

Sanei ESC/POS Android SDK ユーザーマニュアル

2024年2月13日 Rev1.4.0

このマニュアルはSanei ESC/POS Android SDKに関する、お客様がAndroidアプリケーションの構築に必要な設計ガイドラインの情報を示しています。

本書の改定履歴

改定Rev	日付	アプリバージョン	改定内容
Rev1.0.0	2019年12月25日	1.0.0	初版リリース
Rev1.1.0	2020年4月13日	1.1.0	(1) SM4-21 プリンタをサポートプリンタに追加しました。 (2) インタフェースの仕様(BLE/WLAN)を拡張しました。 (3) BuildConfig クラスを追加しました。 (4) getStatus メソッド内の STATUS_2_BOOKED を削除し、 STATUS_2_LOWBATTERY を新たに追加しました。
Rev1.2.0	2021年8月2日	1.2.0	(1) SM4-31、SK1-41、SP1-21プリンタをサポートプリンタ に追加しました。
Rev1.2.1	2022年1月14日	1.2.1	(1) Android 12向けのビルドに対応しました。
Rev1.3.0	2023年1月5日	1.3.0	(1) SK5-31プリンタをサポートプリンタに追加しました。
Rev1.4.0	2024年2月13日	1.4.0	(1) サンプルプログラムをAndroid12以降に対応しました。

ご注意

- Sanei ESC/POS Android SDK は三栄電機株式会社（以下三栄電機といいます。）の著作物であり、本製品にかかる著作権その他の権利は三栄電機に帰属する。
- 三栄電機はSanei ESC/POS Android SDKに対応する三栄電機の製品を利用する目的で使用者に使用する権利を許諾（コピー及び配布は自由）する。
- 三栄電機は Sanei ESC/POS Android SDK に関して欠陥がないこと、このマニュアルに記載されている情報の使用に起因するいかなる損害に対しても責任を負うものではありません。
- 三栄電機は Sanei ESC/POS Android SDK の使用に関連して生じる直接的または、間接的な損失、損害などについていかなる場合も一切責任を負わないものとする。
- 使用者は日本国政府、または該当国の政府より必要な許可等を得ることなしに、Sanei ESC/POS Android SDK の全部または一部を直接または間接的に輸出することはできません。

三栄電機株式会社 2019

無断転載を禁じます。

本書の内容は断り無く変更することがあります。

Androidは、Google Inc.の商標です。

Windowsは、米国Microsoft Corporationの、米国、日本及びその他国における商標です。

Bluetooth のワードマーク及びロゴは Bluetooth SIG,Inc の登録商標です。

商標または登録商標であり、ライセンスに基づき使用されます。

その他の製品名及び会社名は、各社の商標または登録商標です。

1. はじめに	6
1.1 Sanei ESC/POS Android SDK をビルドする	6
1.2 対応プリンタ	8
1.3 関連ソフトウェア	8
1.4 ライブラリの実装	8
2. プリンタデバイスクラス	11
3. プリンタデバイスクラスのメソッド	12
3.1 discoverUsbPrinter メソッド	13
3.1.1 discoverBlePrinter メソッド	13
3.2 connectPrinter メソッド (For USB)	14
3.2.1 connectPrinter メソッド (For BLE)	14
3.2.2 connectPrinter メソッド (For WLAN)	14
3.3 disconnectPrinter メソッド	15
3.4 isPrinterConnected メソッド	15
3.5 isSupportedPrinterDevice メソッド (For BLE)	15
3.6 printString メソッド	16
3.7 setFontStyle メソッド	17
3.8 setFontType メソッド	17
3.9 setFontMagnification メソッド	18
3.10 setFontColor メソッド	18
3.11 setFontSmoothing メソッド	19
3.12 setAlignment メソッド	19
3.13 setCodePage メソッド	20
3.14 setInternationalChar メソッド	21
3.15 printBarcode メソッド	22
3.16 printQR メソッド	23
3.17 printPDF417 メソッド	24
3.18 setBarcodeModification メソッド	25
3.19 printBitmap メソッド	26
3.20 printBitmap メソッド	27
3.21 printBitmapRaster メソッド	28
3.22 lineFeed メソッド	29
3.23 paperFeed メソッド	30
3.24 printStringAndroidFont メソッド	31
3.25 startPageMode メソッド	32
3.26 endPageMode メソッド	32
3.27 setPageAttribute メソッド	33
3.28 getStatus メソッド	34
3.29 initPrinter メソッド	35
3.30 outputRawData メソッド	36
3.31 outputRawData メソッド	36
3.32 inputRawData メソッド	37
3.33 inputRawData メソッド	37

付録 1. プリンターステータス	38
付録 2. ページモード	42
付録 3. SDKバージョンの確認	44

1. はじめに

1.1 Sanei ESC/POS Android SDK をビルドする

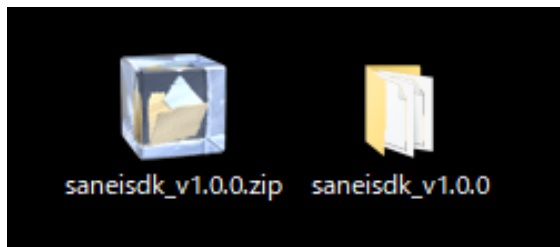
Android SDKをビルドするには、下記のサイトよりパッケージを入手する。

<https://www.sanei-elec.co.jp/support/downloads/android-sdk/>

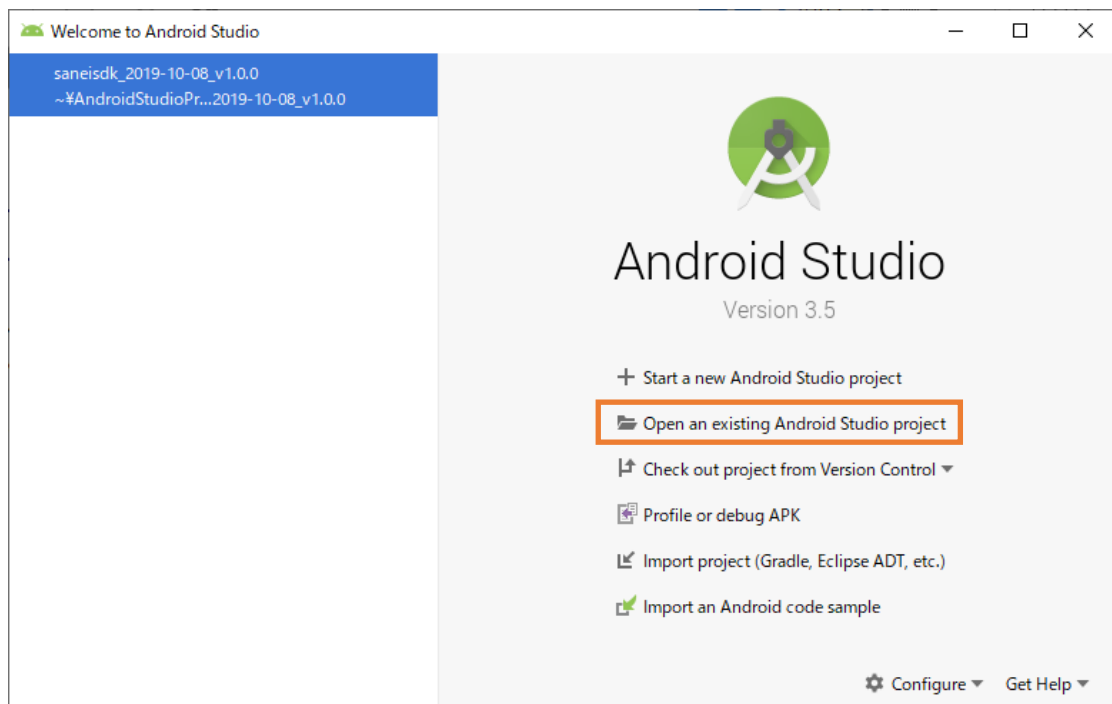
本手順に従って、お客様がAndroidアプリケーションの構築予定のAndroid StudioのプロジェクトにAndroid SDKのプロジェクトを開きます。

