一、SDK导入说明

```
1.把压缩包里面的PrinterSDK.xcframework拖进你的项目中,兼容模拟器和真机
2.需要另外添加SystemConfiguration.framework这个系统库
3.在Build Settings -> Linking -> Other Linker Flags 选项中添加-ObjC(视情况而定)
4.使用蓝牙的地方添加#import <CoreBluetooth/CoreBluetooth.h>, 视情况而定
5.注释警告解除方法: Build Settings -> Documentations Comments -> 将YES改为NO
6.由于CPCL、TSPL接口优化后, 老用户如果使用该版本的SDK, 则修改的内容较多, 因此SDK预留了PTOldCommandCPCL、PTOldCommandTSPL两个类, 他们分别保留了之前旧的接口
7.每次发送数据, SDK都会把receiveDataBlock、sendSuccessBlock、sendFailureBlock、sendProgressBlock置为空
8.引用SDK:
  OC: #import <PrinterSDK/PrinterSDK.h>
  Swift:  import PrinterSDK
```

二、SDK开放类说明

2.0 PTPrinter

> 外设的属性类，比如说外设的名称name、mac地址、蓝牙的广播包advertisement、蓝牙的uuid、信号强度、WiFi相关的router和ip等等，当扫描到外设或者连接外设时，所传的参数就是属性类

```
外设的属性类,详情在PTPrinter文档中
```

2.1 PTDispatcher

```
调度通讯类, 详情在PTDispatcher文档中
```

2.2 PTCommandCPCL

```
CPCL指令接口类, 详情在cpcl指令接口文档
```

2.3 PTCommandESC

```
ESC指令接口类，详情在ESC指令接口文档
```

2.4 PTCommandTSPL

```
TSPL指令接口类，详情在TSPL指令接口文档
```

2.5 PTCommandZPL

```
ZPL指令接口类，详情在ZPL指令接口文档
```

2.6 PTEncode

> 发送Text的编码类，默认是kCFStringEncodingGB_18030_2000的编码

- 编码

  ```
  /** encoding */
  + (NSData *)encodeDataWithString:(NSString *)string;

  /** support various encoding */
  + (NSData *)encodeDataWithString:(NSString *)string encodingType:
  (CFStringEncodings)encodeType;
  ```

~~~
* 解码

  ```objective-c
  /** decoding */
  + (NSString *)decodeStringWithData:(NSData *)data;

  /** support various decoding */
  + (NSString *)decodeDataWithString:(NSData *)data encodingType:
(CFStringEncodings)encodeType;
~~~

2.7 PTBitmap

> 图片处理类,一般在SDK里已经处理

- 枚举

```
*!
 *  \~chinese
 *  压缩模式
 *
 *  \~english
 *  Compress Mode
 */
typedef NS_ENUM(NSInteger,PTBitmapCompressMode) {

    PTBitmapCompressModeNone = 0,   /*! *\~chinese 不压缩 *\~english
None */
    PTBitmapCompressModeZPL2 = 16,  /*! *\~chinese ZPL2压缩算法
*\~english ZPL2 compress */
```

```
    PTBitmapCompressModeTIFF = 32,  /*! *\~chinese TIFF压缩算法
*\~english TIFF compress */
    PTBitmapCompressModeLZO = 48,   /*! *\~chinese LZO压缩算法
*\~english LZO compress */
};

/*!
 *  \~chinese
 *  位图模式
 *
 *  \~english
 *  Bitmap Mode
 */
typedef NS_ENUM(NSInteger, PTBitmapMode) {

    PTBitmapModeBinary = 0,     /*! *\~chinese 黑白二值图像 *\~english
Binary */
    PTBitmapModeDithering = 1,  /*! *\~chinese 灰阶抖动图像 *\~english
Dithering */
    PTBitmapModeColumn = 2,     /*! *\~chinese 无效 *\~english not
supported */
};
```

- 生成打印机图片数据

```
/*!
 *  \~chinese
 *  生成打印机打印图片数据
 *
 *  @param image    图片
 *  @param mode     生成的位图数据类型 简单的黑白二值化或者抖动处理
 *  @param compress 数据压缩类型
 *  @return         提供给打印机使用的图片数据
 *
 *  \~english
 *  Generate data of printing image for the printer
 *
 *  @param image    image
 *  @param mode     Type of generated bitmap data; simple black and
white image or dithering
```

- 其他接口

