卡片机 OCX 编程手册

（DC3300 卡片机 SDK 项目）

草拟	审核	批准

文件变更记录

（A-增加　M-修订　D-删除）

版本号	日期	变更类型（A,M,D）	修改人	摘要	审核人	ocx 版本
0.0.0.1	2018-06-12	新订				1.0.0.26

目录

概述 5
一、打印机连接相关命令 5
1.枚举打印机 5
2. 从打印机列表中得到打印机 5
3. 打开打印机 6
4. 关闭打印机 6
5. 打印机打开状态 6
6. 设置打印机类型 7
二、 打印相关命令 7
1. 打印图像 7
2. 设置 IO 操作超时时间 8
3. 好卡口出卡 8
4. 坏卡口出卡 8
三、 套打接口 9
1. 设置卡片尺寸 9
2. 开始绘图 9
3. 结束绘图 9
4. 绘制文本 10
5. 设置字体及字号大小 10
6. 设置字体的颜色 10
7. 设置字体风格 11
8. 设置文本格式：平躺字体、反向输出、自动换行 11
9. 设置字间距、行间距 11
10. 绘制图块 12
11. 判断是否正在绘图 12
四、 三合一读卡器相关操作接口 12
1. 加载读卡器 DLL 12
2. 枚举 com 口 13
3. 快速打开读卡器 13
4. 关闭读卡器 13
5. 进卡到读卡器 13
6. 退卡到打印机 13
7. 移动卡片 14
8. 寻卡 14
9. 认证 14
10. 读取数据 15
11. 写入数据 15
12. 终止卡片操作 15
13. 复位 CPU 卡 16
14. 复位 PSAM 卡 16
15. CPU 卡/PSAM 卡 APDU 指令交互 17

附录 19
附录 1（错误码） 19
常见使用流程（不考虑失败情况） 21
1. 枚举、打开和关闭打印机 21
2. 套打流程 22
3. 3 合 1 读卡器读写 PSAM 卡/用户 CPU 卡 23

概述

1. 该手册为卡片机浏览器应用开发者使用。提供了得实卡片机 DC3300 的打印、设置、套打、维护、状态等功能；
2. 实例片段采用 C++、MFC 编写；在 vs2010 上 编译通过；
3. 支持 USB 口、网口打印；
4. 目前仅支持 DC3300 型号的打印机；
5. 函数如果有多个信息返回，将使用键值对 string 字符串的方式返回，在网页客户端可转换成 Json 对象以方便处理。

一、打印机连接相关命令

1.枚举打印机

函数原型：

string DS600EnumPrinter(long printerType)

参数说明：

printerType：（传入）指定要枚举何种打印机，1：USB 打印机；2：网络打印机；3：USB 打印机和网络打印机都枚举

返回值：

1. 如果执行成功返回如下形式的字串：

{“**RtnCode**”:0,

”**NumberOfPrinters**”:1,

”**PrinterList**”:”DS 7600@USB001”}

其中：

RtnCode ：表示本次调用的错误码，因为是调用成功，所以值 为 0；

NumberOfPrinters：表示枚举到的打印机数目，这里为 1 台；

PrinterList：打印机列表，表示枚举到的所有打印机；若有多台打印机，将以”\r\n”分割。

2. 如果执行失败返回如下形式的字符串：

{“RtnCode”: 0x28000019} 表示函数执行失败，错误码为 0x28000019。具体参见附录 1 错误码;

2. 从打印机列表中得到打印机

函数原型：

string DS600GetPrinterListItem (string printerList, long index)

参数说明：

printerList：（传入）DS600EnumPrinter 函数返回的打印机列表；

index：（传入）指定要得到打印机信息中第几台的信息，第一台为 0，最后一台为 DS600EnumPrinter 返回 的 NumberOfPrinters 减 1。

返回值 ：

如果 index 在 0 到 NumberOfPrinters-1 范围内，则返回对应的打印机名称；否则函数返回空串。

3. 打开打印机

函数原型：

long DS600OpenPrinter (string Curprinter)

参数说明：

Curprinter：（传入）要打开的打印机，由 DS600GetPrinterListItem 返回。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码;

