POSTEK

Android SDK

开发指南

文档版本：**V1.0**

商标声明

POSTEK是深圳市博思得科技发展有限公司的注册商标。

本文档提及的其它所有商标或注册商标，由各自的所有人拥有。

注意!

由于产品版本升级或其它原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

深圳市博思得科技发展有限公司

地址：深圳市南山区 4068 号智慧广场 B 栋 2 座 18 层　邮编：518052

网址：www.postek.com.cn

客户服务电话：(755) 8324 0988

客户服务传真：(755) 8320 2898

客服邮箱：tech@postek.com.cn

前言

概述

本文档用于指导用户在 Android 移动设备上，通过调用应用程序接口，轻松实现与 POSTEK 打印机的交互。

读者对象

本文档主要适用于以下工程师：

- 软件开发工程师
- 技术支持工程师

版本记录

第一次正式发布。

目 录

1 SDK 使用说明

1.1 简介

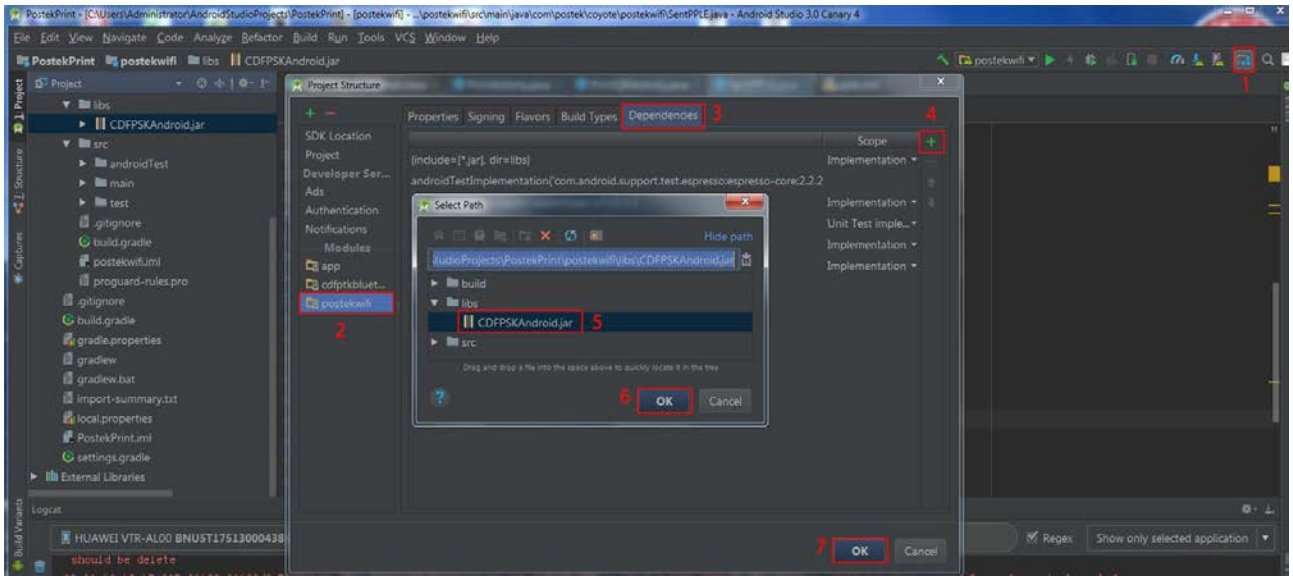
POSTEK Android SDK 为 POSTEK 打印机客户端提供了 API 调用服务，使 Android 应用开发者可以方便快捷的集成打印应用程序。

1.2 配置 SDK

导入 SDK

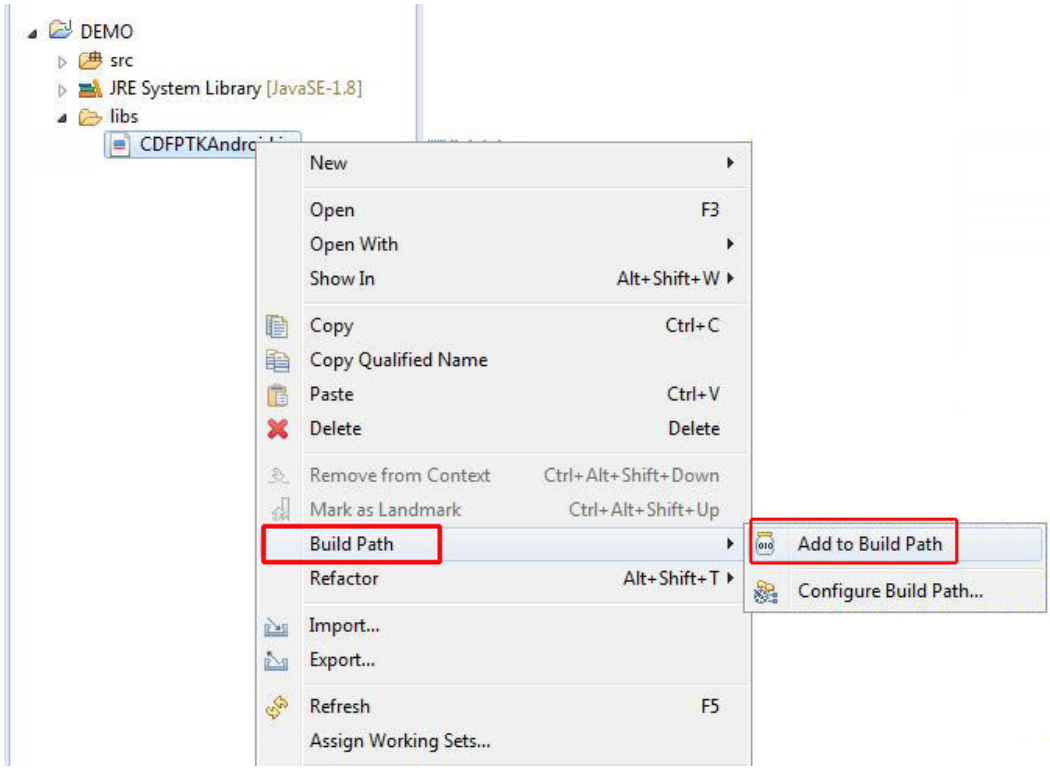
1. 把 CDFPTKAndroid.jar 文件复制到工程的 libs 文件夹中。
2. 导入 jar 包。
   - Studio 开发

     单击，弹出 Project Structure 对话框，选择需要依赖的工程（如 postekwifi），单击“Dependencies”；单击界面右侧的，选择“Jar dependency”，弹出 jar 包的路径选择对话框，选择“libs”-> “CDFPTKAndroid.jar”，单击 OK。



   - Eclipse 开发

     鼠标右键单击 jar 包，在菜单中选择“Build Path”->“Add to Build Path”。



添加权限

使用 WiFi 时，需在清单文件 AndroidManifest.xml 中配置以下权限：
<uses-permission android:name="android. permission.INTERNET"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

使用蓝牙时，需在清单文件 AndroidManifest.xml 中配置以下权限：
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />

注意：Android 6.0 及以上版本须配置位置权限，否则无法搜索到蓝牙设备。
<uses-permissionandroid:name="android.permission.ACCESS_COARSE_LOCATION"/>

CDFPSKAndroid.jar 使用说明

接口：CDFPTKAndroid
实现类：CDFPTKAndroidImpl
实现类构造方法：public CDFPTKAndroidImpl(Context context, Handler handler)；

参数说明：
context：上下文环境。
handler：用于接收返回值。

包中设置的常量：

Message.what: CDFPTKAndroid.PTK_MSG_WHAT_WIFICONNECT
CDFPTKAndroid.PTK_MSG_WHAT_READRFIDMSG

Handler 设置方法：
public void setHandler (Handler handler);
对实现类对象中的某些后台操作方法进行返回值获取。

Jar 包中实现类对象的操作流程：
1. 创建一个 static 全局变量：（单例）
public static CDFPTKAndroid cdf = new CDFPTKAndroidImpl (context, handler);
注意：不允许在整个操作过程中创建新的 CDFPTKAndroid 实例对象，否则将导致连接错误。

2. 通过全局变量 cdf 建立连接：
cdf.PTK_ConnectWiFi (“199.9.10.220”, 9100);

3. 通过全局变量 cdf 调用成员方法：
int nReturn = cdf.PTK_PrintConfiguration ();

1.3 兼容性

POSTEK Android SDK V1.0.0 版本支持 Android 3.5 及以上系统。

2 API 函数说明

2.1 连接 WiFi

public int PTK_ConnectWiFi (final String **IPAddress**, final int **Port**)

功能

该函数的作用是使 Android 移动设备通过 WiFi 与打印机进行连接。

参数

IPAdress：打印机的 IP 地址。

Port：打印机的 TCP 端口号。

返回值

0 --> OK

当 Message.what == CDFPTKAndroid.PTK_MSG_WHAT_WIFICONNECT 时，Message.arg1 的值即为有效的返回值。详见 Demo。

范例

PTK_ConnectWiFi (“199.9.10.220”, 9100);