```
/*!
 *  \~chinese
 *  生成打印机打印图片数据
 *
 *  @param image    图片
 *  @param mode     生成的位图数据类型 简单的黑白二值化或者抖动处理
 *  @param compress 数据压缩类型
 *  @return         提供给打印机使用的图片数据
 *
 *  \~english
 *  Generate data of printing image for the printer
 *
 *  @param image    image
 *  @param mode     Type of generated bitmap data; simple black and
white image or dithering
 *  @param compress Type of data compression
 *  @return         Image data provided to the printer
 */
+ (NSData *)getImageData:(CGImageRef)image mode:(PTBitmapMode)mode
compress:(PTBitmapCompressMode)compress;

/*!
 *  \~chinese
 *  生成打印机打印图片数据
 *
 *  @param image    图片
 *  @param mode     生成的位图数据类型 简单的黑白二值化或者抖动处理
 *  @param compress 数据压缩类型
 *  @param command  指令类型
 *  @return         提供给打印机使用的图片数据
 *
 *  \~english
 *  Generate data of printing image for the printer
 *
 *  @param image    image
 *  @param mode     Type of generated bitmap data; simple black and
white image or dithering
 *  @param compress Type of data compression
 *  @param command  Type of command
 *  @return         Image data provided to the printer
 */
+ (NSData *)getImageData:(CGImageRef)image mode:(PTBitmapMode)mode
compress:(PTBitmapCompressMode)compress command:
(PTBitmapCommand)command;

/*!
 *  \~chinese
 *  用column算法生成的图片数据
 *
 *  @param sourceBitmap  输入数据
 *  @return             位图数据
 *
 *  \~english
 *  Image data generated by the column algorithm
 *
 *  @param sourceBitmap  input data
 *  @return              bitmap Data
 */
+ (NSData *)generateColumnData:(CGImageRef)sourceBitmap;


/*!
```

```
 *  \~chinese
 *  将bitmap数据转成图片
 *
 *  @param image    图片
 *  @param mode     生成的位图数据类型 简单的黑白二值化或者抖动处理
 *  @return         预览的图片
 *
 *  \~english
 *  Convert bitmap data to image
 *
 *  @param image    image
 *  @param mode     Type of generated bitmap data; simple black and
white image or dithering
 *  @return         Review Image
 */
+ (UIImage *)generateRenderingWithImage:(CGImageRef)image mode:
(PTBitmapMode)mode;
```

2.9 PTLabel

> **
>
> - 使用电子面单模板，只需要填充相应的表单数据，即可发送打印出一张面单。
> - 注意： 1. 当使用模板打印时，您必须填充我们提供的模板使用范例中所填充的所有表单项。
>
> 2. 如果有空数据项，比如申明价值为空，则传入@""空字符串。
> 3. 不同的模板，所要填充的数据项是不同的，具体以我们的范例为准。
>
> */
>
> /**

> - By using electronic waybill template, only filling in it accordingly can send and print it out。
> - Note： 1. When using template to print, you should fill in all the blanks as the template sample showed
>
> 2.If there is null data, e.g. claiming value is null, please input null character string @"".
>
> 3.The data to fill in differs depending on the template, please subject to the sample showed.
>
> */

- 属性和方法

```
@interface PTLabel : NSObject

@property(strong,nonatomic,readwrite) NSString *express_company;
 // 快递公司

@property(strong,nonatomic,readwrite) NSString *delivery_number;
 // 运单号
@property(strong,nonatomic,readwrite) NSString *order_number;
// 订单号

@property(strong,nonatomic,readwrite) NSString *distributing;
// 集散地
@property(strong,nonatomic,readwrite) NSString *barcode;
 // 条形码
@property(strong,nonatomic,readwrite) NSString *barcode_text;
// 条形码下方的字符
@property(strong,nonatomic,readwrite) NSString *qrcode;
// 二维码
@property(strong,nonatomic,readwrite) NSString *qrcode_text;
 // 二维码下方的字符
```

