
POS SDK For IOS
User's Manual_Chinese



山东新北洋信息技术股份有限公司

目录

目录	I
关于本用户手册	II
目的	II
内容	II
1. 综述	1
1.1 功能	1
1.2 操作环境	2
1.3 发布包中的文件	2
1.4 版本记录	3
2. 示例程序	4
2.1 例程启用	4
2.2 如何使用例程	5
3. 编程指南	21
3.1 连接打印机	21
3.2 使用静态库	21
4. API 相关	29
4.1 通讯相关	29
4.2 POS SDK API 相关	35
5. 附录	87
附录 A. WIFIPortToFile 类的使用	87
附录 B. 错误码列表	88
附录 C. 条码说明	90
附录 D. 128 码	93
附录 E. 程序流程	97

关于本用户手册

目的

本手册的目的是为客户介绍 POS SDK For IOS 的例程和 API 的使用。

内容

本手册主要由以下几部分组成：

- 第一章 [综述](#)
- 第二章 [示例程序](#)
- 第三章 [编程指南](#)
- 第四章 [API 相关](#)
- 第五章 [附录](#)

1. 综述

本章主要是关于 POS SDK For IOS 的设备搜索连接及关闭、API 函数功能、操作应用环境、发布包中文件及帮助文档版本的说明。

1.1 功能

● WIFI

WIFI 搜索功能、端口连接及关闭功能、数据读写及读写超时设置功能、数据保存到文件功能。

● BlueTooth

BlueTooth 搜索功能、端口连接及关闭功能、数据读写功能。

● BluetoothMFI

Bluetooth(MFI)搜索功能、端口连接及关闭功能、数据读写功能。

● API

1) 系统设置功能(创建 WIFIPort/BlueToothPort/Bluetooth(MFI)类的实例、打印机初始化、选择打印模式、选择纸张类型、设置横纵向移动单位、查询打印机状态、走纸、切纸、下载文件)钱箱打开功能；

2) 打印字符（选择字符集和代码页、设置行高、设置字符间距、选择文本对齐方式、字体选择、反显、粗体、下划线、旋转、选择放大倍数、双色打印、中英文用户自定义字符打印、光栅化字符打印）；

3) 图像打印功能（8 点/24 点单双密度位图打印、RAM 位图下载及打印、Flash 位图下载及打印、光栅化位图打印）；

4) 一维条码打印功能（UPC-A、UPC-E、EAN-8、EAN-13、Code39、Code93、ITF、Codabar、Code128）；

5) 二维条码打印功能（PDF417、QR、Maxicode、GS1 DataBar 和 GS1 复合码）；

6) 标准模式设置（设置打印区域宽度、设置左边距、设置打印横向起始的位置）；

7) 页模式设置（设置打印区域、选择打印方向、设置打印横纵向起始位置、页模式打印及清空）；

8) 读磁道数据（读取第一磁道、第二磁道、第三磁道、三个磁道数据）

9) IC 卡相关（IC 卡复位、控制命令 T0 协议、T1 协议）

1.2 操作环境

● IOS 版本

IOS 系统 8.X 及以上版本 系列版本

● IOS 设备

iPhone（蓝牙只支持 iPhone 4S 以上的 IOS 设备）

iPad/iPad mini（蓝牙只支持 iPad 3 以上的 IOS 设备）

● 打印机

SNBC 的 POS 系列打印机

● 支持端口

WiFi

BlueTooth

Bluetooth(MFI)

● 应用环境

Xcode6.3 以上

1.3 发布包中的文件

文件	描述
POSSDKForIOSlib	POSSDKForIOS.a 、 PortIO.h、 ErrorCode.h、 POSCommand.h、 POSSDK.h、 WIFIPort.h、 WIFIPortToFile.h、 BlueToothPort.h
POSSDKForIOSDemo	例程工程及源码
User's Manual	POS SDK For IOS User's Manual_Chinese.pdf POS SDK For IOS User's Manual_English.pdf

1.4 版本记录

版本	日期	版本记录
V1.00	30/10/2013	首版本
V1.10	30/10/2014	增加蓝牙通讯功能
V1.11	23/04/2014	增加读磁及 IC 卡接口描述
V1.12	1/11/2015	增加经过 MFI 认证的蓝牙接口通讯功能
V2.10	10/09/2018	1. 搜索打印机增加 MAC 地址显示 2. 开启“FF00”服务使用流控方式下发数据，提升打印速度 3. 增加对于苹果设备断开蓝牙及打印机断开蓝牙情况的通知

2. 示例程序

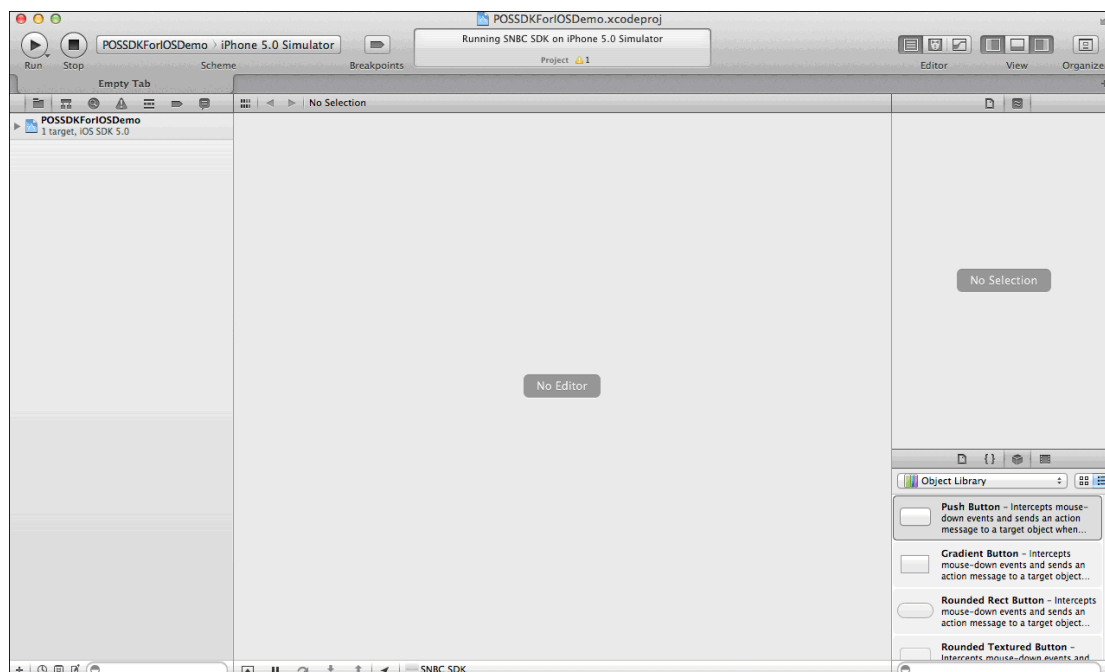
本章主要描述如何使用示例程序。

例程有如下功能：

- 搜索打印机
- 打印机连接及关闭
- 设置标准模式打印参数
- 示例页模式下文本、条码、图像打印
- 文本打印
- 光栅化文本打印
- 用户自定义字符打印
- 条码打印
- PDF417 条码打印
- QR 码打印
- Maxicode 码打印
- GS1 DataBar 和 GS1 符合码打印
- 图像打印（8 点/24 点单双密度图像）
- RAM/Flash 位图下载及打印
- 光栅化位图打印
- 下载文件
- 读取磁道数据
- IC 卡控制命令，以 T0 协议为例

2.1 例程启用

- 1) 从压缩包中直接解压出例程；
- 2) 直接找到文件 `POSSDKForIOSDemo` 并双击 `"POSSDKForIOSDemo.xcodeproj"`，得到如下图：



- 3) 在 "Scheme" 中选择要使用的设备；
- 4) 点按左上方得[Run]键；
- 5) 这样例程便被安装到了目标 IOS 设备上，并且启动了例程。

2.2 如何使用例程

● 建立连接

对于 WIFI 接口打印机，首先要执行下述工作：

- 1) 确保打印机的 WIFI 接口板可用并连入打印机；
- 2) 设置 IOS 设备和打印机的 WIFI 网络的 SSID 一致；
- 3) 设置 IOS 设备的 IP 和打印机的 IP 在同一网段。

对于 BLE 模式蓝牙接口打印机，首先要制定下述工作：

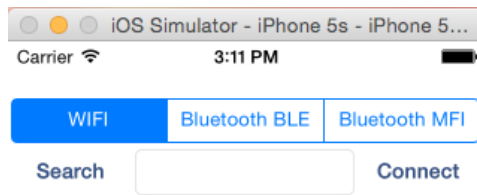
- 1) 确保打印机的 BLE 模式蓝牙接口板可用并连入打印机；

对于经过 MFI 认证的蓝牙接口打印机，首先要制定下述工作：

- 1) 确保打印机的 MFI 蓝牙接口板可用并连入打印机；
- 2) 在 iOS 系统设置页面中连接蓝牙接口

● 使用例程连接打印机

搜索及连接打印机界面，如下：



搜索及连接打印机步骤如下：

- 1) 启动例程，详细参照[例程启用](#)；
- 2) 搜索打印机。选择端口类型，然后点按主面上的 [Search]键,可用的打印机便以下拉菜单的形式被列举出来，你可以从下拉菜单中选择想要连接的打印机；
- 3) 当你选择了打印机，其打印机信息将会被添加到 Label 控件中；
- 4) 选择好打印机后，点按主界面上的 [Connect] 键来打开打印机端口；
- 5) 如果连接成功，[Search]控件就会变暗并且禁止使用，按键 [Connect] 的 Title 变为 [Disconnect]，可以通过点按[Disconnect]关闭已打开的端口。

● 主界面

主界面的功能图，如下：



主界面的功能说明:

处理	描述
设置标准模式打印参数	在主界面上点按[Standard mode]按键.具体细节请参照 设置标准模式 。
页模式打印	在主界面上点按[Page mode print] 按键。
字符打印	在主界面上点按[Text] 按键.具体细节请参照 字符打印 。
光栅化字符打印	在主界面上点按[Text raster print] 按键.具体细节请参照 光栅化字符打印 。
用户自定义字符打印	在主界面上点按[Custom character] 按键。
一维条码打印	在主界面上点按[Barcode] 按键.具体细节请参照 一维条码打印 。
PDF417 条码打印	在主界面上点按[PDF417] 按键.具体细节请参照 PDF417 条码打印 。
QR 码打印	在主界面上点按[QR] 按键.具体细节请参照 QR 码打印 。

Maxicode 码打印	在主界面上点按[Maxicode] 按键。
GS1 Databar 条码打印	在主界面上点按[GS1] 按键.具体细节请参照 GS1 Databar 条码打印 。
图像打印	在主界面上点按[Image print] 按键.具体细节请参照 图像打印 。
RAM/Flash 位图下载并打印	在主界面上点按[Image download] 按键.具体细节请参照 RAM/Flash 位图下载并打印 。
光栅化位图打印	在主界面上点按[Image print raster] 按键.具体细节请参照 光栅化位图打印 。
文件下载	在主界面上点按[Download file] 按键。
读取磁道数据	在主界面上点按[Msr] 按键。以第二磁道为例
IC 卡控制 Tn 协议	在主界面上点按[IC] 按键。以 T0 为例

● 设置标准模式；

1) 标准模式的起始横坐标、左边距、打印区域宽度属性设置，如下图：

Carrier 4:39 PM

Starting position(dots): 0

Left margin(dots): 0

Print area width(dots): 640

Back Select

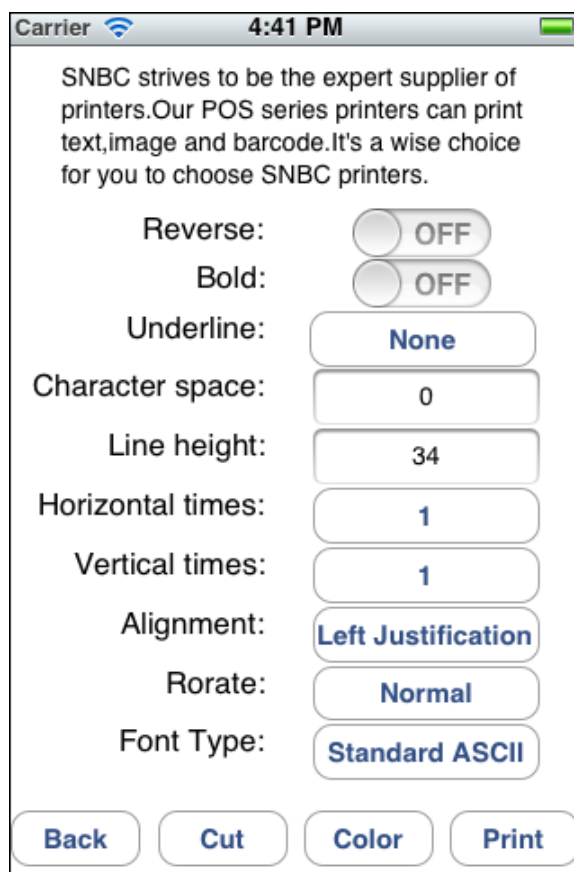
2) 点按 Standard mode 界面中的[Select] 按键可以实现标准模式参数设置。

3) 点按 Standard mode 界面中的[Back] 按键可以实现从[Standard mode]界面

返回到主界面。

● 字符打印

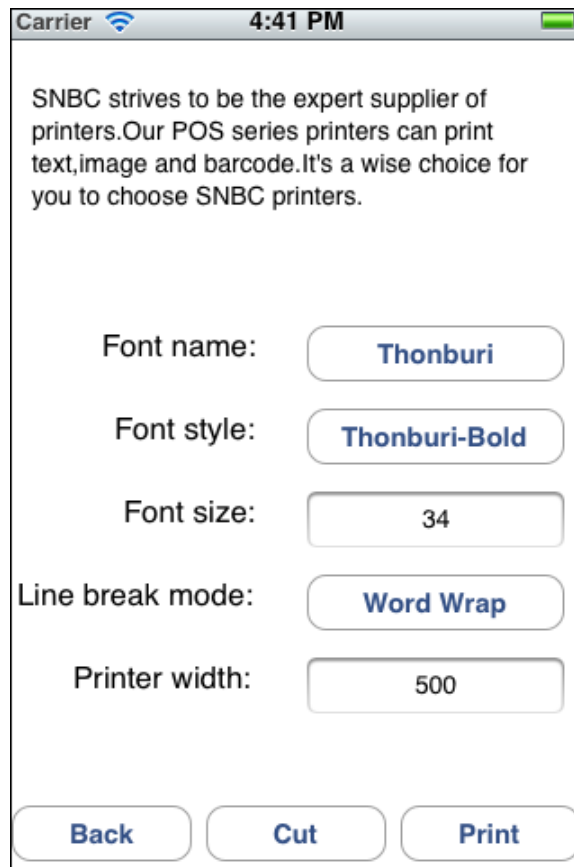
- 1) 在 [textContent] 的 TextView 框中输入要打印的字符串；
- 2) 设置输入字符串的字符属性. 可以设置的字符属性见下图：



- 3) 点按 Text 界面上的[Print] 按键可以实现字符打印；
- 4) 点按 Text 界面上的[Color] 按键可以实现双色打印的另外一种颜色打印；
- 5) 点按 Text 界面上的[Cut] 按键可以实现切纸功能；
- 6) 点按 Text 界面上的[Back] 按键可以实现从 Text 界面返回到主界面。

● 光栅化字符打印

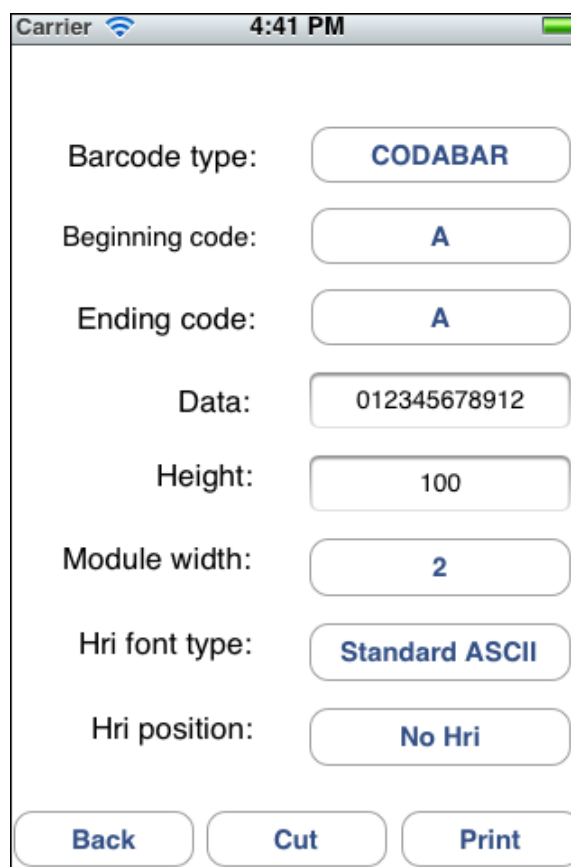
- 1) 在[TextRasterPrintContent] 的 TextView 框中输入要打印的字符串；
- 2) 设置输入字符串的字符属性. 可以设置的字符属性见下图：



- 3) 点按 Text raster print 界面上的[Print] 按键可以实现光栅化字符打印;
- 4) 点按 Text raster print 界面上的[Cut] 按键可以实现切纸功能;
- 5) 点按 Text raster print 界面上的[Back] 按键可以实现从 Text raster print 界面返回到主界面。

● 一维条码打印

- 1) 点按[Barcode type] 选择条码类型;
- 2) 在[BarcodeData]的 Label 框中输入条码数据;
- 3) 设置要打印条码的属性. 可以设置的条码属性如下:



Carrier 4:41 PM

Barcode type: CODABAR

Beginning code: A

Ending code: A

Data: 012345678912

Height: 100

Module width: 2

Hri font type: Standard ASCII

Hri position: No Hri

Back Cut Print

- 4) 点按 Barcode 界面上的[Print] 按键可以实现条码打印功能;
- 5) 点按 Barcode 界面上的[Cut] 按键可以实现切纸功能;
- 6) 点按 Barcode 界面上的[Back] 按键可以实现从 Barcode 界面返回到主界面。

【说明】

- a) 对于 Codabar 条码, 有两个特别的属性型需要选择: 起始字符和终止字符;
- b) 对于 Code128 码, 需要选择字符集。

● PDF417 条码打印

- 1) 在[PDF417Data]的 Label 框中键入要打印的条码数据;
- 2) 设置要打印的PDF417条码的属性。可以设置的PDF417条码的属性如下:

Carrier 4:42 PM

Data: abcdefghijklmnopqrstuvwxyz

Appearance to height: 1

Appearance to width: 1

Rows: 3

Columns: 1

XSize: 4

Line height: 5

Correction: 0

Back Cut Print

- 3) 点按 PDF417 界面上的[Print] 按键可以实现 PDF417 条码打印;
- 4) 点按 PDF417 界面上的[Cut] 按键可以实现切纸功能;
- 5) 点按 PDF417 界面上的[Back] 按键可以实现从 PDF417 界面返回到主界面。

● QR 码打印

- 1) 在[QRData]的 Label 框中键入要打印的条码数据;
- 2) 设置要打印的 QR 条码的属性。可以设置的 QR 条码的属性如下:

Carrier 4:43 PM

Data: QA,AABCDEFGHJKLMNOP123456...

