

- 一、SDK导入说明
- 二、SDK开放类说明
 - 2.0 PTPrinter
 - 2.1 PTDispatcher
 - 2.2 PTCommandCPCL
 - 2.3 PTCommandESC
 - 2.4 PTCommandTSPL
 - 2.5 PTCommandZPL
 - 2.6 PTEncode
 - 2.7 PTBitmapManager
 - 2.9 PTLabel
 - 2.10 PTOldCommandCPCL
 - 2.11 PTOldCommandTSPL
 - 2.12 PTCommandCommon
- 三、如何连接外设说明
- 四、指令使用案例
 - 4.0 SDK提供的功能
 - 4.1 通过模板打印
 - 4.2 CPCL案例
 - 4.3 ESC案例
 - 4.4 TSPL案例
 - 4.5 ZPL案例

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.如果是蓝牙连接, 需要增加两个蓝牙权限:
 - Privacy - Bluetooth Always Usage Description
 - Privacy - Bluetooth Peripheral Usage Description
- 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

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

- 属性

```
/*!
 *  \~chinese
 *  打印机名称
 *
 *  \~english
 *  Printer name
 */
@property(strong, nonatomic, readwrite) NSString *name;

/*!
 *  \~chinese
 *  打印机mac地址
 *
 *  \~english
 *  Printer mac address
 */
@property(strong, nonatomic, readwrite) NSString *mac;

/*!
 *  \~chinese
 *  打印机蓝牙模块
 *
 *  \~english
 *  Printer Bluetooth module
 */
@property(assign, nonatomic, readwrite) PTPrinterModule module;

/*!
 *  \~chinese
 *  发现蓝牙时获取到的广播信息
 *
 *  \~english
 *  The broadcast information obtained when Bluetooth is found
 */
@property(strong, nonatomic, readwrite) NSDictionary *advertisement;

/*!
 *  \~chinese
 *  蓝牙外设UUID
 *
 *  \~english
 *  Bluetooth peripherals UUID
 */
```

```

*/
@property(strong, nonatomic, readwrite) NSString *uuid;

/*!
 * \~chinese
 * 发现外设时获取到的信号强度值，单位分贝
 *
 * \~english
 * The signal strength value obtained when peripherals are found, unit is db
 */
@property(strong, nonatomic, readwrite) NSNumber *rssi;

/*!
 * \~chinese
 * 信号强度等级分0-5级
 *
 * \~english
 * Signal strength level is from 0 to 5
 */
@property(strong, nonatomic, readwrite) NSNumber *strength;

/*!
 * \~chinese
 * 由信号强度计算的距离
 *
 * \~english
 * The distance calculated by signal strength
 */
@property(strong, nonatomic, readwrite) NSNumber *distance;

/*!
 * \~chinese
 * 蓝牙外设
 *
 * \~english
 * Bluetooth peripherals
 */
@property(strong, nonatomic, readwrite) CBPeripheral *peripheral;

/*!
 * \~chinese
 * 外设的ip地址
 *
 * \~english
 * IP
 */
@property(strong, nonatomic, readwrite) NSString *ip;

/*!

```

```

*   \~chinese
*   端口
*
*   \~english
*   port
*/
@property(strong, nonatomic, readwrite) NSString *port;

```

2.1 PTDispatcher

数据的回调类

- 1.三个枚举分别表示蓝牙的通讯方式、打印状态回调和连接失败的类型
- 2.定义数据回调的block
- 3.将block当做方法的参数

- 枚举

```

/*!
 *   \~chinese
 *   连接模式
 *
 *   \~english
 *   Connect Mode
 */
typedef NS_ENUM(NSInteger, PTDispatchMode) {
    /*! *~chinese 未知类型 *~english Unknown */
    PTDispatchModeUnconnect = 0,
    /*! *~chinese 蓝牙 *~english BLE */
    PTDispatchModeBLE = 1,
    /*! *~chinese 无线 *~english WiFi */
    PTDispatchModeWiFi = 2
};

/*!
 *   \~chinese
 *   打印完成后打印机返回的状态
 *
 *   \~english
 *   Printer Status
 */
typedef NS_ENUM(NSInteger, PTPrintState) {
    /*! *~chinese 打印成功 *~english Print success */
    PTPrintStateSuccess = 0xcc00,
    /*! *~chinese 打印失败 (缺纸) *~english Print failure (paper out) */
    PTPrintStateFailurePaperEmpty = 0xcc01,
    /*! *~chinese 打印失败 (开盖) *~english Print failure (cover open) */
    PTPrintStateFailureLidOpen = 0xcc02
};