4. 关闭打印机

函数原型：

void DS600ClosePrinter();

备注：关闭当前打印机。

5. 打印机打开状态

函数原型：

bool DS600IsPrinterOpened();

返回值：

为 true 时表示当前打印机被打开。

6. 设置打印机类型

函数原型：

Void DSSetPrinterType(LPCTSTR model);

参数说明：

Model:当前打印机的型号。如 DC3300 则传入 DC3300 即可。此设定会影响打开读卡器时是否自动排卡。

二、 打印相关命令

1. 打印图像

函数原型：

```
LONG D300CPrintImage(
        string FrontSideImagePath,
        string backSideImagePath,
        LONG F_rotation,
        LONG B_rotation,
        LONG F_scalingmode,
        LONG B_scalingmode,
        LONG F_colormode,
        LONG B_colormode,
        BOOL printToImg );
```

参数说明：

FrontSideImgPath：（传入）指定正面打印图片的路径。

backSideSideImgPath：（传入）指定背面打印图片的路径。

F_rotation：（传入）正面旋转之后打印，取值范围及含义如下：

0：不旋转；

1：旋转 90 度（逆时针）

2：旋转 180 度（逆时针）

3：旋转 270 度（逆时针）

B_rotation：（传入）反面旋转之后打印，取值范围及含义同上。

F_ScalingMode：（传入）正面图片拉伸模式，取值范围及含义如下：

0：拉伸铺满；

1：居中铺满；

2：原始比例铺满；

B_ScalingMode：（传入）反面图片拉伸模式，取值范围及含义同上。

F_colorMode：（传入）正面图片打印颜色模式，取值范围及含义如下：

0：使用 YMC 混色打印；

1：纯黑色使用 K 色打印，其余颜色使用 YMC 混色；

2：仅用 K 色打印，半色调方式为阈值；

3：仅用 K 色打印，半色调方式为误差扩散；

4：仅用 K 色打印，半色调方式为有序抖动；

printToImg：（传入）指定是否仅生成图像，而不发送给打印机，可作为调试使用。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

2. 设置 IO 操作超时时间

函数原型：

```
long    DS600SetIOTimeout(
        long    readTimeout,
        long    writeTimeout);
```

参数说明：

readTimeout：（传入）设置读操作超时值，单位毫秒。

writeTimeout：（传入）设置写操作超时值，单位毫秒。

返回值

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

备注：

默认的读写超时值均为 3000 毫秒。

3. 好卡口出卡

函数原型：

```
DWORD  D300CToGood(HANDLE H)
```

参数说明：

H:（传入）读卡器句柄，该句柄由 D300COpenReader() 获取。

返回值

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

4. 坏卡口出卡

函数原型：

```
DWORD  D300CToBad(HANDLE H)
```

参数说明：

H:（传入）读卡器句柄，该句柄由 D300COpenReader() 获取。

返回值

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

三、 套打接口

1. 设置卡片尺寸

函数原型：

long D300CDrawSetCardRect(DOUBLE w, DOUBLE h);

参数说明：

W：（传入）卡片宽度，单位 mm。

H：（传入）卡片高度，单位 mm。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

备注：

请在 DS600BeginDraw 前调用

2. 开始绘图

函数原型：

long DS600BeginDraw (void);

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

备注：

请在绘制所有元素（文本、一维码、二维码）之前，调用该接口。

3. 结束绘图

函数原型：

string DS600EndDraw (void);

返回值：

1. 如果执行成功返回，则返回如下形式的字串：

{"**RtnCode**":0,

"**imgPath**":”c:\\R600-1234.bmp”

}

其中：

RtnCode ：表示本次调用的错误码，因为是调用成功，所以值 为 0；

imgPath：表示生成图片的路径。

2. 如果执行失败返回如下形式字串，表示函数执行失败，错误码为 0xyyyyyyyy。具体参见附录 1 错误码。

{“RtnCode”:0xyyyyyyyy}

4. 绘制文本

函数原型：

long　DS600DrawText(double ox,double oy,double w,double h,const char* data);