Basic element width: 5

Symbol type: Enhanced Type

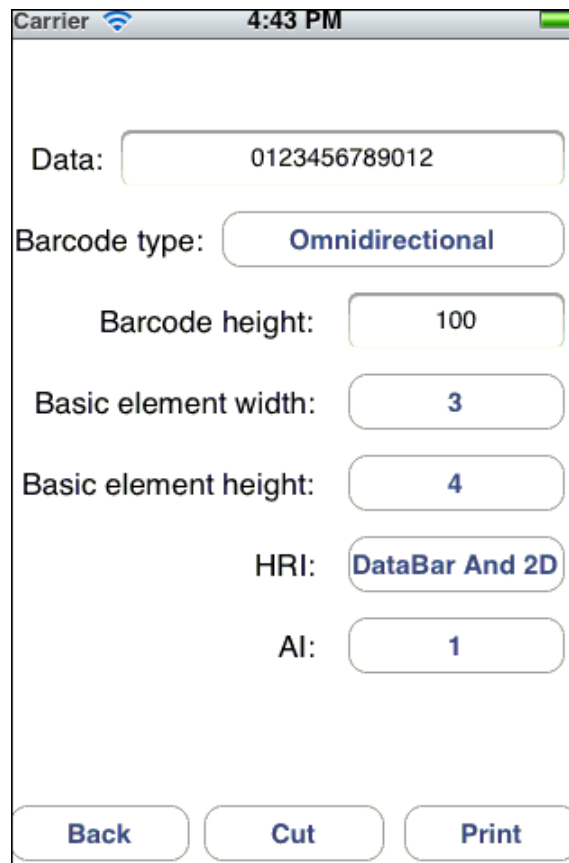
Language mode: Chinese

Back Cut Print

- 3) 点按 QR 界面上的[Print] 按键可以实现 QR 条码打印;
- 4) 点按 QR 界面上的[Cut] 按键可以实现切纸功能;
- 5) 点按 QR 界面上的[Back] 按键可以实现从 QR 界面返回到主界面。

● GS1 Databar 条码打印

- 1) 在[GS1DataBarData]的 Label 框中键入要打印的条码数据;
- 2) 设置要打印的 GS1 DataBar 条码的属性。可以设置的 GS1 DataBar 条码的属性如下:



Carrier 4:43 PM

Data: 0123456789012

Barcode type: Omnidirectional

Barcode height: 100

Basic element width: 3

Basic element height: 4

HRI: DataBar And 2D

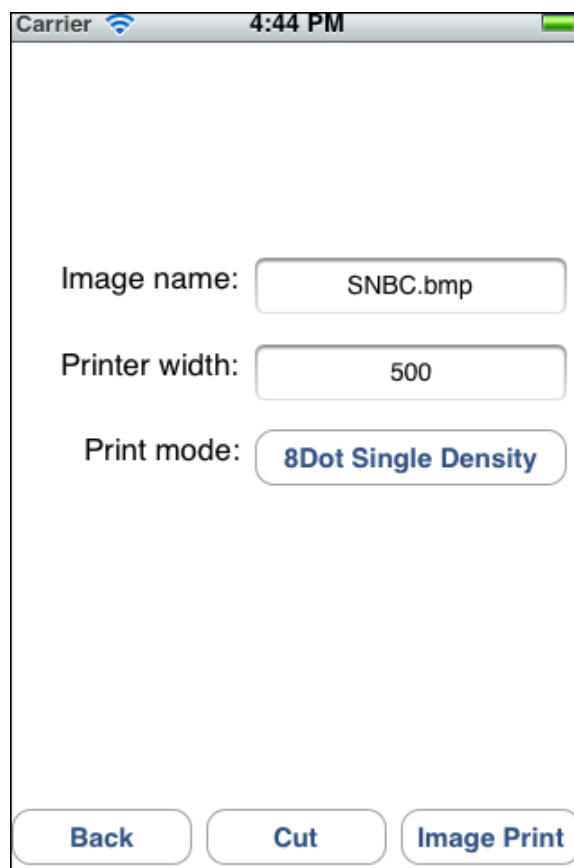
AI: 1

Back Cut Print

- 3) 点按 GS1 界面上的[Print] 按键可以实现 GS1 条码打印;
- 4) 点按 GS1 界面上的[Cut] 按键可以实现切纸功能;
- 5) 点按 GS1 界面上的[Back] 按键可以实现从 GS1 界面返回到主界面。

● 图像打印

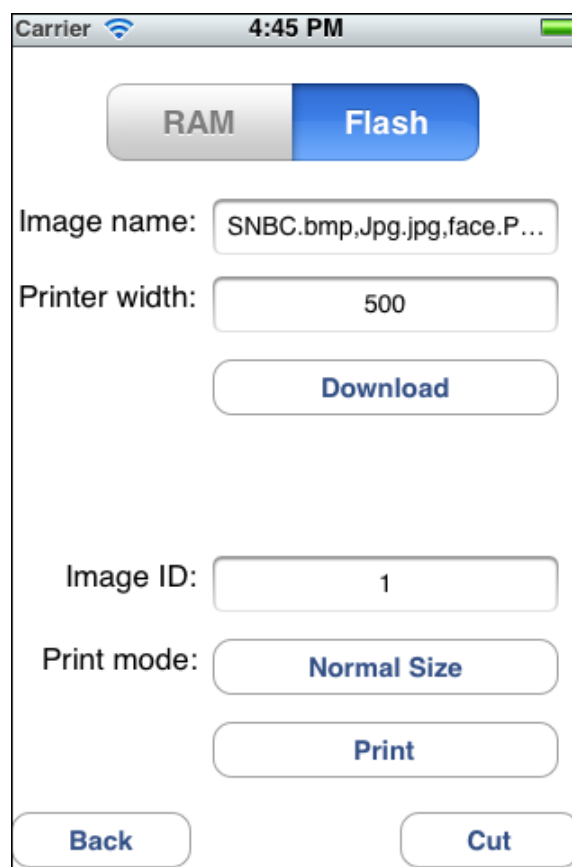
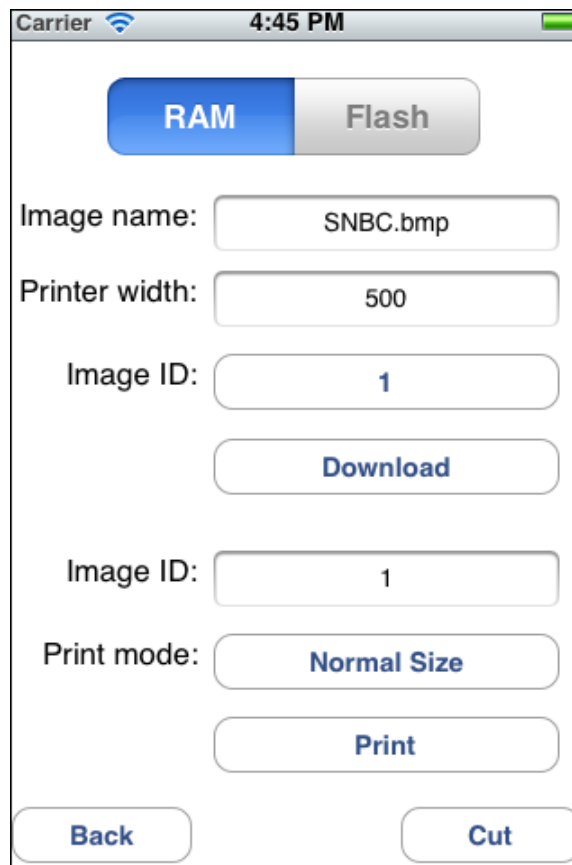
- 1) 在[ImageName]的 Label 框中键入要打印的图像名;
- 2) 设置打印头宽度, 默认为 500;
- 3) 选择打印模式, 默认为 8 点单精度;



- 4)点按 Image 界面上的[Image Print] 按键可以实现打印指定图像;
- 5)点按 Image 界面上的[Cut] 按键可以实现切纸功能;
- 6)点按 Image 界面上的[Back] 按键可以实现从 Image 界面返回到主界面。

● RAM/Flash 位图下载并打印

- 1) 点按 UISegmentedControl 控件[RAMorFlashSelectButton]选择打印 RAM 还是 Flash 位图;
- 2) 在 Label 框输入要打印的图像名, RAM 位图每次只可下载单幅图像, Flash 位图, 可以一次下载多幅图像, 两幅图像之间使用逗号隔开;
- 3) 设置要打印的 RAM/Flash 位图的属性. 可以设置的 RAM/Flash 位图的属性如下:



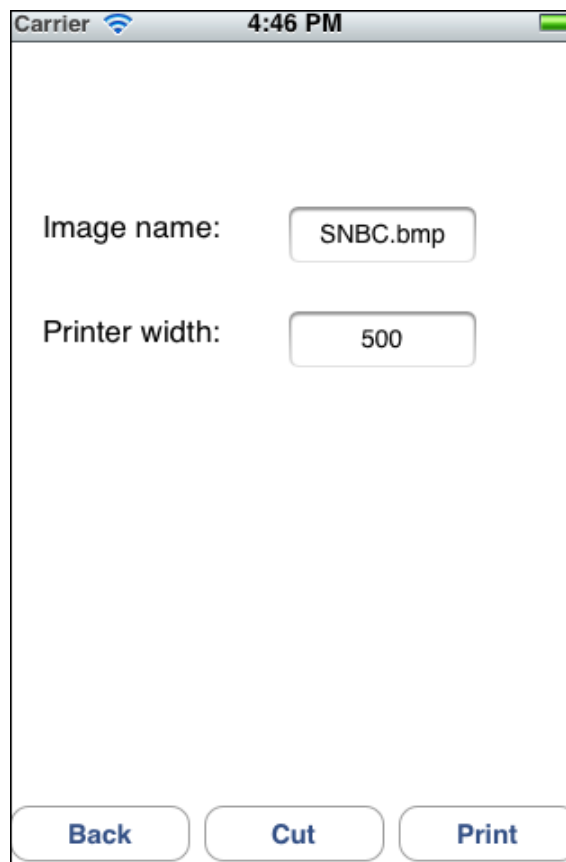
4) 点按 Image download 界面上的[Download] 按键可以实现图像的下载功

能；

- 5) 点按 Image download 界面上的[Print] 按键可以实现图像的打印功能；
- 6) 点按 Image download 界面上的[Cut] 按键可以实现切纸功能；
- 7) 点按 Image download 界面上的[Back] 按键可以实现从 Image download 界面返回到主界面。

● 光栅化位图打印

- 1) 在 Label 框输入要打印的图像名，每次只可下载单幅图像；
- 2) 设置打印头宽度（默认为 500）；



- 3) 点按 Image print raster 界面上的[Print] 按键可以实现光栅图像的打印功能；
- 4) 点按 Image print raster 界面上的[Cut] 按键可以实现切纸功能；
- 5) 点按 Image print raster 界面上的[Back] 按键可以实现从 Image print raster 界面返回到主界面。

3. 编程指南

本章介绍了在应用程序开发时如何使用 POS SDK For IOS 编写程序。

3.1 连接打印机

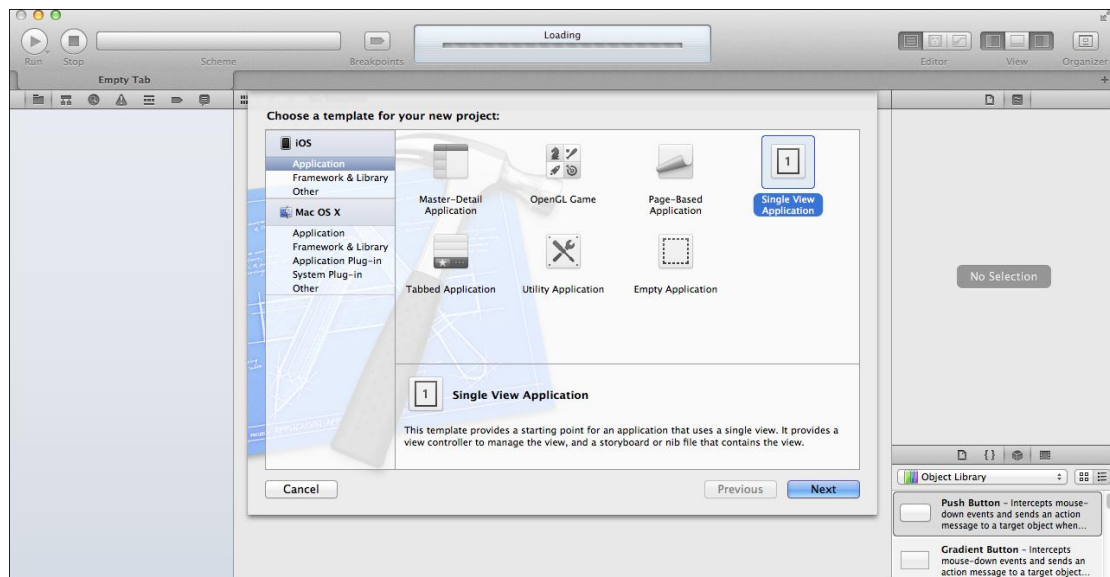
使用 WIFI 或 BlueTooth 或 BlueTooth(MFI)将打印机连接到 IOS 设备，详情请参阅：[建立连接](#)。