2.2 断开 WiFi

public int PTK_CloseConnectWiFi ()

功能

该函数的作用是使 Android 移动设备与打印机断开 WiFi 连接。

参数

无

返回值

0 --> OK

范例

PTK_CloseConnectWiFi ();

2.3 连接蓝牙

public int PTK_ConnectBluetooth (String **deviceAddress)**

功能

该函数的作用是使 Android 移动设备通过蓝牙与打印机进行连接。

参数

deviceAddress：打印机的蓝牙地址。

返回值

0 --> OK

范例

PTK_ ConnectBluetooth (deviceAddress);

2.4 断开蓝牙

public int PTK_ DisconnectBluetooth ()

功能

该函数的作用是使Android移动设备与打印机断开蓝牙连接。

参数

无

返回值

0 --> OK

范例

PTK_ DisconnectBluetooth ();

2.5 清除缓存

public int PTK_ClearBuffer ()

功能

该函数的作用是清除打印机中的指令数据缓存。在发送一张新的标签内容到打印机前，建议先使用此函数清空打印机缓存中的数据内容。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_ClearBuffer ();

注意：

*1. 此函数不能在 **FORM** 的编排过程中使用。*

*2. 打印机缓存中的 **FORM** 和图形不会被清空。*

2.6 设置打印速度

public int PTK_SetPrintSpeed (int **px**)

功能

该函数的作用是设置打印机的打印速度。

参数

px：速度取值。

px 参数取值所对应的打印速度，如下表所示。

px	打印速度
20	2.0 ips (50.80 mm/s)
25	2.5 ips (63.50 mm/s)
30	3.0 ips (76.20 mm/s)
35	3.5 ips (88.90 mm/s)
40	4.0 ips (101.60 mm/s)
50	5.0 ips (127.00 mm/s)
60	6.0 ips (152.40 mm/s)
70	7.0 ips (177.80 mm/s)
80	8.0 ips (203.20 mm/s)
90	9.0 ips (228.60 mm/s)
100	10.0 ips (254.00 mm/s)

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_SetPrintSpeed (20);

注意：此函数适用于 **POSTEK** 所有机型。不同型号打印机的最大打印速度不同（具体请查阅对应用户手册），若设置值大于打印机的最大打印速度，则该设置无效。

2.7 设置标签打印方向

public int PTK_SetDirection (String **direct**)

功能

该函数的作用是设置标签的打印方向。

参数

direct：标签的打印方向。

B - 从标签右下角开始打印；

T - 从标签左上角开始打印。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_SetDirection (“B”);

2.8 设置打印黑度

public int PTK_SetDarkness (int **id**)

功能

该函数的作用是设置打印机的打印黑度。

参数

id：打印黑度。取值范围：0～20。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_SetDarkness (10);

2.9 设置标签的高度和定位间隙/黑线/穿孔的高度

public int PTK_SetLabelHeight (int **lHeight**, int **gapH**, int **gapOffset**, boolean **bFlag**)

功能

该函数的作用是设置标签的高度和定位间隙\黑线\穿孔的高度。

参数

lHeight：标签的高度，以点(dots)为单位。取值范围：0～65535。

gapH： 标签的定位间隙/黑线/穿孔的高度，以点(dots)为单位。取值范围： 0～65535。当 gapH 值为 0 时，纸张探测器只检测纸张是否用尽。

gapOffset：标签间隙/黑线/穿孔的定位偏移值，以点(dots)为单位。

bFlag：间隙偏移是否有效。

True - 有效；

False - 无效。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_SetLabelHeight (160, 24, 0, false);

2.10 设置标签宽度

public int PTK_SetLabelWidth (int **lWidth**)

功能

该函数的作用是设置标签的宽度。

参数

lWidth：标签的宽度，以点(dots)为单位。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_SetLabelWidth (250);
```

2.11 执行打印任务

public int PTK_PrintLabel (int **number**, int **cpnumber**)

功能

该函数的作用是使打印机执行打印任务。

参数

number：打印标签的数量。取值范围：1～65535。

cpnumber：每张标签的复制份数。取值范围：1～65535。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_PrintLabel (2, 3);
```

注意：此函数不能在 FORM 的编排过程中使用。

2.12 自动执行打印任务

public int PTK_PrintLabelAuto (int **number**, int **cpnumber**)

功能

该函数的作用是使打印机自动执行打印任务。当 FORM 中存在变量或者序列号时，用户输入初始值后，打印机将自动执行打印功能。

参数

number：打印标签的数量。取值范围：1～65535。

cpnumber：每张标签的复制份数。取值范围：1～65535。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_FormDel ("LSFORM");                          //删除 Form "TEST"
PTK_FormDownload ("LSFORM");                     //存储 Form "TEST"
PTK_DefineCounter (0,10,78,"+1","InputC0");      //定义一个序列号变量 C0
PTK_DefineCounter (1,10,78,"+1"," InputC1");     //定义一个序列号变量 C1
```

```
PTK_DefineVariable (0,16,78," InputV0");                    //定义变量字符串 V0
PTK_DrawBarcode (100,100,0,"3",2,6,30,66,"C0");             //打印序列号条码
PTK_DrawText  (100,160,0,2,1,1,78,"\"条码内容为：\"C0");    //打印一行文本文字（字符串中有常量和变量 C0）

PTK_DrawText (100,220,0,2,1,1,78,"\"打印序列号：\"C1"); //打印序列号数值 C1
PTK_DrawText (100,280,0,2,1,1,78,"\"打印变量：\"V0");   //打印变量字符串 V0
PTK_DrawText (100,340,0,2,1,1,78,"\"组合功能：序号:\"C1\"变量:\"V0"); //组合打印

PTK_FormEnd ();                                             //结束存储表格
PTK_ExecForm ("LSFORM");                                    //运行指定的表格
PTK_Download ();                                            //下载变量或序列号变量

按已定义的顺序初始化序号及变量（此例定义顺序为 C0，C1，V0）
PTK_DownloadInitVar ("12345678");                           //初始化序列号变量 C0
PTK_DownloadInitVar ("123456");                             //初始化序列号变量 C1
PTK_DownloadInitVar ("1111");                               //初始化变量 V0
PTK_PrintLabelAuto (2, 3);                                  //打印机自动执行打印任务
```

注意：此函数只能在FORM中使用。

2.13 打印软字体名称清单

public int PTK_SoftFontList ()

功能

该函数的作用是打印存储在打印机中的软字体名称清单。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_SoftFontList ();
```

2.14 删除软字体

public int PTK_SoftFontDel (char **id**)

功能

该函数的作用是删除存储在打印机中的一个或所有软字体。

参数

id：软字体 ID，取值范围：A～Z 或 * 。如果 id = ‘ * ’，则存储在打印机中的所有软字体将被删除。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_SoftFontDel (‘A’);
```

2.15 开启撕纸功能

public int PTK_EnableBackFeed ()

功能

该函数的作用是开启打印机的撕纸功能。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_EnableBackFeed ();
```

注意：此函数不适用于带 DIP 开关或 LCD 屏的打印机机型！

2.16 关闭撕纸功能

public int PTK_DisableBackFeed ()

功能

该函数的作用是关闭打印机的撕纸功能。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_DisableBackFeed ();

注意：此函数不适用于带 DIP 开关或 LCD 屏的打印机机型！

2.17 打印配置信息

public int PTK_PrintConfiguration ()

功能

该函数的作用是使打印机打印当前配置信息。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_PrintConfiguration ();

2.18 设置工作状态

public int PTK_SetPrinterState (char **state**)

功能

该函数的作用是设置打印机的工作状态。

参数

state：打印机的工作状态。

D - 热感印（热传导）状态；

P - 连续送纸状态（默认）；

L - 设置打印机打印一张标签后，暂停等待用户确定后再打印下一张标签；
若打印机未安装剥纸器，用户按一次“FEED”键，再打印下一张标签；
若打印机已安装剥纸器，当用户取走标签后，自动打印下一张标签。

C - 切纸状态；

N - 剥纸状态。（不能与 C 值同时设置）

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_SetPrinterState ('D');

注意：此函数不适用于带 DIP 开关或 LCD 屏的打印机机型！

2.19 选择 FLASH 存储

public int PTK_EnableFLASH ()

功能

该函数的作用是选择FLASH存储器进行存储。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_EnableFLASH ();

2.20 取消 FLASH 存储

public int PTK_DisableFLASH ()

功能

该函数的作用是取消选择FLASH存储器进行存储，此时发送到打印机的数据将存储到SDRAM存储器。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_ DisableFLASH ();

2.21 走纸

public int PTK_FeedMedia ()

功能

该函数的作用是使打印机走一张空白标签。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_ FeedMedia ();

2.22 纸张定位校准

public int PTK_MediaDetect ()

功能

该函数的作用是使打印机对标签纸进行定位校准。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_ MediaDetect ();

2.23 复位

public int PTK_Reset ()

