Android Software Development Kit

Star プリンター（StarPRNT）向け

StarIO 使用方法

本 SDK には Android デバイスのための Java Eclipse プロジェクトを含んでいます。

Android バージョン	対応インターフェイス	適用 StarIO JAR ファイル
Android OS 3.1 以上	Bluetooth, USB, Ethernet	StarIOPort3.1.jar (Default)
Android OS 2.2, 2.3, 3.0	Bluetooth, Ethernet	StarIOPort3.1.jar (Default)
Android OS 2.1	Bluetooth, Ethernet	StarIOPort.jar

必要なツール:

- JDK 6
- Eclipse
- Android SDK Manager
- ADT (Android Development Tool) Plugin for Eclipse
- USB Windows Driver by Android Device Manufacturer
- StarIO Android SDK

❖モバイルプリンター

StarPRNT エミュレーションを使用するには

StarPRNT エミュレーションを使用するには、プリンター本体のエミュレーションを“Star Line Mode”に設定する必要があります。（SM-L200 は切り替え不要です。）
エミュレーションの切り替えは以下の手順で行ってください。

◆ ESC/POS ⇔ StarPRNT エミュレーションモード 切り替え方法

1. プリンターの電源を入れ、プリンターカバーを開きます。

2. 電源ボタンと FEED ボタンを同時に長押しした後、ERROR ランプが 5 回点滅したことを確認しすぐに電源ボタンと FEED ボタンから指を放します。エミュレーションの切り替えが自動的に行われます。

3. 用紙をセット後、プリンターカバーを閉めると設定されたエミュレーションが印字されます。
   ESC/POS の場合： EMU = ESC/POS Mode
   StarPRNT の場合： EMU = Star Line Mode

   エミュレーションが正しく切り替わっていない場合、再度 1～3 の手順を行ってください。
   その際、2 の手順においては、点滅中に指を放さなさず、点滅が 5 回完了したことを確認してから指を放すように注意してください。

4. エミュレーションの切り替え後は、プリンターの電源を一度オフにしてから再投入してください。
   選択したエミュレーションは、プリンターの電源を再投入することで有効になります。

目次

- 本書に関して
- Star プリンター対応リスト
- Star プリンターを Android デバイスに接続するには
- Android デバイスの設定(実機でデバッグ)
- StarIO Android SDK を実行するためには
- Star プリンターと SDK を使用する
- Android SDK 概要
- StarIO - (StarIOPort3.1.jar)
- StarIOAndroid SDK 機能
- StarIO を使用するアプリケーション開発のために
- 追加リソース
- ASCII コード表
- SDK マニュアル改訂履歴

本書に関して

このマニュアルは、StarIO とスター モバイルプリンターが通信を行う Android アプリケーションの作成方法を解説しています。

また、このマニュアルは、Java 言語を使用するアプリケーション・システム開発者を対象に作成しており、利用者は Java 言語の基礎を理解していることを前提としています。

この SDK は Android 用に作成されています。
スター精密グローバルサポートサイトの Developers セクションには、その他のオペレーティングシステムとプログラミング言語に利用可能な SDK が用意されています。最新の SDK、テクニカルドキュメント、FAQ、及び、その他の追加情報については、Developers セクションをご確認ください。

表示マークの説明:

警告		潜在的な問題について説明します。
禁止		禁止事項について説明します。
メモ		重要な情報とヒントを提供します。

注意事項:

- Android は、Google Inc.の商標または登録商標です。
- 本マニュアルの内容は、予告無く変更する場合があります。
- スター精密株式会社は、正確な情報を提供するためにあらゆる措置を取っていますが、誤りや不作為について責任を負うものではありません。
- スター精密株式会社は、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。
- 本マニュアルの一部、あるいは全部を無断で複写・複製・転載することは、固くお断りします。

© 2012-2015 Star Micronics Co., Ltd.

Star プリンター 対応リスト

Android OS における Star プリンターのモデル別対応リストを以下に記します。

Android OS Versions 3.1 以上：USB 及び Bluetooth サポート
Android OS Versions 3.0 以下：Bluetooth サポート

◆ Android OS Version 対応リスト

Star Printer		Android OS Version			
Model	Interface	2.1	3.0 2.3 2.2	4.4 5.0 4.2 4.3 4.0 4.1 3.1 3.2	5.1 4.4 5.0 4.2 4.3 4.0 4.1
SM-S210i * (JP モデルのみ)	Bluetooth	✓	✓	✓	
SM-S220i * (欧米モデルのみ)	Bluetooth	✓	✓	✓	
SM-S230i (欧米モデルのみ)	USB			✓	
	Bluetooth	✓	✓	✓	
SM-T300i *	Bluetooth	✓	✓	✓	
SM-T400i *	Bluetooth	✓	✓	✓	
SM-L200	USB**			✓	
	Bluetooth	✓	✓	✓	
mPOP	USB				✓
	Bluetooth				✓

* ファームウェア バージョン 3.0 以降が必要です。
** ファームウェア バージョン 2.0 以降からサポートされます。

◆Line Mode / Raster Mode

Star Printer		Port Discovery	Get Firmware Information	Get Status	Sample Receipts	1D Barcodes	2D Barcodes	Text Formatting	Raster Graphics	Sample Images	MSR	Bluetooth Setting	USB Setting
Model	Interface												
SM-S210i *1 (JP モデルのみ)	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-S220i *1 (欧米モデルのみ)	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-S230i (欧米モデルのみ)	USB	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓		✓
	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-T300i *1	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-T400i *1	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-L200	USB*2	✓	✓	✓	✓*4	✓	✓*5	✓	✓	✓	✓*6		✓
	Bluetooth	✓	✓	✓	✓*4	✓	✓*5	✓	✓	✓	✓*6	✓	
mPOP	USB	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓

✓ : Line Mode のみ対応
✓ : Raster Mode のみ対応

*1 ファームウェアバージョン 3.0 以降が必要です。
*2 ファームウェアバージョン 2.0 からサポートされます。
*3 Line Mode では、Russian(ロシア語)、SimplifiedChinese(簡体字)に対応しておりません。
*4 Line Mode では、SimplifiedChinese(簡体字)はファームウェアバージョン 1.1 から対応します。
*5 PDF417 はファームウェアバージョン 1.1 から対応します。
*6 MSR はファームウェアバージョン 2.0 から対応します。

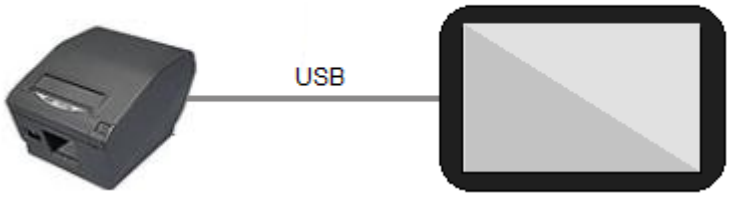
Note: この SDK では一般的な機能について書かれており、全ての機能を搭載しているわけではありません。
アプリケーションに含まれていないコマンドについては、StarPRNT コマンド仕様書をご参照ください。

Star プリンターを Android デバイスに接続する

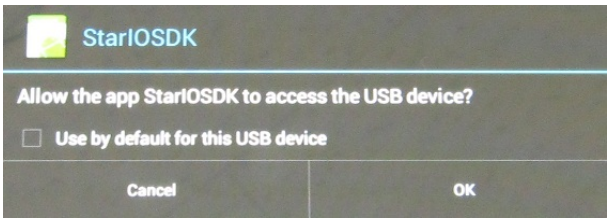
以下は Android 4.0 を搭載したタブレットでの設定方法です。
デバイスの種類や OS のバージョンによって、表示が異なる場合があります。

USB インターフェイス

- USB をご使用する場合は、Android OS Version 3.1 以上を使用してください。 (Versions 2.3 以下では、USB 機器との接続ができません。)
- Android を使用しているデバイスに、プリンタードライバ等のソフトウェアがインストールされることはありません。
- 接続に使用する USB ケーブルはタブレットによって異なります。 ほとんどの Android タブレットは、共通の USB ケーブルをサポートしていません。 ミニ USB ケーブルもしくはアダプタが必要になります。正しいケーブルを使用する為に、タブレットの仕様をご確認ください。
- USB ハブを用いて複数デバイスの接続を行う場合、カスケード接続は 1 段までを推奨します。また、同時に接続できるデバイス数は 4 デバイスを上限とします。



◆USB プリンターと Android デバイスの接続



Star USB プリンターと接続して一番最初に通信を行う時に、Android デバイスによっては上記メッセージが表示されます。

また、以下のコードを AndroidManifest .xml 及び device_filter.xml に追記することで、Android デバイスと Star USB プリンターを接続した一回目のみ許可を求めるダイアログが表示されます。

[AndroidManifest .xml]

```
<intent-filter>
   <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />
</intent-filter>
<meta-data android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"
android:resource="@xml/device_filter" />
```

[device_filter.xml]

```
<?xml version="1.0" encoding="utf-8"?>

<resources>
   <usb-device class="255" subclass="66" protocol="1" />
   <usb-device vendor-id="1305" product-id="0065" /> SM-L200  - mobile printer
   <usb-device vendor-id="1305" product-id="0067" /> SM-S230i  – mobile printer
</resources>
```

Note 1: device_filter.xml に記述された Star USB プリンターについては、接続許可時にチェックボックスにチェックをいれることでケーブルを挿抜しても、アプリケーションを再インストールしない限り許可ダイアログが表示されなくなります。また、挿抜時サンプルアプリケーションが自動で起動するようになります。

Bluetooth

Star デバイスは、工場出荷時の初期設定では各機種ごとに"Star Micronics"等、共通の Bluetooth デバイス名が設定されています。同じ Bluetooth デバイス名の機種を複数台配置して運用される場合、Bluetooth デバイス名の変更を行うと Star モバイルデバイスの判別が付けやすく便利です。

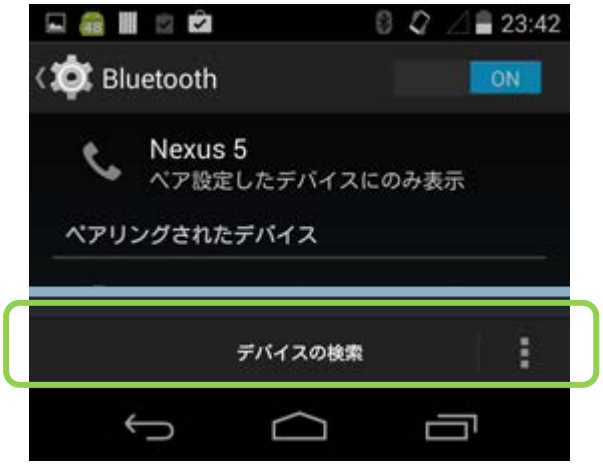
Bluetooth デバイス名の変更等、Star デバイスの LAN/Bluetooth 設定値は、スター精密の提供する Star Setting Utility を使用して変更することができます。Star Setting Utility は Google play よりダウンロードしてください。

◆ペアリング

1. 設定を行う Star モバイルデバイスの Bluetooth が有効になっていることを確認して、デバイスの電源を投入します。
2. Android デバイスにて「設定」をタップします。
3. 「Bluetooth」を ON に設定します。



4. 「デバイスの検索」をタップします。



Bluetooth デバイスのスキャンが行われ、見つかったデバイスが表示されます。
設定を行う Star モバイルデバイスのデバイス名をタップします。

5. PIN Code を使用する場合、PIN 番号（パスキー）を入力します。工場出荷時の初期値は “1234”です。(モバイルプリンターのみ)

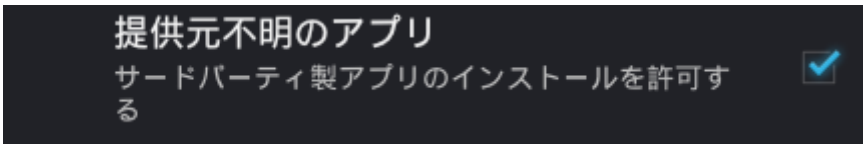


Android デバイスの設定(実機でデバッグ)

1. 「設定」をタップします。
2. 「セキュリティ」をタップします。



3. 「提供元不明のアプリ」のチェックを入れます。



4. 「開発者向けオプション」をタップします。



5. 「USB デバッグ」のチェックを入れます。



StarIO Android SDK を実行するためには

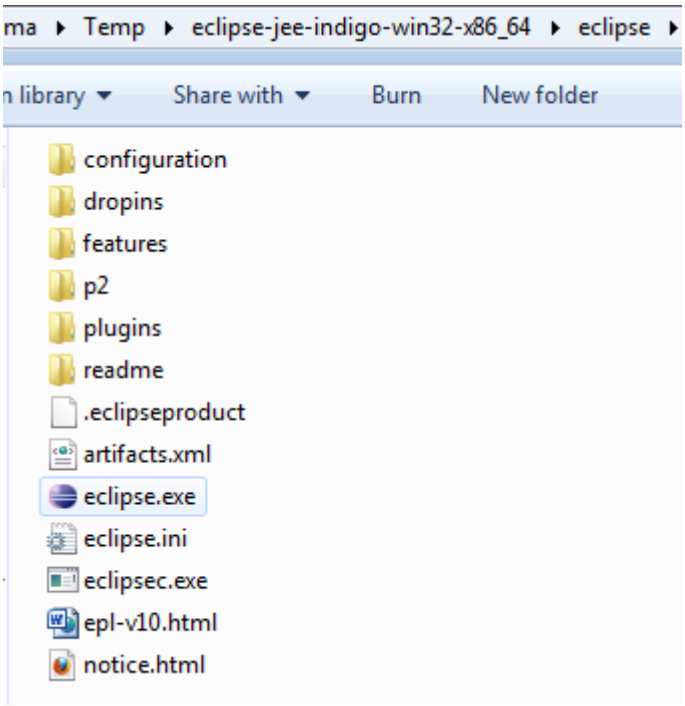
Android アプリケーションを開発するためには、Eclipse がインストールされている必要があります。Android アプリケーション開発環境の設定については、本 SDK の“documentation”フォルダー内の“AndroidDevelopmentSetup_Jp.pdf”をご参照ください。

