

Android ESC SDK 说明文档

Android ESC SDK 说明文档.....	1
一 SDK 的组成.....	4
二 连接方式.....	5
2.1 蓝牙连接.....	5
2.2 WIFI 连接.....	6
2.3 USB 连接.....	7
三 打印指令.....	9
3.1 走纸.....	9
3.2 打印后走纸.....	10
3.3 打印后回退.....	11
3.4 打印并走纸 N 行.....	12
3.5 打印并回退 N 行.....	13
3.6 设置行间距.....	14
3.7 选择字体.....	15
3.8 设置语言.....	16
3.9 对齐方式.....	17
3.10 获取打印机状态.....	18
3.11 初始化打印机.....	22
3.12 设置打印机浓度.....	23

3.13	设置打印机速度.....	24
3.14	切纸.....	25
3.15	钱箱.....	26
3.16	蜂鸣器.....	27
3.17	打印文本.....	28
3.18	打印条码.....	30
3.19	打印二维码.....	33
3.20	打印图片.....	35
3.21	向打印机发送数据.....	37
3.22	从打印机读数据.....	38
3.23	打印 PDF417.....	39
3.24	标签定位.....	42
3.25	页模式.....	43
3.26	设置打印区域（页模式下）	45
3.27	设置打印方向（页模式下）	47
3.28	设置打印坐标（页模式下）	49
3.29	打印（页模式下）	51
3.30	获取 NV 位图列表.....	52
3.31	获取 NV 位图总内存大小.....	53
3.32	获取 NV 位图剩余内存大小.....	54
3.33	打印 NV 位图.....	55
3.34	删除指定 NV 位图.....	56

3.35 删除所有 NV 位图.....	57
3.36 下载图片到打印机.....	58
3.37 打印 Bin 文件.....	59
3.38 获取打印机功能列表.....	60
3.39 清除页模式缓存区数据.....	62
3.40 打印并返回行模式.....	63
3.41 设置左边距.....	64
表 1-1.....	65

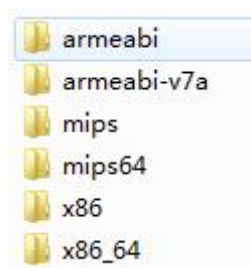
一 SDK 的组成

1 SDK jar 包

这里面有连接打印机的接口, 我们的 SDK 的连接接口有蓝牙、USB、WIFI 及串口。这里面还包含了打印指令的接口, 例如打印文本、条码、图片等。

2 SO 库

如图所示:



二 连接方式

2.1 蓝牙连接

连接蓝牙的函数：

```
int PortOpen(Context context,String portSetting)
```

参数：

context：上下文对象。

portSetting：“Bluetooth,”+蓝牙地址

例子：

```
Print.PortOpen(context,“Bluetooth,”+MAC)
```

MAC:打印机的蓝牙地址。

返回：

0：连接成功。

-1：连接失败。

断开蓝牙：

```
public static boolean PortClose()
```

例子：

```
Print.PorClose()
```

返回：

true：断开成功。

false：断开失败。

蓝牙是否连接：

```
public static boolean IsOpened()
```

例子:

```
Print.IsOpened()
```

返回:

true: 蓝牙已连接。

false: 蓝牙未连接。

2.2 WIFI 连接

连接 WIFI:

```
int PortOpen(Context context,String portSetting)
```

例子:

```
Print.PortOpen(context,"WiFi,"+IP+", "+PortNumber)
```

IP:打印机的 IP 地址。

PortNumber: 端口。 默认: 9100

返回:

0: 连接成功。

-1: 连接失败。

断开 WiFi:

```
public static boolean PortClose()
```

例子:

```
Print.PorClose()
```

返回:

true: 断开成功。

false: 断开失败。

WiFi 是否连接:

```
public static boolean IsOpened()
```

例子:

```
Print.IsOpened()
```

返回:

true: 已连接。

false: 未连接。

2.3 USB 连接

连接 USB:

```
int PortOpen(Context context, UsbDevice usbdevice)
```

例子:

```
Print.PortOpen(context,usbdevice)
```

usbdevice: UsbDevice 的对象

返回：

0：连接成功。

-1：连接失败。

断开 USB：

```
public static boolean PortClose()
```

例子：

Print.PorClose()

