

# Android ESC SDK Manual

Android ESC SDK Manual.....	1
1. SDK Introduction.....	4
2. Connecting Method.....	5
2.1 Bluetooth Connection.....	5
2.2 WIFI Connection.....	6
2.3 USB Connection.....	7
3.Print Command.....	9
3.1 Paper feed.....	9
3.2 Paper feed after printing.....	9
3.3 Reverse feed after printing.....	10
3.4 Print and feed n line.....	10
3.5 Print and reverse feed n line.....	10
3.6 Set text line space.....	11
3.7 Select character font.....	11
3.8 Set language.....	12
3.9 Set justification.....	14
3.10 Get printer status.....	14
3.11 Initializing printer.....	15
3.12 Set print density.....	16

3.13 Set print speed.....	16
3.14 Cut paper.....	17
3.15 Drawer.....	18
3.16 Beep buzzer.....	18
3.17 Print text.....	19
3.18 Print barcode.....	21
3.19 Print 2D code.....	23
3.20 Print bitmap.....	24
3.21 Send data to the printer.....	25
3.22 Read data from the printer.....	26
3.23 Print PDF417.....	27
3.24 Label location.....	30
3.25 Select page mode.....	30
3.26 Set print area in page mode.....	31
3.27 Set print direction in page mode.....	33
3.28 Set print position of x, y in page mode..	34
3.29 Print in page mode.....	35
3.30 Get NV bitmap list.....	36
3.31 Get NV bitmap memory capacity.....	36
3.32 Get NV bitmap remaining memory capacity.....	37
3.33 Print NV bitmap.....	38

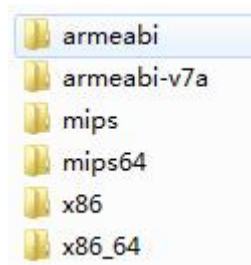
3.34 Delete specified NV bitmap.....	38
3.35 Delete all NV bitmap.....	39
3.36 Download NV bitmap to the printer.....	39
3.37 Print Binary file.....	40
3.38 Get printer function list.....	40
3.39 Clear page mode print area data.....	42
3.40 Print and return standard mode.....	42
Table 1-1.....	43

# 1. SDK Introduction

## 1) SDK jar :

In this jar, there are connectors which connect to the printer. Our SDK connectors include Bluetooth, USB, WIFI and signal interface. It also includes the connector of print commands, such as printing text, bar code, image, and so on.

## 2) SO Library :



## 2. Connecting Method

### 2.1 Bluetooth Connection

#### Connect Bluetooth :

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

Example:

```
Print.PortOpen(context,"Bluetooth,"+MAC)
```

MAC: Bluetooth address of printer

Return:

0: connection success

-1: connection failure

#### Disconnect Bluetooth:

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

Example:

```
Print.PorClose()
```

Return:

True: disconnection success

False: disconnection failure

#### Whether Bluetooth is connected:

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

Example:

```
Print.IsOpened()
```

Return:

True: Bluetooth connected

False: Bluetooth unconnected

## 2.2 WIFI Connection

### Connect WiFi:

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

Example:

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

IP: IP address of printer

PortNumber: port    Default: 9100

Return:

0: connection success

-1: connection failure

### Disconnect WiFi:

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

Example:

```
Print.PorClose()
```

Return:

True: disconnection success

False: disconnection failure

### Whether WiFi is connected:

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

Example:

Print.IsOpened()

Return:

True: connected

False: unconnected

## 2.3 USB Connection

**Connect USB:**

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

Example:

Print.PortOpen(context,usbdevice)

usbdevice: UsbDevice object

Return:

0:connection success

-1:connection failure

**Disconnect USB:**

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

Example:

Print.PorClose()

Return:

true:Disconnection success

false:disconnection failure

**Whether USB is connected:**

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

Example:

Print.IsOpened()

Return:

true:connected

false:unconnected



## 3.Print Command

### 3.1 Paper feed

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

Example:

Print.PrintAndLineFeed()

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.2 Paper feed after printing

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

Example:

Print.PrintAndFeed(distance)

distance:paper feeding length(vertical or horizontal moving unit)