参数说明：

Ox：（传入）指定用于绘制文本的区域左上角 X 坐标，单位：mm。

Oy：（传入）指定用于绘制文本的区域左上角 Y 坐标，单位：mm。

W：（传入）指定用于绘制文本的区域宽度，单位：mm。

H：（传入）指定用于绘制文本的区域高度，单位：mm。

Data：（传入）指定要绘制的文本。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

备注：

请在使用本接口前，调用相关设置字体样式的接口

5. 设置字体及字号大小

函数原型：

```
long　DS600DrawSetFont(
        const char* fontName,
                    LONG size);
```

参数说明：

fontName：（传入）指定要绘制的字体 。

size：（传入）指定字体大小。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

备注：

请在使用 DS600DrawText 接口前调用

6. 设置字体的颜色

函数原型：

```
long　DS600DrawSetTextColor(unsigned char R,
                            unsigned char G,
                            unsigned char B);
```

参数说明：

R：（传入）指定字体颜色的红色分量值，取值范围 0~255。

G：（传入）指定字体颜色的绿色分量值，取值范围 0~255。

B：（传入）指定字体颜色的蓝色分量值，取值范围 0~255。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

备注：

请在使用 DS600DrawText 接口前调用

7. 设置字体风格

函数原型：

long DS600DrawSetTextDecorate(
bool isStrong,
bool isItalic,
bool isUnderline)

参数说明：

isStrong：（传入）指定是否以粗体绘制。

isItalic：（传入）指定是否以斜体绘制。

isUnderline：（传入）指定绘制的字体是否加下划线。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

备注：

请在使用 DS600DrawText 接口前调用

8. 设置文本格式：平躺字体、反向输出、自动换行

函数原型：

long DS600DrawSetTextDirection(bool isLandScape,
bool isReverseSequence,
bool isAutoLineFeed,
bool isLayDown);

参数说明：

isLandScape：（传入）是否以平躺方式绘制文本。

isReverseSequence：（传入）是否反向绘制文本。

isAutoLineFeed:（传入）是否在文本块内自动换行，文本块大小由 DS600DrawText 定义。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

备注：

请在使用 DS600DrawText 接口前调用

9. 设置字间距、行间距

函数原型：

long DS600DrawSetSpacing(DOUBLE lineSpacing,
DOUBLE charSpacing);

参数说明：

lineSpacing：（传入）指定文本块的行间距，单位 mm。

charSpacing：（传入）指定文本块的字间距，单位 mm。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

备注：

请在使用 DS600DrawText 接口前调用

10. 绘制图块

函数原型：

```
long    DS600DrawImage(DOUBLE    ox,
                       DOUBLE    oy,
                       DOUBLE    w,
                       DOUBLE    h,
                       string    imagefile,
                       bool isPhoto);
```

参数说明：

Ox：（传入）指定用于绘制二维码的区域左上角 X 坐标，单位 mm。

Oy：（传入）指定用于绘制二维码的区域左上角 Y 坐标，单位 mm。

w：（传入）指定用于绘制二维码的区域宽度，单位 mm。

h：（传入）指定用于绘制二维码的区域高度，单位 mm。

imagefile：（传入）指定图片路径。

IsPhoto：（传入）是否白色视为透明

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码

11. 判断是否正在绘图

函数原型：

```
Bool    DS600DrawInSession (void);
```

返回值 ：

如果正在绘中，则返回 true，否则返回 false.

四、 三合一读卡器相关操作接口

1. 加载读卡器 DLL

函数原型：

```
long    DS600LoadReaderLib (void)
```

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

2. 枚举 com 口

函数原型：

BSTR DSEnumSerialScanner(void)

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

3. 快速打开读卡器

函数原型：

ULONG DS600OpenReaderFast(string com)

参数说明：

com：（传入）枚举到的 com 口。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码.