返回：

true：断开成功。

false：断开失败。

USB 是否连接：

```
public static boolean IsOpened()
```

例子：

Print.IsOpened()

返回：

true：已连接。

false：未连接。

三 打印指令

3.1 走纸

```
public static int PrintAndLineFeed()
```

例子：

Print.PrintAndLineFeed()

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.2 打印后走纸

```
public static int PrintAndFeed(int distance)
```

例子：

`Print.PrintAndFeed(distance)`

distance:走纸长度（单位： $\text{distance} \times 0.125\text{mm}$ ）。

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.3 打印后回退

```
public static int PrintAndReverseFeed(int distance)
```

例子：

`Print.PrintAndReverseFeed(distance)`

`distance`:回退长度（垂直或水平运动单位）。

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.4 打印并走纸 N 行

```
public static int PrintAndFeedNLine(byte lines)
```

例子：

Print.PrintAndFeedNLine(lines)

lines:N X（当前的行间距）。

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.5 打印并回退 N 行

```
public static int PrintAndReverseFeedNLine(int lines)
```

例子：Print.PrintAndReverseFeedNLine(lines)

lines:N X（当前的行间距）。

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.6 设置行间距

```
public static int SetDefaultTextLineSpace()
```

例子：

Print.SetDefaultTextLineSpace()

注释：设置默认的行间距。（3.75mm）

```
public static int SetTextLineSpace(byte lineSpace)
```

例子：

Print.SetTextLineSpace(byte lineSpace)

lineSpace:行间距（6=1mm）

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.7 选择字体

```
public static int SelectCharacterFont(byte characterFont)
```

例子：

Print.SelectCharacterFont(byte characterFont)

characterFont: 0: FontA 大字体。

1: FontB 小字体。

返回：

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.8 设置语言

```
public static int SetCharacterSet(byte characterSet)
```

例子：

```
Print.SetCharacterSet(byte characterSet)
```

设置中文：

```
Print.LanguageEncode="gb2312"
```

```
Print.SetCharacterSet(0)
```

设置英文：

```
Print.LanguageEncode="iso8859-1"
```

```
Print.SetCharacterSet(0)
```

设置繁体：

```
Print.LanguageEncode="big5"
```

```
Print.SetCharacterSet(0)
```

设置其他语言请查看文档最后的表 1-1。

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.9 对齐方式

```
public static int SetJustification(int justification)
```

例子：

```
Print.SetJustification(int justification)
```

justification: 0: 左对齐。

1: 居中。

2: 右对齐。

返回：

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.10 获取打印机状态

(1) 函数:

GetRealTimeStatus(byte realTimeItem)

参数:

realTimeItem:

- 1: 传输打印机状态。
- 2: 传输打印机状态。
- 3: 传输打印机状态。
- 4: 传输纸张状态。

返回: statusData: 返回的状态 (如下表), 长度为 1。

realTimeItem=1

位	0/1	HEX	Decimal	功能
0	0	00	0	固定为 0
1	0	02	2	固定为 0
2	0	00	0	固定为 0
3	0	00	0	联机
	1	08	8	脱机
4	1	10	16	固定为 0
5	0	00	0	固定为 0
6	0	00	0	打印机状态正常
	1	40	64	打印机状态异常
7	0	00	0	固定为 0

realTimeItem=2

位	0/1	HEX	Decimal	功能
0	0	00	0	固定为 0
1	0	00	0	固定为 0
2	0	00	0	固定为 0
3	0	00	0	固定为 0
4	0	00	0	固定为 0
5	0	00	0	固定为 0
6	0	00	0	打印机状态正常
7	1	40	64	打印机状态异常
	0	00	0	固定为 0