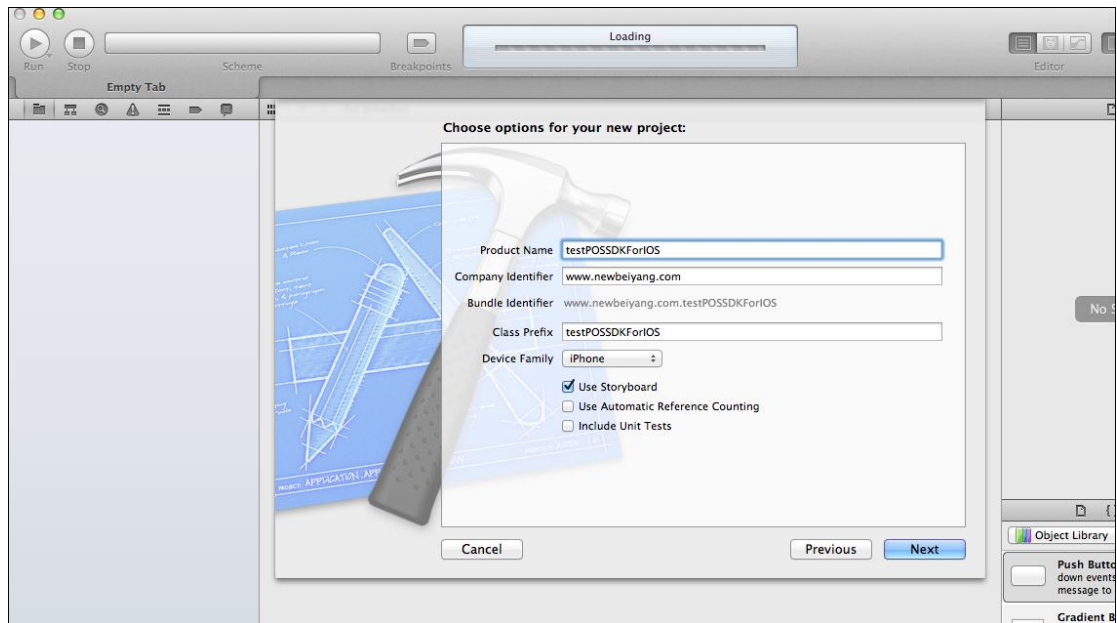
3.2 使用静态库

● 如何添加静态库文件

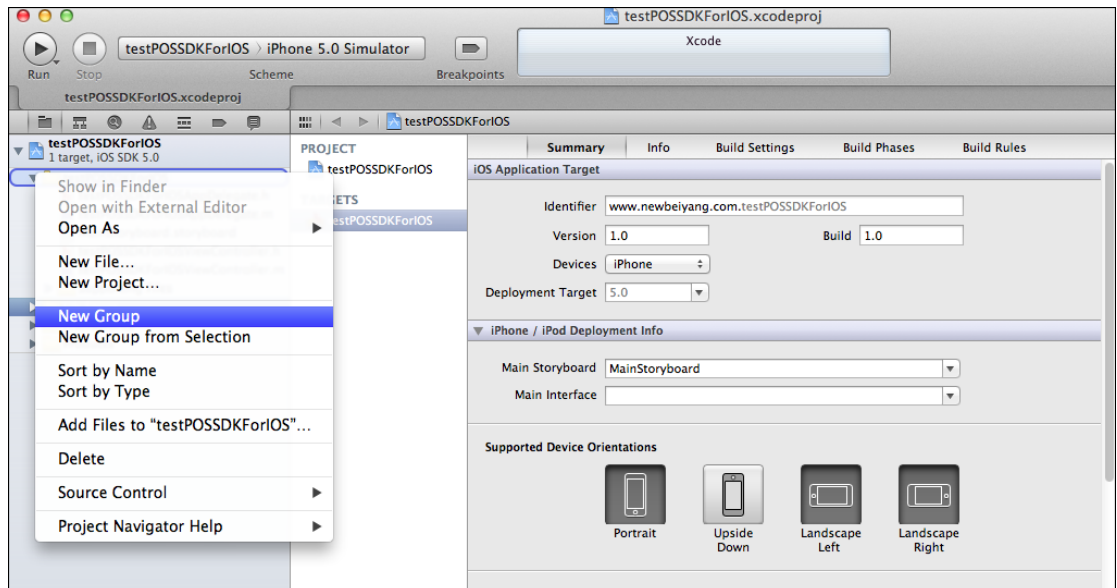
使用如下步骤添加静态库文件：

1) 创建一个新的 IOS 工程，以 “testPOSSDKForIOS “ 命名为例；

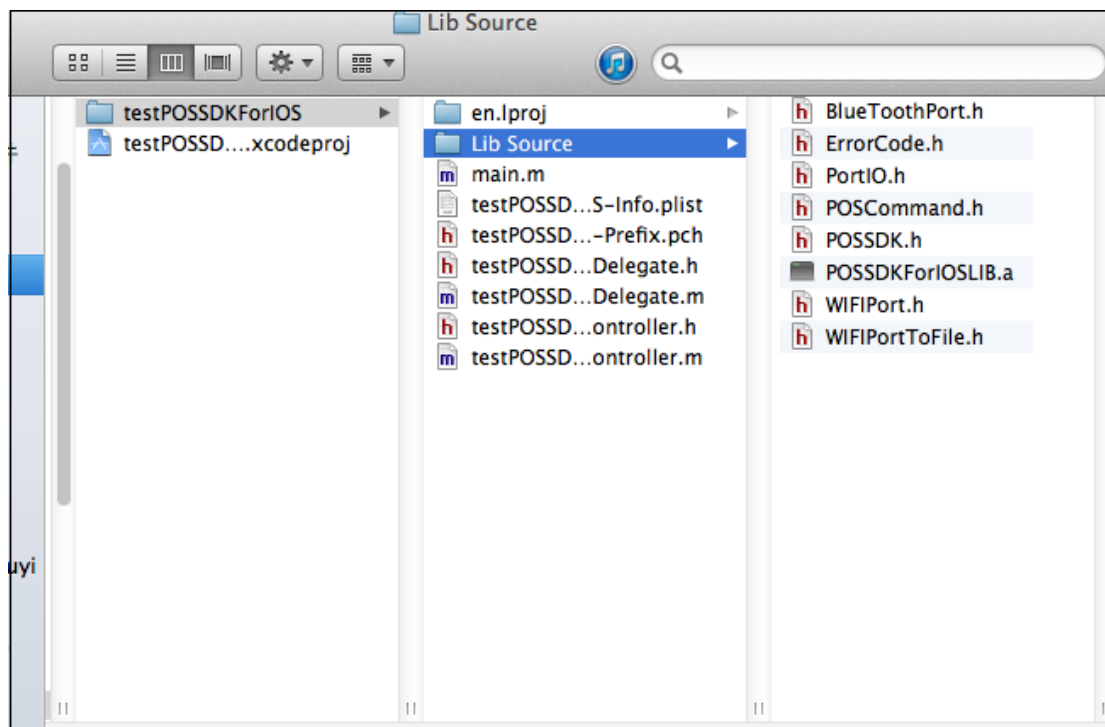




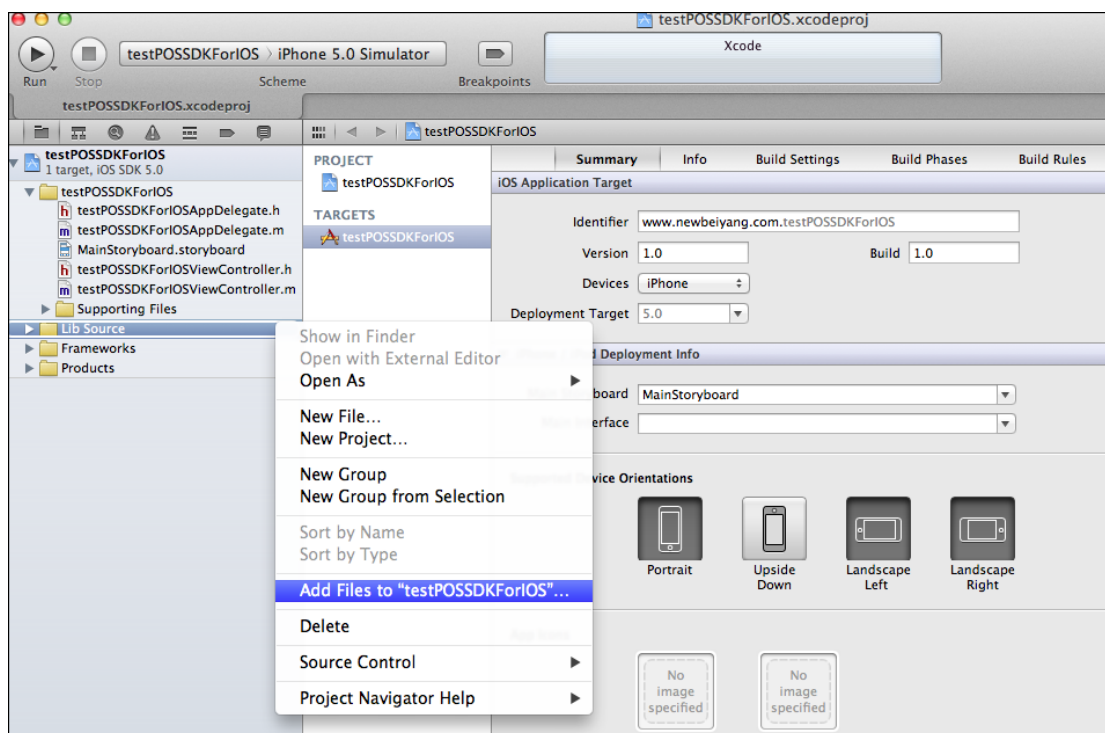
2) 在工程中新建组（New Group），并更改组名（Lib Source）；

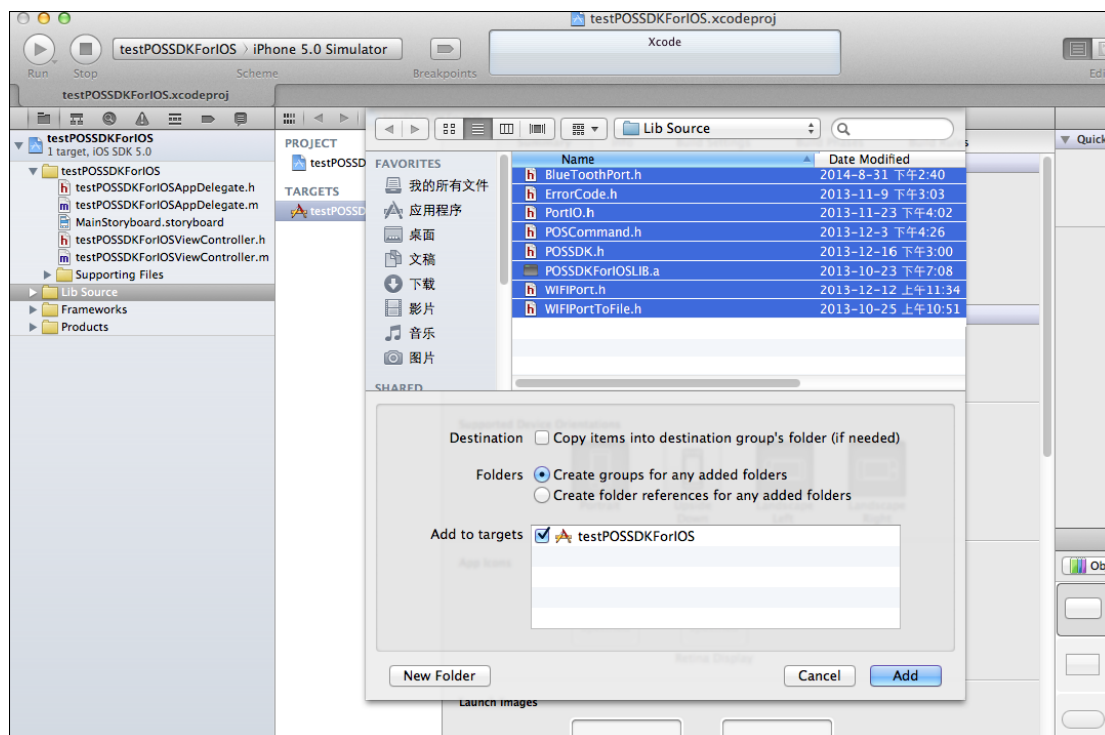


3) 在 testPOSSDKForIOS 文件中创建 Lib Source 文件夹，并将 POSSDKForIOSLIB.a、ErrorCode.h、PortIO.h、WiFiPort.h、WiFiPortToField.h、POSSDK.h、BluetoothPort.h 拷贝到 Lib Source 文件夹；



4) 在工程中将上述.a 、.h 文件添加进 Lib Source 中，点按“Add”键。





● 如何使用静态库

1) 为在 testPOSSDKForIOSViewController 中使用静态库及方法，你需要导入方法的头文件到 testPOSSDKForIOSViewController.h，例如：

```
#import "PortIO.h"
#import "WIFIPort.h"
#import "BluetoothPort.h"
#import "POSSDK.h"
```

2) 你需要在 testPOSSDKForIOSViewController.m 文件中声明类的属性，例如：

```
@interface testPOSSDKForIOSViewController()
@property (nonatomic, retain) POSSDK *pos_sdk;
@property (nonatomic, retain) WIFIPort *printer_port;
@property (nonatomic, retain) BluetoothPort *bluetooth_port;
@end
```

3) 你需要在 testPOSSDKForIOSViewController.m 文件中初始化类，例如：

```
@implementation testPOSSDKForIOSViewController
@synthesize pos_sdk = _pos_sdk;
@synthesize printer_port = _printer_port;
@synthesize bluetooth_port = _bluetooth_port;

- (POSSDK*)pos_sdk
{
    if(_pos_sdk == nil)
    {
        _pos_sdk = [[POSSDK alloc] init];
    }
    return _pos_sdk;
}
```

```

- (WiFiPort*)printer_port
{
    if(_printer_port == nil)
    {
        _printer_port = [[WiFiPort alloc] init];
    }
    return _printer_port;
}

- (BluetoothPort*)bluetooth_port
{
    if(_bluetooth_port == nil)
    {
        _bluetooth_port = [[BluetoothPort alloc] init];
    }
    return _bluetooth_port;
}

```

【说明】

WiFiPort 类也可以如下初始化：设置了 `init:false` 对于端口已成功连接之后断开 WIFI 网络或打印机断电的情况，在五分钟之内还可以继续发送约 128K 的数据，和上述 `[[WiFiPort alloc] init]` 初始化效果相同；设置了 `init:true` 在网络或打印机异常情况下，立刻返回不发送数据。

```

- (WiFiPort*)printer_port
{
    if(_printer_port == nil)
    {
        _printer_port = [[WiFiPort alloc] init:false];
    }
    return _printer_port;
}

```

● 方法举例**搜索打印机**

```

NSMutableArray *port_info_set = nil;
port_info_set = [self.printer_port searchPort];

```

连接打印机

```

error_code = [self.printer_port openPort: @ "192.168.1.200" Timeout: TIME_OUT_CONNECT];

```

关闭连接

```

error_code = [self.printer_port closePort];

```

设置标准模式参数

```

error_code = [self.pos_sdk systemSelectPrintMode:PrintModeStandard];
error_code = [self.pos_sdk standardModeSetStartingPosition:0];
error_code = [self.pos_sdk standardModeSetLeftMarginAndPrintAreaWidth:0 Width:640];

```

页模式下设置参数举例

```
error_code = [self.pos_sdk systemSelectPrintMode:PrintModePage];
error_code = [self.pos_sdk pageModeSetPrintArea:50 Y:0 AreaWidth:580 AreaHeight:600
PrintDirection:LeftToRight];
error_code = [self.pos_sdk pageModeSetStartingPosition:0 Y:100];
error_code = [self.pos_sdk pageModePrint];
error_code = [self.pos_sdk pageModeClearBuffer];
```

文本打印

```
NSString * text_content = nil;
text_content = @"SNBC strives to be the expert supplier of printers.Our POS series printers can
print text,image and barcode.It's a wise choice for you to choose SNBC printers.";
error_code = [self.pos_sdk textPrint:text_content];
```

文本光栅化打印

```
error_code = [self.pos_sdk textStandardModeRasterPrint: text_content FontName:@
“Thonburi-Bold” FontSize:34 LineBreakMode: UILineBreakModeWordWrap PrinterWidth:500];
```

读取第二磁道数据

```
#define size 256
UInt8 buffer[size] = {0};
[self.pos_sdk MsrReadMagneticDataForSecondTrack:buffer DataSize:size];
```

IC 卡控制 T0 协议

```
#define size 256
UInt8 buffer[size] = {0};
SInt32 commandLength = 5;
UInt8 command[5] = {0x00,0x84,0x00,0x00,0x04};
[self.pos_sdk ICControlT0: commandLength Command: command DataBuffer:buffer
DataSize:size];
```

用户自定义字符打印

```

SInt32 BytesOfHeight, DotsOfWidth, StartingCode = 'A', EndingCode = 'B';
NSMutableData *CharacterData = nil;
SInt32 index = 0;
UIImage *image = nil;
NSString *image_name[2] = {@"1.bmp", @"2.bmp"};
NSString *full_path = nil;

CharacterData = [[NSMutableData alloc] initWithCapacity:10240];
for(index = StartingCode; index <= EndingCode; index++)
{
    full_path = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:image_name[index - StartingCode]];
    image = [[UIImage alloc] initWithContentsOfFile:full_path];

    ImageDataRef image_data = {0};

    [self.pos_sdk imageFormatConvertToUserDefinedData:image ditheringSupported: FLSE
image_data:&image_data];

    BytesOfHeight = image_data.image_height >> 3;
    DotsOfWidth = image_data.image_width;
    [CharacterData appendBytes:image_data.data_buf length:image_data.image_data_len];

    if(image_data.data_buf != nil) {free(image_data.data_buf);}
    if(image != nil) {[image release];}
}

error_code = [self.pos_sdk textUserDefinedCharacterDefine:BytesOfHeight
DotsOfWidth:DotsOfWidth StartingCode:StartingCode EndingCode:EndingCode
CharacterData:CharacterData];

error_code = [self.pos_sdk textUserDefinedCharacterEnable:1];
error_code = [self.pos_sdk textPrint:@"123AACBB123"];

```

一维条码打印

```

NSData *data = nil;
///获取 NSData 类型的一维条码数据 data///
error_code = [self.pos_sdk barcodePrint1Dimension: data BarcodeType:BarcodeUPC_A
ModuleWidth:2 BarcodeHeight:100 HriFontType:FontStyleStandardASCII
HriPosition:HRIBelow];

```

PDF417 条码打印

```

NSData *PDF417_Data = nil;
///获取 NSData 类型的 PDF417 条码数据 PDF417_Data//
error_code = [self.pos_sdk barcodePrintPDF417: PDF417_Data AppearanceToHeight:1
AppearanceToWidth:1 RowNumber:3 ColumnNumber:1 XSize:4 LineHeight:15
CorrectionGrade:0];

```

QR 码打印

```

NSData *QR_Data = nil;
///获取 NSData 类型的 QR 码数据 QR_Data//
error_code = [self.pos_sdk barcodePrintQR:QR_Data BasicElementWidth:5
SymbolType:EnhancedType LanguageMode:LanguageChinese];

```

Maxicode 条码打印

```
NSData *Maxicode_Data = nil;
///获取 NSData 类型的 Maxicode 码数据 Maxicode_Data//
error_code = [self.pos_sdk barcodePrintMaxicode: Maxicode_Data];
```

GS1 DataBar 和 GS1 复合条码打印

```
NSData *GS1_Data = nil;
///获取 NSData 类型的条码数据//
error_code = [self.pos_sdk barcodePrintGS1DataBar: GS1_Data
BarcodeType:GS1DataBar_Omnidirectional BasicElementWidth:3 BarcodeHeight:100
BasicElementHeight:4 SeparatorHeight:1 SegmentNumber:2 HRI:DataBarAnd2DHri UseAI:1];
```

位图打印

```
UIImage *image = nil;
///依据图像名称获取 UIImage 类型图像 image//
error_code = [self.pos_sdk imageStandardModePrint: SingleDensity_8 Image:image
StartHorPos:0 PrinterWidth:500];
```

RAM/Flash 位图下载及打印

RAM

```
error_code = [self.pos_sdk imageDownloadToPrinterRAM:0 Image: image PrinterWidth:500];
error_code = [self.pos_sdk imageRAMPrint:0 Mode: NormalMode];
```