```
@property(strong,nonatomic,readwrite) NSString *receiver_name;
 // 收件人 名字
@property(strong,nonatomic,readwrite) NSString *receiver_phone;
// 收件人 电话
@property(strong,nonatomic,readwrite) NSString *receiver_address;
// 收件人 地址
@property(strong,nonatomic,readwrite) NSString *receiver_message;
// 收件人 信息

@property(strong,nonatomic,readwrite) NSString *sender_name;
 // 发件人 名字
@property(strong,nonatomic,readwrite) NSString *sender_phone;
// 发件人 电话
@property(strong,nonatomic,readwrite) NSString *sender_address;
// 发件人 地址
@property(strong,nonatomic,readwrite) NSString *sender_message;
// 发件人 信息

@property(strong,nonatomic,readwrite) NSString *article_name;
// 物品名
@property(strong,nonatomic,readwrite) NSString *article_weight;
// 物品重量

@property(strong,nonatomic,readwrite) NSString *amount_declare;
// 申明价值
@property(strong,nonatomic,readwrite) NSString *amount_paid;
 // 到付金额
@property(strong,nonatomic,readwrite) NSString
*amount_paid_advance;// 预付金额

- (NSData *)dataWithSourceFile:(NSString *)filePath;
- (NSData *)dataWithTSPL;
- (NSData *)getTemplateData:(NSString *)source labelDict:
(NSDictionary *)labelDict orderDetails:(NSArray *)orderDetails;
- (NSData *)getTemplateData:(NSString *)source labelDict:
(NSDictionary *)labelDict;
```

```
+ (NSData *)getPaperStauts; // 获取纸张状态

@end
```

2.10 PTOldCommandCPCL

> 该类是保留SDK3.0.0之前的版本的旧CPCL接口

2.11 PTOldCommandTSPL

> 该类是保留SDK3.0.0之前的版本的旧TSPL接口

2.12 PTCommandCommon

> 该类是共用的指令接口

- 获取打印机型号

```
/*!
 *  \~chinese
 *
 *  获取打印机型号,回的数据格式: 51333142 5400, 最后一个字节00前面的是有效数据
 *
 *  \~english
 *
 *  Get printer model,eg:51333142 5400,The last byte 00 is preceded
by valid data
 *
 */
- (void)getPrinterModelName;
```

三、如何连接外设说明

> 连接外设用到的几个方法，具体情况参考Demo

- BLE

```
//Swift4.2:

//获取蓝牙状态，需要在子线程调用
PTDispatcher.share().getBluetoothStatus()

//开始扫描蓝牙
PTDispatcher.share().scanBluetooth()

//扫描到的蓝牙，以数组的形式返回
PTDispatcher.share()?.whenFindAllBluetooth({ (array) in

 })
 //关闭扫描，连接成功后会自动关闭
 PTDispatcher.share().stopScanBluetooth()

 //连接打印机
 PTDispatcher.share().connect(printer)

 //断开连接
 PTDispatcher.share().unconnectPrinter(PTDispatcher.share().printerC
onnected)

  //连接成功
 PTDispatcher.share().whenConnectSuccess {
  }
//连接失败
PTDispatcher.share().whenConnectFailureWithErrorBlock { (error) in
 }
```

- WiFi

```
//要先连接路由器
let router = PTRouter.init()
if router.connected {
    SVProgressHUD.show(withStatus: PRTLocalKey("Scanning network
segment, pls. wait"))
    PTDispatcher.share().scanWiFi({ [weak self](ptArray) in
         SVProgressHUD.showSuccess(withStatus: PRTLocalKey("Scan
success"))
         self?.printListArray = ptArray as! [PTPrinter]
         self?.tableView.reloadData()
     })
 }else {
   UIAlertController.showActionAlert(PRTLocalKey("Prompt"), message:
    PRTLocalKey("Connect router in iOS system setting"), confirm:
PRTLocalKey("Setting"), confirmHandle: { (_) in
                let url = URL.init(string: "APP-Prefs:root=WIFI")
                UIApplication.shared.openURL(url!)
            })
        }

 //连接打印机
 PTDispatcher.share().connect(printer)

