

打印机接口说明

一 概述.....	3
二 函数说明.....	4
Port Function.....	4
Port_OpenCom.....	4
Port_OpenTcp.....	6
Port_OpenUsb.....	7
Port_OpenLpt.....	8
Port_OpenPrn.....	9
Port_CloseCom.....	10
Port_CloseTcp.....	11
Port_CloseUsb.....	12
Port_CloseLpt.....	13
Port_ClosePrn.....	14
Port_SetPort.....	15
Port_EnumCom.....	16
Port_EnumLpt.....	17
Port_EnumUsb.....	18
Port_EnumPrn.....	19
Page Function.....	20
PAGE_PageEnter.....	20
PAGE_PagePrint.....	21
PAGE_PageExit.....	22
PAGE_SetPrintArea.....	23
PAGE_DrawString.....	25
PAGE_DrawRect.....	27
PAGE_DrawBarcode.....	28
PAGE_DrawQRCode.....	30
PAGE_DrawBitmap.....	31
Pos Function.....	32
POS_TextOut.....	32
POS_SetBarcode.....	34
POS_SetQRCode.....	36
POS_PrintPicture.....	37
POS_SelfTest.....	38
POS_Query.....	39
POS_RTQuery.....	40
POS_TicketSucceed.....	42
POS_FeedLine.....	43
POS_FeedNLine.....	44
POS_FeedNDot.....	45

POS_SetMotionUnit.....	46
POS_SetLineHeight.....	47
POS_SetRightSpacing.....	48
POS_SetAlign.....	49
POS_Reset.....	51
POS_KickOutDrawer.....	52
POS_CutPaper.....	53
POS_FeedAndCut.....	54
POS_Beep.....	55

一 概述

1 PrinterLibs 是在 Windows 平台用 C++编写的 DLL，DLL 导出 C 风格的函数。

2 PrinterLibs 函数有以下几类

A Port_XXX

以 Port 开头的函数，主要是打开端口，关闭端口，枚举端口。
支持通过串口，并口，USB 口，网口进行打印。

备注：Port_SetPort：该函数可以指定 POS_XXX 系列函数所使用的通讯端口。

B PAGE_XXX

以 PAGE 开头的函数，封装了页模式指令，可以控制打印机以页模式的方式打印。

- ① PAGE_PageEnter 进入页模式
- ② PAGE_SetPrintArea 设置页模式打印区域
- ③ PAGE_DrawXXX 系列函数在指定区域打印
- ④ PAGE_PagePrint 打印整个页面
- ⑤ PAGE_PageExit 退出页模式

备注：

②③可以重复调用

仅支持页模式的机型可以使用这些函数

C POS_XXX

以 POS 开头的函数，主要是封装了 ESC/POS 指令，可以控制打印机打印。

- ① 进纸系列函数可以控制打印机进纸
- ② 设置系列函数可以设置打印的格式等
- ③ 打印系列函数可以打印文本，条码，QR 码，图片等
- ④ 查询系列函数可以查询打印机状态
- ⑤ 其他函数可以控制钱箱、切刀、蜂鸣器等

二 函数说明

Port Function

Port_OpenCom

Syntax

BOOL Port_OpenCom(TCHAR * pName, DWORD dwBaudrate, DWORD dwParity)

Parameters

pName

端口名称。

例如: COM1,COM2,COM3...COM11...

dwBaudrate

波特率

一般取 9600,19200,38400,57600,115200.

需要和打印机波特率保持一致, 建议使用高波特率以获得较好的打印速度

dwParity

效验位

取值如下:

#define PARITY_NONE	((WORD)0x0100)
#define PARITY_ODD	((WORD)0x0200)
#define PARITY_EVEN	((WORD)0x0400)
#define PARITY_MARK	((WORD)0x0800)
#define PARITY_SPACE	((WORD)0x1000)

Return value

如果打开成功, 返回 TRUE。否则, 返回 FALSE

Remarks

如果串口被占用，打开串口会失败。

如果波特率和打印机波特率不匹配，则无法打印。

Port_OpenTcp

Syntax

BOOL Port_OpenTcp(TCHAR * szIp, USHORT nPort)