功能

该函数的作用是使打印机复位。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_Reset ();
```

注意：

1. ***此函数在以下情况下无效：***
   - ***当下载软字体、PCX 图形或打印机在 DUMP 状态时；***
   - ***当在 FORM 中使用此函数时；***
   - ***当打印机正在执行打印任务时。***
2. ***打印机初始化的过程可能持续2秒钟以上，此期间打印机不接收任何控制指令。***

2.24 设置切刀频率（重启有效）

public int PTK_CutPage (int **page**)

功能

该函数的作用是设置打印机切刀的工作频率，即切刀有规律地每打印一定数量标签后，切一次纸。

参数

page：打印标签的页数。取值范围：1～999。

返回值

0 -->OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_CutPage (1);
```

注意：此函数设置在重启打印机后仍有效。

2.25 设置切刀频率（重启无效）

public int PTK_CutPageEx (int **page**)

功能

该函数的作用是设置打印机切刀的工作频率，即切刀有规律地每打印一定数量的标签后，切一次纸。

参数

page：打印标签的页数。取值范围：1-999。

返回值

0 -->OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_CutPageEx (1);

注意：此函数设置在重启打印机后无效。

2.26 反馈错误报告

public int PTK_FeedBack ()

功能

该函数的作用是使打印机反馈错误报告。调用此函数后，打印机将传回 4 个字节数据（0xXX XX 0x0d 0x0a）到通讯端口，用户可立即确定打印机的当前错误状态。

0xXX 的取值所对应的错误状态，如下表所示：

状态代码（0xXX）	错误状态
00	无错误
01	语法错误
82	碳带探测出错
83	标签探测出错
86	切刀检测出错
87	打印头未关闭
88	暂停状态
99	其它错误

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_FeedBack ();

2.27 设置网络反馈参数

public int PTK_SetNetworkFeedbackParameter (int **port**, String **ip**, int **cp**)

功能

该函数的作用是设置网络反馈参数。

参数

port：端口号，取值范围为：0～65535。

ip：目标主机的 IP 地址（此 IP 地址不能与打印机的 IP 地址相同）。

cp：通信协议。

0 - UDP；

1 - TCP。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_SetNetworkFeedbackParameter (9200, “199.9.10.288”, 1);
PTK_FeedBack ();
```

注意：

1. ***此函数需与 PTK_FeedBack ()配合使用。***
2. ***在开机状态下，若网络环境不变，则只需配置一次即可；在关机或断电情况下，该网络配置失效。***

2.28 设置反馈端口

public int PTK_SetFeedbackPort (int **port**)

功能

该函数的作用是设置反馈端口。

参数

port：反馈端口。

0- 串口；

1- USB；

2 - lan (NET)。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例 1（设置网络反馈端口）

```
PTK_SetFeedbackPort (2);
PTK_FeedBack ();
```

范例 2（修改网络反馈端口 IP 地址）

```
PTK_SetNetworkFeedbackParameter (9200, “199.9.10.288”, 1);
PTK_SetFeedbackPort (2);
PTK_FeedBack ();
```

注意：

1. *此函数需与 **PTK_FeedBack ()**配合使用。*
2. *此函数仅适用于配有 **WiFi**模块的打印机机型，且反馈端口需设置为网络端口。*

2.29 打印一行文本文字（内部字体）

public int PTK_DrawText (int **px**, int **py**, int **pdirec**, char **pFont**, int **pHorizontal**, int **pVertical**, char **ptext**, String **pstr**)

功能

该函数的作用是通过调用打印机内部字体（内置字体或软字体）打印一行文本文字。内容可以是常量、序列号、变量或组合字符串。

参数

px：X 坐标，以点(dots)为单位。

py：Y 坐标，以点(dots)为单位。

pdirec：旋转角度。

0 - 0°；

1 - 90°；

2 - 180°；

3 - 270°。

pFont：内置字体或软字体。

1～5 - 打印机内置 5 种西文字体；

6 - 打印机内置 1 种中文字体；

若打印机为非 Postek V 系列机型，则 6 表示打印机内置的 24*24 简体汉字；

若打印机为 Postek V 系列机型，则 6 表示打印机内置的黑体。

‘A’～‘Z’ - 下载的软字体。

pHorizontal：当 pFont 设置为内置字体（1～6）时，pHorizontal 为设置点阵的水平放大系数。

当 pFont 设置为下载的软字体（A～Z）时，pHorizontal 为设置字体的宽度，大小不限。

pVertical：当 pFont 设置为内置字体（1～6）时，pVertical 为设置点阵的垂直放大系数。

当 pFont 设置为下载的软字体（A～Z）时，pVertical 为设置字体的高度，大小不限。

ptext：文本的类型。

'N' - 打印正常文本（如黑字白底文本）；

'R' - 打印反色文本（如白字黑底文本）。

pstr：一个长度为 1-100 的字符串。用户可以用"DATA", Cn, Vn 自由排列组合成一个组合字符串，如："data1"CnVn"data2"。

"DATA"：常量字符串，必须用""" 作为起始和结束符号，如"POSTEK Printer"。

Cn：序列号数值，此序列号必须已经定义，n 为序列号 ID。请参考 PTK_DefineCounter。

Vn：变量字符串，此变量字符串必须已经定义，N 为变量 ID 号码。请参考 PTK_DefineVariable。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawText (50,30,0,2,1,1,'N',"\"123456789\"C0");
PTK_DrawText (50,30,0,2,1,1,'N',"\" 123456789\"");
PTK_DrawText (50,30,0,2,1,1,'N',"C1");
PTK_DrawText (50,30,0,2,1,1,'N',"V3");
PTK_DrawText (50,30,0,2,1,1,'N', "\"Printer\" C2V1\"is ok.\ " ");

打印序列号和变量字符串（一般在 Form 中使用）
PTK_FormDel ("LSFORM");                                    //删除 Form "TEST"
PTK_FormDownload ("LSFORM");                               //存储 Form "TEST"
PTK_DefineCounter (0,10,78,"+1","InputC0");                //定义一个序列号变量 C0
PTK_DefineCounter (1,10,78,"+1"," InputC1");               //定义一个序列号变量 C1
PTK_DefineVariable (0,16,78," InputV0");                   //定义变量字符串 V0
PTK_DrawBarcode (100,100,0,"3",2,6,30,66,"C0");            //打印序列号条码
PTK_DrawText (100,160,0,2,1,1,78,"\"条码内容为：\"C0");    //打印一行文本文字（字符
                                                             串中有常量和变量 C0）

PTK_DrawText (100,220,0,2,1,1,78,"\"打印序列号：\"C1");     //打印序列号数值 C1
PTK_DrawText (100,280,0,2,1,1,78,"\"打印变量：\"V0");       //打印变量字符串 V0
PTK_DrawText (100,340,0,2,1,1,78,"\"组合功能：序号:\"C1\"变量:\"V0");   //组合打印
PTK_FormEnd ();                                            //结束存储表格

PTK_ExecForm ("LSFORM");                                   //运行指定的表格
PTK_Download ();                                           //下载变量或序列号变量

按已定义的顺序初始化序号及变量（此例定义顺序为 C0，C1，V0）
PTK_DownloadInitVar ("12345678");                          //初始化序列号变量 C0
PTK_DownloadInitVar ("123456");                            //初始化序列号变量 C1
PTK_DownloadInitVar ("1111");                              //初始化变量 V0
```

```
PTK_PrintLabel(2,1);                                    //命令打印机执行打印任务
```

2.30 打印一行文本文字（Android 字体）

public abstract int PTK_DrawAndroidText (int **px**, int **py**, int **pSize**, int **pType**, String **ptext**,
String **textName**, String **pstr**)

功能

该函数的作用是通过调用 Android 字体打印一行文本文字。文本内容将被存储为 bitmap 格式图形，最终通过转化为 PCX 格式图形打印出来。

参数

px: 起始点的 X 坐标，以点(dots)为单位。
py: 起始点的 Y 坐标，以点(dots)为单位。
pSize: 设置字体的大小。
pType: 设置字体的属性（默认为黑体）。
1 - 粗体；
2 - 粗斜体；
3 - 斜体；
4 - 常规。
ptext: 文本的类型。
'N' - 打印正常文本（如黑字白底文本）；
'R' - 打印反色文本（如白字黑底文本）。
textName: 自定义字体的名称。
pstr: 文本内容。

返回值

0 --> OK

范例

```
PTK_DrawAndroidText(0,0,36,4,"N","postek","博学而笃志");
PTK_DrawAndroidText(0,50,36,4,"R","postek1","切问而近思");
PTK_PrintLabel (1,1);
PTK_PcxGraphicsDel ("postek");
PTK_PcxGraphicsDel ("postek1");
```

说明

如果以上字体属性无法满足您的需求，请参考 Jar 中的源码，添加其它字体属性：
1、常用的字体类型

```
Paint mp = new paint ();
mp.setTypeface (Typeface.DEFAULT_BOLD)

 * Typeface.DEFAULT                    //常规字体类型
 * Typeface.DEFAULT_BOLD               //黑体字体类型
```

```
* Typeface.MONOSPACE          //等宽字体类型
* Typeface.SANS_SERIF         //sans serif 字体类型
* Typeface.SERIF              //serif 字体类型
```