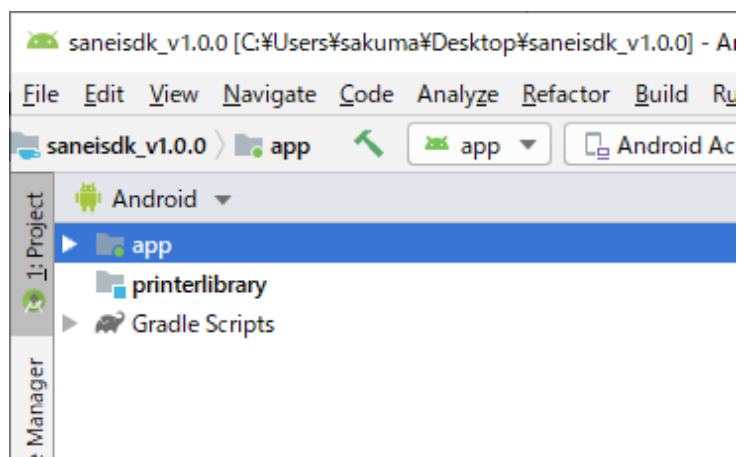
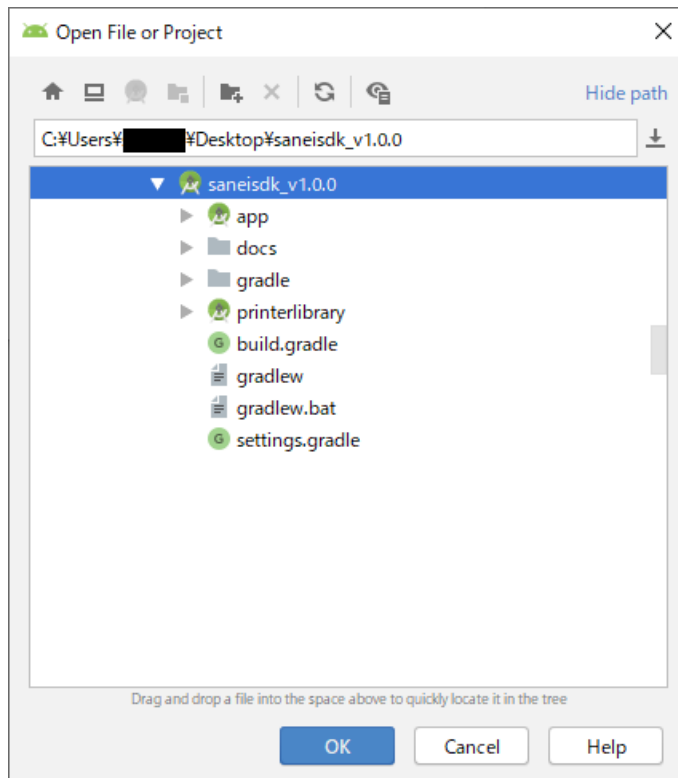
事前に開発を行うPCにAndroid Studio Giraffe以上がインストールされている事を前提とする。

1. Sanei ESC/POS SDKパッケージを解凍する。

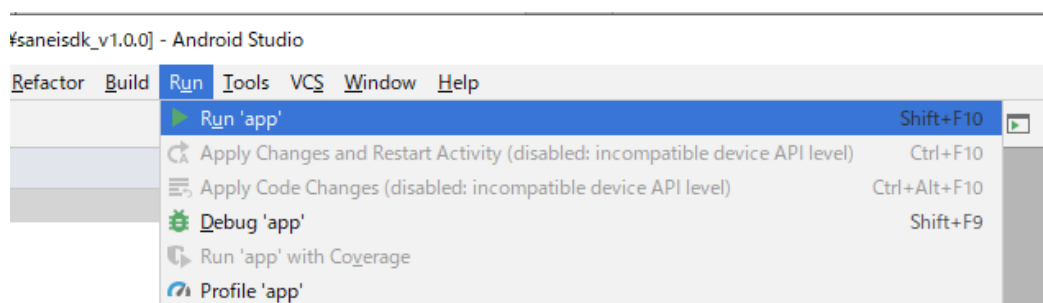


2. Android Studioを実行し、Sanei ESC/POS SDKパッケージを開きます。





3. 上部メニューバーの「Run」をクリックして「Run ‘app’」を選択する。



1.2 対応プリンタ

対応プリンタモデル		F/W バージョン	対応インタフェース	対応コマンド
KIOSK プリンタ	SK4-21	V1.02.00 以降	USB	MODE-A
	SK4-31	V1.02.00 以降	USB	MODE-A
	SK1-2x1	V2.60.00 以降	USB	MODE-A
	SK1-3x1	V2.60.00 以降	USB	MODE-A
	SK1-21H	V2.60.00 以降	USB	MODE-A
	SK1-31H	V2.60.00 以降	USB	MODE-A
	SK1-41	V3.01.00 以降	USB	MODE-A
	SK5-31	V1.00.00 以降	USB,LAN	MODE-A
Desktop プリンタ	SD3-22	V1.05.00 以降	USB	MODE-A
	SD3-21	V1.05.00 以降	USB	MODE-A
Mobile プリンタ	SM4-21	V1.01.00 以降	USB, BLE, WLAN	MODE-A
	SM4-31	V1.01.00 以降	USB, BLE, WLAN	MODE-A
Panel プリンタ	SP1-21	V1.54.00 以降	USB	MODE-A