Flash

```
NSMutableArray *image_set = nil;
NSString *flash_image_name = nil;
NSString *full_path_Flash = nil;
UIImage *image_Flash = nil;
flash_image_name = @"SNBC.bmp,Jpg.jpg,face.PNG";
list = [flash_image_name componentsSeparatedByString:@","];
image_set = [[NSMutableArray alloc] initWithCapacity:[list count]];
for(index = 0; index < [list count]; index++)
{
    full_path_Flash = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:[list objectAtIndex:index]];
    image_Flash = [[UIImage alloc] initWithContentsOfFile:full_path_Flash];
    [image_set addObject:image_Flash];
    [image_Flash release];
}
error_code = [self.pos_sdk imageDownloadToPrinterFlash:image_set PrinterWidth:500];
error_code = [self.pos_sdk imageFlashPrint:1 Mode: Double_width];
error_code = [self.pos_sdk imageFlashPrint:2 Mode: Double_height];
error_code = [self.pos_sdk imageFlashPrint:3 Mode: Quadruple];
```

光栅化位图打印

```
error_code = [self.pos_sdk imageStandardModeRasterPrint:image PrinterWidth:500];
```

文件下载

```
NSString* full_path = nil;
full_path = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:@"sample1.dat"];
[self.pos_sdk systemDownloadFile:full_path];
```

4. API 相关

本章描述在 POS SDK For IOS 中提供的 API 接口函数。

4.1 通讯相关

API	描述
searchPort	搜索打印机
openPort	建立与打印机连接
closePort	关闭与打印机连接
writePort	向端口发送数据
writePort: PortID	向端口发送数据
readPort	从端口读取数据
readPort: PortID	从端口读取数据
recordCommunicationDataEnable	数据记录功能使能

● searchPort

搜索打印机并获取成功搜索到的打印机的信息。

函数

- (NSMutableArray*)searchPort

返回值

返回值	情况
nil	搜索不到打印机
搜索到的打印机的信息列表	成功搜索

打印机的信息列表如下：

```
@interface PortInfoWIFI : NSObject
{
    SInt32      NetPortType;
    NSString    *ModuleName;
    NSString    *SSID;
    NSString    *IPAddr;
    NSString    *Gateway;
    NSString    *MACAddr;
    NSString    *ExpendInfo;
}
```

示例代码

见方法举例中的[搜索打印机](#)。

● openPort

建立与打印机的连接。

函数

- (SInt32)**openPort**:(NSString*)DeviceName TimeOut:(SInt32)TimeOut
- (SInt32)**openPort**:(NSString*)DeviceName
- (SInt32)**openPort**:(EAAccessory*)accessory protocolString:(NSString*)

protocolString

参数

- **DeviceName** 打印机的 IP 地址或 BLE 模式蓝牙设备名称
- **TimeOut** 建立连接的超时时间，合法值为非负整数，单位毫秒，我们为客户定义了几个常用的超时时间如下表：

取值	描述
TIME_OUT_DEFAULT	超时 0ms
TIME_OUT_CONNECT	超时 3000ms
TIME_OUT_SMALL_DATA	超时 1000ms
TIME_OUT_LARGE_DATA	超时 4000ms

- **accessory** MFI 蓝牙设备节点
- **protocolString** 协议名称

返回值

返回值	情况
SUCCESS	成功
ERR_CREATE_CONNECTION	创建套接字失败、超时时间内未等到可用资源、获取套接字失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

见方法举例中的[连接打印机](#)。

● closePort

关闭与打印机连接。

函数

- (SInt32)closePort

返回值

返回值	情况
SUCCESS	成功

示例代码

见方法举例中的[关闭连接](#)。

● writePort

向端口发送数据，数据包大于 4096 字节时拆包为 4096 字节。

函数

- (SInt32)writePort:(const UInt8*)WriteBuffer OffsetSize:(SInt32)OffsetSize

WriteSize:(SInt32)WriteSize WriteTimeOut:(SInt32)WriteTimeOut

参数

- **WriteBuffer** 待发送数据的存储空间地址，非空
- **OffsetSize** 首个发送字节在 WriteBuffer 中的偏移量，非负整数
- **WriteSize** 发送字节数
- **WriteTimeOut** 超时时间，单位毫秒，非负整数

返回值

返回值	情况
发送数据的字节数	成功
ERR_INVALID_CONNECTION	未成功建立连接
ERR_INVALID_ARGUMENT	参数错误
ERR_COMMUNICATE	设置超时时间失败

示例代码

```
data[2] = {0x1b,0x40};
[self.printer_port writePort:data offSize:0 WriteSize:2 WriteTimeOut:
TIME_OUT_SMALL_DATA];
```

【说明】

a) 参数 OffsetSize 和 WriteSize 之和不允许大于待发送数据的数据长度。

b) 在端口已连接状态再断开网络或关闭打印机的情况，详见如何[使用静态库中的【说明】](#)。

● writePort: PortID

向端口发送数据，数据包大于 4096 字节时拆包为 4096 字节。

函数

- (SInt32)writePort:(const UInt8*)WriteBuffer OffsetSize:(SInt32)OffsetSize
WriteSize:(SInt32)WriteSize WriteTimeOut:(SInt32)WriteTimeOut
PortID:(SInt32)PortID

参数

- **WriteBuffer** 待发送数据的存储空间地址，非空
- **OffsetSize** 首个发送字节在 WriteBuffer 中的偏移量，非负整数
- **WriteSize** 发送字节数
- **WriteTimeOut** 超时时间，单位毫秒，非负整数
- **PortID** 选择端口

PortID 取值	描述
NET_PORT_COMMAND	NET_PORT_ID_COMMAND, 选择 9100 端口
NET_PORT_QUERY	NET_PORT_ID_QUERY, 选择 4000 端口
其他	参数错误

返回值

返回值	情况
发送数据的字节数	成功
ERR_INVALID_CONNECTION	未成功建立连接
ERR_COMMUNICATE	设置超时时间失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.printer_port writePort:data offSize:0 WriteSize:2 WriteTimeOut:  
TIME_OUT_SMALL_DATA PortID: NET_PORT_COMMAND];
```

【说明】

- a) 参数 OffsetSize 和 WriteSize 之和不允许大于待发送数据的数据长度。
- b) 在端口已连接状态再断开网络或关闭打印机的情况，详见如何[使用静态库中的【说明】](#)。

● readPort

从端口读取数据。

函数

- (SInt32)**readPort**:(UInt8*)ReadBuffer OffsetSize:(SInt32)OffsetSize
ReadSize:(SInt32)ReadSize ReadTimeOut:(SInt32)ReadTimeOut

参数

- **ReadBuffer** 接收数据缓冲区，非空
- **OffsetSize** 首个接收字节在 ReadBuffer 中的偏移量，非负整数
- **ReadSize** 需要读取的字节数
- **ReadTimeOut** 读超时时间，单位毫秒，非负整数

返回值

返回值	情况
读取的字节数	成功
ERR_INVALID_CONNECTION	未成功建立连接
ERR_INVALID_ARGUMENT	参数错误
ERR_ALLOC_MEMORY	申请空间失败
ERR_COMMUNICATE	设置超时时间失败或实际读取的字节数为 0

示例代码

```
SInt32 read_size;
UInt8 pointBuffer[512];
[self.printer_port readPort: pointBuffer offSize:0 ReadSize: read_size WriteTimeOut:
TIME_OUT_SMALL_DATA];
```

【说明】

- a) 参数 OffsetSize 和 ReadSize 之和不允许大于待接收数据的数据长度。

● readPort:PortID

从端口读取数据。

函数

- (SInt32)**readPort**:(UInt8*)ReadBuffer OffsetSize:(SInt32)OffsetSize
 ReadSize:(SInt32)ReadSize ReadTimeOut:(SInt32)ReadTimeOut
 PortID:(SInt32)PortID

参数

- **ReadBuffer** 接收数据缓冲区，非空
- **OffsetSize** 首个接收字节在 ReadBuffer 中的偏移量，非负整数
- **ReadSize** 需要读取的字节数
- **ReadTimeOut** 读超时时间，单位毫秒，非负整数
- **PortID** 读取数据端口

PortID 取值	描述
NET_PORT_COMMAND	NET_PORT_ID_COMMAND, 选择 9100 端口
NET_PORT_QUERY	NET_PORT_ID_QUERY, 选择 4000 端口
其他	参数错误

返回值

返回值	情况
读取的字节数	成功
ERR_INVALID_CONNECTION	未成功建立连接
ERR_INVALID_ARGUMENT	参数错误
ERR_ALLOC_MEMORY	申请空间失败
ERR_COMMUNICATE	设置超时时间失败或实际读取的字节数为 0

示例代码

```
[self.printer_port readPort: pointBuffer offSize:0 ReadSize: read_size WriteTimeOut:
TIME_OUT_SMALL_DATA PortID: NET_PORT_QUERY];
```

【说明】

- 参数 OffsetSize 和 ReadSize 之和不允许大于待接收数据的数据长度。

● recordCommunicationDataEnable

记录数据功能使能。

函数

- (SInt32)**recordCommunicationDataEnable**:(NSString*)recordFileName

参数

- **recordFileName** 记录文件的文件名

返回值

返回值	情况
SUCCESS	成功
其他	失败

示例代码

```
[self.printer_port recordCommunicationDataEnable: @ "DataFile.dat"];
```

4.2 POS SDK API 相关

前缀	API	Description
前缀 system 系统相关 函数	systemSetPortIO	传入已建立连接的端口对象
	systemSetEncoding	设置文本打印的编码格式
	systemDownloadFile	下载文件
	systemReset	初始化打印机，清除打印缓冲区数据，打印模式被设为上电时的默认值模式。
	systemSelectPrintMode	选择打印模式
	systemSelectPaperType	选择纸张类型
	systemSetMotionUnit	设置横纵向可移动单位
	systemQueryStatus	查询打印机状态
	systemFeedLine	打印机进纸
	systemCutPaper	选择切纸模式并切纸
前缀 cashdrawer 钱箱	cashdrawerOpen	产生钱箱控制脉冲，输出到指定引脚
前缀 text 文本相关 函数	textSelectCharSetAndCodePage	选择字符集和代码页
	textSetLineHeight	设置文本行高
	textSetCharacterSpace	设置字符间距
	textStandardModeAlignment	设置文本对齐方式(只在标准模式下的行首有效)
	textStandardModeUpsideDown	选择是否倒置打印（只在标准模式的行首有效）
	textPrint: String	打印字符串(NSStrng*)

	textPrint: Buffer	打印字符(UInt8*)
	textSelectFontMagnifyTimes	选择横纵向放大倍数
	textStandardModeRotate	标准模式下文本旋转打印
	textSelectFont	选择打印字符的字体及字体风格
	textSetColorPrint	设置双色打印, 并选择打印颜色
	textQuitColorPrint	退出双色打印
	textUserDefinedCharacterEnable	用户自定义字符功能使能
	textUserDefinedCharacterDefine	用户自定义字符下载
	textUserDefinedCharacterCancel	取消用户自定义的字符
	textUserDefinedChineseCharacterDefine	用户自定义汉字
	textStandardModeRasterPrint	文本光栅化打印
前缀 image 图像相关 函数	imageStandardModePrint	标准模式下打印图像
	imageDownloadToPrinterRAM	RAM 图像下载
	imageRAMPrint	RAM 图像打印
	imageDownloadToPrinterFlash	Flash 图像下载
	imageFlashPrint	Flash 图像打印
	imageStandardModeRasterPrint	标准模式下光栅位图打印
	imageFormatConvertToUserDefinedData	将图像转化为适用于用户自定义的图像格式
前缀 barcode 条码相关 函数	barcodeGetIDByName	依据条码 ID 获取条码名称
	barcodeGetNameByID	依据条码名称获取条码 ID
	barcodePrint1Dimension	选择一维条码类型
	barcodePrintQR	打印 QR 码
	barcodePrintPDF417	打印 PDF417 条码
	barcodePrintMaxicode	打印 Maxicode 条码
	barcodePrintGS1DataBar	打印 GS1 DataBar 条码和 GS1 复合码
前缀 standard Mode 标准模式	standardModeSetLeftMarginAndPrintAreaWidth	设置标准模式的打印左边距及打印区域宽度
	standardModeSetHorizontalStartingPosition	设置标准模式的横向起始坐标
前缀 pageMode	pageModeSetStartingPosition	设置页模式的起始坐标
	pageModeSetPrintArea	设置页模式的打印区域

页模式	pageModePrint	页模式打印
	pageModeClearBuffer	页模式清空 Buffer
前缀 Msr 读磁	MsrReadMagneticForFirstTrack	读取第一磁道的数据
	MsrReadMagneticForSecondTrack	读取第二磁道的数据
	MsrReadMagneticForThirdTrack	读取第三磁道的数据
	MsrReadMagneticForThreeTracks	读取第四磁道的数据
前缀 IC	ICRest	复位 IC 卡
	ICControlT0	IC 卡控制命令 T0 协议
	ICControlT1	IC 卡控制命令 T1 协议
无	通知名称 kNotificationBLE	蓝牙状态返回

● systemSetPortIO

设置打印机通讯模块实例。

函数

- (SInt32)systemSetPortIO:(PortIO *)Port_IO

参数

- **Port_IO** 已与打印机建立通讯的实例

返回值

返回值	情况
SUCCESS	成功
ERR_INVALID_ARGUMENT	Port_IO 不为 PortIO 对象或对象实例无效

示例代码

```
[self.pos_sdk systemSetPortIO:self.printer_port];
```

● systemSetEncoding

设置文本打印的编码格式。

函数

- (SInt32)systemSetEncoding:(CFStringEncoding)encoding

参数

- **encoding** IOS 提供的字符的编码格式, 参数取值请参考 IOS 系统提供的

头文件 CFStringEncodingExt.h

返回值

返回值	情况
SUCCESS	成功

示例代码

```
[self.pos_sdk systemSetEncoding: kCFStringEncodingGB_18030_2000];//设置为中文的编码格式
```

● systemDownloadFile

下载文件。

函数

- (SInt32)systemDownloadFile:(NSString*)FileName

DownloadTimeOut:(SInt32)DownloadTimeOut

参数

- **FileName** 下载文件的文件名
- **DownloadTimeOut** 等待时间，单位：秒。可以选择 1-8 秒，如果被指定 8 秒以上，则按照 8 秒执行

返回值

返回值	情况
SUCCESS	成功
ERR_COMMUNICATE	失败
ERR_INVALID_ARGUMENT	文件名错误或文件数据为空

示例代码

```
NSString * full_path = [[[NSBundle mainBundle] bundlePath]
stringByAppendingPathComponent:@"sample.dat"];
[self.pos_sdk systemDownloadFile:full_path DownloadTimeOut:3];
```

● systemReset

初始化打印机，清除打印缓冲区数据，打印模式被设为上电时的默认值模式。

函数

- (SInt32)systemReset

返回值

返回值	情况
SUCCESS	成功
ERR_SYSTEM_RESET	初始化打印机失败

示例代码

```
[self.pos_sdk systemReset];
```

● systemSelectPrintMode

选择打印模式。

函数

- (SInt32)systemSelectPrintMode:(SInt32)PrintMode

参数

- PrintMode 打印模式

PrintMode 取值	描述
PrintModeStandard	标准模式
PrintModePage	页模式
其他	参数错误

返回值

返回值	情况
SUCCESS	成功
ERR_SYSTEM_SELECT_PRINT_MODE	选择打印模式失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk systemSelectPrintMode: PrintModeStandard];//标准模式
[self.pos_sdk systemSelectPrintMode:PrintModePage];//页模式
```

● systemSelectPaperType

选择纸张类型。

函数

- (SInt32)systemSelectPaperType:(SInt32)PaperType

参数

- **PaperType** 纸张类型

PaperType 取值	描述
PaperTypeCoutinuous	连续纸
PaperTypeMarked	标记纸
其他	参数错误

返回值

返回值	情况
SUCCESS	成功
ERR_SYSTEM_SELECT_PAPER_TYPE	选择纸张类型失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk systemSelectPaperType: PaperTypeCoutinuous];//连续纸  
[self.pos_sdk systemSelectPaperType: PaperTypeMarked];// 标记纸
```

● **systemSetMotionUnit**

设置横纵向移动单位。

函数

- (SInt32)**systemSetMotionUnit**:(SInt32)HorizontalUnit

VerticalUnit:(SInt32)VerticalUnit

参数

- **HorizontalUnit** 横向移动单位

HorizontalUnit 取值	描述
0-255	合法值
其他	参数错误

- **VerticalUnit** 纵向移动单位

HorizontalUnit 取值	描述
0-255	合法值
其他	参数错误

返回值

返回值	情况
-----	----

SUCCESS	成功
ERR_SYSTEM_SET_MOTION_UNIT	设置横纵向移动单位失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk systemSetMotionUnit:203 VerticalUnit:203];
```

● systemQueryStatus

查询打印机状态。

函数

-(SInt32)systemQueryStatus:(UInt8*)QueryStatusBuffer

ReadSize:(SInt32)ReadSize

参数

- **QueryStatusBuffer** 查询状态的缓冲区
- **ReadSize** 读入数据的字节数

返回值

返回值	情况
SUCCESS	成功
ERR_SYSTEM_QUERY_STATUS	查询打印机状态失败
ERR_COMMUNICATE	返回的字节数和要读取的字节数不相同

示例代码

```
#define QueryStatusSize 4
UInt8    StatusBuffer[QueryStatusSize] = {0};
[self.pos_sdk systemQueryStatus:StatusBuffer ReadSize:QueryStatusSize];
```

● systemFeedLine

打印机进纸。

函数

-(SInt32)systemFeedLine:(SInt32)LineNum

参数

- **LineNum** 走纸行数

LineNum 取值	描述
1-255	合法值
其他	参数错误

返回值

返回值	情况
SUCCESS	成功
ERR_SYSTEM_FEED_LINE	打印机进纸失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
error_code = [self.pos_sdk systemFeedLine:5];
```

● systemCutPaper

选择切纸模式，打印机进纸并切纸。

函数

- (SInt32)systemCutPaper:(SInt32)CutMode

FeedDistance:(SInt32)FeedDistance

参数

- **CutMode** 走纸行数

LineNum 取值	描述
CutFullImmdediately	全切
CutPartImmdediately	半切
CutPartAfterFeed	走纸并半切
其他	参数错误

- **FeedDistance** 走纸距离

FeedDistance 取值	描述
0-255	走纸距离合法值
其他	参数错误

【说明】

a) 如果参数为 CutFullImmdediately 和 CutPartImmdediately 时，走纸距离参数 FeedDistance 被忽略。

b) 如果参数为 CutPartAfterFeed 时，打印机走纸 FeedDistance 并切纸。

c) 标记纸模式时，走纸距离被忽略，打印机搜索标记并切纸。

d) 对于无切刀的机型此接口无效。

返回值

返回值	情况
SUCCESS	成功
ERR_SYSTEM_CUT_PAPER	打印机切纸失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
error_code = [self.pos_sdk systemCutPaper:CutPartAfterFeed FeedDistance:80];
```