2、其它属性设置

```
Paint mp = new Paint ();
mp.setTextScaleX(Float scaleX);     //调节字间距
mp.setFakeBoldText(true);           //true 为粗体，false 为非粗体
mp.setTextSkewX(-0.5f);             //float 类型参数，负数表示右斜，整数左斜
mp.setUnderlineText(true);          //true 为下划线，false 为非下划线
mp.setStrikeThruText(true);         //true 为删除线，false 为非删除线
```

2.31 打印一维条形码

public int PTK_DrawBarcode (int **px**, int **py**, int **pdirec**, String **pCode**, int **NarrowWidth**, int **pHorizontal**, int **pVertical**, char **ptext**, String **pstr**)

功能

该函数的作用是打印一个一维条形码。内容可以是常量、序列号、变量或组合字符串。

参数

px：X 坐标，以点(dots)为单位。

py：Y 坐标，以点(dots)为单位。

pdirec：旋转角度。

0 - 0°；

1 - 90°；

2 - 180°；

3 - 270°。

pCode：一维条形码的类型。（条形码有字符或字符个数限制，请参考具体标准。）

pcode 参数取值所对应的一维条形码类型，如下表所示：

pCode	一维条形码类型
0	Code 128 UCC (shipping container code)
1	Code 128 AUTO
1A	Code 128 subset A
1B	Code 128 subset B
1C	Code 128 subset C
1E	UCC/EAN
2	Interleaved 2 of 5
2C	Interleaved 2 of 5 with check sum digit
2D	Interleaved 2 of 5 with human readable check digit
2G	German Postcode
2M	Matrix 2 of 5
2U	UPC Interleaved 2 of 5

3	Code 3 of 9
3C	Code 3 of 9 with check sum digit
3E	Extended Code 3 of 9
3F	Extended Code 3 of 9 with check sum digit
9	Code93
E30	EAN-13
E32	EAN-13 2 digit add-on
E35	EAN-13 5 digit add-on
E80	EAN-8
E82	EAN-8 2 digit add-on
E-85	EAN-8 5 digit add-on
K	Codabar
P	Postnet
UA0	UPC-A
UA2	UPC-A 2 digit add-on
UA5	UPC-A 5 digit add-on
UE0	UPC-E
UE2	UPC-E 2 digit add-on
UE5	UPC-E 5 digit add-on

NarrowWidth：一维条形码中窄单元的宽度，以点(dots)为单位。
pHorizontal：一维条形码中宽单元的宽度，以点(dots)为单位。
pVertical：一维条形码的高度，以点(dots)为单位。
ptext：供人识别字符。
‘N’－ 隐藏条形码下方的文字；
‘B’－ 显示条形码下方的文字。
pstr：一个长度为 1-100 的字符串。用户可以用”DATA”, Cn, Vn 自由排列组合成一个组合字符串，如：”data1”CnVn”data2”。
“DATA”：常量字符串，必须用“”” 作为起始和结束符号，如”POSTEK Printer"。
Cn：序列号数值，此序列号必须已经定义，n 为序列号 ID。请参考 PTK_DefineCounter。
Vn：变量字符串，此变量字符串必须已经定义，N 为变量 ID 号码。请参考 PTK_DefineVariable。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawBarcode (50,30,0,”1A”,1,1,10,’N’,”\”123456\””);
PTK_PrintLabel (1, 1);
```

2.32 打印 QR 码

public int PTK_DrawBar2D_QR (int **x**, int **y**, int **w**, int **v**, int **o**, int **r**, int **m**, int **g**, int **s**, String **pstr**)

功能

该函数的作用是打印一个 QR 码。内容可以是常量、序列号、变量或组合字符串。

参数

X：X 坐标，以点(dots)为单位。
y：Y 坐标，以点(dots)为单位。
w：最大打印宽度，以点(dots)为单位。
v：最大打印高度，以点(dots)为单位。
o：旋转角度。
0 - 0°
1 - 90°
2 - 180°
3 - 270°
r：放大倍数。取值范围：1～99。
m：QR 码编码模式。
0 - 数字模式
1 - 数字字母模式
2 - 字节模式 0～256
3 - 中国汉字模式
4 - 混合模式
g：QR 码纠错等级。
0 - L 级
1 - M 级
2 - Q1 级
3 - H1 级
s：QR 码掩模图形。
0 - 掩模图形 000
1 - 掩模图形 001
2 - 掩模图形 010
3 - 掩模图形 011
4 - 掩模图形 100
5 - 掩模图形 101
6 - 掩模图形 110
7 - 掩模图形 111
8 - 自动选择掩模图形
pstr：内容字符串。用户可以用”DATA”，Cn，Vn 自由排列组合成一个组合字符串，如：”data1”CnVn”data2”。
“DATA”：常量字符串，必须用“””作为起始和结束符号，如”POSTEK Printer”。
Cn：序列号数值，此序列号必须已经定义，n 为序列号 ID。请参考 PTK_DefineCounter。

Vn：变量字符串，此变量字符串必须已经定义，N 为变量 ID 号码。请参考 PTK_DefineVariable。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawBar2D_QR (50, 30, 180, 180, 0, 0, 1, 0, 8, "\"123456789\"");
PTK_PrintLabel (1, 1);
```

2.33 打印 DataMatrix 码

public int PTK_DrawBar2D_DATAMATRIX (int **x**, int **y**, int **w**, int **v**, int **o**, int **m**, String **pstr**)

功能

该函数的作用是打印一个DataMatrix码。内容可以是常量、序列号、变量或组合字符串。

参数

x：X 坐标，以点(dots)为单位。

y：Y 坐标，以点(dots)为单位。

v：最大打印高度，以点(dots)为单位。

o：旋转角度。

0 - 0°

1 - 90°

2 - 180°

3 - 270°

m：放大倍数。取值范围：1～9。

pstr：内容字符串。用户可以用"DATA"，Cn，Vn 自由排列组合成一个组合字符串，如："data1"CnVn"data2"。

"DATA"：常量字符串，必须用""" 作为起始和结束符号，如"POSTEK Printer"。

Cn：序列号数值，此序列号必须已经定义，n 为序列号 ID。请参考 PTK_DefineCounter。

Vn：变量字符串，此变量字符串必须已经定义，N 为变量 ID 号码。请参考 PTK_DefineVariable。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawBar2D_DATAMATRIX (50, 30, 0, 0, 0, 5,"\"123456789\"");
PTK_PrintLabel (1, 1);
```

2.34 打印 MaxiCode 码

public int PTK_DrawBar2D_MaxiCode (int **x**, int **y**, int **m**, int **u**, String **pstr**)

功能

该函数的作用是打印一个MaxiCode码。内容可以是常量、序列号、变量或组合字符串。

参数

x：X 坐标，以点(dots)为单位。
y：Y 坐标，以点(dots)为单位。
m：符号体系模式。
2 - 结构化载体信息；
3 - 结构化载体信息；
4 - 标准符号。
u：是否为 UPS 格式。
1 - UPS 格式数据；
0 - 非 UPS 格式数据。
pstr：内容字符串。用户可以用”DATA”，Cn，Vn 自由排列组合成一个组合字符串，如：”data1”CnVn”data2”。
“DATA”：常量字符串，必须用“”” 作为起始和结束符号，如”POSTEK Printer”。
Cn：序列号数值，此序列号必须已经定义，n 为序列号 ID。请参考 PTK_DefineCounter。
Vn：变量字符串，此变量字符串必须已经定义，N 为变量 ID 号码。请参考 PTK_DefineVariable。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawBar2D_MaxiCode(50,30,4,1,”\”1Z000A7&dajc_iaj-3=+～#^$5\””);
PTK_PrintLabel (1, 1);
```

2.35 打印 PDF417 码

public int PTK_DrawBar2D_Pdf417(int **x**, int **y**, int **w**, int **v**, int **s**, int **c**, int **px**,
int **py**, int **r**, int **l**, int **t**, int **o**, String **pstr**)

