

48-51

## 开发经验

### 利用 CCED5.0 直接打印高级语言生成数据文件

辽宁省铁岭市烟草专卖局 杜蕴杰

TP311

前段时间内各报刊中曾介绍过利用 WPS 排版系统打印高级语言产生的报表数据文件的实现技巧。使用过 CCED5.0 字表排版系统的人都知道, CCED5.0 中增加了很多非常好的功能, 如: 方便的斜线命令可产生任意表格斜线, 图象嵌入命令可使各种流行格式的图象文件直接镶嵌到文本文件的任意位置; 灰度命令可以控制在表格栏内填充多种灰度; 负行距命令可使横向表格线不占位置; 表内文字可以任意缩放而不影响表格线的完整性等等。如果在开发的应用程序中直接使用 CCED5.0 排版系统的打印控制命令 ASC II 码, 直接生成 CCED5.0 可直接打印的表报文本, 您就会直接获得更加精美的输出结果。

虽然 CCED5.0 为方便用户使用, 其打印控制符与 WPS 系统完全兼容, 但其扩充和增加了很多新的功能, 因此要利用它来直接打印高级语言程序产生数据文件, 即要用高级编程语言设计出生成 CCED5.0 可直接打印的输出文件, 必须对其控制符的命令码重新进行分析。为方便同行使用, 本人将自己总结出来的 CCED5.0 所有打印控制符命令全部给出:

#### 1. 字体控制符的控制码值

CCED5.0 除了具有 WPS 的 8 种字体 (宋体、仿宋体、楷体、黑体、标宋、隶书、行楷、魏碑) 外, 还增加了字体 A (细圆) 到字体 Z18 种常用字体, 共计 25 种常用字体的控制符。其控制码值的高字节均为 91H, 低字节按帮助菜单出现顺序依次为 80H~90H。

#### 2. 字型控制符的控制码值

CCED5.0 和 WPS 均有 6 种基本字型: 标准型、长型、扁型、自定义型、特大型和统一型。前 4 种字型的控制符控制码值完全相同, 高字节均为 92H, 低字节与字型的对应关系如表 1 所示。

表 1

字 体	标准型	长 型	扁 型	自定义
0-7	80-8EH 偶	90-9EH 偶	A0-1EH 偶	B0-B7
小 0-7	81-8FH 奇	90-9FH 奇	A1-AFH 奇	B0-B7

统一型控制符的控制码值与点 (N) 的取值范围有关, N 值的取值范围为 1~38, 划分为三段, 其对应关系如表 2 所示。

表 2

N 取值	高字节	低字节计算公式	H 的取值 1-8 对应
1-16	9C	$80H + 8 * (N - 1) + H - 1$	1/3, 1/2, 2/3
17-32	9D	$80H + 8 * (N - 17) + H - 1$	3/4, 4/3, 3/2
33-38	9E	$80H + 8 * (N - 33) + H - 1$	2/1, 3/1

特大型控制符的控制码值高字节为 93H, 低字节与 N 的取值范围有关, 其计算公式为:  $80H + (N - 2)$ , 其中 N 的取值范围为 2-76。

#### 3. 汉字修饰控制符的控制码值

CCED5.0 的汉字修饰功能包括: 空心开始、空心结束、加框开始、加框结束、虚体开始、虚体结束、上标开始、上标结束、下标开始、下标结束、左转 90°、旋转 180°、取消转角、左斜开始、右斜开始、斜体结束、上齐、下齐、本行居中、本行右齐、笔画加重和取消加重。其中笔画加重和取消加重控制符的控制码值高字节为 88H, 低字节为 FEH-FFH; 其余控制符的控制码值高字节为 94H, 低字节依次为 80H-90H 和 9BH-9EH。

#### 4. 划线修饰控制符的控制码值

CCED5.0 的划线类型控制符与 WPS 完全相同, 其控制码高字节为 94H, 低字节均为 91H-9AH。

#### 5. 背景、阴影、前景控制符的控制码值

CCED5.0 的背景、阴影、前景修饰类型控制符和 WPS 也完全相同, 控制符的控制码值的高字节为 95H, 低字节分别为 80-87H (背景)、88H-8FH (阴影) 和 90H-97H (前景)。

#### 6. 英文字体控制符的控制码值

CCED5.0 的英文字体控制符与 WPS 完全相同, 共 11 种控制字型, 包括前 10 种比例字体和最后一种非比例字体。其控制符的高字节为 96H, 低字节分别为 91H-9AH。

#### 7. 打印格式控制符的控制码值

CCED5.0 的字符后退 N 个半角字控制符的控制码值高字节为 97H, 低字节由 N (范围 0-127) 来计算, 公式为:  $80H + N$ ;

字符升高 N 个点的控制符的控制码值高字节为 98H, 低字节由 N 值 (取值范围 -63-64) 来计算, 公式

为  $80H + N + 63$ ;

