode.4=0 时，显示为 W i F i；

C_mode.4=1 时，显示为 WiFi。

C_Mode.4=1 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）。

1.4 文本背景色不显示



文本框是纯色背景的实现步骤

- 使用 0x42 指令取背景色，取色位置（x-1,y）；
- 显示文本，显示时设置背景色、前景色都显示。

优点：刷新数据显示时，由于背景色、前景色都显示，会自动覆盖掉原来的数据。

文本框是非纯色（图片）背景的实现步骤

方法 1：

- 先使用 0x71 图片剪切指令恢复文本框显示区域背景（上图的青色方框区域），同时清除原来的文本显示；
- 再使用 0x98 文本显示指令显示，并设置文本显示的背景色不显示（C_Mode.6=0）。

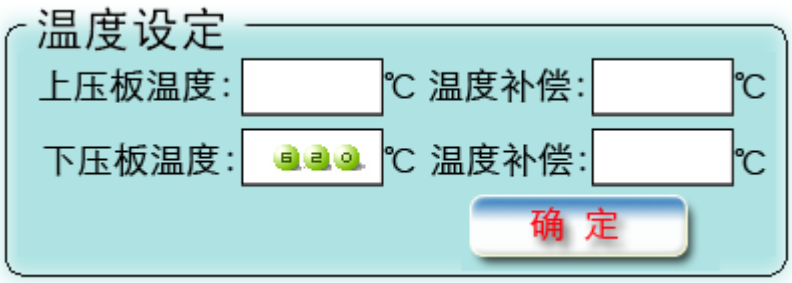
优点：适合在任何图片背景上叠加文本显示，但文本指令只能是 0x98。

方法 2（仅 H600、K600+支持）：

- 先使用 0xE0 指令修改 Para2.4=0，显示文本时自动恢复背景；
- 再使用文本显示指令显示，文本背景色自动被忽略。

优点：适合所有文本显示指令，但是背景必须是当前图片，而不能是从其它页面剪切、粘贴过来的背景。

1.5 艺术字（特殊字符）显示



要实现上图中“620”的显示：

- **显示区域在当前图片上：**

  使用 0x9D 指令从图标库（ 0 1 2 3 4 5 6 7 8 9 ）中剪切粘贴到显示区域；

```
LOGOD9D(300,200,6);
LOGOD9D(330,200,2);
LOGOD9D(360,200,0);

void LOGOD9D(uint x,uint y,uchar i)
{  uint logoxy[]={0,0,40,40,40,0,80,40,80,0,120,40,120,0,160,40,160,0,200,40,
   200,0,240,40,240,0,280,40,280,0,320,40,320,0,360,40,360,0,400,40};  //图标坐标位置
   uchar j;
   txword(0xaa9d);
   txword(50);    //图标保存的页面位置
   for(j=0;j<4;j++)
{  txword(logoxy[i*4+j];}
   txxy(x,y);
   txeof();         //发送帧结束符 0xCC 33 C3 3C
}
```

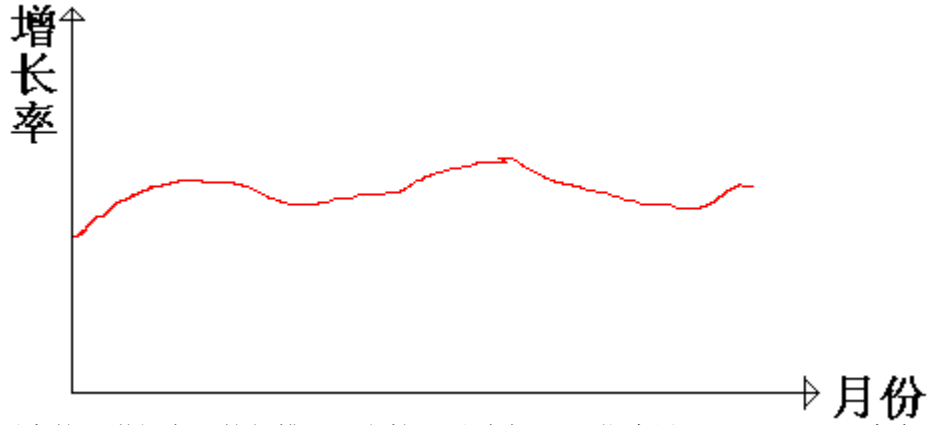
- **显示区域不在当前图片上（比如临时弹出的报警信息框）：**

  先使用 0x71 恢复显示区域以擦除上一次的显示内容；

  再用 0x9C 指令从图标库中剪切数值图标粘贴到显示区域。

```
piccut(0x71,7,0,0,90,30,300,200)      //假设背景是 0x07 页面上的一个区域图标
LOGOD9C(300,200,6);
LOGOD9C(330,200,2);
LOGOD9C(360,200,0);
```

1.6 竖排文本显示



类似于上图中的“增长率”的竖排显示文档，通过在 0x98 指令设置 C_Mode.5=1 来实现。

1.7 数值参数显示

很多时候，参数在用户程序中是以 HEX 方式来保存的，而在显示时需要转换成 ASCII 码：

```
INTDSP(100,200,620);

void INTDSP(uint x,uint y,int value)
{  uint i=10000;
   uchar j,k;
   txword(0xaa98);
   txxy(x,y);
   txbye(0x21);          //16*16 GBK
   txword(0xc201);       //前景色、背景色都显示，GBK 编码，16*16 点阵
   txword(0x0000);       //字体为黑色
   txlong(0xffff);       //背景为白色
   if(value<0)     txbyte("-");     //显示负号
   for(j=0;j<5;j++)
{  k=value/i;
   txbyte(k+0x30);
   value=value-k*i;
   i=i/10;}
   txeof();         //发送帧结束符 0xCC 33 C3 3C
}
```

但是，如果考虑到参数显示还要处理对齐，无效零不显示等问题，用户程序处理起来就会比较麻烦，使用 0xC108 指令可以方便的解决这个问题。

- 先用 0xC0 指令把要显示的参数按照格式写到暂存缓冲区；
- 再使用 0xC108 指令调用参数显示，一次可以调用多个参数；

```
INTDC108(0x6305,100,200,123450);     //在（100,200）位置右靠齐显示 16*32 点阵的 123.450

void INTDC108(unit m,uint x,uint y,long value)
{  txword(0xaac0); //先写暂存缓冲区
   txword(0x0000); //暂存缓冲区地址
   txword(m);      //显示模式
   txxy(x,y);
   txword(0x0000)  //字体为黑色
   txword(0xffff);  //背景为白色
   txlong(value);
   txeof();            //发送帧结束符 0xCC 33 C3 3C
   txbyte(0xaa);
   txword(0xc108);
   txword(0x0000);
   txbyte(0x01);
   txeof();}
```

提示：0xC108 指令另外一个作用是可以实现大批量参数的同步显示。（在显示参数很多时，如果每个参数单独发送指令显示，由于通信和数据处理要占用时间，会导致显示不同步，造成视觉上参数一个一个“蹦”出来的效果。）

1.8 文本滚动显示

有些应用中，由于显示区域大小的局限，或者为了引起用户的注意，需要进行文本滚动显示，如下图所示：

把整个文本看成一个字符串指针（数组），滚动其实就是不断修改显示起始位置的结果，横向文本滚动（单行文本框滚动显示）使用 0x71 和 0x98 指令配合实现，流程如下：

- A.先使用 0x71 指令恢复显示区域的背景；
- B.使用 0x98 指令显示指针指定位置开始的文本（注意 C_Mode.6=0，无背景色显示文本）；
- C.修改指针位置；
- 定时重复 A-C 即实现了文本的滚动显示。

如果文本显示区域的背景是纯色，那么使用 0x42 指令取色，用 0x98 指令显示文本时设置背景色也显示，则可以不必用 0x71 指令恢复背景，显示速度可以很快。

类似的，如果我们在屏幕上用 0x42 指令设置一个文本框限制，把文本框看做是文本字符串的一个"片断"，通过修改显示缓冲区指针，则可以轻松实现翻页、上下滚屏等显示效果。

```
void hzroll(uint xs,uint ys,uint xe,uint ye,uchar pos,uchar num,uchar *str)
{ uchar i,j,k;
  i=uchar(sizeof(str));     //找到指针尾位置
  logod71(0,xs,ys,xe,ye,xs,ys);   //恢复背景
  if(pos>i)  pos=0;
  txword(0xaa98);         //显示文本
  txxy(xs,ys);
  txword(0x2381);       //32*32 点阵汉字，背景色不显示
  txbyte(0x03);
  txword(0xf800);       //字体颜色为红色
  txword(0x0000);
  k=pos;
  for(j=0;j<8;j++)       //一行最多显示 8 个
{ txbyte(str+k);
  k++;
  txbyte(str+k);
  k++;
  if(k>i)  k=0;}
  pos=pos+2;}           //移动指针位置，实现滚动，每次移动 1 个汉字（2 个字节）
```

ASM51 示例

```
        JNB     RTCOK,START
        LCALL   HZROLL          //定时调用，实现滚动效果
        CLR     RTCOK

TEST:DB'尊敬的客户您好，我行新推出基金、理财产品，欢迎选购!详情咨询 9 5 5 8 0 。',0FFH,0FFH
CMD71:  DW 0AA71H,0,0,440,271,479,0,440,0CC33H,0C33CH,0FEFEH
CMD98:  DW 0AA98H,0,440
        DB 23H,81H,03H
        DW 0F800H,0000H,0FEFEH

HZROLL: MOV     DPTR,#CMD71     //恢复背景
        LCALL   TXROMS
        MOV     DPTR,#CMD98
        LCALL   TXROMS
        MOV     R7,#8           //一行只能显示 8 个汉字（272/32=8.5），数字采用全角显示
        MOV     DPH,HZ_DPH
        MOV     DPL,HZ_DPL      //显示的起始地址
HZROLL1:CLR     A
        MOVC    A,@A+DPTR
        MOV     R6,A
        INC     DPTR
        CLR     A
        MOVC    A,@A+DPTR
        MOV     R5,A
        INC     DPTR
        CJNE    R6,#0FFH,HZROLL2
        CJNE    R5,#0FFH,HZROLL2
        MOV     R4,DPH          //保存指针尾
        MOV     R3,DPL
        MOV     DPTR,#TEST
        SJMP    HZROLL1
HZROLL2:MOV     A,R6
        LCALL   TXBYTE
        MOV     A,R5
        LCALL   TXBYTE
        DJNZ    R7,HZROLL1
        MOV     A,HZ_DPL        //修改指针移动位置
        ADD     A,#2            //每次移动一个汉字（2 个字节）
        MOV     HZ_DPL,A
        CLR     A
        ADDC    A,HZ_DPH
        MOV     HZ_DPH,A
        MOV     A,HZ_DPL          //判断指针位置是否越界（超过指针尾）
        SUBB    A,R3
        MOV     A,HZ_DPH
        SUBB    A,R4
        JC      HZROLL3
        MOV     DPTR,#TEST
        MOV     HZ_DPH,DPH
        MOV     HZ_DPL,DPL
HZROLL3:MOV     DPTR,#CMDTTL    //帧结束符 CC 33 C3 3C
        LCALL   TXROMS
        RET
```