功能

该函数的作用是打印一个PDF417码。内容可以是常量、序列号、变量或组合字符串。

参数

x：X 坐标，以点(dots)为单位。

y：Y 坐标，以点(dots)为单位。
w：最大打印宽度，以点(dots)为单位。
v：最大打印高度，以点(dots)为单位。
s：错误校正等级。取值范围：0～8。
c：资料压缩等级，可选择 0 或 1。
px：模组宽度，以点(dots)为单位。取值范围：2～9。
py：模组高度，以点(dots)为单位。取值范围：4～99。
r：最大行数。
l：最大列数。
t：截取标志。
0 - 不截取；
1 - 截取。
o：旋转角度。
0 - 0°
1 - 90°
2 - 180°
3 - 270°
pstr：内容字符串。用户可以用”DATA”，Cn，Vn 自由排列组合成一个组合字符串，如：”data1”CnVn”data2”。
“DATA”：常量字符串，必须用“””作为起始和结束符号，如”POSTEK Printer”。
Cn：序列号数值，此序列号必须已经定义，n 为序列号 ID。请参考 PTK_DefineCounter。
Vn：变量字符串，此变量字符串必须已经定义，N 为变量 ID 号码。请参考 PTK_DefineVariable。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawBar2D_Pdf417 (10,10,400,300,0,0,3,7,10,2,0,0,”\”POSTEKINFO\””);
PTK_PrintLabel (1, 1);
```

2.36 打印汉信码

public int PTK_DrawBar2D_HANXIN (int **x**, int **y**, int **w**, int **v**, int **m**, int **o**,
int **r**, int **g**, int **s**, String **pstr**)

功能

该函数的作用是打印一个汉信码。内容可以是常量、序列号、变量或组合字符串。

参数

x：X 坐标，以点(dots)为单位。
y：Y 坐标，以点(dots)为单位。

w：最大打印宽度，以点(dots)为单位。

v：最大打印高度，以点(dots)为单位。

o：旋转角度。

0 - 0°

1 - 90°

2 - 180°

3 - 270°

r：放大倍数。取值范围：0～30。

m：汉信码编码模式。

0 - 数字模式

1 - Text 模式

2 - 二进制字节模式

3 - 常用汉字 1 区模式

4 - 常用汉字 2 区模式

5 - GB 18030 双字节区模式

6 - GB 18030 四字节区模式

g：汉信码纠错等级。

0 - L1 级

1 - L2 级

2 - L3 级

3 - L4 级

s：汉信码掩模图形。

0 - 掩模图形 00

1 - 掩模图形 01

2 - 掩模图形 10

3 - 掩模图形 11

pstr：内容字符串。用户可以用”DATA”，Cn，Vn 自由排列组合成一个组合字符串，如：”data1”CnVn”data2”。

“DATA”：常量字符串，必须用“”” 作为起始和结束符号，如”POSTEK Printer”。

Cn：序列号数值，此序列号必须已经定义，n 为序列号 ID。请参考 PTK_DefineCounter。

Vn：变量字符串，此变量字符串必须已经定义，N 为变量 ID 号码。请参考 PTK_DefineVariable。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawBar2D_HANXIN (50, 30, 0, 0, 0, 5, 1, 3, 2,”\”POSTEK\””);
PTK_PrintLabel (1, 1);
```

2.37 画矩形

public int PTK_DrawRectangle (int **px**, int **py**, int **thickness**, int **pEx**, int **pEy**)

功能

该函数的作用是画距形。

参数

px：起始点的 X 坐标，以点(dots)为单位。
py：起始点的 Y 坐标，以点(dots)为单位。
thickness：边框的粗细，以点(dots)为单位。
pEx：终止点的 X 坐标，以点(dots)为单位。
pEy：终止点的 Y 坐标，以点(dots)为单位。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawRectangle (50,120,5,250,150);
PTK_PrintLabel (1, 1);
```

2.38 画直线（相交处“异或”处理）

public int PTK_DrawLineXor (int **px**, int **py**, int **pbyte**, int **pH**)

功能

该函数的作用是画直线（两直线相交处作“异或”处理）。

参数

px：X 坐标，以点(dots)为单位。
py：Y 坐标，以点(dots)为单位。
pbyte：直线的水平长度，以点(dots)为单位。
pH：直线的垂直高度，以点(dots)为单位。

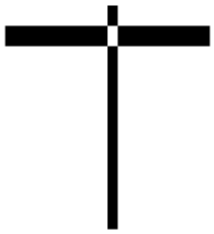
返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawLineXor (100,20,5,110);
PTK_DrawLineXor (50,30,100,10);
PTK_PrintLabel (1, 1);
```

打印效果，如下图所示：



2.39 画直线（相交处“或”处理）

public int PTK_DrawLineOr (int **px**, int **py**, int **plength**, int **pH**)

功能

该函数的作用是画直线（两直线相交处作“或”处理）。

参数

px: X 坐标，以点(dots)为单位。

py: Y 坐标，以点(dots)为单位。

plength：直线的水平长度，以点(dots)为单位。

pH：直线的垂直高度，以点(dots)为单位。

返回值

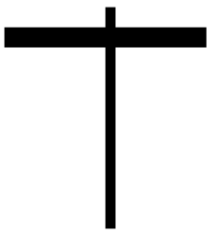
0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawLineOr (100,20,5,110);
PTK_DrawLineOr (50,30,100,10);
PTK_PrintLabel (1, 1);
```

打印效果，如下图所示：



2.40 画斜线

public int PTK_DrawDiagonal (int **px**, int **py**, int **thickness**, int **pEx**, int **pEy**)

功能

该函数的作用是画斜线。

参数

px：斜线起始点的 X 坐标，以点(dots)为单位。
py：斜线起始点的 Y 坐标，以点(dots)为单位。
thickness：斜线的粗细，以点(dots)为单位。
pEx：斜线终止点的 X 坐标，以点(dots)为单位。
pEy：斜线终止点的 Y 坐标，以点(dots)为单位。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawDiagonal (50, 30, 10, 100, 80);
PTK_PrintLabel (1, 1);
```

2.41 画白色直线

public int PTK_DrawWhiteLine (int **px**, int **py**, int **plength**, int **pH**)

功能

该函数的作用是画白色直线。

参数

px：X 坐标，以点(dots)为单位。
py：Y 坐标，以点(dots)为单位。
plength：直线的水平长度，以点(dots)为单位。
pH：直线的垂直高度，以点(dots)为单位。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DrawWhiteLine (100, 20, 5, 110);
PTK_PrintLabel (1, 1);
```

2.42 打印 PCX 格式图形名称清单

public int PTK_PcxGraphicsList ()

功能

该函数的作用是打印存储在打印机中的PCX格式图形名称清单。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_PcxGraphicsList ();

2.43 删除 PCX 格式图形

public int PTK_PcxGraphicsDel (String **pid**)

功能

该函数的作用是删除存储在打印机中的一个或所有PCX格式图形。

参数

pid：待删除的图形名称，最大长度为 16 个字符。

如果 pid = “*”，则存储在打印机中的所有 PCX 格式图形将被删除。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_PcxGraphicsDel (“PCX2”);

2.44 存储一个 PCX 格式图形

public int PTK_PcxGraphicsDownload (String **pcxname**, byte[] **pcxfile**)

功能

该函数的作用是存储一个PCX格式图形到打印机。

参数

pcxname：自定义图形的名称，最大长度为 16 个字符。

pcxfile：PCX 图形在 android 平台下生成的 byte[]数据流。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
InputStream is =getResources().openRawResource(R.raw.luf);
 byte[] pcxFile=null;
  try {
        pcxFile = new byte[is.available()];
       is.read(pcxFile);
  }catch (Exception e){
        try {
            if(is !=null)
                is.close();
      }
      catch (Exception ex) { }
  }
cdf.PTK_EnableFLASH();
      cdf.PTK_PcxGraphicsDownload("luf",pcxFile);
```

2.45 打印指定图形

public int PTK_DrawPcxGraphics (int **px**, int **py**, String **gname**)

功能

该函数的作用是打印存储在打印机中的图形。您可以通过 PTK_PcxGraphicsDownload()将图形存储到打印机中。

参数

px：X 坐标，以点(dots)为单位。

py：Y 坐标，以点(dots)为单位。

gname：待打印图形的名称，最大长度为 16 个字符。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
InputStream is =getResources().openRawResource(R.raw.luf);
        byte[] pcxFile=null;
        try {
            pcxFile = new byte[is.available()];
            is.read(pcxFile);
        }catch (Exception e){
            try {
                if(is !=null)
```

```
                    is.close();
                }
                catch (Exception ex) { }
            }
              cdf.PTK_EnableFLASH();
             cdf.PTK_PcxGraphicsDownload("luf",pcxFile);
             cdf.PTK_DrawPcxGraphics(0,0,"luf");
             cdf.PTK_PrintLabel(1,1);
            cdf.PTK_PcxGraphicsDel("lufei");
```

2.46 打印一个 PCX 格式图形

public int PTK_PrintPCX (int **px**, int **py**, String **pcxname**, byte[] **pcxfile**)

功能

该函数的作用是打印一个PCX格式图形。该函数等同于PTK_PcxGraphicsDownload()和PTK_DrawPcxGraphics()同时使用。

参数

px：X 坐标，以点(dots)为单位。
py：Y 坐标，以点(dots)为单位。
pcxname：PCX 图形的名称。
pcxfile：PCX 图形在 android 平台下生成的 byte[]数据流。

返回值

0 --> OK

范例

```
PTK_PrintLable (1,1)
PTK_PrintPCX (10, 100, "pcx1", pcxfile);
```

2.47 存储一个 BMP 格式图形

public int PTK_BmpGraphicsDownload (String **pcxname**, Bitmap **bmp**, int **iDire**)

功能

该函数的作用是存储一个 BMP 格式图形到打印机。BMP 格式图形将会被转换为 PCX 格式存储到打印机中。

参数

pcxname：自定义图形的名称，最大长度为 16 个字符。
bitmap: 需要下载到打印机中的 BMP 图形。
iDire : 旋转角度。

0 – 0°
1 – 90°
2 – 180°
3 – 270°

返回值

0 --> OK

范例