★ メモリスイッチの設定

Sanei ESC/POS Android SDK を使用する上では、全てのプリンタモデルに対するメモリスイッチの設定を以下のとおり設定して使用する。

OFFLINE BUSY = OFF

Act. For Driver = INVALID

1.3 関連ソフトウェア

本バージョンにおいては、関連ソフトウェアはありません。

1.4 ライブラリの実装

1. Sanei ESC/POS Android SDK においてアプリケーション側の API レベルを 15 以上に設定する。

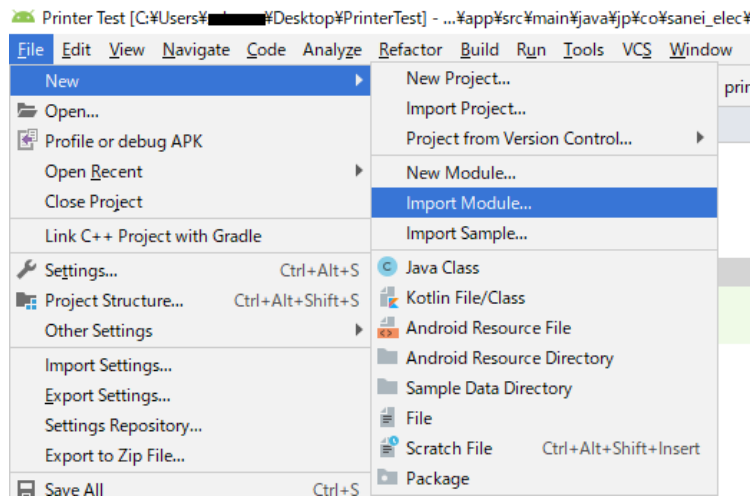
アプリケーションの build.gradle を開いて「minSdkVersion」の値を確認できます。

```

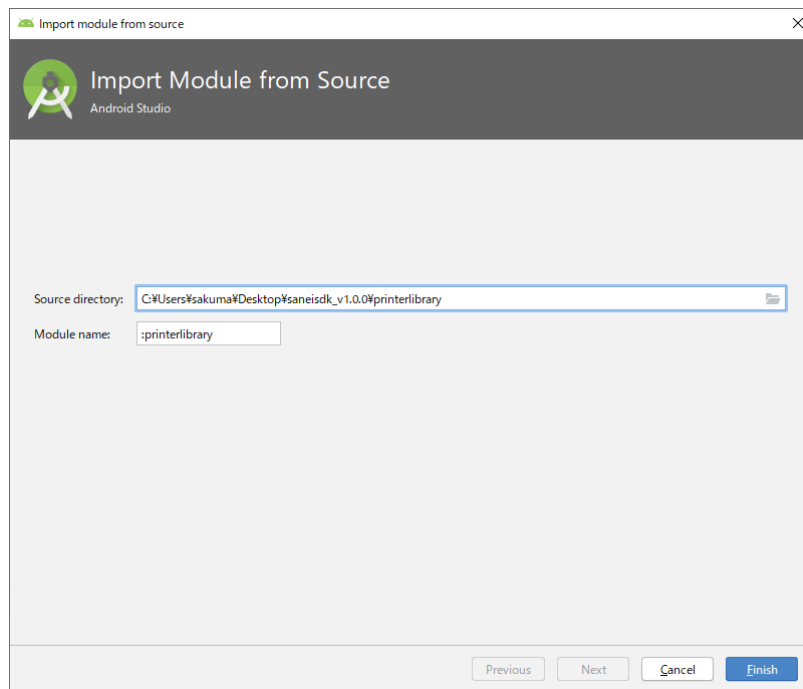
1  apply plugin: 'com.android.application'
2
3  android {
4      compileSdkVersion 29
5      defaultConfig {
6          applicationId "jp.co.sanei_elec.printertest"
7          minSdkVersion 15
8          targetSdkVersion 29
9          versionCode 1
10         versionName "1.0"
11         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
12     }
13     buildTypes {
14         release {
15             minifyEnabled false
16             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
17         }
18     }
19 }

```


2. Android Studio の File メニュー → New → Import Module を選択する。



3. Source Directory に解凍したフォルダ内の「printerlibrary」フォルダを指定する。



4. アプリケーションの build.gradle の dependencies に以下の 1 行を追加する。

「implementation project(':printerlibrary')」

```
21 dependencies {
22     implementation project(':printerlibrary')
23     implementation fileTree(dir: 'libs', include: ['*.jar'])
24     implementation 'androidx.appcompat:appcompat:1.1.0'
25     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
26     testImplementation 'junit:junit:4.12'
27     androidTestImplementation 'androidx.test:runner:1.2.0'
28     androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
29 }
```

5. 使用したいソースに以下の import 指定を追加する。

```
Import jp.co.sanei_elec.printerlibrary.api.BarcodeSystem;  
import jp.co.sanei_elec.printerlibrary.api.HRI;  
import jp.co.sanei_elec.printerlibrary.api.HorizontalAlignment;  
import jp.co.sanei_elec.printerlibrary.api.InternationalCharset;  
import jp.co.sanei_elec.printerlibrary.api.PageDirection;  
import jp.co.sanei_elec.printerlibrary.api.PrinterDevice;  
import jp.co.sanei_elec.printerlibrary.api.PrinterListener;  
import jp.co.sanei_elec.printerlibrary.api.PrinterStatus;
```



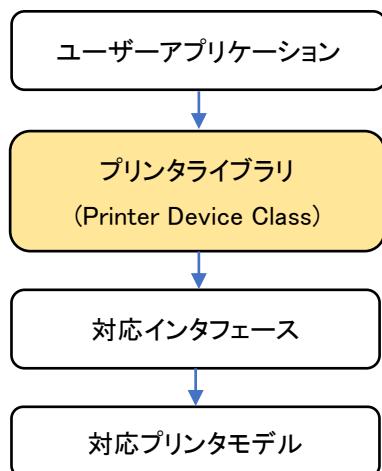
```
1 package jp.co.sanei_elec.printertest;  
2  
3 import androidx.appcompat.app.AppCompatActivity;  
4  
5 import jp.co.sanei_elec.printerlibrary.api.BarcodeSystem;  
6 import jp.co.sanei_elec.printerlibrary.api.HRI;  
7 import jp.co.sanei_elec.printerlibrary.api.HorizontalAlignment;  
8 import jp.co.sanei_elec.printerlibrary.api.InternationalCharset;  
9 import jp.co.sanei_elec.printerlibrary.api.PageDirection;  
10 import jp.co.sanei_elec.printerlibrary.api.PrinterDevice;  
11 import jp.co.sanei_elec.printerlibrary.api.PrinterListener;  
12 import jp.co.sanei_elec.printerlibrary.api.PrinterStatus;  
13  
14 import android.os.Bundle;  
15  
16 public class MainActivity extends AppCompatActivity {  
17  
18     @Override  
19     protected void onCreate(Bundle savedInstanceState) {  
20         super.onCreate(savedInstanceState);  
21         setContentView(R.layout.activity_main);  
22     }  
23 }  
24
```