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.3 Reverse feed after printing

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

Example:

Print.PrintAndReverseFeed(distance)

distance: reverse feed length (distance\*0.125mm)

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.4 Print and feed n line

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

Example:

Print.PrintAndFeedNLine(lines)

lines:N X (current line spacing)

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.5 Print and reverse feed n line

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

Example:Print.PrintAndReverseFeedNLine(lines)

lines:N X (current line spacing)

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.6 Set text line space

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

Example:

Print.SetDefaultTextLineSpace()

Note: Set default text line (3.75mm)

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

Example:

Print.SetTextLineSpace(byte lineSpace)

lineSpace: line spacing (6=1mm)

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.7 Select character font

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

Example:

Print.SelectCharacterFont(byte characterFont)

characterFont:

0:FontA big font

1:FontB small font

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.8 Set language

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

Example:

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

Set simple Chinese:

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

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

Set English:

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

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

Set traditional Chinese:

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

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

To set other languages, please refer to Table 1-1 on the last two pages.

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.9 Set justification

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

Example:

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

justification: 0: left justifying

1: center

2:right justifying

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.10 Get printer status

```
public static int GetTransmitStatus(int transmitItem,byte[] statusData)
```

Function:

```
Print.GetTransmitStatus(int transmitItem,byte[] statusData)
```

transmitItem: 1:Get paper status

2:Get drawer status

statusData:return status(see as below), length is 1

Search paper:

Bit	OFF/ON	Hex	Decimal	Status
0,1	OFF	00	0	Paper near end sensor: enough paper
	ON	03	3	Paper near end sensor: paper near end
2,3	OFF	00	0	Paper out sensor: with paper
	ON	0c	12	Paper out sensor: without paper
4	OFF	00	0	Fixed
5,6	--	--	--	Reserved
7	OFF	00	0	Fixed

Search drawer:

Bit	OFF/ON	Hex	Decimal	Status
0	OFF	00	0	Signal of drawer pin 3 is low
	ON	01	1	Signal of drawer pin 3 is high
1-3	--	--	--	Reserved
4	OFF	00	0	Fixed
5,6	--	--	--	Reserved
7	OFF	00	0	Fixed

Example:

```
statusData=new byte[1];
```

```
Print.GetTransmitStatus(1, statusData);
```

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.11 Initializing printer

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

Example:

```
Print.Initialize()
```

Restore the printer to the start-up status.

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.12 Set print density

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

Function:

Print.SetPrintDensity(byte density)

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.13 Set print speed

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

Function:

Print.SetPrintSpeed(byte speed)

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure



### 3.14 Cut paper

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

Function:

Print.CutPaper(int cutMode)

cutMode: default is 1

#### Feed paper and then cut paper

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

Function:

Print.CutPaper(int cutMode,int distance)

cutMode: default is 1

distance: feeding distance (6=1mm)

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.15 Drawer

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

Function:

Print.OpenCashdrawer(int openMode)

openMode:        0:Open No.1 drawer  
                  1:Open No.2 drawer  
                  2:Open two drawers

Return:

≠-1:sending (to printer) success  
-1:sending (to printer) failure

### 3.16 Beep buzzer

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

Function:

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

times: times of beep

t1:     time of beep (t1× 100ms)。

t2:     time of stop (t2× 100ms)。

Return:

≠-1:sending (to printer) success  
-1:sending (to printer) failure

### 3.17 Print text

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

Function:

Print.PrintText(String data)

data: text content

Example:

Print.PrintText("TEXT\n")

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

Function:

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

data: text content

alignment: alignment method    0:left alignment

1:center

2:right alignment

attribute: sample

Bit	5	4	3	2	1	0
FontB	-	-	-	-	-	1
Bold	-	-	-	-	1	-
Underline	-	-	-	1	-	-
Reverse	-	-	1	-	-	-
Double height	-	1	-	-	-	-
Double width	1	-	-	-	-	-

textSize: size of text character

[Range]: $0 \leq n \leq 7$ ,  $16 \leq n \leq 23$ ,  $32 \leq n \leq 39$ ,  $48 \leq n \leq 55$ ,  $64 \leq n \leq 71$ ,  $80 \leq n \leq 87$ ,