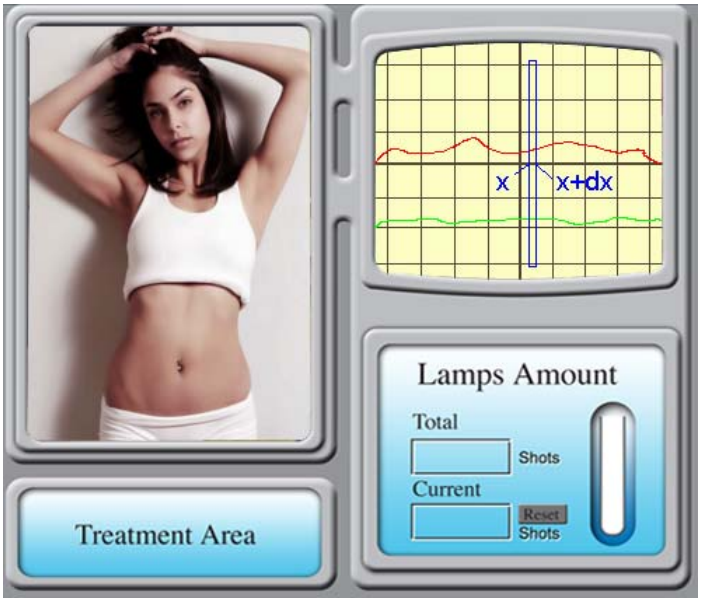
2 曲线

指令	说 明
0x40	设置调色板。
0x56	按照指定点连线。
0xC106	使用暂存缓冲区数据缩放显示带窗口限制双向折线图。

2.1 动态曲线：通过曲线移动实现

实现过程：

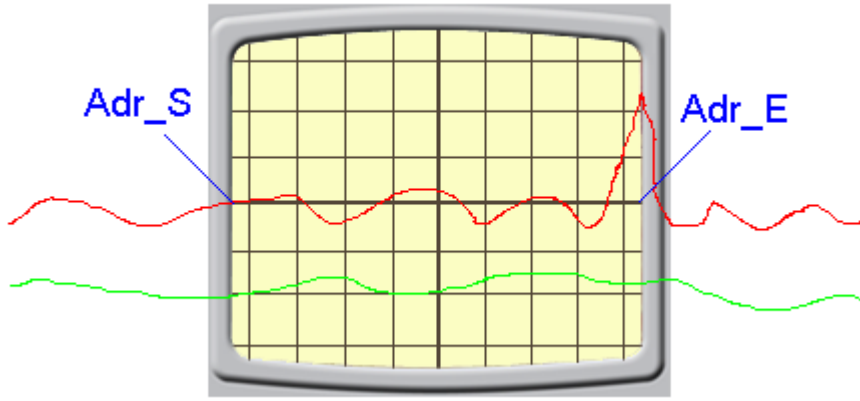
- A. 用 0x71 指令恢复（X,Ymin）（X+dX,Ymax）区域的背景图片以擦除原来的曲线显示；
- B. 使用 0x40 指令设置前景色为红色，使用 0x56 连线指令把（X，Y1now-1）和（X+dx,Y1now）连线，画出红色曲线；
- C. 使用 0x40 指令设置前景色为绿色，使用 0x56 连线指令把（X，Y2now-1）和（X+dx,Y2now）连线，画出绿色曲线；
- D. 使用 0xD0 指令强制刷新显示一次，确保看到曲线的平滑移动；
- E. x=x+dx,重复 A-E 步。




```
if(RTCOK)                       //A/D 采样时间到
{y1now=185-rdad(0x00)/2;        //第一条曲线是 CH0 8bitA/D 值,185 是 Y 坐标零点
 y2now=185-rdad(0x01)/2;
piccut(0x71,3,x,ymin,x+dx,ymax,x,ymin);          //恢复当前背景以擦除历史曲线
 setcolor(0xF800,0x0000);
 line(x,y1old,x+dx,y1now);
 setcolor(0x07e0,0x0000);
 line(x,y2old,x+dx,y2now);
 fdisp();                  //AA D0 CC 33 C3 3C 强制刷新 1 次显示
 y1old=y1now;
 y2old=y2now;
 RTCOK=0;
 x=x+dx;
 if(x>Xmax)   x=Xmin;}
```

由于串口通信速度和指令执行时间限制，曲线移动方式实现的动态曲线显示方式只适合低采样率的场合（一般不超过 10 次/秒）。

2.2 动态曲线：通过窗口移动实现并有缩放和历史回放功能



借助暂存缓冲区，我们可以把曲线“暂存”在 HMI 中，当前显示窗口不过是整个曲线的一个“片断”而已，通过修改显示的缓冲区起始地址，可以非常方便的实现曲线移动和历史回放，如上图所示。

0xC106 指令对暂存在缓冲区的曲线进行自动比例缩放显示，并对曲线窗口越界进行判断。

```
if(RTCOK)            //A/D 采样时间到
{WRbuffer(Buf_adr,rdad(0x00));  //保存 A/D 值到暂存缓冲区
 Buf_adr++;
  WRbuffer(Buf_adr,rdad(0x01));
 Buf_adr++;
 if(Buf_adr>40959)    Buf_adr=0;}

if(DSPOK)            //显示刷新时间到
{piccut(0x71,3,xmin,ymin,xmax,ymax,xmin,ymin);       //恢复当前窗口背景以擦除历史曲线
 DisBufLine(Dis_adr,100,0xf800);      //显示第 1 条曲线
 DisBufLine(Dis_adr+1,100,0x07e0);  //显示第 2 条曲线
 Dis_adr=Dis_adr+200;            //窗口宽度为 100 点，两条曲线，每次挪动整个窗口长度
if(Dis_adr>40959) Dis_adr=0; }

void DisBufLine(uint adr,unit num,unit color)    //0xC106 显示暂存缓冲区的曲线
{txbyte(0xaa);
 txword(0xc106);
 txword(adr);
 txword(xmin);
 txword(ymax);     //Y 轴零点位置，如果曲线有负值，则为(ymin+ymax)/2
 txword(num);
 txword(0x0202)    //缓冲区地址间隔为 2，X 坐标间隔为 2
 txword(0x0102)    //曲线放大 0x01/0x02=0.5 倍
 txword(color);
 txword(ymin);     //显示曲线窗口限制
 txword(ymax);
 txeof();}
```

迪文 HMI 暂存缓冲区最大长度为 40Kword，可以保存最大 40960/N 个采样点(N 为曲线条数)，对于 1 条曲线，200SPS（每秒 200 条记录）采样速率下，可以存储 204.8 秒的历史曲线。

更具体的 ASM51 代码实例，请参考附录 3。

3 图片、图标和动画

指令	说 明
0x70	显示保存在终端中的一幅全屏图片。
0x71	从保存在终端中的一幅图片上剪切一个区域粘贴到当前显示页面的指定位置。
0x9C	从保存在终端中的一幅图片上剪切一个区域，过滤掉背景后粘贴到当前显示页面的指定位置。
0x9D	从保存在终端中的一幅图片上剪切一个区域，过滤掉背景后粘贴到当前显示页面的指定位置；在粘贴前先自动执行一次 0x71 指令，用当前页面恢复背景。
0x9E	从保存在终端中的一幅图片上剪切一个区域，旋转指定角度后，粘贴到当前显示页面的指定位置；粘贴可以选择是否过滤掉背景色。仅 H600、K600+支持。
0x97	显示 ICON 图标库中的一个图标。仅 K600+支持。
0xE2	保存当前显示图片到终端中。
0x59	以前景色显示矩形框。
0x69	以背景色显示矩形框。
0x5A	以前景色填充矩形区域。
0x5B	以背景色填充矩形区域。
0x5C	对指定矩形区域进行反色操作（再反色将恢复原来显示）。
0x64	对指定的封闭区域进行填充。
0x9A	开启/关闭自动指令循环；指令配置文件保存在 0x1C 字库位置。

理论上，由于迪文 HMI 支持 24bit 不压缩图片，所以任意复杂的 GUI 都可以分解成图标来实现。

3.1 进度条显示

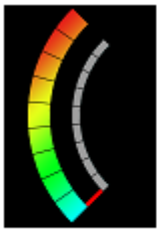
以上面的进度条为例，进度条的实现包括以下 3 步（更炫的水晶进度条用图标剪切方式来实现）：

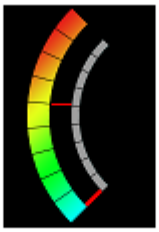
- A.先用 0x40 指令设置好前景色（黑色边框）、背景色（白色背景）；
- B.用 0x59 画出边框，用 0x5B 填充背景；
- 用 0x40 设置好前景色（黑色字体）、背景色（蓝色进度）；
- D.按照进度，用 0x5B 指令进行填充；
- E.把进度值显示出来，显示使用无背景文本显示（0x98，C_Mode.6=0）。

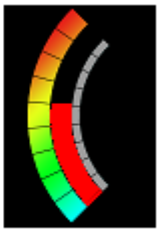
```
stepdsp(100,100,0x0000,0xffff,0x0000,0x001f,100,75)

//显示一个进度条，(x,y) 是显示位置，w 是进度条宽度，高度为 22，字体为 16 点阵，st 是进度
void stepdsp(uint x,uint y,uint color1,uint color2,uint color3,uint color4,uchar w,uchar st)
 {setcolor(color1,color2);
  fillw(0x59,x,y,x+2+w, y+22);
  fillw(0x5a,x+1,y+1,x+3+w,y+21);          //以上显示了一个带边框的实矩形
  setcolor(color3,color4);
  fillw(0x5a,x+2,y+2,x+2+w*st/100,y+20); //进度填充
  hex2dsp(x+w/2,y+3,st);                  //进度显示，注意使用 0x98 指令，C_Mode.6=0
  print98(0x21,0x81,0x01,color3,color4,x+24+w/2,y+3,"%");}
```

3.2 模拟表盘显示

模拟表盘其实是一种“非线性”的特殊进度条，使用 0x64 指令直接填充即可，过程如下图所示：


底图


画好刻度线


0x64填充刻度区

```
setcolor(0xf800,0x0000);  //设置连线颜色为红色
line(30,55,40,56);        //连线
fillfree(48,100,0xf800);  //以红色填充需要显示的半圆区域

//以（x,y)为种子点，以 color 颜色向四周任意边界区域填充
void fillfree(uint x,uint y,uint color);
{ txword(0xaa64);
  txxy(x,y);
  txword(color);
  txeof();}
```

如果填充区域边界不是凸多边形连通，那么用户程序需要进行分段填充。

3.3 图标叠加显示

有时候需要在图片上叠加显示图标（比如上图左下角的电池图标），使用 0x9D 指令实现。

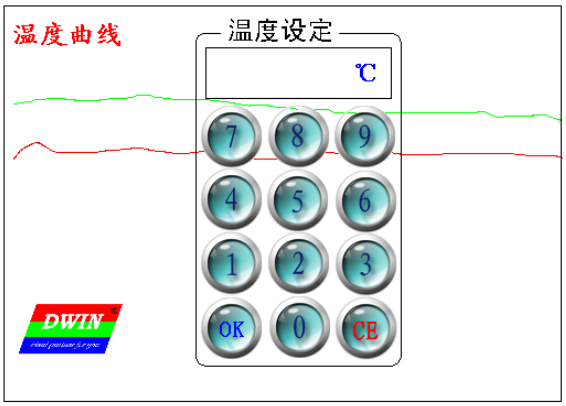
- 先设计好图标，保存在一幅图片上，注意图标背景（比如图中的灰色）必须是纯色，如下图所示：

- 使用 0x9D 指令根据需要剪切相应的区域，粘贴到需要的位置，比如上面的例子：

```
logod9d(50,70,0,140,40,0,440);      //假设电池图标保存在第 50 幅图片上
```

注意：由于 0x9D 指令自动恢复的是当前图片的背景，如果要粘贴的区域本身也是从别的图片上剪切过来的图标，那么就使用 0x71 指令先恢复背景，再用 0x9C 指令来显示图标。
对于 K600+内核迪文屏，推荐使用 0x97 指令来更方便的设计和调用图标。

3.4 备份和恢复当前界面



如上图所示，应用中经常会遇到“临时中断”当前界面的情况，描述如下：

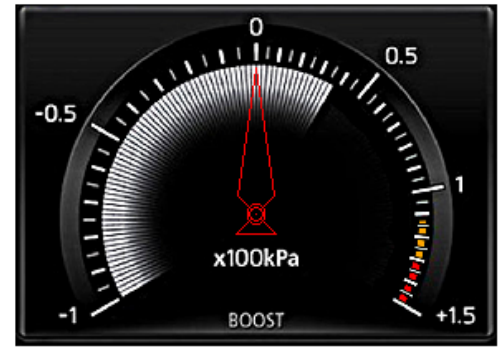
即在当前的非图片背景（文本、曲线）下需要弹出一个对话框（比如上图中的键盘），在操作完成后再回到正常的显示状态，为了恢复原来的背景，要进行参数状态的记录并重新刷新，程序设计比较麻烦。

