**打印机windows动态库V1.4说明**

**一 说明**

本公司打印机windows动态库支持5种接口：网口，WiFi，U口，串口，并口。其中网口和WIFI 使用同一组函数，都是TCP协议。

本动态库只包含基本的接口操作函数，可分为几类：

1. 打开接口
2. 通过接口发送指令(打印内容也算指令的一部分)
3. 通过接口接收打印机返回数据
4. 关闭接口

本动态库所有入参，出参，都使用ANSI编码。如果使用unicode编码的代码，请先把入参转成ANSI编码再传入

本动态库都采用C++ \_stdcall 方式

**二 接口函数介绍**

1. 通用类函数

BOOL \_stdcall uniInitNetSev();

int \_stdcall uniConnectNetPortByIp(char \*ip, int port=9100,int Timeout=5000);

int \_stdcall uniOpenUsbByVidPid(int vid, int pid);

int \_stdcall uniOpenUsb();

int \_stdcall uniOpenLpt(char \*lpLptName);

int \_stdcall OpenComA(char \*lpCom, int BaudRate);

int \_stdcall uniWrite(int fs, char \*SendBuf, int SendBufSize);

int \_stdcall uniRead(int fs, char \*RecvBuf, int RecvBufSize);

BOOL \_stdcall uniClose(int fs);

BOOL \_stdcall uniCloseNetServ();

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数原型 | 功能说明 | 返回值 | 参数说明 | 备注 |
| BOOL uniInitNetSev(); | 网口环境初始化函数 | 成功：TRUE,失败：FALSE | 无 | 必须在调用其他网口函数之前调用,最好只调用一次，在程序启动时候初始化 |
| int uniConnectNetPortByIp(char \*ip, int port=9100,int Timeout=5000); | 连接打印机 | 返回网口打印机的句柄。  成功:大于等于1  失败：小于等于0 | Ip：IP地址，ANSIC编码  Port：端口号固定9100，  Timeout：连接的超时时间，超过此时间还没连上，自动返回错误，单位ms(毫秒) |  |
| int uniOpenUsb() | 连接打印机 | 返回U口打印机的句柄。  成功:大于等于1  失败：小于等于0 | 无 | 自动连接本公司的任一打印机,请确保只有一台本公司打印机与电脑U口相连。此函数比较通用，建议使用 |
| int uniOpenUsbByVidPid(int vid, int pid); | 连指定VID,PID的打印机 | 返回指定VID,PID的U口打印机的句柄。  成功:大于等于1  失败：小于等于0 | VID：打印机驱动设备属性里的VID  PID：打印机驱动设备属性里的VID |  |
| int \_stdcall uniOpenLpt(char \*lpLptName); | 连接打印机 | 返回并口打印机的句柄。  成功:大于等于1  失败：小于等于0 | 并口名,比如：LPT1，LPT2...  采用ANSIC编码 |  |
| int \_stdcall uniOpenCom(char \*lpCom, int BaudRate); | 连接打印机 | 返回串口打印机的句柄。  成功:大于等于1  失败：小于等于0 | LpCom:并口名,比如：COM1，COM2...  采用ANSIC编码  BaudRate:波特率。如：9600,19200,38400,115200 |  |
| BOOL uniClose(int fs); | 关闭一个现有的打印机连接句柄 | 成功:TRUE  失败：FALSE | Fs：连接成功的句柄 | 关闭由uniConnectNetPortByIp(),  uniOpenCom(),  uniOpenLpt(),  uniOpenUsb(),  uniOpenUsbByVidPid()  成功返回的句柄 |
| BOOL uniCloseNetServ(); | 网口环境反初始化函数 | 成功:TRUE  失败：FALSE | 无 | 必须和uniInitNetSev()成对使用，在程序结束前调用uniCloseNetServ()反初始化 |

1. 功能类

int \_stdcall uniPrintImg1d76(int fs,char\* imgpath,int printwidth);

int \_stdcall uniPrintImg1b2a(int fs,char\* imgpath);

int \_stdcall uniDownLogo(int fs, char \*strBmpFilePath);

int \_stdcall uniprintBmpInFLASH(int fs, int n, int m);

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 函数原型 | 功能说明 | 返回值 | 参数说明 | 备注 |
| int uniPrintImg1d76(int fs,char\* imgpath,int printwidth); | 通过1D 76指令打印一个黑白bmp图片 | 成功>0,  失败<=0 | fs:连接句柄  Imgpath:图片路径,.bmp格式，最好黑白。  Printwidth：打印机的宽度  0 384,1 512,  2 576 3 640 | 图片的宽度（像素点）必须大于打印机的宽度  打印机宽度一般是 384(58机型) 576（80机型） |
| int uniPrintImg1b2a(int fs,char\* imgpath); | 通过1b 2a指令打印一个黑白bmp图片 | 成功>0,  失败<=0 | fs:连接句柄  Imgpath:图片路径,.bmp格式，最好黑白。 |  |
| int \_stdcall uniDownLogo(int fs, char \*strBmpFilePath) | 把图片下载到打印机（黑白bmp图片） | 成功>0,  失败<=0 | fs:连接句柄  Imgpath:图片路径,.bmp格式，最好黑白。 |  |
| int \_stdcall uniprintBmpInFLASH(int fs, int n, int m) | 打印下载到FLASH中的位图 | 成功>0,  失败<=0 | fs:连接句柄  n：0-255，表示位图，  m：0-3 | 此指令跟 uniDownLogo  配套使用 |

1. 函数调用顺序
2. NET

BOOL bret = uniInitNetSev();

if(!bret)

{

return;

}

int fs = uniConnectNetPortByIp(“192.168.2.136”,9100,5000);

if(fs<=0)

{

return;

}

char SendBuf[30] = {0};

SendBuf[0] = 0x31; // char 1

SendBuf[1] = 0x32; // char 2

SendBuf[2] = 0x33; // char 3

SendBuf[3] = 0x0a; // print cmd

int nret = uniWrite(fs,SendBuf,4);

if(nret <=0)

{

return;

}

nret =uniPrintImg1b2a(fs,”D:\1b2a.bmp”);

if(nret <=0)

{

return;

}

........

........

........

uniClose(fs);

uniCloseNetServ();

1. USB

int fs = uniOpenUsbByVidPid(0x0483,0x5740);

OR

int fs = uniOpenUsb();

if(fs<=0)

{

return;

}

char SendBuf[30] = {0};

SendBuf[0] = 0x31; // char 1

SendBuf[1] = 0x32; // char 2

SendBuf[2] = 0x33; // char 3

SendBuf[3] = 0x0a; // print cmd

int nret = uniWrite(fs,SendBuf,4);

if(nret <=0)

{

return;

}

nret =uniPrintImg1b2a(fs,”D:\1b2a.bmp”);

if(nret <=0)

{

return;

}

........

........