设定字符 X 点字间距的控制符的控制码值高字节为  $99H$ 、低字节由  $N$  (取值范围  $-63 \sim 64$ ) 值来计算, 公式为  $80H + N + 63$ ;

设定  $N$  点行间距的控制符的控制码值, 高字节和低字节均由  $N$  值来确定:

当  $N$  为  $0 \sim 127$  时: 高字节为  $9BH$ , 低字节为  $80H + N$ ;

当  $N$  为  $-63 \sim -1$  时: 高字节为  $88H$ , 低字节为  $80H - (N + 1)$ 。

### 8. 斜线和灰度控制符的控制码值

CCED5.0 的斜线控制符的控制码值由所定义的矩形块来决定, 当高度值  $Y$  大于零时, 其控制符的 ASC 码值为 "89 DE 89 DC 89 A8 89 B1 + X 89

AC 89 B1 + Y 89 A9", 当  $Y$  值小于零时, 其控制符的控制码值为 "89 DE 89 DC 89 A8 89 B1 + X 89 AC 89 AD 89 B1 + Y 89 A9";

CCED5.0 的灰度控制符的控制码值也是由所定义的矩形块来决定, 当高度值  $Y$  大于零时, 其控制符的 ASC 码值为 "89 DE 89 AA 89 A8 89 B1 + X 89

AC 89 B1 + Y 89 AC 89 B0 + M 89 AD 89 AA 89 A8 89 B1 + X 89 AC 89

AD 89 B1 + Y 89 AC 89 B0 + M 89 A9";

其中  $X$  为定义表格块的宽度值,  $Y$  为定义表格块的高度值,  $M$  为灰度类型的取值: 1 ( $1/4$ ), 2 ( $1/8$ ), 3 ( $1/16$ ), 4 ( $1/32$ ), 5 ( $1/64$ ), 6 ( $1/16$ ) 左斜线, 7 ( $1/16$ ) 右斜线。

### 9. 专用码打印控制符的应用

由于只使用驱动程序的兼容码处理方式不能包容全部打印控制功能, 如不能直接使用打印机的控制命令等, CCED5.0 中提供了 WPS 排版系统所不具有的专用码打印输出方式。CCED5.0 的 B 方式专门设置了控制码, 它用一串半角字符的特别组合来定义, 第一个控制字符为半角字符的 ":", 叫专用码, 专用码打印控制符主要包括:

① :~ 强行分页控制符;

② :~ 防打控制符, 即该控制符后面部分不打印;

③ :~ 页眉说明符, 必须在文件前五行内;

④ : \ (N1, N2) 斜线控制符,

$N1$  为矩形块高度, 总为正值

$N2$  为矩形块的宽度  $N2$  为正表示从左向右斜

$N2$  为负表示从右向左斜

⑤ : \* (N1, N2, N3) 灰度控制符,  $N1$  和  $N2$  同上, 并且  $N2$  小于 0 时自动带边界,  $N3$  为灰度级控制, 取值范围为  $1 \sim 5$ , 数值越大灰度越浅;

⑥ : ~... ~ 原码发送控制符, 直接将中间的代码送打印机, 并串中可含 "&" 函数控制符;

由上述控制命令中可看出, CCED5.0 中的斜线和灰度控制功能也可用打印方式中的专用码控制符来完成。专用码打印控制符应用示例: 如直接使用 2 倍放大, 硬字库字体的命令为:

: & (28) (33) (12) ~

该命令是将横纵向均放大 2 倍的硬字体命令 "28 33 12" 直接发送给打印机。

### 10. 嵌入图象控制符的使用技巧

CCED5.0 中还可使用 WPS 所不具备的图象嵌入功能。使用 CCED5.0 可打印 PCX, PUT, TIF, IMG 等各种未经压缩的图象, 其控制命令为:

: & (FILENAME, N1, N2, N3, N4, N5, N6, N7)

其中: "&" 是控制符;

FILENAME 为图象文件名, 可带路径;

$N1 \sim N3$  为控制打印图象块;

$N1$  为图象块相对图象起始行的点偏移量, 默认值为 0;

$N2$  为图象块相对图象起始列的字节偏移量, 默认值为 0;

$N3$  为图象块的点行高度, 缺省值 0;

$N4$  为图象块的字节宽度, 缺省值 0,  $N4 < 0$  可镜像打印图象;

$N5$  为定义图象文件模式, 默认值为 0,

= 0 原图象为黑白双色

= 1 原图象为黑白双色并取反相

= 2 原图象为黑白双色并四倍放大

= 4 (低四位) 原图象为 16 色

= 8 (低四位) 原图象为 256 色

= 4 或 = 8 时高四位 = 0 表示取图象的全部色素, > 0 表示取部分色素;

