

iOSPrinterSDK development document

一、 Introduction

This development kit encapsulates the BLE Bluetooth communication function and the commonly used ESC/POS command set

1、Static library name: libPrintfESCLib.a

2、Header file name:

Head File Name	Discription
PrinterBluetoothManager.h	BLE Bluetooth communication
PrintfESCManager.h	ESC/POS command package
ESCTable.h	Form printing package

PrinterBluetoothManager.h The methods provided by the header file are as follows:

1. Global singleton **PrinterBluetoothManager**:

a) Get a single instance

+(instancetype) printerBluetoothManagerInstance;

2. Start or stop scanning for Bluetooth devices

a) Start scanning

-(void)beginScan;

b) Stop scanning

-(void)stopScan;

3.Connecting to Bluetooth devices

a) Connect the device

-(void)connect:(CBPeripheral *)peripheral;

b) Connect to the last connected Bluetooth, not connected by default

-(void)setIsOpenDefaultConnect:(BOOL)defaultConnect;

c) Is Bluetooth connected?

-(BOOL)isConnected;

4.Send data

a) Send data to the printer

-(BOOL)writeNSData:(NSData *)nsData;

b) Send data to the printer and read the result through block callback

-(void)writeAndRead:(NSData *)nsData readNSData:(ReadNSData)

readNSData;

5.Bluetooth related callback function

a) The callback closure for scanning to the device

-(void)blePeripheralFound:(void (^)(CBPeripheral *peripheral, NSNumber *rssi))block;

b) Callback closure for successful device connection

```
- (void)blePeripheralConnected:(void (^)(CBCentralManager
*central,CBPeripheral *peripheral))block;
```

b) Callback closure for device disconnection

```
- (void)blePeripheralDisconnected:(void (^)(CBCentralManager
*central,CBPeripheral *peripheral))block;
```

c) Read and print device data callback

```
@property (strong, nonatomic) ReadNSData _Nullable readNSData;
```

PrintfESCManger.h The methods provided by the header file are as follows:

1. Gloabl singleton **PrintfESCManger**:

a) Get a single instance

```
+(instancetype) createNew;
```

2. Control printer related

a) Initialize the printer

```
-(void)initPrinter;
```

b) Print and wrap

```
-(void)setPrint;
```

c) Print and feed paper n lines

```
-(void)setPrintAndFeedRow:(int)row;
```

d) Set absolute print position

```
-(void)setAbsolutePrintPosition:(int)where;
```

e) Choose alignment

```
-(void)setSelectJustification:(int)n;
```

When n is 0: left-justified

When n is 1: center aligned

When n is 2: right justified

f) Set left margin

```
-(void)setLeftMargin:(int)left;
```

3. Text printing related

a) Print Text

```
-(void)setText:(NSString*)content;
```

b) Set default line spacing

```
-(void)setDefaultLineSpace;
```

c) Set line spacing

```
-(void)setLineSpace:(int)space;
```

d) Select print mode

```
-(void)setPrintMode:(int)mode;
```

0: Standard ASCII code font (12 × 24)

1: Compressed ASCII code font B (9 × 17)

16: select double height mode

32: select double width mode

Multiple modes, just add them up

e) Select/cancel underline mode

`-(void)setUnderlineMode:(int)n;`

When n is 0: cancel the underline mode

When n is 1: select underline mode

f) Select/cancel bold mode

`-(void)setBoldMode:(int)mode;`

When n is 0, the bold mode is cancelled

When n is 1, select bold mode

g) Choose font

`-(void)setSelectFont:(int)font;`

When the n bit is 0, select the standard ASCII code font (12 × 24)

When the n bit is 1, select the compressed ASCII code font (9 × 17)

4. Barcode printing related

a) Print barcode

`-(void)setBarCodeWithType:(int)type barcodeStringPosition:(int)position
barcodeHeightInDot:(int)height barcodeWidth:(int)width
barcodeContent:(NSString*)content;`

type: Barcode type. 0: UPC-A, 1: UPC-B, 2: EAN13, 3: EAN8, 4: CODE39, 5: ITF, 6: CODABAR, 7: CODE93, 8: CODE128

barcodeStringPosition: Bar code character printing position. 0: Do not print, 1: Above the barcode, 2: Below the barcode, 3: Print both above and below the barcode.

barcodeHeightInDot: Bar code height. Height range $1 \leq n \leq 255$.

barcodeWidth: Bar code width. Barcode width range $2 \leq n \leq 6$

b) Print QR code

`-(void)setQRCode:(NSString *)content;`

5. Picture printing related

a) Print Image

`-(void)setImage:(UIImage *)image left:(int)left;`

6. Other

a) Full dotted line, 58mm

`-(void)setPlusLine_58;`

b) Full dotted line, 80mm

`-(void)setPlusLine_80;`

c) Print title and content, divided into two sides

`-(void)setTwoColumn:(NSString *)title content:(NSString *)content
paperMM:(int)paperMM;`

title: Title, which is the text on the left

content: Content, which is the text on the right

paperMM: Paper size, 58mm printer passes 48, 80mm printer passes 72

d) Print form

-(void)setTable:(ESCTable*)table;

e) Get print data

-(NSData *)getData;

f) Clear print data

-(void)clearData;

ESCTable.h The methods provided by the header file are as follows:

1. Form initialization

-(instancetype)init:(NSString*)column

regularExpression:(NSString*)regularExpression columnWidthArray:(int*)array

columnWidthLength:(int)length;

column: Table header

regularExpression: The cutting symbol of each heading of the column above

columnWidthArray: The number of bytes occupied by each header (each column), one English symbol, one byte, and one Chinese character, two bytes. 58mm printer, one line can print 32 bytes

columnWidthLength: How many columns

2. Set the alignment of the column data in the Table, the default is right-aligned

-(void)setColumnAlignRight:(BOOL)right;

3. Add a row of data

-(void)addRow:(NSString*)row;

Add a row of data. The data format is consistent with the header format. If the data of a cell exceeds the limited character width

It will automatically wrap and print, if you need to manually wrap, you can add "\n" where you need to wrap.

4. Get the text content generated by the table

-(NSString*)getTableText;