Parameters

szIp

IP 地址

例如: 192.168.1.87

nPort

端口号

固定值: 9100

Return value

如果打开成功, 返回 TRUE。否则, 返回 FALSE

Remarks

PC 和打印机需要同网段的才可以连接

Port_OpenUsb

Syntax

```
BOOL Port_OpenUsb(TCHAR * pName)
```

Parameters

pName

端口名称。

可以通过 Port_EnumUSB 来得到打印机的名称。

也可以使用任意其他字符串，这时候，如果找到 USB 打印机，会直接打开

Return value

如果打开成功，返回 TRUE。否则，返回 FALSE

Remarks

USB 打印机接到电脑上，如果设备管理器中出现了 USB Printing Support，则可以使用该函数打开。

如果出现的是 Prolific USB-to-Serial Comm Port，则说明这是 USB 虚拟串口，需要使用 Port_OpenCom。

Port_OpenLpt

Syntax

BOOL Port_OpenLpt(TCHAR * pName)

Parameters

pName

端口名称。

例如：LPT1,LPT2,LPT3...

Return value

如果打开成功，返回 TRUE。否则，返回 FALSE

Remarks

并口只有单向通讯，只可写不可读。

一切查询状态的函数，对并口来说均是无效的。

Port_OpenPrn

Syntax

BOOL Port_OpenPrn(TCHAR * pName)

Parameters

pName

打印机名称。

例如: KP80 Printer

Return value

如果打开成功，返回 TRUE。否则，返回 FALSE

Remarks

打开打印机端口。必须使用我们的打印机驱动才可以。

Port_CloseCom

Syntax

VOID Port_CloseCom()

Parameters

Return value

Remarks

关闭端口

Port_CloseTcp

Syntax

VOID Port_CloseTcp()

Parameters

Return value

Remarks

关闭端口

Port_CloseUsb

Syntax

VOID Port_CloseUsb()

Parameters

Return value

Remarks

关闭端口

Port_CloseLpt

Syntax

VOID Port_CloseLpt()

Parameters

Return value

Remarks

关闭端口

Port_ClosePrn

Syntax

VOID Port_ClosePrn()

Parameters

Return value

Remarks

关闭端口

Port_SetPort

Syntax

VOID Port_SetPort(DWORD dwPortType)

Parameters

dwPortType

端口类型。

#define KCPORTYPE_COM	0x1
#define KCPORTYPE_ETH	0x2
#define KCPORTYPE_USB	0x4
#define KCPORTYPE_LPT	0x8
#define KCPORTYPE_PRN	0x10

Return value

Remarks

需要设置端口类型，POS_XXX 系列函数才能正常使用。

Port_EnumCom

枚举串口

Syntax

```
VOID Port_EnumCom(TCHAR * pBuf, int cbBuf, int * pcbNeeded, int * pcnReturned)
```

Parameters

pBuf

枚举到的端口列表。

cbBuf

pBuf 缓冲区字节数

pcbNeeded

需要的缓冲区长度

pcnReturned

返回的端口数目

Return value

Remarks

使用范例代码如下：C++

```
int cbNeeded = 0;
int cnReturned = 0;
Port_EnumCom(NULL, 0, &cbNeeded, &cnReturned);
if (cbNeeded)
{
    TCHAR * pBuf = (TCHAR *)malloc(cbNeeded);
    if (pBuf)
    {
        Port_EnumCom(pBuf, cbNeeded, &cbNeeded, &cnReturned);
        TCHAR * pDevice = pBuf;
        for (int i = 0; i < cnReturned; ++i)
        {
            ComboBox_AddString(hCbx, pDevice);
            pDevice += lstrlen(pDevice) + 1;
        }
        free(pBuf);
    }
}
```


Port_EnumLpt

枚举并口

```
VOID Port_EnumLpt(TCHAR * pBuf, int cbBuf, int * pcbNeeded, int * pcnReturned)
```