realTimeItem=3

位	0/1	HEX	Decimal	功能
0	0	00	0	固定为 0
1	0	00	0	固定为 0
2	0	00	0	固定为 0
3	0	00	0	固定为 0
4	0	00	0	固定为 0
5	0	00	0	上盖关
	1	20	00	上盖开
6	0	00	0	打印头温度正常
	1	40	64	打印头温度异常
7	0	00	0	固定为 0

realTimeItem=4

位	1/0	HEX	Decimal	功能
0	0	00	0	固定为 0
1	0	00	0	固定为 0
2,3	0	00	0	纸存在传感器检测到无纸
	1	0C	12	纸存在传感器检测到有纸
4	0	00	0	固定为 0
5,6	0	00	0	有纸
	1	60	96	纸尽
7	0	00	0	固定为 0

statusData.length==0 获取失败

例子：

```
byte statusData = Print.GetRealTimeStatus((byte)4);
```

(2) 函数：（适用于串口读取）

Print.GetTransmitStatus(int transmitItem)

transmitItem: 1: 获取纸张状态。

2: 获取钱箱状态。

返回：

statusData: 返回的状态（如下表），长度为 1。

查询纸张：

位	关/开	十六进制	十进制	状态
0, 1	关	00	0	纸将尽传感器：纸充足
	开	03	3	纸将尽传感器：纸将尽
2, 3	关	00	0	纸尽传感器：纸存在
	开	0c	12	纸尽传感器：纸不存在
4	关	00	0	固定
5, 6	--	--	--	保留
7	关	00	0	固定

查询钱箱：

位	关/开	十六进制	十进制	状态
0	关	00	0	钱箱插座引脚3信号为低
	开	01	1	钱箱插座引脚3信号为高
1~3	--	--	--	保留
4	关	00	0	固定
5,6	--	--	--	保留
7	关	00	0	固定

例子：

```
byte statusData = Print.GetTransmitStatus(1);
```

3.11 初始化打印机

```
public static int Initialize()
```

例子：

Print.Initialize()

将打印机还原成开机时的状态。

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.12 设置打印机浓度

```
public static int SetPrintDensity(byte density)
```

函数：

Print.SetPrintDensity(byte density)

density： 浓度。

返回：

不等于-1： 发送给打印机成功。

-1： 发送失败。

3.13 设置打印机速度

```
public static int SetPrintSpeed(byte speed)
```

函数：

Print.SetPrintSpeed(byte speed)

speed: 速度。

返回：

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.14 切纸

```
public static int CutPaper(int cutMode)
```

函数：

Print.CutPaper(int cutMode)

cutMode: 默认为 1。

先走纸再切纸

```
public static int CutPaper(int cutMode,int distance)
```

函数：

Print.CutPaper(int cutMode,int distance)

cutMode: 默认为 1。

distance: 走纸距离（6=1mm）

返回：

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.15 钱箱

```
public static int OpenCashdrawer(int openMode)
```

函数：

Print.OpenCashdrawer(int openMode)

openMode: 0: 打开 1 号钱箱。

1: 打开 2 号钱箱。

2: 2 个钱箱都打开。

返回：

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.16 蜂鸣器

```
public static int BeepBuzzer(byte times,byte t1,byte t2)
```

函数：

Print.BeepBuzzer(byte times,byte t1,byte t2)

times: 响的次数。

t1: 响的时间 ($t1 \times 100ms$)。

t2: 停止的时间 ($t2 \times 100ms$)。

返回：

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.17 打印文本

```
public static int PrintText(String data)
```

函数：

Print.PrintText(String data)

data: 文本内容。

例子：

Print.PrintText("TEXT\n")

```
public static int PrintText(String data,int alignment,int attribute,int textSize)
```

函数：

PrintText(String data,int alignment,int attribute,int textSize)

data: 文本内容。

alignment: 对齐方式。 0: 左对齐。

1: 居中。

2: 右对齐。

attribute: 样式。

位	5	4	3	2	1	0
FontB	-	-	-	-	-	1
加粗	-	-	-	-	1	-
下划线	-	-	-	1	-	-
反白	-	-	1	-	-	-
倍高	-	1	-	-	-	-