● cashdrawerOpen

产生钱箱控制脉冲，输出到指定引脚。

函数

- (SInt32)cashdrawerOpen:(SInt32)CashdrawerID

PulseOnTimes:(SInt32)PulseOnTimes PulseOffTimes:(SInt32)PulseOffTimes

参数

- **CashdrawerID** 钱箱引脚

CashdrawerID 取值	描述
0	钱箱插座的引脚 2
1	钱箱插座的引脚 5
其他	参数错误

- **PulseOnTimes** 钱箱开启脉冲高电平时间

PulseOnTimes 取值	描述
0-255	合法值
其他	参数错误

- **PulseOffTimes** 钱箱开启脉冲低电平时间

PulseOffTimes 取值	描述
0-255	合法值
其他	参数错误

返回值

返回值	情况
SUCCESS	成功

ERR_CASH_DRAWER_OPEN	钱箱弹开失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk cashdrawerOpen:0 PulseOnTimes:100 PulseOffTimes:100];
```

● textSelectCharSetAndCodePage

选择国际字符集和代码页。

函数

- (SInt32)textSelectCharSetAndCodePage:(SInt32)CharSet
codePage:(SInt32)CodePage

参数

- **CharSet** 国际字符集

CharSet 取值	描述
CharacterSetUSA	美国 (U.S.A)
CharacterSetFrance	法国 (France)
CharacterSetGermany	德国 (Germany)
CharacterSetUK	英国 (U.K)
CharacterSetDenmark_I	丹麦 I (Denmark I)
CharacterSetSweden	瑞典 (Sweden)
CharacterSetItaly	意大利 (Italy)
CharacterSetSpain_I	西班牙 I (Spain I)
CharacterSetJapan	日本 (Japan)
CharacterSetNorway	挪威 (Norway)
CharacterSetDenmark_II	丹麦 II (Denmark II)
CharacterSetSpain_II	西班牙 II (Spain II)
CharacterSetLatin_America	拉丁美洲 (Latin America)
CharacterSetKorea	韩国 (Korea)
其他	参数错误

- **CodePage** 国际字符集

CodePage 取值	描述
CodePagePC437	PC437
CodePageKatakana	Katakana

CodePagePC850	PC850
CodePagePC860	PC860
CodePagePC863	PC863
CodePagePC865	PC865
CodePage851_Greek	851[Greek]
CodePagePC857	PC857
CodePage737_Greek	737[Greek]
CodePage928_Greek	928[Greek]
CodePageWPC125	WPC1252
CodePagePC866	PC866
CodePagePC852	PC852
CodePagePC858	PC858
CodePageThaiTis42_Thai3	Thai Tis42(Thai3)
CodePageThaiTis11_Thai5	Thai Tis11(Thai5)
CodePageThaiTis_Thai2	Thai Tis(Thai2)
CodePageThaiKu_Thai1	Thai Ku(Thai1)
CodePageThaiTis14_Thai4	Thai Tis14(Thai4)
CodePageThaiTis18_Thai6	Thai Tis18(Thai6)
CodePageHebrew1	Hebrew1
CodePageHebrew2	Hebrew2
CodePageHebrew3	Hebrew3
CodePage775_Baltic	775[Baltic]
CodePage855_Cyrillic	855[Cyrillic]
CodePage862_hebrew	862[hebrew]
CodePage864_Arabic	864[Arabic]
CodePage869_Greek	869[Greek]
CodePageFrasI	FrasI
CodePage772_Lithuanian	772[Lithuanian]
CodePage1250_Latin_2	1250[Latin-2]
CodePage1251_Cyrillic	1251[Cyrillic]
CodePage1253_Greek	1253[Greek]
CodePage1254_Turkish	1254[Turkish]
CodePage1255_Hebrew	1255[Hebrew]
CodePage1256_Arabic	1256[Arabic]

CodePage1257_Baltic	1257[Baltic]
CodePage771	771
CodePage774_Lithuanian	774[Lithuanian]
CodePage3840_IBM_Russian	3840 (IBM-Russian)
CodePage3841_Gost	3841 (Gost)
CodePage3843_Polish	3843 (Polish)
CodePage3844_CS2	3844 (CS2)
CodePage3845_Hungarian	3845 (Hungarian)
CodePage3846_Turkish	3846 (Turkish)
CodePage3847_Brazil_ABNT	3847 (Brazil-ABNT)
CodePage3848_Brazil_ABICOMP	3848 (Brazil-ABICOMP)
CodePage1001	1001
CodePage2001	2001
CodePage3001_Estonian_1	3001 (Estonian-1)
CodePage3002_Estonian_2	3002 (Estonian-2)
CodePage3011_Latvian_1	3011 (Latvian-1)
CodePage3012_Latvian_2	3012 (Latvian-2)
CodePage3021_Bulgarian	3021 (Bulgarian)
CodePage3041_Maitese	3041 (Maltese)
CodePage8859	8859
CodePagePersia	Persia

【说明】

a) 几种类型的打印机可能不能支持所有的代码页。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_SELECT_CHAR_SET	选择字符集失败
ERR_TEXT_SELECT_CODE_PAGE	选择代码页失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```

UInt8 textdata[MAX_COMMAND_LENGTH] = {0x80};
[self.pos_sdk textSelectCharSetAndCodePage: CharacterSetUSA codePage: CodePagePC437];
[self.pos_sdk textPrint:@ “#$@{}[]~”];
[self.pos_sdk textPrint: textdata Length:1];

```



```
error_code = [self.pos_sdk systemFeedLine:1];
```

● textSetLineHeight

设置行高。

函数

- (SInt32)textSetLineHeight:(SInt32)Height

参数

- **Height** 行高点数

Height 取值	描述
0-255	合法值
其他	参数错误

【说明】

- 如果参数 Height 为 0 时，行高将设置为默认值(1/6 inch)。
- 如果参数 Height 小于字符的高度，打印机将设置行高为字符的高度值。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_SET_LINE_HEIGHT	设置行高失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk textSetLineHeight:34];
```

● textSetCharacterSpace

设置字符间距。

函数

- (SInt32)textSetCharacterSpace:(SInt32)LeftSpace

RightSpace:(SInt32)RightSpace Mode:(SInt32)Mode

参数

- **LeftSpace** 字符左间距（设置中文字符间距时，此参数需有效）

LeftSpace 取值	描述
--------------	----

0-255	合法值
其他	参数错误

- **RightSpace** 字符右间距

RightSpace 取值	描述
0-255	合法值
其他	参数错误

- **Mode** 字符模式

Mode 取值	描述
ChineseCharacterMode	中文字符模式
EnglishCharacterMode	英文字符模式
其他	参数错误

【说明】

a) 如果参数 Mode 为 ChineseCharacterMode 时, 参数 LeftSpace 和 RightSpace 必须同时合法, 如果参数 Mode 为 EnglishCharacterMode 时, 只要参数 RightSpace 合法便可, 此时参数 LeftSpace 被忽略。

b) 调用一次函数 textSetCharacterSpace: 只可以单独更改英文或中文的字符间距, 若想更改中英文字符间距需要调用函数两次。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_SET_CHARACTER_SPACE	设置字符间距失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk textSetCharacterSpace:10 RightSpace:50 Mode:EnglishCharacterMode];
```

● textStandardModeAlignment

设置文本打印的对齐方式。

函数

-(SInt32)textStandardModeAlignment:(SInt32)Alignment

参数

- **Alignment** 对齐方式

Alignment 取值	描述
--------------	----

TextAlignmentLeft	左对齐
TextAlignmentCenter	居中
TextAlignmentRight	右对齐
其他	参数错误

【说明】

- a) 该函数只在标准模式的行首有效。
- b) 该函数也对一二维条码有效。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_STANDARD_MODE_ALIGNMENT	设置文本打印的对齐方式失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk textStandardModeAlignment: TextAlignmentLeft];
```

● textStandardModeUpsideDown

选择是否倒置打印（只在标准模式的行首有效）。

函数

- (SInt32)textStandardModeUpsideDown:(SInt32)UpsideDown

参数

- **UpsideDown** 选择是否倒置打印

UpsideDown	取值	描述
FontStyleUpsideDown		倒置打印
其他		不倒置打印

【说明】

- a) 调用 textStandardModeUpsideDown: 的参数 UpsideDown 为 FontStyleUpsideDown 的打印效果和调用
- (SInt32)textStandardModeRotate:(SInt32)Rotate 将参数 Rotate 设置为 RotatePrint180 效果相同。

b) 先调用 textStandardModeUpsideDown: 再调用 textStandardModeRotate, 结果同旋转结果; 先调用旋转后调用倒置, 可能会出现未知结果。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_STANDARD_MODE_UPSIDEDOWN	选择是否倒置打印失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk textStandardModeUpsideDown: FontStyleUpsideDown];
```

● **textPrint: String**

打印字符串。

函数

- (SInt32)**textPrint**:(NSString*)String

参数

- **String** 要打印的字符串

【说明】

- a) 如果想调用本函数打印英文以外的字符，如中文，需要首先调用函数 `systemSetEncoding` 来设置文本打印的编码格式。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_PRINT	打印文本失败
ERR_INVALID_ARGUMENT	字符串空

示例代码

```
[self.pos_sdk textPrint:@ "123456SNBC"];
```

● **textPrint: Buffer**

打印字符。

函数

- (SInt32)**textPrint**:(UInt8*)Buffer Length:(SInt32)Length

参数

- **Buffer** 要打印的字符数组
- **Length** 要打印的数据长度

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_PRINT	打印文本失败
ERR_INVALID_ARGUMENT	字符数组为空或 Length 小于等于 0

示例代码

```
UInt8 textdata[MAX_COMMAND_LENGTH] = {0x80};
[self.pos_sdk textPrint: textdata Length:1];
```

● textSelectFontMagnifyTimes

设置字符大小。

函数

- (SInt32)textSelectFontMagnifyTimes:(SInt32)HorizontalTimes

VerticalTimes:(SInt32)VerticalTimes

参数

- **HorizontalTimes** 横向放大倍数

HorizontalTimes	取值	描述
1-6		合法横向放大倍数
其他		非法参数

- **VerticalTimes** 纵向放大倍数

VerticalTimes	取值	描述
1-6		合法纵向放大倍数
其他		非法参数

【说明】

a) 在标准模式下，纵向是进纸方向，横向是垂直于进纸的方向。但是当字符顺时针旋转 90°时，横向和纵向颠倒。

b) 页模式下，横向和纵向取决于区域的方向。

c) 同一行字符的放大倍数不同时，所有的字符以底线对齐。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_SELECT_MAGNIFY_TIMES	设置字符大小失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk textSelectFontMagnifyTimes:2 VerticalTimes:3];
```

● textStandardModeRotate

设置字符旋转打印的旋转度数。

函数

- (SInt32)**textStandardModeRotate**:(SInt32)Rotate

参数

- **Rotate** 设置旋转度数

Rotate 取值	描述
RotatePrintNormal	不旋转
RotatePrintR90	顺时针旋转 90 度
RotatePrint180	旋转 180 度
RotatePrintL90	逆时针旋转 90 度
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_STANDARD_MODE_ROTATE	设置字符旋转打印失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk textStandardModeRotate: RotatePrintR90];// 顺时针旋转 90 度
```

● textSelectFont

选择打印字符的字体及字体风格。

函数

- (SInt32)**textSelectFont**:(SInt32)FontType FontStyle:(SInt32)FontStyle

参数

- **FontType** 字体

FontType	取值	描述
FontTypeStandardASCII		标准 ASCII
FontTypeCompressedASCII		压缩 ASCII
FontTypeUserDefined		用户自定义字符
FontTypeChinese		中文字符
其他		参数错误

- **FontStyle** 字体风格。允许字体风格参数进行叠加，但同时设置参数 FontStyle 为 FontStyleUnderlineOneDotThick 和 FontStyleUnderlineTwoDotThick 时，打印两点粗下划线效果。

FontStyle	取值	描述
FontStyleReverse		反显
FontStyleBold		粗体
FontStyleUpsideDown		倒置
FontStyleUnderlineOneDotThick		一点粗下划线
FontStyleUnderlineTwoDotThick		两点粗下划线
其他		参数错误

【说明】

- 在黑白反显打印模式选择时，下划线模式不起作用
- 不对顺时针旋转 90 度和 270 度的字符加下划线。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_SELECT_FONT_TYPE	选择字体失败
ERR_TEXT_SET_FONT_STYLE_REVERSE	反显失败
ERR_TEXT_SET_FONT_STYLE_BOLD	粗体失败
ERR_TEXT_SET_FONT_STYLE_UNDERLINE	下划线失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk textSelectFont: FontTypeStandardASCII FontStyle: FontStyleBold];
```

● **textSetColorPrint**

进入双色打印，并选择不同于黑色的另一种颜色。

函数

- (SInt32)**textSetColorPrint**

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_ENTER_QUIT_COLOR_PRINT	进入双色打印失败
ERR_TEXT_SET_COLOR_PRINT	设置颜色失败

示例代码

```
[self.pos_sdk textSetColorPrint];
```

● **textQuitColorPrint**

退出双色打印。

函数

- (SInt32)**textQuitColorPrint**

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_ENTER_QUIT_COLOR_PRINT	退出双色打印失败

示例代码

```
[self.pos_sdk textQuitColorPrint];
```

● **textUserDefinedCharacterEnable**

选择或取消用户自定义字符。

函数

- (SInt32)**textUserDefinedCharacterEnable:(SInt32)Enable**

参数

- **Enable** 选择或者取消用户自定义字符

Enable 取值	描述
FontUserDefinedDisable	取消用户自定义字符

FontUserDefinedEnable	选择用户自定义字符
其他	非法参数

【说明】

a) 可以选择和取消全部的自定义字符。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_FONT_USER_DEFINED_ENABLE	选择和取消全部的自定义字符失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

详细使用见方法举例中的[用户自定义字符](#)。

● textUserDefinedCharacterDefine

定义用户自定义字符。

函数

- (SInt32)textUserDefinedCharacterDefine:(SInt32)BytesOfHeight
DotsOfWidth:(SInt32)DotsOfWidth StartingCode:(SInt32)StartingCode
EndingCode:(SInt32)EndingCode CharacterData:(NSData*)CharacterData

参数

- **BytesOfHeight** 指定纵向字节数
- **DotsOfWidth** 指定横向点数
- **StartingCode** 起始字符代码
- **EndingCode** 终止字符代码
- **CharacterData** 下载字符的数据

各参数合法值

参数	合法取值
BytesOfHeight	3
DotsOfWidth	9 或 12
StartingCode	32-127
EndingCode	32-127

【说明】

- a) 参数 BytesOfHeight 必须为 3。
- b) 自定义的字符的图像必须为 9*17 或 12*24。
- c) 字符打印的反显、下划线、间距设置、行高设置、对齐方式设置、旋转打印等字符打印风格，均能影响自定义字符打印。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_FONT_USER_DEFINED	定义用户自定义字符失败
ERR_INVALID_ARGUMENT	参数错误或数据个数不匹配

示例代码

详细使用见方法举例中的[用户自定义字符](#)。

● textUserDefinedCharacterCancel

取消指定的自定义字符。

函数

- (SInt32)textUserDefinedCharacterCancel:(SInt32)CharCode

参数

- **CharCode** 被取消的用户自定义字符的代码

CharCode	取值	描述
32-127		合法值
其他		非法参数

【说明】

- a) 可以取消指定 CharCode 代码的单个用户自定义字符。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_FONT_USER_DEFINED_CANCEL	取消指定的自定义字符失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

详细使用见方法举例中的[用户自定义字符](#)。

● textUserDefinedChineseCharacterDefine

用户自定义中文字符。

函数

- (SInt32)textUserDefinedChineseCharacterDefine:(SInt32)BytesOfHeight
DotsOfWidth:(SInt32)DotsOfWidth Code:(SInt32)Code
CharacterData:(NSData*)CharacterData

参数

- **BytesOfHeight** 指定纵向字节数
- **DotsOfWidth** 指定横向点数
- **Code** 用户自定义汉字的第二个字符
- **CharacterData** 下载字符的数据

各参数合法值

参数	合法取值
BytesOfHeight	3
DotsOfWidth	24
Code	161-254

【说明】

- 参数 BytesOfHeight 必须为 3。
- 自定义的字符的图像必须为 24*24。

返回值

返回值	情况
SUCCESS	成功
ERR_TEXT_FONT_USER_DEFINED	用户自定义中文字符失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
SInt32 BytesOfHeight, DotsOfWidth, SecondCode = 162;
NSMutableData *CharacterData = nil;
UIImage *image = nil;
NSString *image_name = @"ChineseCharacter.PNG";

///获取 UIImage 类型图像 image///
ImageDataRef image_data = {0};
[self.pos_sdk imageFormatConvertToUserDefinedData:image ditheringSupported:FALSE
```

```

image_data:&image_data];

///获取 BytesOfHeight、DotsOfWidth 及 CharacterData///
error_code = [self.pos_sdk textUserDefinedChineseCharacterDefine:BytesOfHeight
DotsOfWidth:DotsOfWidth SecondCode:SecondCode CharacterData:CharacterData];

UInt8 buffer[2] = {0};
buffer[0] = 0xfe;
buffer[1] = SecondCode;
error_code = [self.pos_sdk textPrint:buffer length:2];
error_code = [self.pos_sdk systemFeedLine:1];