Parameters

pBuf

枚举到的端口列表。

cbBuf

pBuf 缓冲区字节数

pcbNeeded

需要的缓冲区长度

pcnReturned

返回的端口数目

Return value

Remarks

使用范例代码如下：C++

```
int cbNeeded = 0;
int cnReturned = 0;
Port_EnumLpt(NULL, 0, &cbNeeded, &cnReturned);
if (cbNeeded)
{
    TCHAR * pBuf = (TCHAR *)malloc(cbNeeded);
    if (pBuf)
    {
        Port_EnumLpt(pBuf, cbNeeded, &cbNeeded, &cnReturned);
        TCHAR * pDevice = pBuf;
        for (int i = 0; i < cnReturned; ++i)
        {
            ComboBox_AddString(hCbx, pDevice);
            pDevice += lstrlen(pDevice) + 1;
        }
        free(pBuf);
    }
}
```

Port_EnumUsb

枚举 USB 端口

```
VOID Port_EnumUsb(TCHAR * pBuf, int cbBuf, int * pcbNeeded, int * pcnReturned)
```

Parameters

pBuf

枚举到的端口列表。

cbBuf

pBuf 缓冲区字节数

pcbNeeded

需要的缓冲区长度

pcnReturned

返回的端口数目

Return value

Remarks

使用范例代码如下：C++

```
int cbNeeded = 0;
int cnReturned = 0;
Port_EnumUsb(NULL, 0, &cbNeeded, &cnReturned);
if (cbNeeded)
{
    TCHAR * pBuf = (TCHAR *)malloc(cbNeeded);
    if (pBuf)
    {
        Port_EnumUsb(pBuf, cbNeeded, &cbNeeded, &cnReturned);
        TCHAR * pDevice = pBuf;
        for (int i = 0; i < cnReturned; ++i)
        {
            ComboBox_AddString(hCbx, pDevice);
            pDevice += lstrlen(pDevice) + 1;
        }
        free(pBuf);
    }
}
```

Port_EnumPrn

枚举打印机列表

```
VOID Port_EnumPrn(TCHAR * pBuf, int cbBuf, int * pcbNeeded, int * pcnReturned)
```

Parameters

pBuf

枚举到的端口列表。

cbBuf

pBuf 缓冲区字节数

pcbNeeded

需要的缓冲区长度

pcnReturned

返回的端口数目

Return value

Remarks

使用范例代码如下： C++

```
int cbNeeded = 0;
int cnReturned = 0;
Port_EnumPrn(NULL, 0, &cbNeeded, &cnReturned);
if (cbNeeded)
{
    TCHAR * pBuf = (TCHAR *)malloc(cbNeeded);
    if (pBuf)
    {
        Port_EnumPrn(pBuf, cbNeeded, &cbNeeded, &cnReturned);
        TCHAR * pDevice = pBuf;
        for (int i = 0; i < cnReturned; ++i)
        {
            ComboBox_AddString(hCbx, pDevice);
            pDevice += lstrlen(pDevice) + 1;
        }
        free(pBuf);
    }
}
```

Page Function

PAGE_PageEnter

选择页模式

Syntax

BOOL PAGE_PageEnter()

Parameters

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

PAGE_PagePrint

页模式下打印页面内容

Syntax

BOOL PAGE_PagePrint()

Parameters

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

PAGE_PageExit

退出页模式

Syntax

BOOL PAGE_PageExit()

Parameters

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

PAGE_SetPrintArea

页模式下设置打印区域

Syntax

BOOL PAGE_SetPrintArea(int left, int top, int right, int bottom, int direction)

Parameters

left
打印区域左上角 x 坐标

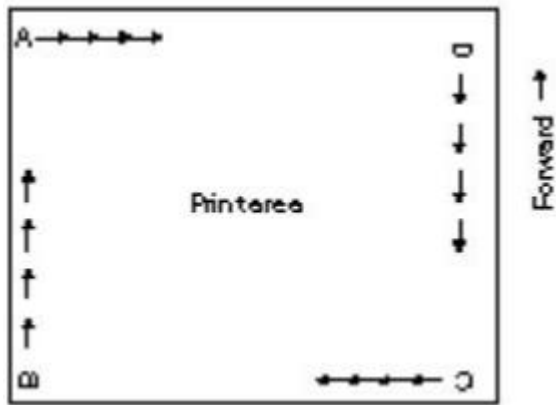
top
打印区域左上角 y 坐标

right
打印区域右下角 x 坐标

bottom
打印区域右下角 y 坐标

direction
打印区域方向

direction	打印方向	起始位置
0	自左向右	左上角(图中的 A)
1	自下向上	左下角(图中的 B)
2	自右向左	右下角(图中的 C)
3	自上向下	右上角(图中的 D)