........

uniClose(fs);

1. COM

int fs = uniOpenCom(“COM1”,19200);

if(fs<=0)

{

return;

}

char SendBuf[30] = {0};

SendBuf[0] = 0x31; // char 1

SendBuf[1] = 0x32; // char 2

SendBuf[2] = 0x33; // char 3

SendBuf[3] = 0x0a; // print cmd

int nret = uniWrite(fs,SendBuf,4);

if(nret <=0)

{

return;

}

nret =uniPrintImg1b2a(fs,”D:\1b2a.bmp”);

if(nret <=0)

{

return;

}

........

........

........

uniClose(fs);

1. LPT

int fs = uniOpenLpt(“LPT1”);

if(fs<=0)

{

return;

}

char SendBuf[30] = {0};

SendBuf[0] = 0x31; // char 1

SendBuf[1] = 0x32; // char 2

SendBuf[2] = 0x33; // char 3

SendBuf[3] = 0x0a; // print cmd

int nret = uniWrite(fs,SendBuf,4);

if(nret <=0)

{

return;

}

nret =uniPrintImg1b2a(fs,”D:\1b2a.bmp”);

if(nret <=0)

{

return;

}

........

........

........

uniClose(fs);

1. 其他可用例子（函数形式,此例子基本实现了所有打印机的指令，

但有可能有部分错误，以编程手册指令为准）

/\*\*

\*

\* HT<br>

\* 水平定位指

\* @return int;

\* \*/

int \_stdcall unihorizontalPositioning(int fs){

char data[] = { 0x09 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* LF<br>

\* 打印并换行

\* @return int;

\* \*/

int \_stdcall uniprintAndFeedLine(int fs){

char data[] = { 0x0A };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FF<br>

\* 打印并回到标准模式

\* @return int;

\* \*/

int \_stdcall uniprintAndBackStandardmodel(int fs){

char data[] = { 0x0C };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* CR<br>

\* 打印并跳格

\* @return int

\* \*/

int \_stdcall uniPrintAndCarriageReturn(int fs){

char data[] = { 0x0D };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* CAN<br>

\* 页模式下取消打印数据

\* @return int;

\* \*/

int \_stdcall unicanclePrintDataByPagemodel(int fs){

char data[] = { 0x18 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* DLE EOT n<br>

\* 实时状态传送

\* @param n,1<=n<=4;

\* @return int;

\* \*/

int \_stdcall unisendRealtimestatus(int fs, int n){

char data[] = { 0x10, 0x04, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* DLE ENQ n<br>

\* 实时对打印机请求

\* @param n,1<=n<=2

\* @return int

\* \*/

int \_stdcall unirequestRealtimeForPrint(int fs, int n){

char data[] = { 0x10, 0x05, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* DLE DC4 n m t<br>

\* 实时产生开钱箱脉冲

\* @param m:0,1;

\* @param t,1<=t<=8;

\* @return int

\* \*/

int \_stdcall uniopenCashboxRealtime(int fs, int m, int t){

char data[] = { 0x10, 0x14, 0x01, (BYTE)m, (BYTE)t };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC FF<br>

\* 页模式下打印<br>

\* 该命令只在页模式下有效

\* @return int

\* \*/

int \_stdcall uniprintByPagemodel(int fs){

char data[] = { 0x1B, 0x0C };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* 设置字符右间距<br>

\* @param n,代表间距大小的变量，范围0-255；

\* @return int

\* \*/

int \_stdcall unisetCharRightSpace(int fs, int n){

char data[] = { 0x1B, 0x20, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC ！n<br>

\* 选择打印模式

\* @param n代表打印模式的变量，范围0-255

\* @return int

\* \*/

int \_stdcall uniselectPrintModel(int fs, int n){

char data[] = { 0x1B, 0x21, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC $ nL nH<br>

\* 设置绝对打印位置

\* @param m，n，m+n\*256代表距离行首位置，范围0-255；

\* @return int

\* \*/

int \_stdcall unisetAbsolutePrintPosition(int fs, int m, int n){

char data[] = { 0x1B, 0x24, (BYTE)m, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC % n<br>

\* 选择或取消用户自定义字符

\* @param n，0-255，最低位为1，使用；最低位为0，不使用；

\* @return int

\* \*/

int \_stdcall uniselectOrCancleCustomChar(int fs, int n){

char data[] = { 0x1B, 0x25, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC $ Y c1 c2 [x1 d1...d(x1\*y)]...[xk d1...d(y\*xk)]<br>

\* 定义用户自定义字符

\* @param c1,c1是起始字符代码,范围32<=c1<=c2<=127

\* @param c2，c2是终止字符代码，范围32<=c1<=c2<=127;

\* @param b，字节数组b的格式为[x1 d1...d(x1\*y)]...[xk d1...d(y\*xk)]，为字符下载的字符的数据的集合,自定义字符的规则详情，请参考编程手册；

\* @return int

\* \*/

int \_stdcall unidefineuserDefinedCharacters(int fs, int c1, int c2, char b[], int nlen){

char data[] = { 0x1B, 0x26, 0x03, (BYTE)c1, (BYTE)c2 };

char data2[200 \* 1024] = { 0 };

byteMerger(data, sizeof(data), b, nlen, data2);

return uniWrite(fs, data2, sizeof(data) + nlen);

}

/\*\*

\* ESC \* m nL nH d1 d2...dk<br>

\* 选择位图模式<br>

\* 该指令详情请参考编程手册

\* @param m范围：0,1,32，33,代表不同点的密度

\* @param nL范围：0-255

\* @param nH范围：0-3

\* @param b，b={d1 d2...dk}，m=0或1,k=nL+nH\*256;m=32或33,k=(nL+nH\*256)\*3;0<=d<=255;

\* @return int

\* \*/

int \_stdcall uniselectBmpModel(int fs, int m, int nL, int nH, char b[]){

char data[] = { 0x1B, 0x2A, (BYTE)m, (BYTE)nL, (BYTE)nH };

int len = 0;

if ((0 == m) || (1 == m))

{

len = nL + nH \* 256;

}

else

if ((32 == m) || (33 == m))

{

len = (nL + nH \* 256) \* 3;

}

else

{

return -1;

}

char datadest[200 \* 1024] = { 0 };

byteMerger(data, sizeof(data), b, len, datadest);

return uniWrite(fs, datadest, len + 5);

}

/\*\*

\* ESC-n<br>

\* 选择或者取消下划线模式

\* @param n:0,48 取消下划线；1,49选择下划线（1点宽）;2,50选择下划线（2点宽）;

\* @return int

\* \*/

int \_stdcall uniselectOrCancelUnderlineModel(int fs, int n){

char data[] = { 0x1B, 0x2D, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC 2<br>

\* 设置默认行间距

\* 大约3.75mm

\* @return int

\* \*/

int \_stdcall unisetDefultLineSpacing(int fs){

char data[] = { 0x1B, 0x32 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC 3 n<br>

\* 设置行间距

\* @param n范围：0-255

\* \*/

int \_stdcall unisetLineSpaceing(int fs, int n){

char data[] = { 0x1B, 0x33, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC=n<br>

\* 选择打印机

\* @param n：0或1

\* @return int

\* \*/

int \_stdcall uniselectPrinter(int fs, int n){

char data[] = { 0x1B, 0x3D, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC ? n<br>