```

● textStandardModeRasterPrint

光栅化字符打印（只在标准模式下有效）。

函数

```

- (SInt32)textStandardModeRasterPrint:(NSString*)TextToPrint
FontName:(NSString*)FontName FontSize:(CGFloat)FontSize
LineBreakMode:(UILineBreakMode)LineBreakMode
PrinterWidth:(SInt32)PrinterWidth

```

参数

- **TextToPrint** 要打印的字符串
- **FontName** IOS 系统自带的字体类型的可选的字体风格
- **FontSize** 字体大小, 当 Printer Width 不为 0 时字体大小不能大于 Printer Width 大小
- **LineBreakMode** 断行方式

LineBreakMode	取值	描述
UILineBreakModeWordWrap		以单词断行
UILineBreakModeCharacterWarp		以字符断行
UILineBreakModeClip		回旋断行
UILineBreakModeHeadTruncation		头部截断
UILineBreakModeTailTruncation		尾部截断
UILineBreakModeMiddleTruncation		中间截断
其他		非法参数

- **PrinterWidth** 打印头宽度

PrinterWidth	取值	描述
--------------	----	----

0	图像不缩放
64-2040	合法的打印头宽度，图像正常缩放
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_IMAGE_STANDARD_MODE_RASTER_PRINT	光栅化字符打印失败
ERR_INVALID_DATA	转换后的光栅化图像为空
ERR_INVALID_ARGUMENT	参数错误或 FontSize>PrinterWidth

示例代码

见方法举例中[文本光栅化打印](#)。

【说明】

a) 本函数中包含系统初始化指令，会取消之前设置的打印效果。

● imageStandardModePrint

下载并打印位图（只在标准模式下有效）。

函数

- (SInt32)imageStandardModePrint:(SInt32)Mode Image:(UIImage*)Image
StartHorPos:(SInt32)StartHorPos PrinterWidth:(SInt32)PrinterWidth

参数

- **Mode** 位图模式
- **Image** 要打印的图像
- **StartHorPos** 参数不启用，固化为 0

Mode 取值	描述
SingleDensity_8	8 点单密度
DoubleDensity_8	8 点双密度
SingleDensity_24	24 点单密度
DoubleDensity_24	24 点双密度
其他	非法参数

- **PrinterWidth** 打印头宽度

PrinterWidth 取值	描述
0	图像不缩放
64-65535	合法的打印头宽度，图像正常缩放
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_IMAGE_DOWNLOAD_AND_PRINT	图像下载并打印失败
ERR_ALLOC_MEMORY	申请空间储存图像数据失败
ERR_INVALID_ARGUMENT	参数错误或图像为空

示例代码

见方法举例中的[位图打印](#)。

【说明】

- 调用本函数会取消之前的字符旋转设置。
- 调用本函数后字符行高会改变，因此每行的走纸距离会变化，若想恢复成默认行高可以调用 `textSetLineHeight:0`。

● imageDownloadToPrinterRAM

下载 RAM 图像。

函数

- (SInt32)imageDownloadToPrinterRAM:(SInt32)ImageID

Image:(UIImage*)Image PrinterWidth:(SInt32)PrinterWidth

参数

- **ImageID** 下载图像的 ID 号

ImageID 取值	描述
0-7	合法 ID 号
其他	非法参数

- **Image** 要下载的图像
- **PrinterWidth** 打印头宽度

PrinterWidth 取值	描述
0	图像不缩放

64-65535	合法的打印头宽度，图像正常缩放
其他	非法参数

【说明】

- a) 每次只能下载单幅图像，并指定其 ID 号。
- b) 要下载的图像高度不能大于 2040。
- c) 打印机重新上电或者重新建立端口连接时，已下载的图像被删除。

返回值

返回值	情况
SUCCESS	成功
ERR_IMAGE_DOWNLOAD_RAM	下载 RAM 图像失败
ERR_INVALID_ARGUMENT	1. 参数错误 2. 图像为空 3. 下载的图像高度大于 2040
ERR_ALLOC_MEMORY	申请空间存放图像失败
ERR_INVALID_DATA	格式转换后图像数据为空

示例代码

见方法举例中的 [RAM/Flash 位图下载及打印](#)

● imageRAMPrint

打印 RAM 中下载的位图。

函数

- (SInt32)imageRAMPrint:(SInt32)ImageID Mode:(SInt32)Mode

参数

- **ImageID** 打印图像的 ID 号

ImageID 取值	描述
0-7	合法 ID 号
其他	非法参数

- **Mode** 图像打印模式

Mode 取值	描述
NormalMode	正常大小
Double_width	图像倍宽

Double_height	图像倍高
Quadruple	图像倍宽倍高
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_IMAGE_RAM_PRINT	打印 RAM 位图失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

见方法举例中的 [RAM/Flash 位图下载及打印](#)

● imageDownloadToPrinterFlash

下载 Flash 位图。

函数

- (SInt32)imageDownloadToPrinterFlash:(NSMutableArray*)ImageArray
PrinterWidth:(SInt32)PrinterWidth

参数

- **ImageArray** 待下载到 flash 中的原始图像数组 数量范围[1, 255]
- **PrinterWidth** 打印头宽度

PrinterWidth 取值	描述
0	图像不缩放
64-65535	合法的打印头宽度，图像正常缩放
其他	非法参数

【说明】

a) 每次下载删除所有的之前由函数 imageDownloadToPrinterFlash: 下载到 Flash 中的图像。

b) 因执行该函数后，Flash 需要擦除之前下载的图像，所以需要等待一段时间打印机才能完成本次图像的下载，最长可能需要 30 秒左右的时间，请在此时间内不要将打印机断电。

c) 对于下载多幅图像失败情况，其中可能有部分位图下载成功。打印下载到 Flash 中的位图以打印机实际位图存储情况为准。

d) 要下载的图像高度不能大于 2040。

e) 打印机重新上电或者重新建立端口连接时，已下载的图像不会被删除。

f) 多幅图像下载时，图像名之间以逗号分隔开，如：

SNBC.bmp,Jpg.jpg,face.PNG 。

返回值

返回值	情况
SUCCESS	成功
ERR_IMAGE_DOWNLOAD_FLASH	下载 Flash 位图失败
ERR_INVALID_ARGUMENT	1. 数组为空 2. 参数错误或下载的图像高度大于 2040 3. 下载的图像数据为空
ERR_ALLOC_MEMORY	申请空间存放图像失败
ERR_INVALID_DATA	格式转换后图像数据为空

示例代码

见方法举例中的 [RAM/Flash 位图下载及打印](#)

● imageFlashPrint

打印 Flash 位图。

函数

- (SInt32)imageFlashPrint:(SInt32)ImageID Mode:(SInt32)Mode

参数

- **ImageID** 打印图像的 ID 号

ImageID 取值	描述
1-255	合法 ID 号
其他	非法参数

- **Mode** 图像打印模式

Mode 取值	描述
NormalMode	正常大小
Double_width	图像倍宽
Double_height	图像倍高
Quadruple	图像倍宽倍高
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_IMAGE_FLASH_PRINT	打印 Flash 位图失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

见方法举例中的 [RAM/Flash 位图下载及打印](#)

● imageStandardModeRasterPrint

标准模式下打印光栅位图。

函数

- (SInt32)imageStandardModeRasterPrint:(UIImage*)Image
PrinterWidth:(SInt32)PrinterWidth

参数

- **Image** 要打印的原始图像
- **PrinterWidth** 打印头宽度

PrinterWidth 取值	描述
0	图像不缩放
64-2040	合法的打印头宽度，图像正常缩放
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_IMAGE_STANDARD_MODE_RASTER_PRINT	打印光栅化位图失败
ERR_INVALID_ARGUMENT	参数错误
ERR_INVALID_DATA	转换为光栅化图像数据为空

示例代码

见方法举例中的 [光栅化位图打印](#)。

【说明】

- a) 本函数中包含系统初始化指令，会取消之前设置的打印效果。

● **imageFormatConvertToUserDefinedData**

将图像数据转换为适用于用户自定义字符的图像数据。

函数

- (SInt32)**imageFormatConvertToUserDefinedData**:(UIImage*)Image
ditheringSupported:(BOOL)ditheringSupported
image_data:(ImageDataRef*)image_data

参数

- **Image** 要转换的原始图像
- **ditheringSupported** 是否选择抖动
- **image_data** 转换后的适用于用户自定义的图像数据

返回值

返回值	情况
SUCCESS	成功
ERR_ALLOC_MEMORY	申请空间失败

示例代码

```
ImageDataRef image_data = {0};

[self.pos_sdk imageFormatConvertToUserDefinedData:image ditheringSupported: FLSE
image_data:&image_data];
```

● **barcodeGetIDByName**

依据条码名称获取条码 ID 号。

函数

- (SInt32)**barcodeGetIDByName**:(NSString*)BarcodeName

参数

- **BarcodeName** 条码名称

BarcodeName	取值	描述
BarcodeUPC-A		对应 BarcodeUPC_A 的 ID 号
BarcodeUPC-E		对应 BarcodeUPC_E 的 ID 号
BarcodeJAN13orEAN13		对应 BarcodeJAN13orEAN13 的 ID 号
BarcodeJAN8orEAN8		对应 BarcodeJAN8orEAN8 的 ID 号
BarcodeCODE39		对应 BarcodeCODE39 的 ID 号

BarcodeITF	对应 BarcodeITF 的 ID 号
BarcodeCODABAR	对应 BarcodeCODABAR 的 ID 号
BarcodeCODE93	对应 BarcodeCODE93 的 ID 号
BarcodeCODE128	对应 BarcodeCODE128 的 ID 号
BarcodePDF417	对应 BarcodePDF417 的 ID 号
BarcodeQR	对应 BarcodeQR 的 ID 号
BarcodeMaxicode	对应 BarcodeMaxicode 的 ID 号
BarcodeGS1	对应 BarcodeGS1 的 ID 号
其他	非法参数

返回值

返回值	情况
条码名称对应的 ID 号	成功
-1	失败

示例代码

```
barcode_id = [self.pos_sdk barcodeGetIDByName:BarcodeUPC-A];
```

● barcodeGetNameByID

根据条码的 ID 号获得条码名称。

函数

-(NSString*)barcodeGetNameByID:(SInt32)BarcodeID

参数

- BarcodeID 条码的 ID 号

BarcodeID 取值	描述
0-8	合法 ID 号
其他	非法参数

返回值

返回值	情况
条码名称	成功
nil	失败

示例代码

```
[self.pos_sdk barcodeGetNameByID:0];
```

● barcodePrint1Dimension

打印一维条码。

函数

- (SInt32)barcodePrint1Dimension:(NSData*)Data BarcodeType:(SInt32)Type
ModuleWidth:(SInt32)ModuleWidth BarcodeHeight:(SInt32)Height
HriFontType:(SInt32)HriFontType HriPosition:(SInt32)HriPosition

参数

- **Data** 条码数据
- **BarcodeType** 条码类型

BarcodeType	取值	描述	条码数据长度	取值
BarcodeUPC_A		UPC-A	11	-12
BarcodeUPC_E		UPC-E	11	-12
BarcodeJAN13orEAN13		EAN13	12	-13
BarcodeJAN8orEAN8		EAN-8	7	-8
BarcodeCODE39		Code39	1	-255
BarcodeITF		交叉 25 码	1	-255
BarcodeCODABAR		CodaBar	1	-255
BarcodeCODE93		Code93	1	-255
BarcodeCODE128		Code128	2	-255
其他		非法参数		

- **ModuleWidth** 基本模块宽度

ModuleWidth	取值	描述
2-6		合法的条码基本模块宽度
其他		非法参数

- **BarcodeHeight** 条码高度

BarcodeHeight	取值	描述
1-255		合法的条码高度参数
其他		非法参数

- **HriFontType** 字体类型

HriFontType	取值	描述
FontTypeStandardASCII		标准 ASCII
FontTypeCompressedASCII		压缩 ASCII

其他	非法参数
----	------

- **HriPosition** HRI 字符的打印位置

HriPosition 取值	描述
HRINone	不打印 HRI 字符
HRIAbove	HRI 字符打印于条码上方
HRIBelow	HRI 字符打印于条码下方
HRIAboveAndBelow	HRI 字符在条码上方和下方均打印
其他	非法参数

【说明】

- a) 具体说明见[附录 C. 条码说明](#) 和 [附录 D. 128 码](#)。

返回值

返回值	情况
SUCCESS	成功
ERR_BARCODE_PRINT_1D	打印一维条码失败
ERR_BARCODE_SELECT_MODULE_WIDTH	设置模块宽度失败
ERR_BARCODE_SELECT_BARCODE_HEIGHT	设置条码高度失败
ERR_BARCODE_SELECT_HRI_FONT_TYPE	设置 HRI 字体类型失败
ERR_BARCODE_SELECT_HRI_FONT_POSITION	设置 HRI 字符打印位置失败
ERR_BARCODE_1D_SEND_DATA	发送一维条码数据失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
NSData *data_1D = nil;
///获取 NSData 类型的条码数据 data_1D///
[self.pos_sdk barcodePrint1Dimension:data_1D BarcodeType: BarcodeUPC_A ModuleWidth:3
BarcodeHeight:100 HriFontType: FontTypeStandardASCII HriPosition: HRIBelow];
```

● barcodePrintQR

设置 QR 码参数并打印 QR 码。

函数

- (SInt32)**barcodePrintQR**:(NSData*)Data
BasicElementWidth:(SInt32)BasicElementWidth SymbolType:(SInt32)SymbolType

LanguageMode:(SInt32)LanguageMode

参数

- **Data** 要打印的条码数据
- **BasicElementWidth** 条码基本元素宽度

BasicElementWidth 取值	描述
1-255	合法的条码基本元素宽度
其他	非法参数

- **LanguageMode** 语言模式

LanguageMode 取值	描述
LanguageChinese	中文
LanguageJapanese	日文
其他	非法参数

- **SymbolType** 符号类型

SymbolType 取值	描述
OriginalType	原始类型
EnhancedType	增强型（推荐）
其他	非法参数

【说明】

- 设置条码数据和基本元素宽度使条码超出打印的页面时，不打印条码。
- 推荐使用 EnhancedType 类型。

返回值

返回值	情况
SUCCESS	成功
ERR_BARCODE_QR_SET_PARAM	设置 QR 码参数失败
ERR_BARCODE_PRINT_2D	选择条码类型为 QR 失败
ERR_BARCODE_QR_SEND_DATA	发送条码数据失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

见方法举例中的 [QR 码打印](#)。

● barcodePrintPDF417

设置 PDF417 条码参数并打印 PDF417 条码。

函数

- (SInt32)**barcodePrintPDF417**:(NSData*)Data

AppearanceToHeight:(SInt32)AppearanceToHeight

AppearanceToWidth:(SInt32)AppearanceToWidth RowNumber:(SInt32)RowNumber

ColumnNumber:(SInt32)ColumnNumber XSize:(SInt32)XSize

LineHeight:(SInt32)LineHeight CorrectionGrade:(SInt32)CorrectionGrade

参数

- **Data** 要打印的条码数据
- **AppearanceToHeight** 外观比高度
- **AppearanceToWidth** 外观比宽度
- **RowNumber** 行数
- **ColumnNumber** 列数
- **XSize** X 尺寸
- **LineHeight** 行高
- **CorrectionGrade** 纠错等级

各参数合法值列表如下：

参数	合法值
AppearanceToHeight	1-10
AppearanceToWidth	1-100
RowNumber	3-90
ColumnNumber	1-30
XSize	1-7
LineHeight	2-25
CorrectionGrade	0-8

【说明】

- 数据超界，不打印条码。
- 条码大小超出页面范围不打印。

返回值

返回值	情况
SUCCESS	成功
ERR_BARCODE_PDF417_SET_SIZE	设置 PDF417 大小失败
ERR_BARCODE_PDF417_SELECT_CORREC	设置纠错等级失败

TION_GRADE	
ERR_BARCODE_PRINT_2D	选择打印 PDF417 条码失败
ERR_BARCODE_PDF417_SEND_DATA	发送条码数据失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

见方法举例中的 [PDF417 条码打印](#)。

● barcodePrintMaxicode

打印 Maxicode 条码。

函数

- (NSInteger)barcodePrintMaxicode:(NSData*)Data

参数

- **Data** 要打印的条码数据

【说明】

- 数据超界，不打印条码。

返回值

返回值	情况
SUCCESS	成功
ERR_BARCODE_PRINT_2D	选择打印 Maxicode 条码失败
ERR_BARCODE_MAXICODE_SEND_DATA	发送条码数据失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

见方法举例中 [Maxicode 条码打印](#)。

● barcodePrintGS1DataBar

设置 GS1 DataBar 条码参数。

函数

- (NSInteger)barcodePrintGS1DataBar:(NSData*)Data

BarcodeType:(NSInteger)BarcodeType BasicElementWidth:(NSInteger)BasicElementWidth

BarcodeHeight:(SInt32)BarcodeHeight

BasicElementHeight:(SInt32)BasicElementHeight

SeparatorHeight:(SInt32)SeparatorHeight SegmentNumber:(SInt32)SegmentNumber

HRI:(SInt32)HRI UseAI:(SInt32)AI

参数

- **Data** 要打印的条码数据
- **BarcodeType** 条码类型

BarcodeType	取值	描述
GS1DataBarOmnidirectional		全向型 GS1DataBar Omnidirectional
GS1DataBarTruncated		截短型 GS1DataBar Truncated
GS1DataBarStacked		层排型 GS1 DataBar Stacked
GS1DataBarStackedOmnidirectional		全向层排型 GS1 DataBar Stacked Omnidirectiona
GS1DataBarLimited		限定性 GS1 DataBar Limited
GS1DataBarExpanded		扩展型 GS1 DataBar Expanded
GS1DataBarExpandedStacked		扩展层排型 GS1 DataBar ExpandedStacked
其他		非法参数

- **BasicElementWidth** 基本元素宽度

BasicElementWidth	取值	描述
1-6		合法基本元素宽度
其他		非法参数

- **BarcodeHeight** DataBar 的高度，如果条码是 GS1DataBarStacked、GS1DataBarStackedOmnidirectional、GS1DataBarExpandedStacked 是指多行条码符号中每一行的高度。

BarcodeHeight	取值	描述
2-250		合法高度
其他		非法参数

- **BasicElementHeight** 复合码中 2D 条码符号基本元素高度

BasicElementHeight	取值	描述
1-10		合法基本元素高度
其他		非法参数

- **SeparatorHeight** 分隔符的高度。条码类型是 DataBar 复合码或者独立的 GS1DataBarStacked、GS1DataBarStackedOmnidirectional、GS1DataBarExpandedStacked 需要设置此参数。

SeparatorHeight 取值	描述
1-10	合法分隔符的高度
其他	非法参数

- **SegmentNumber** 每行条码符号的段数，只有条码类型是 GS1DataBarExpandedStacked 时需要设置此参数。

SegmentNumber 取值	描述
2-20	合法的独立的扩展层排型条码参数范围
4-20	合法的复合的扩展层排型条码时参数范围
其他	非法参数

- **HRI** 注释字符内容

HRI 取值	描述
DataBarAnd2DHri	复合码时注释字符是 DataBar 和 2D 两部分的数据，独立码时注释字符是 DataBar 数据
Only DataBarHri	复合码或独立码 打印 DataBar 部分的数据
Only2DHri	复合码时打印 2D 部分的数据 独立码时不打印
NoHri	无注释字符
其他	非法参数

- **AI** 是否应用 AI（应用标识符）：0 表示不应用 AI；1 表示应用 AI。

【说明】

- 几种型号打印机无法打印 GS1 的所有类型条码。
- 数据超界，不打印条码。
- 条码大小超出页面范围不打印。

返回值

返回值	情况
SUCCESS	成功
ERR_BARCODE_GS1DATABAR_SET_PARAM	设置 GS1 参数失败
ERR_BARCODE_PRINT_2D	选择 GS1 DataBar 类型失败
ERR_BARCODE_GS1DATABAR_SEND_DATA	发送条码数据失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

见方法举例中 [GS1 DataBar](#) 和 [GS1 复合条码打印](#)。

● standardModeSetLeftMarginAndPrintAreaWidth

标准模式设置打印左边距及区域宽度。

函数

(SInt32)standardModeSetLeftMarginAndPrintAreaWidth:(SInt32)LeftMargin
Width:(SInt32)Width

参数

- **LeftMargin** 左边距

LeftMargin	取值	描述
0-65535		合法左边距
其他		非法参数

- **Width** 打印区域宽度

Width	取值	描述
0-65535		合法打印区域宽度
其他		非法参数

【说明】

- 参数 LeftMargin 和 Width 设置，不影响用户自定义字符、光栅化位图、光栅化字符的打印范围。
- 页模式此接口函数无效。

返回值

返回值	情况
SUCCESS	成功
ERR_STANDARD_MODE_SET_PRINTAREA_WIDTH	设置区域宽度失败
ERR_STANDARD_MODE_SET_LEFT_MARGIN	设置左边距失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk standardModeSetLeftMarginAndPrintAreaWidth:100 Width:500];
```