使用 0xE2 和 0x70 指令配合可以很好的解决这个问题。

- 弹出窗口前，先把当前显示用 0xE2 指令备份保存在一个空余图片位置（比如第 50 幅）；
- 再弹出窗口；
- 窗口操作完成后，使用 0x70 指令把原来保存的界面恢复出来。

迪文 HMI 内置的图片存储器使用 NAND Flash，同一个图片位置最多可重复保存 10 万次。

3.5 指针表盘显示



图片 0　　图片 1　　图片 0 叠加指针后效果

如上图所示的指针表盘具有直观、动态感强的特点，在工程显示中应用很多，利用 0x9E 指令（仅 H600、K600+支持）可以很方便的实现。

- 用户先把表盘和指针分开设计，如上图的图片 0 和图片 1，为了软件调用方便，把指针原始图标放在“0 刻度”位置；
- 把图片 0 和图片 1 保存到终端中；
- 程序中，根据指针位置计算出旋转角度，使用 0x9E 指令把指针图标剪切、旋转粘贴到表盘中，粘贴前先用 0x71 指令恢复表盘背景以擦除上一个指针位置。

```
posdsp(1.2);
//指针位置显示，原始表盘在 0 页面，指针在 1 页面
//value  0-1=0-80° ，1-1.5=80-135°  0--1=360-225°
void posdsp(float value)
{  uint alpha;     //指针偏转角度
   if(value<0)   alpha=360-value*135;          //根据参数计算出旋转角度
   else
   {if(value>1)  alpha=80+(value-1)*110;
    else   alpha=value*80;}
   alpha=alpha*2;            //旋转角度的单位是 0.5°
   piccut(0x71,0,304,158,509,338,304,158)     //恢复表盘背景以擦除原来的指针显示
    txword(0xaa9e);
    txbyte(0x00);                //背景色滤除
    txword(0x0001);              //指针在图片 1 上
    txarea(761,483,790,586);  //指针在图片 1 上的区域位置
    txxy(776,572);               //指针在图片 1 上的旋转中心坐标
    txword(alpha);               //指针旋转角度
    txxy(401,247);               //指针粘贴到图片 0 后旋转中心坐标
   txeof();}
```

由于 0x9E 指令可以对任意的图标进行剪切、旋转、粘贴，除指针表盘外可以衍生出很多其它的应用：

- 电机、齿轮等圆周运动图标的转动；
- 设计一些卡通图标，使用 0x9E 指令剪切粘贴时，修改旋转角度和旋转中心坐标，可以很容易的获得滚动、跳跃的动画效果；
- “指南针显示模式”：即不管用户怎么摆放显示屏，通过角度传感器（重力传感器或者码盘）反馈控制调整，始终让显示内容正对操作者。

3.6 流程图动画（指令定时自动执行）

在自动化工艺流程图的显示中，我们希望能够动态的把工艺流程展现给用户，那么就涉及到很多小区域的动画（定时把显示内容更新）显示，以呈现用户运动的效果，比如传送带的运转，管线里面液体的流动等等。

如果这些动画都依靠用户程序定时刷新，会带来 3 个问题：

（1）客户需要写大量代码，而这些代码在界面修改时需要重新修改，维护工作量大；

（2）频繁的发送数据，会占用用户 CPU 较多的资源；

（3）受串口通信速度限制，难以实现高速、大范围的动画效果。

使用 0x1C 配置文件和 0x9A 指令配合可以简单的解决这个问题，步骤如下：

- A.把实现动画的指令编辑成配置文件（每个指令组最多可以包含 64 条指令，最多 8 组）；
- B.把配置文件下载到终端的 0x1C 字库位置；
- C.根据需要，在用户程序中开启或中止指令组的自动执行；指令组开启后将按照用户设置的时间间隔自动执行，不再需要用户程序干涉。

```
  void cmdon(uchar cmd_id)
{ txword(0xaa9a);
  txbyte(cmd_id);       //开启 cmd_id 指定的指令组
  txeof();}

  void cmdoff()
{ txword(0xaa9a);
  txbyte(0xff);              //关闭指令组自动执行
  txeof();}
```

4 触摸屏界面

指令	说 明
0xE4	触摸屏校准。
0xE0	配置触摸屏处理模式，参数掉电不保存。
0xE3	配置触摸屏处理模式，参数掉电保存。
0x7C	使用触摸屏输入 ASCII 字符或中文，仅 H600、K600+支持。
0x72	HMI 上传数据：触摸坐标位置。
0x73	
0x78	HMI 上传数据：触控键码。
0x79	
配置文件	说 明
0x01	如果使用触摸屏中文输入法，须在 0x01 位置下载 GBK 词库。
0x1A	触控指令文件，用于保存用户需要特殊回传的指令。
0x1E	触控切换配置文件，使用迪文提供的 PC 工具软件生成。

完备的触控和中英文触摸屏输入功能，使基于迪文 HMI 的人机交互设计真正接近“零代码”!

4.1 触控界面（无须用户代码干涉）设计

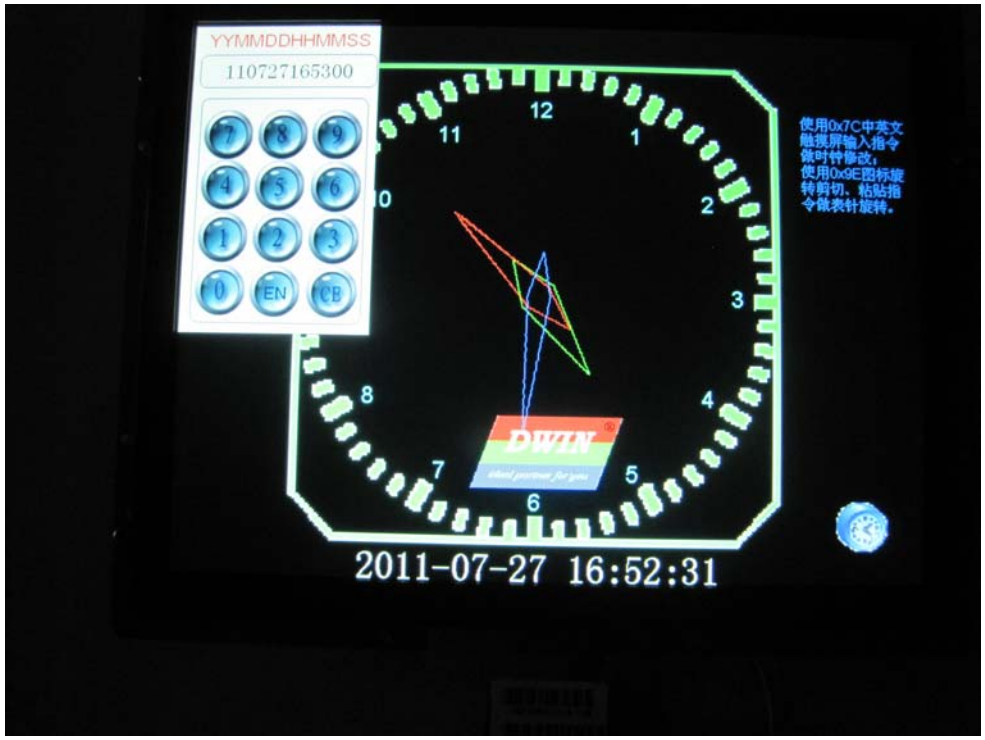
迪文触控界面配置软件界面说明

触控界面开发的步骤：

- 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；
- 第 2 步：使用“迪文触控界面配置软件”生成配置文件；
- 第 3 步：把配置文件下载到 HMI（终端）中，0xE0 指令配置 HMI 为触控模式；
- 第 4 步：配合界面切换写一些辅助代码，比如曲线、参数的刷新，参数录入等。
- 调试、定版。

采用触控方式时，用户软件架构是“消息触发”的并行功能模块，调试和维护非常方便。

4.2 触摸屏参数录入



如上图所示，希望点击时钟右下角的“闹钟”图标时弹出左上角的对话框进行时间的修改，使用 0x7C01 触摸屏 ASCII 字符输入指令可以很方便的实现。

```
uchar CMD7C[]={0xaa,0x7C,0x01,0x00,0x01,0,1,0,34,0,40,12,3,0,0,0xcc,0x33,0xc3,0x3c};

    if((Rxcmd==0x78)&&(Rxdat0==0x00)&&(Rxdat1==0x01))    //收到触控键码
    { piccut(0x71,1,0,0,206,344,0,0);      //图标剪切方式弹出输入小键盘
      txstr(CMD7C);                  //发送触摸屏 ASCII 字符输入指令，后面由 HMI 完成输入
    }
    if((Rxcmd==0x7c)&&(Rxdat0==0x00)&&(Rxdat1==0x01))    //触摸屏输入结束，检查 R_ID 是否匹配
    { piccut(0x71,0,0,0,206,344,0,0);      //关闭输入小键盘
     wrrtc();                  //输入数据保存串口接收缓冲区
    }
```

- 0x7C 指令让用户指定一个输入返回的 ID，做为“消息”以触发用户程序进行相应的处理。
- HMI 处理 0x7C 指令期间，除触摸屏完全由 HMI 处理，不再返值给用户外，HMI 其它功能都正常，即输入法期间，用户 CPU 可以正常处理其它程序而不用等待触摸屏返回，很方便的实现多任务调度。比如在上面的例子中，触摸屏输入期间，时钟和表盘指针仍旧正常走动。

0x7C 指令仅 H600、K600+支持。

本示例的完整 ASM51 代码请见附录 2：转动的时钟。

4.3 触摸屏中文录入（GBK 字库）

如上图所示的触摸屏中英文混合输入来修改 24 个不同的参数，可以使用 0x7C02 触摸屏中英文字符输入指令实现。0x7C02 指令和 0x7C01 的调用基本相同，下面以 ASM51 来说明：

```
        MOV     A,RXCMD                 ;串口收到有效数据帧
        CJNE    A,#78H,START2           ;触控键码
        MOV     DPTR,#CMD7C1            ;启动中英文混合输入法
        LCALL   TXROMS
        MOV     R0,#RXDAT
        MOV     R7,#2                   ;把键码做为参数返回的索引
        LCALL   TXREG
        MOV     DPTR,#CMD7C2
        LCALL   TXROMS
        LJMP    STARTE
START2: CJNE    A,#7CH,STARTE           ;触摸屏输入法结束返回
        MOV     DPTR,#CMD70
        LCALL   TXROMS                  ;回到界面 0
        MOV     A,RXDAT
        CJNE    A,#02H,START20
        MOV     A,RXIDL                 ;预设的 R_ID，用以指示修改哪个参数
        MOV     B,#32
        MUL     AB
        MOV     DPH,B
        MOV     DPL,A
        MOV     R0,#RXNUM
        MOV     A,@R0
        JZ      STARTE                  ;返回的参数数据长度
        LCALL   WRXRAM                  ;保存参数
```

0x7C 指令仅 H600、K600+支持。

5 系统配置和外设

指令	说 明
0x5E	关闭背光或设置触控背光模式。
0x5F	调节背光亮度。
0x79	蜂鸣器鸣叫指定长度时间。
0x9B	在当前界面 叠加/关闭叠加 RTC 时钟显示。
0x9B5A	读取 RTC 公历时间。
0x9B5B	读取 RTC 农历时间。
0xE7	调整 RTC 时间。
0xE0	配置背光、蜂鸣器、显示视角、串口模式，参数掉电保存。
0x90	写内部数据库。
0x91	读内部数据库。
0x7A	HMI 和视频功能切换。
0xD0	强制刷新一次全屏显示（仅 K600、H600、K600+支持）。