倍宽	1	-	-	-	-	-
----	---	---	---	---	---	---

textSize: 字体大小。

[范围]: $0 \leq n \leq 7, 16 \leq n \leq 23, 32 \leq n \leq 39, 48 \leq n \leq 55, 64 \leq n \leq 71, 80 \leq n \leq 87, 96 \leq n \leq 103, 112 \leq n \leq 119$;

[描述]: 用 0 到 2 位设定字符高度, 4 到 7 位设定字符宽度 如下所示。

位	开/关	十六进制	十进制	功能	
0-2	见表高度设定				
3	关	00	0	保留	
4-6	见表宽度设定				
7	关	00	0	保留	

字符高度设定

十六进制	十进制	高度
00	0	1(普通)
01	1	2(倍高)
02	2	3
03	3	4
04	4	5
05	5	6
06	6	7
07	7	8

字符宽度设定

十六进制	十进制	高度
00	0	1 (普通)
10	16	2 (倍宽)
20	32	3
30	48	4
40	64	5
50	80	6
60	96	7
70	112	8

例子:

Print.PrintText("TEXT\n",0,63,0)//包含了所有的样式。

返回:

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.18 打印条码

```
public static int PrintBarcode(int bcType,String bcData)
```

函数：

PrintBarcode(int bcType,String bcData)

bcType: 条码类型。

<i>m</i>	Bar code system	Range of <i>n</i>	Range of <i>d</i>
65	UPC-A	$n = 11, 12$	$48 \leq d \leq 57$
66	UPC-E	$n = 11, 12$	$48 \leq d \leq 57$ [where $d1 = 48$]
67	JAN13 / EAN13	$n = 12, 13$	$48 \leq d \leq 57$
68	JAN8 / EAN8	$n = 7, 8$	$48 \leq d \leq 57$
69	CODE39	$1 \leq n \leq 255$	$48 \leq d \leq 57, 65 \leq d \leq 90,$ $d = 32, 36, 37, 42, 43, 45, 46, 47$
70	ITF	$2 \leq n \leq 254$ (even number)	$48 \leq d \leq 57$
71	CODABAR (NW-7)	$2 \leq n \leq 255$	$48 \leq d \leq 57, 65 \leq d \leq 68,$ $97 \leq d \leq 100,$ $d = 36, 43, 45, 46, 47, 58$ [where $65 \leq d1 \leq 68, 65 \leq dn \leq 68,$ $97 \leq d1 \leq 100, 97 \leq dn \leq 100$]
72	CODE93	$1 \leq n \leq 255$	$0 \leq d \leq 127$
73	CODE128	$2 \leq n \leq 255$	$0 \leq d \leq 127$ [where $d1 = 123, 65 \leq d2 \leq 67$]

n 表示条码数据字节数。

d 指定条形码数据 。

bcData: 条码内容。

```
public static int PrintBarcode(int bcType,String bcData,int width,int height,int HRIPosition, int justification)
```

函数：

PrintBarcode(int bcType,String bcData,int width,int height,int HRIPosition, int justification)

参数：

bcType: 条码类型（同上）。

bcData: 条码内容。

width: 条码宽度。范围：（1-6）

	宽度（mm）	窄条码（mm）	宽条码（mm）
1	0.125	0.125	0.250
2	0.25	0.25	0.625
3	0.375	0.375	2.303
4	0.5	0.5	1.250
5	0.625	0.625	1.625
6	0.750	0.750	2

height: 条码高度。范围：1-255。

HRIPosition: 打印条形码时选择 HRI 字符的打印位置 。

n	打印位置
0,48	不打印
1,49	在条形码上方
2,50	在条形码下方
3,51	在条形码上方及下方

justification: 对齐方式。

0: 左对齐。

1: 居中。

2: 右对齐。

例子:

```
Print.PrintBarcode(73,"{BS/N:{C\014\042\070\116{A3",  
1,50,2,0);//这个是打印 code128 内容是:  
S/N:123456783
```