```
PTK_BmpGraphicsDownload (“BMPA”, bitmap, 0);
PTK_DrawPcxGraphics (100, 50, “BMPA”);
PTK_PrintLabel (1, 1);
```

注意：BMP图形仅支持单色位图形。

2.48 打印一个 BMP 格式图形

public int PTK_PrintBMP (int **px**, int **py**, String **pcxname**, Bitmap **bmp**, int **iDire**)

功能

该函数的作用是打印一个BMP格式图形。

参数

px：X 坐标，以点(dots)为单位。
py：Y 坐标，以点(dots)为单位。
pcxname：自定义图形的名称，最大长度为 16 个字符。
bmp：需要下载到打印机中的 BMP 图形。
iDire：旋转角度。

0 – 0°
1 – 90°
2 – 180°
3 – 270°

返回值

0 --> OK

范例

```
PTK_PrintBMP (100, 50, “BMPA”, Bitmap, 0);
PTK_PrintLabel (1, 1);
```

2.49 打印二进制图形名称清单

public int PTK_BinGraphicsList ()

功能

该函数的作用是打印存储在打印机中的二进制图形名称清单。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_BinGraphicsList ();

2.50 删除二进制图形

public int PTK_BinGraphicsDel (String **pid**)

功能

该函数的作用是删除存储在打印机中的一个或所有二进制图形。

参数

pid：待删除图形的名称，最大长度为 16 个字符。

如果 pid = “*”，则存储在打印机中的所有二进制图形将被删除。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_BinGraphicsDel (“Bin2”);

2.51 存储一个二进制图形

public int PTK_BinGraphicsDownload (String **binName**, int **pbyte**, int **pH**, byte[] **Gdata**)

功能

该函数的作用是存储一个二进制图形到打印机。

参数

binName：自定义图形的名称，最大长度为 16 个字符。

pbyte：一行数据的字节数（1Byte = 8bits）。如果一行数据的点数不能整除 8，则其字节数应该等于商取整加 1。

pH：二进制图形的高度，以点(dots)为单位。

Gdata：二进制图形数据，数据量大小= pbyte * pH (Bytes)。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
Byte buf[] = {0xff,0xff,0xe0,0x1f,0xff,0xff…};
PTK_BinGraphicsDownload (“BinA”, 3, 24, buf);
```

2.52 打印一个二进制图形（0 不打印，1 打印）

public int PTK_RecallBinGraphics (int **px**, int **py**, String **binName**)

功能

该函数的作用是打印一个二进制图形。您可以通过 PTK_BinGraphicsDownload ()将二进制图形存储到打印机中。

参数

px：X 坐标，以点(dots)为单位。

py：Y 坐标，以点(dots)为单位。

binName：二进制图形文件的名称。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_RecallBinGraphics (10,100,"BinA");
PTK_PrintLabel (1, 1);
```

2.53 打印一个二进制图形（0 打印，1 不打印）

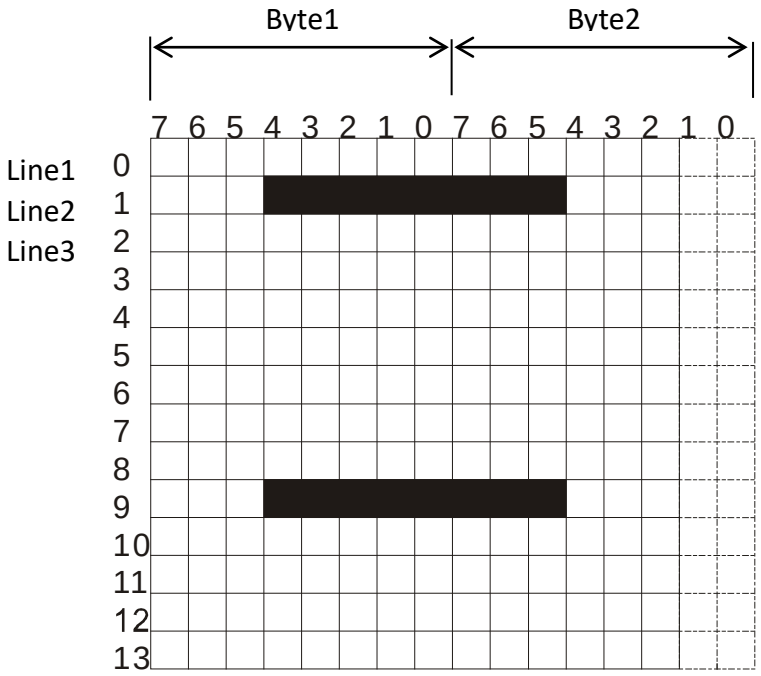
public int PTK_DrawBinGraphics (int **px**, int **py**, int **pbyte**, int **pH**, byte[] **Gdata**)

功能

该函数的作用是打印一个二进制图形。

说明

二进制数据传输顺序为从左到右，从上到下；Bit 值为 0 时此点打印，为 1 时此点不打印。如下图所示：



数据传输顺序为：Line1 的 Byte1(0xff)，Line1 的 Byte2(0xff)；Line2 的 Byte1(0xe0)，Line2 的 Byte2(0x1f)；Line3 的 Byte1(0xff)，Line3 的 Byte2(0xff)…，其中虚线部分是非图形区域，对应的 bit 值为 1。

参数

px：X 坐标，以点(dots)为单位。
py：Y 坐标，以点(dots)为单位。
pbyte：一行数据的字节数(1Byte = 8bits)。如果一行数据的点数不能整除 8，则其字节数应该等于商取整加 1。
ph：图形的高度，以点(dots)为单位。
Gdata：二进制图形数据，数据量大小= pbyte * pH (Bytes)。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
byte buf[] = {0xff,0xff,0xe0,0x1f,0xff,0xff…};
PTK_DrawBinGraphics (20, 30, 4, 14, buf);
PTK_PrintLabel (1, 1);
```

2.54 打印表单名称清单

public int PTK_FormList ()

功能

该函数的作用是打印存储在打印机中的表单名称清单。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_FormList ();

2.55 删除表单

public abstract int PTK_FormDel (String **pid**)

功能

该函数的作用是删除存储在打印机中的一个或所有表单。

参数

pid：待删除表单的名称，最大长度为 16 个字符。

如果 pid = ”*”，则存储在打印机中的所有表单将被删除。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_FormDel (“FORMNAME”);

2.56 存储一个表单

public int PTK_FormDownload (String **pid**)

原型

该函数的作用是存储一个表单到打印机。

参数

pid：自定义表单的名称，最大长度为 16 个字符。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_FormDownload (“FORMNAME”);

…

PTK_FormEnd ();

注意：此函数需与 PTK_FormEnd ()配合使用。

2.57 结束存储表单

public int PTK_FormEnd ()

功能

该函数的作用是结束存储表单。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_FormDownload (“Form1”) ;
…
PTK_FormEnd ();
```

注意：此函数需与 PTK_FormDownload ()配合使用。

2.58 运行表单

public int PTK_ExecForm (String **pid**)

功能

该函数的作用是运行指定的表单。具体用法请参考PTK_DrawText范例。

参数

pid：待运行表单的名称，最大长度为 16 个字符。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_ExecForm (“FORM1”);
```

2.59 定义一个序列号变量

public int PTK_DefineCounter (int **id**, int **maxNum**, char **ptext**, String **pstr**, String **pMsg**)

功能

该函数的作用是定义一个序列号变量。

参数

id：序列号 ID，取值范围：0～9。

maxNum：序列号最大位数，取值范围：1～40。

ptext：对齐方式。

L - 左对齐；

R - 右对齐；

C - 居中；

N - 不对齐。

pstr：序列号的变化规律，由“+”或“-”加上一个数字，再加上一个变化标志（D -十进制，B -二进制，O -八进制，H -十六进制，X -自定义模式，允许用户设置最多64个字符）组成，例如：

“+3D” - 每次增加 3，按照十进制计算，依次为 1234，1237，1240，…；

“-1B” - 每次减少 1，按照二进制计算，依次为 1111，1110，1101，…；

“-4O” - 每次减少 4，按照八进制计算，依次为 1234，1230，1224，…；

“-6H” - 每次减少 6，按照十六进制计算，依次为 1234，122E，1228，…；

“+3X” - 每次增加 3，若自定义序列号为：TE2DOKLU046MNY37，起始值是 T062，则依次为 T062，T06K，T060，…；

说明：如果不加变化标志，则默认按照十进制计算。

pMsg：提示信息字符串，可在打印机的 LCD 屏上或可编程键盘（KDU）的显示屏上显示。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

```
PTK_DefineCounter (0,6,’N’,”+1”, “EnterCode:");
```

2.60 定义一个变量字符串

public int PTK_DefineVariable (int **pid**, int **pmax**, char **porder**, String **pMsg**)

功能

该函数的作用是在FORM中定义一个变量字符串。

参数

pid：变量 ID 号码，取值范围：00～99。

pmax：最大字符数，取值范围：1～99。如果在可编程键盘（KDU）的显示屏上显示，则最大字符数为 16。

porder：对齐方式。

L - 左对齐；

R - 右对齐；

C - 居中；
N - 不对齐。

pMsg：提示信息字符串，可在打印机的 LCD 屏上或可编程键盘（KDU）的显示屏上显示。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