4. 关闭读卡器

函数原型：

LONG DS600CloseReader (HANDLE icdev)

参数说明：

Icdev：（传入）读卡器句柄

返回值：

如果执行成功返回值为读卡器句柄，否则返回 0

5. 进卡到读卡器

函数原型：

DWORD D300CToCardReader(HANDLE H);

参数说明：

H:（传入）读卡器句柄

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

6. 退卡到打印机

函数原型：

DWORD D300CBackToPrinter(HANDLE H)

参数说明：

H：（传入）设备标识符。

pos 传入：pos模式。

* @n 0x00 - 移动到磁条卡操作位置。

* @n 0x01 - 移动到接触卡操作位置。

* @n 0x02 - 移动到非接触卡操作位置。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

7. 移动卡片

函数原型：

DWORD D300CCardMove(HANDLE H,int pos)

参数说明：

H：（传入）设备标识符。

pos 传入：pos 模式。

* @n 0x00 - 移动到磁条卡操作位置。

* @n 0x01 - 移动到接触卡操作位置。

* @n 0x02 - 移动到非接触卡操作位置。

返回值：

成功返回 0，否则返回非 0，具体参见附录 1 错误码。

备注：

移动卡片，操作前设备内无卡则错误。

8. 寻卡

函数原型：

String DS600_dc_card_hex (HANDLE icdev)

参数说明：

1. 如果执行成功，返回如下形式的字串：

{"**RtnCode**":0,

"**snrstr**":”01234567”}

其中：

RtnCode ：表示本次调用的错误码，因为是调用成功，所以值 为 0；

snrstr：寻到的卡片序列号

2. 如果执行失败返回如下形式字串，表示函数执行失败，错误码为 0xyyyyyyyy。具体参见附录 1 错误码。

{“RtnCode”:0xyyyyyyyy}

9. 认证

函数原型：

LONG　DS600_dc_authentication_pass(LONG icdev, LONG _Mode, LONG _Addr, string pwd)

参数说明：

Icdev：（传入）读卡器句柄

_ Mode：（传入）0 表示验证 A 密码，4 表示验证 B 密码

Pwd：（传入）6 个字符的密码

返回值：

如果执行成功返回值为读卡器句柄，否则返回 0

10. 读取数据

函数原型：

BSTR　DS600_dc_read_hex(LONG icdev, LONG _Adr)

参数说明：

Icdev：（传入）读卡器句柄

_Adr：（传入）要读取的块，取值 0~63

返回值：

1. 如果执行成功，返回如下形式的字串：

   {"**RtnCode**":0,

   "**data**":”01020304050607080 90A0B0C0D0E0F00”}

   其中：

   RtnCode ：表示本次调用的错误码，因为是调用成功，所以值 为 0；

   data：读取到的 HEX 数据

2. 如果执行失败返回如下形式字串，表示函数执行失败，错误码为 0xyyyyyyyy。具体参见附录 1 错误码。

{“RtnCode”:0xyyyyyyyy}

11. 写入数据

函数原型：

LONG　DS600_dc_write_hex(LONG icdev, LONG _Adr, string data)

参数说明：

Icdev：（传入）读卡器句柄

_Adr：（传入）要写入的块，取值 0~63

Data：（传入）要写入的 HEX 数据

返回值：

如果执行成功返回值为读卡器句柄，否则返回 0

12. 终止卡片操作

函数原型：

LONG　DS600_dc_halt(LONG icdev)

参数说明：

Icdev：（传入）读卡器句柄

返回值：

如果执行成功返回值为读卡器句柄，否则返回 0

备注：

调用此接口使卡片进入终止状态，必须将卡片一移出感应区在移入感应区才能寻到卡。

13. 复位 CPU 卡

函数原型：

String D300CResetUserCard(LONG H, ULONG SlotNum);

参数说明：

H：（传入）读卡器句柄

SlotNum：（传入）卡槽编号：

0x00 – 非接触式 CPU 卡。

0x0C - 附卡座/接触式 CPU1 卡座。

0x0B - 接触式 CPU2 卡座。

返回值：

1. 如果执行成功，返回如下形式的字串：

{"**RtnCode**":0,

"**ResetInfo**":" 1078809002209000000000000b35b8e44",

"uid": "010200405"

}

其中：

RtnCode ：表示本次调用的错误码，因为是调用成功，所以值 为 0；

ResetInfo：复位信息

Uid: 卡序列号

2. 如果执行失败返回如下形式字串，表示函数执行失败，错误码为 0xyyyyyyyy。具体参见附录 1 错误码。

{"RtnCode":0xyyyyyyyy}

14. 复位 PSAM 卡

函数原型：

String DS600ResetPsam (LONG H, ULONG SlotNum);