\* 取消用户自定义字符

\* 取消用户自定义字符中代码为n的字符

\* @param n范围:32-127

\* @return int

\* \*/

int \_stdcall unicancelUserDefinedCharacters(int fs, int n){

char data[] = { 0x1B, 0x3F, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC @<br>

\* 初始化打印机

\* @return int

\* \*/

int \_stdcall uniinitializePrinter(int fs){

char data[] = { 0x1B, 0x40 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC D n1...nk NUL;<br>

\* 设置横线跳格位置

\* @param b={n1...nk};k<=32;跳格距离=字符宽度\*n；<br>

\* 参数b内的n值大小和个数设置请参考编程手册

\* @return int

\* \*/

int \_stdcall unisetHorizontalmovementPosition(int fs, char b[], int nblen){

char data[] = { 0x1B, 0x44 };

char nul = { 0x00 };

char data2[200 \* 1024] = { 0 };

byteMerger(data, sizeof(data), b, nblen, data2);

return uniWrite(fs, data2, sizeof(data) + nblen + 1);

}

/\*\*

\* ESC E n<br>

\* 选择或取消加粗模式

\* @param n，0-255，最低位为1，选择加粗；最低位为0，取消加粗；

\* @return int

\* \*/

int \_stdcall uniselectOrCancelBoldModel(int fs, int n){

char data[] = { 0x1B, 0x45, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC G n<br>

\* 选择或取消双重打印模式

\* @param n，0-255，最低位为1，选择双重打印；最低位为0，取消双重打印；

\* @return int

\* \*/

int \_stdcall uniselectOrCancelDoubelPrintModel(int fs, int n){

char data[] = { 0x1B, 0x47, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC J n<br>

\* 打印并走纸

\* @param n走纸距离，单位英寸，范围：0-255

\* @return int

\* \*/

int \_stdcall uniprintAndFeed(int fs, int n){

char data[] = { 0x1B, 0x4A, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC L<br>

\* 选择页模式

\* @return int

\* \*/

int \_stdcall uniselectPageModel(int fs){

char data[] = { 0x1B, 0x4C };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC M n<br>

\* 选择字体

\* @param n 范围：0，1，48，49；0，48标准字体；1，49压缩字体

\* @return int

\* \*/

int \_stdcall uniselectFont(int fs, int n){

char data[] = { 0x1B, 0x4D, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC R n<br>

\* 选择国际字符集

\* @param n：0-15；代表某一种国际字符集

\* @return int

\* \*/

int \_stdcall uniselectInternationalCharacterSets(int fs, int n){

char data[] = { 0x1B, 0x52, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC S<br>

\* 选择标准模式

\* @return int

\* \*/

int \_stdcall uniselectStandardModel(int fs){

char data[] = { 0x1B, 0x53 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC T n<br>

\* 在页模式下选择打印区域方向

\* @param n：0<=n<=3,48<=n<=51;n指定打印区域的方向和起始位置

\* @return int

\* \*/

int \_stdcall uniselectPrintDirectionUnderPageModel(int fs, int n){

char data[] = { 0x1B, 0x54, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC V n<br>

\* 选择或取消顺时针旋转90度<br>

\* 此命令只在页模式下有效

\* @param n:0<=n<=1,48<=n<=49;0,48代表取消，1,49代表选择

\* @return int

\* \*/

int \_stdcall uniselectOrCancelCW90(int fs, int n){

char data[] = { 0x1B, 0x56, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC W xL xH yL yH dxL dxH dyL dyH<br>

\* 页模式下设置打印区域

\* @param 0<=xL xH yL yH dxL dxH dyL dyH<=255

\* @return int

\* \*/

int \_stdcall unisetPrintAreaUnderPageModel(int fs, int xL, int xH, int yL, int yH, int dxL, int dxH, int dyL, int dyH){

char data[] = { 0x1B, 0x57, (BYTE)xL, (BYTE)xH, (BYTE)yL, (BYTE)yH, (BYTE)dxL, (BYTE)dxH, (BYTE)dyL, (BYTE)dyH };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC \ nL nH<br>

\* 设置相对横向打印位置<br>

\* 指令使用详情请参考编程手册

\* @param nL,nH范围：0-255

\* @return int

\* \*/

int \_stdcall unisetRelativeHorizontalPrintPosition(int fs, int nL, int nH){

char data[] = { 0x1B, 0x5c, (BYTE)nL, (BYTE)nH };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC a n<br>

\* 选择对齐方式

\* @param n:0-2或48-50；分别表示：左对齐，居中，右对齐

\* @return int

\* \*/

int \_stdcall uniselectAlignment(int fs, int n){

char data[] = { 0x1B, 0x61, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC c 3 n<br>

\* 选择打印机纸传感器以输出缺纸信号

\* @param n：0-255

\* @return int

\* \*/

int \_stdcall uniselectPrintTransducerOutPutPageOutSignal(int fs, int n){

char data[] = { 0x1B, 0x63, 0x33, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC c 4 n<br>

\* 选择打印机传感器一停止打印

\* @param n:0-255

\* @return int

\* \*/

int \_stdcall uniselectPrintTransducerStopPrint(int fs, int n){

char data[] = { 0x1B, 0x63, 0x34, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC c 5 n<br>

\* 允许或禁止按键

\* @param n：0-255，n的最后一位为0，按键启用，为1，按键禁止；

\* @return int

\* \*/

int \_stdcall uniallowOrForbidPressButton(int fs, int n){

char data[] = { 0x1B, 0x63, 0x35, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC d n<br>

\* 打印并向前走纸n行

\* @param n:0-255；

\* @return int

\* \*/

int \_stdcall uniprintAndFeedForward(int fs, int n){

char data[] = { 0x1B, 0x64, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC p m t1 t2<br>

\* 产生钱箱控制脉冲

\* @param m ：连接引脚，范围：0,1,48,49；

\* @param t1,t2：0-255

\* @return int

\* \*/

int \_stdcall unicreatCashboxContorlPulse(int fs, int m, int t1, int t2){

char data[] = { 0x1B, 0x70, (BYTE)m, (BYTE)t1, (BYTE)t2 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC t n<br>

\* 选择字符代码表

\* @param n：0<=n<=10,16<=n<=19;

\* @return int

\* \*/

int \_stdcall uniselectCharacterCodePage(int fs, int n){

char data[] = { 0x1B, 0x74, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC { n<br>

\* 选择或取消倒置打印模式

\* @param n：最低位0取消，最低位1选择，范围：0-255；

\* @return int

\* \*/

int \_stdcall uniselectOrCancelConvertPrintModel(int fs, int n){

char data[] = { 0x1B, 0x7B, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS p n m<br>

\* 打印下载到FLASH中的位图

\* @param n，表示位图，0-255

\* @param m：0-3或48-51

\* @return int

\* \*/

int \_stdcall uniprintBmpInFLASH(int fs, int n, int m){

char data[] = { 0x1C, 0x70, (BYTE)n, (BYTE)m };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS q n [xL xH yL yH d1...dk]...[xL xH yL yH d1...dk]<br>

\* 定义FLASH位图<br>

\* FLASH的最大存储空间为8kb，即[xL xH yL yH d1...dk]...[xL xH yL yH d1...dk]的总和不能超过8096个元素<br>

\* 单个bitmap需要满足：n=(width+7)/8;h=(height+7)/8;n\*h<=1023,1<=n<=1023,1<=h<=288,否则返回的位图数据为new byte[0].