 //断开连接
 PTDispatcher.share().unconnectPrinter(PTDispatcher.share().printerC
onnected)

  //连接成功
 PTDispatcher.share().whenConnectSuccess {
  }
//连接失败
PTDispatcher.share().whenConnectFailureWithErrorBlock { (error) in
 }
```

- MFI

```
DispatchQueue.main.async {
            PTDispatcher.share()?.showBluetoothAccessorys("",
completion: {
                guard let error = $0 as NSError? else { return }
                switch error.code {
                    case 0:
                        SVProgressHUD.showInfo(withStatus: "已经连
接")
                    case 1:
                        SVProgressHUD.showInfo(withStatus: "未找到
设")
                    case 2:
                            SVProgressHUD.showInfo(withStatus: "取消连
接")
                        case 3:
                            SVProgressHUD.showInfo(withStatus: "连接失
败")
                        default:
                            break
                    }
                })
            }

PTDispatcher.share()?.connect(mfiDevices[indexPath.row])

        PTDispatcher.share()?.whenConnectSuccess { [weak self] in
            guard let self = self else { return }
            SVProgressHUD.dismiss()
            self.showAlert(title: "Select paper size".localized,
buttonTitles: ["2\" (384dots)", "3\" (576dots)","3\" (640dots)",
"4\" (832dots)","8\" (2360dots)", "12\" (3540dots)"], handler: {
(selectedButtonIndex) in
                self.didSelectPaperSize(buttonIndex:
selectedButtonIndex)
            })
        }

        PTDispatcher.share()?.whenConnectFailureWithErrorBlock {
(error) in
            var errorStr: String?
            switch error {
                case .bleTimeout:
                    errorStr = "Connection timeout".localized
                case .bleValidateTimeout:
                    errorStr = "Vertification timeout".localized
                case .bleUnknownDevice:
                    errorStr = "Unknown device".localized
                case .bleSystem:
                    errorStr = "System error".localized
                case .bleValidateFail:
                    errorStr = "Vertification failed".localized
                case .bleDisvocerServiceTimeout:
                    errorStr = "Connection failed".localized
                default:
                    break
            }
            if let temp = errorStr {
                SVProgressHUD.showError(withStatus: temp)
            }
        }

        PTDispatcher.share()?.whenReceiveData({ (_) in

        })
```

- 发送数据

```
PTDispatcher.share().send(data)

 //进度
PTDispatcher.share()?.whenSendProgressUpdate({ (progress) in


})

//发送成功
PTDispatcher.share().whenSendSuccess {
 }

//发送失败
PTDispatcher.share().whenSendFailure { [weak self] in

```

```
 }

// 接收蓝牙返回数据
PTDispatcher.share().whenReceiveData { (temp) in

 }
```

四、指令使用案例

4.0 SDK提供的功能

- 打印格式条形码
- 打印二维码
- 打印文本
- 打印图片（黑白、灰阶抖动）
- 打印小票

通过本 Demo，您可以了解到：

- 如何导入、链接和使用 `PrinterSDK.framework` 框架
- 如何通过 Bluetooth 4.0和 便携式 BLE 蓝牙打印机进行通讯
- 如何通过 WiFi 和便携式 WiFi 打印机进行通讯
- 如何使用打印机指令集中的基础指令，并根据自己的需求分装成为功能

4.1 通过模板打印

　通过模板打印，电子面单的样式已经预先编辑好，只需要填充相应的数据项，即可打印输出一张面单。这里以申通快递为例，具体代码见 PTTestTSC 类。

1.填充数据

```
// 使用说明:
// 1. 初始化一个 NSMutableDictionary，在相应的键值下塞入对应的的数据，键值必须是
下面样例中用到的键值。
// 2. 如果一个数据项没有数据 那么也需要设置成空字符串@""，比如 [templateDict
setObject:@"" forKey:kCollection];
PTLabelTemplate *template = [[PTLabelTemplate alloc] init];
NSMutableDictionary *templateDict = [[NSMutableDictionary alloc] init];

[templateDict setObject:@"363604310467" forKey:LTBarcode];
[templateDict setObject:@"上海 上海市 长宁区" forKey:LTDistributing];