参数说明：

H：（传入）读卡器句柄

SlotNum：（传入）卡槽编号：

0x0D - SAM1 卡座。

0x0E - SAM2 卡座。

0x0F - SAM3 卡座。

0x11 - SAM4 卡座。

0x12 - SAM5 卡座。
0x13 - SAM6 卡座/ESAM 芯片。
0x14 - SAM7 卡座。
0x15 - SAM8 卡座。

（如果以上编号都无法复位成功，则卡槽编号放宽为从 0x00——0xff）

返回值：

1. 如果执行成功，返回如下形式的字串：

{"**RtnCode**":0,
"**ResetInfo**":" 10788090022090000000000b35b8e44"
}

其中：

RtnCode ：表示本次调用的错误码，因为是调用成功，所以值 为 0；

ResetInfo：复位信息

2. 如果执行失败返回如下形式字串，表示函数执行失败，错误码为 0xyyyyyyyy。具体参见附录 1 错误码。

{"RtnCode":0xyyyyyyyy}

备注：

调用此接口使卡片进入终止状态，必须将卡片一移出感应区在移入感应区才能寻到卡。

15. CPU 卡/PSAM 卡 APDU 指令交互

函数原型：

String DS600ExchangeAPDU (LONG H, LONG SlotNum, string capdu);

参数说明：

H：（传入）读卡器句柄

SlotNum：（传入）卡槽编号：

0x00 – 非接触式 CPU 卡。
0x0C - 附卡座/接触式 CPU1 卡座。
0x0B - 接触式 CPU2 卡座。
0x0D - SAM1 卡座。
0x0E - SAM2 卡座。
0x0F - SAM3 卡座。
0x11 - SAM4 卡座。
0x12 - SAM5 卡座。
0x13 - SAM6 卡座/ESAM 芯片。
0x14 - SAM7 卡座。
0x15 - SAM8 卡座。

（如果以上编号都无法复位成功，则卡槽编号放宽为从 0x00——0xff）

Capdu：（传入）发送的 APDU 指令

返回值：

1. 如果执行成功，返回如下形式的字串：

{"**RtnCode**":0,
" **rapdu** ":" 10788090022090000000000b35b8e44"

}

其中：

RtnCode ：表示本次调用的错误码，因为是调用成功，所以值 为 0；

rapdu：接收到的 APDU 数据

2. 如果执行失败返回如下形式字串，表示函数执行失败，错误码为 0xyyyyyyyy。具体参见附录 1 错误码。

{“RtnCode”:0xyyyyyyyy}

附录

附录 1（错误码）

错误码分为 4 类，成功（0x0）、参数型错误（0x1 打头的错误）、CLIENT 错误（0x2 打头的错误，比如调用顺序、内存分配、网络通讯、绘图）、SERVICE 错误（0x3 打头）等。

成功	0x00000000
参数无效	0x10000000
接收缓冲区太小	0x10000001
参数为空	0x10000002
参数格式错误	0x10000003
参数长度错误	0x10000004
系统不支持	0x10000005
参数超出范围	0x10000006
接收缓冲区太小	0x10000007
参数太长	0x10000008
打印机配置空	0x10000009
缓冲区太大	0x1000000a
调用顺序错	0x1000000B
没有找到相应的文件	0x1000000c
客户端一般错误	0x20000000
内存分配失败	0x28000000
内存分配失败	0x28000001
内存分配失败	0x28000002
内存分配失败	0x28000003
内存分配失败	0x28000004
TLS 超出索引范围	0x28000005
报文校验错误	0x28000006
作业为空	0x28000007
创建 SOCKET 失败	0x28000008
SOCKET 连接失败	0x28000009
SOCKET 发送数据失败	0x28000010
SOCKET 接收数据失败	0x28000011
创建 USB 文件失败	0x28000012
USB 写文件失败	0x28000013
创建 SOCKET 失败	0x28000015
SOCKET 连接失败	0x28000016
SOCKET 发送数据失败	0x28000017
SOCKET 接收数据失败	0x28000018