5.1 系统配置参数说明

发送和返回的指令：

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

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

0xE0 配置参数掉电不保存；需要配置参数掉电保存，请使用迪文提供的工具软件。

- <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，指令执行后会自动后延刷新时间，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	0=停止自动刷新，显示必须通过 0xD0 指令来刷新； 1=启用自动刷新，刷新间隔由 Para2.7 决定。
.1	保留，写 1
.0	保留，写 1

5.2 RTC 时钟

绝大多数迪文 HMI 都内置有 20000-2099 的 RTC 时钟，可以使用指令修改或读取，也可以按照用户要求在指定的背景图片上自动叠加显示。

HMI 内置有备份电池，断电后 RTC 仍旧正常运行。

```
void rtcon(unit x,unit y,unit color,uchar mode,uchar scale) //在（x,y)位置启用 RTC 叠加显示
{ txword(0xaa9b);
  txbyte(0xff);
  txbyte(mode);
  txbyte(scale);
  txword(color);
  txxy(x,y);
  txeof();}

void rtcoff()        //关闭 RTC 叠加显示
{ txword(0xaa9b);
  txbyte(0x00);
  txeof();}

void rtcset(uchar *s)  //设置 RTC，S=YYMMDDHHMMSS，BCD 码 比如 0x11 0x07 0x25 0x12 0x00 0x00
{ txword(0xaae7);
  txword(0x55aa);
  txword(0x5aa5);
  txstr(s);
  txeof();}

void rdrtc_en()    //读取公历时钟，回传 BCD 码，格式为 YY MM DD WW HH MM SS,WW 为星期
{ txword(0xaa9b);
  txbyte(0x5a);
  txeof();}

void rdrtc_cn()    //读取农历时钟，格式为 YY MM DD 生肖（GBK） 天干（GBK） 地支（GBK）
{ txword(0xaa9b);
  txbyte(0x5b);
  txeof();}
```

5.3 背光亮度调整和屏保亮度设置

只要是 LED 背光的迪文 HMI，均支持 64 级背光亮度调整；极少数 CCFL 背光 HMI，考虑到亮度调整会影响灯管寿命和显示效果，只支持背光开关控制。

```
light(50);     //把背光亮度调整到 50%
void light(uchar i)    //i=0-100%,对应 0-64 的亮度调节
{txword(0xaa5f);
 txbyte(uchar((i*64)/100));}
```

在应用中，有时候需要进行屏保亮度控制，即：

- 用户正常操作时，亮度正常，假设为 L1 亮度；
- 用户长时间（比如 t 时间）不点击触摸屏或者键盘时，把亮度调低到 L2 亮度；
- 亮度 L2 屏保时，用户再次点击触摸屏或者键盘时，第一次点击被忽略，但亮度调高到 L1。

这种需求，对于带有触摸屏的 HMI，可以使用 0x5E 和 0xE0 指令配合设置一次即可：

```
lightset(100,20,10);    //设置正常为最大亮度，屏保为 20%亮度，触摸屏触发时间为 10 秒
//设置触摸屏背光控制：点亮亮度为 l1,屏保亮度为 l2,触摸屏触发有效时间为 t 秒(0-127)
void lightset(uchar l1,uchar l2,uchar t)
{rdconfig();          //回读配置参数
PARA1=PARA1|0x10;   //配置背光可以由触摸屏控制
wrconfig();          //0xe0 指令改写配置参数
txword(0xaa5e);
txxy(0xaa55,0x5aa5);
txbyte(uchar((l1*64)/100));
txbyte(uchar((l2*64)/100));
if(t>127)   t=127;
txbyte(t*2);
txeof();}
```

5.4 视角调整

通过 0xE0（或者 0xE3）指令的调整，迪文 HMI 可以实现任意视角的软件调整显示，如下图所示：

PARA1.2=0 PARA2.5=1

PARA1.2=1 PARA2.5=1

PARA1.2=0 PARA2.5=0

PARA1.2=1 PARA2.5=0

上面图片上的“**迪文科技**”都是使用文本显示指令写在（0,0）位置的，图片下载时须在工具栏进行相应设置，如下图所示：

工具栏选择
0° =0° 和 180° ；
90° =90° 和 270° 。

5.5 内置数据库读写

迪文 HMI 内置最大 30.063MB（30MB 顺序存储，64KB 随机存储）NAND Flash 数据库，可以通过 0x90、0x91 指令进行读写，以方便用户二次开发数据记录、存储功能。

```
void wrdbaser(uint adr,uchar *s)  //写随机数据存储器
{ txword(0xaa90);
  txword(0x55aa);
  txword(0x5aa5);
  txword(0x01de);
  txword(adr);
  txstr(s);
  txeof();}

void wrdbases(long adr,uchar *s)  //写随机数据存储器
{ txword(0xaa90);
  txword(0x55aa);
  txword(0x5aa5);
  if(adr>0x01dfffff)   adr=0;   //地址越界检查
  txlong(adr);
  txstr(s);
  txeof();}

void rddbaser(uint adr,uint len,uchar *s)  //读随机数据存储器
 {uint i;
  txword(0xaa91);
  txword(0x01de);
  txword(adr);
  txword(len);
  txeof(); //数据接收，如果回读数据比较多，串口接收缓冲区必须足够大，否则采用下面的查询接收方式
  EA=0;            //关闭中断
  while(!(RXDATA==0xaa))  rxbyte(200);   //串口接收一个字节，200mS 没有接收到（超时）退出
  for(i=0;i<6;i++)
 {rxbyte(200);}
  for(i=0;i<len;i++)
 {rxbyte();
  *s=RXDATA;
  s++;}
  EA=1;}

void rddbases(long adr,uint len,uchar *s)  //读顺序数据存储器
 {uint i;
  txword(0xaa91);
  txlong(adr);
  txword(len);
  txeof();  //数据接收，如果回读数据比较多，串口接收缓冲区必须足够大，否则采用下面的查询接收方式
  EA=0;            //关闭中断
  while(!(RXDATA==0xaa))   rxbyte(200);  //串口接收一个字节，200mS 没有接收到（超时）退出
  for(i=0;i<6;i++)
 {rxbyte(200);}
  for(i=0;i<len;i++)
 {rxbyte();
  *s=RXDATA;
  s++;}
  EA=1;}
```

5.6 视频播放

有些型号的迪文 HMI 支持模拟视频（CVBS 信号，比如 DVD 视频输出，摄像头视频输出）播放或者基于 SD 卡的数字多媒体（DVD、MP3、MP4）播放，这些播放都是全屏的，用户需要在 HMI 模式和视频模式之间进行切换。

```
videoplay(PAL,0x00)  //播放 CH0 通道，PAL 格式的视频

sdplay(PLAY);  //播放当前文件目录下的音视频文件
sdplay(NULL);

//模拟视频播放
#define PAL  0x00    //PAL video
#define NTSC 0x01    //NTSC video
//数字视频播放
#define NULL   0x00   //空操作，每次按键松开须发送一次
#define PLAY   0x01   //播放、暂停
#define SPKON  0x02   //扬声器输出打开
#define SPKOFF 0x03   //扬声器输出关闭
#define ENTER  0x04   //确认
#define DOWN   0x05   //后一个或快进
#define UP     0x06   //前一个或快退
#define CANCEL 0x07   //取消
#define SPKADD 0x08   //音量增加
#define SPKSUB 0x09   //音量减小
#define LANG   0x0a   //中文、英文菜单提示切换

void hmimode()    //返回 HMI 模式
{ txword(0xaa7a);
  txbyte(0x00);
  txword(0x0000);
  txeof();}

void videoplay(uchar mode,uchar ch)   //切换到模拟视频显示模式
{ txword(0xaa7a);
  txbyte(0x01);
  txbyte(mode);
  txbyte(ch);
  txeof();}

void sdplay(uchar sdkey)   //SD 数字视频播放控制
{ txword (0xaa7a);
  txbyte(0x01);
  txbyte(sdkey);
  txeof();}
```

在视频播放模式，HMI 功能仍旧正常，只是屏幕没有显示出来而已。

5.7 蜂鸣器

绝大多数迪文 HMI 都带有一个蜂鸣器，用做触摸屏伴音；用户也可以发送指令控制蜂鸣器鸣叫：

```
buzzon(100);       //蜂鸣器鸣叫 1 秒

void buzzon(uchar ontime)  //蜂鸣器鸣叫指定时间，单位为 10mS
 { txword(0xaa79);
   txbyte(ontime);
   txeof();}
```

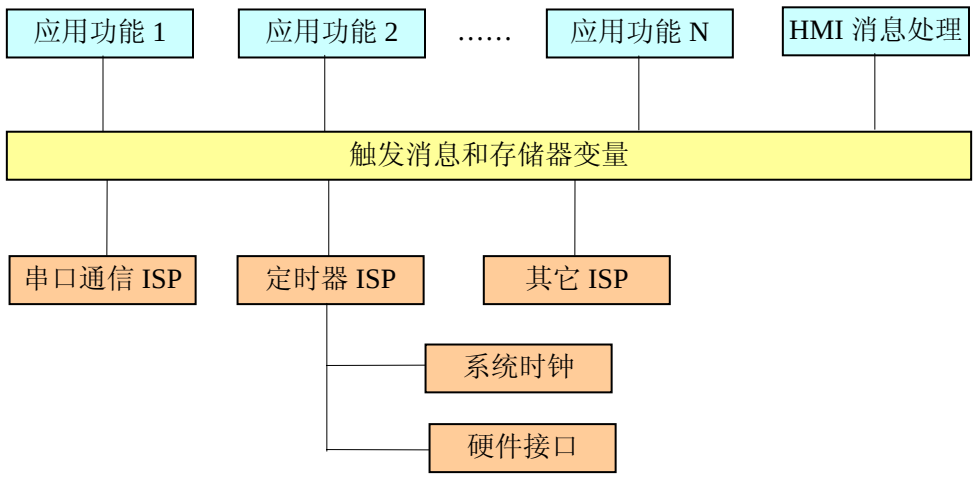
6 用户程序设计建议

对于使用迪文 HMI 来做人机交互的用户，尤其是使用迪文触摸屏 HMI 的用户，其用户程序基本上被简化成两大块功能：

- 产品算法功能的实现，比如数据采集、外设控制等，这是用户非常熟悉的程序工作；
- 人机交互的实现，借助迪文 HMI 来完成，一般新用户不是特别熟悉。

由于迪文 HMI 只通过串口和用户程序“沟通”，属于相对独立的系统，并且 HMI 处理的是“随机”的事件（和操作员打交道），而产品算法功能是属于处理相对“固定”的事件，所以用户程序设计上建议按照下面的架构来安排，以提高开发效率并提高程序的稳定性：

- 采用前后台的程序架构，把应用程序放在前台，使用查询、扫描的方式来处理；把和硬件直接打交道的程序，比如串口通信、A/D 转换等放在后台，用中断方式处理；
- 前后台程序通过存储器（全局变量）来交换数据；
- 前台程序通过后台“消息”来触发；
- 使用一个定时器产生内部时序来协调不同应用程序；
- 所有的应用程序按照功能来设计成独立的子程序，并在前台程序中处于并行的位置，以方便调试和移植（可以多个工程师同时独立工作，调试完成直接集成）。



基于迪文带触摸屏的 HMI 来开发产品，其基本流程如下：

- 硬件工程师设计硬件；
- 专业美工进行界面图片设计（此项工作往往不引起工程师重视，导致最终产品很难“出彩”，其实委托专业美工设计，一般收费不过 50-200 人民币/页）；
- 使用迪文触控配置软件进行触控界面设计；
- 让客户（或者研发团队内部）对产品界面进行试用操作、修改，直到满意为止；
- 软件工程师开始软件设计，如果功能比较多，可以按照上面的架构图分解成几个独立模块来并行进行开发；
- 集成、调试、测试、验收、定版。