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.19 打印二维码

```
public static int PrintQRCode(String bcData)
```

函数：

PrintQRCode(String bcData)

参数：

bcData:二维码内容。

```
public static int PrintQRCode(String bcData,int sizeOfModule,int errorLevel,int justification)
```

函数：

PrintQRCode(String bcData,int sizeOfModule,int
errorLevel,int justification)

参数：

bcData: 二维码内容。

sizeOfModule: 二维码大小。范围 1-16;

errorLevel: 纠错等级。

N	功能	参考：可恢复字码比例
48	选择纠错等级 L	7%
49	选择纠错等级 M	15%
50	选择纠错等级 Q	25%
51	选择纠错等级 R	30%

justification: 对齐方式。

0: 左对齐。

1: 居中。

2: 右对齐。

例子：

`Print.PrintQRCode("二维码内容",6,48,0)`

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.20 打印图片

```
public static int PrintImage(String filePath,byte halftoneType,byte scaleMode)
```

函数：

PrintImage(String filePath,byte halftoneType,byte
scaleMode,int printdpi)

参数：

filePath： 图片路径（必须是 SD 卡的路径）。

halftoneType： 图片的算法类型。

0： 二值算法。

1： 半色调算法。

scaleMode： 打印机的模式选择。（默认： 0）

m	模式	垂直点密度	水平点密度
0,48	普通	203dpi	203dpi
1,49	倍宽	203dpi	101dpi
2,50	倍高	101dpi	203dpi
3,51	四倍大小	101dpi	101dpi

printdpi： 打印机分辨率（默认是 203， MPT8 是 300）

```
public static int PrintBitmap(Bitmap bmp,byte halftoneType,byte scaleMode)
```

函数：

PrintBitmap(Bitmap bmp,byte halftoneType,byte
scaleMode,int printdpi)

参数：

bmp: 图片对象。

halftoneType: 图片的算法类型。(同上)

scaleMode: 打印机的模式选择。(同上)

printdpi: 打印机分辨率 (同上)

例子:

```
Print.PrintBitmap(bmp,(byte)0,(byte)0,203)
```

返回:

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.21 向打印机发送数据

函数：

int WriteData(**byte**[] bData)

参数：

bData：向打印机写入的数据。

例子：

Print.WriteData("123abc\n".getBytes("GB2312"))// 向打印机发送文本是 123abc 的数据给打印机。

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.22 从打印机读数据

函数：

```
byte[] ReadData(int time)
```

参数：

time: 超时时间（单位秒）。

例子：

```
Print.ReadData(2)
```

```
//读取打印机的数据。
```

返回：

从打印机返回的数据，length 等于 0 打印机无数据返回。

3.23 打印 PDF417

函数:

```
int PrintPDF417(String bcData,  
  
                 byte dataColumns,  
  
                 byte dataRows,  
  
                 byte moduleWidth,  
  
                 byte rowHeight,  
  
                 byte errorMode,  
  
                 byte errorLevel,  
  
                 byte options)
```

参数:

bcData: 数据内容。

dataColumns: 设置打印数据区域的列数（范围：0-30）。

0: 自动处理。根据打印范围来确定打印列数。

dataRows: 设置 PDF417 的行数（范围：0，3-90）。

0: 自动处理。根据打印范围来确定打印行数。

moduleWidth: 设置模块宽度（范围：2-8）。

rowHeight: 设置模块高度= $n \times$ 宽度（范围：2-n-8）。

errorMode: 纠错模式。

48: 等级模式。

49: 比率模式。

errorLevel: 根据纠错模式分为两种 (n)。

等级模式:

n	功能	纠错码字数量
48	选择纠错等级 0	2
49	选择纠错等级 1	4
50	选择纠错等级 2	8
51	选择纠错等级 3	16
52	选择纠错等级 4	32
53	选择纠错等级 5	64
54	选择纠错等级 6	128
55	选择纠错等级 7	256
56	选择纠错等级 8	512

比率模式: $\lceil \text{数据码字} \times n \times 0.1 = (A) \rceil$ (小数部分四舍五入)