打印机名称为空	0x28000019
打印的卡片为空	0x2800001a
厂商请求失败	0x2800001b
打印机未准备好	0x2800001c
不支持的请求	0x2800001d
打印机非空	0x2800001e
UDP 请求失败	0x2800001F
UDP 请求超时	0x28000020
HID 设备未找到	0x28000021
打印机未响应	0x28000022
套打类错误	0x27000000
图形初始化失败	0x27000001
不能在保护区绘制	0x27000002
创建 QR 码失败	0x27000003
卡片尺寸不支持	0x27000004
卡片类型不支持	0x27000005
卡片尺寸已经设置	0x27000006
超过可打印区域	0x27000007
读卡器出错	0x29000000
读卡器打开失败	0x29000001
读卡器读写失败	0x29000002
读卡器库函数失败	0x29000003
读卡器未打开	0x29000004
读卡器内部有卡	0x29000005
读卡器内部没卡	0x29000006
导入读卡器接口出错	0x29000007
读卡器库未加载	0x29000008
读卡器关闭失败	0x29000009
打印机内部有卡	0x2900000a
读卡器和打印机内部都有卡	0x2900000b
SERVICE 一般错误	0x30000000
未知错误	0x30000000
SERVICE 未生效	0x30000001
无效的格式	0x30000002
没有找到指定的打印机	0x30000003
没有找到指定的作业	0x30000004
指定的作业已经终止	0x30000005
指定的作业已完成	0x30000006
指定的作业正在打印	0x30000007
无效的打印机名	0x30000008

指定的作业无效	0x30000009
无效的型号	0x3000000a
无效的连接串	0x3000000b
名字重复	0x3000000c
连接串重复	0x3000000d
正在打印最后一张卡片	0x3000000e
作业数量为零	0x3000000f
打印机离线	0x30000010
指定的打印作业正在排队	0x30000011
指定的作业已暂停	0x30000012

常见使用流程（不考虑失败情况）

1. 枚举、打开和关闭打印机

```
var pterList = ocx.DS600EnumPrinter(3);          //枚举 USB、网口打印机
var obj = eval('(' +    pterList    + ')');      //由传出的字串转为 JSON 对象
if(obj.RtnCode == 0)
{
      output("枚举成功");
      var pterArr = obj.PrinterList.split("\r\n");    //得到打印机列表
      var pterNum = obj.NumberOfPrinters;             //得到枚举的打印机数量
}
else
{
      Output(“枚举失败”);
}
```

2. 套打流程

```
ocx.D300CDrawSetCardRect(85.6,53.66);              //设置卡片大小
ocx.DS600BeginDraw();                                    //开始绘图

ocx.DS600DrawSetFont(“黑体”,14);                    //设置字体为黑体，大小 14 号
ocx.DS600DrawSetTextDecorate(true,true,false);          //设置 粗体、斜体、无下划线
ocx.DS600DrawSetSpacing(1,1);              //设置文本行间距为 1mm，字间距为 1mm
ocx.DS600DrawSetTextColor(255,0,0);                     //设置字体颜色为红色
ocx.DS600DrawText(5,5,25,10,”示例文本”);    //在原点（5，5），大小（25，10）的范围内绘制文本
ocx.DS600DrawImgae(40,5,20,20,”a.jpg”,true);                  //绘制图块

Var tempImg = ocx.D300CEndDraw(tempImg);                  //结束绘图，得到临时图片
var obj = eval('(' +    tempImg    + ')');                 //由传出的字串转为 JSON 对象

ocx.D300CPrintImage(obj. imgPath,NULL,0,0,0,0,0,0,false);          //正面彩色打印
```