```

```
};

typedef NS_ENUM(NSInteger, PTConnectError) {
    PTConnectErrorBleTimeout = 0, ///< ~chinese 蓝牙连接超时
    \~english Bluetooth connection timed out
    PTConnectErrorBleDisvocerServiceTimeout = 1, ///< ~chinese 获取服务超时
    \~english Get service timed out
    PTConnectErrorBleValidateTimeout = 2, ///< ~chinese 验证超时 \~english
    Print Verification timed out
    PTConnectErrorBleUnknownDevice = 3, ///< ~chinese 未知设备 \~english
    Unknown device
    PTConnectErrorBleSystem = 4, ///< ~chinese 系统错误 \~english
    System error
    PTConnectErrorBleValidateFail = 5, ///< ~chinese 验证失败 \~english
    Verification failed
    PTConnectErrorStreamTimeout = 6, ///< ~chinese 流打开超时
    \~english Stream open timeout
    PTConnectErrorStreamEmpty = 7, ///< ~chinese 打开的是空流
    \~english Empty stream
    PTConnectErrorStreamOccured = 8, ///< ~chinese 流发生错误
    \~english An error has occurred on the stream
};
```

- 属性

```
/*!
 * \~chinese
 * 连接成功后的打印机属性类
 *
 * \~english
 * Printer property after connect success
 */
@property (strong, nonatomic, readonly) PTPrinter
    *printerConnected;

/*!
 * \~chinese
 * 连接方式
 *
 * \~english
 * Connect style
 */
@property (assign, nonatomic) PTDispatchMode mode;

/*!
 * \~chinese
 * 数据发送成功
 */
```

```

*   \~english
*   Send data success
*/
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
sendSuccessBlock;

/*!
*   \~chinese
*   数据发送失败
*
*   \~english
*   Send data fail
*/
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
sendFailureBlock;

/*!
*   \~chinese
*   发送数据的进度条
*
*   \~english
*   Send progress
*/
@property (copy, nonatomic, readwrite) PTNumberParameterBlock
sendProgressBlock;

/*!
*   \~chinese
*   接收外设返回的数据
*
*   \~english
*   ReceiveDara
*/
@property (copy, nonatomic, readwrite) PTDataParameterBlock
receiveDataBlock;

/*!
*   \~chinese
*   打印完成后返回的状态
*
*   \~english
*   PrintStatus
*/
@property (copy, nonatomic, readwrite) PTPrintStateBlock
printStateBlock;

/*!
*   \~chinese
*   发现外设

```

```

*
*  \~english
*  FindDevice
*/
@property (copy, nonatomic, readwrite) PTPrinterParameterBlock
    findBluetoothBlock;

/*!
*  \~chinese
*  发现所有的外设
*
*  \~english
*  FindAllDevice
*/
@property (copy, nonatomic, readwrite) PTPrinterMutableArrayBlock
    findAllPeripheralBlock;

/*!
*  \~chinese
*  连接成功
*
*  \~english
*  Connect success
*/
@property (copy, nonatomic, readwrite) PTEmptyParameterBlock
    connectSuccessBlock;

/*!
*  \~chinese
*  连接失败
*
*  \~english
*  Connect fail
*/
@property (copy, nonatomic, readwrite) PTBluetoothConnectFailBlock
    connectFailBlock;

/*!
*  \~chinese
*  断开连接
*
*  \~english
*  unconnect
*/
@property (copy, nonatomic, readwrite) PTUnconnectBlock
    unconnectBlock;

/*!
*  \~chinese

```

```

*  外设的信号强度
*
*  \~english
*  Rssi
*/
@property (copy, nonatomic, readonly) PTNumberParameterBlock
readRSSIBlock;

/*!
*  \~chinese
*  外设过滤器
*
*  \~english
*  Peripheral filter
*/
@property (copy, nonatomic, readonly) PTPeripheralFilterBlock
peripheralFilterBlock;

```

- 方法

```

/*!
*  \~chinese
*  发送数据
*
*  @param data  发送的数据
*
*  \~english
*  Send data
*
*  @param data Send data
*/
- (void)sendData:(NSData *)data;

/*!
*  \~chinese
*  暂停发送
*
*  \~english
*  pause send
*
*/
- (void)pausewriteData;

/*!
*  \~chinese

```



```

* 继续发送
*
*
* \~english
* resume send
*
*/
- (void)resumewriteData;

/*!
* \~chinese
* 开始扫描蓝牙
*
* \~english
* Start scanning Bluetooth
*/
- (void)scanBluetooth;

/*!
* \~chinese
* 停止扫描蓝牙，连接成功后SDK会自动停止扫描
*
* \~english
* Stop scanning Bluetooth, The SDK will automatically stop scanning after the
connection is successful.
*/
- (void)stopScanBluetooth;

/*!
* \~chinese
* 扫描Wi-Fi
*
* @param wifiAllBlock 扫描到的外设数组
*
* \~english
* Scan Wi-Fi
*
* @param wifiAllBlock Scanned peripheral array
*/
- (void)scanWiFi:(PTPrinterMutableArrayBlock)wifiAllBlock;

/*!
* \~chinese
* 获取已发现的所有打印机，每新发现新的打印机或隔三秒调用一次
*
* @param bluetoothBlock 外设数组
*
* \~english