A	功能	纠错码字数量
0~3	选择纠错等级 1	4
4~10	选择纠错等级 2	8
11~20	选择纠错等级 3	16
21~45	选择纠错等级 4	32
46~100	选择纠错等级 5	64
101~200	选择纠错等级 6	128
201~400	选择纠错等级 7	256
400 以上	选择纠错等级 8	512

options: 选择可选项。

0: 选择标准 PDF417

1: 选择压缩 PDF417

返回:

≠-1: 发送给打印机成功。

=-1: 发送失败。

例子:

```
Print.PrintPDF417("123456",(byte)0,(byte)0,(byte)3,(byte)3,(byte)49,(byte)1,(byte)0)
```

3.24 标签定位

函数：

```
int GotoNextLabel()
```

注意：

该指令只用于标签的定位，连续纸不可用。

例子：

```
Print.GotoNextLabel()
```

```
//定位到标签纸缝标。
```

返回：

≠-1：发送给打印机成功。

=-1：发送失败。

3.25 页模式

```
public static int SelectPageMode()  
{
```

例子：

```
Print.SelectPageMode()
```

注释：必须是打印机支持页模式功能才能进入。

在页模式下你可以设置你想要打印的区域，坐标，方向。

```
//进入页模式。
```

```
Print.SelectPageMode()
```

```
//设置打印区域。
```

```
Print.SetPageModePrintArea(0,0,200,200)
```

```
//设置打印方向
```

```
Print.SetPageModePrintDirection(0)
```

```
//设置 X,Y 的坐标。
```

```
Print.SetPageModeAbsolutePosition(0,0)
```

```
//打印二维码（你也可以打印文字和条码）。
```

```
Print.PrintQRCode("abcdef",4,48,1)
```

```
//打印。
```

```
Print.PrintDataInPageMode()
```

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.26 设置打印区域（页模式下）

```
int SetPageModePrintArea(int horizontal,int  
vertical,int width,int height)
```

注释：必须是打印机支持页模式功能。

并且已经进入页模式才会生效。

参数：

horizontal:起始点的横坐标。

vertical: 起始点的纵坐标。

width: 区域的宽度。

height: 区域的高度。

例子：

//进入页模式。

```
Print.SelectPageMode()
```

//设置打印区域。

```
Print.SetPageModePrintArea(0,0,200,200)
```

//设置打印方向

```
Print.SetPageModePrintDirection(0)
```

//设置 X,Y 的坐标。

```
Print.SetPageModeAbsolutePosition(0,0)
```

//打印二维码（你也可以打印文字和条码）。

```
Print.PrintQRCode("abcdef",4,48,1)
```

//打印。

`Print.PrintDataInPageMode()`

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.27 设置打印方向（页模式下）

int SetPageModePrintDirection(**int** direction)

注释：必须是打印机支持页模式功能并且已经进入页模式后才会生效。

参数：

direction:打印方向。

0: 0 度。

1: 90 度。

2: 180 度。

3: 270 度。

例子：

//进入页模式。

Print.SelectPageMode()

//设置打印区域。

Print.SetPageModePrintArea(0,0,200,200)

//设置打印方向

Print.SetPageModePrintDirection(0)

//设置 X,Y 的坐标。

Print.SetPageModeAbsolutePosition(0,0)

//打印二维码（你也可以打印文字和条码）。

Print.PrintQRCode("abcdef",4,48,1)

//打印。

Print.PrintDataInPageMode()

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.28 设置打印坐标（页模式下）

```
int SetPageModeAbsolutePosition(int xPositon, int
yPosition)
```

注释：必须是打印机支持页模式功能并且已经进入页模式后才会生效。

参数：

xPosition: X 坐标。

yPosition: Y 坐标。

例子：

//进入页模式。

```
Print.SelectPageMode()
```

//设置打印区域。

```
Print.SetPageModePrintArea(0,0,200,200)
```

//设置打印方向

```
Print.SetPageModePrintDirection(0)
```

//设置 X,Y 的坐标。

```
Print.SetPageModeAbsolutePosition(0,0)
```

//打印二维码（你也可以打印文字和条码）。

```
Print.PrintQRCode("abcdef",4,48,1)
```

//打印。

```
Print.PrintDataInPageMode()
```

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.29 打印（页模式下）

int PrintDataInPageMode()