具体的软件架构示例请参考本文档附录部分的案例代码。

7 软件相关参数说明

7.1 字库说明

字库 ID	字库大小	内容	说明	备注
0x00	128KB	DWINASC.HZK	不同点阵 ASCII 字符的字库集	出厂预装
0x01	128KB	DWIN_PY_GBK_01.BIN	GBK 拼音输入法词库	保留
0x02-0x19	128KB	未定义	用户可以存储自己需要的字库	
0x1A	128KB	用户定义	0x1E 触控指令索引的用户指令库	保留
0x1B	128KB	用户定义	键控配置文件	保留
0x1C	128KB	用户定义	自动循环执行指令库	保留
0x1D	128KB	用户定义	图标显示配置文件	保留
0x1E	128KB	用户定义	触控配置文件	保留
0x1F	128KB	未定义	用户可以存储自己需要的字库	
0x20	1MB	GBK12_宋体.HZK	12×12 点阵，宋体，GBK 编码字库	出厂预装
0x21	1MB	GBK16_宋体.HZK	16×16 点阵，宋体，GBK 编码字库	出厂预装
0x22	1MB	GB2312_24_宋体.HZK	24×24 点阵，宋体，GB2312 编码字库	出厂预装
0x23	1MB	GB2312_32_宋体.HZK	32×32 点阵，宋体，GB2312 编码字库	出厂预装
0x24-0x3B	1MB	未定义	用户可以存储自己需要的字库	

7.2 图片存储数量

分辨率	M100、M600、K600 内核迪文屏		H600、K600+内核迪文屏	
	标准（128MB）	扩展（1GB）	标准（128/256MB）	扩展（2/3GB）
320×240	374	3888	708	7456
480×272	374	3888	708	7456
640×480	149	1555	236	2485
800×480	124	1296	236	2485
800×600	93	972	177	1864
1024×600	74	777	141	1491
1024×768	62	648	118	1242
1280×800	***	***	88	932
1365×768	***	***	88	932

如果用户使用 64KB 的随机数据库，图片存储数量不会减少；

如果用户使用 30MB 的顺序数据库，图片存储数量按照标准（128MB）数目减少 1/3。

7.3 典型指令执行时间

分辨率	刷新频率 (Hz)			0x70 (全屏图片显示,mS)			0x54 (16*16 点阵汉字显示,mS)			0xD0 (刷新全屏,mS)	
	M100 M600	K600	H600	M100 M600	K600	H600	M100 M600	K600	H600	K600	H600
320×240	70	70	70	60	12	9	0.40	0.11	0.09	3.5	3.5
480×272	60	70	70	85	20	15	0.30	0.11	0.09	6.0	6.0
640×480	55	67	67	225	48	38	0.38	0.11	0.09	14.0	14.0
800×480	***	68	68	***	125	55	***	0.11	0.09	20.0	20.0
800×600	***	68	68	***	130	56	***	0.11	0.09	21.0	21.0
1024×600	***	75	75	***	205	67	***	0.11	0.09	29.0	29.0
1024×768	***	68	68	***	260	85	***	0.11	0.11	27.0	27.0
1280×800	***	***	67	***	***	145	***	***	0.12	***	48.0
1365×768	***	***	81	***	***	125	***	***	0.11	***	37.0

0x70 指令执行时间包括了自动执行一次 0xD0 指令时间；

0x54 指令时间指显示一个 16×16 点阵汉字的时间，其它大小汉字按照点阵规模比例换算；

H600，800×600 分辨率典型指令时间（其它内核和分辨率按 0x54 指令相对比例换算）如下表：

指令	0x51	0x42	线段（比如 0x56）	区域（比如 0x5A）	0xE2
H600_800*600 时间	1.4uS/点	1.1uS	(0.8-1.5)uS/点	(1.6*行数+0.4uS*点数) uS	80mS

附录 1 迪文 HMI 指令一览表

类 别	指令	说 明
握 手	0x00	查看配置和版本信息。
显示参数配置	0x40	设置调色板。
	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 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	任意点阵，任意编码字符串显示。
	0x45	开启/关闭文本框限制。
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	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	快速显示连续的同底垂直线段（频谱）。
	0x76	快速显示折线图。
	0x78	偏移量连线。
圆弧和圆域	0x57	反色/显示 多个圆弧或圆域。
矩形框	0x59	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	双色位图填充指定区域。
	0x64	指定区域填充。
	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	显示一幅全屏图像。
	0x7B	显示一幅全屏图像并计算 CRC 校验和。
	0x71	从指定图片剪切图标粘贴到当前显示页。
	0x9C	从指定图片剪切图标滤除背景粘贴到当前显示页，粘贴前先自动恢复原背景。
	0x9D	从指定图片剪切图标滤除背景粘贴到当前显示页。
	0x9E	从指定图片剪切图标，旋转指定角度后粘贴到当前显示页，背景滤除可选择。
	0xE2	将当前显示画面保存到终端中。
	0x97	显示 ICON 图标
	0x99	用户自定义图标显示。
	0xD0	强制刷新一次全屏显示。
动画支持	0x9A	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	写数据到暂存缓冲区。
	0xC101	显示暂存缓冲区预置的数据点。
	0xC102	显示暂存缓冲区预置的数据线。

	0xC103	使用暂存缓冲区的数据点连线（曲线动态缩放）。
	0xC104	使用暂存缓冲区的数据点高速无闪烁连线（示波器）。
	0xC105	使用暂存缓冲区数据缩放显示折线图。
	0xC106	使用暂存缓冲区数据缩放显示窗口限制双向折线图。
	0xC107	清空置点缓冲区。
	0xC107	在置点缓冲区置点。
	0xC107	恢复置点缓冲区显示到当前显示页面。
	0xC108	使用暂存缓冲区来显示参数。
	0xC110	使用暂存缓冲区缓冲指令实现同步显示。
	0xC2	从暂存缓冲区回读数据。
数据库操作	0xF2	下载字库。
	0x90	写数据到用户数据库。
	0x91	从用户数据库读数据。
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，触摸屏松开时，预设键码自动上传。
	0x79	触控界面自动切换模式下，触摸屏按下时，预设键码自动上传。
	0x7C01	触摸屏输入 ASCII 字符串。
	0x7C02	触摸屏输入中英文混合字符串。
蜂鸣器控制	0x79	蜂鸣器鸣叫一声。
视频切换	0x7A	切换视频和 HMI 功能。
背光控制	0x5E	关闭背光或设置触控（键控）背光模式。
	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；读取当前时钟。
	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、背光控制模式,显示模式。
实用算法	0xB001	基于一级字库的拼音输入法。
	0xB002	计算(A×B+C)/D,E 是 4 字节商，F 是 2 字节余数。
	0xB003	无符号整数（2 字节）数组排序。
	0xB004	基于 GBK 字库的拼音输入查询。
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。
软件升级	DWIN_M600_BOOT!	串口在线升级内核软件。

附录 2 ASM51 应用实例：转动的时钟

使用 DMG80600S104_02WT 设计一个带表针的时钟，利用 0x9E 实现表针旋转，0x7C 实现触摸屏输入修改时间。

```
$include (MOD52)
SYSFLG  EQU     20H
        TIOFLG  BIT     SYSFLG.7        ;传回给主程序的串口发送 1 个字节结束（TI）标记
        RXAAOK  BIT     SYSFLG.6        ;串口接收到 0xAA 帧头
        RXFRMOK BIT     SYSFLG.5        ;串口收到一个数据帧
        RTCOK   BIT     SYSFLG.4        ;250mS 定时器标记
RTC250  EQU     30H             ;100mS 定时器计数次数
RTCOLD  EQU     31H
RXCC    EQU     43H             ;串口接收帧结束符判别数组
RX33    EQU     44H
RXC3    EQU     45H
RX3C    EQU     46H
RXPOS   EQU     47H             ;48H-9FH
RXCMD   EQU     48H             ;串口接收缓冲区 48H-67H 最大 32Byte
RXDAT   EQU     49H
RTCYEAR EQU     4AH
RTCMON  EQU     4BH
RTCDAY  EQU     4CH
RTCWEK  EQU     4DH
RTCHR   EQU     4EH
RTCMIN  EQU     4FH
RTCSEC  EQU     50H

        ORG     0000H
        LJMP    MAIN

        ORG     000BH
        LJMP    SYSCLK          ;10mS 定时器

        ORG     0023H
        LJMP    UARTPRO         ;串口中断

        ORG     0080H
MAIN:   CLR     EA
        MOV     SP,#68H
        ORL     PCON,#80H       ;串口初始化
        MOV     SCON,#50H
```

```
        MOV     TMOD,#21H
        MOV     TH1,#255        ;115200 bps  115200/(256-TH1)
        MOV     TL1,#255
        SETB    TR1
        MOV     TH0,#0B8H       ;10mS
        MOV     TL0,#00H
        SETB    TR0
        MOV     RTC250,#200
        MOV     SYSFLG,#00H
        SETB    ET0
        SETB    ES
        SETB    EA
        JNB     RTCOK,$
        CLR     RTCOK           ;上电延时 2 秒
        MOV     A,#00H
        LCALL   PICDSP
        MOV     DPTR,#RTCON
        LCALL   TXROMS
        MOV     RTCOLD,#00H
;前台主程序部分，通过巡检消息来触发不同功能模块
START:  JNB     RTCOK,START1
        MOV     DPTR,#RDRTC
        LCALL   TXROMS
        CLR     RTCOK
START1: JNB     RXFRMOK,START
        MOV     A,RXCMD                 ;串口收到有效数据帧
        CJNE    A,#9BH,START2
        MOV     A,RXDAT
        CJNE    A,#5AH,STARTE
        LCALL   RTCDSP                  ;显示指针位置
        LJMP    STARTE
START2: CJNE    A,#78H,START3
        MOV     DPTR,#CMD7C             ;修改时间
        LCALL   TXROMS
        LJMP    STARTE
START3: CJNE    A,#7CH,STARTE           ;输入法结束返回
        MOV     A,#00H
        LCALL   PICDSP
        MOV     A,RTCDAY
        CJNE    A,#12,STARTE
        LCALL   WRRTC
STARTE: CLR     RXFRMOK
        LJMP    START

PICDSP: PUSH    ACC
        MOV     A,#0AAH
        LCALL   TXBYTE
        MOV     A,#70H
        LCALL   TXBYTE
        POP     ACC
        LCALL   TXBYTE
        MOV     DPTR,#CMDTTL
        LCALL   TXROMS
        RET
```

```
;把时钟修改后发给 HMI
WRRTC:  MOV     DPTR,#CMDE7
        LCALL   TXROMS
        MOV     R0,#RTCWEK
        MOV     R1,#6
WRRTC1: MOV     A,@R0
        CLR     C
        SUBB    A,#30H
        SWAP    A
        MOV     R7,A
        INC     R0
        MOV     A,@R0
        INC     R0
        CLR     C
        SUBB    A,#30H
        ADD     A,R7
        LCALL   TXBYTE
        DJNZ    R1,WRRTC1
        MOV     DPTR,#CMDTTL
        LCALL   TXROMS
        RET

RTCDSP: MOV     R0,#RTCHR
        MOV     R1,#3                   ;转换成 HEX
RTCDSP1:MOV     A,@R0
        LCALL   BCDHEX
        MOV     @R0,A
        INC     R0
        DJNZ    R1,RTCDSP1
        MOV     R0,#RTCSEC
        MOV     A,@R0
        CJNE    A,RTCOLD,RTCDSP2
        RET
RTCDSP2:MOV     RTCOLD,A
        MOV     DPTR,#CMD71
        LCALL   TXROMS
        ;计算 HOUR 的角度
        MOV     R0,#RTCHR
        MOV     A,@R0
        MOV     B,#60
        MUL     AB
        MOV     R7,B
        MOV     R6,A
        MOV     R0,#RTCMIN
        MOV     A,@R0
        ADD     A,R6
        MOV     R6,A
        CLR     A
        ADDC    A,R7
        MOV     R7,A
        MOV     DPTR,#HRPOS
        LCALL   TXROMS
        MOV     A,R7
        LCALL   TXBYTE
```

```
        MOV     A,R6
        LCALL   TXBYTE
        MOV     DPTR,#RTCCEN
        LCALL   TXROMS
        ;计算 MIN 的角度
        MOV     R0,#RTCMIN
        MOV     A,@R0
        MOV     B,#12
        MUL     AB
        MOV     R7,B
        MOV     R6,A
        MOV     R0,#RTCSEC
        MOV     A,@R0
        MOV     B,#5
        DIV     AB
        ADD     A,R6
        MOV     R6,A
        CLR     A
        ADDC    A,R7
        MOV     R7,A
        MOV     DPTR,#MINPOS
        LCALL   TXROMS
        MOV     A,R7
        LCALL   TXBYTE
        MOV     A,R6
        LCALL   TXBYTE
        MOV     DPTR,#RTCCEN
        LCALL   TXROMS
        ;计算 SEC 的角度
        MOV     R0,#RTCSEC
        MOV     A,@R0
        MOV     B,#12
        MUL     AB
        MOV     R7,B
        MOV     R6,A
        MOV     DPTR,#SECPOS
        LCALL   TXROMS
        MOV     A,R7
        LCALL   TXBYTE
        MOV     A,R6
        LCALL   TXBYTE
        MOV     DPTR,#RTCCEN
        LCALL   TXROMS
        MOV     DPTR,#CMDD0
        LCALL   TXROMS
        RET

;发送 1 个字节到串口
TXBYTE: MOV     SBUF,A
        JNB     TI0FLG,$
        CLR     TI0FLG
        RET
```

```
        ;把 BCD 码转化成 HEX
        BCDHEX: MOV     B,#16
                DIV     AB
                PUSH    B
                MOV     B,#10
                MUL     AB
                POP     B
                ADD     A,B
                RET

        ;发送一个固定的数据串到串口，0xFE 为数据串结尾
        TXROMS: CLR     A
                MOVC    A,@A+DPTR
                CJNE    A,#0FEH,TXROMS1
                RET
        TXROMS1:MOV     SBUF,A
                INC     DPTR
                JNB     TIOFLG,$
                CLR     TIOFLG
                SJMP    TXROMS
                RET
```