[templateDict setObject:@"申大通" forKey:LTReceiver];
[templateDict setObject:@"13826514987" forKey:LTReceiverContact];
[templateDict setObject:@"上海市宝山区共和新路4719弄共和小区12号306室"
forKey:LTReceiverAddress];

[templateDict setObject:@"快小宝" forKey:LTSender];
[templateDict setObject:@"13826514987\r\n" forKey:LTSenderContact];
[templateDict setObject:@"上海市长宁区北瞿路1178号（鑫达商务楼）1号楼305室"
forKey:LTSenderAddress];

[templateDict setObject:@"SHENTONG" forKey:LTExpressCompany];

NSData *cmdData = [template getShenTongTemplate:templateDict];
```

2.读入模板

> 使用时需要把模板文件拖入你的工程中。模板文件是 TXT 纯文本文件。

```
NSString *path = [[NSBundle mainBundle] pathForResource:@"ShenTong"
ofType:@"txt"];
NSData *templateData = [template getTemplateDataWithFilePath:path];
```

3.结合模板和数据

```
NSMutableData *finalData = [[NSMutableData alloc]
initWithData:templateData];
[finalData appendData:cmdData];
```

4.发送数据

```
[[PTDispatcher share] sendData:finalData];
```

4.2 CPCL案例

```
PTCommandCPCL *cmd = [[PTCommandCPCL alloc] init];
//初始化标签
[cmd cpclLabelWithOffset:0 hRes:200 vRes:200 height:300 quantity:1];

 UIImage *logo = [UIImage imageNamed:@"abcd.jpg"];
 NSData *bmpData = [PTBitmap getImageData:logo.CGImage
mode:PTBitmapModeDithering compress:PTBitmapCompressModeNone];
[cmd cpclCompressedGraphicsWithImageWidth:logo.size.width
imageHeight:logo.size.height x:20 y:0 bitmapData:bmpData];
//打印
[cmd cpclPrint];
[[PTDispatcher share] sendData:cmd.cmdData];
```

4.3 ESC案例

```
PTCommandESC *cmd = [[PTCommandESC alloc] init];
    [cmd initializePrinter];
    [cmd setJustification:0];
    [cmd setLineSpacing:10];
    UIImage *logoImage = [UIImage imageNamed:@"boliji.jpg"];
    //PTBitmapCompressModeLZO压缩算法:  支持汉码机型
    BOOL ret = [cmd appendRasterImage:logoImage.CGImage
mode:PTBitmapModeBinary compress:PTBitmapCompressModeLZO package:YES];
    [cmd printAndReturnStandardMode];
    if (ret) {
        NSData *sendData = [cmd getCommandData];
        [PrinterPort sendWithData:sendData];
    }else {
        NSLog(@"The data exceeds the cache and cannot be printed.");
        [ProgressHUD showError:@"print fail"];
    }
```

4.4 TSPL案例

```
PTCommandTSPL *tspl = [[PTCommandTSPL alloc] init];
//清除缓存
[tspl setCLS];
//设置打印区域
[tspl setPrintAreaSizeWithWidth:80 Height:80];
[tspl addBitmapWithXPos:0 YPos:0 Mode:1 image:image.CGImage
bitmapMode:PTBitmapModeBinary compress:PTBitmapCompressModeNone];
//设置打印份数
[tspl printWithSets:1 Copies:1];
[[PTDispatcher share] sendData:tspl.cmdData];
```

4.5 ZPL案例

```
PTCommandZPL *zpl = [[PTCommandZPL alloc] init];

//新的标签起始格式
[zpl XA_FormatStart];
```

```
    [zpl LL_LabelLength:400];
    [zpl PW_PrintWidth:700];
    [zpl A_SetFontWithOrientation:@"N" height:50 width:60 location:@"B"
    fontName:@"CYRI_UB" extension:@"FNT"];
    [zpl FO_FieldOriginWithXAxis:100 YAxis:100];
    [zpl FD_FieldData:@"Wireless Printer Fonts"];
    //字段分隔符
    [zpl FS_FieldSeparator];
    //结束格式
    [zpl XZ_FormatEnd];
    //发送数据
    [[PTDispatcher share] sendData:zpl.cmdData];
```