6. これで SDK が使用できるようになります。

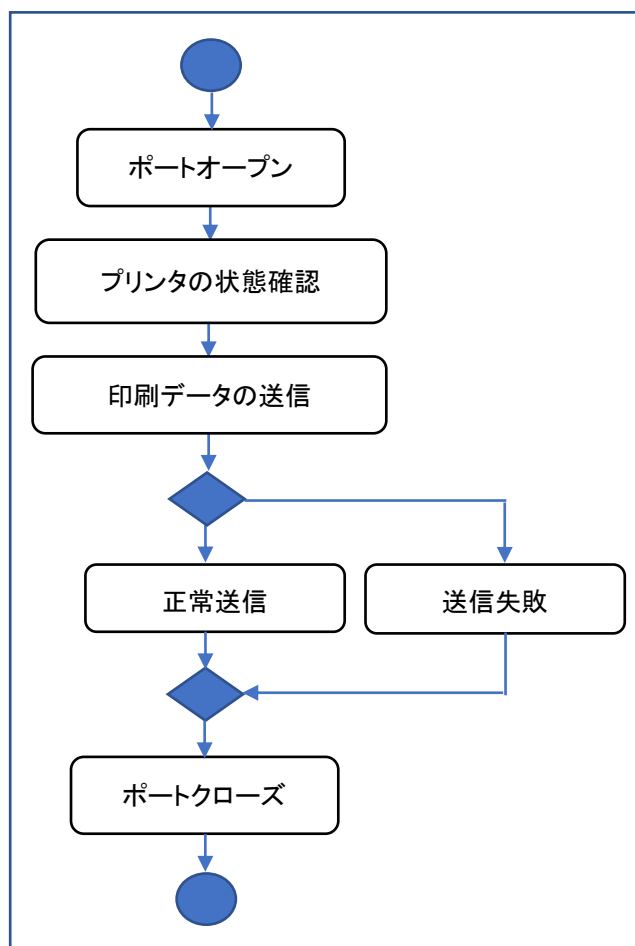
2. プリンタデバイスクラス

PrinterDeviceクラスを利用した全体構成及び印刷までのプロセスを下記に示します。

● 全体構成



● 印刷までのプロセス



サンプルコードのイメージ:

```
// ポートの接続
If (ConnectPrinter ("USB:SK1-31:00") == true)
{
    // プリンタ状態の確認
    int[] sts = GetStatus();

    If (sts[0] == Ready)
    {
        // 印字データの送信
        If (PrintString ("Hello World!\n") == false)
        {
            System.out.println("Print Error");
        }
    }
}

// ポートクローズ
DisconnectPrinter();
}
```

3. プリンタデバイスクラスのメソッド

プリンタデバイスクラスのメソッド一覧は、以下の通りです。

カテゴリー	メソッド	機能
接続・切断	discoverUsbPrinter	USBに接続されているプリンタポート名を取得する。
	discoverBlePrinter	Bluetooth(BLE)で接続されているプリンタポート名を取得する。
	connectPrinter (For USB)	指定されたプリンタポート(取得されたプリンタポート名から選択)に接続する。
	connectPrinter (For BLE)	指定されたプリンタポート(取得されたプリンタポート名から選択)に接続する。
	connectPrinter (For WLAN)	指定されたプリンタポート(取得されたIPアドレスとポート番号)に接続する。
	disconnectPrinter	指定されているプリンタポートを切断する。
	isPrinterConnected	プリンタポートに接続されているか確認する。
	isSupportedPrinterDevice	Bluetooth(BLE)で接続されているデバイスがサポートプリンタかどうか確認する。
標準印字	printString	ユニコードで文字列(漢字文字も対応可)を指定しプリンタフォントを印字する。
	setFontStyle	プリンタフォントに対する文字修飾を設定する。
	setFontType	プリンタフォントのタイプ(フォントA又はフォントB)を設定する。
	setFontMagnification	プリンタフォントのサイズ(縦倍数と横倍数)を設定する。
	setFontColor	プリンタフォントに対する白黒反転を指定または解除する。
	setFontSmoothing	縦又は横倍数指定されたプリンタフォントに対するスムージング処理を指定または解除する。
	setAlignment	印字データに対する印刷位置((左揃い、センタリング、右揃い)を指定する。
	setCodePage	プリンタフォントに対するコードページを指定する。
	setInternationalChar	プリンタフォントに対する国際キャラクタ文字を指定する。
	printBarcode	1Dのバーコードを印字する。
	printQR	QRコードを印字する。
	printPDF417	PDF417を印字する。
	setBarcodeModification	1Dバーコードの修飾情報(高さ、HRI文字)を設定する。
	printBitmap	アンドロイドのビットマップクラスをビットイメージに変換し印字する。
	printBitmap	ビットマップ配列のデータをビットイメージに変換し印字する。
	printBitmapRaster	ビットマップ配列のデータをラスターイメージに変換し印字する。
	lineFeed	指定された行数の改行を実行する。
	paperFeed	指定されたドットライン数の正方向または逆方向に紙送りを実行する。
	printStringAndroidFont	指定された文字列をAndroidフォントに変換しビットイメージで印字する。
	initPrinter	プリンタにセットされている修飾情報を初期化する。
ページ印字	startPageMode	標準モードからページモードに移行する。
	endPageMode	ページメモリの指定領域を印字しページモードから標準モードに移行する。
	setPageAttribute	ページモードの属性情報(印字方向、印字領域)を設定する。
ステータス	getStatus	プリンタの状態をプリンタステータスクラスとして取得する。
バイナリー	outputRawdata	バイト配列のバイナリーデータをプリンタポートに送信する。
	outputRawdata	1バイトのバイナリーデータをプリンタポートに送信する。
	inputRawdata	プリンタポートからバイト配列へ読み取られたバイナリーデータを取得する。
	inputRawdata	1バイトのバイナリーデータをプリンタポートから取得する。

3.1 discoverUsbPrinter メソッド

USB に接続されているプリンタポート名を取得する。

宣言:

```
public android.hardware.usb.UsbDevice[] discoverUsbPrinter(int timeoutMillis)
```

引数:

timeoutMillis 内部制御及び本メソッド内のタイムアウト時間

戻り値:

UsbDevice[] USB に接続されているプリンタポート名

備考:

使用許可されている、全ての USB 接続されているプリンタを取得する。

3.1.1 discoverBlePrinter メソッド

Bluetooth(BLE)で接続されているプリンタポート名を取得する。

宣言:

```
public android.bluetooth.BluetoothDevice[] discoverBlePrinter(int timeoutMillis)
```

引数:

timeoutMillis 内部制御及び本メソッド内のタイムアウト時間

戻り値:

BluetoothDevice [] Bluetooth(BLE)に接続されているプリンタポート名

備考:

使用許可されている、全ての Bluetooth 接続されているプリンタを取得する。

3.2 connectPrinter メソッド (For USB)

指定されたプリンタポート(取得されたプリンタポート名から選択)に接続する。

宣言:

```
public void connectPrinter(android.hardware.usb.UsbDevice device)
```

引数:

device USB 接続されている一つのプリンタポート

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.2.1 connectPrinter メソッド (For BLE)

指定されたプリンタポート(取得されたプリンタポート名から選択)に接続する。

宣言:

```
public void connectPrinter(android.bluetooth.BluetoothDevice device)
```

引数:

device Bluetooth(BLE)で接続されている一つのプリンタポート

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.2.2 connectPrinter メソッド (For WLAN)

指定されたプリンタポート(取得されたIPアドレスとポート番号)に接続する。

宣言:

```
public void connectPrinter(String host, int port);
```

引数:

host IP アドレスを入力する。(入力例 192.168.127.10 の場合、"192.168.127.10")

port ポート番号を入力する。(入力例 9100 の場合、9100)

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.3 disconnectPrinter メソッド

接続されているプリンタポートを切断する。

宣言

```
public void disconnectPrinter()
```

引数: なし

戻り値: なし

3.4 isPrinterConnected メソッド

プリンタポートに接続されているか確認する。

宣言:

```
public boolean isPrinterConnected()
```

引数: なし

戻り値: true プリンタポートに接続されている。
 false プリンタポートに接続されていない。

3.5 isSupportedPrinterDevice メソッド (For BLE)