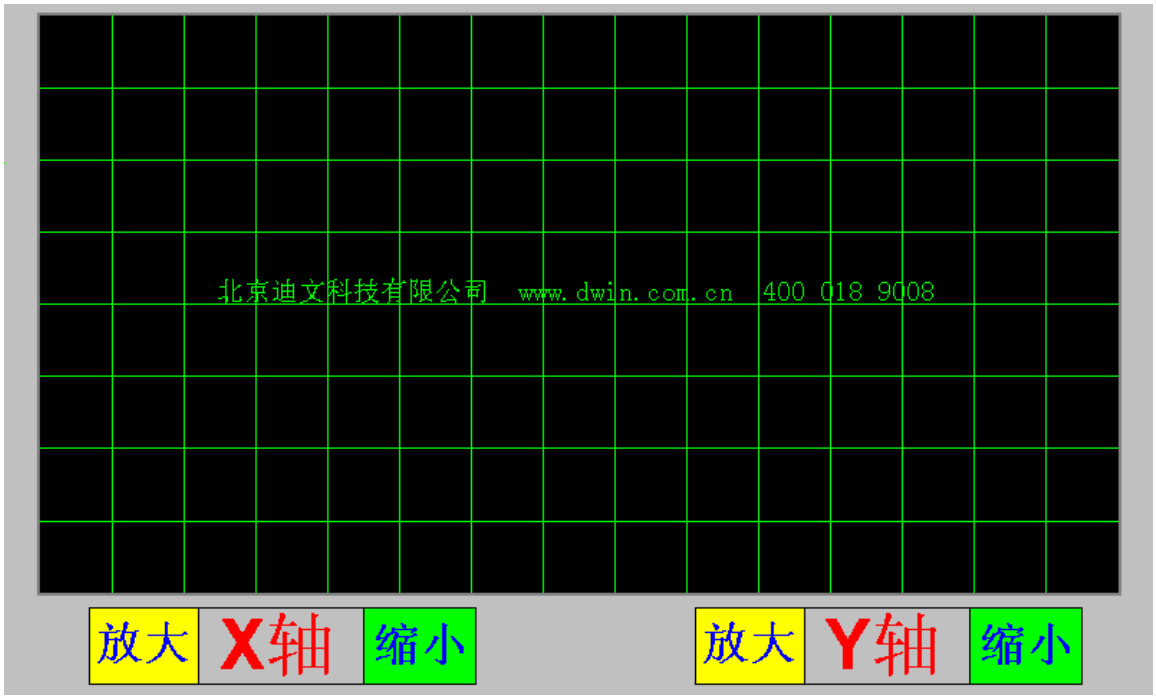
;后台中断服务程序

```
;***************串口中断服务程序*************
        UARTPRO:PUSH    ACC
                PUSH    PSW
                SETB    RS0
                SETB    RS1
                JB      RI,UARTRCV
                CLR     TI
                SETB    TIOFLG
                LJMP    UARTEND
        UARTRCV:MOV     A,SBUF
                CLR     RI
                JB      RXFRMOK,UARTEND         ;串口有数据没有处理
                JB      RXAAOK,URCVD
                CJNE    A,#0AAH,UARTEND
                SETB    RXAAOK
                MOV     RXPOS,#RXCMD
                LJMP    UARTEND
        URCVD:  MOV     R0,RXPOS
                MOV     @R0,A
                INC     RXPOS
                MOV     RXCC,RX33
                MOV     RX33,RXC3
                MOV     RXC3,RX3C
                MOV     RX3C,A
                CJNE    A,#3CH,UARTEND
                MOV     A,RXC3
                CJNE    A,#0C3H,UARTEND
                MOV     A,RX33
                CJNE    A,#33H,UARTEND
                MOV     A,RXCC
                CJNE    A,#0CCH,UARTEND
                SETB    RXFRMOK
```

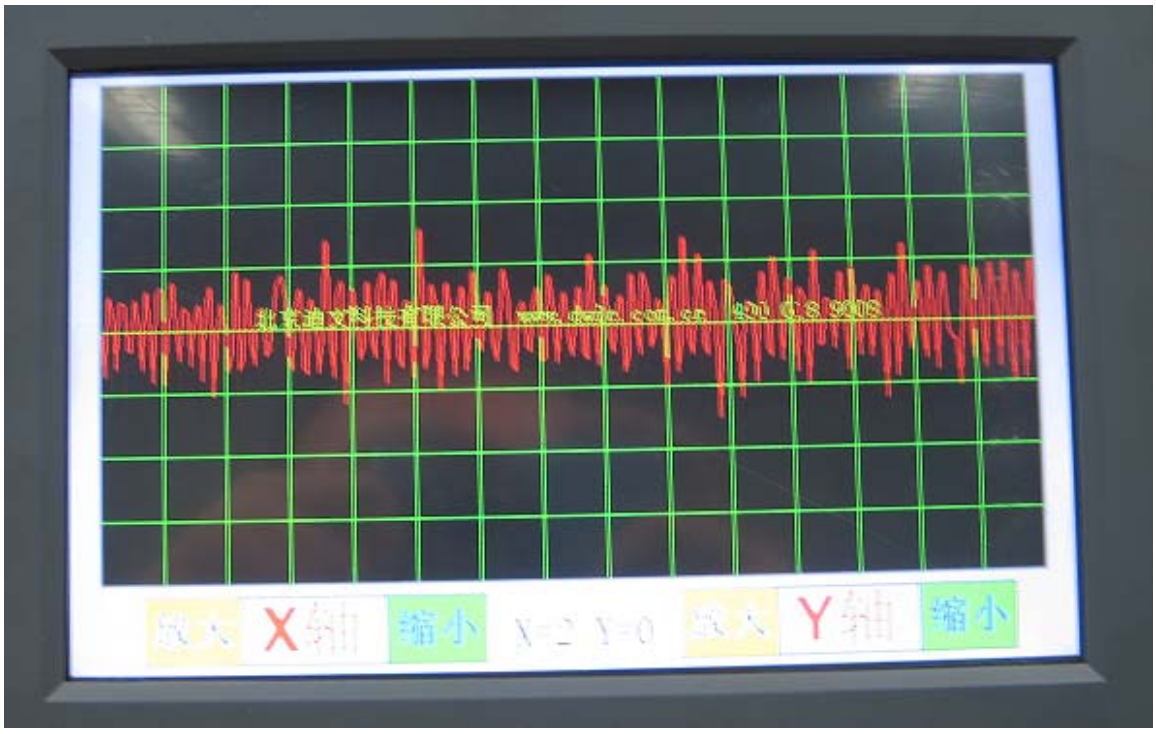
```
                CLR     RXAAOK
        UARTEND:POP     PSW
                POP     ACC
                RETI

;***************10mS 定时器中断服务程序***********
        SYSCLK: PUSH    PSW
                CLR     TF0
                MOV     TH0,#0B8H
                MOV     TL0,#00H
                DJNZ    RTC250,SYSCLKE
                MOV     RTC250,#10              ;100mS 定时器
                SETB    RTCOK
        SYSCLKE:POP     PSW
                RETI

        CMDD0:  DB 0AAH,0D0H,0CCH,33H,0C3H,3CH,0FEH     ;立即刷新显示

        CMD7C:  DB 0AAH,71H,01H                         ;切换第 1 页上的键盘到当前页面
                DW 0,0,206,344,0,0
                DB 0CCH,33H,0C3H,3CH
                DB 0AAH,07CH,01H,00H,01H                ;使用英文输入 12 个 ASCII 字符来修改时钟
                DW 1,34,40,0C03H,0000H
                DB 0CCH,33H,0C3H,3CH,0FEH

        CMDE7:  DB 0AAH,0E7H,55H,0AAH,05AH,0A5H,0FEH    ;修改 HMI 时钟
                RTCON:  DB 0AAH,9BH,0FFH,01H,05H,0FFH,0FFH      ;显示 HMI 内置的时钟
                DW 200,567,0CC33H,0C33CH,0FEFEH

        RDRTC:  DB 0AAH,9BH,5AH,0CCH,33H,0C3H,3CH,0FEH  ;从 HMI 回读时间

        CMD71:  DB 0AAH,71H,00H
                DW 207,96,593,520,207,96,0CC33H,0C33CH,0FEFEH   ;恢复表盘背景

        HRPOS:  DB 0AAH,9EH,00H,00H,01H                 ;时分秒的图标位置
                DW 449,281,494,432,471,382,0FEFEH
        MINPOS: DB 0AAH,9EH,00H,00H,01H
                DW 383,248,427,434,406,382,0FEFEH
        SECPOS: DB 0AAH,9EH,00H,00H,01H
                DW 335,232,374,434,356,382,0FEFEH
        RTCCEN: DW 386,301,0CC33H,0C33CH,0FEFEH         ;表盘指针的旋转中心
        CMDTTL: DB 0CCH,33H,0C3H,3CH,0FEH               ;帧结束符

                END
```

附录 3 ASM51 应用实例：最简单的示波器

基于 DMG80480C070-01WT，利用 C8051F410 自带的 12bit A/D 采集驻极体 Mic 输出的语音信号并显示在屏幕上。使用 0xC106 指令，可以触摸屏实时调整 X 轴（时间）和 Y 轴（幅度）的缩放比例。