\* 位图的高度height,最好能被8整除

\* @param list bitmap列表，存放你要存储到flash里的位图对象,list为空或size为0，返回一个new byte[0]；

\* @param n 位图个数，也是list的size

\* @param bmptype你所希望的位图打印的类型，二值法和抖动法

\* @return int

\*

\* \*/

// int \_stdcall unidefinedFlashBmp(int fs, List<Bitmap> list, int n, BmpType bmpType){

// intdata = { 0x1C, 0x71, (byte)n };

// if (list == null || list.size(int fs) == 0) {

// return new byte[0];

// }

// for (int i = 0; i < list.size(int fs); i++) {

// Bitmap bitmap = list.get(i);

// int bmp = BitmapToByteData.flashBmpToSendData(bitmap, bmpType);

// data = byteMerger(data, bmp);

// }

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* GS ! n<br>

\* 选择字符大小

\* @param n 范围0-255；n的0到3位设定字符高度，4-7位用来设定字符宽度

\* @return int

\* \*/

int \_stdcall uniselectCharacterSize(int fs, int n){

char data[] = { 0x1D, 0x21, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS $ nL nH<br>

\* 页模式下设置绝对位置

\* @param nL,nH,nl+nh\*256代表位置，单位英寸；nl和nh的范围：0-255；

\* @return int

\* \*/

int \_stdcall unisetAbsolutePositionUnderPageModel(int fs, int nL, int nH){

char data[] = { 0x1D, 0x24, (BYTE)nL, (BYTE)nH };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS x y [ d1...d(xy\*8)]<br>

\* 定义下载位图<br>

\*

\* 单个bitmap需要满足：n=(width+7)/8;h=(height+7)/8;n\*h<=912,1<=n<=255,1<=h<=48,否则返回的位图数据为new byte[0].

\* 位图的高度height,最好能被8整除，否则有可能导致打印出错

\*

\* @param bitmap 你所希望下载到打印机的位图

\* @param bmptype你所希望的位图打印的类型，二值法和抖动法

\* @return int

\* \*/

// int \_stdcall unidefinedDownLoadBmp(int fs, Bitmap bitmap, BmpType bmpType){

// char data[] = { 0x1D, 0x2A };

// int bmp = BitmapToByteData.downLoadBmpToSendData(bitmap, bmpType);

// data = byteMerger(data, bmp);

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* GS ( A pL pH n m<br>

\* 执行打印机数据十六进制转储

\* @return int

\* \*/

int \_stdcall uniexecutePrintDataSaveByTransformToHex(int fs){

char data[] = { 0x1D, 0x28, 0x41, 0x02, 0x00, 0x00, 0x01 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS / m<br>

\* 打印下载位图

\* @param m，指定打印模式，范围：0-3或48-51；

\* @return int

\* \*/

int \_stdcall uniprintDownLoadBmp(int fs, int m){

char data[] = { 0x1D, 0x2F, (BYTE)m };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS ：<br>

\* 开始或结束宏定义

\* @return int

\* \*/

int \_stdcall unistartOrStopMacrodeFinition(int fs){

char data[] = { 0x1D, 0x3A };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS B n<br>

\* 选择或取消黑白反显打印模式

\* @param n：0-255，最低位为0，取消反显打印，最低位为1，选择反显打印

\* @return int

\* \*/

int \_stdcall uniselectOrCancelInvertPrintModel(int fs, int n){

char data[] = { 0x1D, 0x42, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS H n<br>

\* 选择HRI字符打印位置;

\* HRI是对条码内容注释的字符;

\* @param n:范围0-3或者48-51；代表字符在条码的打印位置

\* @return int

\* \*/

int \_stdcall uniselectHRICharacterPrintPosition(int fs, int n){

char data[] = { 0x1D, 0x48, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS L nL nH<br>

\* 设置左边距

\* @param nL，nH，范围0-255，（nl+nh\*256）\*横向移动单位，代表设置的左边距，单位：英寸

\* @return int

\* \*/

int \_stdcall unisetLeftSpace(int fs, int nL, int nH){

char data[] = { 0x1D, 0x4C, (BYTE)nL, (BYTE)nH };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS P x y<br>

\* 设置横向和纵向移动单位<br>

\* @param x设置横向移动单位的参数，0-255

\* @param y设置纵向移动单位的参数，0-255

\* 当x和y为0时，xy被设置为默认值

\* @return int

\* \*/

int \_stdcall unisetHorizontalAndVerticalMoveUnit(int fs, int x, int y){

char data[] = { 0x1D, 0x50, (BYTE)x, (BYTE)y };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS V m<br>

\* 选择切纸模式并切纸

\* @param m：0，48，全切；1,49，半切

\* @return int

\* \*/

int \_stdcall uniselectCutPagerModerAndCutPager(int fs, int m){

char data[] = { 0x1D, 0x56, (BYTE)m };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS V m n<br>

\* 选择切纸模式并切纸

\* @param m：只能是66

\* @param n:进纸n，然后半切纸

\* @return int

\* \*/

int \_stdcall uniselectCutPagerModerAndCutPager2(int fs, int m, int n){

char data[] = { 0x1D, 0x56, (BYTE)66, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS W nL nH<br>

\* 设置打印区域宽度

\* @param nL,(nl+nh\*256)\*横向移动单位，代表打印区域宽度

\* @param nH,(nl+nh\*256)\*横向移动单位，代表打印区域宽度

\*

\* @return int

\* \*/

int \_stdcall unisetPrintAreaWidth(int fs, int nL, int nH){

char data[] = { 0x1D, 0x57, (BYTE)nL, (BYTE)nH };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS \ nL nH<br>

\* 页模式下设置相对打印位置

\* @param nL,(nl+nh\*256)\*纵向移动单位，表示相对当前打印位置纵向移动距离

\* @param nH,(nl+nh\*256)\*纵向移动单位，表示相对当前打印位置纵向移动距离

\* @return int

\* \*/

int \_stdcall unisetVerticalRelativePositionUnderPageModel(int fs, int nL, int nH){

char data[] = { 0x1D, 0x5C, (BYTE)nL, (BYTE)nH };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS ^ r t m<br>

\* 执行宏命令

\* @param r：0-255，指定执行次数

\* @param t：0-255，执行等等时间

\* @param m：0或1，执行模式

\* @return int

\* \*/

int \_stdcall uniexecuteMacrodeCommand(int fs, int r, int t, int m){

char data[] = { 0x1D, 0x5E, (BYTE)r, (BYTE)t, (BYTE)m };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS a n<br>

\* 打开或关闭自动状态返回功能

\* @param n，0-255，每一位代表不同的状态返回

\* @return int

\* \*/

int \_stdcall uniopenOrCloseAutoReturnPrintState(int fs, int n){

char data[] = { 0x1D, 0x61, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS f n<br>