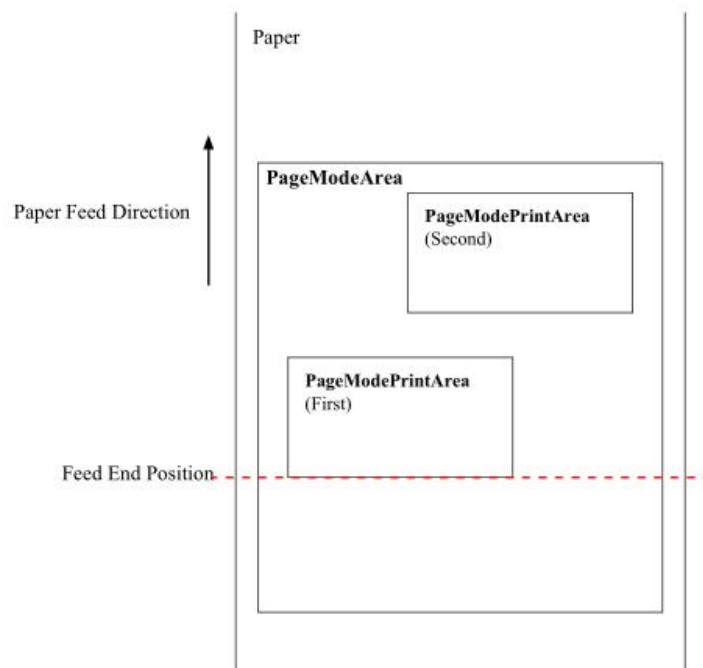


Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

下图展示了打印区域的概念



PAGE_DrawString

画文本

Syntax

BOOL PAGE_DrawString(TCHAR * pszString, int x, int y, int nWidthScale, int nHeightScale, int nFontType, int nFontStyle)

Parameters

pszString

要打印的内容。UNICODE 编码字符串。

x

指定水平方向的起始点位置离打印区域左边界的点数。（横坐标）

支持左对齐，居中，右对齐

传入 x 见下表

x	含义
-1	左对齐
-2	居中对齐
-3	右对齐
大于等于 0	横坐标

y

指定垂直方向的起始点位置离打印区域上边界的点数。（纵坐标）

nWidthScale

指定宽度放大倍数 [0,7]

nHeightScale

指定高度放大倍数 [0,7]

nFontType

字体类型

0 标准字体

1 压缩字体

nFontStyle

指定字体风格，可以为下表中的一个或者若干个（相加即可）

Value	Meaning
0x00	正常

0x08	加粗
0x80	1 点粗的下划线
0x100	2 点粗的下划线
0x200	倒置（只在行首有效）
0x400	反显（黑底白字）
0x1000	每个字符顺时针旋转 90 度

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

页模式 Draw 函数并不立刻打印，只是填在页面中，直到调用了 PAGE_PagePrint，才开始打印。

PAGE_DrawRect

画矩形

Syntax

```
BOOL PAGE_DrawRect(int x, int y, int nWidth, int nHeight, int nColor)
```

Parameters

x

指定水平方向的起始点位置离打印区域左边界的点数。（横坐标）

y

指定垂直方向的起始点位置离打印区域上边界的点数。（纵坐标）

nWidth

指定矩形宽度

nHeight

指定矩形高度

nColor

指定矩形颜色

0 白色

1 黑色

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

如果想画线，只需要把宽度设置为 1（若想画粗一点的线，可设置大一点）即可。

注意：不要画太大区域矩形，否则电源撑不住打印机会复位。

PAGE_DrawBarcode

画条码

Syntax

BOOL PAGE_DrawBarcode(TCHAR * pszBarcodeContent, int x, int y, int nBarcodeUnitWidth, int nBarcodeHeight, int nHriFontType, int nHriFontPosition, int nBarcodeType)

