一、SDK导入说明

1.A4的蓝牙返回数据SDK通过PLMT800Resolver类做了解析，通过PLDataDispatcher调度类的resolver属性（这个属性是PLDataResolver协议）返回一个Block，这个Block的返回值是一个id类型，基本上返回的数据类型是PLMT800ResolverModel模型类，只需要解析PLMT800ResolverModel模型即可

2.由于SDK用PLMT800Resolver做了解析，SDK并没有去初始化这个PLMT800Resolver类，因为SDK中其他的机型不需要这个解析，所以APP需要初始化这个PLMT800Resolver，我建议是在连接时根据机型去初始化，参考Demo：
```
PLDataDispatcher.shared().resolver = PLMT800Resolver()
```

3.固件升级通过- (void)writeFirmwareData:(NSData *)data progress:(void(^_Nullable)(NSProgress *))block fail:(void(^_Nullable)(void))failBlock接口发送

4.A4的图片发送通过- (void)writeA4ImageDatas:(NSMutableArray *)datas progress:(void(^_Nullable)(NSProgress *))block fail:(void(^_Nullable)(void))failBlock接口

5.其他数据通过- (void)writeData:(NSData *)data或者- (void)writeData:(NSData *)data progress:(void(^_Nullable)(NSProgress *))block fail:(void(^_Nullable)(void))failBlock接口发送

6.由于SDK变成动态库，在导入时需要在TARGETS->General->Frameworks中选择对应的SDK，在Embed中选择Embed&Sign

```
7.使用MFI时在Info.plist增加一下代码:
  <key>UISupportedExternalAccessoryProtocols</key>
  <array>
    <string>com.hprt.printer</string>
  </array>
  注:不用mfi框架时删除
8.如果有用MFI连接,那么上架的app审核信息中,在备注中填写MFi Product Plan ID

注:必须要初始化PLMT800Resolver类,PLDataDispatcher.shared().resolver = PLMT800Resolver()
,参考Demo

9.支持cocoaPods
  pod 'PLPrinterSDK ','~> 0.1.1'
```

二、接口说明

2.0 获取打印机信息

- 数据格式

```
1.自定义解析:
包头:rtsts(5byte)
打印机状态(2Byte)
打印机是否空闲(1 Byte):   1-空闲    0-非空闲
电池电量百分比(1Byte)
自动关机时间(4Byte)
打印浓度(1Byte)
纸张类型(1Byte)
TPH温度(2Byte,int16类型)
+END(字符)

2.SDK解析:SDK已经通过PLMT800Resolver解析,通过PLMT800ResolverModel模型的type属性判断返回
的类型,这个接口用PLMT800Info类去解析,直接初始化传数据即可解析,参考Demo
```

- 接口

```
+ (NSData *)getPrinterStatusInfo;
```

2.1 获取碳带剩余量

- 数据格式

```
1.自定义解析：包头“rbspl” + 碳带余量 int型，4Byte， 单位mm
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型
```

- 接口

```
+ (NSData *)getRibbonRemainCount;
```

2.2 获取碳带耗材品牌

- 数据格式

```
1.自定义解析：包头“rbbnd” + 字符串长度1字节（包括00） + 字符串，以00结尾
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型
```

- 接口

```
+ (NSData *)getRibbonConsumablesBrandInfo;
```

2.3 获取打印机序列号

- 数据格式

```
1.自定义解析：包头“sn”+ 字符串长度1字节（包括00）+字符串，以00结尾
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型
```

- 接口

```
+ (NSData *)getPrinterSerialNumber;
```

2.4 获取打印机固件版本

- 数据格式

```
1.自定义解析：包头“ver” + 字符串长度1字节（包括00）+ 字符串，以00结尾
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型
```

- 接口

```
+ (NSData *)getPrinterFirmwareVersion;
```

2.5 清空打印机缓存

- 数据格式

```
1.自定义解析：包头“can” + 00/01
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型
```

- 接口

```
+ (NSData *)clearPrinterBuffer;
```

2.6 获取打印机型号

- 数据格式

```
1.自定义解析：”getkey” + 0x0004(2bytes) + 0x20 (1bytes) + 型号数据
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型
```

- 接口

```
+ (NSData *)getPrinterName;
```

2.7 获取打印浓度

- 数据格式

```
1.自定义解析：”getkey” + 0x00cb(2bytes) + 0x01 (1bytes) + 浓度数据
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型，该类型是PLMT800CmdTypeCommonGet，对返回的model.data用PLMT800CommonCmdKey进行判断是哪种获取指令，参考Demo
```

- 接口

```
+ (NSData *)getPrinterDensity;
```

2.8 设置打印浓度

- 参数说明

```
1.density的范围1~3，越大则浓度越深
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型，该类型是PLMT800CmdTypeCommonSet，对返回的model.data用PLMT800CommonCmdKey进行判断是哪种设置指令，参考Demo
```