\* 选择HRI使用字体

\* @param n ：0,48代表标准；1，49代表压缩字体；

\* @return int

\* \*/

int \_stdcall uniselectHRIFont(int fs, int n){

char data[] = { 0x1D, 0x66, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS h n<br>

\* 选择条码高度

\* @param n ：1-255；默认162

\* @return int

\* \*/

int \_stdcall unisetBarcodeHeight(int fs, int n){

char data[] = { 0x1D, 0x68, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS k m d1...dk NUL<br>

\* 打印条码

\* @param m 条码类型：0-6

\* @param content 条码内容字符串，字符串的长度和字符范围，参考打印条码指令

\*

\* @return int

\* \*/

// int \_stdcall uniprintBarcode(int fs, int m, char \*content){

// char data[] = { 0x1D, 0x6B, (byte)m };

// int end = { 0x00 };

// int text = strTobytes(content);

// data = byteMerger(data, text);

// data = byteMerger(data, end);

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* GS k m n d1...dn<br>

\* 打印条码

\* @param m 条码类型：65-73，当m=73时，打印条码的内容必须加上字符集选择，如：{A，{B，{C等；

\* @param n 指示条码数据的个数,字符串长度不能超过n规定的范围

\* @param content 条码内容字符串，字符串的长度和字符范围，参考打印条码指令

\*

\* @return int

\* \*/

// int \_stdcall uniprintBarcode(int fs, int m, int n, String content){

// char data[] = { 0x1D, 0x6B, (byte)m, (byte)n };

// int text = strTobytes(content);

// data = byteMerger(data, text);

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* GS r n<br>

\* 返回状态

\* @param n 范围：1,2,49,50;1,49返回传感器状态,2,50返回钱箱状态;

\* @return int

\*

\* \*/

int \_stdcall unireturnState(int fs, int n){

char data[] = { 0x1D, 0x72, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS v 0 m <br>

\* 打印光栅位图

\* @param m 打印模式0-3或48-51；正常打印m=0；

\* @param bitmap你希望打印光栅位图的图片的bitmap对象，该模式的位图打印规则请参考编程手册

\* @param bmpType你希望的位图处理成单色图的方式

\* @return int

\* \*/

// int \_stdcall uniprintRasterBmp(int fs, int m, Bitmap bitmap, BmpType bmpType, AlignType alignType, int pagewidth){

// char data[] = BitmapToByteData.rasterBmpToSendData(m, bitmap, bmpType, alignType, pagewidth);

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* GS w n<br>

\* 设置条码宽度

\* @param n：2-6；默认3；

\* @return int

\* \*/

int \_stdcall unisetBarcodeWidth(int fs, int n){

char data[] = { 0x1D, 0x77, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS ! n<br>

\* 设置汉字字符模式

\* @param n:0-255；字节的不同位定义字符模式

\* @return int

\* \*/

int \_stdcall unisetChineseCharacterModel(int fs, int n){

char data[] = { 0x1C, 0x21, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS &<br>

\* 选择汉字模式

\* @return int

\* \*/

int \_stdcall uniselectChineseCharModel(int fs){

char data[] = { 0x1C, 0x26 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS - n<br>

\* 选择或取消汉字下划线模式

\* @param n：0-2或48-50；

\* @return int

\* \*/

int \_stdcall uniselectOrCancelChineseCharUnderLineModel(int fs, int n){

char data[] = { 0x1C, 0x2D, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS .<br>

\* 取消汉字模式

\* @return int

\* \*/

int \_stdcall uniCancelChineseCharModel(int fs){

char data[] = { 0x1C, 0x2E };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS 2 c1 c2 d1...dk<br>

\* 定义用户自定义汉字

\*

\* @param c2:A1H<=c2<=FEH

\* @param b,长度为72，代表定义的汉字的数据int

\* @return int

\* \*/

// int \_stdcall unidefinedUserDefinedChineseChar(int fs, int c2, int b){

// char data[] = { 0x1C, 0x32, (byte)0xFE, (byte)c2 };

// data = byteMerger(data, b);

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* FS S n1 n2<br>

\* 设置汉字字符左右间距

\* @param n1,设置左间距，0-255；

\* @param n2,设置右间距，0-255；

\* @return int

\* \*/

int \_stdcall unisetChineseCharLeftAndRightSpace(int fs, int n1, int n2){

char data[] = { 0x1C, 0x53, (BYTE)n1, (BYTE)n2 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* FS W n<br>

\* 选择或取消汉字倍高倍宽

\* @param n：0-255；只有最低位有效，0，取消；1，选择

\* @return int

\* \*/

int \_stdcall uniselectOrCancelChineseCharDoubleWH(int fs, int n){

char data[] = { 0x1C, 0x57, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC B n t<br>

\* 打印机来单打印蜂鸣提示

\* @param n 蜂鸣次数.1-9

\* @param t 每次蜂鸣时间，t\*50ms,1-9

\* @return int

\* \*/

int \_stdcall uniprinterOrderBuzzingHint(int fs, int n, int t){

char data[] = { 0x1B, 0x42, (BYTE)n, (BYTE)t };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* ESC C m t n<br>

\* 打印机来单蜂鸣提示及报警灯闪烁

\* @param m 蜂鸣次数报警灯闪烁次数.范围1-20

\* @param t 每次鸣叫或闪烁间隔时间，（t\*50ms），范围1-20

\* @param n 0-3；分别表示是否鸣叫，闪烁；

\* @return int

\* \*/

int \_stdcall uniprinterOrderBuzzingAndWarningLight(int fs, int m, int t, int n){

char data[] = { 0x1B, 0x43, (BYTE)m, (BYTE)t, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* GS ( k p L p H cn fn n (fn = 65)<br>

\* Sets the number of columns of the data area for PDF417<br>

\* (p L + p H × 256) = 3 (p L = 3, p H = 0)<br>

\* cn = 48<br>

\* fn = 65<br>

\* @param 0 ≤ n ≤ 30;

\* @return int

\* \*/

int \_stdcall uniSetsTheNumberOfColumnsOfTheDataAreaForPDF417(int fs, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 48, 65, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Sets the number of rows of the data area for PDF417<br>

\* GS ( k p L p H cn fn n (fn = 66)<br>

\* (p L + p H × 256) = 3 (p L = 3, p H = 0)<br>

\* cn = 48<br>

\* fn = 66<br>

\* @param n=0,3 ≤ n ≤ 90;

\* @return int

\* \*/

int \_stdcall uniSetsTheNumberOfRowsOfTheDataAreaForPDF417(int fs, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 48, 66, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Sets the module width of one PDF417 symbol to n dots.<br>

\* GS ( k p L p H cn fn n (fn = 67)<br>

\* [Range] (p L + p H × 256) = 3 (p L = 3, p H = 0)<br>

\* cn = 48 <br>

\* fn = 67 <br>

\* @param 2 ≤ n ≤ 8 <br>

\* [Default] n =3<br>

\* @return int

\*

\* \*/

int \_stdcall uniSetsTheModuleWidthOfPDF417(int fs, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 48, 67, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Sets the module height to [(module width) × n].<br>