Parameters

pszBarcodeContent

条码内容

x

指定水平方向的起始点位置离打印区域左边界的点数。（横坐标）
支持左对齐，居中，右对齐
传入 x 见下表

x	含义
-1	左对齐
-2	居中对齐
-3	右对齐
大于等于 0	横坐标

y

指定垂直方向的起始点位置离打印区域上边界的点数。（纵坐标）

nBarcodeUnitWidth

指定条码的基本元素宽度。
可以为以下列表中所列值（n）之一。

n	单基本模块宽度 (连续型)	双基本模块宽度（离散型）	
		窄元素宽度	宽元素宽度
2	0. 25mm	0. 25mm	0. 625mm
3	0. 375mm	0. 375mm	1. 0mm
4	0. 5mm	0. 5mm	1. 25mm
5	0. 625mm	0. 625mm	1. 625mm
6	0. 75mm	0. 75mm	1.875mm

nBarcodeHeight

条码高度

nHriFontType

指定 HRI（Human Readable Interpretation）字符的字体类型。
可以为以下列表中所列值之一。

Value	Meaning
0x00	标准 ASCII
0x01	压缩 ASCII

nHriFontPosition

指定 HRI（Human Readable Interpretation）字符的位置。
可以为以下列表中所列值之一。

Value	Meaning
0x00	不打印
0x01	只在条码上方打印
0x02	只在条码下方打印
0x03	条码上、下方都打印

nBarcodeType

可以为以下列表中所列值之一。

Value	Meaning
0x41	UPC-A
0x42	UPC-C
0x43	JAN13(EAN13)
0x44	JAN8(EAN8)
0x45	CODE39
0x46	ITF
0x47	CODEBAR
0x48	CODE93
0x49	CODE 128

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

PAGE_DrawQRCode

画二维码

Syntax

BOOL PAGE_DrawQRCode(TCHAR * pszContent, int x, int y, int nQRCodeUnitWidth, int nVersion, int nEcLevel)

Parameters

pszContent

二维码文本

x

指定水平方向的起始点位置离打印区域左边界的点数。（横坐标）
支持左对齐，居中，右对齐

传入 x 见下表

x	含义
-1	左对齐
-2	居中对齐
-3	右对齐
大于等于 0	横坐标

y

指定垂直方向的起始点位置离打印区域上边界的点数。（纵坐标）

nQRCodeUnitWidth

QR 码单元宽度，范围[1,16]。
QR 码单元宽度越大，QR 码越大。

nVersion

QR 码版本。0 表示自动计算版本。
QR 码版本越大，能编码的字符就越多，QR 码也越大。

nEcLevel

QR 码纠错等级。[1,4]

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

PAGE_DrawBitmap

画 BMP 位图

Syntax

BOOL PAGE_DrawBitmap(TCHAR * FileName, int x, int y, int dwWidth, int dwHeight)

Parameters

FileName

位图文件路径

x

指定水平方向的起始点位置离打印区域左边界的点数。（横坐标）
支持左对齐，居中，右对齐
传入 x 见下表

x	含义
-1	左对齐
-2	居中对齐
-3	右对齐
大于等于 0	横坐标

y

指定垂直方向的起始点位置离打印区域上边界的点数。（纵坐标）

dwWidth

要打印的宽度

dwHeight

要打印的高度

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

Pos Function

POS_TextOut

打印文本

Syntax

BOOL POS_TextOut(TCHAR * pszString, int x, int nWidthScale, int nHeightScale, int nFontType, int nFontStyle)

Parameters

pszString

要打印的内容。UNICODE 编码字符串。

x

指定水平方向的起始点位置离左边界的点数。

nWidthScale

指定宽度放大倍数 [0,7]

nHeightScale

指定高度放大倍数 [0,7]

nFontType

字体类型

0 标准字体

1 压缩字体

nFontStyle

指定字体风格，可以为下表中的一个或者若干个（相加即可）

Value	Meaning
0x00	正常
0x08	加粗
0x80	1 点粗的下划线
0x100	2 点粗的下划线
0x200	倒置（只在行首有效）
0x400	反显（黑底白字）
0x1000	每个字符顺时针旋转 90 度

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_TextOut 并不立刻打印，需要调用 POS_FeedXXX 系列函数，才会把行缓冲区中的内容打印出来。