96≤n≤103, 112≤n≤119;

[Description]:Set the character height by 0-2. Set the character width by

4-7. See as below:

Bit	ON/OFF	Hex	Decimal	Function
0-2	See table 1			
3	OFF	00	0	Reserved
4-6	See table 2			
7	OFF	00	0	Reserved

Table 1 Character Height Setting

Hex	Decimal	Height
00	0	1(General)
01	1	2(Double height)
02	2	3
03	3	4
04	4	5
05	5	6
06	6	7
07	7	8

Table 2 Character Width Setting

Hex	Decimal	Width
00	0	1(General)
10	16	2(Double width)
20	32	3
30	48	4
40	64	5
50	80	6
60	96	7
70	112	8

Example:

Print.PrintText("TEXT\n",0,63,0)//includes all the samples.

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.18 Print barcode

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

Function:

PrintBarcode(int bcType,String bcData)

bcType:type of bar code

<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* indicates number of bytes of bar code data

*d* specifies bar code data

bcData: data of bar code

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

Function:

PrintBarcode(int bcType,String bcData,int width,int height,int

HRIPosition, int justification)

Parameter:

bcType: type of bar code

bcData: data of bar code

width:bar code width Range:(1-6)

	Width (mm)	Narrow Bar Code (mm)	Wide Bar Code (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:height of bar code range:1-255。

HRIPosition:

Select the HRI print position when printing bar code.

n	Print Position
0,48	No print
1,49	Above the bar code
2,50	Below the bar code
3,51	Both above and below the bar code

justification: justification method

0: left justifying

1: center

2: right justifying

Example:

```
Print.PrintBarCode(73,"{BS/N:{C\014\042\070\116{A3",1,50,2,0);//Print
```

code128:

S/N:123456783

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.19 Print 2D code

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

Function:

PrintQRCode(String bcData)

Parameter:

bcData: data of 2D code

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

Function:

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

Parameter:

bcData: data of 2D code

sizeOfModule: size of 2D code range 1-16;

errorLevel: level of error correction

N	Function	Refer: recoverable character ratio
48	Select error correction L	7%
49	Select error correction M	15%
50	Select error correction Q	25%
51	Select error correction R	30%

justification: justification method

0: left justifying

1: center

2: right justifying

Example:

```
Print.PrintQRCode("data of 2D code",6,48,0)
```

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.20 Print bitmap

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

Function:

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

Parameter:

filePath: path of image (should be the path of SD card)

halftoneType:algorithm type of image

0:binary algorithm

1:halftone algorithm

scaleMode:selection of printer mode (default:0)

N	Mode	Density of vertical point	Density of horizontal point
0,48	General	203dpi	203dpi
1,49	Double width	203dpi	101dpi
2,50	Double height	101dpi	203dpi
3,51	Four-times size	101dpi	101dpi

printdpi: printer DPI ( default is 203, MPT8 is 300 )

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

Function:



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

Parameter:

bmp: image object

halftoneType: algorithm type of image

scaleMode: selection of printer mode

printdpi: printer DPI

Example:

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

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.21 Send data to the printer

Function:

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

Parameter:

bData: the data sent to the printer

ExamplePrint.WriteData("123abc\n".getBytes("GB2312"))//Send the data of '123abc' to the printer.

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.22 Read data from the printer

Function:

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

Parameter:

Time: time of timeout (Unit: second)

Example:

Print.ReadData(2)

//The data read from the printer.

Return:

The data returned from the printer, length = 0 no data returned from the printer.

### 3.23 Print PDF417

Function:

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

Parameter:

bcData: the content of data

dataColumns: sets the number of columns in data print area (range: 0-30).

0: automatic setting. The number of print columns is set according to the print range.

dataRows: sets the number of rows of PDF417 (range: 0, 3-90).