设计的显示界面



实际运行结果（X=6.67mS/DIV Y=50mV/DIV）

ASM51 代码如下：

```
$include (C8051F410.inc)
WDT     BIT     P0.7                    ;看门狗喂狗信号
AD_IN   BIT     P1.4                    ;A/D
SYSFLG  EQU     20H
        TIOFLG  BIT     SYSFLG.7        ;传回给主程序的串口发送 1 个字节结束（TI）标记
        RXAAOK  BIT     SYSFLG.6        ;串口接收到 0xAA 帧头
        RXFRMOK BIT     SYSFLG.5        ;串口收到一个数据帧
        RTCOK   BIT     SYSFLG.4        ;定时器标记
        NUMOK   BIT     SYSFLG.3        ;HMI 缓冲区数据传送 OK
RTC250  EQU     31H             ;定时器计数次数
ADS_H   EQU     32H             ;A/D 数据保存指针 0000-03FFF
ADS_L   EQU     33H
ADT_H   EQU     34H             ;A/D 数据发送指针 0000-03FFF
ADT_L   EQU     35H
ADN_H   EQU     36H             ;已经采集未发送的点数
ADN_L   EQU     37H
KY      EQU     38H             ;Y 轴放大倍数
KX      EQU     39H             ;X 轴放大倍数
CO_DPH  EQU     3AH
CO_DPL  EQU     3BH             ;HMI 缓冲区指针位置
RXCC    EQU     70H             ;串口接收帧结束符判别数组
RX33    EQU     71H
RXC3    EQU     72H
RX3C    EQU     73H
RXPOS   EQU     74H             ;78H-9FH
RXCMD   EQU     78H             ;串口接收缓冲区 78H-9FH 最大 40Byte
RXDAT0  EQU     79H
RXDAT1  EQU     7AH

        ORG     0000H
        LJMP    MAIN

        ORG     0023H
        LJMP    UARTPRO         ;串口中断

        ORG     002BH
        LJMP    SYSCLK          ;0.4mS 定时器，用来进行 A/D 采样

        ORG     0100H
MAIN:   CLR     EA
        MOV     SP,#0B0H
        LCALL   INITCPU         ;配置串口为 115200bps，8n1 模式
        MOV     SYSFLG,#00H
        SETB    ES0             ;开串口中断
        SETB    ET2             ;开定时器 2 中断（10mS 系统定时器）
        MOV     ADS_H,#00H
        MOV     ADS_L,#00H
        SETB    EA
        LCALL   ACKLCD          ;检查 HMI 上电是否 OK
        MOV     A,#00H
        LCALL   PICDSP
        CLR     EA
        MOV     ADS_H,#00H
        MOV     ADS_L,#00H
        MOV     ADT_H,#00H
        MOV     ADT_L,#00H
        MOV     ADN_H,#00H
        MOV     ADN_L,#00H
        MOV     CO_DPH,#00H
        MOV     CO_DPL,#00H
        MOV     KX,#3
        MOV     KY,#1
        SETB    EA
        LCALL   XYDSP           ;显示 X、Y 放大倍数
```

```
START:  CPL     WDT
        JNB     RXFRMOK,START1
        MOV     A,RXCMD          ;串口收到有效数据帧
        CJNE    A,#78H,START0E   ;触控指令
        MOV     A,RXDAT1
        CJNE    A,#00H,START01   ;X 放大
        MOV     A,KX
        CLR     C
        SUBB    A,#9
        JNC     START0E
        INC     KX
        LJMP    START0E
START01:CJNE    A,#01H,START02   ;X 缩小
        MOV     A,KX
        JZ      START0E
        DEC     KX
        LJMP    START0E
START02:CJNE    A,#02H,START03   ;Y 放大
        MOV     A,KY
        CLR     C
        SUBB    A,#9
        JNC     START0E
        INC     KY
        LJMP    START0E
START03:CJNE    A,#03H,START0E   ;Y 缩小
        MOV     A,KY
        JZ      START0E
        DEC     KY
START0E:CLR     RXFRMOK
        LCALL   XYDSP
START1: MOV     DPTR,#60         ;缓冲区数据超过 120 字节再发送，提高串口效率
        CLR     C
        MOV     A,ADN_L
        SUBB    A,DPL
        MOV     A,ADN_H
        SUBB    A,DPH
        JC      START
        CLR     EA
        MOV     A,ADN_L
        CLR     C
        SUBB    A,DPL
        MOV     ADN_L,A
        MOV     A,ADN_H
        SUBB    A,DPH
        MOV     ADN_H,A
        SETB    EA
        MOV     A,KX             ;根据 X 轴放大，计算出需要显示的点数 R7:R6
        CLR     C
        RLC     A
        PUSH    ACC
        MOV     DPTR,#NUM_TAB
        MOVC    A,@A+DPTR
        MOV     R7,A
        POP     ACC
        INC     A
        MOVC    A,@A+DPTR
        MOV     R6,A
        MOV     A,#0AAH
        LCALL   TXBYTE
        MOV     A,#0C0H
        LCALL   TXBYTE
        MOV     A,C0_DPH
        LCALL   TXBYTE
        MOV     A,C0_DPL
        LCALL   TXBYTE
```

```
        MOV     DPH,ADT_H
        MOV     DPL,ADT_L
        MOV     R1,#60
START11:MOVX    A,@DPTR
        INC     DPTR
        LCALL   TXBYTE
        MOVX    A,@DPTR
        INC     DPTR
        LCALL   TXBYTE
        ANL     DPH,#03H
        MOV     ADT_H,DPH
        MOV     ADT_L,DPL
        MOV     A,C0_DPL
        ADD     A,#01H
        MOV     C0_DPL,A
        CLR     A
        ADDC    A,C0_DPH
        MOV     C0_DPH,A        ;C0_DPTR++
        CLR     C
        MOV     A,R6
        SUBB    A,C0_DPL
        MOV     A,R7
        SUBB    A,C0_DPH
        JNC     START12
        SETB    NUMOK
        MOV     C0_DPH,#00H
        MOV     C0_DPL,#00H
        LJMP    START13
START12:DJNZ    R1,START11
        CLR     NUMOK
START13:MOV     DPTR,#CMDTTL
        LCALL   TXROMS
        JNB     NUMOK,START2
        MOV     DPTR,#CMDC106A  ;把曲线显示出来
        LCALL   TXROMS
        MOV     A,R7
        LCALL   TXBYTE          ;显示数目
        MOV     A,R6
        LCALL   TXBYTE
        MOV     DPTR,#CMDC106B
        LCALL   TXROMS
        MOV     A,KX
        INC     A
        LCALL   TXBYTE
        MOV     A,KY            ;放大比例从 1-10
        INC     A
        LCALL   TXBYTE
        MOV     DPTR,#CMDC106C
        LCALL   TXROMS
        MOV     DPTR,#CMDD0
        LCALL   TXROMS
START2: NOP
        LJMP    START

;发送 0x00 握手指令，确认 HMI 上电 OK
ACKLCD: CLR     RXFRMOK
ACKLCD1:MOV     DPTR,#CMD00
        LCALL   TXROMS
        CLR     RTCOK
        MOV     RTC250,#100
        JNB     RTCOK,$
        CLR     RTCOK
        JNB     RXFRMOK,ACKLCD1
        CLR     RXFRMOK
        MOV     A,RXCMD
```

```
        CJNE    A,#00H,ACKLCD1
        RET

;显示出放大比例
XYDSP:  MOV     DPTR,#CMD401
        LCALL   TXROMS
        MOV     DPTR,#CMD551
        LCALL   TXROMS
        MOV     A,KX
        ADD     A,#30H
        LCALL   TXBYTE
        MOV     DPTR,#CMD552
        LCALL   TXROMS
        MOV     A,KY
        ADD     A,#30H
        LCALL   TXBYTE
        MOV     DPTR,#CMDTTL
        LCALL   TXROMS
        MOV     DPTR,#CMDD0
        LCALL   TXROMS
        RET

;发送数据个数，x 轴 750 点，N=750/（kx＋1）
NUM_TAB:DW 750,375,250,187,150,125,107,93,83,75
CMD401: DW 0AA40H,0000H,001FH,0CC33H,0C33CH
        DW 0AA42H,0,0,0CC33H,0C33CH,0FEFEH
CMD551: DW 0AA55H,349,434
        DB 'X=',0FEH
CMD552: DB ' Y=',0FEH
CMDD0:  DB 0AAH,0D0H,0CCH,33H,0C3H,3CH,0FEH
CMDC106A:DW 0AA71H,0,24,7,776,409,24,7,0CC33H,0C33CH
        DB 0AAH,0C1H,06H,00H,00H
        DW 25,208,0FEFEH
CMDC106B:DB 01H,0FEH
CMDC106C:DB 1
        DW 0F800H,8,408,0CC33H,0C33CH,0FEFEH
CMD00:  DB 0AAH,00H,0CCH,033H,0C3H,3CH,0FEH
CMDTTL: DB 0CCH,33H,0C3H,3CH,0FEH

;发送 1 个字节到串口
TXBYTE: MOV     SBUF0,A
        JNB     TI0FLG,$
        CLR     TI0FLG
        RET

;发送 1 个字节到串口
TXWORD: MOV     SBUF0,DPH
        JNB     TI0FLG,$
        CLR     TI0FLG
        MOV     SBUF0,DPL
        JNB     TI0FLG,$
        CLR     TI0FLG
        RET

;发送一个固定的数据串到串口，0xFE 为数据串结尾
TXROMS: CLR     A
        MOVC    A,@A+DPTR
        CJNE    A,#0FEH,TXROMS1
        RET
TXROMS1:MOV     SBUF0,A
        INC     DPTR
        JNB     TI0FLG,$
        CLR     TI0FLG
        CPL     WDT
        SJMP    TXROMS
```

```
                RET

        PICDSP: PUSH    ACC
                MOV     A,#0AAH
                LCALL   TXBYTE
                MOV     A,#70H
                LCALL   TXBYTE
                POP     ACC
                LCALL   TXBYTE
                MOV     DPTR,#CMDTTL
                LCALL   TXROMS
                RET

;***************串口中断服务程序*************
        UARTPRO:PUSH    ACC
                PUSH    PSW
                SETB    RS0
                SETB    RS1
                JB      RI0,UARTRCV
                CLR     TI0
                SETB    TI0FLG
                LJMP    UARTEND
        UARTRCV:MOV     A,SBUF0
                CLR     RI0
                JB      RXFRMOK,UARTEND         ;串口有数据没有处理
                JB      RXAAOK,URCVD
                CJNE    A,#0AAH,UARTEND
                SETB    RXAAOK
                MOV     RXPOS,#RXCMD
                LJMP    UARTEND
        URCVD:  MOV     R0,RXPOS
                MOV     @R0,A
                INC     RXPOS
                MOV     RXCC,RX33
                MOV     RX33,RXC3
                MOV     RXC3,RX3C
                MOV     RX3C,A
                CJNE    A,#3CH,UARTEND
                MOV     A,RXC3
                CJNE    A,#0C3H,UARTEND
                MOV     A,RX33
                CJNE    A,#33H,UARTEND
                MOV     A,RXCC
                CJNE    A,#0CCH,UARTEND
                SETB    RXFRMOK
                CLR     RXAAOK
        UARTEND:POP     PSW
                POP     ACC
                RETI

;***************0.4mS 定时器中断服务程序***********
        SYSCLK: PUSH    PSW
                SETB    RS1
                CLR     RS0
                PUSH    DPH
                PUSH    DPL
                MOV     TMR2CN,#04H
                MOV     R7,ADC0H
                MOV     R6,ADC0L
                MOV     ADC0CN,#90H
                MOV     DPTR,#511
                MOV     A,ADN_L
                SUBB    A,DPL
                MOV     A,ADN_H
                SUBB    A,DPH
```

```
                JNC     SYSCK2
                CLR     C
                MOV     A,R6
                SUBB    A,#00H                  ;去掉直流分量
                MOV     R4,A
                MOV     A,R7
                SUBB    A,#08H
                MOV     R5,A
                JNC     SYSCK1
                MOV     A,#00H
                CLR     C
                SUBB    A,R6
                MOV     R4,A
                MOV     A,#08H
                SUBB    A,R7
                SETB    ACC.7
                MOV     R5,A
        SYSCK1: MOV     DPH,ADS_H
                MOV     DPL,ADS_L
                MOV     A,R5
                MOVX    @DPTR,A
                INC     DPTR
                MOV     A,R4
                MOVX    @DPTR,A
                INC     DPTR
                ANL     DPH,#03H
                MOV     ADS_H,DPH
                MOV     ADS_L,DPL
                MOV     A,ADN_L
                ADD     A,#01H
                MOV     ADN_L,A
                CLR     A
                ADDC    A,ADN_H
                MOV     ADN_H,A
        SYSCK2: DJNZ    RTC250,SYSCLKE
                MOV     RTC250,#250     ;100mS 定时器
                SETB    RTCOK
        SYSCLKE:POP     DPL
                POP     DPH
                POP     PSW
                RETI

;***************CPU 初始化***********
        INITCPU:MOV     VDMOCN,#0C0H
             MOV     RSTSRC,#02H
             MOV     PCAOCN,#00H
             MOV     PCAOMD,#00H
             MOV   P0MDIN,#0FFH
             MOV   P0MDOUT,#90H     ;0001 0000B P0.4 IS TXD
             MOV   P0SKIP,#00H

             MOV   P1MDIN,#0EBH     ;1110 1011
             MOV   P1MDOUT,#00H
             MOV   P1SKIP,#014H     ;0000 0100B SKIP VERF-M     A/D

                MOV     XBR0,#01H
                MOV     XBR1,#40H

             MOV   OSCXCN,#00H
             MOV   OSCICN,#0C7H
             CLR   A
             DJNZ  ACC,$
                CPL     WDT
             MOV   CLKMUL,#00H
             NOP
```

```
        ORL    CLKMUL,#03H
        NOP
        ORL    CLKMUL,#10H
        NOP
        ORL    CLKMUL,#80H
        NOP
        CLR    A
        DJNZ   ACC,$
        NOP
        ORL    CLKMUL,#0C0H
        NOP
OSCWT2:     MOV    A,CLKMUL
        JNB    ACC.5,OSCWT2
        NOP
        MOV    CLKSEL,#02H      ;49MHz SYSTEM CLOCK
            MOV       A,#07H            ;串口波特率设置，7=115200
            MOV       B,#2
            MUL       AB
            MOV       DPTR,#BODE_TAB
            MOV       B,A
            MOVC      A,@A+DPTR
            MOV       CKCON,A
            MOV       A,B
            INC       DPTR
            MOVC      A,@A+DPTR
            MOV       TH1,A
            MOV       TL1,A
            MOV       SCON0,#50H
            SETB      TR1
            MOV       TMR2RLH,#0F9H     ;2mS=0DF73 1mS=EFB9  0.4mS=F97D
            MOV       TMR2H,#0F9H
            MOV       TMR2RLL,#07DH
            MOV       TMR2L,#07DH
            MOV       TMR2CN,#04H
            MOV       IE,#00H
            MOV       EIE1,#00H
        MOV       REF0CN,#33H
        MOV    ADC0MX,#0CH      ;A/D=P1.4
        MOV    ADC0CF,#28H      ;AD 转换时钟为 系统时钟 10 分频
        MOV     ADC0CN,#90H       ;AD0BUSY 控制 AD 转换
        MOV    ADC0TK,#0CH     ;AD 仅仅转换一次
            RET

;波特率寄存器表 1.2 2.4 4.8 9.6 19.2 38.4 57.6 115.2
;0=2041667 1=6125000 2=510417
BODE_TAB:DB 2,43          ;2400   00
         DB 2,43          ;2400   01
         DB 2,150         ;4800   02
         DB 0,43          ;9600   03
         DB 0,150         ;19200  04
         DB 1,96          ;38400  05
         DB 1,150         ;57600  06
         DB 1,203         ;115200 07

         END
```