このセクションでは、以下の項目について説明します。

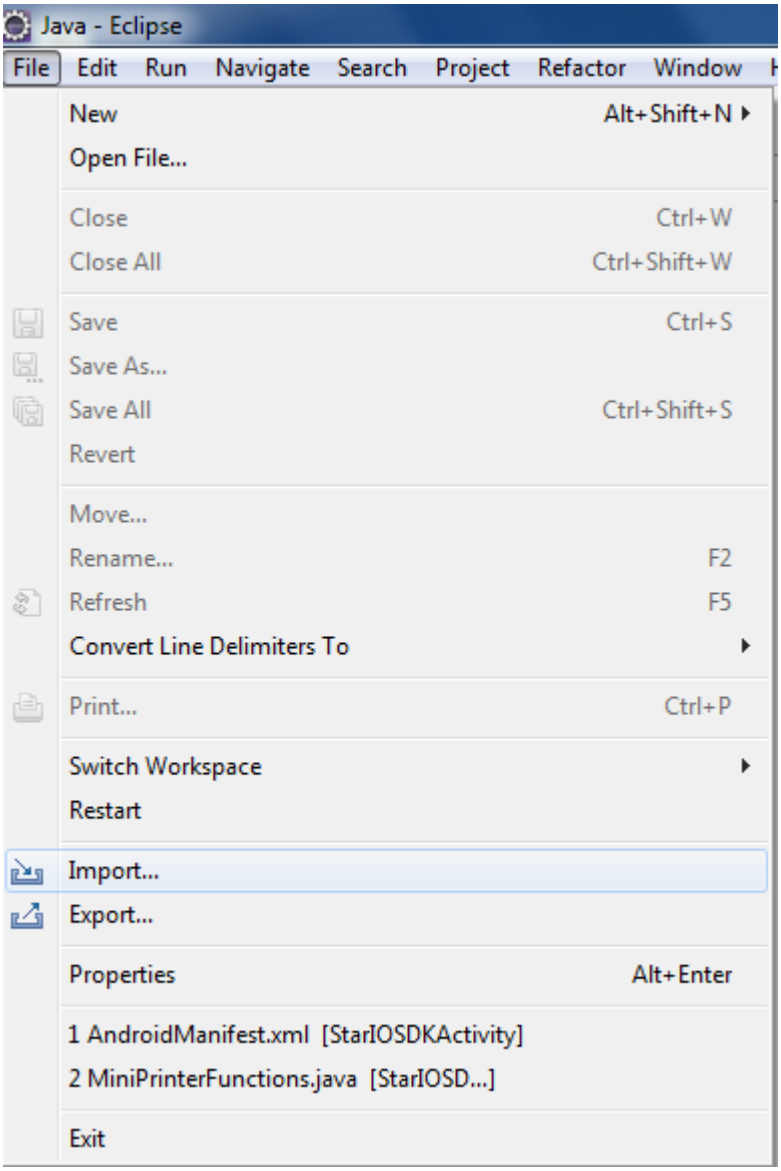
1. Android SDK プロジェクトの Eclipse への追加
2. Eclipse のデバッグモード設定
3. プロジェクトの実行