注释：必须是打印机支持页模式功能并且已经进入页模式后才会生效。

例子：

//进入页模式。

Print.SelectPageMode()

//设置打印区域。

Print.SetPageModePrintArea(0,0,200,200)

//设置打印方向

Print.SetPageModePrintDirection(0)

//设置 X,Y 的坐标。

Print.SetPageModeAbsolutePosition(0,0)

//打印二维码（你也可以打印文字和条码）。

Print.PrintQRCode("abcdef",4,48,1)

//打印。

Print.PrintDataInPageMode()

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.30 获取 NV 位图列表

int RefreshImageList(List<**byte**[]> lbImageIndex)

注释：必须是打印机支持 NV 位图功能才会生效。

参数：

lbImageIndex：图片列表序列号。

例子：

Print.RefreshImageList(lbImageIndex)

返回：

-1：打印机不支持 NV 位图功能。

1：获取列表成功。

3.31 获取 NV 位图总内存大小

int QueryNVStoreCapacity(**int**[] iSpace)

注释：必须是打印机支持 NV 位图功能才会生效。

参数：

iSpace：内存大小。

例子：

```
iSpace=new int[1];
```

```
Print.QueryNVStoreCapacity(iSpace);
```

返回：

-1：打印机不支持 NV 位图功能。

1：获取成功。

3.32 获取 NV 位图剩余内存大小

```
int QueryNVStoreRemainingCapacity(int[]  
storeRemainingCapacity)
```

注释：必须是打印机支持 NV 位图功能才会生效。

参数：

storeRemainingCapacity： 剩余内存大小。

例子：

```
storeRemainingCapacity=new int[1];  
Print.QueryNVStoreRemainingCapacity(storeRemaini  
ngCapacity);
```

返回：

-1： 打印机不支持 NV 位图功能。

1： 获取成功。

3.33 打印 NV 位图

int PrintNVImage(String imageNo,**int** scaleMode)

注释：必须是打印机支持 NV 位图功能才会生效。

参数：

imageNo：图片序列号。

scaleMode：模式（默认 0）。

例子：

```
Print.PrintNVImage(imageNo,0);
```

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.34 删除指定 NV 位图

int DeleteSpecifiedNVImage(String sImageIndex)

注释：必须是打印机支持 NV 位图功能才会生效。

参数：

sImageIndex：图片序列号。

例子：

```
Print.DeleteSpecifiedNVImage(sImageIndex);
```

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.35 删除所有 NV 位图

int DeleteAllNVImage()

注释：必须是打印机支持 NV 位图功能才会生效。

例子：

```
Print.DeleteAllNVImage();
```

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.36 下载图片到打印机

int DefineNVImage(String[] sArrFile, Handler handler)

注释：必须是打印机支持 NV 位图功能才会生效。

参数：

sArrFile：图片路径。

handler：Handler 的对象。

message.what：最大包数。

message.arg1：下载进度。

例子：

```
Print.DefineNVImage(sArrFile,handler);
```

返回：

不等于-1：发送给打印机成功。

-1：发送失败。

3.37 打印 Bin 文件

boolean PrintBinaryFile(String strPRNFile)

参数:

strPRNFile: bin 文件路径。

例子:

```
Print.PrintBinaryFile(strPRNFile);
```

返回:

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.38 获取打印机功能列表

```
int CapturePrinterFunction(int  
hprtModelPropertyKeyBeep,  
    int[] propType, byte[] value, int[] dataLen)
```