POS_SetBarcode

打印条码

Syntax

BOOL POS_SetBarcode(TCHAR * pszBarcodeContent, int nBarcodeUnitWidth, int nBarcodeHeight, int nHriFontType, int nHriFontPosition, int nBarcodeType)

Parameters

pszBarcodeContent

条码内容

nBarcodeUnitWidth

指定条码的基本元素宽度。
可以为以下列表中所列值（n）之一。

n	单基本模块宽度 (连续型)	双基本模块宽度（离散型）	
		窄元素宽度	宽元素宽度
2	0. 25mm	0. 25mm	0. 625mm
3	0. 375mm	0. 375mm	1. 0mm
4	0. 5mm	0. 5mm	1. 25mm
5	0. 625mm	0. 625mm	1. 625mm
6	0. 75mm	0. 75mm	1.875mm

nBarcodeHeight

条码高度

nHriFontType

指定 HRI（Human Readable Interpretation）字符的字体类型。
可以为以下列表中所列值之一。

Value	Meaning
0x00	标准 ASCII
0x01	压缩 ASCII

nHriFontPosition

指定 HRI（Human Readable Interpretation）字符的位置。
可以为以下列表中所列值之一。

Value	Meaning
-------	---------

0x00	不打印
0x01	只在条码上方打印
0x02	只在条码下方打印
0x03	条码上、下方都打印

nBarcodeType

可以为以下列表中所列值之一。

Value	Meaning
0x41	UPC-A
0x42	UPC-C
0x43	JAN13(EAN13)
0x44	JAN8(EAN8)
0x45	CODE39
0x46	ITF
0x47	CODEBAR
0x48	CODE93
0x49	CODE 128

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_SetQRCode

打印二维码（QR 码）

Syntax

```
BOOL POS_SetQRCode(TCHAR * pszContent, int nQRCodeUnitWidth, int nVersion, int nEcLevel)
```

Parameters

pszContent

二维码文本

nQRCodeUnitWidth

QR 码单元宽度，范围[1,16]。

QR 码单元宽度越大，QR 码越大。

nVersion

QR 码版本。0 表示自动计算版本。

QR 码版本越大，能编码的字符就越多，QR 码也越大。

nEcLevel

QR 码纠错等级。[1,4]

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_PrintPicture

打印 BMP 位图

Syntax

```
BOOL POS_PrintPicture(TCHAR * FileName, DWORD dwWidth, DWORD dwHeight)
```

Parameters

FileName

位图文件完整路径

dwWidth

要打印的宽度

dwHeight

要打印的高度

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_SelfTest

打印自检页

Syntax

```
BOOL POS_SelfTest()
```

Parameters

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_Query

查询打印机状态

Syntax

BOOL POS_Query(unsigned char status[1])

Parameters

status

打印机状态

该值目前无意义

Return value

如果状态查询成功，返回 TRUE。否则，返回 FALSE

Remarks

该命令返回 TRUE 表明打印机处于可打印状态。

POS_RTQuery

实时查询打印机状态

Syntax

```
BOOL POS_RTQuery(unsigned char status[4])
```

Parameters

status
打印机状态

各字节含义见下表：

status[0]：打印机状态

位	0/ 1	十六进制码	十进制码	功能
0	0	00	0	固定为 0
1	1	02	2	固定为 1
2	0	00	0	一个或两个钱箱打开 （没有钱箱的机器该位固定为零）
	1	04	4	两个钱箱都关闭
3	0	00	0	联机
	1	08	8	脱机
4	1	10	16	固定为 1
5, 6		--	--	未定义
7	0	00	00	纸已撕走
	1	80	96	纸未撕走

status[1]：传送脱机状态

位	0/ 1	十六进制码	十进制码	功能
0	0	00	0	固定为 0
1	1	02	2	固定为 1
2	0	00	0	上盖关
	1	04	4	上盖开
3	0	00	0	未按走纸键
	1	08	8	按下走纸键
4	1	10	16	固定为 1
5	0	00	0	打印机不缺纸