Android SDK プロジェクトの Eclipse への追加

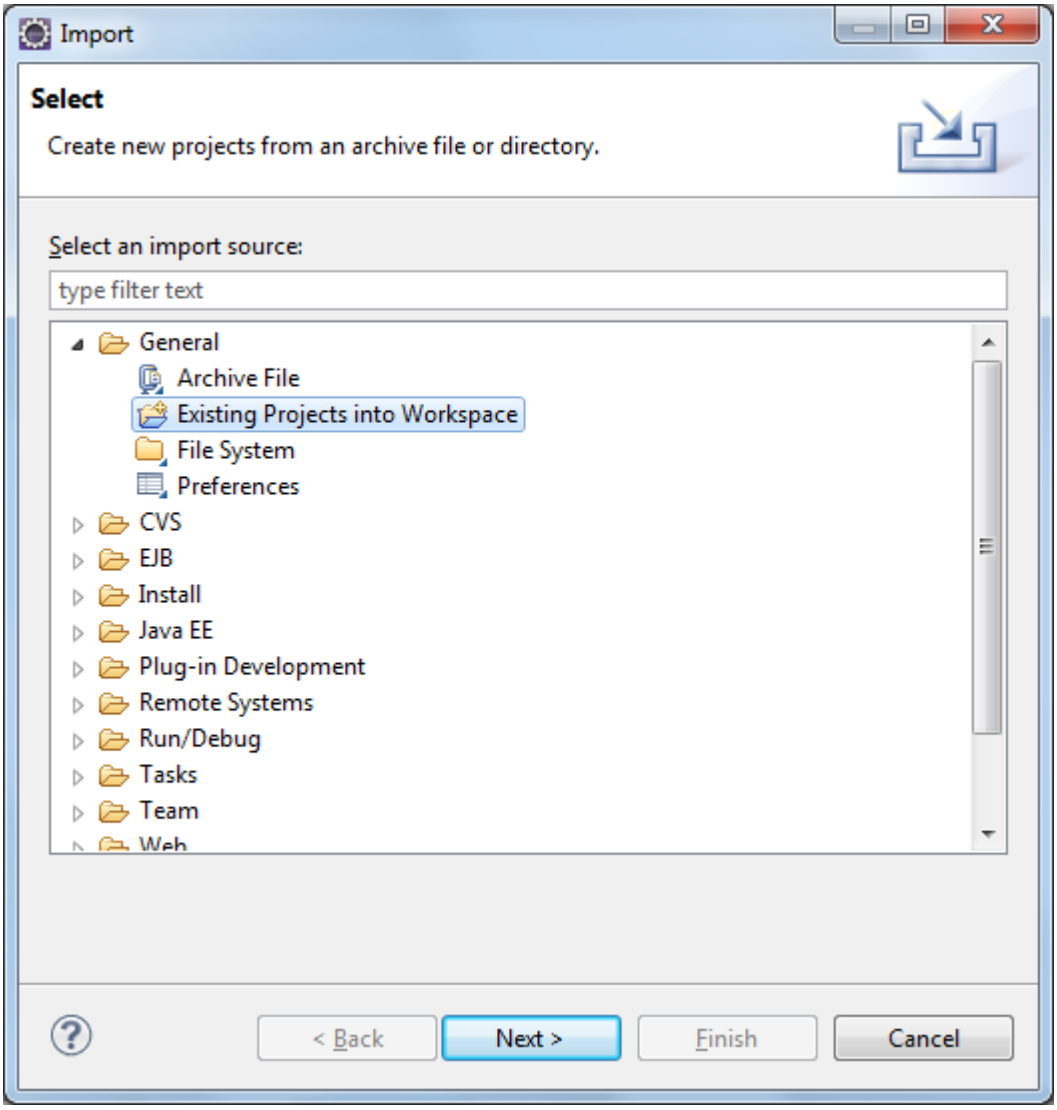
1. Eclipse を起動します。



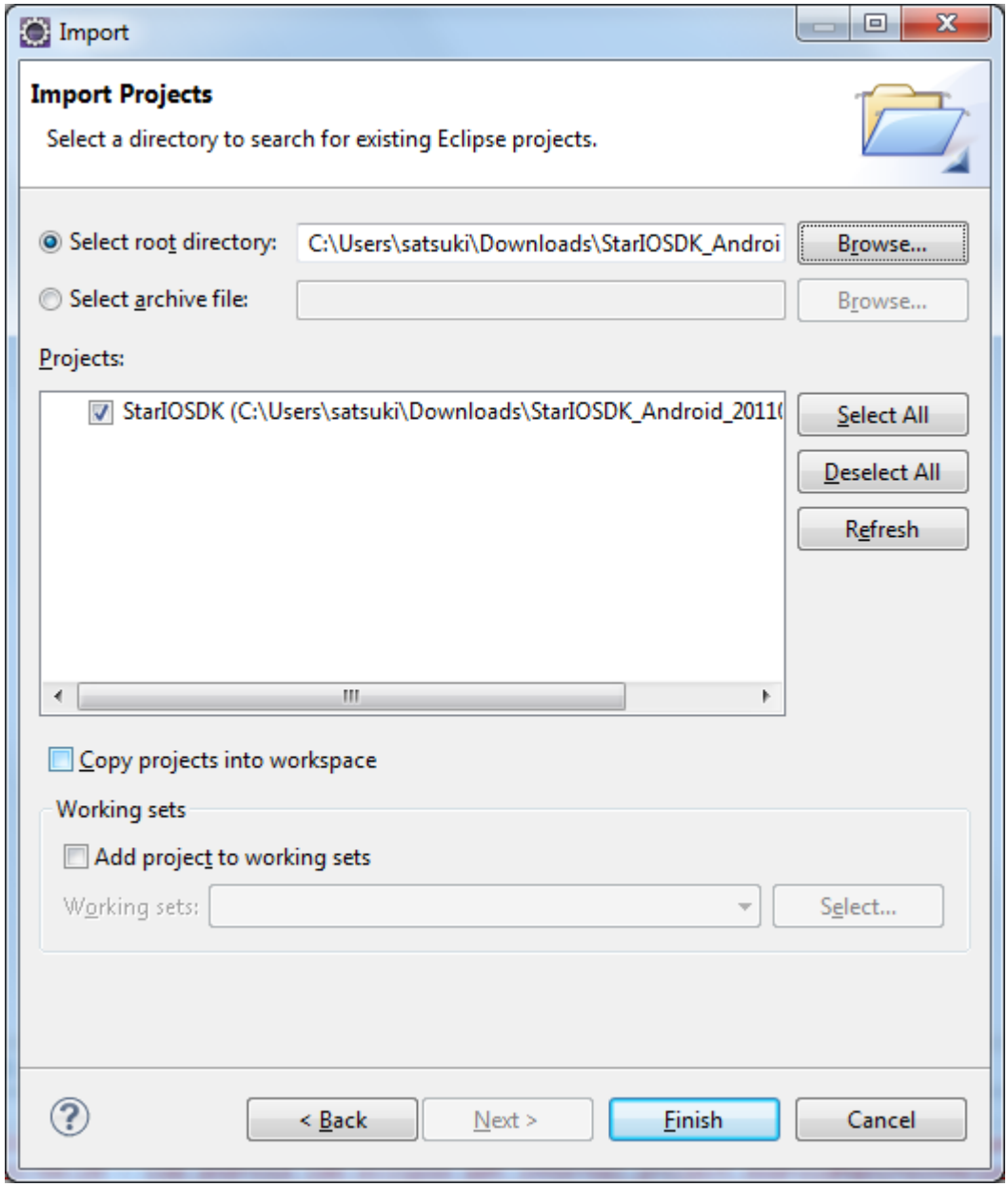
2. File → Import をクリックします。



3. General → Existing Projects into Workspace を選択し、「Next」をクリックします。

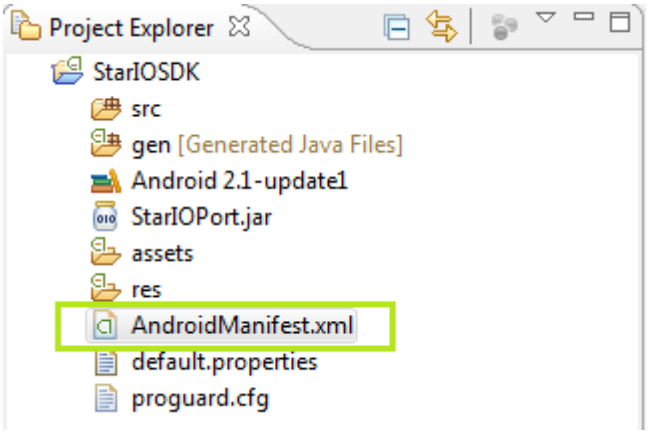


4. Star Android SDK を解凍したフォルダーを参照し、「Finish」をクリックします。

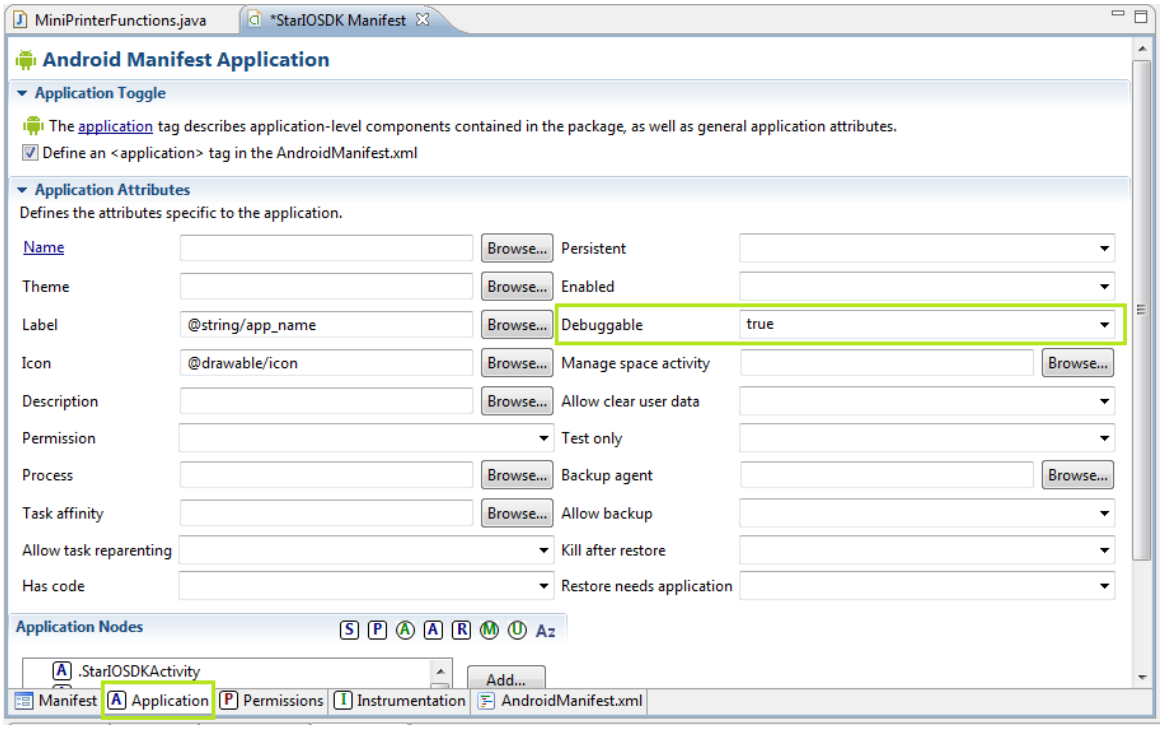


Eclipse のデバッグモード設定

1. “AndroidManifest.xml”をクリックします。

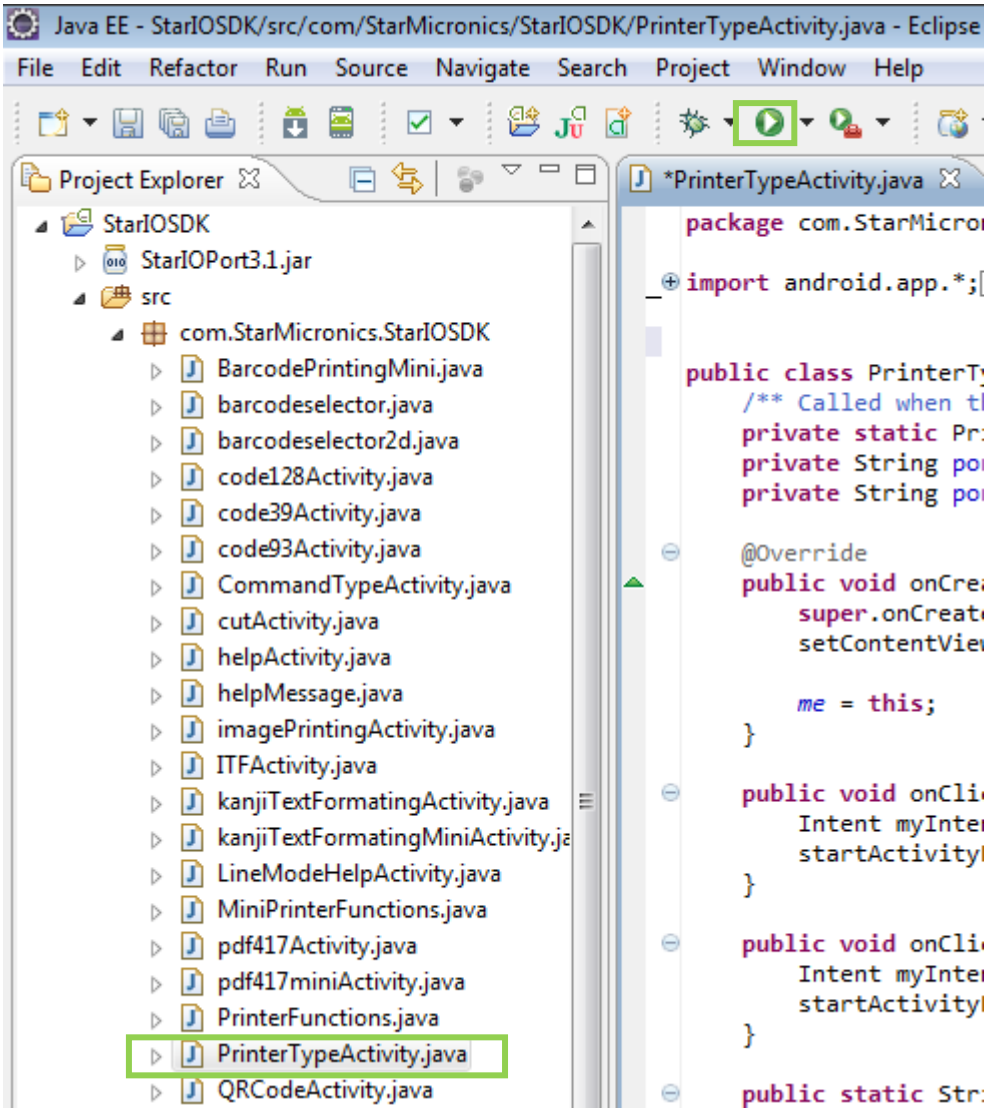


2. “Click the Application”タブをクリックします。
   “Debuggable”に true を設定します。

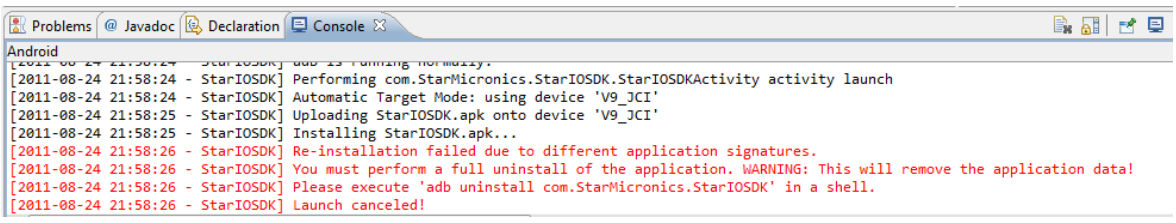


プロジェクトの実行

1. PrinterTypeActivity.java を選択して、  をクリックします。



※Android デバイス上に別 PC の“apk”を再インストールすると、以下のエラーが発生します。
このエラーを回避するには、Android デバイス上のアプリケーションを一旦削除してから、再度実行してください。



Star プリンターと SDK を使用する

Star プリンター対応リストをご確認ください。

ポート名とインターフェイスの関係:

StarIO は、プリンターと通信するために特定のポート名を使用します。
ポート名は、以下の通り正しく設定しないとプリンターとの通信を行えません。また、用途に合わせて PortSettings 等の設定も必要です。詳しくはこちらをご参照ください。

Interface	Port Name
USB	USB: USB:シリアルナンバー
Bluetooth	BT: BT:デバイス名 BT:MAC アドレス

◆既知の問題（モバイルプリンターのみ）

PIN コード設定の Bluetooth インターフェイスを使用する場合に、端末によってはペアリング後、getPort()メソッドを実行すると PinCode 入力が要求され、PinCode を入力しても getPort メソッドが失敗して印刷またはプリンターステータスの取得が行えない現象が確認されています。

上記の現象が発生した場合は、以下の【回避策】をお試しください。

※上記現象が確認されているデバイス

- Arrows Tab Wi-Fi FAR75A(Android OS バージョン 4.0.3)

【回避策】

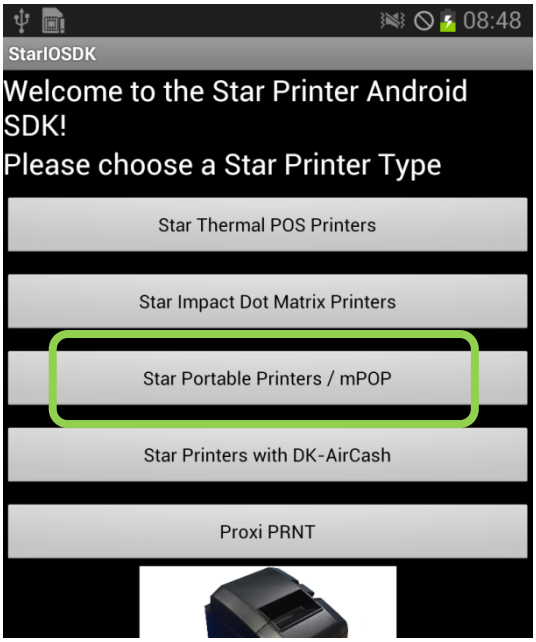
StarIOPort クラスの portSetting プロパティに空文字を指定してください。

現行： portSettings = “portable;p”;

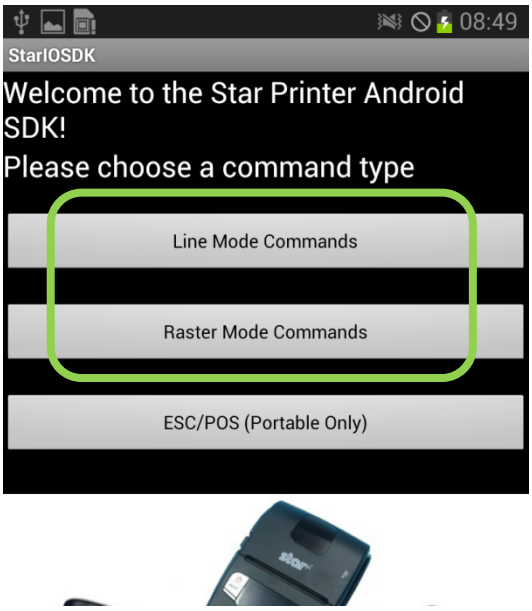
変更後： portSettings = “portable”;

スタープリンターを使用する

1. “Star Portable Printers/mPOP”をタップします。



2. 使用したいコマンドの種類をタップしてください。コマンドの種類の選択によってプリンターへのデータの送り方が変わります。



◆Line Mode Commands

プリンターへのデータ送信を 1 行ごとに行います。データは小さな単位でプリンターに送信されるため、開発者が任意の場所のコマンドでカスタマイズを行うのに適しています。

このモードではプリンターにインストールされているデバイスフォントのみ利用できます。

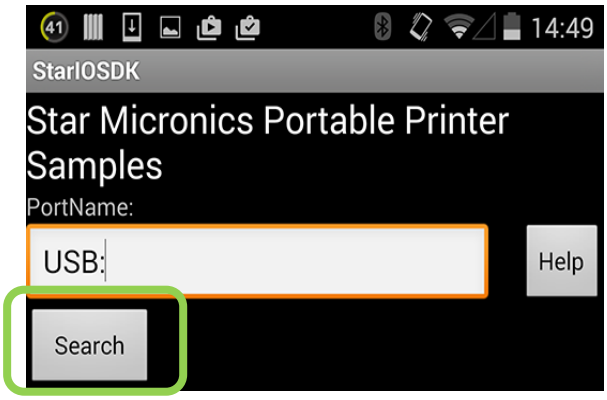
◆Raster Mode Commands

プリンターはすべての情報をグラフィックイメージで受信します。それにより、見栄えの良い TrueType フォントを利用して、高い印字速度での印刷を実現できます。

Raster コマンドでは、印字データをプリンターに送信する前にレシート全体のグラフィックデータを生成する必要があり、Line Mode コマンドよりも複雑なコーディングが必要になります。

USB プリンター

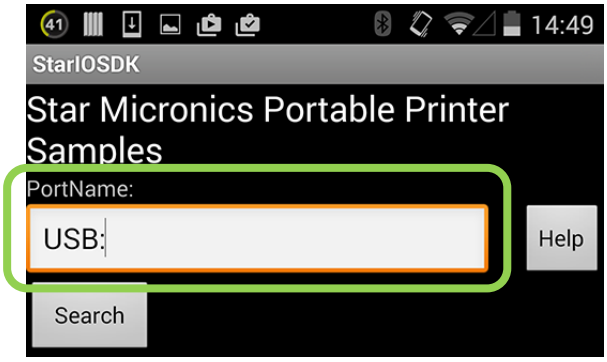
1. USB 接続されている全ての Star の USB プリンターを検索するために”Search”をタップします。



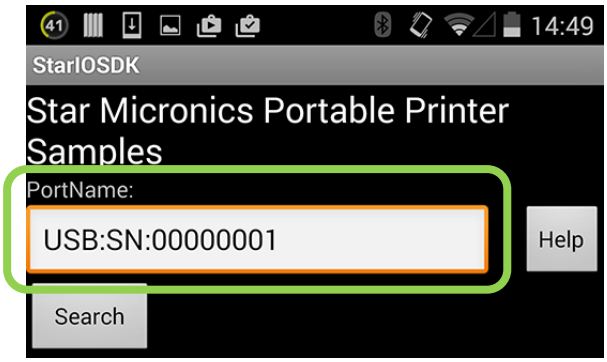
USB プリンターを接続し直した際は、再度“Search”をタップしてください。

また、以下の手順にて手動でも設定することが出来ます。

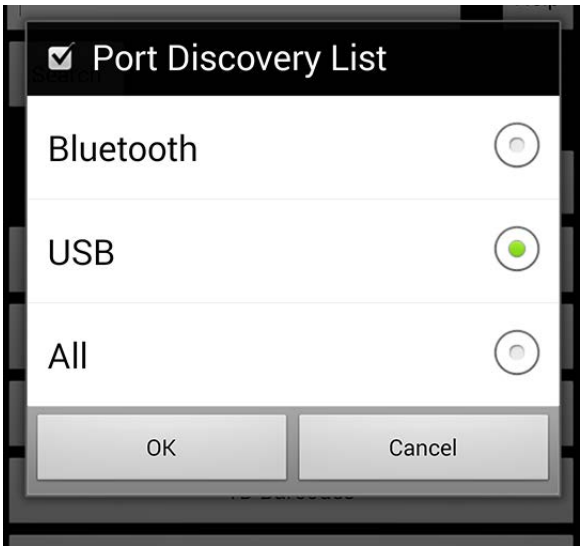
1） USB:
Star USB プリンターが一台のみ接続されている場合、パラメータ無しの“USB:”を指定することで、接続されているプリンターと自動的に通信します。



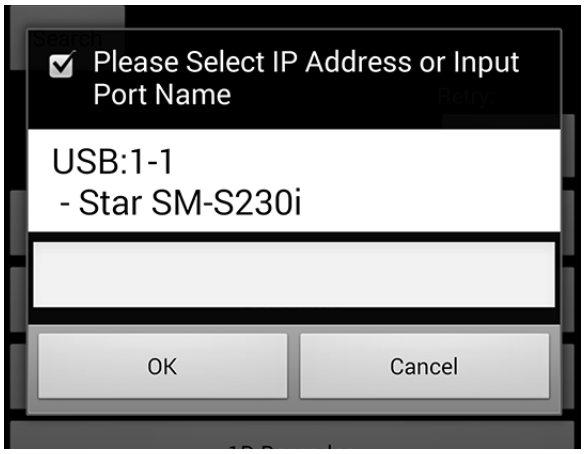
2） USB:SN: <シリアルナンバー>
“USB:SN:”の後にシリアルナンバーを追加します。（括弧は不要です）



2. USB もしくは All をタップしてください。

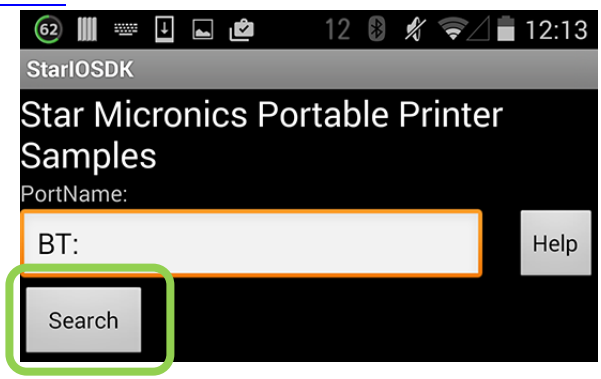


3. 接続したいプリンターの名前をタップしてください。プリンターのシリアルナンバーとモデル名も同時に表示されます。（USB インターフェースカードの種類によってはモデル名が取得できないものもあります）



Bluetooth プリンター – ポート検索

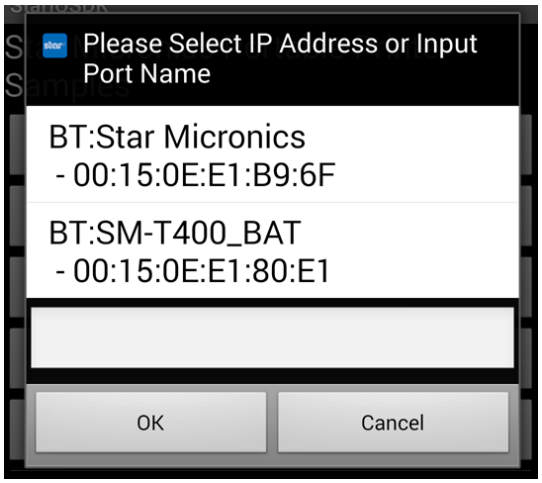
1. ペアリングされている全ての Star のプリンターを検索するために、“Search”をタップしてください。また、ポート名は”PortName”に手動で入力することも出来ます。詳細はこちらを参照してください。



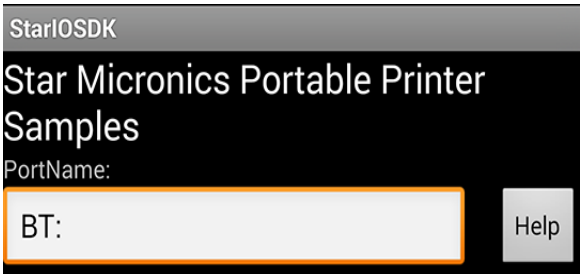
2. Bluetooth もしくは All をタップしてください。



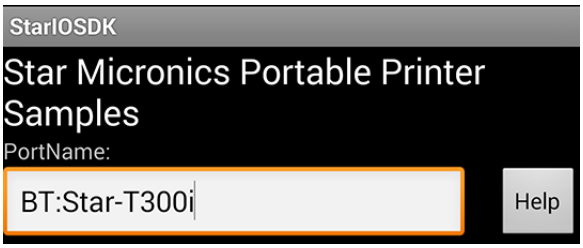
3. 接続したいプリンターの名前を選択してください。



Bluetooth プリンター – 手動ポート入力



Star Bluetooth プリンターを一台のみで使用する場合、パラメータの入力は必要ありません。



複数の Star Bluetooth プリンターを使用する場合、パラメータを入力します。

Star の Bluetooth プリンターを使用するためのポート設定には、3 通りの方法があります。

1) BT:

Star Bluetooth プリンターが一台のみ接続されている場合、パラメータ無しの“BT:”指定で、自動的に接続されているプリンターと通信します。

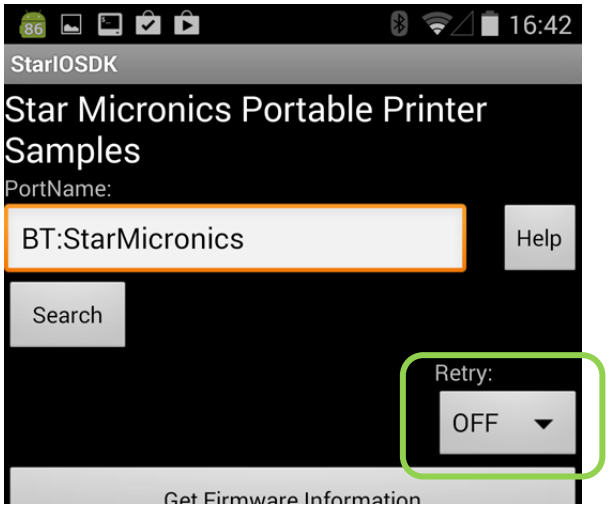
2) BT:<デバイス名>

“BT:”の後にプリンターのデバイス名を追加します。(括弧は不要です)

3) BT:<MAC アドレス>

“BT:”の後にプリンターの MAC アドレスを追加します。(括弧は不要です)
MAC アドレスは、プリンターの自己印字で確認できます。自己印字を行うには、プリンターの電源をオフにします。“Feed”ボタンを押したまま、電源をオンにします。プリンターの電源がオンになったら、“Feed”ボタンを放します。

Bluetooth デバイス - リトライ設定



この設定によって StarIO が Connect のリトライ設定を切り替えます。
リトライ時間は getPort メソッドで指定する timeout 時間が設定されます。

ON の場合リトライを行います。
　getPort メソッドの portSetting のパラメータには、”portable;l”が設定されます。

OFF の場合リトライは行いません。
　getPort メソッドの portSetting のパラメータには、”portable;”が設定されます。

Android エミュレーター(AVD)を使用する

AVD(Android Virtual Device)を使用して、SDK の動作を確認できます。
ただし、Star Bluetooth/USB プリンターとの通信は行えません。

Android SDK 概要

この概要では、SDK の主要コンポーネントについて簡単に説明します。

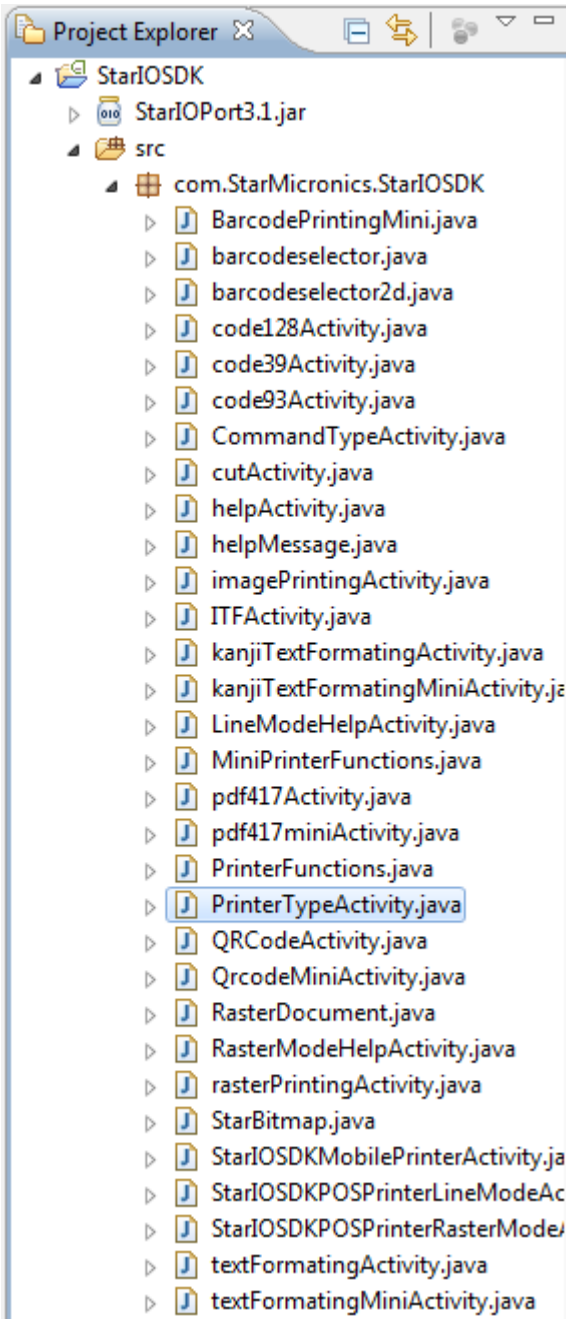
全ての機能が src の com.StarMicronics.StarIOSDK パッケージに位置しています。

PrinterTypeActivity.java を選択しプログラムを実行してください。このソースコードが POS プリンターおよびプリンターの両方の起点となります。

他のソース・ファイルをクリックすることにより、特定の機能がどのように動作するか確かめてください。例えば、“code128Activity.java”は GUI の中の 1D barcode Code128 に相当します。

全ての機能が両方のプリンター・タイプに利用可能ではありませんので、ご注意ください。各 SDK マニュアルの最初のページには、どの機能がサポートされているか記載されています。

また、“Mini”を含むソース・ファイルは、モバイルプリンターを ESC/POS モードで使用する場合のサンプルコードです。StarBitmap.java は、両方のプリンター・タイプに適用されます。



StarIO – (StarIOPort.jar)

StarIOPort.jar もしくは StarIOPort3.1.jar ファイルをプロジェクトに追加することで、StarIO メソッドを利用することができます。

StarIO Library の選択

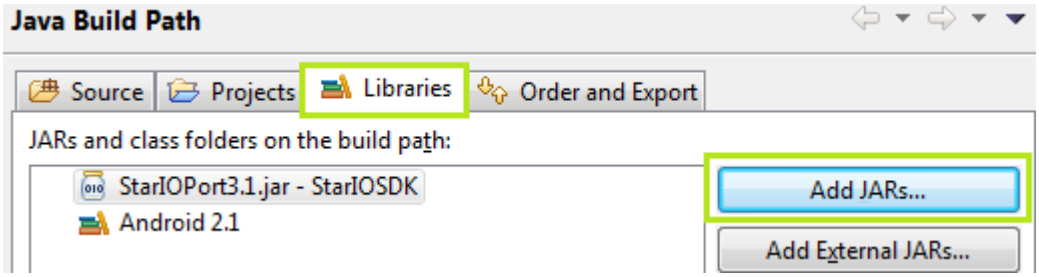
StarIO SDK には、２つのライブラリファイルが含まれています。アプリケーションが開発されている Android OS のバージョンに合わせて、いずれかを使用します。

Android OS Version	Star Interfaces	StarIO JAR File
Android OS 3.1 and Higher	Bluetooth, USB	StarIOPort3.1.jar (Default)
Android OS 2.2, 2.3, 3.0	Bluetooth	StarIOPort3.1.jar (Default)
Android OS 2.1	Bluetooth	StarIOPort.jar

プロジェクトへの StarIO ライブラリの追加

プロジェクトへの StarIO ライブラリの追加は以下の手順で行います。

1. SDK のパッケージから、Project Explorer に StarIO ライブラリをドラッグ＆ドロップします。
2. プロジェクト・フォルダーを右クリックして、プロパティを開きます。
3. Java Build Path パスをクリックします。
4. 「Libraries」タブをクリックし、“Add JARs”ボタンを押下します。
   StarIO Library を選択し、“OK”ボタンを押下します。



5. main code の先頭に以下を追加してください。

```
import com.starmicronics.stario.StarIOPort;
import com.starmicronics.stario.StarIOPortException;
import com.starmicronics.stario.StarPrinterStatus;
```

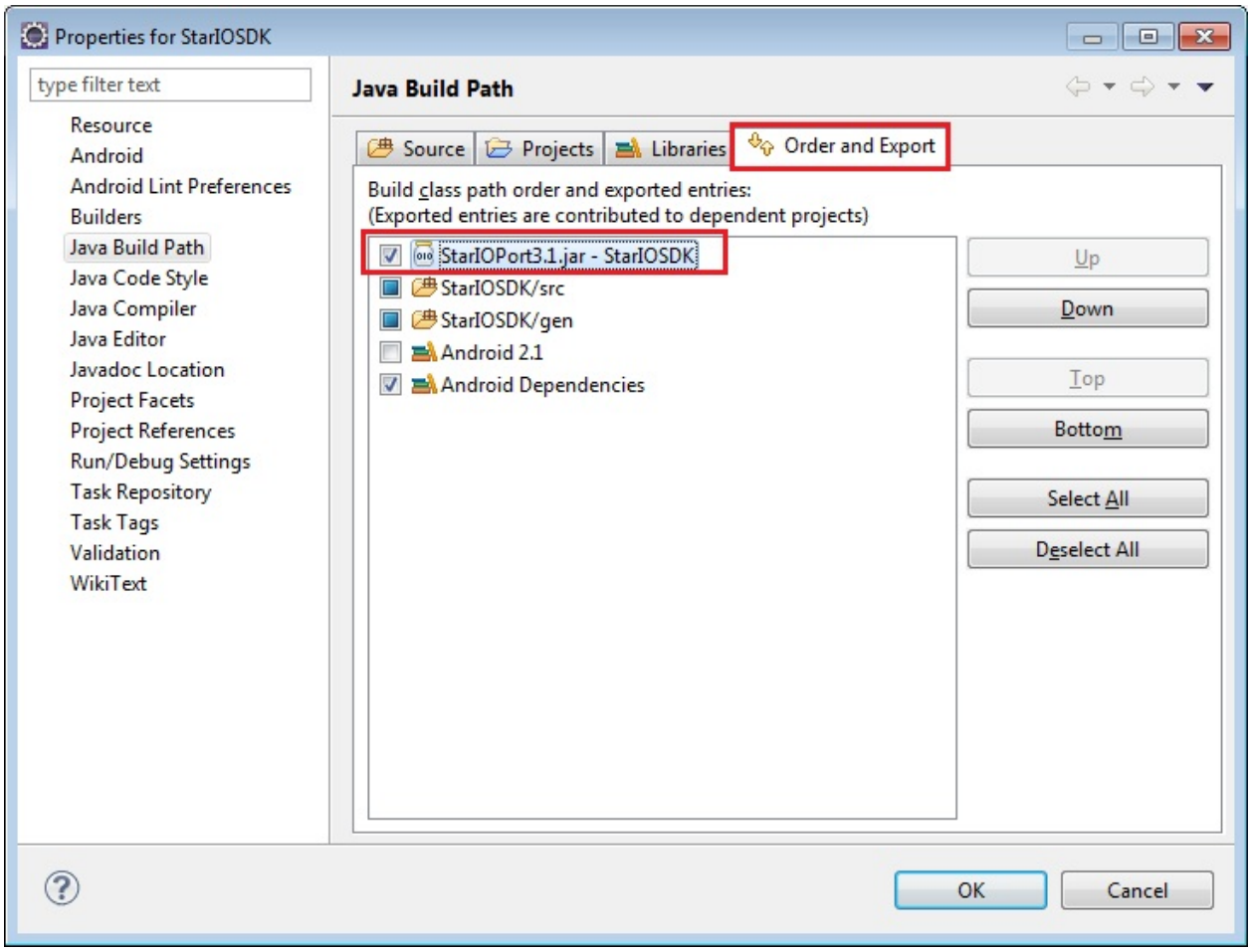
以上で、全ての StarIO メソッドを利用可能となります。

◆Android SDK Tools r17 以上でライブラリの追加をする際の注意

Android SDK Tools r17 から、ライブラリの管理方法が変更となりました。

必ず「StarIOSDK/src」の上に「StarIOPort3.1.jar」を配置する必要があります。
このメソッドは、r16 以前では問題がありませんでしたが、r17 以上を使用する場合にはエラーが発生する原因となります。

ご使用の環境により「StarIOPort.jar」を利用する場合にも、同様の注意が必要です。



1. プロジェクト名を右クリックして「Properties」を開きます。
2. 「Java Build Path」をクリックして「Order and Export」タブを開きます。
3. 「StarIOPort JAR ファイル」を選択してアプリケーションコードの上（最上位）に移動します。

◆StarIOPort ライブラリの設定

Android OS V2.2 以降：

Include the default library: StarIOPort3.1.jar

上記の StarIO ライブラリを選択することで、StarIO SDK アプリケーションが動作します。
ソースコードの変更は不要です。

<対応インターフェイス>

Android OS 3.1 以上：	Bluetooth, USB
Android OS 2.2, 2.3, 3.0：	Bluetooth

Android OS V2.1 以前：

Include the legacy library: StarIOPort.jar

上記の StarIO ライブラリを選択することで、StarIO SDK アプリケーションが動作します。
ソースコードの変更が必要です。

PrinterFunctions.java

```
StarIOPort port = null;
try
{
    /*
        using StarIOPort3.1.jar (support USB Port)
        Android OS Version: upper 2.2
    */
    port = StarIOPort.getPort(portName, portSettings, 10000, context);
    /*
        using StarIOPort.jar
        Android OS Version: under 2.1
        port = StarIOPort.getPort(portName, portSettings, 10000);
    */
```

1. ソースコード→コメント：
   port = StarIOPort.getPort(portName, portSettings, 10000, context);
2. コメント→ソースコード：
   port = StarIOPort.getPort(portName, portSettings, 10000);

この変更により、従来の StarIOPort.jar のライブラリが使用できるようになります。

StarIO メソッド概要

StarIO Port クラス

StarIO Port クラスは、portName (String)、portSettings (String)、timeout (int)の 3 つのプロパティを持ちます。



各変数は、getPort(portName,portSettings,timeout);で使用します。getPort メソッドを使用するためには、値を持つ 3 つのクラス変数を割り当てます。

portName は、プリンターへの通信ポートを指定します。

例): "BT" / USB:

portSettings は、用途に合わせて以下の設定を行います。

・プリンター使用の設定（必ず指定）
・リトライの設定
・データタイムアウト機能の設定

Interface	Port Name	Port Settings	
USB	USB: USB:シリアルナンバー	portable;	モバイルプリンター使用(必ず指定)
Bluetooth	BT: BT:デバイス名 BT:MAC アドレス	portable;	モバイルプリンター使用(必ず指定)
		portable;l	リトライ有効
		portable;d[数値]	データタイムアウト機能*有効
		portable;u	Android OS Ver. 4.0 または 4.1 を使用しており、ペアリングのセキュリティ方式に SSP を使用（必ず指定）

***データタイムアウト機能**
プリンターが印刷中にエラーになった場合のほか、指定時間、プリンターに対してデータが送られなかった場合にもデータキャンセル機能を動作させます。
指定時間は [数値] に 0～255（単位：秒）を設定します。0 を設定した場合、または、妥当な値を設定しなかった場合、この機能は無効となります。