$N6$  为图象文件记录的宽度字节数,  $N6 = 0$  默认值, 表示 CCED 能够自动的宽度字节数,

= 1 表示 PUT 格式

= 2 表示 TIF 格式

= 3 表示 IMG 格式

= 4 表示 SPI 的图象文件

$N7$  为文件头字节数, 当  $N6$  赋值后  $N7$  由 CCED 自动给出, 否则  $N7 = 0$ 。

具体示例:

^&(ZZZ.PCX, 0, 0, 0, -226) 镜像嵌入图象  
ZZ.PCX

^&(ZZZ.IMG, 0, 2, 0, 1, 1, 3) 反相嵌入图  
象 ZZZ.IMG

^&(ZZZ.PUT, 0, 0, 0, 0, 4, 1) 全色嵌入图  
象 ZZZ.PUT。

通过上述分析结果,就可以在高级语言程序中直接利用 CCED5.0 打印控制码,生成 CCED5.0 可直接打印的数据文件,从而得到非常整洁、精美的输出结果。其接口程序及调用实例如下:

```
#include <stdio.h>
#include <ctype.h>
#include <string.h> /* 定义控制代码集 */
typedef enum {CHS, CHF, CHK, CHH, CHB, CHL, CHX, CHW,
    CHA, CHC, CHD, CHE, CHG, CHI, CHJ, CHN, CHM,
    CHO, CHP, CHQ, CHR, CHT, CHU, CHV, CHY, CHZ
    } CHINA_CODE; /* 定义汉字字体控制代码 */
typedef enum {BZX, SCX, BXX, ZDX} ZX_CODE; /* 定义字型控制
    代码 */
typedef enum {ZT0, ZT1, ZT2, ZT3, ZT4, ZT5, ZT6, ZT7,
    ZTX0, ZTX1, ZTX2, ZTX3, ZTX4, ZTX5, ZTX6, ZTX7
    } ZT_CODE; /* 定义字体控制代码 */
typedef enum {KX1, KX0, JK1, JK0, XT1, XT0, XB1, SB0, XB1, XB0,
    ZY1, YZ1, XZ1, XZ0, ZX1, YX1, XXE, SQ1, SQ0, HZ1,
    HY1, JZ1, JZ0} XS_CODE; /* 定义修饰代码 */
typedef enum {SH1, SH0, XH1, XH2, XH3, XH4, XH5, XH6, XH7,
    XH0
    } HX_CODE; /* 定义划线类型控制代码 */
typedef enum {WDB, WGB, ZXB, YXB, JCB, SCB, FSB, QXB,
    YY1, YY2, YY3, YY4, YY5, YY6, YY7, YY0,
    WDQ, HXQ, SXQ, WGQ, ZXQ, YXQ, JCQ, QXQ
    } BYQ_CODE; /* 定义背景阴影前景控制代码
    */
typedef enum {YW1, YW2, YW3, YW4, YW5, YW6, YW7, YW8,
    YW9,
    YWA, YWO} CHAR_CODE; /* 定义英文字体控
    制代码 */
typedef enum {HTN, SCN, ZJJ, HJJ} MOVE_CODE; /* 设置字符位
    移控制代码 */

/* 1. 设置汉字字体功能函数 */
char * china_style(CHINA_CODE code)
/* code 为字型控制代码 */
static char save_code[3];
save_code[0] = 0x91; /* 字体控制代码高字节 */
save_code[2] = 0x00;
if (code <= CHZ) save_code[1] = 0x80 + code;
```

```
else save_code[0] = 0x00;
return save_code;
}

/* 2. 设置汉字字型功能函数 */
char * ZX_style(ZX_CODE zx, ZT_CODE zt)
/* zx 为字型控制代码,包括标准型、竖长型、扁型和自定义
    型;zt 为字体控制代码 */
static char save_code[3];
save_code[0] = 0x92; /* 四种字型控制代码高字节 */
save_code[2] = 0x00;
switch(zx) {
    /* 设置字型控制代码 */
    case BZX: save_code[1] = 0x80; break; /* 标准型起始低字节
        */
    case SCX: save_code[1] = 0x90; break; /* 长型起始低字节
        */
    case BXX: save_code[1] = 0xa0; break; /* 扁型起始低字节
        */
    case ZDX: save_code[1] = 0xb0; break; /* 自定义型起始低字
        节 */
    default: save_code[0] = 0x00; break;
}
if (zx == ZDX) /* 设置字型中字号控制代码 */
    if (zt < ZTX0) save_code[1] += zt; /* 自定义型低字节
        80H- B7h */
    else save_code[0] = 0x00;
else if (zx < ZDX) /* 其它三种低字节 80H- 8FH 90H-
    9FH A0H- A7H */
    if (zt < ZTX0) save_code[1] += zt * 2;
    else if (zt <= ZT * 7) save_code[1] += (zt - 8) * 2 + 1;
    else save_code[0] = 0;
return save_code;

/* 3. 设置统一型汉字字型功能函数 */
char * zx_tya_style(unsigned n, unsigned h)
/* n 为统一型字体控制中的点数值
    h 为统一型字体控制中的字体高宽比值 */
static char save_code[3];
save_code[2] = 0x00;
if (n < 39) save_code[0] = 0x9c + (n - 1) * 16;
else save_code[0] = 0x00;
if (0 < h < 91) save_code[1] = 0x80 + 8 + (n - 1) *
    (16 * 16 + 1) + h - 1;
else save_code[0] = 0x00;
return save_code;

/* 4. 设置特大型汉字字型功能函数 */
char * zx_tdx_style(unsigned int n)
/* n 为特大型字体控制中的点数值 */
```

```
static char save_code[3];
save_code[0] = 0x93; /* 特大型控制代码高字节 */
save_code[2] = 0x00;
if (1 < n < 77) save_code[1] = 0x80 + n - 2;
else save_code[0] = 0x00;
return save_code;
}
```