	1	20	32	打印机缺纸
6	0	00	00	没有出错情况
	1	40	64	有错误情况
7	0	00	0	固定为 0

status[2]: 传送错误状态

位	0/ 1	十六进制码	十进制码	功能
0	0	00	0	固定为 0
1	1	02	2	固定为 1
2		--	--	未定义
3	0	00	0	切刀无错误
	1	08	8	切刀有错误
4	1	10	16	固定为 1
5	0	00	0	无不可恢复错误
	1	20	32	有不可恢复错误
6	0	00	00	打印头温度和电压正常
	1	40	64	打印头温度或电压超出范围
7	0	00	0	固定为 0

status[3]: 传送纸传感器状态

位	0/ 1	十六进制码	十进制码	功能
0	0	00	0	固定为 0
1	1	02	2	固定为 1
2, 3	0	00	0	有纸
	1	0C	12	纸将近
4	1	10	16	固定为 1
5, 6	0	00	0	有纸
	1	60	96	纸尽
7	0	00	0	固定为 0

Return value

如果状态查询成功，返回 TRUE。否则，返回 FALSE

Remarks

任何时候打印机收到该命令都会立刻返回。

POS_TicketSucceed

单据打印结果查询

Syntax

```
BOOL POS_TicketSucceed(int dwSendIndex)
```

Parameters

dwSendIndex

单据索引

可以从 1 开始依次递增，目前并无实际意义

Return value

单据打印完成，并且没有因为缺纸而中断，则返回 **TRUE**。

否则，没有查到状态，或返回因为缺纸或其他错误导致打印中断，则返回 **FALSE**。

Remarks

为了保证单据打印的可靠性，请每批次打印任务完成之后，调用一次该函数确认单据打印结果。

POS_FeedLine

Syntax

BOOL POS_FeedLine()

Parameters

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

打印机进纸一行

POS_FeedNLine

Syntax

BOOL POS_FeedNLine(int nLine)

Parameters

nLine

进纸行数

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

进纸 nLine 行

POS_FeedNDot

Syntax

BOOL POS_FeedNDot(int nDot)

Parameters

nDot

进纸点数

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

进纸 nDot 点。一般情况下，1mm 有 8 个点。

POS_SetMotionUnit

设置水平和垂直移动单位

Syntax

```
BOOL POS_SetMotionUnit(int nHorizontal, int nVertical)
```

Parameters

nHorizontal

水平移动单位

nVertical

垂直移动单位

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

当 nHorizontal 和 nVertical 都设置为 200 时，1 点就是 0.125mm。

POS_SetLineHeight

设置行高

Syntax

```
BOOL POS_SetLineHeight(int nDistance)
```

Parameters

nDistance

行高

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_SetRightSpacing

设置字符右边空白

Syntax

```
BOOL POS_SetRightSpacing(int nDistance)
```

Parameters

nDistance

字符右边空白

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_SetAlign

设置对齐方式

Syntax

```
BOOL POS_SetAlign(int nAlign)
```

Parameters

nAlign

对齐方式

- 0 左对齐
- 1 居中对齐
- 2 右对齐

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_Reset

复位打印机。会清空设置。

Syntax

```
BOOL POS_Reset()
```

Parameters

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_KickOutDrawer

打开钱箱

Syntax

```
BOOL POS_KickOutDrawer(int nID, int nOnTimes, int nOffTimes);
```

Parameters

nID

钱箱编号

0 钱箱引脚 2

1 钱箱引脚 5

nOnTimes

钱箱脉冲高电位 ms 时间

nOffTimes

钱箱脉冲低电位 ms 时间

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_CutPaper

直接切纸

Syntax

```
BOOL POS_CutPaper(int nMode)
```

Parameters

nMode

切纸模式

0 全切

1 半切

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_FeedAndCut

打印机进纸 [打印位置到切刀之间距离 + nDistance × (纵向移动单位)] 然后切纸

Syntax

```
BOOL POS_FeedAndCut(int nDistance)
```

Parameters

nDistance
额外进纸距离

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks

POS_Beep

蜂鸣器鸣叫

Syntax

```
BOOL POS_Beep(int nBeepCount, int nBeepMillis)
```

Parameters

nBeepCount

鸣叫次数

nBeepMillis

蜂鸣时间：100ms 为单位

Return value

如果指令写入成功，返回 TRUE。否则，返回 FALSE

Remarks