この機能により、データ送信中に Bluetooth 接続が切断された場合に、次回の印刷が不正な内容になることを防ぐことができます。

Port Settings プロパティ 指定例）

portSettings = “portable;d127” …印刷中に 127 秒間データが送られなかった場合に、
データキャンセル機能を動作する場合

timeout は、ミリ秒のタイムアウト値で、内部制御と API での通信に使用されるものです。(このパラメータは、API が制限された時間内で完了することを保証しますが、正確なタイムアウトの長さを保証するものではありません。)



portSettings パラメータで[データタイムアウト機能の設定]を使用する場合、データタイムアウト機能の指定時間より 3 秒以上長くなるように設定してください。3 秒未満に設定した場合、内部制御により endCheckedBlock メソッドのタイムアウト時間は自動的に 3 秒長くなるように設定されます。

●メソッド

getPort

```
public static StarIOPort getPort(String portName, String portSettings, int TimeoutMillis)
                                                                    throws StarIOPortException
```

getPort は、プリンターポートのオープンに使用されます。前述で説明しました portName と portSettings に有効な値を使用し、プライベート変数に正しく設定することで、StarIO クラスに接続文字列を渡すことができます。

```
//The following would be an actual usage of getPort:
StarIOPort port = null;
try
{
   port = StarIOPort.getPort(portName, portSettings, 10000);
}
catch (StarIOPortException e)
{
   //There was an error opening the port
}
```

getPort メソッドを呼び出すことで、StarIOPort クラスのインスタンスを生成することができます。上記の例では、“port”を作成し、null を割り当てています。getPort を含む下の行で、実際のポートハンドルを取得しています。



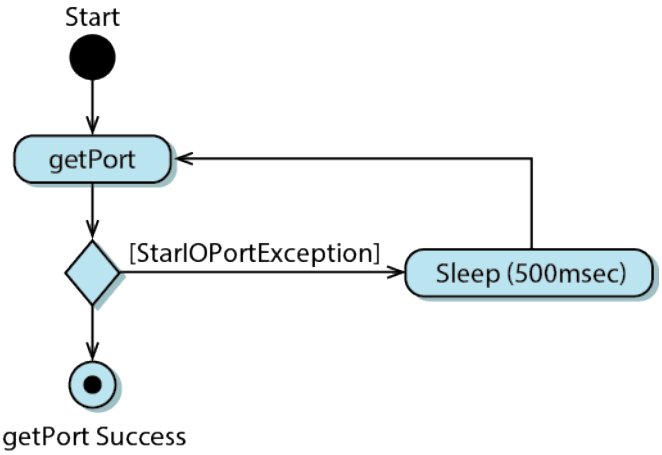
getPort を使用する場合は、常に try catch を使用してください。上記の例のような try catch を使用しなければ、通信エラーでポートオープンができなかった場合、プログラムはクラッシュする可能性があります。

Bluetooth プリンタを使用する場合、端末、使用環境によって、getPort が失敗し、Bluetooth デバイスとの接続ができず印刷、ステータス取得ができない場合があります。

getPort メソッドの PortSettings パラメータに“portable;l”を指定することで、Bluetooth デバイスとの接続が失敗した場合にリトライを実行します。timeout パラメータに指定した時間の間繰り返し、timeout 時間経過しても接続が成功しない場合、StarIOPortException が発生します。

getPort が失敗した場合(StarIOPortException が発生します)、下記 2.3 を繰り返す処理を行ってください。

◆getPort リトライのフロー



1.getPort が失敗する
(StarIOPortException が発生する)

2.一定時間(例：500msec)おく

3.getPort を実行する

4.getPort が成功した場合、印刷またステータス取得などの処理に進んでください。

(*) 一定時間の値や getPort の繰り返し回数は、端末や使用環境によって異なるため、十分な検証を行い決定してください。

searchPrinter

```
public static ArrayList<PortInfo> searchPrinter(String target, Context Context)
                                                        throws StarIOPortException
```

searchPrinter は、Star プリンターを検索し、検索結果を ArrayList<PortInfo>にて返します。引数 target には “BT:”, “USB:”を指定します。

戻り値の PortInfo クラスは、ポート名、プリンターの MAC アドレス、プリンターモデル名、USB シリアルナンバーを持ち、それぞれ String getPortName(), String getMacAddress(), String getModelName(), getUSBSerialNumber()メソッドで取得することができます。ポート名は、getPort()の portName 引数として使用することができます。

```
//The following would be an actual usage of searchPrinter:
try
{
   List<PortInfo> portList = StarIOPort.searchPrinter(“TCP:”);
   for (PortInfo port : portList) {
      Log.i (“LOG”, “Port Name: “ + port.getPortName());
      Log.i (“LOG”, “MAC Address: ” + port.getMacAddress());
      Log.i (“LOG”, “Model Name: “ + port.getModelName());
   }
 }
catch (StarIOPortException e)
{
   //There was an error opening the port
}
```

上記の例では、ネットワーク上のプリンターを検索して一覧を取得し、その内容をログに出力します。



searchPrinter()は、スターモバイルプリンターだけでなくすべての Bluetooth デバイスを検索します。識別しやすくするために、プリンターには固有の名称を付けて使用することを推奨します。



本 API はデバイスを確実に検出する事を保証するものではありません。



USB プリンターを接続し直した際は、再度“Search”をタップしてください。

readPort

```
public int readPort(Byte[] readBuffer, int offset, int size)
                                                    throws StarIOPortException
```

このメソッドは、デバイスからデータを読み込みます。プリンターから Raw byte を読み取る必要のある場合のみ、ご使用ください。



Raw Status の取得にこのメソッドを使用しないでください。

Status の取得は、RetreiveStatus メソッドを使用してください。

Parameters:

readbuffer	– データが読み込まれる Byte 配列のバッファ
offset	- ReadBuffer にデータを書き込み始める場所を指定
size	– 読み込むバイト数の合計

Returns:

実際に読み込まれたバイト数。データが全て読み取れなかった時でも、この関数は成功します。アプリケーションは、期待されるデータが全て読み取れるまで、この関数を複数回、呼び出す必要があります。または、しきい値に達するまで再試行をするようにします。

Throws:

StarIOPortException – 通信エラーが発生したとき

releasePort

```
public static void releasePort(StarIOPort port)
```

このメソッドは、指定されたポートへの接続をクローズします。

Parameters:

port –以前に初期化されたポートを表すポートタイプ



データ通信を行わない時は、ポートをリリース(クローズ)してください。
ポートをオープンしたままにしておくと、今後のポートオープンの呼び出し要求が失敗する可能性があります。

writePort

```
public void writePort(Byte[] writeBuffer, int offset, int size)
                                                    throws StarIOPortException
```

このメソッドは、デバイスにデータを書き込みます。コマンド等を送信し、プリンターに印刷をする時に使用します。次に、このメソッドの使用例を示します。

下記は、プリンターにデータを送信する最も簡単な方法です。
SDK の“printToPrinter”には、プリンターにデータが送信されたことを確認する方法を示すコードを記述してあります。しかし、下記コードは単にデータを送信しているだけです。

```
//Set a byte array to send to the printer
//command = { A, B, C, D, Feed 3mm, Full Cut}
Byte[] command = new Byte[]{ 0x41, 0x42, 0x43, 0x44, 0x1B, 0x7A, 0x00, 0x1B, 0x64,
0x02 };

try
{
   port.writePort(command, 0, command.length);
}
catch (StarIOPortException e)
{
   //There was an error writing to the port
}
```

※安全なプログラミングをするために try catch を使用してください。

Parameters:

writeBuffer – 出力データを格納する Byte 配列のバッファ
offset – writeBuffer からデータを読み込み始める場所を指定
size – 書き込むバイト数の合計

Throws:

StarIOPortException – 通信エラーが発生したとき

retreiveStatus

public StarPrinterStatus **retreiveStatus** ()
throws StarIOPortException

このメソッドは、StarIO で詳細なステータスをプリンターから取得します。

Returns:

StarPrinterStatus 構造体は、現在のデバイスのステータスを保持します。

Throws:

StarIOPortException – 通信エラーが発生したとき

このメソッドは StarPrinterStatus と呼ばれる StarIO の構造体を使用します。
この構造体は、ブーリアン型とバイナリーの両方の形式でプリンターステータスを保持します。
下記を行うことにより、プロジェクトで StarPrinterStatus オブジェクトを作成してください。

```
StarPrinterStatus status = port.retreiveStatus();

if (status.offline == false)
{
    if (status.blackMarkError == true) {
        //There was a black mark error
    }

    if (status.compulsionSwitch == true) {
        //Cash drawer is open
    }
    else {
        //Cash drawer is closed
    }
}
else {
    //If true, then the printer is offline.
}
```

StarPrinterStatus 構造体　ステータスリスト

メンバ名	説明	型	詳細
blackMarkError	ブラックマークエラー	boolean	ブラックマークエラー(ブラックマーク設定時に非ブラックマーク用紙を使って印刷した場合等に発生)の時に true となる。通常時は false。
compulsionSwitch	コンパルジョン SW	boolean	ドロワのコンパルジョン SW が押されていると true となる。通常は false。
coverOpen	カバーの状態	boolean	カバーが開いている場合に true となる。 閉じている場合は false。
cutterError	オートカッターエラー	boolean	カッターエラー発生時に true となる。 通常は false。
etbAvailable	ETB 使用可否	boolean	ETB が使用可能な場合に true となる。 使用できない場合は false。
etbCounter	ETB カウンタ	int	現在の ETB カウンタの値。
headThermistorError	ヘッドサーミスタエラー	boolean	ヘッドサーミスタ異常値検出時に true となる。 通常は false。
offline	ON-LINE/OFF-LINE 状態	boolean	オフラインの場合に true となる。 オンライン時は false。
overTemp	印字ヘッド高温による停止中	boolean	ヘッドが高温になり印刷停止している状態で true となる。通常は false。
presenterPaperJamError	プレゼンター紙ジャムエラー	boolean	プレゼンター装着時、プレゼンターで用紙ジャムが発生すると true となる。通常は false。
presenterState	プレゼンタ用紙位置	int	以下のいずれかの数値。 0: プレゼンタ内部に用紙がない 1: 用紙を給紙した状態(ループ状態) 3: 用紙を排出した状態 6: 用紙回収状態 7: 用紙が引き抜かれた状態
raw	ステータスのバイト列	byte[63]	ステータスのバイト列 (例: HEX 23 86 00 00 00 00 00 00 00)
rawLength	raw の長さ	int	raw の長さ
receiptPaperEmpty	用紙エンド	boolean	用紙切れの場合は true となる。 通常は false。
receiptPaperNearEmptyInner	用紙ニアエンド(内側)	boolean	用紙ニアエンド状態の時に true となる。 通常は false。
receiveBufferOverflow	受信バッファオーバーフロー	boolean	受信バッファフルの時に true となる。 通常は false。
unrecoverableError	復帰不可能エラー	boolean	復帰不可能エラー(ヘッドサーミスタエラー、オートカッターエラー、電源電圧エラー等)が発生した場合に true となる。通常は false。
voltageError	電源電圧エラー	boolean	電源電圧異常値検出時に true となる。 通常は false。

StarPrinterStatus 構造体　機種別対応リスト

メンバ名	SM-L200	SM-S210i (JP モデルのみ)	SM-S220i (欧米モデルのみ)	SM-S230i (欧米モデルのみ)	SM-T300i	SM-T400i	mPOP
blackMarkError	✓	✓	✓	✓	✓	✓	
compulsionSwitch							✓
coverOpen	✓	✓	✓	✓	✓	✓	✓
cutterError							✓
etbAvailable	✓	✓	✓	✓	✓	✓	✓
etbCounter	✓	✓	✓	✓	✓	✓	✓
headThermistorError							✓
offline	✓	✓	✓	✓	✓	✓	✓
overTemp	✓	✓	✓	✓	✓	✓	✓
presenterPaperJamError							
presenterState							
raw	✓	✓	✓	✓	✓	✓	✓
rawLength	✓	✓	✓	✓	✓	✓	✓
receiptPaperEmpty	✓	✓	✓	✓	✓	✓	✓
receiptPaperNearEmptyInner							
receiveBufferOverflow							
unrecoverableError							✓
voltageError							✓

beginCheckedBlock

public StarPrinterStatus **beginCheckedBlock** ()
throws StarIOPortException

このメソッドは、endCheckedBlock メソッドとセットで使用して印字終了の確認を行います。最後まで正しく印刷されたか確認したい場合に、印刷データ送信の直前に本メソッドを、印刷データ送信の直後に endCheckedBlock メソッドを実行します。

Returns:
StarPrinterStatus 構造体は、現在のデバイスのステータスを保持します。

Throws:
StarIOPortException – 通信エラーが発生したとき
プリンターがオフラインのとき

サンプルコードはこちらをご参照ください。

endCheckedBlock

public StarPrinterStatus **endCheckedBlock** ()
throws StarIOPortException

このメソッドは、beginCheckedBlock メソッドとセットで使用します。

プリンタの状態を監視し、送信した印刷データの印刷が完了すると制御を返します。
印刷データ以外を送信した場合は、そのコマンドがプリンタに処理されると制御を返します。

タイムアウト時間(*1) 内に印刷が完了しなかった場合や、印刷中にプリンタエラーが発生した場合は、例外 StarIOPortException をスローします。

(*1) タイムアウト時間は、setEndCheckedBlockTimeoutMillis メソッドで設定した値が使用されます。初期値は getPort で指定したタイムアウト時間となります。
setEndCheckedBlockTimeoutMillis メソッドで設定する値は、印刷時間より長くなるよう調整してください。また、10 秒未満の値が指定された場合にはタイムアウトは 10 秒になります。

Returns:
StarPrinterStatus 構造体は、現在のデバイスのステータスを保持します。

Throws:
StarIOPortException
- 通信エラーが発生した
- タイムアウト時間内にプリンターからの応答がない

サンプルコードはこちらをご参照ください。

```
try
{
    port = StarIOPort.getPort(portName, portSettings, 10000, context);

    //Start checking the completion of printing
    StarPrinterStatus status = port.beginCheckedBlock();

    //Printing
    byte[] command = PrinterFunctions.createPrintData(paperWidthInch);
    port.writePort(command, 0, command.length);

    //End checking the completion of printing
    status = port.endCheckedBlock();

    if (status.offline == true) {
       //If true, then the printer is offline.
    }
}
catch (StarIOPortException e)
{
    Log.d("StarIOSample", "An timeout error has occurred during printing.");
}
finally
{
    if (port != null)
    {
        try
        {
            StarIOPort.releasePort(port);
         }
         catch (StarIOPortException e) {}
    }
}
```

setEndCheckedBlockTimeoutMillis

public void **setEndCheckedBlockTimeoutMillis** (int endCheckedBlockTimeoutMillis)

endCheckedBlock メソッドのタイムアウト時間を設定します。[単位: ミリ秒]

印刷に時間がかかる場合、この値を大きくする事で endCheckedBlock メソッドの印刷完了待ち時間を伸ばす事ができます。

初期値は、getPort メソッドで指定したタイムアウト時間となります。



指定した時間が 10 秒未満の場合、タイムアウトは 10 秒になります。

また、getPort メソッドの portSettings パラメータで[データタイムアウト機能の設定]を使用する場合、このメソッドで設定するタイムアウト時間には、データタイムアウト機能の指定時間より 3 秒以上長くなるように設定してください。3 秒未満に設定した場合、内部制御により自動的に 3 秒長くなるように設定されます。

getFirmwareInformation

public Map<String, String> **getFirmwareInformation**()
throws StarIOPortException