- 接口

```
+ (NSData *)setPrinterDensity:(Byte)density;
```

2.9 获取关机时间

- 数据格式

```
1.自定义解析：”getkey” + 0x0191(2bytes) + 0x04 (1bytes) + 时间数据
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型，该类型是PLMT800CmdTypeCommonGet，对返回的model.data用PLMT800CommonCmdKey进行判断是哪种获取指令，参考Demo
```

- 接口

```
+ (NSData *)getPrinterShutdownTime;
```

2.10 设置关机时间

- 参数说明

```
1.time = 0表示不自动关机，单位是分钟,下发4个字节数据
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型，该类型是PLMT800CmdTypeCommonSet，对返回的model.data用PLMT800CommonCmdKey进行判断是哪种设置指令，参考Demo
```

- 接口

```
+ (NSData *)setPrinterShutdownTime:(uint32_t)time;
```

2.11 打印图片

- 接口

```
/**
 * 图片分包数组
 * @param data 图片数据，数据大小为292 x 3288,图片的尺寸为2336 x 3288
 * @param height 图片高度 3288
 */
+ (NSMutableArray<PLMT800BitmapSlice *> *)sliceImageBitmap:(NSData *)data height:(uint16_t)height;
```

- 参数说明

```
1.data:数据大小为292 x 3288,图片的尺寸为2336 x 3288
2.height：传图片高度3288
```

- 图片数据获取
  - 可通过`PLBitmapManager`类的`+ (NSData *)generateGenralDataWithImage:(UIImage *)image watermark:(BOOL)watermark mode:(PLBitmapMode)mode compress:(PLBitmapCompressMode)compress package:(BOOL)package`接口获取
  - 也可以通过自己的图片算法得到的数据传入，不过需要注意图片数据大小，图片数据大小：292 x 3288

2.12 升级打印机固件

- 参数说明

```
1.固件数据
2.通过PLDataDispatcher类的- (void)writeFirmwareData:(NSData *)data progress:(void(^_Nullable)(NSProgress *))block fail:(void(^_Nullable)(void))failBlock接口下发
```

- 接口

```
/**
 * 封装好的固件数据
 * @param data 固件数据
 */
+ (NSData *)getFirmwareData:(NSData *)data;
```

2.13 自动回传状态

> 1.打印机状态变更时，打印机会自动回传实时状态
>
> 2.案例仅供参考，有时会返回多种状态，需要开发者自己计算

- 状态详情

```
typedef NS_OPTIONS(uint16_t, PLMT800PrinterStatus) {
    PLMT800PrinterStatusOK = 0,
    PLMT800PrinterStatusPaperAbsent = 1 << 0,///缺纸
    PLMT800PrinterStatusHighTemperature = 1 << 1,///高温
    PLMT800PrinterStatusLowTemperature = 1 << 2,///低温
    PLMT800PrinterStatusLowVoltage = 1 << 3,///低电量
    PLMT800PrinterStatusHeadOpened = 1 << 4,///开盖
    PLMT800PrinterStatusCarbonRibbonEnd = 1 << 5,///碳带用尽
    PLMT800PrinterStatusPaperSmashe = 1 << 6,///纸装歪
    PLMT800PrinterStatusCarbonRibbonNotAuthorization = 1 << 7,///碳带未授权
    PLMT800PrinterStatusBufferFull = 1 << 8,///缓存已满
};
```

- 接口

```
/** A4自动返回异常状态回调 */
- (void)autoA4DeviceStateBlock:(PLAutoMT800StateAction)automaticStateBlock;
```

2.14 自定义位图打印

- 接口

```
/**
 * 图片分包数组
 * @param data 算法处理后的图片数据，2336x3288
 * @param height 图片高度 3288
 * @param compress 图片是否要压缩,该版本暂不支持压缩
 * @param package 是否要分包，该版本暂不支持分包
 */
+ (NSMutableArray<PLMT800BitmapSlice *> *)customSliceImageBitmap:(NSData *)data
                                                          height:(uint16_t)height
                                                        compress:
(PLBitmapCompressMode)compress
                                                         package:(BOOL)package;
```

- 参数

参数	描述
data	通过自己的算法处理后的图片数据，2336x3288
height	图片的高度，3288
compress	图片是否需要压缩，暂时不支持压缩
package	是否需要分包，暂不支持分包

2.15 设置纸张类型

- 参数说明

```
1.type的范围0~2，0：连续纸 1：A4纸 2：黑标纸
2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型，该类型是PLMT800CmdTypeCommonSet，对返回的model.data用PLMT800CommonCmdKey进行判断是哪种设置指令，参考Demo
```

- 接口

```
+ (NSData *)setPrinterPaper:(Byte)type;
```

2.16 获取纸张类型

- 数据格式

  ```
  1.自定义解析：”getkey” + 0x00ce(2bytes) + 0x01 (1bytes) + 浓度数据
  2.SDK解析：SDK已经通过PLMT800Resolver解析，通过PLMT800ResolverModel模型的type属性判断返回的类型，该类型是PLMT800CmdTypeCommonGet，对返回的model.data用PLMT800CommonCmdKey进行判断是哪种获取指令，参考Demo
  ```

- 接口

  ```
  + (NSData *)getPrinterPaperType;
  ```