参数:

hprtModelPropertyKeyBeep: 功能代号。

HPRT_MODEL_PROPERTY_KEY_BEEP: 蜂鸣器。

HPRT_MODEL_PROPERTY_KEY_CUT: 切纸。

HPRT_MODEL_PROPERTY_KEY_DRAWER: 钱箱。

HPRT_MODEL_PROPERTY_KEY_BARCODE: 条码。

HPRT_MODEL_PROPERTY_KEY_PAGEMODE: 页模式。

HPRT_MODEL_PROPERTY_KEY_GET_REMAINING_POWER: 电源。

HPRT_MODEL_PROPERTY_CONNECT_TYPE: 连接方式。

HPRT_MODEL_PROPERTY_KEY_PRINT_RECEIPT: 小票。

propType: 种类编号。

Value: 是否支持。

(蜂鸣器、切纸、钱箱、页模式、电源、小票)

Value[0]==0 支持。

否则不支持。

(条码)

String barcode = new String(Value);

barcode 包含 QRCODE 支持二维码

barcode 包含 PDF417 支持 PDF417

dataLen: 返回数据的长度。

例子:

```
int[] propType=new int[1];
```

```
byte[] Value=new byte[500];
```

```
int[] DataLen=new int[1];
```

```
Print.CapturePrinterFunction(hprtModelPropertyKey  
Beep,propType,Value,DataLen);
```

返回:

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.39 清除页模式缓存区数据

int ClearPageModePrintAreaData()

例子:

```
Print.ClearPageModePrintAreaData();
```

返回:

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.40 打印并返回行模式

int PrintAndReturnStandardMode()

例子:

```
Print.PrintAndReturnStandardMode();
```

返回:

不等于-1: 发送给打印机成功。

-1: 发送失败。

3.41 设置左边距

int SetLeftMargin(**int** iLeftMargin)

参数:

iLeftMargin: 左边距 (单位 px)

例子:

```
Print.PrintAndReturnStandardMode();
```

返回:

不等于-1: 发送给打印机成功。

-1: 发送失败。

表 1-1

名称	characterSet	codepage
Default	0	gb2312
Chinese Simplified	0	gb2312
Chinese Traditional	0	big5
PC437(USA)	0	iso8859-1
KataKana	1	Shift_JIS
PC850(Multilingual)	2	iso8859-3
PC860(Portuguese)	3	iso8859-6
PC863(Canadian-French)	4	iso8859-1
PC865(Nordic)	5	iso8859-1
PC857(Turkish)	13	IBM857
PC737(Greek)	14	iso8859-7
ISO8859-7(Greek)	15	iso8859-7
WCP1252	16	iso8859-1
PC866(Cyrillic #2)	17	iso8859-5
PC852(Latin 2)	18	iso8859-2
PC858(Euro)	19	iso8859-15
KU42	20	ISO8859-11
TIS11(Thai)	21	ISO8859-11
TIS18(Thai)	26	ISO8859-11

PC720	32	iso8859-6
WPC775	33	iso8859-1
PC855(Cyrillic)	33	iso8859-5
PC862(Hebrew)	36	iso8859-8
PC864(Arabic)	37	iso8859-6
ISO8859-2(Latin2)	39	iso8859-2
ISO8859-15(Latin9)	40	iso8859-15
WPC1250	45	iso8859-2
WPC1251(Cyrillic)	46	iso8859-5
WPC1253	47	iso8859-7
WPC1254	48	iso8859-3
WPC1255	49	iso8859-8
WPC1256	50	iso8859-6
WPC1257	51	iso8859-1
WPC1258	52	bg2312
MIK(Cyrillic/Bulgarian)	54	iso8859-15
CP755	55	iso8859-5
Iran	56	iso8859-6
Iran II	57	iso8859-6
Latvian	58	iso8859-4
ISO-8859-1(West Europe)	59	iso8859-1
ISO-8859-3(Latin 3)	60	iso8859-3

ISO-8859-4(Baltic)	61	iso8859-4
ISO-8859-5(Cyrillic)	62	iso8859-5
ISO-8859-6(Arabic)	63	iso8859-6
ISO-8859-8(Hebrew)	64	iso8859-8
ISO-8859-9(Turkish)	65	iso8859-9
PC856	66	iso8859-8
ABICOIM	67	iso8859-15