このメソッドは、プリンターのモデル名とファームウェアバージョンを取得します。

Returns:
取得結果を Map<String, String>にて返します。
Key に”ModelName”を設定することで戻り値からモデル名を取得します。
Key に”FirmwareVersion”を設定することで戻り値からファームウェアバージョンを取得します。

Throws:

StarIOPortException – 通信エラーが発生したとき

Note:
・取得に失敗した場合、空文字を返します。

getStarIOVersion

```
public static synchronized String getStarIOVersion()
```

このメソッドは、StarIO のバージョンを取得します。

Returns:
StarIO バージョン

StarBluetoothManager クラス

StarBluetoothManager クラスは、Bluetooth インターフェイスの各種設定を行うためのクラスです。StarIO Port クラスと同時に使用しないでください。

●コンストラクタ

StarBluetoothManager

```
public StarBluetoothManager(String portName, String portSetting, int timeoutMillis,
StarBluetoothManager.StarDeviceType starDeviceType)
                                                    throws StarIOPortException
```

Parameters:

portName – 接続するデバイスのポート名
例）BT:StarMicronics

portSetting - デバイスに接続するためのポート設定

timeoutMillis
– ミリ秒のタイムアウト値で、内部制御と API での通信に使用される
（このパラメータは、API が制御された時間内で完了することを保証
しますが、正確なタイムアウトの長さを保証するものではありません。）

starDeviceType
– 検索するデバイスの種類
例）StarDeviceTypeDesktopPrinter

Throws:

StarIOPortException – starDeviceType に StarDeviceTypePortablePrinter が設定されているとき
– portName に BT:が設定されていないとき

●メソッド

open

```
public void open()
```

StarBluetooth デバイスとの接続を開きます。
Open メソッド実行後は、必ず loadSetting メソッドを実行し、現在の設定を取得してください。

Throws:

StarIOPortException – 通信エラーが発生したとき

loadSetting

```
public void loadSetting()
```

接続した StarBluetooth デバイスから設定を取得します。

Throws:

StarIOPortException – 通信エラーが発生したとき

close

```
public void close()
```

StarBluetooth デバイスとの接続を閉じます。

Throws:

StarIOPortException – 通信エラーが発生したとき

apply

```
public void apply()
```

StarBluetooth デバイスには以下のメソッドで指定した値を設定します。

- setBluetoothDevice メソッド
- setiOSPortName メソッド
- setAutoConnect メソッド
- setPinCode メソッド
- setSecurityType メソッド

Throws:

StarIOPortException – 通信エラーが発生したとき

– SecurityType が PIN コードに設定されていて、AutoConnect が ON に設定されているとき

getBluetoothDeviceName

```
public String getBluetoothDeviceName()
```

loadSetting メソッドで取得した、または SetBluetoothDeviceName メソッドで設定した Bluetooth デバイス名を取得します。loadSetting メソッド、または setBluetoothDeviceName メソッドを呼び出した後に使用してください。

Returns:

BluetoothDeviceName

setBluetoothDeviceName

public void **setBluetoothDeviceName**(String BluetoothDeviceName)

Bluetooth デバイス名を 1 文字以上、16 文字以下で設定します。
Bluetooth デバイス名を変更するには、setBluetoothDeviceName メソッド後に apply メソッドを実行してください。

Parameters:

BluetoothDeviceName – Bluetooth デバイス名

Throws:

StarIOPortException – 使用可能文字列以外を設定したとき
指定した文字列が 1 文字以上、16 文字以下でないとき

使用可能文字列： 0-9 a-z A-Z ; : ! ? # $ % & , . @ _ - = Space
/ * + ~ ^ [{ ( ] }) | \

getiOSPortName

public String **getiOSPortName**()

loadSetting メソッドで取得した、または setiOSPortName メソッドで設定した iOSPort 名を取得します。loadSetting メソッド、または setiOSPortName メソッドを呼び出した後に使用してください。

Returns:

iOSPortName

setiOSPortName

```
public void setiOSPortName(String iOSPortName)
```

iOSPort 名を 1 文字以上 16 文字以下で設定します。
iOSPort 名を変更するには、setiOSPortName メソッド後に apply メソッドを実行してください。

Parameters:

iOSPortName – iOS ポート名

Throws:

StarIOPortException – 使用可能文字列以外を設定したとき
指定した文字列が 1 文字以上、16 文字以下でないとき

使用可能文字列： 0-9 a-z A-Z ; : ! ? # $ % & , . @ _ - = Space
/ * + ~ ^ [{ ( ] }) | \

getPinCode

```
public String getPinCode()
```

setPinCode メソッドで設定した PIN コードの設定を取得します。setPinCode メソッドを呼び出した後に使用してください。

Returns:

PinCode



mPOP は非対応です。

setPinCode

```
public void setPinCode(String pinCode)
```

StarBluetooth デバイスの PIN コードを SM-L200 では 4 文字(固定)、それ以外の機種は 4 文字以上、16 文字以下で設定します。
PIN コードを変更するには、setPinCode メソッド後に apply メソッドを実行してください。

Parameters:

PinCode – PIN コード
例）1234

Throws:

StarIOPortException – 使用可能文字列以外を設定したとき
指定した文字列が 4 文字以上、16 文字以下でないとき

使用可能文字列： 0-9 （SM-L200）

0-9 a-z A-Z （その他のモバイルプリンター ）



mPOP は非対応です。

getAutoConnect

```
public boolean getAutoConnect()
```

loadSetting メソッドで取得した、または setAutoConnect メソッドで設定した AutoConnection の設定を取得します。loadSetting メソッド、または setAutoConnect メソッドを呼び出した後に使用してください。

Returns:

true - AutoConnection 設定が ON の場合
false - AutoConnection 設定が OFF の場合



Android デバイスで使用する場合は AutoConnection を
OFF にしてください。

setAutoConnect

```
public void setAutoConnect(boolean autoConnect)
```

StarBluetooth デバイスの AutoConnection を設定します。
AutoConnection の設定を変更するには、setAutoConnect メソッド後に apply メソッドを実行してください。

Parameters:

autoConnect　　例）false



Android デバイスで使用する場合は AutoConnection を OFF にしてください。

getSecurityType

```
public StarBluetoothManager.StarBluetoothSecurity getSecurityType()
```

loadSetting メソッドで取得した、または setSecurityType で設定した Bluetooth セキュリティタイプ（SSP or PIN Code）を取得します。loadSetting メソッドまたは setSecurityType メソッドを呼び出した後に使用してください。

Returns:

StarBluetoothManager.StarBluetoothSecurity

setSecurityType

```
public void setSecurityType(StarBluetoothManager.StarBluetoothSecurity securityType)
```

StarBluetooth デバイスのセキュリティタイプ（SSP or PIN Code）を設定します。セキュリティタイプを変更するには、setSecurityType メソッド後に apply メソッドを実行してください。

Parameters:

securityType　　例）StarBluetoothManager.StarBluetoothSecurity.SSP

getPortName

```
public String getPortName()
```

StarBluetoothManager コンストラクタで指定した portName を取得します。

Returns:

portName

getPortSetting

```
public String getPortSetting()
```

StarBluetoothManager コンストラクタで指定した portSetting を取得します。

Returns:

portSetting

getTimeoutMillis

```
public int getTimeoutMillis()
```

StarBluetoothManager コンストラクタで指定した timeoutMillis を取得します。

Returns:

timeoutMillis

getDeviceType

```
public String getDeviceType()
```

StarBluetoothManager コンストラクタで指定した starDeviceType を取得します。

Returns:

StarBluetoothManager.StarDeviceType

isOpened

```
public boolean isOpened()
```

ポートの状態を取得します。

Returns:

true - ポートがオープンされている

StarIO Android SDK 機能

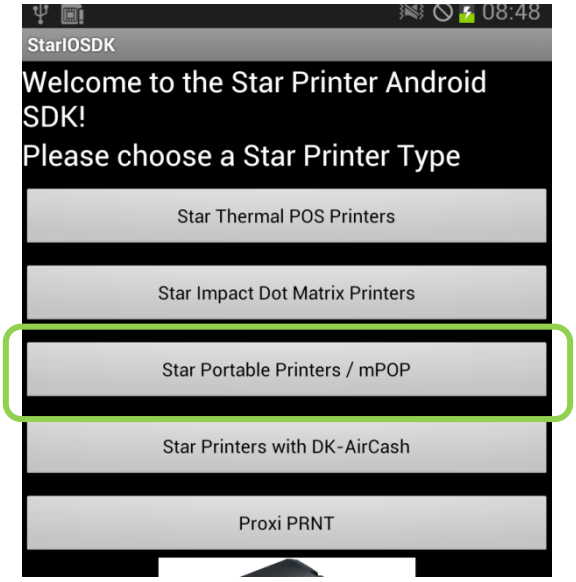
SDK 機能と StarIO のプリンターコマンドについての概要

これらのコマンドは、全て StarPRNT コマンド仕様書に記載されています。

特定のコマンドに関する詳細な情報が必要な場合は、StarPRNT コマンド仕様書をダウンロードし参照してください。

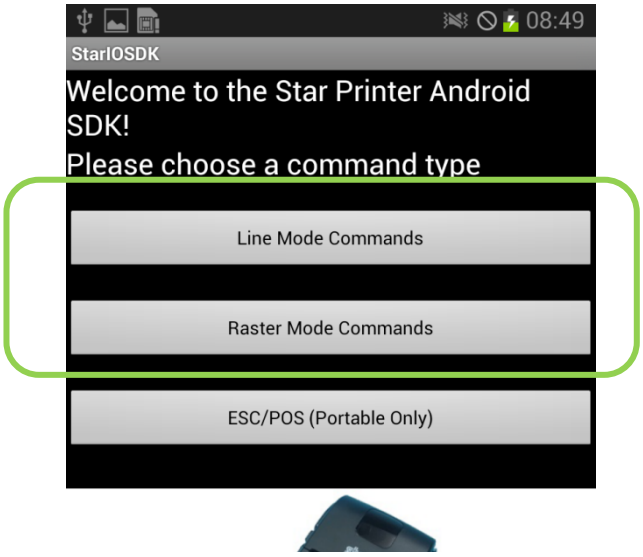
プリンターとコマンド種類の選択

1. “Star Portable Printers / mPOP”をタップします。



2. “Line Mode Commands”、”Raster Mode Commands”を選択してください。

“Line Mode”、”Raster Mode”の詳細についてはここを参照してください。



コマンド別サンプル機能対応

Line Mode Command Samples Include:

Port Discovery
Get Firmware Information
Get Status
Sample Receipt
1D Barcodes
2D Barcodes
MSR
Text Formatting
Bluetooth Setting
USB Setting

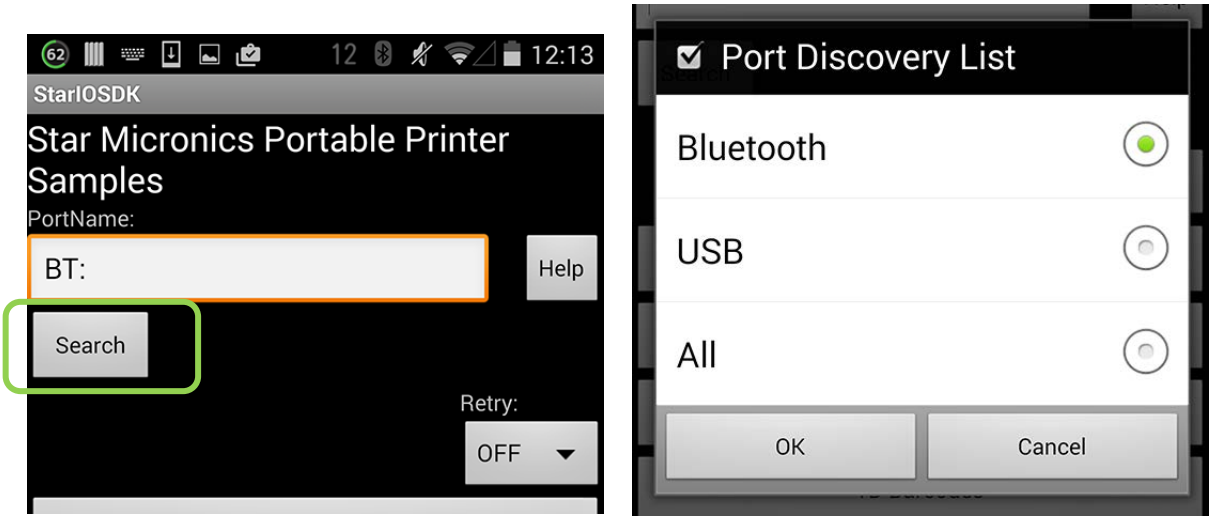
Raster Mode Command Samples Include:

Port Discovery
Get Firmware Information
Get Status
Sample Receipt
Raster Graphics Text Printing
Image File Printing
MSR
Bluetooth Setting
USB Setting