```

```

    * Get all the printers found, trigger once when finding new printer or trigger
    once every 3 seconds
    *
    * @param bluetoothBlock Scanned peripheral array
    */
- (void)whenFindAllBluetooth:(PTPrinterMutableArrayBlock)bluetoothBlock;

/*!
 * \~chinese
 * 发现蓝牙回调, coreBlueTooth框架每发现一台打印机就会调用
 *
 * @param bluetoothBlock 参数为发现的打印机对象
 *
 * \~english
 * Trigger this method when finding Bluetooth, coreBlueTooth will trigger it when
    finding one printer
 *
 * @param bluetoothBlock The parameter is the discovered printer object
 */
- (void)whenFindBluetooth:(PTPrinterParameterBlock)bluetoothBlock;

/*!
 * \~chinese
 * 连接设备更新RSSI回调
 *
 * @param readRSSIBlock 参数是型号强度
 *
 * \~english
 * Trigger this method when connecting new device to update RSSI
 *
 * @param readRSSIBlock Trigger block, parameter is the signal strength of
    connecting printer
 */
- (void)whenReadRSSI:(PTNumberParameterBlock)readRSSIBlock;

/*!
 * \~chinese
 * 连接打印机
 *
 * @param printer 要连接的打印机
 *
 * \~english
 * Connect printer
 *
 * @param printer Connected printer
 */
- (void)connectPrinter:(PTPrinter *)printer;

/*!

```

```

*   \~chinese
*   断开打印机连接
*
*   @param printer 要断开的打印机
*
*   \~english
*   Disconnect printer
*
*   @param printer Unconnect printer
*/
- (void)unconnectPrinter:(PTPrinter *)printer;

/*!
*   \~chinese
*   连接成功回调
*
*   @param connectSuccessBlock 连接成功的回调
*
*   \~english
*   Trigger this method when connecting successfully
*
*   @param connectSuccessBlock Trigger block
*/
- (void)whenConnectSuccess:(PTEmptyParameterBlock)connectSuccessBlock;

/*!
*   \~chinese
*   连接失败的回调
*
*   @param connectFailBlock 连接失败返回的错误类型
*
*   \~english
*   When connect error is occurred
*
*   @param connectFailBlock block block with connect error parameter
*/
- (void)whenConnectFailureWithErrorBlock:
(PTBluetoothConnectFailBlock)connectFailBlock;

/*!
*   \~chinese
*   断开连接的回调
*
*   @param unconnectBlock 回调的Block
*
*   \~english
*   Trigger this method when disconnecting
*
*   @param unconnectBlock Trigger block

```

```

*/
- (void)whenUnconnect:(PTUnconnectBlock)unconnectBlock;

/*!
 * \~chinese
 * 数据发送成功的回调
 *
 * @param sendSuccessBlock 回调block
 *
 * \~english
 * Callback for successful data transmission
 *
 * @param sendSuccessBlock Trigger block
 */
- (void)whenSendSuccess:(PTEmptyParameterBlock)sendSuccessBlock;

/*!
 * \~chinese
 * 数据发送失败的回调
 *
 * @param sendFailureBlock 回调block
 *
 * \~english
 * Data send failure
 *
 * @param sendFailureBlock Trigger block
 */
- (void)whenSendFailure:(PTEmptyParameterBlock)sendFailureBlock;

/*!
 * \~chinese
 * 数据发送进度的回调
 *
 * @param sendProgressBlock 回调block
 *
 * \~english
 * Callback of data transmission progress
 *
 * @param sendProgressBlock Trigger block
 */
- (void)whenSendProgressUpdate:(PTNumberParameterBlock)sendProgressBlock;

/*!
 * \~chinese
 * 接收到数据回调
 *
 * @param receiveDataBlock 回调block
 *
 * \~english

```

```

* Received data callback
*
* @param receiveDataBlock Trigger block
*/
- (void)whenReceiveData:(PTDataParameterBlock)receiveDataBlock;

/*!
* \~chinese
* 接收到打印机打印状态回调，针对CPCL ESC指令
*
* @param printStateBlock 回调block
*
* \~english
* Trigger this method when receiving print state ,For CPCL and ESC instructions
*
* @param printStateBlock Trigger block
*/
- (void)whenUpdatePrintState:(PTPrintStateBlock)printStateBlock;

/*!
* \~chinese
* 手机的蓝牙是否打开,YES:打开, NO:关闭
*
* @return 状态
*
* \~english
* whether the Bluetooth of the mobile phone is turned on,YES:open NO:close
*
* @return status
*/
- (BOOL)getBluetoothStatus;

/*!
* \~chinese
* 设置蓝牙连接超时时间
*
* @param timeout 超时时间，单位秒
*
* \~english
* Set the time of Bluetooth timeout
*
* @param timeout timeout,unit is second
*/
- (void)setupBleConnectTimeout:(double)timeout;

/*!
* \~chinese
* 设置外设过滤block
*