Bluetooth (BLE) で接続されているデバイスがサポートプリンタかどうか確認する。

宣言:

```
public boolean isSupportedPrinterDevice(android.bluetooth.BluetoothDevice device)
```

引数:

device Bluetooth (BLE) で接続されている一つのプリンタポート

戻り値: true サポートプリンタ
 false サポートされていないプリンタ又はデバイス

備考:

Bluetooth デバイス名の内部モデルリストの参照からサポートプリンタを識別する。

3.6 printString メソッド

ユニコードで文字列を指定しプリンタフォントで印字する。

宣言:

```
public void printString(java.lang.String data)
public void printString(java.lang.String data, java.lang.String charsetName)
```

引数:

data ユニコードの文字列

charsetName エンコーディングキャラクタセットを文字列で指定する。

キャラクタセットはデバイスに依存し、正規名は「IANA Charset Registry」で確認する。
Java プラットフォームの実装によりキャラクタセットをサポートする必要があります。
EUC-JP の場合、“Extended_UNIX_Code_Packd_Format_for_Japanese”の名称を使用する。

指定する文字列の例
“US-ASCII”
“ISO-8859-1”
“UTF-8”
“UTF-16”
“Extended_UNIX_Code_Packd_Format_for_Japanese”

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.7 setFontStyle メソッド

プリンタフォントに対する文字修飾を設定する。

宣言：

```
public void setFontStyle(17inefee bold, 17inefee italic, 17inefee underline)
```

引数：

bold ボールド書体の指定 (true) 又は解除 (false)

italic イタリック書体の指定 (true) 又は解除 (false)

underline 下線の指定 (true) 又は解除 (false)

戻り値： なし

備考：

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
プリンタフォントの漢字文字に対しても有効です。

3.8 setFontType メソッド

プリンタフォントのタイプ (フォントA又はフォントB) を設定する。

宣言：

```
public void setFontType(17inefee compact)
```

引数：

compact フォントA (false)、フォントB (true)

戻り値： なし

備考：

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
プリンタフォントの漢字文字に対しても有効です。

3.9 setFontMagnification メソッド

プリンタフォントのサイズ(縦倍数と横倍数)を設定する。

宣言:

```
public void setFontMagnification(int horizontalRatio, int verticalRatio)
```

引数:

horizontalRatio 横倍数の1～8を指定する

verticalRatio 縦倍数の1～8を指定する

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
プリンタフォントの漢字文字に対しても有効です。

3.10 setFontColor メソッド

プリンタフォントに対する白黒反転を指定または解除する。

宣言:

```
public void setFontColor(boolean reverse)
```

引数:

reverse 白黒反転の指定(true)、解除(false)

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
プリンタフォントの漢字文字に対しても有効です。

3.11 setFontSmoothing メソッド

縦倍又は横倍数の指定されたプリンタフォントに対するスムージング処理を指定または解除する。

宣言:

```
public void setFontSmoothing(boolean on)
```

引数:

on スムージングの指定 (true)、解除 (false)

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
プリンタフォントの漢字文字に対しても有効です。

3.12 setAlignment メソッド

印字データに対する印刷位置 (左揃い、センタリング、右揃い) を指定する。

宣言:

```
public void setAlignment(HorizontalAlignment alignment)
```

引数:

alignment 印字位置 (HorizontalAlignment クラスより選択する)

HorizontalAlignment
Left : 左揃い
Center : センタリング
Right : 右揃い

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.13 setCodePage メソッド

プリンタフォントに対するコードページを指定する。

宣言:

```
public void setCodePage(CodePage 20inefeed)
```

引数:

20inefeed コードページ (CodePage クラスより選択する)

CodePage
katakana
PC1253
PC437
PC737
PC850
PC852
PC857
PC858
PC860
PC862
PC863
PC864
PC865
PC866
WPC1250
WPC1251
WPC1252
WPC1252_2
WPC1254

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.14 setInternationalChar メソッド

プリンタフォントに対する国際キャラクタ文字を指定する。

宣言:

```
public void setInternationalChar(InternationalCharset internationalCharset)
```

引数:

internationalCharset 国際キャラクタ文字 (InternationalCharset クラスより選択する)

InternationalCharset
Denmark
England
France
Germany
Italy
Japan
Spain
Sweden
USA

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.15 printBarcode メソッド

1D のバーコードを印字する。

宣言:

```
public void printBarcode(BarcodeSystem barcodeSystem, byte[] barcode)
```

```
public void printBarcode(BarcodeSystem barcodeSystem, java.lang.String barcode)
```

引数:

barcodeSystem バーコード種別 (BarcodeSystem クラスより選択する)

BarcodeSystem
CODABAR
CODE128
CODE39
ITF
JAN13
JAN8
UPCA
UPCE

barcode バーコードデータ

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

Barcode 変数の型式は、バーコードデータに制御コードの有無などで使い分ける。

3.16 printQR メソッド

QRコードを印字する。

宣言:

```
public void printQR(int size, int eccLevel, byte[] barcode)
public void printQR(int size, int eccLevel, java.lang.String barcode)
```

引数:

size シンボリサイズ(1～40)を指定する。

eccLevel エラーコントロールレベル(1～4)を指定する。

- 1: L (7%)
- 2: M (15%)
- 3: Q (25%)
- 4: H (30%)

barcode バーコードデータ

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

Barcode 変数の型式は、バーコードデータに制御コードの有無などで使い分ける。

3.17 printPDF417 メソッド

PDF417 を印字する。

宣言：

```
public void printPDF417(24inefee truncate,24inefee binaryEncode, int eccLevel, int size, byte[] barcode)
public void printPDF417(24inefee truncate,24inefee binaryEncode, int eccLevel, int size, java.lang.String barcode)
```

引数

truncate (コンパクト)PDF417 に指定する(true) 指定しない(false)

binaryEncode バイト符号化モード(true)、自動符号化モード(false)

eccLevel 誤り訂正レベル(0～7)を指定する。

Size 組み合わせ表の通り、バーコードサイズを指定する。

Size	詳細 (X:列 / Y=ステップ)	size	詳細 (X:列 / Y=ステップ)
0	X 2: Y 4	8	X 12: Y 4
1	X 2: Y 9	9	X 12: Y 9
2	X 2: Y 15	10	X 12: Y 15
3	X 2: Y 20	11	X 12: Y 20
4	X 7: Y 4	12	X 20: Y 4
5	X 7: Y 9	13	X 20: Y 9
6	X 7: Y 15	14	X 20: Y 15
7	X 7: Y 20	15	X 20: Y 20

barcode バーコードデータ

備考：

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
Barcode 変数の型式は、バーコードデータに制御コードの有無などで使い分ける。

3.18 setBarcodeModification メソッド

1D バーコードの修飾情報(高さ、HRI 文字)を設定する。

宣言:

```
public void setBarcodeModification(HRI hri, int width, int height)
```

引数:

hri HRI 文字の修飾 (HRI クラスより選択する)

HRI
Above : HRI 文字をバーコードの上に印字する。
Both : HRI 文字をバーコードの上と下に印字する。
None : HRI 文字を印字しない。
Under : HRI 文字をバーコードの下に印字する。