#### \* 5. 设置汉字修饰类型功能函数 \*

```
char * xs_style(XS_CODE xs)
```

/\* xs 为汉字修饰类型 \*/

```
static char save_code[3];
save_code[0] = 0x94; /* 汉字修饰控制代码高字节 */
save_code[2] = 0x00;
if (xs < JZ1) if (xs < SQ1) save_code[1] = 0x80 + xs;
else save_code[1] = 0x80 + xs + 10;
else if (xs < JZ0)
    save_code[0] = 0x88; /* 高字节值 */
    save_code[1] = 0x00 + xs - 21; /* 低字节值 */
    else save_code[0] = 0x00;
return save_code;
```

#### \* 6. 设置划线类型功能函数 \*

```
char * hx_style(HX_CODE hx)
```

/\* hx 为划线类型数据 \*/

```
static char save_code[3];
save_code[0] = 0x94; /* 划线类型高字节代码 */
save_code[2] = 0x00;
if (hx < HX0) save_code[1] = 0x91 + hx;
else save_code[0] = 0x00;
return save_code;
```

#### \* 7. 设置背景 - 阴影 - 前景功能函数 \*

```
char * hvq_style(BYQ_CODE hvq)
```

/\* hvq 为背景 - 阴影 - 前景控制代码 \*/

```
static char save_code[3];
save_code[0] = 0x95; /* 背景 - 阴影 - 前景高字节代码 */
save_code[2] = 0x00;
if (hvq < QXQ) save_code[1] = 0x80 + hvq;
else save_code[0] = 0x00;
return save_code;
```

#### \* 8. 设置英文字体功能函数 \*

```
char * char_style(CHAR_CODE xa)
```

/\* YW 为设置英文字体控制代码 \*/

```
static char save_code[3];
save_code[0] = 0x96;
```

```
save_code[2] = 0x00;
if (yw < YW0) save_code[1] = 0x80 + yw;
else save_code[0] = 0x00;
return save_code;
}
```

因篇幅有限,设置字符后退、升高、字间距、行间距功能函数;设置斜线打印功能函数;设置灰度打印功能函数等略去。若有需要者,请与编辑部联系

## 51—53 解决以代码为字段名的多字段 录入的汉文提示问题

黄浩然 (郑州拖拉机厂) 白琳 (郑州大学)

TP311

### 1. 具体问题提出

笔者在建立当年统计数据库时,要以 150 个经济指标为字段名,各项指标名称多达十几个汉字,而在 dBASE III PLUS 环境下,数据库中字段名不能超过 10 个字符即最多五个汉字,且最多设置 128 个字段。为了有利于问题分析和编程处理,我们对各项指标分类编码,代码首位字符表示指标分类:A 产量、B 销量、C 库存、D 产值、E 财务、F 劳资、H 消耗。根据指标间相关程度和有无计算关系,建立两个数据库,A、B、C 三类数据存于 DNZL.DBF,其它 D、E、F、G、H 类数据存于 DN-ZL2.DBF。各库每一个记录为一个月的数据,共 12 个记录。在编制录入子程序时,用户提出凡是机器可算出的累计项(代码尾标第四位为“B”)不用录入,其它指标均应在各代码和汉字名称同时指示的情况下,进行录入的要求,因为代码太多,很容易因弄不清中文含义而出错。

### 2. 解决办法

根据以上要求,笔者用数据库结构伸展命令建立了指标目录库 DNZBML3.DBF,再增加指标名称“MI”字段,录入全部代码所对应的指标中文名称。录入程序如下:

```
/* dnzbl.prg 录入子程序
set talk off
clear
XX = 1
n = month(DATE())
@4,20 say "请您输入要录入哪月份的数据:" get n put "99"
range 1,12
read
do while .
```