3. 3 合 1 读卡器读写 PSAM 卡/用户 CPU 卡

1. 读写用户非接触式 CPU 卡

```
HANDLE H = NULL;                                    //读卡器句柄
DWORD RE = DS600LoadReaderLib();                    //加载 dcrf32.dll
RE = DS600OpenReader(&H);                           //打开读卡器
RE = DS600ToCardReader(H);                          //进卡到读卡器

//移卡到非接触式位置， 参数 2 传 1 时为接触位，传 2 时为非接触式位。
RE = DS600CardMove(H,2);
char resetInfo[256]={0};
int resetInfolen = 256;
char uid[256] = {0};
int uidlen = 256;
//复位用户 CPU 卡，第二个参数为 0 代表是非接触式 CPU 卡
RE = DS600ResetUserCard(H, 0, resetInfo, &resetInfolen, uid, &uidlen );

byte sendBuf[] = {0x00,0x84,0x00,0x00,0x04};
char recvBuf[1024]={0};
int rapdulen = 0;
//与客户非接卡进行apdu交互，第二个参数取与DS600ResetUserCard第二个参数相同值
RE = DS600ExchangeAPDU(H,0,sendBuf,writen,recvBuf,&rapdulen);

RE = DS600BackToPrinter(H);                         //退卡到打印机打印位

RE = DS600CloseReader(H);                           //关闭读卡器
```

2. 读写用户接触式 CPU 卡

```
……
//移卡到接触式位置， 参数 2 传 1 时为接触位，传 2 时为非接触式位。
RE = DS600CardMove(H,1);
char resetInfo[256]={0};
int resetInfolen = 256;
char uid[256] = {0};
int uidlen = 256;

//复位用户 CPU 卡，第二个参数为 0x0C 代表是附卡座/接触式 CPU1 卡座。
RE = DS600ResetUserCard(H, 0x0C, resetInfo, &resetInfolen, uid, &uidlen );

byte sendBuf[] = {0x00,0x84,0x00,0x00,0x04};
char recvBuf[1024]={0};
int rapdulen = 0;
```

```
//与客户非接卡进行apdu交互，第二个参数取与DS600ResetUserCard第二个参数相同值
RE = DS600ExchangeAPDU(H, 0x0C,sendBuf,writen,recvBuf,&rapdulen);
.....
```

3. 读写 PSAM 卡

```
.....
char resetInfo[256]={0};
int resetInfolen = 256;
char uid[256] = {0};
int uidlen = 256;

//复位用户 CPU 卡，第二个参数为 0x0D 代表是 SAM1 卡座，其它卡座请参见 DS600ResetUserCard
//的具体说明
RE = DS600ResetUserCard(H, 0x0D, resetInfo, &resetInfolen, uid, &uidlen );

byte sendBuf[] = {0x00,0x84,0x00,0x00,0x04};
char recvBuf[1024]={0};
int rapdulen = 0;
//与客户非接卡进行apdu交互，第二个参数取与DS600ResetUserCard第二个参数相同值
RE = DS600ExchangeAPDU(H, 0x0D,sendBuf,writen,recvBuf,&rapdulen);
.....
```

4. 非接触式读卡器读写 M1

```
ocx.DS600LoadReaderLib();  //加载 dcrf32.dll  读卡器库
Var  handle = ocx.DS600OpenReader();  //打开读卡器
Var ret = DS600_dc_card_hex (HANDLE  icdev);  //寻卡
Ret = eval('('+ret')');
If(Ret.RtnCode != 0)
{
    Output(“寻卡失败”);
    Return;
}
Output(“寻卡成功，卡片 sn = ”+Ret. Snrstr);          //卡号

Var sectorIndex = 4;
Ret = ocx. DS600_dc_authentication_pass(handle, 0, sectorIndex, “ffffffffffff”);  //使用 A 密码
认证第 4 扇区
Ret = ocx. DS600_dc_read_hex(handle, sectorIndex*4+0);  //读取第四扇区第一块数据
Ret = eval('(' +Ret + ')');
if(Ret.RtnCode != 0)
{
    output("读取失败");
```

```
    return;
}
Output(Ret.data);    //读取到的数据

Var data = “00112233445566778899aabbccddeeff”;
Ret = ocx. DS600_dc_write_hex(handle, sectorIndex*4+0, data);   //写第四扇区第一块数据
If(Ret != 0)
{
    Output(“写入失败”);
    Return;
}
Output(“写入成功”);

ret = ocx.DS600_dc_halt(H);              //终止该卡的操作
```