0: automatic setting. The number of print rows is set according to the print range.

moduleWidth: sets the width of the module (range: 2-8).

rowHeight: sets the height of the module = n\*width (range: 2-n-8).

errorMode: error correction mode

48: level mode

49: ratio mode

errorLevel: two modes (n)

Level mode:

n	Function	Error Correction Number of PDF417 Code
48	Select error correction level 0	2
49	Select error correction level 1	4
50	Select error correction level 2	8
51	Select error correction level 3	16
52	Select error correction level 4	32
53	Select error correction level 5	64
54	Select error correction level 6	128
55	Select error correction level 7	256
56	Select error correction level 8	512

Ratio mode:  $\lfloor \text{Data code} \times n \times 0.1 = (A) \rfloor$  (decimal part round-off)

A	Function	Error Correction Number of PDF417 Code
0~3	Select error correction level 1	4
4~10	Select error correction level 2	8
11~20	Select error correction level 3	16
21~45	Select error correction level 4	32
46~100	Select error correction level 5	64
101~200	Select error correction level 6	128
201~400	Select error correction level 7	256
Above 400	Select error correction level 8	512

Options: select the options

0: select standard PDF417

1: select compacted PDF417

Return:

≠-1: sending (to printer) success

=-1: sending (to printer) failure

Example:

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

### 3.24 Label location

Function:

```
int GotoNextLabel()
```

Note:

This command is only for label location, not applicable for continuous paper.

Example:

```
Print.GotoNextLabel()
```

```
//Locate to the gap of label paper.
```

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.25 Select page mode

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

Example:

```
Print.SelectPageMode()
```

Note: The printer should support page mode function.

Under page mode you can set the position which you want to print.

```
//Enter page mode
```

```
Print.SelectPageMode()
```

```
//Set print area
Print.SetPageModePrintArea(0,0,200,200)

//Set print direction
Print.SetPageModePrintDirection(0)

//Set position of x, y
Print.SetPageModeAbsolutePosition(0,0)

//Print 2D code (can also print text and bar code)
Print.PrintQRCode("abcdef",4,48,1)

//Print
Print.PrintDataInPageMode()

Return:

≠-1:sending (to printer) success
-1:sending (to printer) failure
```

### 3.26 Set print area in page mode

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

Note: The printer should support page mode function. And it can take effect only when entering the page mode.

Parameter:

horizontal: x-coordinate of start point

vertical: y-coordinate of start point

width: width of the area

height: height of the area

Example:

```
//Enter page mode
```

```
Print.SelectPageMode()
```

```
//Set print area
```

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

```
//Set print direction
```

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

```
//Set position of x, y
```

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

```
//Print 2D code (can also print text and bar code)
```

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

```
//Print
```

```
Print.PrintDataInPageMode()
```

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure



### 3.27 Set print direction in page mode

**int** SetPageModePrintDirection(**int** direction)

Note: The printer should support page mode function. And it can take effect only when entering the page mode.

Parameter:

direction: print direction

0: 0°

1: 90°

2: 180°

3: 270°

Example:

```
//Enter page mode
```

```
Print.SelectPageMode()
```

```
//Set print area
```

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

```
//Set print direction
```

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

```
//Set position of x, y
```

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

```
//Print 2D code (can also print text and bar code)
```

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

```
//Print
```

Print.PrintDataInPageMode()

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.28 Set print position of x, y in page mode

**int** SetPageModeAbsolutePosition(**int** xPosition, **int** yPosition)

Note: The printer should support page mode function. And it can take effect only when entering the page mode.

Parameter:

xPosition: X-coordinate

yPosition: Y-coordinate

Example:

```
//Enter page mode
```

```
Print.SelectPageMode()
```

```
//Set print area
```

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

```
//Set print direction
```

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

```
//Set position of x, y
```

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

```
//Print 2D code (can also print text and bar code)
```

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

```
//Print
```

```
Print.PrintDataInPageMode()
```

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### **3.29 Print in page mode**

```
int PrintDataInPageMode()
```

Note: The printer should support page mode function. And it can take effect only when entering the page mode.