● standardModeSetHorizontalStartingPosition

标准模式设置打印起始横坐标位置。

函数

- (SInt32)standardModeSetStartingPosition:(SInt32)X

参数

- **X** 横向绝对起始坐标

Distance 取值	描述
0-65535	合法值
其他	非法参数

【说明】

a) 参数 X 设置，不影响用户自定义字符、光栅化位图、光栅化字符的打印范围。

返回值

返回值	情况
SUCCESS	成功
ERR_STANDARD_MODE_SET_HORIZONTAL_STARTING_POSITION	设置标准模式打印起始横坐标位置失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk standardModeSetStartingPosition:100];
```

● pageModeSetStartingPosition

页模式设置打印的横向、纵向起始位置。

函数

- (SInt32)pageModeSetStartingPosition:(SInt32)X Y:(SInt32)Y

参数

- **X** 横向绝对起始坐标

X 取值	描述
0-65535	合法值
其他	非法参数

- **Y** 纵向绝对起始坐标

Y 取值	描述
0-65535	合法值
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_STANDARD_MODE_SET_HORIZONTAL_STARTING_POSITION	设置标准模式横向起始坐标失败
ERR_PAGE_MODE_SET_VERTICAL_STARTING_POSITION	设置页模式纵向起始坐标失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
error_code = pos_sdk.pageModeSetStartingPosition:203 Y:203;
```

● pageModeSetPrintArea

页模式的打印区域设置。

函数

- (SInt32)**pageModeSetPrintArea**:(SInt32)X Y:(SInt32)Y
AreaWidth:(SInt32)AreaWidth AreaHeight:(SInt32)AreaHeight
PrintDirection:(SInt32)Direction

参数

- **X** 横向起始位置
- **Y** 纵向起始位置
- **AreaWidth** 打印区域宽度
- **AreaHeight** 打印区域高度

X/Y/AreaWidth/AreaHeight 取值	描述
0-65535	合法值
其他	非法参数

- **Direction** 打印方向

Direction 取值	描述
LeftToRight	从左向右打印
BottomToTop	从下向上打印
RightToLeft	从右向左打印
TopToBottom	从上向下打印
其他	非法参数

返回值

返回值	情况
SUCCESS	成功
ERR_PAGE_MODE_SET_PRINT_AREA	设置页模式打印区域失败
ERR_PAGE_MODE_SET_PRINT_DIRECTION	设置页模式打印方向失败
ERR_INVALID_ARGUMENT	参数错误

示例代码

```
[self.pos_sdk pageModeSetPrintArea:100 Y:0 AreaWidth:400 AreaHeight:1000  
PrintDirection:LeftToRight];
```

● pageModePrint

页模式下打印。

函数

- (SInt32)pageModePrint

返回值

返回值	情况
SUCCESS	成功
ERR_PAGE_MODE_PRINT	页模式打印失败

示例代码

```
[self.pos_sdk pageModePrint];
```

● pageModeClearBuffer

页模式下清空 Buffer。

函数

- (SInt32)pageModeClearBuffer

返回值

返回值	情况
SUCCESS	成功
ERR_PAGE_MODE_CLEAR_BUFFER	页模式下清空 Buffer 失败

示例代码

```
[self.pos_sdk pageModeClearBuffer];
```

● MsrReadMagneticDataForFirstTrack

读取第一磁道的数据。

函数

- (NSString*) MsrReadMagneticDataForFirstTrack:(UInt8*)DataBuffer
DataSize:(Sint32)DataSize

参数

- **DataBuffer** 存放读取到的数据的缓冲区
- **DataSize** 读入数据的字节数

返回值

返回值	情况
<\x02>Success ! 参数 DataBuffer 返回读 取的第一磁道的数据 <\x0D><\x0A><\x03>	成功
read magnetic failed !	下发读取第一磁道数据指令失败
<\x02>read magnetic failed ! : <\x0D><\x0A><\x03>	读取第一磁道的数据失败

示例代码

```
#define size 256  
UInt8 buffer[size] = {0};  
[self.pos_sdk MsrReadMagneticDataForFirstTrack:buffer DataSize:size];
```

● MsrReadMagneticDataForSecondTrack

读取第二磁道的数据。

函数

- (NSString*) MsrReadMagneticDataForSecondTrack:(UInt8*)DataBuffer
DataSize:(Sint32)DataSize

参数

- **DataBuffer** 存放读取到的数据的缓冲区

- **DataSize** 读入数据的字节数

返回值

返回值	情况
<\x02>Success ! 参数 DataBuffer 返回读取的第二磁道的数据<\x0D><\x0A><\x03>	成功
read magnetic failed !	下发读取第二磁道数据指令失败
<\x02>read magnetic failed ! :<\x0D><\x0A><\x03>	读取第二磁道的数据失败

示例代码

```
#define size 256
UInt8 buffer[size] = {0};
[self.pos_sdk MsrReadMagneticDataForSecondTrack:buffer DataSize:size];
```

● MsrReadMagneticDataForThirdTrack

读取第三磁道的数据。

函数

- (NSString*) MsrReadMagneticDataForThirdTrack:(UInt8*)DataBuffer

DataSize:(Sint32)DataSize

参数

- **DataBuffer** 存放读取到的数据的缓冲区
- **DataSize** 读入数据的字节数

返回值

返回值	情况
<\x02>Success ! 参数 DataBuffer 返回读取的第三磁道的数据<\x0D><\x0A><\x03>	成功
read magnetic failed !	下发读取第三磁道数据指令失败
<\x02>read magnetic failed ! :<\x0D><\x0A><\x03>	读取第三磁道的数据失败

示例代码

```
#define size 256
UInt8 buffer[size] = {0};
```

```
[self.pos_sdk MsrReadMagneticDataForThirdTrack:buffer DataSize:size ];
```

● MsrReadMagneticDataForThreeTracks

读取三个磁道的数据。

函数

- (NSString*) MsrReadMagneticDataForThreeTracks:(UInt8*)DataBuffer
DataSize:(Sint32)DataSize

参数

- **DataBuffer** 存放读取到的数据的缓冲区
- **DataSize** 读入数据的字节数

返回值

返回值	情况
<\x02>Success ! 参数 DataBuffer 返回读取的 第三个磁道的数据<\x0D><\x0A><\x03>	成功
read magnetic failed !	下发读取三个磁道数据指令失败
<\x02>read magnetic failed ! : <\x0D><\x0A><\x03>	读取三个磁道的数据失败

示例代码

```
#define size 256  
UInt8 buffer[size] = {0};  
[self.pos_sdk MsrReadMagneticDataForThreeTracks:buffer DataSize:size ];
```

● ICRest

复位 IC 卡。

函数

- (NSString*) ICRest:(UInt8*)DataBuffer DataSize:(Sint32)DataSize

参数

- **DataBuffer** 存放 IC 卡复位读取到的数据的缓冲区
- **DataSize** 读入数据的字节数

返回值

返回值	情况
<\x02>Success ! 参数 DataBuffer 返回 IC 卡复位读取的数据<\x0D><\x0A><\x03>	成功
Command runs failed !	下发复位指令失败
<\x02>Command runs failed ! error code:错误码<\x0D><\x0A><\x03>	IC 卡复位失败

示例代码

```
#define size 256
UInt8 buffer[size] = {0};
[self.pos_sdk ICRest:buffer DataSize:size];
```

● ICControlT0

IC 卡控制命令 T0 协议。

函数

- (NSString*) ICControlT0: (SInt32) CommandLength
Command:(UInt8*)Command DataBuffer:(UInt8*)DataBuffer
DataSize:(Sint32)DataSize

参数

- **CommandLength** 发送到 IC 卡的 T0 命令长度
- **Command** 发送到 IC 卡的 T0 命令
- **DataBuffer** 存放 IC 卡读取到的数据的缓冲区
- **DataSize** 读入数据的字节数

返回值

返回值	情况
<\x02>Success ! 参数 DataBuffer 返回读取的数据<\x0D><\x0A><\x03>	成功
Command runs failed !	下发 T0 控制协议指令失败
<\x02>command runs failed ! error code:错误码<\x0D><\x0A><\x03>	T0 协议响应失败

示例代码

```
#define size 256
```

```

UInt8 buffer[size] = {0};
SInt32 commandLength = 5;
UInt8 command[5] = {0x00,0x84,0x00,0x00,0x04};
[self.pos_sdk ICControlT0: commandLength Command: command DataBuffer:buffer
DataSize:size ];

```

● ICControlT1

IC 卡控制命令 T1 协议。

函数

- (NSString*) ICControlT1: (SInt32) CommandLength
 Command:(UInt8*)Command DataBuffer:(UInt8*)DataBuffer
 DataSize:(Sint32)DataSize

参数

- **CommandLength** 发送到 IC 卡的 T1 命令长度
- **Command** 发送到 IC 卡的 T1 命令
- **DataBuffer** 存放 IC 卡复位读取到的数据的缓冲区
- **DataSize** 读入数据的字节数

返回值

返回值	情况
<\x02>Success ! 参数 DataBuffer 返回读取的数据<\x0D><\x0A><\x03>	成功
Command runs failed !	下发 T1 控制协议指令失败
<\x02>command runs failed ! error code: 错误码<\x0D><\x0A><\x03>	T1 协议响应失败

示例代码

```

[self.pos_sdk ICControlT1: commandLength Command: command DataBuffer:buffer
DataSize:size ];

```

● kNotificationBLE

蓝牙状态返回。当状态变化时通知 APP 层，状态变化情况为：苹果设备蓝牙断开、打印机蓝牙断开。

当有苹果设备蓝牙断开、打印机蓝牙断开情况，通过通知发送。

定 义 通 知 名 称 NSString *kNotificationBLE =
@"com.SNBC.PocketSDK.BLEStatus"

Key 为@ “BLE Status”

@ “Central BLE Powered off!” 为苹果设备蓝牙断开；

@ “Printer is disconnect!” 为打印机断电。

5. 附录

附录 A. WIFIPortToFile 类的使用

WIFIPort 类可以实现向打印机成功下发数据的同时将数据保存到文件，也就是说成功打开端口是保存数据到文件的前提。有时客户可能不需要向打印成功发送数据，只需要将数据保存到文件，为此我们提供了 WIFIPortToFile 类。

- 添加 WIFIPortToFile.h

如何添加 WIFIPortToFile.h 文件请参照[如何添加静态库文件](#)。

- 使用 WIFIPortToFile.h

1)导入 WIFIPortToFile.h

```
#import "WIFIPortToFile.h"
```

2)声明类:

```
@property (nonatomic, retain) WIFIPortToFile *printer_port_file;
```

3)定义类:

```
@synthesize printer_port_file = _printer_port_file;

- (WIFIPortToFile *) printer_port_file
{
    if(_printer_port_file == nil)
    {
        _printer_port_file = [[WIFIPortToFile alloc] init];
    }
    return _printer_port_file;
}
```

- 类中方法说明

WIFIPortToFile 类中保留 WIFIPort 类的所有函数接口。其中，调用函数接口 searchPort 直接返回 nil；调用 openPort、closePort、readPort 直接返回 SUCCESS；WIFIPortToFile 类主要通过 writePort 下发数据，recordCommunicationDataEnable 将下发的数据保存到文件。保存数据到文件的示例代码如下：

```
[self. printer_port_file recordCommunicationDataEnable: @ "DataFile.dat"];
```

下发数据的示例代码如下：

```
data[2] = {0x1b,0x40};
[self. printer_port_file writePort:data offSize:0 WriteSize:2 WriteTimeOut:
TIME_OUT_SMALL_DATA];
```

附录 B. 错误码列表

错误码	描述
SUCCESS	成功
ERR_INVALID_ARGUMENT	参数错误
ERR_INVALID_DATA	经格式转换后的图像为空
ERR_INVALID_CONNECTION	未成功建立连接
ERR_CREATE_CONNECTION	创建套接字失败、超时时间内未等到可用资源、获取套接字失败
ERR_COMMUNICATE	1.设置超时时间失败 2.查询打印机状态时，返回的字节数和要读取的字节数不相同
ERR_ALLOC_MEMORY	申请空间失败
ERR_SYSTEM_RESET	打印机初始化失败
ERR_SYSTEM_SELECT_PRINT_MODE	设置打印模式失败
ERR_SYSTEM_SELECT_PAPER_TYPE	选择纸张类型失败
ERR_SYSTEM_SET_MOTION_UNIT	设置横纵向移动单位失败
ERR_SYSTEM_QUERY_STATUS	查询打印机状态失败
ERR_SYSTEM_FEED_LINE	打印机进纸失败
ERR_SYSTEM_CUT_PAPER	打印机切纸失败
ERR_CASH_DRAWER_OPEN	钱箱弹出失败
ERR_TEXT_SELECT_CHAR_SET	选择字符集失败
ERR_TEXT_SELECT_CODE_PAGE	选择代码页失败
ERR_TEXT_SET_LINE_HEIGHT	设置字符行高失败
ERR_TEXT_SET_CHARACTER_SPACE	设置字符间距失败
ERR_TEXT_STANDARD_MODE_ALIGNM ENT	标准模式下设置对齐方式失败
ERR_TEXT_SELECT_FONT_TYPE	选择字体类型失败
ERR_TEXT_SET_FONT_STYLE_REVERSE	反显打印失败
ERR_TEXT_SET_FONT_STYLE_BOLD	粗体打印失败
ERR_TEXT_SET_FONT_STYLE_UNDERLI NE	打印下划线效果失败
ERR_TEXT_STANDARD_MODE_UPSIDED	倒置打印失败