SDK を実行すると、それぞれに機能が割り振られたボタンが表示されます。
このセクションでは、各機能の動作及び、使用しているコマンドについて説明します。

Port Discovery

◆Line ◆Raster

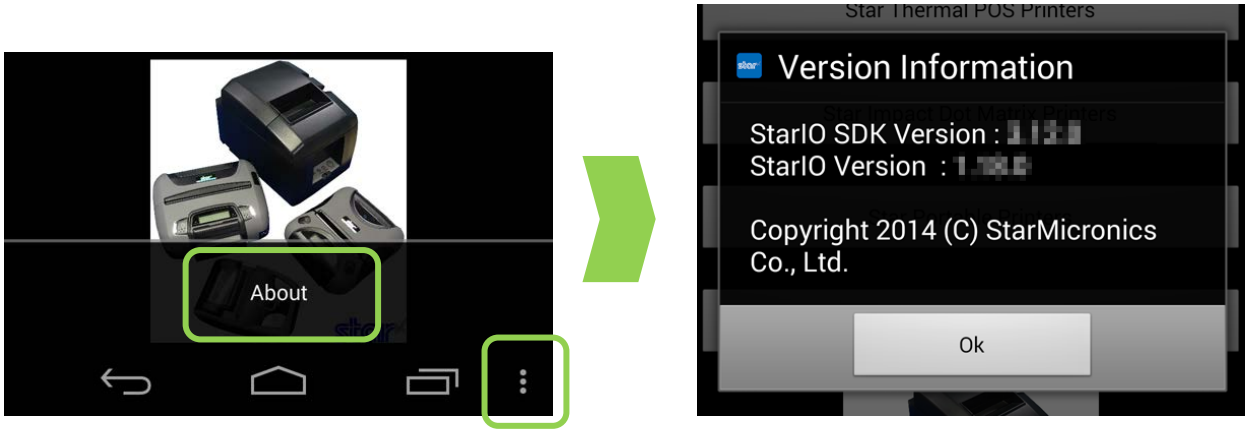


Star プリンターを自動的に検索します。検索結果より、
接続したいプリンターを選択してください。詳細はここを参照してください。

---

get StarIOVersion

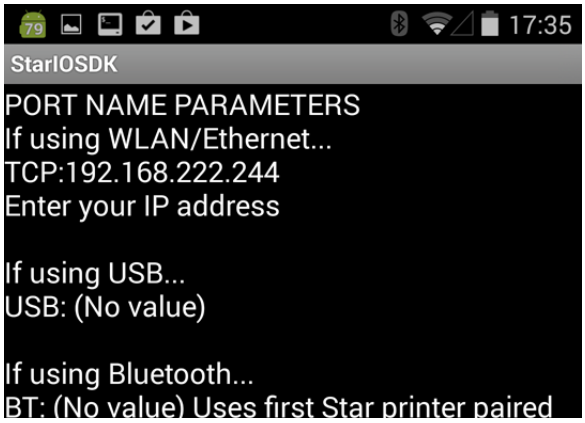
◆Line ◆Raster



StarIO のバージョンを取得します。

Help

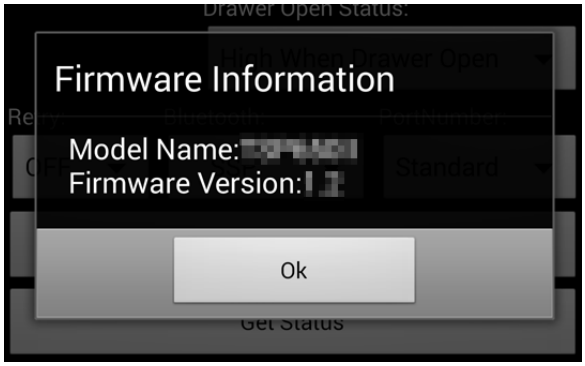
◆Line ◆Raster



Help は、Bluetooth/USB/Ethernet のパラメータと StarIO ポートの設定に関する情報を表示します。

Get Firmware Information

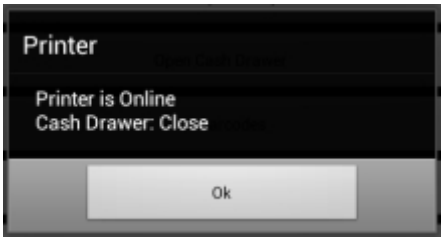
◆Line ◆Raster



Port Name に設定されたプリンターのファームウェア情報を表示します。

Get Status

◆Line ◆Raster



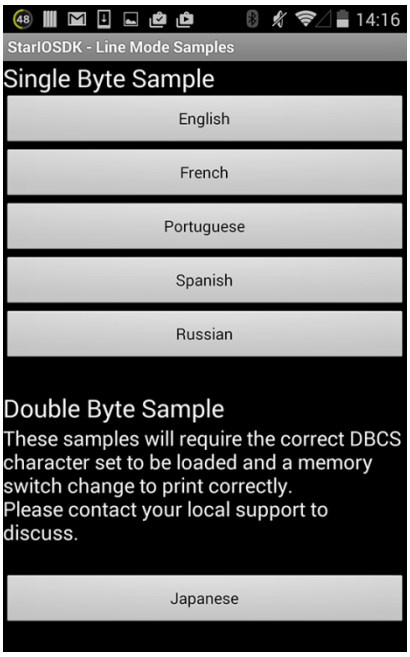
StarPrinterStatus

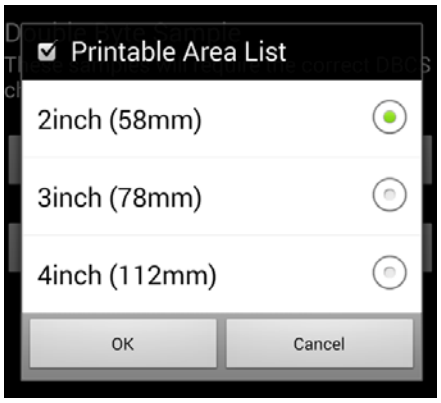
public boolean retreiveStatus()	ステータスの戻り値はここを参照してください
offline	false = プリンターオンライン
	true = プリンターオフライン
other	ステータスの戻り値はここを参照してください

Sample Receipt

◆Line ◆Raster

Raster mode の場合:





選択したコマンド種類で、選択した言語でのサンプルレシートの印刷をします。
レシート幅を選択し、OK ボタンをタップして印刷を開始してください。

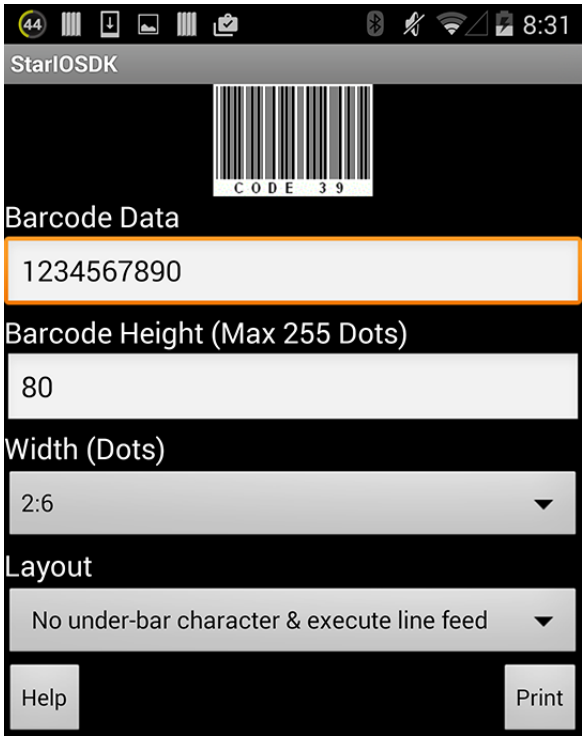
※Raster Mode で使用する場合、Android 端末によってサンプルレシートのレイアウトが崩れる場合があります。その場合は、Typeface の指定を変更するかレイアウトの調整を行ってください。

△ Line Mode では、SM-S210i, SM-S220i, SM-S230i, SM-T300i, SM-T400i でロシア語(Russian)、簡体字(Simplified Chinese)の印刷に対応しておりません。

△ Line Mode では、SM-L200 での簡体字(Simplified Chinese)の印刷にはファームウェアバージョン 1.1 から対応いたします。

1D Barcodes

◆Line



バーコードの設定

ESC b n1 n2 n3 n4 d1 ... dk RS

n1 = バーコード種選択
0 = UPC-E *1 1 = UPC-A *1 2 = JAN/EAN8 *1 3 = JAN/EAN13 *1
4 = Code39 5 = ITF 6 = Code128 7 = Code93 8 = NW-7 *1

n2 = バー下文字選択、及び、改行付加選択
1 = バー下文字を付加しない。 バーコード印字後、改行を実行する。
2 = バー下文字を付加する。 バーコード印字後、改行を実行する。
3 = バー下文字を付加しない。 バーコード印字後、改行を実行しない。
4 = バー下文字を付加する。 バーコード印字後、改行を実行しない。

n3 = バーコードモード選択

n4 = バーコード高さ(ドット数)

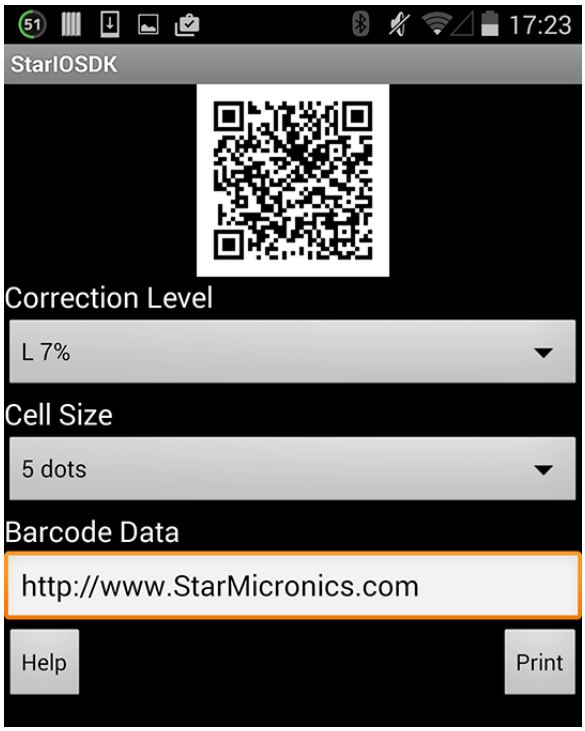
*1: これらのバーコードは、Star プリンターでサポートされていますが、SDK では実装されていません。

注記：このサンプルアプリケーションは Raster モードでの 1D Barcodes 印刷に対応していません。
Raster モードではバーコードを画像データとしてプリンターに送信する必要があります。

2D Barcodes

◆Line

＜＜QR Code＞＞



QR Code の指定

QR Code に関するコマンドは、その機能により以下の 5 つに分類される。

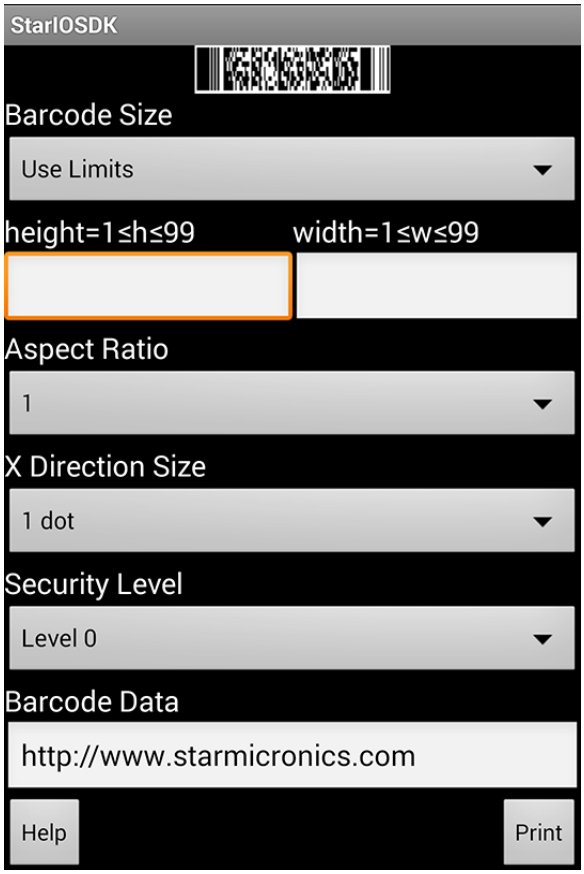
(1) モデルの指定	ESC GS y S 0 n
(2) エラー訂正レベルの指定	ESC GS y S 1 n
(3) セルサイズの指定	ESC GS y S 2 n
(4) QR Code データの指定	ESC GS y D 1 NUL nL nH d1…dk
(5) QR Code の印刷	ESC GS y P

以下の順番でコマンドを送信することにより、QR Code が印刷できます。
モデル指定 + エラー訂正レベル + セルサイズ + QR Code データ + QR Code 印刷

※QR Code コマンドの詳細は、StarPRNT コマンド仕様書をご参照ください。

注記：このサンプルアプリケーションは Raster Mode での 2D Barcodes 印刷に対応していません。
Raster Mode ではバーコードを画像データとしてプリンターに送信する必要があります。

＜＜PDF417＞＞



PDF417 の指定

PDF417 に関するコマンドは、その機能により以下の 7 つに分類される。

(1) サイズの指定	ESC GS x S 0 n p1 p2
(2) エラー訂正レベルの指定	ESC GS x S 1 n
(3) モジュールサイズ(X 方向)の指定	ESC GS x S 2 n
(4) モジュールサイズ(Y 方向)の指定	ESC GS x S 3 n
(5) PDF417 データの指定	ESC GS x D nL nH d1 d2…dk
(6) PDF417 の印刷	ESC GS x P

以下の順番でコマンドを送信することにより、PDF417 が印刷できます。
サイズ + エラー訂正レベル + モジュール(X) + モジュール(Y) + PDF417 データ + PDF417 印刷

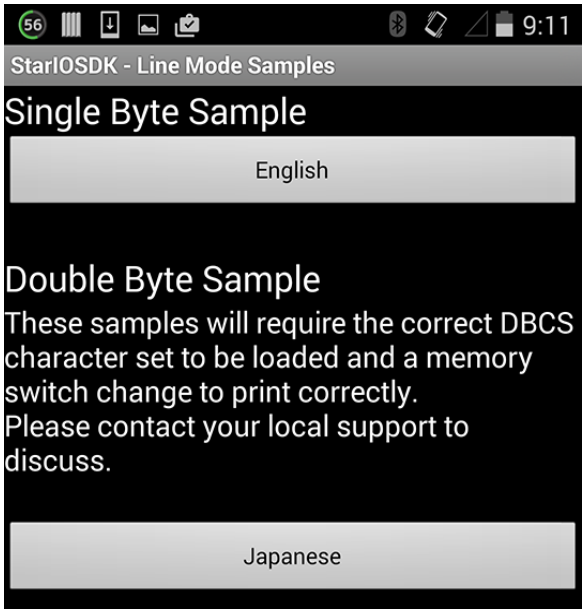
※PDF417 コマンドの詳細は、StarPRNT コマンド仕様書をご参照ください。

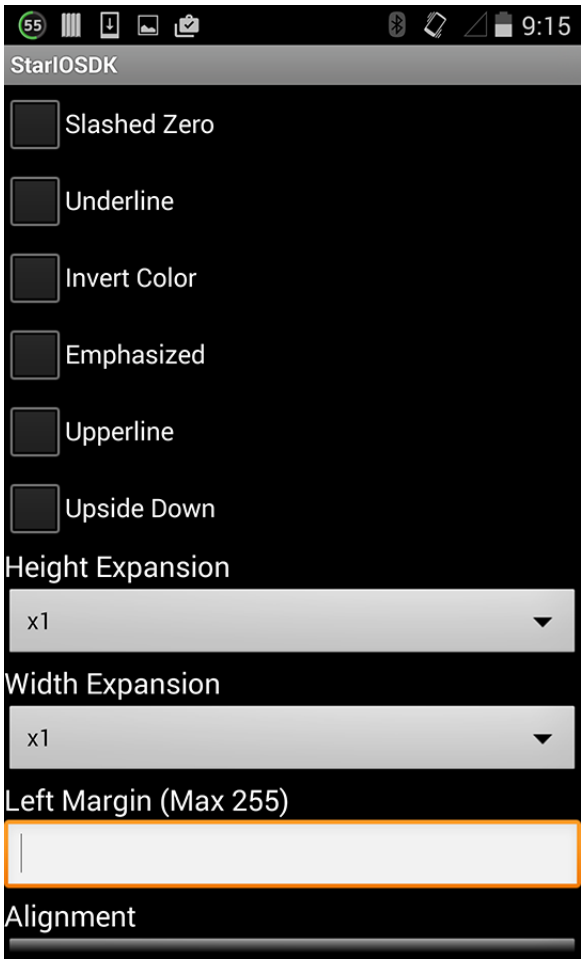
注記：このサンプルアプリケーションは Raster Mode での 2D Barcodes 印刷に対応していません。
Raster Mode ではバーコードを画像データとしてプリンターに送信する必要があります。

SM-L200 はファームウェアバージョン 1.1 から PDF417 に対応します。

Text Formatting

◆Line





***English screen**

文字の書式設定

Slashed Zero:スラッシュゼロ（English のみ）

ESC / n　　1 = 設定　　0 = 解除 [デフォルト]