附录 4 C51 串口通信程序参考

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

bit RXAAOK; //串口接收到 0xAA 帧头
bit RXFRMOK;//串口收到一个有效数据帧
bit TIOFLAG;//传回给主程序的串口发送 1 个字节结束（TI）标记

uchar RXBUF[32];  //串口接收缓冲区，存储时把 0xAA 帧头去掉,HMI 应答不是固定格式的都有长度返回，故
                  //不需要记录长度
uchar Rx_P;  //串口接收，保存数据位置

//串口中断服务程序
void UART0_ISR() interrupt 4
{ uchar i;
   if(RI)      //串口接收中断
{    i=SBUF;
     RI=0;
     if(RXFRMOK==0)  //如果 RXFRMOK=1 表示前台程序未处理完当前数据，丢弃
 {    if(RXAAOK)       //接收到 0xAA
 {    RXBUF[Rx_P]=i;
if((Rx_P>3)&&(RXBUF[Rx_P-3]==0xCC)&&(RXBUF[Rx_P-2]==0x33)&&(RXBUF[Rx_P-1]==0xC3)&&(RXBUF[RX_P]==0x3C))
    {RXFRMOK=1;
     RXAAOK=0;}
     Rx_P++;}
     if(!RXAAOK&&(i==0xaa))
 {    RXAAOK=1;
     Rx_P=0;}}
}
  if(TI)
{  TI=0;
   TIOFLAG=1;}
  //对于 DSP、ARM 等带 FRAME_ERROR 中断的 CPU，此处应添加对 FSTA 寄存器的清零
}


    if(RXFRMOK)                    //前台主程序检查串口有没有收到 HMI 数据
{    if(RXBUF[0]==0x78)    //触控按钮
     {}
     if(RXBUF[0]==0x7C)    //触摸屏拼音输入结束
     {}
     RXFRMOK=0;             //处理完毕，清除消息
 }
```

附录 5 文档涉及的 C51 函数原型

```
void T2_ISR(void) interrupt 5   //使用 10mS 定时器 2 中断来产生系统定时和演示
{   TF2H=0;
    RTCOK=1;
    if(delay10ms)  delay10ms--;}     //delay10ms,RTCOK 都是全局变量

void delay10ms(uchar i)  //延时 i*10mS
{  delay10ms=i;
   while(delay10ms);}

void acklcd()          //上电发送 0x00 握手指令检查 LCD 是否已经就绪，否则等待 LCD 就绪
{  RXFRMOK=0;
   while(RXFRMOK==0)
{  txword(0xaa00);
   txeof();
   delay10ms(10);}
   RXFRMOK=0;}

void txbyte(uchar i)    //查询方式发送一个字节到串口
 { SBUF=i;
   whilie(TIOFLG==0);
   TIOFLG=0;}

void txword(uint i)    //发送一个字到串口
{  txbyte(i/256);
   txbyte(i%256);}

void txlong(ulong i)  //发送 2 个字到串口
{  txword(i/65536);
   txword(i%65536);}

void txxy(uint x,uint y) //发送坐标到串口
{  txword(x);
   txword(y);}

void txarea(uint xs,uint ys,uint xe,unit ye)  //发送一个矩形区域坐标到串口
{  txword(xs);
   txword(ys);
   txword(xe);
   txword(ye);}

void txstr(uchar *s)  //发送字符串到串口，字符串以 0x00 结尾
{ while(*s)
{ txbyte(*s);
  s++;}}

void txroms(uchar *s)  //发送字符串到串口，字符串以 0xFE 结尾
{ whilie(!(*s==0xFE))
{ txbyte(*s);
  s++;}}

void txeof()         //发送帧结束符到串口
{  txxy(0xcc33,0xc33c);}
```

```
void fdisp()        //强制刷新一次显示
{  txword(0xaad0);
   txeof();}

void setcolor(uint color,uint bcolor)   //设置调色板
{  txword(0xaa40);
   txxy(color,bcolor);
   txeof();}

void line(uint xs,uint ys,uint xe,uint ye)  //连线
{  txword(0xaa56);
   txarea(xs,ys,xe,ye);
   txeof();}

void picdsp(uint p)    //显示图片
{  txword(0xaa70);
   txword(p);
   txeof();}

void piccut(uint cmd,uint p,uint xs,uint ys,uint xe,uintye,unit x,uint y)  //图标剪切显示
{  txbyte(0xaa);
   txbyte(cmd);
   txword(p);
   txarea(xs,ys,xe,ye);
   txxy(x,y);
   txeof();}

void prints(uchar cmd,uint x,uint y,uchar *s)   //非 0x98 指令显示字符串，字符串以 0x00 结尾
{  txbyte(0xaa);
   txbyte(cmd);
   txxy(x,y);
   txstr(s);}    //如果指针字符串未包含帧结束符，需要额外发送

//0x98 指令显示字符串，字符串以 0xfe 结尾
void print98(uchar lib,uchar mode,uchar dot,uint color,uint bcolor,uint x,uint y uchar *s)
{  txword(0xaa98);
   txxy(x,y);
   txbyte(lib);
   txbyte(mode);
   txbyte(dot);
   txxy(color,bcolor);
   txroms(s);}   //如果指针字符串未包含帧结束符，需要额外发送

void fillw(uchar cmd,uint xs,uint ys,uint xe,uint ye) //区域指令 0x59、0x69、0x5A、0x5B、0x5C
{  txbyte(0xaa);
   txbyte(cmd);
   txarea(xs,ys,xe,ye);
   txeof()}

void pixel(uint x,uint y)   //置点
{  txword(0xaa51);
   txxy(x,y);
   txeof();}
```

```
//触摸屏输入中英文（中文为 GBK 字库）
void scans(uint rid,uint pid,uint x,uint y,uint x1,uint yi1,uchar len,uint color0,uint color1)
{txword(0xaa7c);
 txbyte(0x02);  //01=ASCII  02=中英文混合输入
 txword(rid);   //输入结束返回的索引 ID
 txword(pid);   //触控配置索引 ID
 txxy(x,y);     //录入文本的显示区域起始坐标
 txbyte(len);   //最大录入长度
 txbyte(0x00);
 txxy(color0,color1);  //color0=录入文本的显示颜色  color1=录入过程显示文本颜色
 txxy(x1,y1);     //录入过程显示文本的起始坐标
 txeof();}
//录入过程迪文 HMI 自动完成，无须用户软件干涉
//录入结束，HMI 返回录入字符串结果
if((rxcmd==0x7C)&&(rxdat[2]==0x01))   //0x01 是用户事先指定的返回 ID，对应上面代码的 rid
{wrxram(rxdat[3],0x0800);}  //保存参数，radat[3]是录入参数长度
```

北京迪文科技有限公司是全球领先的从事工业 HMI（智能串口显示终端）技术研发、产品生产和营销的高新技术企业。

公司现有员工约 150 人，总部位于被称为中国"硅谷"的北京中关村核心区，在深圳、上海、北欧、美国设有驻外办事处，在印度、土耳其、巴西等地建立了销售渠道，并在北京上地信息产业基地建立了包括高洁净度装配线、质量提升中心、实验室和老化车间，总面积约 5000㎡的生产基地。

迪文自 2003 年成立至今，秉承"专业素养、诚实守信、追求卓越"的经营理念，强调为客户创造价值，实现共赢，使公司获得了快速发展和壮大。依托产品在功能、品质、服务、价格上的领先和竞争优势，迪文 HMI 在工业用 TFT 显示屏市场，尤其在注重可靠性的工程机械、医疗仪器、出口机电产品上占据重要地位，赢得了诸如三一 SANY、通用 、西门子 SIEMENS 等众多客户的信任和支持。

迪文始终坚持为客户创造价值，专注科技创新，力求与人双赢、共同成长。

北京迪文科技有限公司

地址：北京市海淀区知春路 108 号豪景大厦 A 座 9 层　　邮编：100086
电话：400 018 9008

未开通 400 电话地区，请拨打以下电话：

- 产品购买　　010-62104079 62101462 62104686
- 技术支持　　010-62555372 62102321 62102090
- 投　　诉　　010-62101180
- 公司联络　　010-62102630 62105007 62636805

传真：010-62628562
网站：www.dwin.com.cn
邮箱：dwinhmi@263.net