Example:

```
//Enter page mode
```

```
Print.SelectPageMode()
```

```
//Set print area
```

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

```
//Set print direction
```

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

```
//Set position of x, y
```

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

```
//Print 2D code (can also print text and bar code)
```

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

```
//Print
```

```
Print.PrintDataInPageMode()
```

Return:

≠-1:sending (to printer) success

-1:sending (to printer) failure

### 3.30 Get NV bitmap list

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

Note: Only when printer supports NV bitmap function can it take effect.

Parameter:

lbImageIndex: serial number of image list

Example:

```
Print.RefreshImageList(lbImageIndex)
```

Return:

-1: Printer does not support NV bitmap function.

1: Successfully get NV bitmap list.

### 3.31 Get NV bitmap memory capacity

```
int QueryNVStoreCapacity(int[] iSpace)
```

Note: Only when printer supports NV bitmap function can it take effect.

Parameter:

iSpace: memory capacity

Example:

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

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

Return:

-1: Printer does not support NV bitmap function.

1: Successfully get NV bitmap list.

### **3.32 Get NV bitmap remaining memory capacity**

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

Note: Only when printer supports NV bitmap function can it take effect.

Parameter:

storeRemainingCapacity: remaining memory capacity

Example:

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

```
Print.QueryNVStoreRemainingCapacity(storeRemainingCapacity);
```

Return:

-1: Printer does not support NV bitmap function.

1: Successfully get NV bitmap list.

### 3.33 Print NV bitmap

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

Note: Only when printer supports NV bitmap function can it take effect.

Parameter:

imageNo: serial number of image

scaleMode: mode (default: 0)

Example:

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

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.34 Delete specified NV bitmap

**int** DeleteSpecifiedNVImage(String slmageIndex)

Note: Only when printer supports NV bitmap function can it take effect.

Parameter:

slmageIndex: serial number of image

Example:

```
Print.DeleteSpecifiedNVImage(slmageIndex);
```

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.35 Delete all NV bitmap

**int** DeleteAllNVImage()

Note: Only when printer supports NV bitmap function can it take effect.

Example:

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

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.36 Download NV bitmap to the printer

**int** DefineNVImage(String[] sArrFile, Handler handler)

Note: Only when printer supports NV bitmap function can it take effect.

Parameter:

sArrFile: image path

Handler: Handler object

message.what: maximum number of data package

message.arg1: download progress

Example:

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

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.37 Print Binary file

**boolean** PrintBinaryFile(String strPRNFile)

Parameter:

strPRNFile: bin file path

Example:

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

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.38 Get printer function list

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

Parameter:

hprtModelPropertyKeyBeep: function code

*HPRT\_MODEL\_PROPERTY\_KEY\_BEEP: beeper*

*HPRT\_MODEL\_PROPERTY\_KEY\_CUT: cut paper*

*HPRT\_MODEL\_PROPERTY\_KEY\_DRAWER: drawer*

*HPRT\_MODEL\_PROPERTY\_KEY\_BARCODE: bar code*

*HPRT\_MODEL\_PROPERTY\_KEY\_PAGEMODE: page mode*

*HPRT\_MODEL\_PROPERTY\_KEY\_GET\_REMAINING\_POWE: power*

*HPRT\_MODEL\_PROPERTY\_CONNECT\_TYPE: connection type*



## ***HPRT\_MODEL\_PROPERTY\_KEY\_PRINT\_RECEIPT: receipt***

propType: type number

Value: whether to support

(beeper, cut paper, drawer, page mode, power, receipt)

Value[0]==0 support.

Otherwise nonsupport

(Bar code)

String barcode =new String(Value);

barcode contains QRCODE, support 2D code

barcode contains PDF417, support PDF417

dataLen: length of return data

Example:

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

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

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

```
Print.CapturePrinterFunction(hprtModelPropertyKeyBeep,propType,Value,DataLen);
```

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.39 Clear page mode print area data

**int** ClearPageModePrintAreaData()

Example:

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

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

### 3.40 Print and return standard mode

**int** PrintAndReturnStandardMode()

Example:

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

Return:

≠-1: sending (to printer) success

-1: sending (to printer) failure

**Table 1-1**

Name	Character Set	Code page
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