シフト JIS 漢字モードの指定（Japanese のみ）

ESC $ n　　1 = シフト JIS 漢字モード設定 0 = 解除 [デフォルト]

Underline:アンダーライン

ESC - n　　1 = 設定　　0 = 解除 [デフォルト]

Invert Color:白黒反転印字

ESC 4 = 設定　　ESC 5 = 解除 [デフォルト]

Emphasized:強調印字

ESC E = 設定　　ESC F = 解除 [デフォルト]

Upperline:アッパーライン

ESC _ n　　1 = 設定　　0 = 解除 [デフォルト]

Upside Down:倒立印字

SI = 設定　　DC2 = 解除 [デフォルト]

文字サイズの拡大

Height Expansion:高さ	ESC h n	0 ≦ n ≦ 5 (SM-L200 以外)
		0 ≦ n ≦ 2 (SM-L200)
Width Expansion:幅	ESC W n	0 ≦ n ≦ 5

Left Margin:左マージンの設定

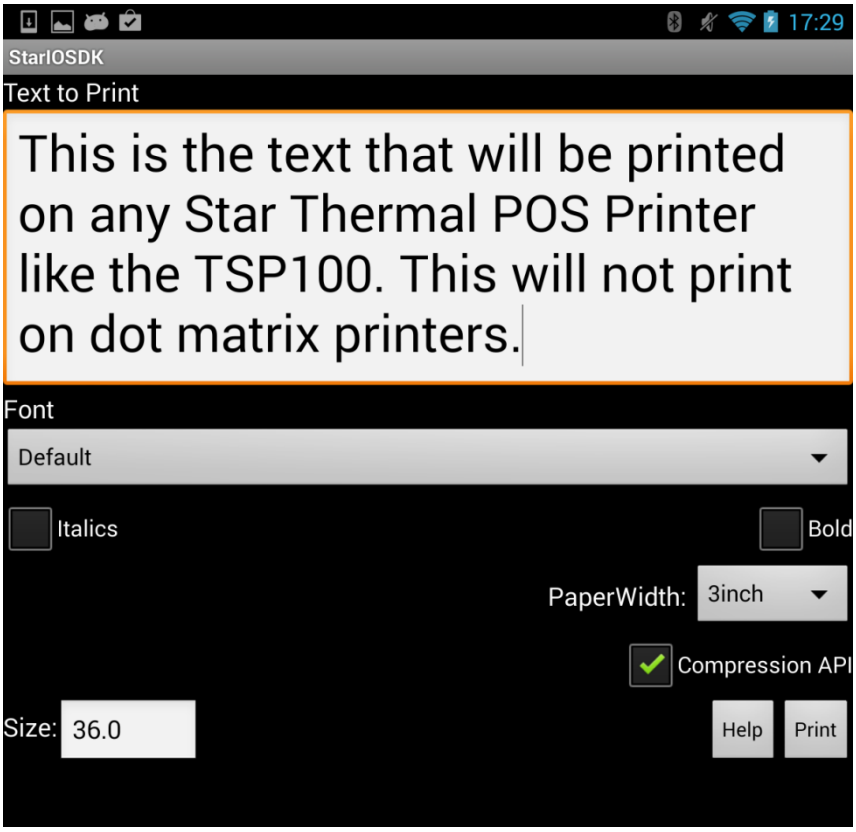
ESC l n　　0 ≦ n ≦ 255

Alignment:位置揃え

左揃え [デフォルト]	ESC GS a 0
中央寄せ	ESC GS a 1
右揃え	ESC GS a 2

注記：Raster Mode を使用されている場合は、Raster Graphical Text Printing をご参照ください。

Raster Graphical Text Printing ◆Raster



Raster Mode では、全ての印刷データをイメージデータに変換してプリンターへ送信します。これにより、文字データおよびロゴ／クーポンの付加された印刷を高速で行うことができます。

Raster コマンドの詳細については、「StarPRNT コマンド仕様書」をご参照ください。また、画面右下に表示される[Help]をタップすることにより、Raster コマンドについて確認できます。

“Compression API”を使用することにより、スループットの向上が望めます

※大容量の Raster コマンドデータを送信する場合、データ落ちを防ぐため writePort する前後に **beginCheckedBlock** / **endCheckedBlock** を使用してください。
詳細は”PrinterFunctions.java”内にある”printBitmap”メソッドをご参照ください。

Image File Printing

◆Raster



イメージデータの Raster 印刷に使用する画像を、画面上のドロップダウンリストから選択してください。

Raster Mode では、全ての印刷データをイメージデータに変換してプリンターへ送信します。これにより、文字データおよびロゴ／クーポンの付加された印刷を高速で行うことができます。

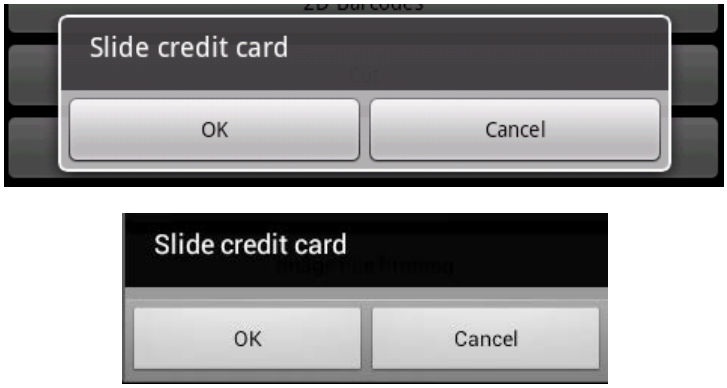
“Compression API”を使用することにより、スループットの向上が望めます

注記: このイメージデータは 80mm 幅のレシートに合わせて作成されています。
80mm 幅以外の用紙を使用されている場合、自動的にリサイズされません。

Raster コマンドの詳細については、「StarPRNT コマンド仕様書」をご参照ください。

また、画面右下に表示される[Help]をタップすることにより、Raster コマンドについて確認できます。

MSR (磁気カードリーダー)



ISO トラックのデータ読み取り

ESC M C　　第一または、第二トラックを読み取る

ISO トラックのデータ読み取り

ESC M D　　第二または、第三トラックを読み取る

ISO 2 トラックのデータ読み取り

ESC M E　　第一、第二または、第二第三トラックを同時に読み取る

JIS-II 磁気カードのデータ読み取り

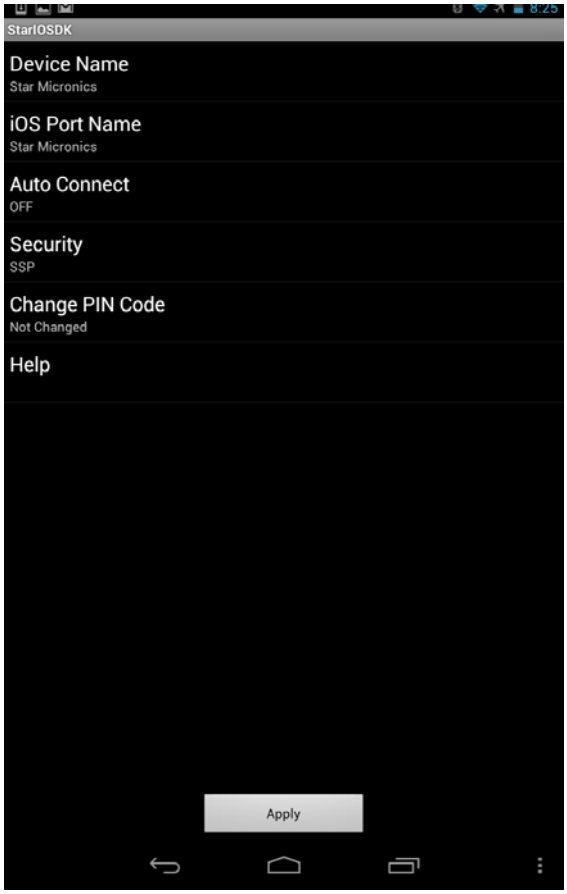
ESC M J　　JIS-II(JIS)カードのデータを読み取る

磁気カード読み取りモードのキャンセル

EOT　　読み取りを中止し磁気カード読み取りモードを終了する

Bluetooth Setting

◆Line ◆Raster

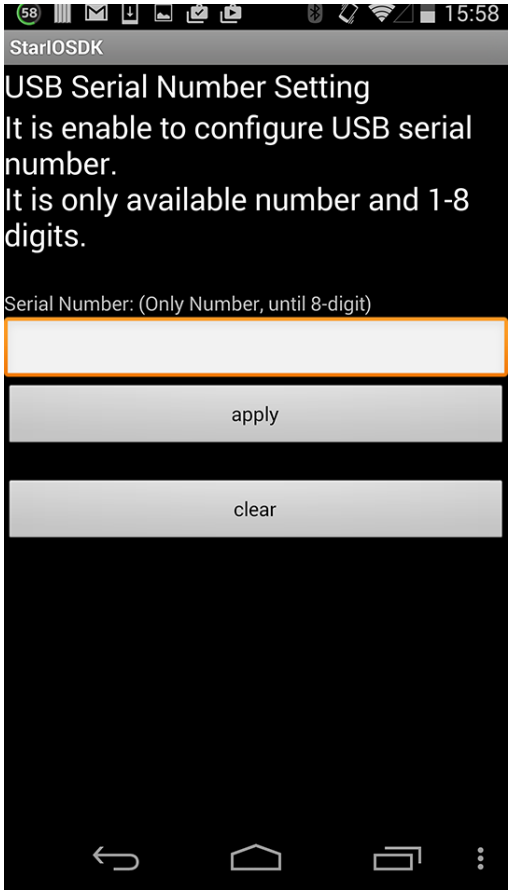


PortName に指定された Bluetooth デバイスに接続して、Bluetooth インターフェイスの各種設定を変更します。

変更された値は、デバイスの電源再投入・再ペアリングを行った後に有効になります。

USB Setting

◆Line ◆Raster



PortName に指定された Bluetooth デバイスに接続して、Bluetooth インターフェイスの各種設定を変更します。

変更された値は、デバイスの電源再投入を行った後に有効になります。

StarIO を使用するアプリケーション開発のために

#1: 大規模なプロジェクトをコーディングしている場合、全ての印刷メソッドを抽象化してクラスを作成してください。これはソースコードの再利用に役立ちます。また、特定のコードを見つけるのが容易になり、時間の節約にもなります。StarIO を唯一のクラスに存在させることによって、オブジェクト指向プログラミングを実現できます。

#2: ASCII と Unicode、16 進と 10 進、及び、Byte と Char の違いと定義が何であるかを確認してください。1 バイトは、通常、8 桁の 2 進数(1 と 0)で 8 ビットです。これらのバイトは、ちょうど 8 ビットのバイナリーデータです。しかし、byte は int または char でもありえます。3 つの異なる変数の型は、基本的に同じ方法でデータを保持しますが、わずかな違いがあります。印刷ジョブのデータを格納する変数を選択する際、byte の代わりに char、int、または string で試してみてください。
ASCII から Unicode(また逆も同様に)への変換は、時として安全ではありません。そのため、Encoding クラスがどのように機能するのかを確認してください。Unicode でよく見られる間違いの例として、“Culture-sensitive searching and casing”、“Surrogate pairs”、“Combining characters”、“Normalization”などがあります。こちらのサイトをご確認ください。

#3: 16 進ダンプモードについて 作成したアプリケーションのデバッグの際、プリンターの 16 進ダンプモードを使用してください。16 進ダンプモードを使用することで、アプリケーションから送信したデータをプリンターが正しく受信できたか確認する事ができます。16 進ダンプモードへの設定は、各プリンターの製品仕様書を参照ください。

#4: StarIO コマンド・コードをリバース・エンジニアリングしないでください。全ての StarIO コマンドは、コマンド仕様書より参照可能です。また、本 SDK を活用することで、アプリケーション作成時の工数を大幅に削減可能です。

#5: 本 SDK に記載されていないコマンドについては、SDK のコードサンプルを参照してください。また、**スター精密グローバルサポートサイト**の Developers セクションにアクセスすることで、より詳細な情報を入手可能です。

#6: Android 以外の OS(例: iOS) に対応した SDK をお探しの場合には、**スター精密グローバルサポートサイト**の Developers セクションをご覧ください。

追加リソース

以下のリンクよりプログラマーマニュアルを入手してください。

スター精密 サポート情報

スター精密グローバルサポートサイト

FAQ を参照してください。
グローバルサポートサイトより、以下の情報を入手できます。

- 最新バージョンの SDK マニュアル／ソースコード
- アドイス／業界情報
- スター精密プリンタードライバ
- 技術的な質問／サポート

その他の印刷に関する詳細情報

Android Developers Resource
Android の公式開発リソース

System Requirements for Android Development
Windows、Linux、Mac のシステム要件に関して

Unicode.org
ユニコードコンソーシアム – Unicode の詳細について

1D Barcodes
Barcode Island - バーコードの詳細について

2D Barcodes
QR Codes、PDF417 に関する情報について

Code Pages
コード・ページに関する情報について

ASCII コード表

10 進	16 進	文字	10 進	16 進	文字	10 進	16 進	文字	10 進	16 進	文字
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?

10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	¥	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	□

SDK マニュアル改訂履歴

Release Date	SDK Package Version	Update
Apr. 17 2015	3.14.0	- 初版発行
Aug. 31 2015	3.15.0	- mPOP 対応追加

Star Micronics is a global leader in the manufacturing of small printers. We apply over 50 years of knowhow and innovation to provide elite printing solutions that are rich in stellar reliability and industry-respected features. Offering a diverse line of Thermal, Hybrid, Mobile, Kiosk and Impact Dot Matrix printers, we are obsessed with exceeding the demands of our valued customers every day.

We have a long history of implementations into Retail, Point of Sale, Hospitality, Restaurants and Kitchens, Kiosks and Digital Signage, Gaming and Lottery, ATMs, Ticketing, Labeling, Salons and Spas, Banking and Credit Unions, Medical, Law Enforcement, Payment Processing, and more!

High Quality POS Receipts, Interactive Coupons with Triggers, Logo Printing for Branding, Advanced Drivers for Windows, Mac and Linux, Complete SDK Packages, Android, iOS, Blackberry Printing Support, OPOS, JavaPOS, POS for .NET, Eco-Friendly Paper and Power Savings with Reporting Utility, ENERGY STAR, MSR Reading, *future*PRNT, StarPRNT… How can Star help you fulfill the needs of your application?

Don't just settle on hardware that won't work as hard as you do. Demand everything from your printer. Demand a Star!

Star Micronics Worldwide

Star Micronics Co., Ltd.
536 Nanatsushinya
Shimizu-ku, Shizuoka 424-0066
Japan
+81-54-347-2163
http://www.star-m.jp/eng/index.htm

Star Micronics America, Inc.
65 Clyde Road. Suite G
Somerset, NJ 08873
USA
1-848-216-3300
http://www.starmicronics.com

Star Micronics EMEA
Star House
Peregrine Business Park, Gomm Road
High Wycombe, Buckinghamshire HP13 7DL
UK
+44-(0)-1494-471111
http://www.star-emea.com

Star Micronics Southeast Asia Co., Ltd.
Room 2902C. 29th Fl. United Center Bldg.
323 Silom Road, Silom Bangrak,
Bangkok 10500
Thailand
+66-2-631-1161 x 2
http://www.starmicronics.co.th/