OWN	
ERR_TEXT_SELECT_MAGNIFY_TIMES	选择放大倍数失败
ERR_TEXT_STANDARD_MODE_ROTATE	标准模式下旋转打印失败
ERR_TEXT_ENTER_QUIT_COLOR_PRINT	选择/取消双色打印失败
ERR_TEXT_SET_COLOR_PRINT	设置颜色打印失败
ERR_TEXT_FONT_USER_DEFINED_ENABLE	选择用户自定义字符功能失败
ERR_TEXT_FONT_USER_DEFINED	用户自定义字符失败
ERR_TEXT_FONT_USER_DEFINED_CANCEL	取消用户自定义字符失败
ERR_TEXT_PRINT	打印文本失败
ERR_IMAGE_DOWNLOAD_AND_PRINT	图像下载并打印失败
ERR_IMAGE_DOWNLOAD_RAM	下载 RAM 位图失败
ERR_IMAGE_RAM_PRINT	打印 RAM 中下载的位图失败
ERR_IMAGE_DOWNLOAD_FLASH	下载 Flash 位图失败
ERR_IMAGE_FLASH_PRINT	打印 Flash 中下载的位图失败
ERR_IMAGE_STANDARD_MODE_RASTER_PRINT	标准模式下打印光栅位图失败
ERR_STANDARD_MODE_SET_PRINTAREA_WIDTH	设置标准模式的打印区域宽度失败
ERR_STANDARD_MODE_SET_LEFT_MARGIN	设置标准模式的打印左边距失败
ERR_STANDARD_MODE_SET_HORIZONTAL_STARTING_POSITION	设置标准模式的横向起始位置失败
ERR_PAGE_MODE_SET_VERTICAL_STARTING_POSITION	设置页模式的纵向起始坐标失败
ERR_PAGE_MODE_SET_PRINT_AREA	设置页模式的打印区域失败
ERR_PAGE_MODE_SET_PRINT_DIRECTION	设置页模式的打印方向失败
ERR_PAGE_MODE_PRINT	页模式下打印失败
ERR_PAGE_MODE_CLEAR_BUFFER	页模式清空 Buffer 失败
ERR_BARCODE_PRINT_1D	打印一维条码失败
ERR_BARCODE_PRINT_2D	选择要打印的二维条码失败

ERR_BARCODE_SELECT_MODULE_WIDTH	选择条码基本模块宽度失败
ERR_BARCODE_SELECT_BARCODE_HEIGHT	选择条码高度失败
ERR_BARCODE_SELECT_HRI_FONT_TYPE	选择条码的 HRI 字体失败
ERR_BARCODE_SELECT_HRI_FONT_POSITION	选择 HRI 字符的打印位置失败
ERR_BARCODE_QR_SET_PARAM	设置 QR 码参数失败
ERR_BARCODE_PDF417_SELECT_CORRECTION_GRADE	设置 PDF417 的纠错等级失败
ERR_BARCODE_PDF417_SET_SIZE	设置 PDF417 的大小失败
ERR_BARCODE_GS1DATABAR_SET_PARAM	设置 GS1 DataBar 的参数失败
ERR_BARCODE_1D_SEND_DATA	发送一维条码数据失败
ERR_BARCODE_QR_SEND_DATA	发送 QR 码数据失败
ERR_BARCODE_PDF417_SEND_DATA	发送 PDF417 条码数据失败
ERR_BARCODE_MAXICODE_SEND_DATA	发送 Maxicode 条码数据失败
ERR_BARCODE_GS1DATABAR_SEND_DATA	发送 GS1 DataBar 条码数据失败
ERR_RECEIVED_DATA	读取的数据不正确, 如第一个字节不是 0x02

附录 C. 条码说明

各条码类型对应的条码数据长度及字符集, 如下表:

条码类型	数据长度	字符集的 ASCII 值	备注
UPC-A	11 ~ 12	48 ~ 57	
UPC-E	11 ~ 12	48 ~ 57	第一个字符必须为 0
JAN13 (EAN13)	12 ~ 13	48 ~ 57	
JAN 8 (EAN8)	7 ~ 8	48 ~ 57	
CODE39	1 ~ 255	45 ~ 57, 65 ~ 90,	

		32, 36, 37,43	
ITF	1 ~ 255	48 ~ 57	
CODABAR	1 ~ 255	48 ~ 57 65 ~ 68, 36, 43,45,46,47 58	CODEBAR 码起始符和 结束符都必须为 A、B、 C、D 四个字母中的一个， 结束符可以使用 T、 E、*、N 四字符来代替
CODE93	1 ~ 255	0 ~ 127	
CODE128	2 ~ 255	0 ~ 127	条码数据前必须先选择 字符集
PDF417	1 ~ 255	0 ~ 255	
QRCODE	4 ~ 255	0 ~ 255	
MAXICODE	1 ~ 138	48 ~ 57,65 ~ 90	
GS1	1 ~ 255	由选择的 GS1 参数决 定，见下表	

GS1条码类型对应的条码数据长度及字符集

条码类型	数据长度	字符集
全向型 GS1DataBar Omnidirectional	14 位， 13 位数字+1 位 校验字符	数字 0-9
截短型 GS1DataBar Truncated	14 位， 13 位数字+1 位 校验字符	数字 0-9
层排型 GS1 DataBar Stacked	14 位， 13 位数字+1 位 校验字符	数字 0-9
全向层排型 GS1 DataBar Stacked Omnidirectiona	14 位， 13 位数字+1 位 校验字符	数字 0-9
限定性 GS1 DataBar Limited	14 位， 13 位数字+1 位 校验字符	数字 0-9
扩展型 GS1 DataBar Expanded	最大 74 个数字或 41 字 母	数字 0 ~ 9、A ~ Z、 a ~ z !" % & ' () * + , - . / : ; < = > ? _ 空格 FNC1
扩展层排型 GS1 DataBar ExpandedStacked	最大 74 个数字或 41 字 母	数字 0 ~ 9、A ~ Z、 a ~ z !" % & ' () * + ,

		- ./ : ; < = > ? _ 空格 FNC1
--	--	-------------------------------

[应用注释]

- 当选择UPC-A 、UPC-E、JAN13 (EAN13)或者JAN8 (EAN8)码时，如果条码数据个数超出了规定的取值范围，不打印条码。

[应用注释 （标准模式）]

- 如果条码数据超出了规定的字符集取值范围，不打印条码。
- 如果条码横向超出了打印区域，不打印条码。
- 不管由- (SInt32)textSetLineHeight:设定的行高是多少，走纸距离都与设定的条码高度相等。
- 只有在打印缓冲区没有数据时才有效，如果打印缓冲区有数据，该命令被忽略。
- 打印条码后，将打印位置设置在行首。
- 打印模式设置- (SInt32)textSelectFont: (FontStyle 为

FontStyleReverse/FontStyleBold/FontStyleUnderlineOneDotThick/FontStyleUnderlineTwoDotThick) 不影响打印条码，但是倒置模式 (- (SInt32)textStandardModeUpsideDown) 对条码打印有影响。

[应用注释 （页模式）]

- 将条码图形生成到打印缓冲区，但是并不打印。处理完条码数据后将打印位置移到条码的右边。
- 如果条码数据超出了规定的字符集取值范围，不打印条码。
- 如果条码宽度超出了打印区域，不打印条码。

当选择 CODE128时：

- CODE 128的相关信息和字符集，见附录 128 码。
- 在使用CODE 128 时，按照下列说明进行编码：
 - ①在条码数据前必须先选择字符集（CODE A、CODE B 和 CODE C中的一个）。
 - ②选择字符集是通过发送字符“{” 和另外一个字符结合来完成的；ASCII字符“{”通过连续发送字符“{”两次来完成。

指定字符集	发送数据
SHIFT	{S
CODE A	{A
CODE B	{B
CODE C	{C
FNC1	{1
FNC2	{2

FNC3	{3
FNC4	{4
"{"	{{

例如：选择CODE B打印“123456”，需要输入：{B123456

- 如果在条码数据的最前端不是字符集选择，则不打印条码。
- 如果“{”和紧接着它的那个字符不是上面所指定的组合则不打印条码。
- 如果打印机接收的字符不是条码字符集数据，则不打印条码。
- 打印机打印HRI字符时，不打印shift字符和字符集选择数据。
- 功能字符的HRI字符不打印。
- 控制字符（<00>H to <1F>H and <7F>H）的HRI字符也不打印。
- 一定要保证条码的左右间隙，间隙因条码类型不同而不同。

附录 D. 128 码

1. 128码综述

128码通过交替使用字符集A、字符集B和字符集C，能够对128个ASCII字符和00~99的100个数字以及一些特殊字符进行编码。每个字符集编码的字符如下：

- 字符集 A： ASCII 字符 00H 到 5FH
- 字符集 B： ASCII 字符 20H 到 7FH
- 字符集 C： 00~99的100个数字

128码也能对下列特殊字符进行编码：

- SHIFT 字符

“SHIFT”能使条码符号SHIFT字符后边第一个字符从字符集A转换到字符集B，或从字符集B转换到字符集A，从第二个字符开始恢复到SHIFT以前所用的字符集。“SHIFT”字符仅能在字符集A和字符集B之间转换使用，它无法使当前的编码字符进入或退出字符集C的状态。

- 字符集选择字符（CODE A、CODE B、 CODE C）

这些字符能将其后边的编码字符转换到字符集A、B或C。

- 功能字符（FNC1、 FNC2、FNC3、FNC4）

这些功能符的用处取决于应用软件。在字符集C中，只有FNC1 可用。

2. 字符集

字符集A中的字符

字符	发送数据		字符	发送数据		字符	发送数据	
	Hex	Decimal		Hex	Decimal		Hex	Decimal
NULL	00	0	(28	40	P	50	80
SOH	01	1)	29	41	Q	51	81
STX	02	2	*	2A	42	R	52	82
ETX	03	3	+	2B	43	S	53	83
EOT	04	4	,	2C	44	T	54	84
ENQ	05	5	-	2D	45	U	55	85
ACK	06	6	.	2E	46	V	56	86
BEL	07	7	/	2F	47	W	57	87
BS	08	8	0	30	48	X	58	88
HT	09	9	1	31	49	Y	59	89
LF	0A	10	2	32	50	Z	5A	90
VT	0B	11	3	33	51	[5B	91
FF	0C	12	4	34	52	\	5C	92
CR	0D	13	5	35	53]	5D	93
SO	0E	14	6	36	54	^	5E	94
SI	0F	15	7	37	55	_	5F	95
DLE	10	16	8	38	56	FNC1	7B,31	123,49
	11	17	9	39	57	FNC2	7B,32	123,50
	12	18	:	3A	58	FNC3	7B,33	123,51
	13	19	;	3B	59	FNC4	7B,34	123,52
	14	20	<	3C	60	SHIFT	7B,53	123,83
	15	21	=	3D	61	CODEB	7B,42	123,66
	16	22	>	3E	62	CODEC	7B,43	123,67
	17	23	?	3F	63			
	18	24	@	40	64			
	19	25	A	41	65			
	1A	26	B	42	66			
	1B	27	C	43	67			
	1C	28	D	44	68			
	1D	29	E	45	69			
	1E	30	F	46	70			
NAK	1F	31	G	47	71			
SYN	20	32	H	48	72			

ETB	21	33	I	49	73			
CAN	22	34	J	4A	74			
EM	23	35	K	4B	75			
SUB	24	36	L	4C	76			
ESC	25	37	M	4D	77			
FS	26	38	N	4E	78			
GS	27	39	O	4F	79			
RS								
US								
SP								
!								
"								
#								
\$								
%								
&								
'								

字符集B中的字符

字符	发送数据		字符	发送数据		字符	发送数据	
	Hex	Decimal		Hex	Decimal		Hex	Decimal
SP	20	32	H	48	72	p	70	112
!	21	33	I	49	73	q	71	113
"	22	34	J	4A	74	r	72	114
#	23	35	K	4B	75	s	73	115
\$	24	36	L	4C	76	t	74	116
%	25	37	M	4D	77	u	75	117
&	26	38	N	4E	78	v	76	118
'	27	39	O	4F	79	w	77	119
(28	40	P	50	80	x	78	120
)	29	41	Q	51	81	y	79	121
*	2A	42	R	52	82	z	7A	122
+	2B	43	S	53	83	{	7B,7B	123,123
,	2C	44	T	54	84		7C	124
-	2D	45	U	55	85	}	7D	125
.	2E	46	V	56	86	—	7E	126
/	2F	47	W	57	87	DEL	7F	127
0	30	48	X	58	88	FNC1	7B,31	123,49

1	31	49	Y	59	89	FNC2	7B,32	123,50
2	32	50	Z	5A	90	FNC3	7B,33	123,51
3	33	51	[5B	91	FNC4	7B,34	123,52
4	34	52	\	5C	92	SHIFT	7B,53	123,83
5	35	53]	5D	93	CODEA	7B,41	123,65
6	36	54	^	5E	94	CODEC	7B,43	123,67
7	37	55	_	5F	95			
8	38	56	`	60	96			
9	39	57	a	61	97			
:	3A	58	b	62	98			
;	3B	59	c	63	99			
<	3C	60	d	64	100			
=	3D	61	e	65	101			
>	3E	62	f	66	102			
?	3F	63	g	67	103			
@	40	64	H	68	104			
A	41	65	i	69	105			
B	42	66	j	6A	106			
C	43	67	k	6B	107			
D	44	68	l	6C	108			
E	45	69	m	6D	109			
F	46	70	n	6E	110			
G	47	71	o	6F	111			

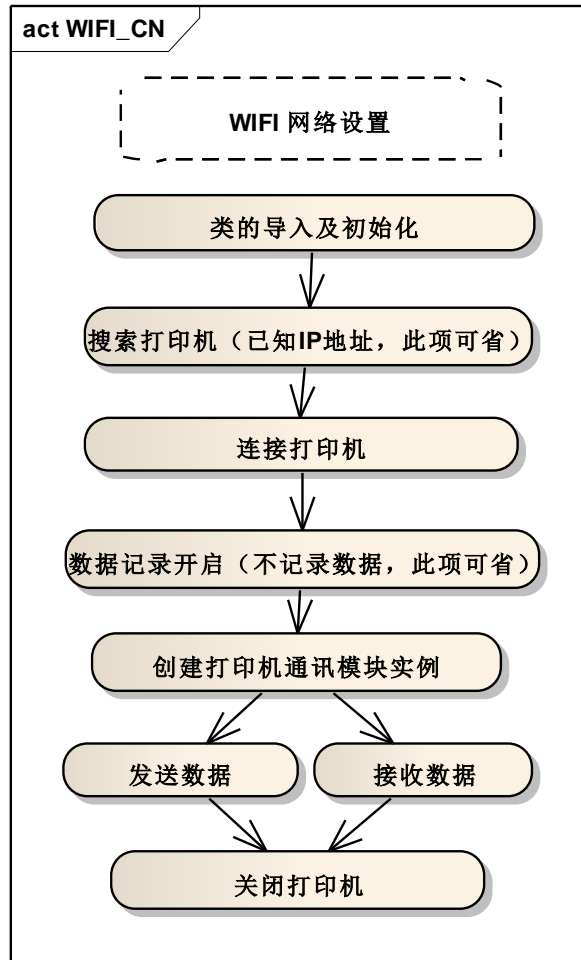
字符集C中的字符

字符	发送数据		字符	发送数据		字符	发送数据	
	Hex	Decimal		Hex	Decimal		Hex	Decimal
00	00	0	40	28	40	80	50	80
01	01	1	41	29	41	81	51	81
02	02	2	42	2A	42	82	52	82
03	03	3	43	2B	43	83	53	83
04	04	4	44	2C	44	84	54	84
05	05	5	45	2D	45	85	55	85
06	06	6	46	2E	46	86	56	86
07	07	7	47	2F	47	87	57	87
08	08	8	48	30	48	88	58	88
09	09	9	49	31	49	89	59	89
10	0A	10	50	32	50	90	5A	90

11	0B	11	51	33	51	91	5B	91
12	0C	12	52	34	52	92	5C	92
13	0D	13	53	35	53	93	5D	93
14	0E	14	54	36	54	94	5E	94
15	0F	15	55	37	55	95	5F	95
16	10	16	56	38	56	96	60	96
17	11	17	57	39	57	97	61	97
18	12	18	58	3A	58	98	62	98
19	13	19	59	3B	59	99	63	99
20	14	20	60	3C	60	FNC1	7B,31	123,49
21	15	21	61	3D	61	CODEA	7B,41	123,65
22	16	22	62	3E	62	CODEB	7B,42	123,66
23	17	23	63	3F	63			
24	18	24	64	40	64			
25	19	25	65	41	65			
26	1A	26	66	42	66			
27	1B	27	67	43	67			
28	1C	28	68	44	68			
29	1D	29	69	45	69			
30	1E	30	70	46	70			
31	1F	31	71	47	71			
32	20	32	72	48	72			
33	21	33	73	49	73			
34	22	34	74	4A	74			
35	23	35	75	4B	75			
36	24	36	76	4C	76			
37	25	37	77	4D	77			
38	26	38	78	4E	78			
39	27	39	79	4F	79			

附录 E. 程序流程

1. 接口调用流程如下(以 WIFI 接口为例):



以打印光栅化图像为例，简单介绍如何调用函数：

(1) 使用 WIFI 将打印机连接到 IOS 设备。

(2) 如何导入头文件、声明类的属性、初始化类等，详细请参阅：[如何使用静态库](#)。

(3) 搜索打印机：

```
NSMutableArray *port_info_set      = nil;
NSMutableArray *array_SearchPrinter = nil;
PortInfoWIFI    *port_info        = nil;
SInt32          index             = 0;
//调用搜索打印机
port_info_set = [self.printer_port searchPort];
//将搜索到的打印机的 IP 地址信息存放到 array_SearchPrinter 中
for(index = 0; index != [port_info_set count]; index++)
{
```

```
port_info = [port_info_set objectAtIndex:index];
[array_SearchPrinter addObject:port_info.IPAddr];
}
```

(4) 连接端口：可以依据步骤(3) 搜索到的打印机的 IP 地址信息建立连接，也可以直接输入 IP 地址建立连接，以输入 IP 地址建立连接为例：

```
[self.printer_port openPort: @"192.168.1.200" Timeout: TIME_OUT_CONNECT];
```

(5)记录数据功能使能（不记录通讯数据情况时，此项可省略）：

```
[self.printer_port recordCommunicationDataEnable:@"DataFile.dat"];
```

(6)创建打印机通讯模块实例：

```
[self.pos_sdk systemSetPortIO:self.printer_port];
```

(7)打印，以打印光栅化图像为例：

```
UIImage *image = nil;
[self.pos_sdk imageStandardModeRasterPrint:image PrinterWidth:640];
```

对于选择了数据记录功能的情况，会在.app 文件下生成数据记录文件，如生成 DataFile.dat.write9100。

(8)关闭连接

```
[self.pos_sdk systemSetPortIO:nil];
[self.printer_port closePort];
```