\* GS ( k p L p H cn fn n (fn = 68)<br>

\* [Range] (p L + p H × 256) = 3 (p L = 3, p H = 0)<br>

\* cn = 48<br>

\* fn = 68<br>

\* @param 2 ≤ n ≤ 8<br>

\* [Default] n = 3<br>

\* @return int

\* \*/

int \_stdcall uniSetsTheModuleHeightForPDF417(int fs, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 48, 68, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Sets the error correction level for PDF417 symbols.<br>

\* GS ( k p L p H cn fn m n (fn = 69)<br>

\* [Range] (p L + p H × 256) = 4 (p L = 4, p H = 0)<br>

\* cn = 48<br>

\* fn = 69<br>

\* @param m = 48, 49<br>

\* @param n:48 ≤ n ≤ 56 (when m = 48 is specified),1 ≤ n ≤ 40 (when m = 49 is specified)<br>

\* [Default] m = 49, n = 1<br>

\* @return int

\* \*/

int \_stdcall uniSetsTheErrorCorrectionLevelForPDF417(int fs, int m, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x04, 0x00, 48, 69, (BYTE)m, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Specifies or cancels various PDF417 symbol options<br>

\* GS ( k p L p H cn fn m (fn = 70)<br>

\* [Range] (p L + p H × 256) = 3 (p L = 3, p H = 0)<br>

\* cn = 48<br>

\* fn = 70<br>

\* @param m = 0 or 1<br>

\* [Default] m = 0

\* @return int

\* \*/

int \_stdcall uniSpecifiesOrCancelsVariousPDF417SymbolOptions(int fs, int m){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 48, 70, (BYTE)m };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Stores symbol data (d1...dk) in the PDF417 symbol storage area.<br>

\* GS ( k p L p H cn fn m d1 … dk (fn = 80)<br>

\* [Range] 4 ≤ (p L + p H × 256) ≤ 65535 (0 ≤ p L ≤ 255, 0 ≤ p H ≤ 255)<br>

\* cn = 48<br>

\* fn = 80<br>

\* m = 48<br>

\* @param b={d1...dk},k为b的长度<br>

\* 0 ≤ d ≤ 255<br>

\* k = (p L + p H × 256) – 3<br>

\* @return int

\* \*/

int \_stdcall uniStoresSymbolDataInThePDF417SymbolStorageArea(int fs, int pL, int pH, char b[], int nblen){

char data[] = { 0x1D, 0x28, 0x6B, (BYTE)pL, (BYTE)pH, 48, 80, 48 };

char data2[200 \* 1024] = { 0 };

byteMerger(data, sizeof(data), b, nblen, data2);

return uniWrite(fs, data2, sizeof(data) + nblen);

}

/\*\*

\* Prints the PDF417 symbol data in the symbol storage area.<br>

\* GS ( k p L p H cn fn m (fn = 81)<br>

\* [Range] (p L + p H × 256) = 3 (p L = 3, p H = 0)<br>

\* cn = 48<br>

\* fn = 81<br>

\* m = 48<br>

\* @return int

\* \*/

int \_stdcall uniPrintsThePDF417SymbolDataInTheSymbolStorageArea(int fs){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 48, 81, 0x30 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Transmits the size of the symbol data in the symbol storage area.<br>

\* GS ( k p L p H cn fn m (fn = 82)<br>

\* Hex 1D 28 6B p L p H cn fn m<br>

\* [Range] (p L + p H × 256) = 3 (p L = 3, pH = 0)<br>

\* cn = 48<br>

\* fn = 82<br>

\* m = 48<br>

\* @return int

\* \*/

int \_stdcall uniTransmitsTheSizeOfTheSymbolDataInTheSymbolStorageAreaPDF417(int fs){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 48, 82, 48 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Specifies the mode for QRCode symbol by n1;<br>

\* GS ( k p L p H cn fn n1 n2 (fn = 65);<br>

\* Hex 1D 28 6B p L p H cn fn n1 n2;<br>

\* [Range] p L = 4, p H = 0;<br>

\* cn = 49;<br>

\* fn = 65;<br>

\* @param n1:n1 = 49, 50;

\* n2 = 0;<br>

\* [Default] n1 = 50;<br>

\* n2 = 0<br>

\* @return int

\* \*//\*

int \_stdcall uniSpecifiesTheModeForQRCodeSymbolByn1(int n1){

char data[]={0x1D,0x28,0x6B,0x04,0x00,49,65,(byte) n1,0x00};

return uniWrite(fs,data,sizeof(data));}\*/

/\*\*

\* Sets the size of the QRCode symbol module to [n dots × n dots]<br>

\* GS ( k p L p H cn fn n (fn = 67)<br>

\* Hex 1D 28 6B p L p H cn fn n<br>

\* [Range] p L = 3, p H = 0<br>

\* cn = 49<br>

\* fn = 67<br>

\* @param n 1<=n<=16

\* [Default] n = 3<br>

\* @return int

\* \*/

int \_stdcall uniSetsTheSizeOfTheQRCodeSymbolModule(int fs, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x30, 0x67, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Sets the error correction level for QRCode symbol.<br>

\* GS ( k p L p H cn fn n (fn = 69)<br>

\* Hex 1D 28 6B p L p H cn fn n<br>

\* [Range] p L = 3, p H = 0<br>

\* cn = 49<br>

\* fn = 69<br>

\* @param 48 ≤ n ≤ 51

\* [Default] n = 48<br>

\* @return int

\* \*/

int \_stdcall uniSetsTheErrorCorrectionLevelForQRCodeSymbol(int fs, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x30, 0x69, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Stores symbol data (d1…dk) in the QRCode symbol storage area.<br>

\* GS ( k p L p H cn fn m d1…dk (fn = 80)<br>

\* Hex 1D 28 6B p L p H cn fn m d1…dk<br>

\* [Range] 4 ≤ (p L + p H × 256) ≤ 7092 (0 ≤ p L ≤ 255, 0 ≤ p H ≤ 28)<br>

\* cn = 49<br>

\* fn = 80<br>

\* m = 48<br>

\* 32≤ d ≤ 255<br>

\*

\* @param code qrcode content<br>

\*

\* k = (p L + p H × 256)<br>

\* @return int

\* \*/

// int \_stdcall uniStoresSymbolDataInTheQRCodeSymbolStorageArea(int fs, String code){

// int b = strTobytes(code);

// int a = b.length;

// int pL = 0, pH = 0;

// if (a <= 255) {

// pL = a;

// pH = 0;

// }

// else {

// pH = a / 256;

// pL = a % 256;

// }

// char data[] = { 0x1D, 0x28, 0x6B, 0x30, (byte)0x80, (byte)pL, (byte)pH };

// data = byteMerger(data, b);

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* Prints the QRCode symbol data in the symbol storage area.<br>