```

```

* @param block 回调block
*
* \~english
* Set peripheral filter block
*
* @param block Trigger block
*/
- (void)setupPeripheralFilter:(PTPeripheralFilterBlock)block;

/*!
* \~chinese
* 设置SDK中心
*
*
* @param manager 中心
* @param delegate 接收中心代理消息的对象
*
* \~english
* Set the SDK Center
*
* @param manager Center
* @param delegate The object that receives the central proxy message
*
*/
- (void)registerCentralManager:(CBCentralManager *)manager delegate:
(id<CBCentralManagerDelegate>)delegate;

/*!
* \~chinese
* 注销代理
*
* \~english
* unregister Delegate
*
*/
- (void)unregisterDelegate;

```

2.2 PTCommandCPCL

CPCL指令接口类，详情在cpc1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
- 解码

2.7 PTBitmapManager

图片处理类,一般在SDK里已经处理

- 枚举

```
typedef NS_ENUM(NSInteger, PTBitmapCompressMode) {
    /*! *~chinese 不压缩 *~english None */
    PTBitmapCompressModeNone = 0,
    /*! *~chinese LZ0压缩算法 *~english LZ0 compress */
    PTBitmapCompressModeLZ0 = 48
};

typedef NS_ENUM(NSInteger, PTBitmapMode) {
    /*! *~chinese 黑白二值图像 *~english Binary */
    PTBitmapModeBinary = 0,
    /*! *~chinese 扩散抖动 *~english Diffusion dithering algorithm */
    PTBitmapModeDithering = 1,
    /*! *~chinese 聚集抖动算法 *~english Aggregate dithering algorithm */
    PTBitmapModeCluster = 2
};
```

- 接口

```

/// 生成二值抖动图片数据
/// @param image 图片
/// @param mode 图片效果，这边如果选择灰阶模式，那么则默认SDK处理图片
/// @param compress 压缩模式
/// @param package 是否需要分包
/// @param reverse 数据是否要反转，eg:TSPL的图片需要反转
+ (NSData *)generateGenralDataWithImage:(UIImage *)image mode:(PTBitmapMode)mode
compress:(PTBitmapCompressMode)compress package:(BOOL)package reverse:
(BOOL)reverse;

/// 生成位图数据(该数据需要指令类的对应接口再处理，不可直接下发)
/// @param image 图片
/// @param mode 图片效果
+ (NSData *)generateBitmapDataWithImage:(UIImage *)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

```
//Swift5.0:  
  
//获取蓝牙状态, 该接口需要在AppDelegate先初始化PTDispatcher.share()  
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().disconnectPrinter(PTDispatcher.share().printerConnected)

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

```

- WiFi

```

let printer = PTPrinter()
printer.ip = "xxx.xxx.xxx.xxx"
printer.module = .WiFi
printer.port = "9100"

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

//断开连接
PTDispatcher.share().disconnectPrinter(PTDispatcher.share().printerConnected)

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

}

/// POS ESC command support
PTDispatcher.share().whenESCPrintSuccess { _ 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:LTRceiver];
[templateDict setObject:@"13826514987" forKey:LTRceiverContact];
[templateDict setObject:@"上海市宝山区共和新路4719弄共和小区12号306室"
forKey:LTRceiverAddress];

[templateDict setObject:@"快小宝" forKey:LTSender];
[templateDict setObject:@"13826514987\r\n" forKey:LTSenderContact];
[templateDict setObject:@"上海市长宁区北翟路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 cpc1LabelWithOffset: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 cpc1CompressedGraphicsWithImageWidth:logo.size.width imageHeight:logo.size.height
x:20 y:0 bitmapData:bmpData];
//打印
[cmd cpc1Print];
[[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:@"boliiji.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 *tsp1 = [[PTCommandTSPL alloc] init];
//清除缓存
[tsp1 setCLS];
//设置打印区域
[tsp1 setPrintAreaSizeWithwidth:80 Height:80];
[tsp1 addBitmapWithXPos:0 YPos:0 Mode:1 image:image.CGImage
bitmapMode:PTBitmapModeBinary compress:PTBitmapCompressModeNone];
//设置打印份数
[tsp1 printwithSets:1 Copies:1];
[[PTDispatcher share] sendData:tsp1.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];
```