PTK_DefineVariable (0,16, L,"Enter Title:");

2.61 下载变量或序列号变量

public int PTK_Download ()

功能

该函数的作用是下载变量或序列号变量。具体用法请参考PTK_DrawText范例。

参数

无

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

PTK_Download ();

2.62 设置变量或序列号变量初始值

public int PTK_DownloadInitVar (String **pstr**)

功能

该函数的作用是设置变量或序列号变量的初始值。

参数

pstr：变量或序列号变量的初始值，最大长度为 16 个字符。

返回值

0 --> OK

范例

PTK_DownloadInitVar ("123456");

注意：此函数需与 PTK_Download ()配合使用。

2.63 调整文字间距

public int PTK_SetFontGap (int **gap**)

功能

该函数的作用是调整打印文字的字间距。

参数

gap：字间距的调节值，以点(dots)为单位，取值范围：-99～99。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

int return = PTK_SetFontGap (10);

注意：

1. ***打印文字的实际字间距 = 初始字间距 + 可调节字间距。***
2. ***此函数仅针对打印机内部字体有效。***
3. ***此函数仅适用于支持调整文字间距功能的打印机。***

2.64 重命名下载软字体

public int PTK_RenameDownloadFont (int **StoreType**, char **Fontname**, String **DownloadFontName**)

功能

该函数的作用是将下载到打印机中的字体进行重命名。

参数

StoreType：下载字体在打印机中的存储位置。

0 – SDRAM；

1 – FLASH。

Fontname：下载字体的 ID，取值范围：A～Z。

DownloadFontName：下载字体的名称。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

int return = PTK_RenameDownloadFont (1,'A',"arial");

2.65 校准超高频 RFID 芯片读写位置

public int PTK_RFIDCalibration ()

功能

该函数的作用是校准超高频RFID芯片读写位置。

参数

无

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_RFIDCalibration ();

注意：RFID打印机固件V1.73及以后版本支持此函数。

2.66 写超高频 RFID 标签

public int PTK_RWRFIDLabel (int **nRWMode**, int **nWForm**, int **nStartBlock**,
int **nWDataNum**, int **nWArea**, String **pstr**)

功能

该函数的作用是写超高频RFID标签。

参数

nRWMode：超高频 RFID 操作方式。
0 - 预留（暂无功能）；
1 - 写 RFID。
nWForm：RFID 写入格式。
0 - HEX（十六进制）；
1 - ASCII。
nStartBlock：写入起始块。
nWDataNum：写入字节数。
nWArea：写入区域。
0 - Reserved（保留区）；
1 - EPC；
3 - USER。
pstr：一个常量字符串。

该参数的格式由 nWForm 的取值来决定：如果 nWForm=1，写入数据长度必须以 2 个字节为单位，有效数据长度为 2 个字节的整数倍；如果 nWForm=0，写入数据长度以 4 个字节为单位，有效数据长度为 4 个字节的整数倍。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例 1

int return = PTK_RWRFIDLabel (1,0,2,6,1,"313233343536");

输出结果：

读取 EPC 区（Start=2，size=3word）

313233343536

范例 2

int return = PTK_RWRFIDLabel (1,1,0,6,3, ”POSTEK”);

输出结果：

读取 USER 区（Start=0，size=3word）

504F5354454B

2.67 设置超高频 RFID 标签密码及锁定超高频 RFID 标签

public int PTK_SetRFLabelPWAndLockRFLabel (int **nOperationMode**,
int **OperationnArea**, String **pstr**)

功能

该函数的作用是设置超高频RFID标签密码和锁定超高频RFID标签。

参数

nOperationMode：操作方式。

0 - 解锁；
1 - 锁定；
2 - 完全解锁；
3 - 完全锁定；
4 – 销毁密码写入。

OperationnArea：操作区域。

0 - 销毁密码区；
1 - 访问密码区；
2 - EPC；
3 - TID；
4 - USER。

pstr：一个常量字符串。（格式限制为 8 位 HEX 字符）

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例 1

int return = PTK_SetRFLabelPWAndLockRFLabel (1,1,"73BE115B");

输出结果：

读取访问密码区（password=“00000000”）

Cannot Read

读取访问密码区（password=“73BE115B”）

73BE115B

范例 2

int return = PTK_SetRFLabelPWAndLockRFLabel (4,0, ”5462EF21”);

输出结果：

读取销毁密码区

5462EF21

2.68 设置超高频 RFID 读写属性

public int PTK_SetRFID (int **nReservationParameters**, int **nReadWriteLocation**, int **ReadWriteArea**, int **nMaxErrNum**, int **nErrProcessingMethod**)

功能

该函数的作用是设置超高频RFID读写属性。

参数

nReservationParameters：预留参数。
nReadWriteLocation：超高频 RFID 读写位置。取值范围：0～999，单位为 mm。
ReadWriteArea：预留参数。
nMaxErrNum：最大错误标签数量。取值范围：0～9。
nErrProcessingMethod：预留参数。

返回值

0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
int return = PTK_SetRFID (0,0,0,2,0);
```

注意：此函数需与PTK_RWRFIDLabel ()配合使用。

2.69 写超高频 RFID 标签（不被清空）

public int PTK_RWRFIDLabelEx (int **nRWMode**, int **nWForm**, int **nStartBlock**,
int **nWDataNum**, int **nWArea**, String **pstr**)

功能

该函数的作用是写超高频 RFID 标签。

参数

nRWMode：超高频RFID操作方式。
0 - 保留；
1 - 写RFID。
nWForm：超高频RFID写入格式。
0 - HEX（十六进制）；
1 - ASCII。
nStartBlock：写入起始地址。
nWDataNum：写入字节数。
nWArea：写入区域。
0 - Reserved（保留区）；
1 - EPC；
3 - USER。
pstr：一个常量字符串。
该参数的格式由 **nWForm** 的取值来决定：如果 nWForm=1，写入数据长度必须以 2 个字节为单位，有效数据长度为 2 个字节的整数倍；如果 nWForm=0，写入数据长度以 4 个字节为单位，有效数据长度为 4 个字节的整数倍。

返回值

0 --> OK

其它返回值，请参考附录 错误返回值解析。

范例

PTK_RWRFIDLabelEx (1,0,0,2,1,“EFABCDFF”);
PTK_PrintLabel (1,1);

注意：

1. RFID 打印机固件 V1.74 及以后版本支持此函数。
2.此函数不会被 PTK_ClearBuffer ()清空。

2.70 读取超高频 RFID 标签信息

public int PTK_ReadRFTagDataNet (final String **IPAddress**, final int **Port**, final int **nDataBlock**, final int **nRFPower**, final int **bFeed**)

功能

该函数的作用是读取超高频RFID标签信息。该函数通过指定TCP/IP端口发送读取指令到打印机，打印机接收到指令后，反馈读取到的RFID标签信息。

参数

IPAddress：打印机的 IP 地址。
Port：打印机的网络端口。
nDataBlock：选择读取超高频 RFID 标签数据的区域。
0 - TID;
1 - EPC;
2 - TID+EPC。
nRFPower：打印机读取超高频 RFID 的功率，取值范围：0～30dBm。
bFeed：读取指令后是否向前走一张标签。
TRUE - 向前走一张标签有效；
FALSE - 向前走一张标签无效。

返回值

message.arg1 == 0 --> OK
其它返回值，请参考附录 错误返回值解析。

范例

```
handler = new Handler(){
        @Override
        public void handleMessage(Message msg) {
            // TODO Auto-generated method stub
            super.handleMessage(msg);
            if(msg.what ==  CDFPTKAndroid.PTK_MSG_WHAT_READRFIDMSG){
                if(msg.arg1 == 0){
```

```
                    tv_RRS.setText((String)msg.obj);
                }
                cdf.PTK_CloseConnectWiFi();
            }
        }
    };
 CDFPTKAndroid cdf = new CDFPTKAndroidImpl(RFIDReadActivity.this, handler);
cdf.PTK_ReadRFTagDataNet ("199.9.10.230",9100,1, 0,true);
```

注意：

1. *此函数只能在打印机配置 **WiFi** 模块的情况下调用。*
2. *此函数独立于其它函数，调用此函数不需要先连接 **WiFi**。*
3. *此函数的返回值需通过 **handler** 来接收，判断条件：**message.what == CDFPTKAndroid.PTK_MSG_WHAT_READRFIDMSG**。*
4. *在通过 **handler** 接收到返回信息后，请务必通过同一个 **CDFPTKAndroid** 实例对象调用 **PTK_CloseConnectWiFi ()**断开连接。*

2.71 打印 300DPI 打印机模板

PTK_Print300DPIDemo

功能

该函数的作用是打印一个模板，此模板为 300DPI 打印机而设计。

范例

```
public static CDFPTKAndroid cdf = new CDFPTKAndroidImpl (context, handler);
                                  cdf.PTK_Print300DpiDemo ();
```

2.72 打印 200DPI 打印机模板

PTK_Print200DPIDemo

功能

该函数的作用是打印一个模板，此模板为 203DPI 打印机而设计。

范例