\* GS ( k p L p H cn fn m (fn = 81)<br>

\* Hex 1D 28 6B p L p H cn fn m<br>

\* [Range] p L = 3, p H = 0<br>

\* cn = 49<br>

\* fn = 81<br>

\* m = 48<br>

\* @return int

\* \*/

int \_stdcall uniPrintsTheQRCodeSymbolDataInTheSymbolStorageArea(int fs){

char data[] = { 0x1D, 0x28, 0x6B, 0x30, (BYTE)0x81 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* 封装好的QRcode打印指令<br>

\* @param n 单元大小，默认值为n=3

\* @param errLevel 纠错等级 48-51；

\* @param code 二维码内容字符串

\*

\* \*/

// int \_stdcall uniprintQRcode(int fs, int n, int errLevel, String code){

// int b = strTobytes(code);

// int a = b.length;

// int nL = 0, nH = 0;

// if (a <= 255) {

// nL = a;

// nH = 0;

// }

// else {

// nH = a / 256;

// nL = a % 256;

// }

// char data[] = { 0x1D, 0x28, 0x6B, 0x30, 0x67, (byte)n, 0x1D, 0x28, 0x6B, 0x30, 0x69, (byte)errLevel, 0x1D, 0x28, 0x6B, 0x30, (byte)0x80, (byte)nL, (byte)nH };

// data = byteMerger(data, b);

// int c = { 0x1D, 0x28, 0x6B, 0x30, (byte)0x81 };

// data = byteMerger(data, c);

// return uniWrite(fs,data,sizeof(data));

//}

/\*\*

\* Transmits the size of the symbol data in the symbol storage area<br>

\* GS ( k p L p H cn fn m (fn = 82)<br>

\* Hex 1D 28 6B p L p H cn fn m<br>

\* [Range] p L = 3, pH = 0<br>

\* cn = 49<br>

\* fn = 82<br>

\* m = 48<br>

\* @return int

\* \*/

int \_stdcall uniTransmitsTheSizeOfTheSymbolDataInTheSymbolStorageAreaQRCode(int fs){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 49, 82, 48 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Specifies the mode for MaxiCode symbol<br>

\* GS ( k p L p H cn fn n (fn = 65)<br>

\* Hex 1D 28 6B p L p H cn fn n<br>

\* [Range] p L = 3, p H = 0<br>

\* cn = 50<br>

\* fn = 65<br>

\* @param 50 ≤ n ≤ 54<br>

\* [Default] n = 50<br>

\* @return int

\* \*/

int \_stdcall uniSpecifiesTheModeForMaxiCodeSymbol(int fs, int n){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 50, 65, (BYTE)n };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Stores symbol data (d1…dk) in the MaxiCode symbol storage area.<br>

\* GS ( k p L p H cn fn m d1…dk (fn = 80)<br>

\* Hex 1D 28 6B p L p H cn fn m d1…dk<br>

\* [Range] 4 ≤ (p L + p H × 256) ≤ 141 (0 ≤ p L ≤ 141, p H = 0)<br>

\* cn = 50<br>

\* fn = 80<br>

\* m = 48<br>

\* 1 ≤ d ≤ 255<br>

\* k = (p L + p H × 256) – 3<br>

\* @param pL :int

\* @param pH :int

\* @param b :int

\* @return int

\* \*/

int \_stdcall uniStoresSymbolDataInItheMaxiCodeSymbolStorageArea(int fs, int pL, int pH, char b[], int nblen){

char data[] = { 0x1D, 0x28, 0x6B, (BYTE)pL, (BYTE)pH, 50, 80, 48 };

char data2[200 \* 1024] = { 0 };

byteMerger(data, sizeof(data), b, nblen, data2);

return uniWrite(fs, data2, sizeof(data) + nblen);

}

/\*\*

\* Prints the MaxiCode symbol data in the symbol storage area<br>

\* GS ( k p L p H cn fn m (fn = 81)<br>

\* Hex 1D 28 6B p L p H cn fn m<br>

\* [Range] p L = 3, p H = 0<br>

\* cn = 50<br>

\* fn = 81<br>

\* m = 48<br>

\* @return int

\* \*/

int \_stdcall uniPrintsTheMaxiCodeSymbolDataInTheSymbolStorageArea(int fs){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 50, 81, 48 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* Transmits the size of the encoded symbol data in the symbol storage area<br>

\* ASCII GS ( k p L p H cn fn m<br>

\* Hex 1D 28 6B p L p H cn fn m<br>

\* [Range] p L = 3, pH = 0<br>

\* cn = 50<br>

\* fn = 82<br>

\* m = 48<br>

\* @return int

\* \*/

int \_stdcall uniTransmitsTheSizeOfTheEncodedSymbolDataInTheSymbolStorageAreaMaxiCode(int fs){

char data[] = { 0x1D, 0x28, 0x6B, 0x03, 0x00, 50, 82, 48 };

return uniWrite(fs, data, sizeof(data));

}

/\*\*

\* 字符串转byte数组<br>

\* @param 文本内容

\* @return int

\* \*/

//private int strTobytes(String str){

// int b = null, data = null;

// try {

// b = str.getBytes("utf-8");

// if (charsetName == null | charsetName == "") {

// charsetName = "gbk";

// }

// data = new String(b, "utf-8").getBytes(charsetName);

// }

// catch (UnsupportedEncodingException e) {

// // TODO Auto-generated catch block

// e.printStackTrace(int fs);

// }

// return data;

//}

BOOL \_stdcall uniDownLogo(int fs, char \*strBmpFilePath)

{

if (-1 == (\_access(strBmpFilePath, 0)))

{

return FALSE;

}

CString spath = strBmpFilePath;

CImage Timg;

Timg.Load(spath); //load an image

DWORD dwWidth;

DWORD dwHeight;

DWORD x\_width, X;

DWORD y\_hight, Y;

COLORREF Color;

int n;

int m;

char bByteTemp;

char \*SendBuf;

X = x\_width = Timg.GetWidth();

Y = y\_hight = Timg.GetHeight();

if (X % 8) X = 8 \* (1 + X / 8);

if (Y % 8) Y = 8 \* (1 + Y / 8);

DWORD BmpSize = X\*(Y / 8) + 10;

BOOL result = TRUE;

SendBuf = (char \*)malloc(BmpSize);

try

{

memset(SendBuf, 0, BmpSize);

m = 7;

for (dwWidth = 0; dwWidth < x\_width; dwWidth++)

{

for (dwHeight = 0; dwHeight < Y;)

{

bByteTemp = 0;

for (n = 0; n <= 7; n++)

{

if (dwHeight >= y\_hight)

{

dwHeight = dwHeight + 1;

continue;

}

Color = Timg.GetPixel(dwWidth, dwHeight);

if (Color < 8000000)

{

bByteTemp |= (0x80 >> n);

}

dwHeight = dwHeight + 1;

}

SendBuf[m] = bByteTemp;

m = m + 1;

}

}

SendBuf[0] = 0x1C;

SendBuf[1] = 0x71;

SendBuf[2] = 1;

SendBuf[3] = X / 8 % 256; //xL

SendBuf[4] = X / 8 / 256; //xH

SendBuf[5] = Y / 8 % 256; //yL

SendBuf[6] = Y / 8 / 256; //yH

SendBuf[m] = 0;

SendBuf[m + 1] = 0;

SendBuf[m + 2] = 0;

int nsend = uniWrite(fs, SendBuf, BmpSize);

if (nsend)

{

BOOL result = TRUE;

}

else

{

BOOL result = FALSE;

}

}

catch (...)