Width バーコードのモジュール幅(1～4)を指定する。

Height バーコードの高さ(1～255ドットピッチ)を指定する。

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.19 printBitmap メソッド

アンドロイドのビットマップクラスをビットイメージコマンドに変換して印字する。

宣言:

```
public void printBitmap(android.graphics.Bitmap bitmap)
public void printBitmap(android.graphics.Bitmap bitmap,HorizontalAlignment align,26inefee dither)
```

引数:

bitmap ビットマップデータ

alignment 印字位置(HorizontalAlignment クラスより選択する)

HorizontalAlignment	
Left	: 左揃い
Center	: センタリング
Right	: 右揃い

dither true フロイド・スタインバーグディザリングを指定する。
False ディザリングを指定しない。

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

Bitmap 指定のみの指定は、alignment = Left、dither = true で動作する。

3.20 printBitmap メソッド

int 配列の画像データをビットイメージコマンドに変換して印字する。

宣言:

```
public void printBitmap(int[] argb, int width, int height, HorizontalAlignment align, boolean dither)
```

引数:

argb int 配列の画像データ(1 ドット=32 ビットカラー)

width 画像データの横幅

height 画像データの高さ

alignment 印字位置 (HorizontalAlignment クラスより選択する)

HorizontalAlignment	
Left	: 左揃い
Center	: センタリング
Right	: 右揃い

dither true フロイド・スタインバーグディザリングを指定する。
 False ディザリングを指定しない。

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.21 printBitmapRaster メソッド

int 配列の画像データをラスタービットイメージコマンドに変換して印字する。

宣言:

```
public void printBitmapRaster(int[] argb, int width, int height, HorizontalAlignment align,  
                               28inefee dither, 28inefee crop)
```

引数:

argb int 配列の画像データ(1ドット=32ビットカラー)

width 画像データの横幅

height 画像データの高さ

alignment 印字位置(HorizontalAlignment クラスより選択する)

HorizontalAlignment	
Left	: 左揃い
Center	: センタリング
Right	: 右揃い

dither true フloyd・スタインバーグディザリングを指定する。
 False ディザリングを指定しない。

Crop true ビットイメージに変換したときに余分な余白を削除する。
 False 余白を削除しない。

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.22 lineFeed メソッド

指定された行数の改行を実行する。

宣言:

```
public void lineFeed(int lines)
```

引数:

lines 改行数(1～255)を指定する。

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

標準モードでは、ラインバッファ内のデータを印刷し、指定された行数だけ前方に紙送りする。

ページモードでは、指定した行数だけ、y 軸方向に移動する。

3.23 paperFeed メソッド

指定されたドットライン数の正方向または逆方向に紙送りを実行する。

宣言:

```
public void paperFeed(int lines)
```

引数:

lines -255～255のドットピッチ値の紙送りを指定する。
 パラメータが負数の場合、コマンドは印刷と逆方向紙送りを実行する。
 整数の場合、印刷と正方向紙送りを実行する。

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
標準モードでは、ラインバッファ内のデータを印刷し、指定されたドットピッチだけ前方に紙送りする。
ページモードでは、指定したドットピッチだけ、y 軸方向に移動する。

3.24 printStringAndroidFont メソッド

指定された文字列を Android フォントに変換しビットイメージで印字する。

宣言:

```
public void printStringAndroidFont(java.lang.String text, int x, int y, android.graphics.Paint paint)
```

引数:

text 文字列を指定する

x 水平方向の印字位置

y 垂直方向の印字位置

paint Android Paint クラス。
Android フォントの属性(サイズ、スタイル、色など)を指定する。

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.25 startPageMode メソッド

標準モードからページモードに移行する。

宣言:

```
public void startPageMode()
```

引数: なし

戻り値 なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.26 endPageMode メソッド

ページメモリの指定領域を印字しページモードから標準モードに移行する。

宣言:

```
public void endPageMode()
```

引数: なし

戻り値 なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.27 setPageAttribute メソッド

ページモードの指定領域(印字方向、印字領域)を設定する。

宣言:

```
public void setPageAttribute(int x, int y, int width, int height)
public void setPageAttribute(int x, int y, int width, int height, PageDirection direction)
```

引数:

x 印字領域(x 軸)の始点

y 印字領域(y 軸)の始点

width 印字領域(x 軸)の横幅

height 印字領域(y 軸)の高さ

direction 印字領域内の展開方向(PageDirection クラスから選択)

PageDirection
Normal : 正方向
Clockwise270 : 左90度回転
Clockwise180 : 逆方向
Clockwise90 : 右90度回転

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

印字領域はページモードを印字するまでの間、何度でも指定ができる。

ページモードを印字する場合、プリンタ内で自動的にその時点の最大印字領域で印字を行う。

印字領域の指定可能範囲は、プリンタモデルと MSW の設定に依存する。

3.28 getStatus メソッド

プリンタの状態を PrinterStatus クラスとして取得する。

宣言：

```
public PrinterStatus getStatus()  
public PrinterStatus getStatus(int timeoutMillis)
```

引数：

timeoutMillis 内部制御及び本メソッド内のタイムアウト時間

戻り値：

PrinterStatus PrinterStatus クラス

PrinterStatus	
status1	STATUS_1_ERROR STATUS_1_MOVING STATUS_1_WAITING
status2	STATUS_2_NORMAL STATUS_2_DEVICE_ERROR STATUS_2_HEAD_OPEN STATUS_2_NEAR_END STATUS_2_PAPER_EMPTY STATUS_2_LOWBATTERY STATUS_2_PAPER_IN_BEZEL STATUS_2_PULL_OUT_THE_PAPER
status3	STATUS_3_NO_ERROR STATUS_3_PAPER_JAM_ERROR STATUS_3_TEMPERATURE_ERROR STATUS_3_VOLTAGE_ERROR
status4	0 (Reserved)

備考：

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。
プリンタステータスの詳細は「付録 1. プリンタステータス」を参照とする。

3.29 initPrinter メソッド

プリンタにセットされている修飾情報を初期化する。

宣言:

```
public void initPrinter()
```

引数: なし

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.30 outputRawData メソッド

バイト配列のバイナリーデータをプリンタポートに送信する。

宣言:

```
public void outputRawData(byte[] output)
```

引数:

Output バイト配列のバイナリーデータ

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.31 outputRawData メソッド

1バイトのバイナリーデータをプリンタポートに送信する。

宣言:

```
public void outputRawData(int b)
```

引数:

b 1バイトのバイナリーデータ

戻り値: なし

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.32 inputRawData メソッド

バイト配列のバイナリーデータをプリンタポートから取得する。

宣言:

```
public int inputRawData(byte[] input, int timeoutMillis)
public int inputRawData(byte[] input, int offset, int length, int timeoutMillis)
```

引数:

input 読み込まれるバイト配列のバイナリーデータ

offset バイト配列に対するスタートオフセット

length 読み込まれたデータの書き込み最大バイト数

timeoutMillis 内部制御及び本メソッド内のタイムアウト時間

戻り値:

input 変数(バッファ)に書き込まれた総バイト数を返す。

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

3.33 inputRawData メソッド

1バイトのバイナリーデータをプリンタポートから取得する。

宣言:

```
public int inputRawData(int timeoutMillis)
```

引数:

timeoutMillis 内部制御及び本メソッド内のタイムアウト時間

戻り値:

プリンタポートから取得したバイナリーデータ

備考:

java.io.IOException 接続されているプリンタポートにエラーが発生した場合、例外を通知する。

付録1. プリンタステータス

プリンタステータスとは、プリンタの状態を4バイトに構成されて、PrinterStatus メソッド要求時に返されます。
ホストは常に本メソッドを使用して受信した全てのバイトのデータでプリンタの状態を識別する。
又、これらのプリンタステータスはプリンタモデルとその構成により対応可能な応答値が変わります。
本章にてプリンタモデルと対応可能なステータスタイプを示します。

SK4-21/31:

PrinterStatus		SK4-21 SK4-31	with Bezel (Bezel Mode=A or C)	with Bezel (Bezel Mode=B)
status1	STATUS_1_ERROR	O	O	O
	STATUS_1_MOVING	O	O	O
	STATUS_1_WAITING	O	O	O
status2	STATUS_2_NORMAL	O	O	O
	STATUS_2_DEVICE_ERROR	O	O	O
	STATUS_2_HEAD_OPEN	O	O	O
	STATUS_2_NEAR_END	O	O	O
	STATUS_2_LOWBATTERY	---	---	---
	STATUS_2_PAPER_EMPTY	O	O	O
	STATUS_2_PAPER_IN_BEZEL	---	O	---
	STATUS_2_PULL_OUT_THE_PAPER	---	---	O
status3	STATUS_3_NO_ERROR	O	O	O
	STATUS_3_PAPER_JAM_ERROR	---	---	O
	STATUS_3_TEMPERATURE_ERROR	O	O	O
	STATUS_3_VOLTAGE_ERROR	O	O	O
status4	0 (Reserved)	O	O	O

SK1-2x1/3x1 (SK1-21H/SK1-31H):

PrinterStatus		SK1-2x1 SK1-3x1	with Bezel (Bezel Mode=A or C)	with Bezel (Bezel Mode=B)	with Presenter
status1	STATUS_1_ERROR	0	0	0	0
	STATUS_1_MOVING	0	0	0	0
	STATUS_1_WAITING	0	0	0	0
status2	STATUS_2_NORMAL	0	0	0	0
	STATUS_2_DEVICE_ERROR	0	0	0	0
	STATUS_2_HEAD_OPEN	0	0	0	0
	STATUS_2_NEAR_END	0	0	0	0
	STATUS_2_PAPER_EMPTY	0	0	0	0
	STATUS_2_LOWBATTERY	--	--	--	--
	STATUS_2_PAPER_IN_BEZEL	--	0	--	--
	STATUS_2_PULL_OUT_THE_PAPER	--	--	0	0
status3	STATUS_3_NO_ERROR	0	0	0	0
	STATUS_3_PAPER_JAM_ERROR	--	--	0	0
	STATUS_3_TEMPERATURE_ERROR	0	0	0	0
	STATUS_3_VOLTAGE_ERROR	0	0	0	0
status4	0 (Reserved)	0	0	0	0

SK1-41:

PrinterStatus		SK1-41	with Presenter
status1	STATUS_1_ERROR	0	0
	STATUS_1_MOVING	0	0
	STATUS_1_WAITING	0	0
status2	STATUS_2_NORMAL	0	0
	STATUS_2_DEVICE_ERROR	0	0
	STATUS_2_HEAD_OPEN	0	0
	STATUS_2_NEAR_END	0	0
	STATUS_2_PAPER_EMPTY	0	0
	STATUS_2_LOWBATTERY	--	--
	STATUS_2_PAPER_IN_BEZEL	--	--
	STATUS_2_PULL_OUT_THE_PAPER	--	0
status3	STATUS_3_NO_ERROR	0	0
	STATUS_3_PAPER_JAM_ERROR	--	0
	STATUS_3_TEMPERATURE_ERROR	0	0
	STATUS_3_VOLTAGE_ERROR	0	0
status4	0 (Reserved)	0	0

SD3-21/22:

PrinterStatus		SD3-21	SD3-22
status1	STATUS_1_ERROR	0	0
	STATUS_1_MOVING	0	0
	STATUS_1_WAITING	0	0
status2	STATUS_2_NORMAL	0	0
	STATUS_2_DEVICE_ERROR	0	0
	STATUS_2_HEAD_OPEN	0	0
	STATUS_2_NEAR_END	0	0
	STATUS_2_PAPER_EMPTY	0	0
	STATUS_2_LOWBATTERY	--	--
	STATUS_2_PAPER_IN_BEZEL	--	--
	STATUS_2_PULL_OUT_THE_PAPER	--	--
status3	STATUS_3_NO_ERROR	0	0
	STATUS_3_PAPER_JAM_ERROR	--	--
	STATUS_3_TEMPERATURE_ERROR	0	0
	STATUS_3_VOLTAGE_ERROR	0	0
status4	0 (Reserved)	0	0

SM4-21/31:

PrinterStatus		SM4-21	SM4-31
status1	STATUS_1_ERROR	0	0
	STATUS_1_MOVING	0	0
	STATUS_1_WAITING	0	0
status2	STATUS_2_NORMAL	0	0
	STATUS_2_DEVICE_ERROR	0	0
	STATUS_2_HEAD_OPEN	—	0
	STATUS_2_NEAR_END	—	—
	STATUS_2_PAPER_EMPTY	0	0
	STATUS_2_LOWBATTERY	0	0
	STATUS_2_PAPER_IN_BEZEL	--	--
	STATUS_2_PULL_OUT_THE_PAPER	--	--
status3	STATUS_3_NO_ERROR	0	0
	STATUS_3_PAPER_JAM_ERROR	--	--
	STATUS_3_TEMPERATURE_ERROR	0	0
	STATUS_3_VOLTAGE_ERROR	0	0
status4	0 (Reserved)	0	0

SP1-21:

PrinterStatus		SP1-21
status1	STATUS_1_ERROR	O
	STATUS_1_MOVING	O
	STATUS_1_WAITING	O
status2	STATUS_2_NORMAL	O
	STATUS_2_DEVICE_ERROR	O
	STATUS_2_HEAD_OPEN	O
	STATUS_2_NEAR_END	O
	STATUS_2_PAPER_EMPTY	O
	STATUS_2_LOWBATTERY	--
	STATUS_2_PAPER_IN_BEZEL	--
	STATUS_2_PULL_OUT_THE_PAPER	--
status3	STATUS_3_NO_ERROR	--
	STATUS_3_PAPER_JAM_ERROR	--
	STATUS_3_TEMPERATURE_ERROR	--
	STATUS_3_VOLTAGE_ERROR	--
status4	0 (Reserved)	O

SK5-31:

PrinterStatus		SK5-31
status1	STATUS_1_ERROR	O
	STATUS_1_MOVING	O
	STATUS_1_WAITING	O
status2	STATUS_2_NORMAL	O
	STATUS_2_DEVICE_ERROR	O
	STATUS_2_HEAD_OPEN	O
	STATUS_2_NEAR_END	O
	STATUS_2_PAPER_EMPTY	O
	STATUS_2_LOWBATTERY	--
	STATUS_2_PAPER_IN_BEZEL	--
	STATUS_2_PULL_OUT_THE_PAPER	O
status3	STATUS_3_NO_ERROR	O
	STATUS_3_PAPER_JAM_ERROR	O
	STATUS_3_TEMPERATURE_ERROR	O
	STATUS_3_VOLTAGE_ERROR	O
status4	0 (Reserved)	O