```
public static CDFPTKAndroid cdf = new CDFPTKAndroidImpl (context, handler);
                                  cdf.PTK_Print200DpiDemo ();
```

附录 错误返回值解析

函数	错误返回值	解析
PTK_ClearBuffer ()	-230003	输入输出流异常
	-230004	设备未连接
	-230005	字符集转换出现异常
PTK_SetPrintSpeed ()	-280001	参数输入异常
	-280003	输入输出流异常
	-280004	设备未连接
	-280005	字符集转换出现异常
PTK_SetDirection ()	-350001	参数输入异常
	-350003	输入输出流异常
	-350004	设备未连接
	-350005	字符集转换出现异常
PTK_SetDarkness ()	-170003	输入输出流异常
	-170004	设备未连接
	-170005	字符集转换出现异常
PTK_SetLabelHeight ()	-260003	输入输出流异常
	-260004	设备未连接
	-260005	字符集转换出现异常
PTK_SetLabelWidth ()	-520003	输入输出流异常
	-520004	设备未连接
	-520005	字符集转换出现异常
PTK_PrintLabel ()	-320001	参数输入异常
	-320003	输入输出流异常
	-320004	设备未连接
	-320005	字符集转换出现异常
PTK_PrintLabelAuto ()	-321001	参数输入异常
	-321003	输入输出流异常
	-321004	设备未连接
	-321005	字符集转换出现异常
PTK_SoftFontList ()	-141803	输入输出流异常
	-141804	设备未连接
	-141805	字符集转换出现异常
PTK_SoftFontDel ()	-142001	参数输入异常
	-142003	输入输出流异常
	-142004	设备未连接
	-142005	字符集转换出现异常
PTK_EnableBackFeed ()	-191503	输入输出流异常
	-191504	设备未连接
	-191505	字符集转换出现异常
	-191103	输入输出流异常

函数	错误返回值	解析
PTK_DisableBackFeed ()	-191104	设备未连接
	-191105	字符集转换出现异常
PTK_PrintConfiguration ()	-300003	输入输出流异常
	-300004	设备未连接
	-300005	字符集转换出现异常
PTK_SetPrinterState ()	-240001	参数输入异常
	-240003	输入输出流异常
	-240004	设备未连接
	-240005	字符集转换出现异常
PTK_EnableFLASH ()	-352803	输入输出流异常
	-352804	设备未连接
	-352805	字符集转换出现异常
PTK_DisableFLASH ()	-352303	输入输出流异常
	-352304	设备未连接
	-352305	字符集转换出现异常
PTK_FeedMedia ()	-152203	输入输出流异常
	-152204	设备未连接
	-152205	字符集转换出现异常
PTK_MediaDetect ()	-221303	输入输出流异常
	-221304	设备未连接
	-221305	字符集转换出现异常
PTK_CutPage ()	-990403	输入输出流异常
	-990404	设备未连接
	-990405	字符集转换出现异常
PTK_CutPageEx ()	-122903	输入输出流异常
	-122904	设备未连接
	-122905	字符集转换出现异常
PTK_SetNetworkFeedbackParameter ()	-231503	输入输出流异常
	-231504	设备未连接
	-231505	字符集转换出现异常
PTK_SetFeedbackPort ()	-151101	参数输入异常
	-151103	输入输出流异常
	-151104	设备未连接
	-151105	字符集转换出现异常
PTK_DrawText ()	-290001	参数输入异常
	-290003	输入输出流异常
	-290004	设备未连接
	-290005	字符集转换出现异常
PTK_DrawBarcode ()	-110001	参数输入异常
	-110003	输入输出流异常
	-110004	设备未连接
	-110005	字符集转换出现异常

函数	错误返回值	解析
PTK_DrawBar2D_QR ()	-370101	参数输入异常
	-370103	输入输出流异常
	-370104	设备未连接
	-370105	字符集转换出现异常
PTK_DrawBar2D_DATAMATRIX ()	-370201	参数输入异常
	-370203	输入输出流异常
	-370204	设备未连接
	-370205	字符集转换出现异常
PTK_DrawBar2D_MaxiCode ()	-370301	参数输入异常
	-370303	输入输出流异常
	-370304	设备未连接
	-370305	字符集转换出现异常
PTK_DrawBar2D_HANXIN ()	-370401	参数输入异常
	-370403	输入输出流异常
	-370404	设备未连接
	-370405	字符集转换出现异常
PTK_DrawBar2D_Pdf417 ()	-370501	参数输入异常
	-370503	输入输出流异常
	-370504	设备未连接
	-370505	字符集转换出现异常
PTK_DrawRectangle ()	-330003	输入输出流异常
	-330004	设备未连接
	-330005	字符集转换出现异常
PTK_DrawLineXor ()	-211403	输入输出流异常
	-211404	设备未连接
	-211405	字符集转换出现异常
PTK_DrawLineOr ()	-212403	输入输出流异常
	-212404	设备未连接
	-212405	字符集转换出现异常
PTK_DrawDiagonal ()	-212803	输入输出流异常
	-212804	设备未连接
	-212805	字符集转换出现异常
PTK_DrawWhiteLine ()	-213203	输入输出流异常
	-213204	设备未连接
	-213205	字符集转换出现异常
PTK_PcxGraphicsList ()	-161803	输入输出流异常
	-161804	设备未连接
	-161805	字符集转换出现异常
PTK_PcxGraphicsDel ()	-162001	参数输入异常
	-162003	输入输出流异常
	-162004	设备未连接
	-162005	字符集转换出现异常

函数	错误返回值	解析
PTK_PcxGraphicsDownload ()	-162201	参数输入异常
	-162203	输入输出流异常
	-162204	设备未连接
	-162205	字符集转换出现异常
PTK_DrawPcxGraphics ()	-161601	参数输入异常
	-161603	输入输出流异常
	-161604	设备未连接
	-161605	字符集转换出现异常
PTK_BinGraphicsList ()	-111803	输入输出流异常
	-111804	设备未连接
	-111805	字符集转换出现异常
PTK_BinGraphicsDel ()	-112003	输入输出流异常
	-112004	设备未连接
	-112005	字符集转换出现异常
PTK_BinGraphicsDownload ()	-161303	输入输出流异常
	-161304	设备未连接
	-161305	字符集转换出现异常
PTK_RecallBinGraphics ()	-161203	输入输出流异常
	-161204	设备未连接
	-161205	字符集转换出现异常
PTK_DrawBinGraphics ()	-163203	输入输出流异常
	-163204	设备未连接
	-163205	字符集转换出现异常
PTK_FormList ()	-151803	输入输出流异常
	-151804	设备未连接
	-151805	字符集转换出现异常
PTK_FormDel ()	-152001	参数输入异常
	-152003	输入输出流异常
	-152004	设备未连接
	-152005	字符集转换出现异常
PTK_FormDownload ()	-152801	参数输入异常
	-152803	输入输出流异常
	-152804	设备未连接
	-152805	字符集转换出现异常
PTK_FormEnd ()	-151403	输入输出流异常
	-151404	设备未连接
	-151405	字符集转换出现异常
PTK_ExecForm ()	-152701	参数输入异常
	-152703	输入输出流异常
	-152704	设备未连接
	-152705	字符集转换出现异常
	-120001	参数输入异常

函数	错误返回值	解析
PTK_DefineCounter ()	-120003	输入输出流异常
	-120004	设备未连接
	-120005	字符集转换出现异常
PTK_DefineVariable ()	-310001	参数输入异常
	-310003	输入输出流异常
	-310004	设备未连接
	-310005	字符集转换出现异常
PTK_SetFontGap ()	-420003	输入输出流异常
	-420004	设备未连接
	-420005	字符集转换出现异常
PTK_RenameDownloadFont ()	-121503	输入输出流异常
	-121504	设备未连接
	-121505	字符集转换出现异常
PTK_RFIDCalibration ()	-222703	输入输出流异常
	-222704	设备未连接
	-222705	字符集转换出现异常
PTK_RWRFIDLabel ()	-271503	输入输出流异常
	-271504	设备未连接
	-271505	字符集转换出现异常
PTK_SetRFLabelPWAndLockRFLabel ()	-273503	输入输出流异常
	-273504	设备未连接
	-273505	字符集转换出现异常
PTK_SetRFID ()	-272803	输入输出流异常
	-272804	设备未连接
	-272805	字符集转换出现异常
PTK_RWRFIDLabelEx ()	-271003	输入输出流异常
	-271004	设备未连接
	-271005	字符集转换出现异常
PTK_ReadRFTagDataNet ()	-272702	指示主机 IP 地址无法确定
	-272703	输入输出流异常
PTK_Reset ()	-990303	输入输出流异常
	-990304	设备未连接
	-990305	字符集转换出现异常
PTK_FeedBack ()	-990503	输入输出流异常
	-990504	设备未连接
	-990505	字符集转换出现异常
PTK_Download ()	-990803	输入输出流异常
	-990804	设备未连接
	-990805	字符集转换出现异常
PTK_DownloadInitVar ()	-990903	输入输出流异常
	-990904	设备未连接
	-990905	字符集转换出现异常