{

free(SendBuf);

}

return result;

}

/\*\*

\* 打印制定文字的二维码

\* @fs 句柄

\* @strTrxData 文本内容

\* @nZoomIn 放大系数 1-16

\* @nErrorLevel 错误修正级别 1=7%，2=15%，3=25%，4=30%

\* @bCut 是否切纸

\* @return >0 成功，<= 0 失败

\* \*/

int \_stdcall uniTwobarCodes(int fs, char \*strTrxData, int nZoomIn = 7, int nErrorLevel = 1, int bCut = 0)

{

// TODO: 在此添加控件通知处理程序代码

//cmd: 1D 28 6B 30 67 n 设置QR Code的单元大小为n点

//cmd: 1D 28 6B 30 69 n QR Code：设置错误纠正等级

//cmd: 1D 28 6B 30 80 pl ph d1...dk 传输QRCODE 的数据（d1…dk）到编码缓存

//cmd: 1D 28 6B 30 81 打印QR Code编码缓存的编码数据

char cmd1[9] = { 0x1B, 0x40, 0x0A, 0x1D, 0x28, 0x6B, 0x30, 0x67, 0x01 };

char cmd2[6] = { 0x1D, 0x28, 0x6B, 0x30, 0x69, 0x48 };

char cmd3[9] = { 0x1D, 0x28, 0x6B, 0x30, 0x80, 0x01, 0x01, 0 }; //print bar data

char send1[11] = { 0x1D, 0x28, 0x6B, 0x30, 0x81, 0x1B, 0x4A, 250, 0x1D, 0x56, 0x01 };

char send2[8] = { 0x1D, 0x28, 0x6B, 0x30, 0x81, 0x1B, 0x4A, 200 };

int sendlen;

char \*temp;

char \*SendBuf;

cmd1[8] = (BYTE)nZoomIn;

int nsend = 0;

nsend = uniWrite(fs, cmd1, 9);

if (!nsend)

{

return -1;

}

switch (nErrorLevel){

case 0:

cmd2[5] = 0x48;

break;

case 1:

cmd2[5] = 0x49;

break;

case 2:

cmd2[5] = 0x50;

break;

case 3:

cmd2[5] = 0x51;

break;

default:

return -100;

}

nsend = uniWrite(fs, cmd2, 6);

if (!nsend)

{

return -1;

}

//print bar data

//USES\_CONVERSION;

temp = strTrxData;

DWORD WriteSize = strlen(temp) + 1;

sendlen = WriteSize + 7;

//SendBuf = new char[7 + WriteSize + 1 / 2];

SendBuf = (char \*)malloc(7 + WriteSize + 1 / 2);

memcpy(&SendBuf[7], temp, WriteSize);

memcpy(SendBuf, cmd3, 7);

SendBuf[5] = WriteSize % 256;

SendBuf[6] = WriteSize / 256;

nsend = uniWrite(fs, SendBuf, sendlen);

if (!nsend)

{

return -1;

}

delete[]SendBuf;

if (bCut)

{

nsend = uniWrite(fs, send1, 11);

if (!nsend)

{

return -1;

}

}

else

{

nsend = uniWrite(fs, send2, 8);

if (!nsend)

{

return -1;

}

}

return 1;

}

/\*\*

\* 打印制定文字的二维码

\* @fs 句柄

\* @nCodeType 条码类型 65 UPC\_A ,66 UPC\_C ,67 EN(JAN)13 ,68 EN(JAN)8 ,69 CODE39 ,70 ITF ,71 CODABAR ,72 CODE93,73 CODE128

\* @High 1<= High <= 255

\* @align 0 左对齐 1 中间对齐 2 右对齐

\* @codeTextalign n 条码字符打印位置

0, 48 不打印

1, 49 条码上方

2, 50 条码下方

3, 51 条码上、下方都打印

\* @width 宽度 根据类型而定，有范围

\* @codeText 字符根据 类型而定，可参考编程手册

\* @return >0 成功，<= 0 失败

\* \*/

int \_stdcall uniPrintbarCodes(int fs, char \*code, char \*codebarText, int nCodeType = 65,

int width = 2, int High = 162, int align = 1, int codeTextalign = 2)

{

int nResult = 0;

char SendBuf[256] = { 0x1B, 0x40, 0x1B, 0x61, 0x01, 0x1D, 0x48, 0x00, 0x1d, 0x77, 0x03, 0x1D, 0x68,

0x03, 0x1D, 0x6B, 0x41, 0x0B };

char endbuf[5] = { 0x0A, 0x0A, 0x1B, 0x40, '\0' };

int codeLen = strlen(code);

int codeTextLen = strlen(codebarText);

SendBuf[4] = align;

SendBuf[7] = codeTextalign;

SendBuf[10] = width;

SendBuf[13] = High;

SendBuf[16] = nCodeType;

SendBuf[17] = codeLen;

memcpy\_s((void\*)&SendBuf[18], 200, code, codeLen);

memcpy\_s((void\*)&SendBuf[18 + codeLen], 200, codebarText, codeTextLen);

memcpy\_s((void\*)&SendBuf[18 + codeLen + codeTextLen], 5, endbuf, 5);

nResult = uniWrite(fs, SendBuf, 18 + codeLen + codeTextLen + 5);

if (!nResult)

{

return -1;

}

return 1;

}

/\*\*

\* 查找全网段在线的打印机

\* @return >0 成功，空 失败

\* \*/

BOOL bret = uniInitNetSev();

if(!bret)

{

return;

}

CString fs = uniNetworksegmentquery ();

if(fs!=””)

{

return;

}

uniCloseNetServ();

例：返回

00-77-00-00-00-00|192.168.2.133|255.255.255.0|192.168.2.1|9100|0;00-72-9E-5B-F0-F8|192.168.2.13|255.255.255.0|192.168.2.1|9100|0;

Print1：

物理地址：00-77-00-00-00-00

IP地址：192.168.2.133

子网掩码：255.255.255.0

网络网关：192.168.2.1

端口号：9100

DHCP：0-关 1-开

/\*\*

\* 跨网段修改IP

\* @mac MAC地址

\* @ip IP地址

\* @subnet 子网掩码

\* @gateway 网络网关

\* @dhcp DHCP开关

0 关闭 1 开启

\* @return >0 成功，<= 0 失败

\* \*/

BOOL bret = uniInitNetSev();

if(!bret)

{

return;

}

Int bret = uniNetworksegmentSet("00-77-00-00-00-00", "192.168.2.133", "255.255.255.0", "192.168.2.1", 0);

if(bret<0)

{

return;

}

uniCloseNetServ();

如有疑问可致电公司技术人员查询，使用愉快。