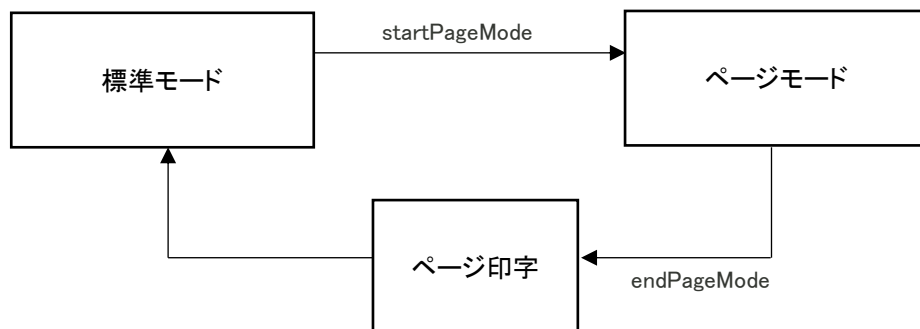
付録2. ページモード

本プリンタには、印字モードが標準モードとページモードの2種類あります。

標準モード(電源投入時は、このモードから始まる)は、印字メソッドを送信する度に印字動作を行なうモードです。

ページモードは、印字メソッドを送信しても印字動作を行わず、ページメモリの領域上に印字データを書き込みます。endPageMode メソッドの送信により、ページメモリの領域を一括して印字し標準モードに戻ります。

ページモードと標準モードの関係は、以下のようになります。



1. ページ領域について

setPageAttribute メソッドで指定する引数は、プリンタモデルと印字幅により設定可能な値が異なります。

設定可能範囲は下表のとおり示します。

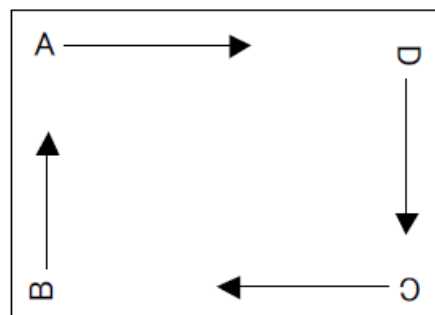
(値:ドットライン/ピッチ)

対応モデル	印字幅	x (始点)	y (始点)	width (横幅)	height (高さ)
SK4-31	72mm	0 - 574	0 - 2398	1 - 575	1 - 2399
SK4-21	54mm	0 - 430	0 - 2398	1 - 431	1 - 2399
SK1-3x1 (SK1-31H)	80mm	0 - 638	0 - 2398	1 - 639	1 - 2399
	72mm	0 - 574	0 - 2398	1 - 575	1 - 2399
SK1-2x1 (SK1-21H)	56mm	0 - 446	0 - 2398	1 - 447	1 - 2399
	54mm	0 - 430	0 - 2398	1 - 431	1 - 2399
SK1-41	104mm	0 - 830	0 - 2798	1 - 831	1 - 2799
SD3-22	54mm	0 - 430	0 - 1598	1 - 431	1 - 1599
SD3-21	48mm	0 - 382	0 - 1598	1 - 383	1 - 1599
SM4-21	48mm	0 - 382	0 - 2398	1 - 383	1 - 2399
SM4-31	72mm	0 - 574	0 - 2398	1 - 575	1 - 2399
SP1-21	48mm	0 - 382	0 - 926	1 - 383	1 - 927
SK5-31	80mm	0 - 638	0 - 2398	1 - 639	1 - 2399
	72mm	0 - 574	0 - 2398	1 - 575	1 - 2399

2. 印字方向と始点

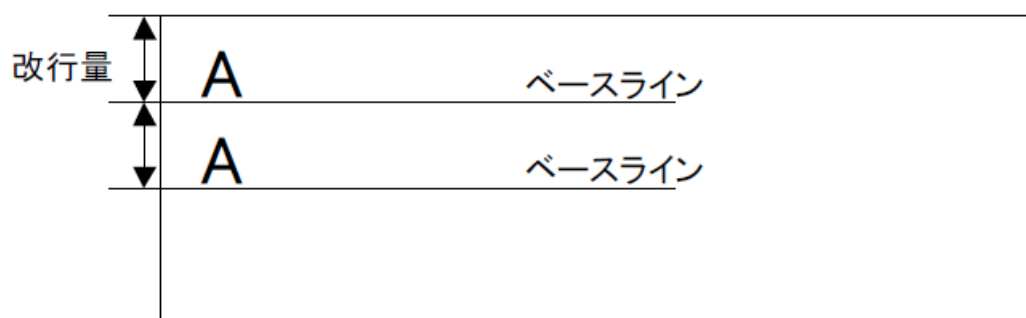
PageDirection クラスで選択される始点と印字データの展開方向を以下のとおり示します。

PageDirection	始点および展開方向
Normal :正方向	A
Clockwise270 :左90度回転	B
Clockwise180 :逆方向	C
Clockwise90 :右90度回転	D

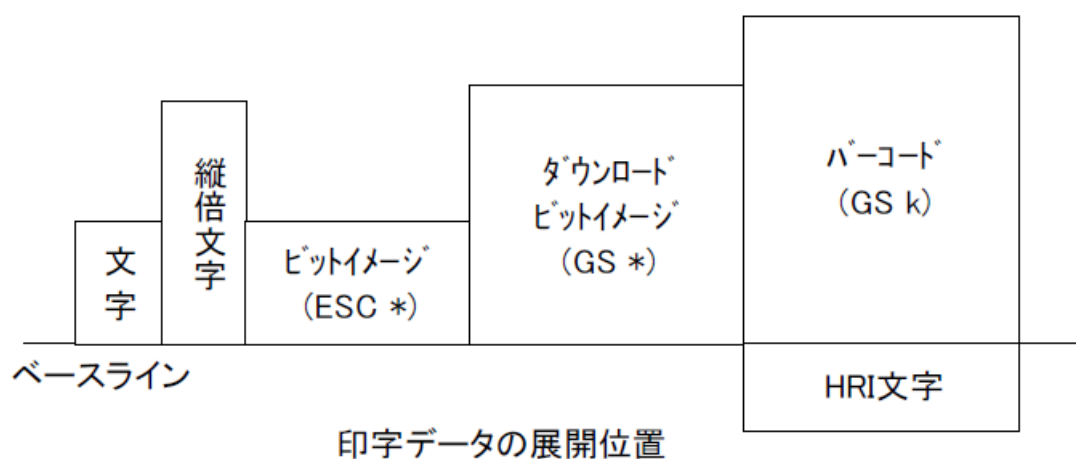


3. ページモードにおける展開

ページモードにおけるプリンタフォント、ビットイメージおよびバーコードなどの改行に伴うベースラインと印字データ展開位置を下图のとおりに示します。



文字データの展開位置



付録3. SDKバージョンの確認

SDK のバージョンをプログラム中で確認する方法を説明します。

バージョンを確認する場合、BuildConfig クラスを使用します。

```
String ver = [" + jp.co.sanei_elec.printerlibrary.BuildConfig.VERSION_NAME + "];
```

バージョン表示例:

