Android Software Development Kit

How to Use StarIO for Star Printers (StarPRNT)

This SDK contains a Java Eclipse project for Android devices.

Supported Versions	Supported Star Interfaces	Compatible StarIO JAR File
Android OS 3.1 and Higher	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

Tools Needed:

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

❖Portable Printer

When using StarPRNT emulation:

To use the StarPRNT emulation, set the emulation setting of the printer to "Star Line Mode". To change the emulation, proceed as follows. For SM-L200, it does not need to switch the emulation.

◆ **Switching over between ESC/POS and StarPRNT emulations**

1. Turn the printer power and open the printer cover.
2. Press and hold the POWER button and the FEED button simultaneously. As soon as the ERROR lamp flashes five times, release the buttons. The emulation switchover takes place automatically.
3. After setting a paper, close the printer cover. The set emulation is printed out.
   ESC/POS : EMU = ESC/POS Mode
   StarPRNT : EMU = Star Line Mode

   If the emulation is not switched correctly, repeat the above steps 1 to 3.
   At that time, in step 2, make sure not to release the buttons until the lamp completes the 5th flash.
4. Please reboot the printer after switching the emulation.
   *It will be valid after rebooting the printer.

Table of Contents

- About this Manual
- Star Printer Compatibility Chart
- Connecting a Star Printer to an Android Device
- Configuring an Android Device for Development
- How to Import and Run the Android SDK
- Using the SDK with Star Micronics POS Printers
- Overview of how this Android SDK is designed
- StarIO - (StarIOPort3.1.jar)
- Functionality
- Tips for software application development when using StarIO
- Additional Resources
- ASCII Table
- SDK Version History

About this Manual

This manual is designed to help you understand StarIO and how to build an Android application to interact with Star Micronics Printers. It is important to understand the basics of the Java language. Although this SDK is for Android, there are SDKs available for many different operating systems and programming languages at our Global Support Site. Check the Developers Section of our site for the newest SDKs, technical documentation, FAQs, and many more additional resources.

Key Legend:

Warning		Explains potential issues
Avoid Doing This		Explains things not to do
Note		Provides important information and tips

CAUTION:

- Android is a trademark of Google Inc.
- The information in this manual is subject to change without notice.
- STAR MICRONICS CO., LTD. has taken every measure to provide accurate information, but assumes no liability for errors or omissions.
- STAR MICRONICS CO., LTD. is not liable for any damages resulting from the use of information contained in this manual.
- Reproduction in whole or in part is prohibited.

©2012-2015 Star Micronics Co., Ltd.

Star Printer Compatibility Chart

The below chart summarizes the Star Printer Models supported on Android Operating Systems.

Android OS Versions 3.1 and higher support USB, Bluetooth and Network Interfaces.
Android OS Versions 3.0 and lower support Bluetooth and Network Interfaces.

◆Star Printer Models supported on Android Operating Systems

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 model only)	Bluetooth	✓	✓	✓	
SM-S220i * (Models for US and Europe only)	Bluetooth	✓	✓	✓	
SM-S230i (Models for US and Europe only)	USB			✓	
	Bluetooth	✓	✓	✓	
SM-T300i *	Bluetooth	✓	✓	✓	
SM-T400i *	Bluetooth	✓	✓	✓	
SM-L200	USB **			✓	
	Bluetooth	✓	✓	✓	
mPOP	USB				✓
	Bluetooth				✓

* Frmware version 3.0 or later is required.

** USB communication will be supported from frmware version 2.0.

◆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 model only)	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-S220i *1 (Models for US and Europe only)	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-S230i (Models for US and Europe only)	USB	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓		✓
	Bluetooth	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-T300i *1	Ethernet	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-T400i *1	Ethernet	✓	✓	✓	✓*3	✓	✓	✓	✓	✓	✓	✓	
SM-L200	USB *2	✓	✓	✓	✓*4	✓	✓*5	✓	✓	✓	✓*6		✓
	Bluetooth	✓	✓	✓	✓*4	✓	✓*5	✓	✓	✓	✓*6	✓	
mPOP	USB	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓
	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓			✓

✓ : Line Mode only

✓ : Raster Mode only

*1 Firmware version 3.0 or later is required.

*2 USB communication will be supported from firmware version 2.0.

*3 When using Line Mode commands, printing in Russian and Simplified Chinese is not supported.

*4 When using Line Mode commands, Simplified Chinese will be supported from firmware version 1.1.

*5 PDF417 will be supported from firmware version 1.1.

*6 MSR capability will be supported from firmware version 2.0.

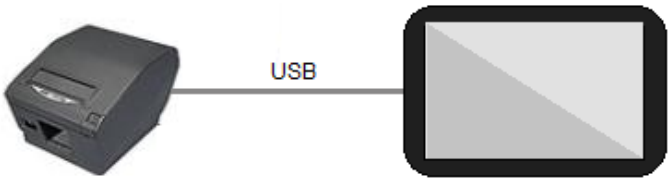
Note: This SDK offers the most popular features, but not all printer functionality has been included. The commands not included in this SDK application are available in the StarPRNT Programming Manual.

Connecting a Star Printer to an Android Device

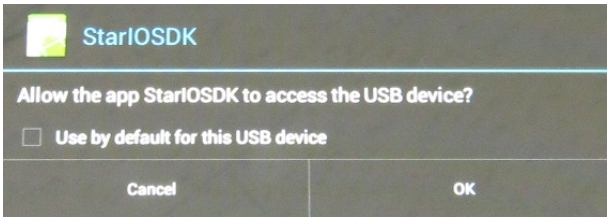
These screenshots were captured using an Android 4.0 tablet. Screenshots and wording can vary between Android operating systems.

USB Interface

1. The Android device must be running OS Version 3.1 or higher. (Versions 3.0 and lower are incompatible with USB tethering to external devices like printers.)
2. No specific Star Micronics driver or printer software needs to be installed on the Android device.
3. The USB cable needed can vary by tablet. Most Android tablets do not support the common A to B USB cable. Some require mini/micro USB cables or adapters/docks. Review the specifications for your tablet to ensure the correct cable is being used; there is no specific pin out for Star USB Printers.
4. When connecting multiple devices using a USB hub, the recommended cascade connection is maximum one stage.
   The maximum number of devices which can be connected at the same time is 4.



USB Access Permission



Depending on Android device, the above message may appear the first time you interact (get status or print) with the Star USB Printer.

To remedy this, enter the following code into the AndroidManifest.xml and device_filter.xml file.

[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>
```

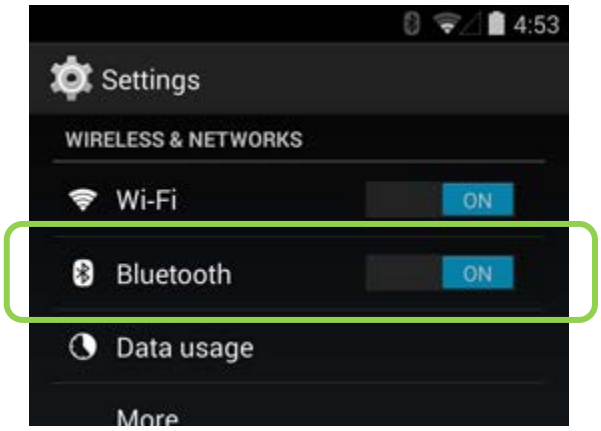
Bluetooth

All Star Bluetooth devices have each initial device name as a factory default setting, such as "Star Micronics" and "DK-AirCash". When using multiple devices which have same device name, it is useful to change the device name for identifying each device easily.

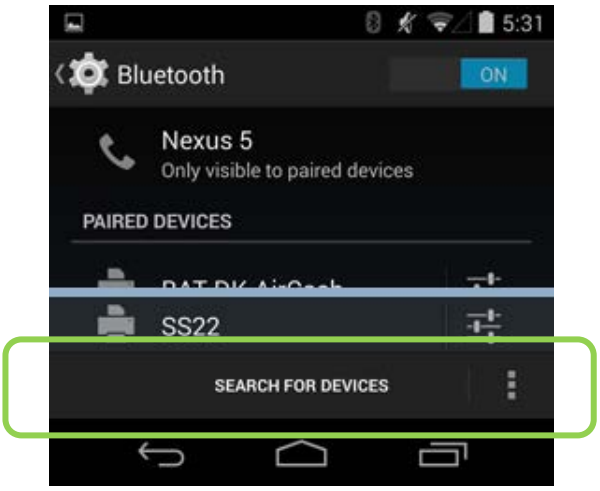
You can change the LAN/Bluetooth settings of Star POS devices such as a device name setting by using the Star Setting Utility provided by Star Micronics. Please download it from Google play.

- Pairing of a Star printer with your Android Device

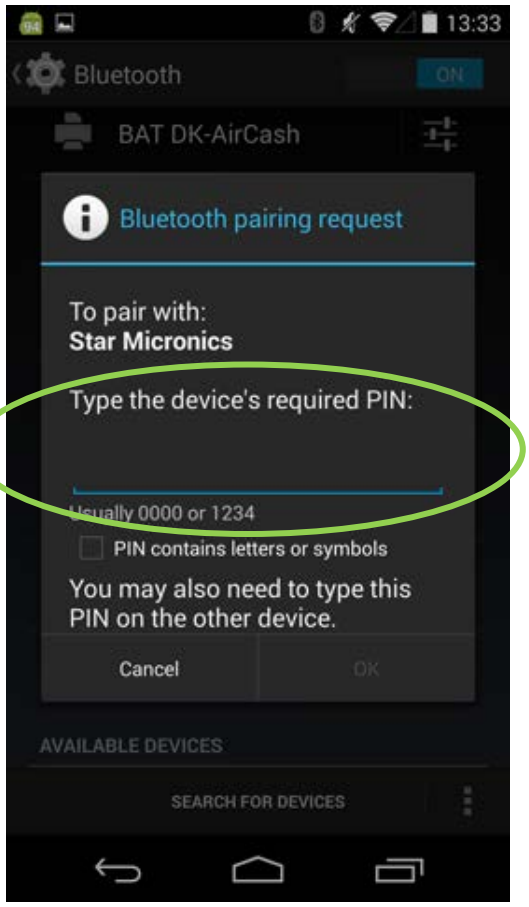
1. Ensure Bluetooth is enabled and the Star POS Device is powered on.
   When the security setting of the Star POS device is set to SSP, press the PAIR button for more than 5 seconds to make it available for pairing.
2. Tap Settings.
3. Tap Bluetooth to ON.



4. Tap Search for Devices.
   Find the Star POS Device you wish to connect to and tap it to pair.



5. If use the "PIN Code", enter the PIN number. The factory default for standard Star POS Device is "1234". (Portable Printer only)

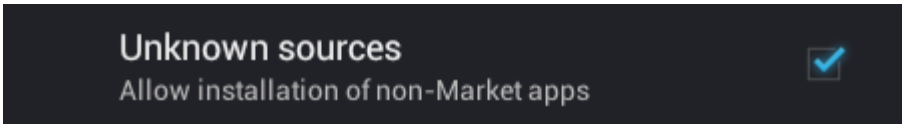


Configuring an Android Device for Development

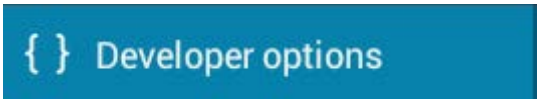
1. Tap Settings.



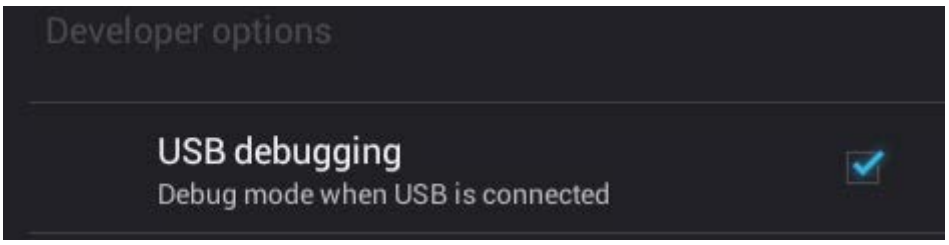
2. Tap Security.



3. Enable Unknown sources.



4. Tap Developer options.



5. Enable USB debugging.

Importing and Running the StarIO Android SDK

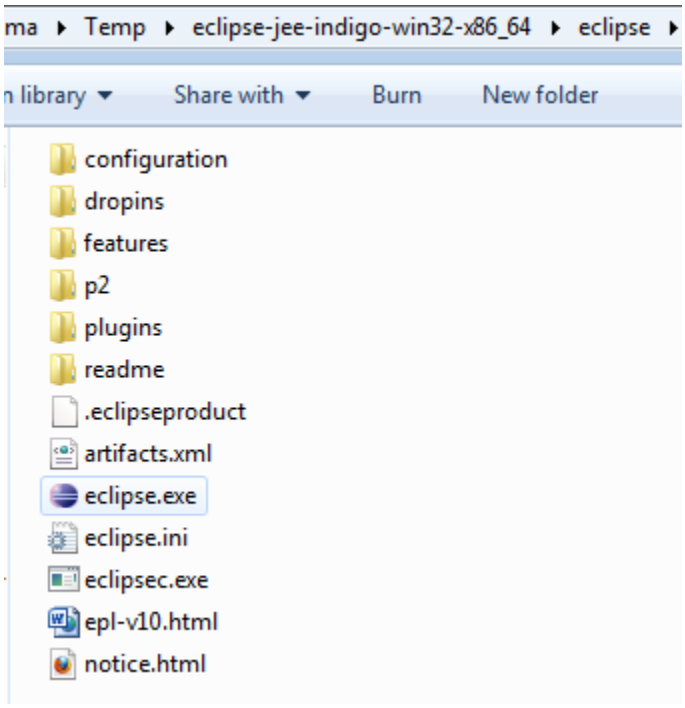
It is assumed Eclipse has already been configured to support Android development. Should you need assistance with this, refer to Star's Android Setup Guide in the documentation folder of this SDK package.

This section will explain:

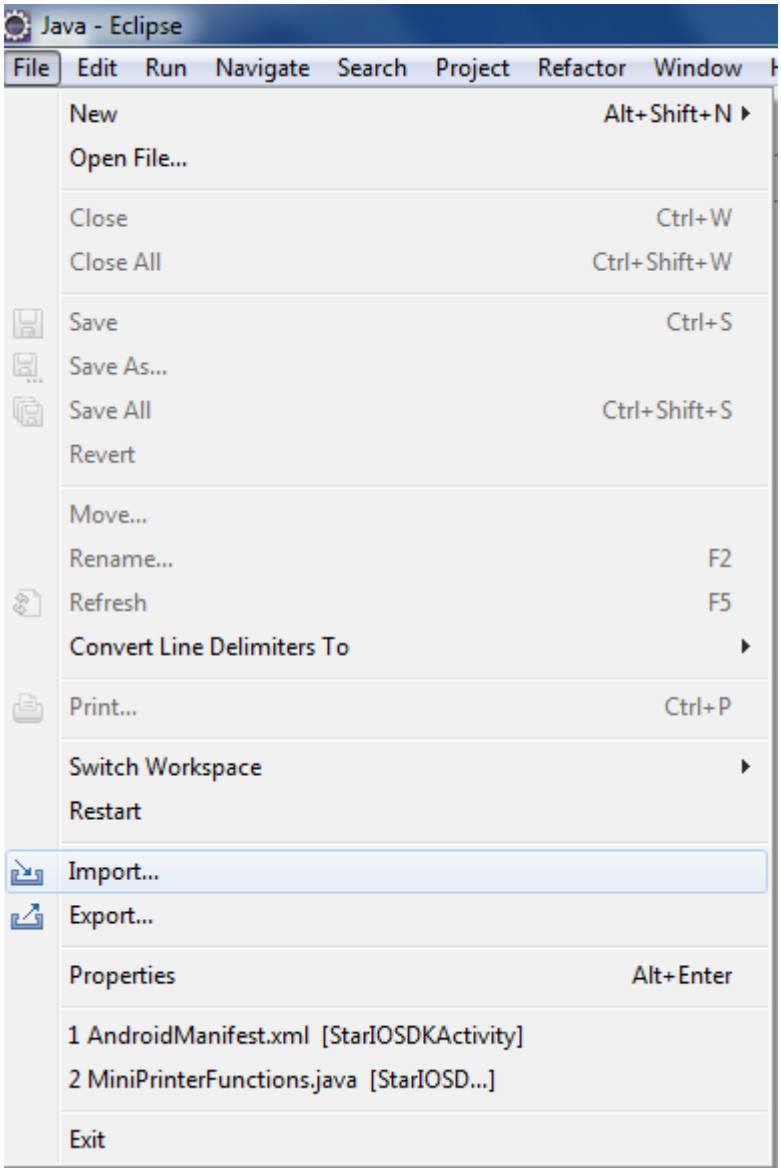
1. How to import the Android SDK project into Eclipse.
2. Enabling debug mode in Eclipse.
3. Running the project.

How to import the Android SDK project into Eclipse:

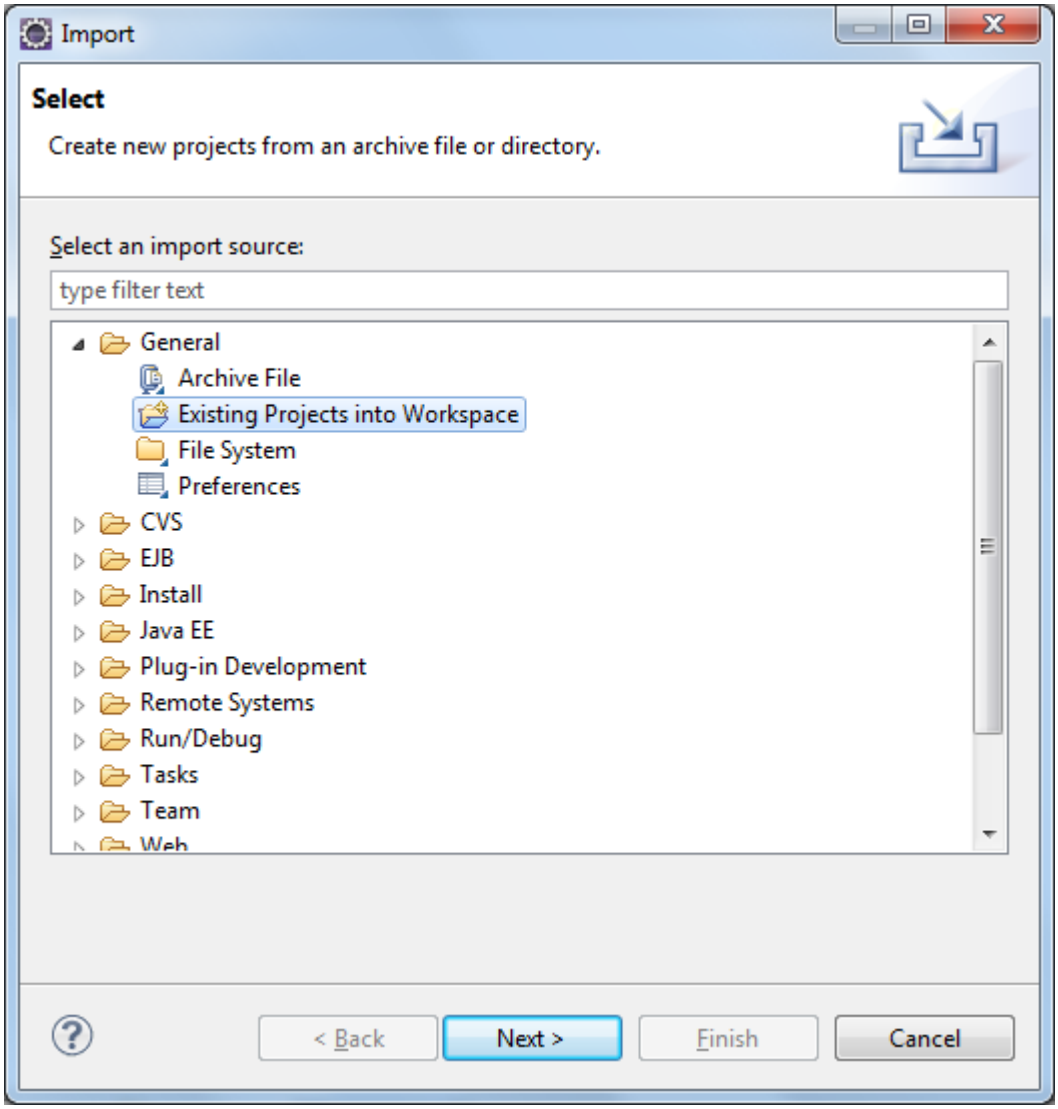
1. Execute Eclipse.



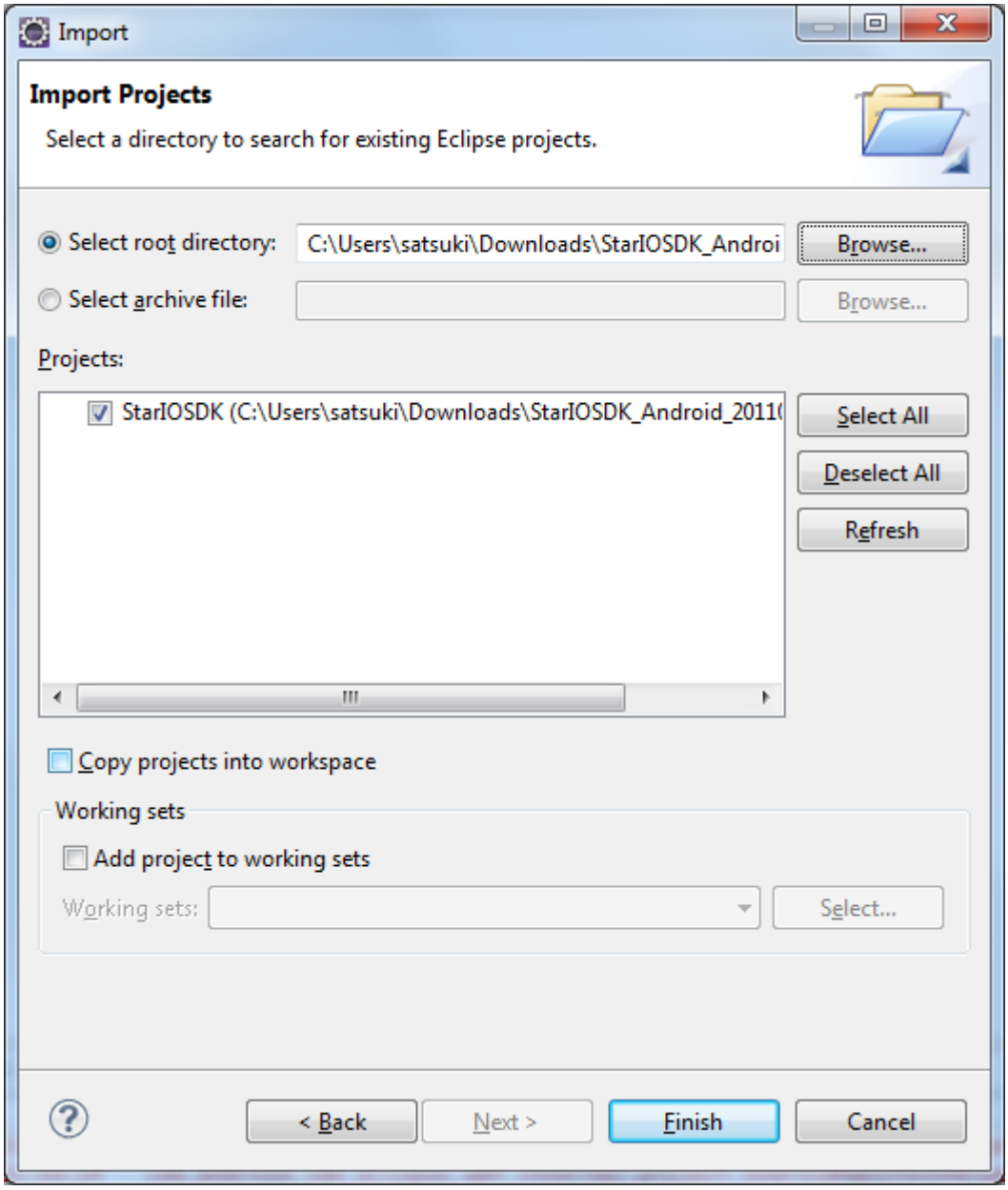
2. File > Import



3. General > Existing Projects into Workspace

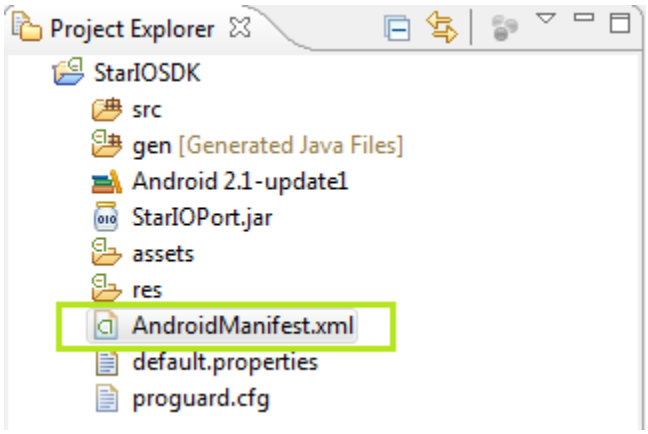


4. Browse to the location where the Star SDK is saved. Click Finish.



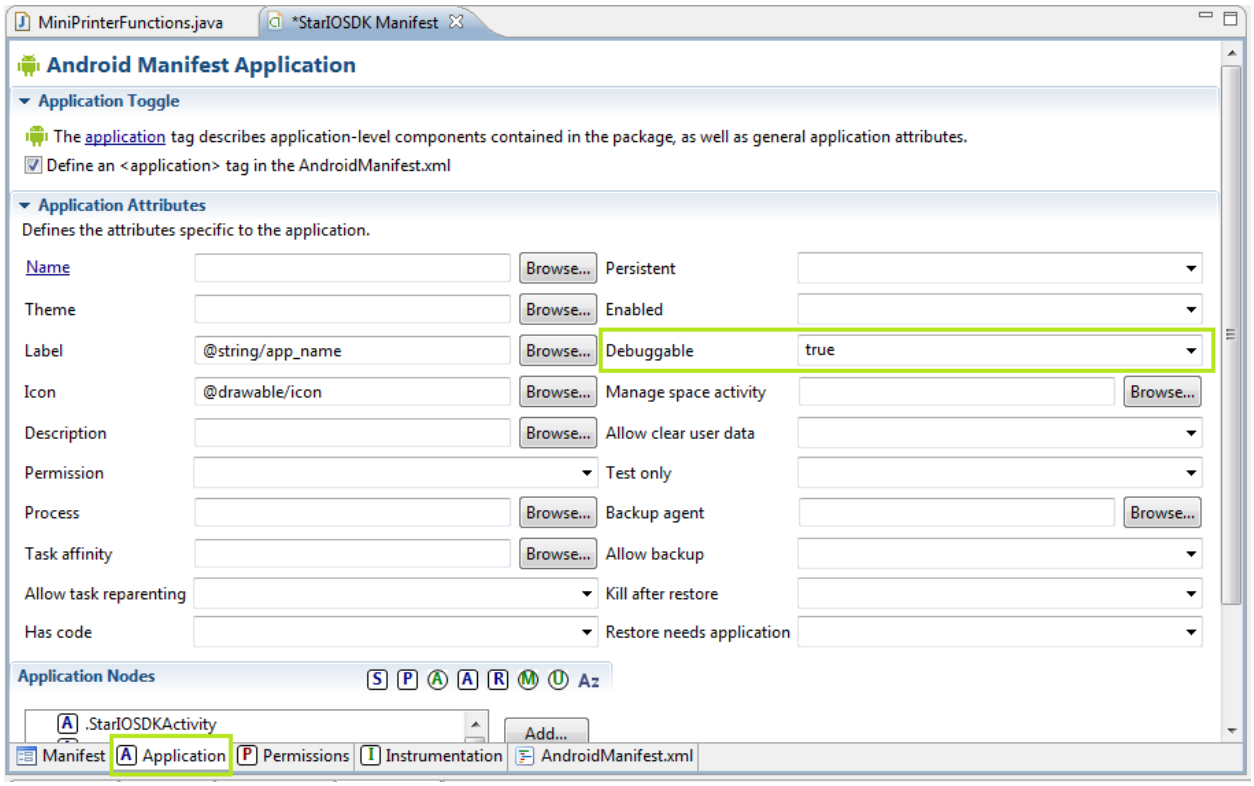
Enabling debug mode in Eclipse:

1. Click AndroidManifest.xml.



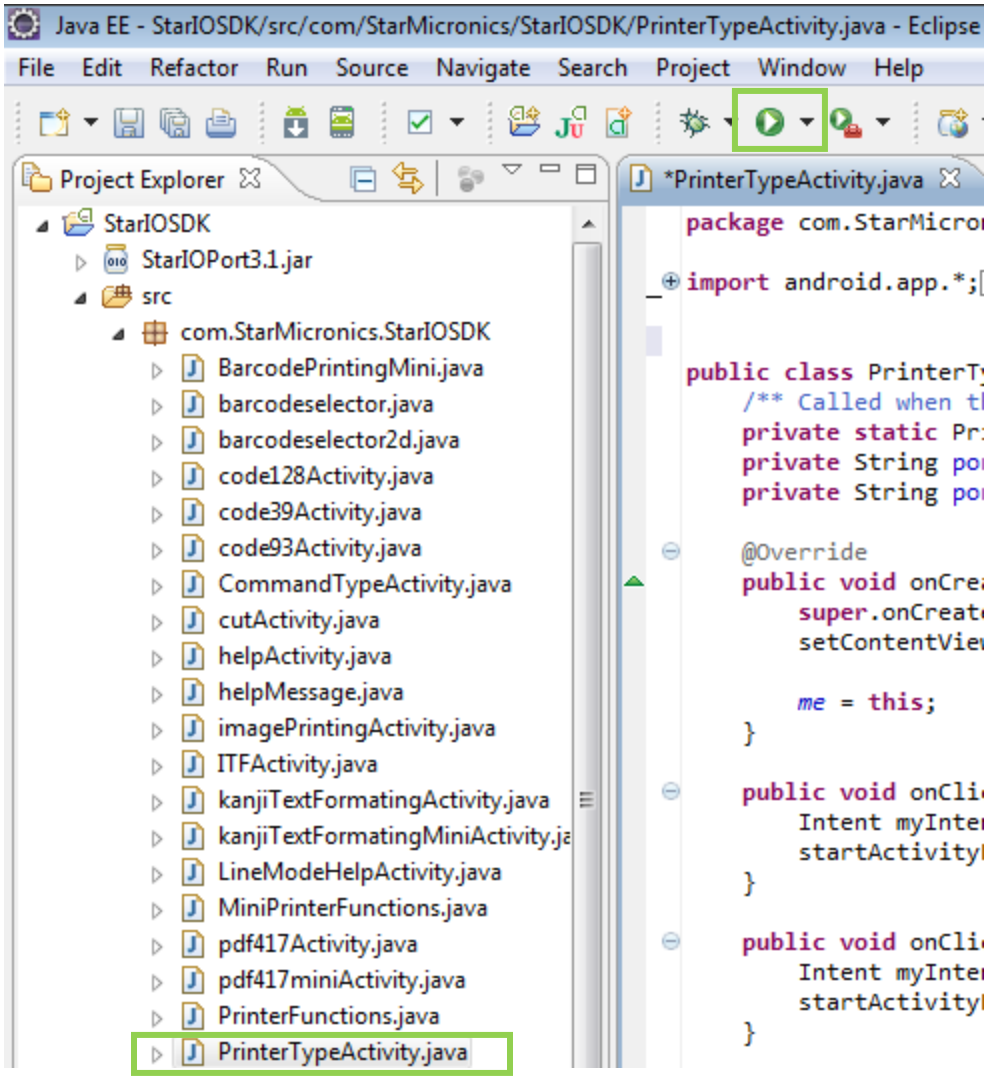
2. Click the Application tab.

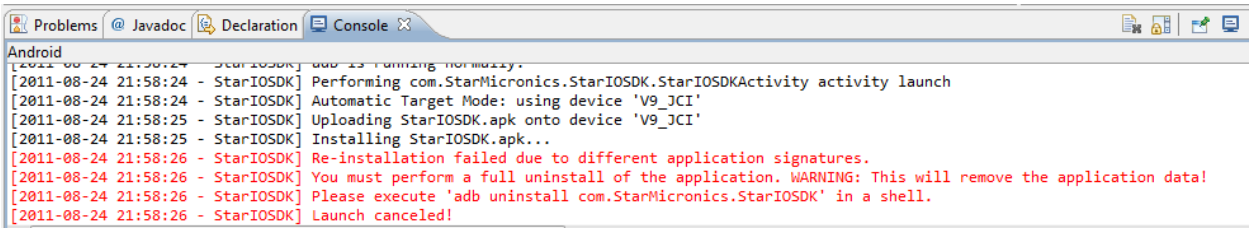
3. Set Debuggable to true.



Running the project:

1. Choose PrinterTypeActivity.java.
2. Click the green arrow.





Tip: The above error occurs if you attempt to reinstall the .apk from a different host. To remedy it, delete the application on your Android device and rerun the app.

Using the SDK with Star Printers

Please make sure you have a compatible Star Printer Model.

Port Name and Interface Relation:

StarIO uses specific port names to identify what port will be used. If these port names are not entered correctly, the application will fail to communicate with the printer. For detailed information, please see here.

Interface	Port Name
USB	USB: USB:Serial Number
Bluetooth	BT: BT:Device Name BT:Printer MAC Address

◆Notification (Portable Printer only)

In case of using Bluetooth interface with PIN code setting, when you execute getPort method, some terminals(*) require you to input PIN code again.

Even though you input it, the application fails getPort method and cannot print or cannot get status.

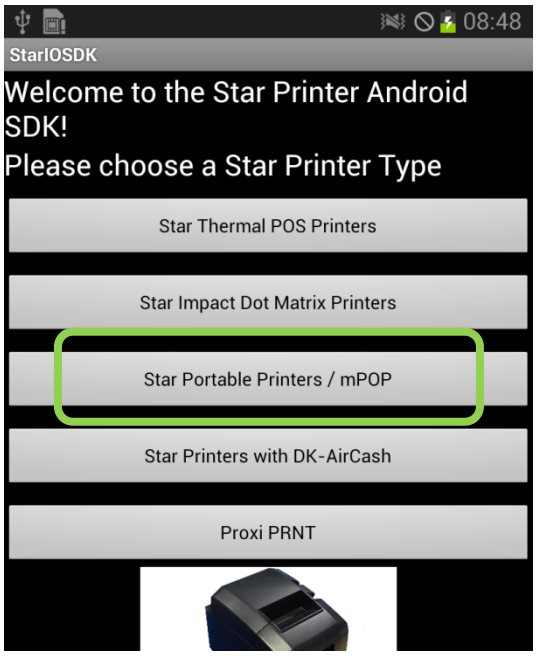
In this case, please try the following countermeasure.

Change port settings from [portSettings = "portable;p";] to [portSettings = "portable";]

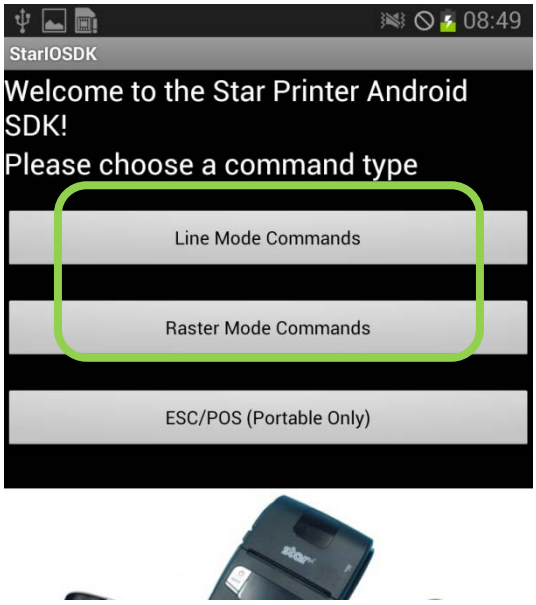
(*) We found this issue with Arrows Tab Wi-Fi FAR75A
(Android OS V.4.0.3, Japan model)

Using a POS Printer

1. Tap “Star Portable printers / mPOP”.



2. Tap the desired command type to access the samples for that mode. The mode chosen results in which samples can be sent to the printer.

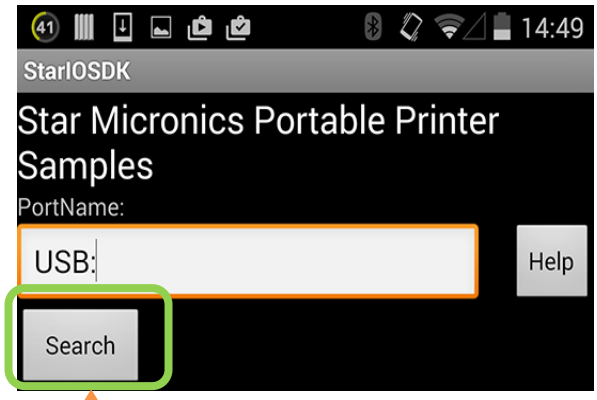


Line Mode Printers accept commands and print data line-by-line. The data is transferred to the printer in small pieces, allowing developers to customize receipt output with commands in any place they are needed. This mode alone can only make use of Device Fonts installed on the printer, which can be less visually appealing than TrueType Fonts.

Raster Mode Printers receive all print data graphically, allowing them to natively support the printing of engaging TrueType Fonts and output receipts at a lightning fast pace. Coding Raster commands is more complicated than Line Mode commands, as Raster commands require the entire receipt to be generated in graphical data before being sent to the printer.

USB Printers

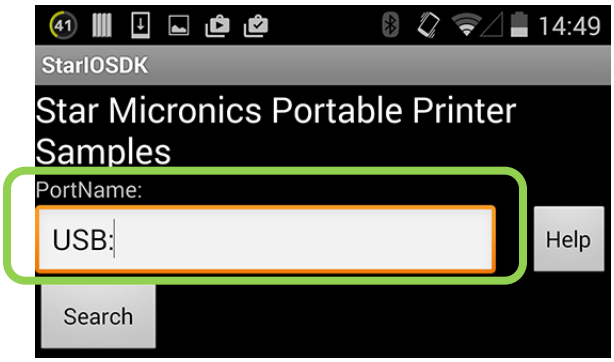
1. Tap “Search” to find all connected Star USB Printers.



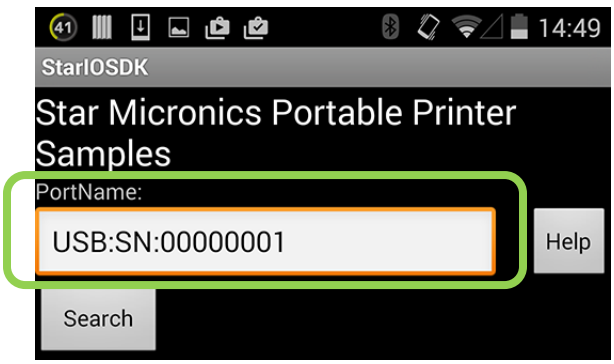
When you reconnect the USB printer, tap “Search” again.

If manually setting, proceed as follows.

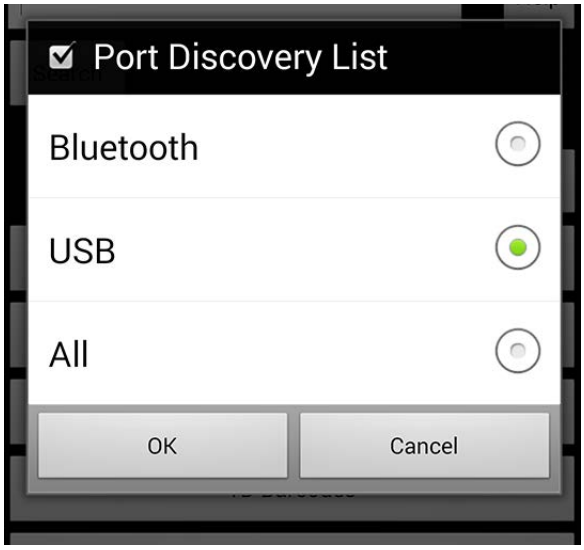
1) USB:
If only one Star USB Printer is connected, using “USB:” with no parameters will automatically communicate to the only printer connected.



2) USB:SN:<Serial Number>
Add the serial number after “USB:SN:” without brackets.

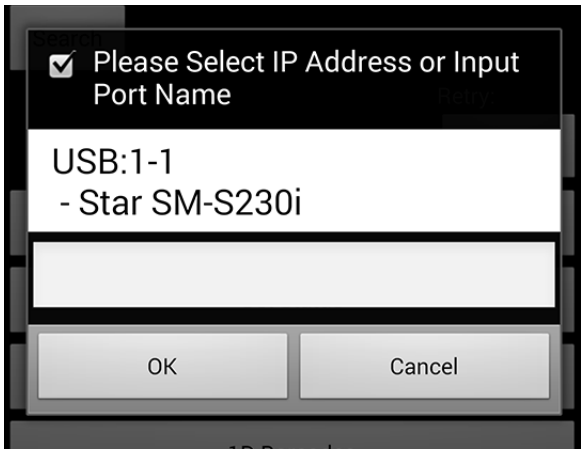


2. Tap the interface type of the printer you want to connect to.
When tapping "USB", the model names of USB printers you can connect to be displayed.



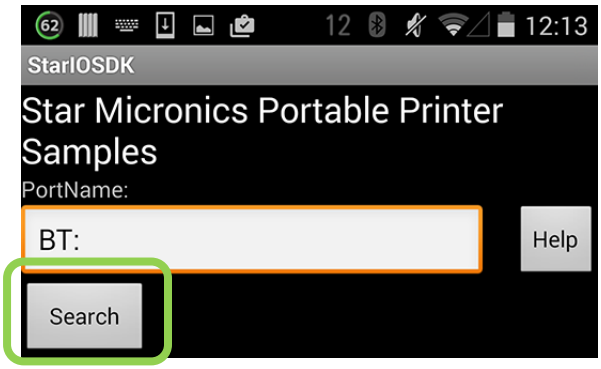
3. Tap the name of printer you want to connect to. The printer's serial number and model name are displayed at the same time.

(Some USB interface card cannot acquire the model name.)

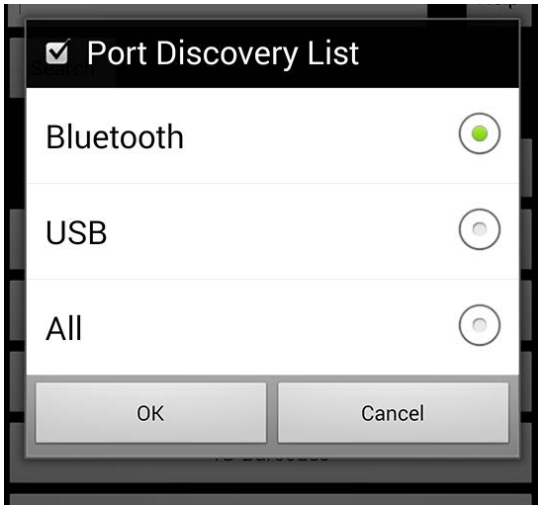


Bluetooth Printers - Port Discovery

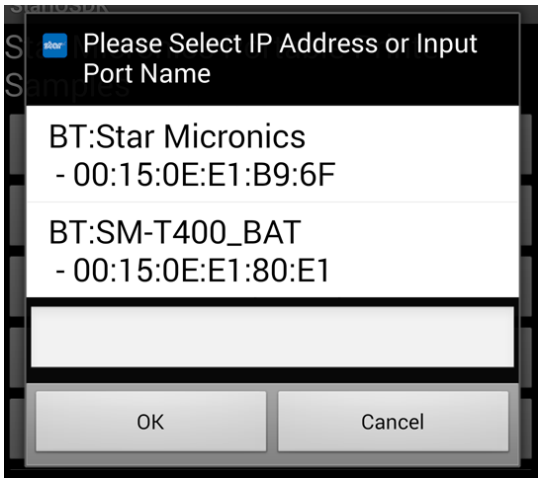
1. Tap “Search” to find all connected Star Printers. Alternatively, the device name can be manually typed into the “PortName” field. See here for details.



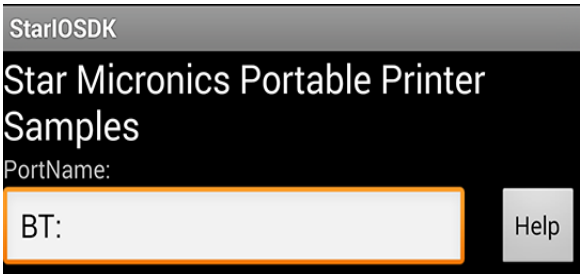
2. Tap the interface type of the printer you want to connect to.
When tapping "Bluetooth", the port names of paired printers you can connect to are displayed.



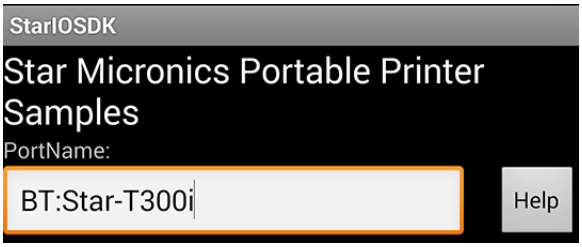
3. Tap the name of printer you want to connect to.



Bluetooth Printers – Manual Port Entry



Above: No parameters needed if using only one Star Bluetooth Printer



Above: Using the Device Name as the parameter

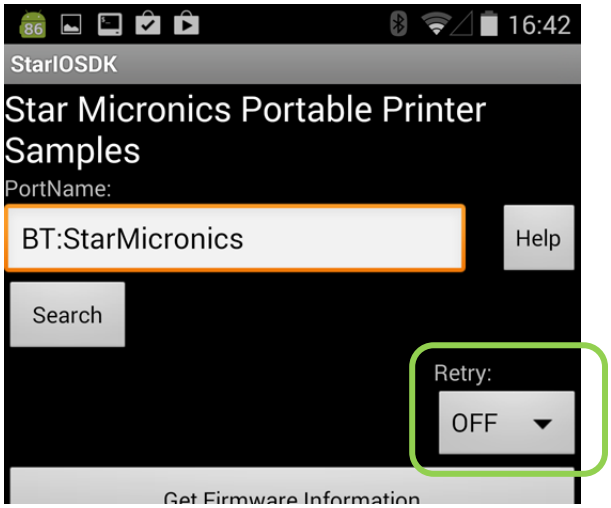
There are three types of port details that can be used for Star Bluetooth Printers:

1. BT: If only one Star Bluetooth Printer is connected, using "BT:" with no parameters will automatically communicate to the only printer connected.

2. BT:<DeviceName> Add the full device name after "BT:" without brackets. This is case sensitive so ensure it matches exactly.

3. BT:<MAC Address> Add the printer's MAC Address after "BT:" without brackets.

Bluetooth Device - Retry Setting



This setting shifts the Connection retry setting for StarIO between ON and OFF.
The retry time value is the timeout value specified in getPort method parameter.

When this setting is ON, a retry will be performed.
"portable;l" will be specified for the portSetting parameter of the getPort method.

When this setting is OFF, no retry will be performed.
"portable;" will be specified for the portSetting parameter of the getPort method.

Using an Android emulator (AVD)

The operation of SDK can be checked by using the AVD (Android Virtual Device).
It is impossible to communicate with Star Bluetooth/USB printers.

Overview of How This Android SDK is Designed

This overview will touch briefly on key components of the SDK.

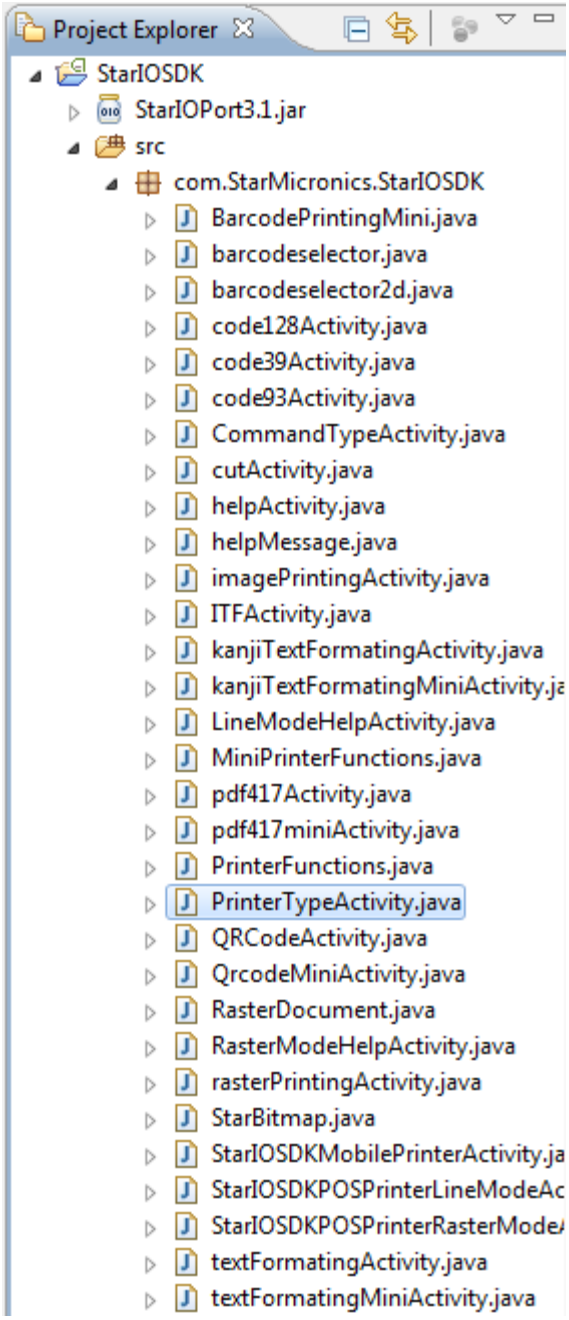
All functionality is located in the src folder in the com.StarMicronics.StarIOSDK package.

Run the program by selecting PrinterTypeActivity.java; this source code is the starting point for both POS and Mobile Printers.

See how specific functions work by clicking on the other source files. For example, “code128Activity.java” corresponds to the 1D barcode Code128 in the GUI.

It is important to note that not every function is available for both printer types. The first page of each SDK manual shows which functions are supported.

Source files containing “Mini” are sample codes for portable printer models with ESC/POS mode. StarBitmap.java applies to both printer types.



StarIO – (StarIOPort3.1.jar)

The file StarIOPort3.1.jar is a library that you can include into your Java projects to expose StarIO methods.

Selecting a StarIO Library:

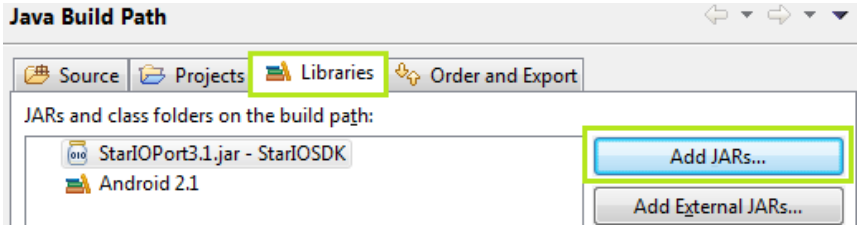
The StarIO SDK contains two library files. Use the one which corresponds to the Android OS Version your application is being developed for.

Supported Versions	Supported Star Interfaces	Compatible 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

How to include StarIO into your project:

To include this library into your project:

1. Drag the appropriate StarIO library file into the Project Explorer from the SDK package
2. Right click the project folder and choose Properties
3. Click Java Build Path
4. Click Libraries and the Add JARs button



5. At the top of your main code add:

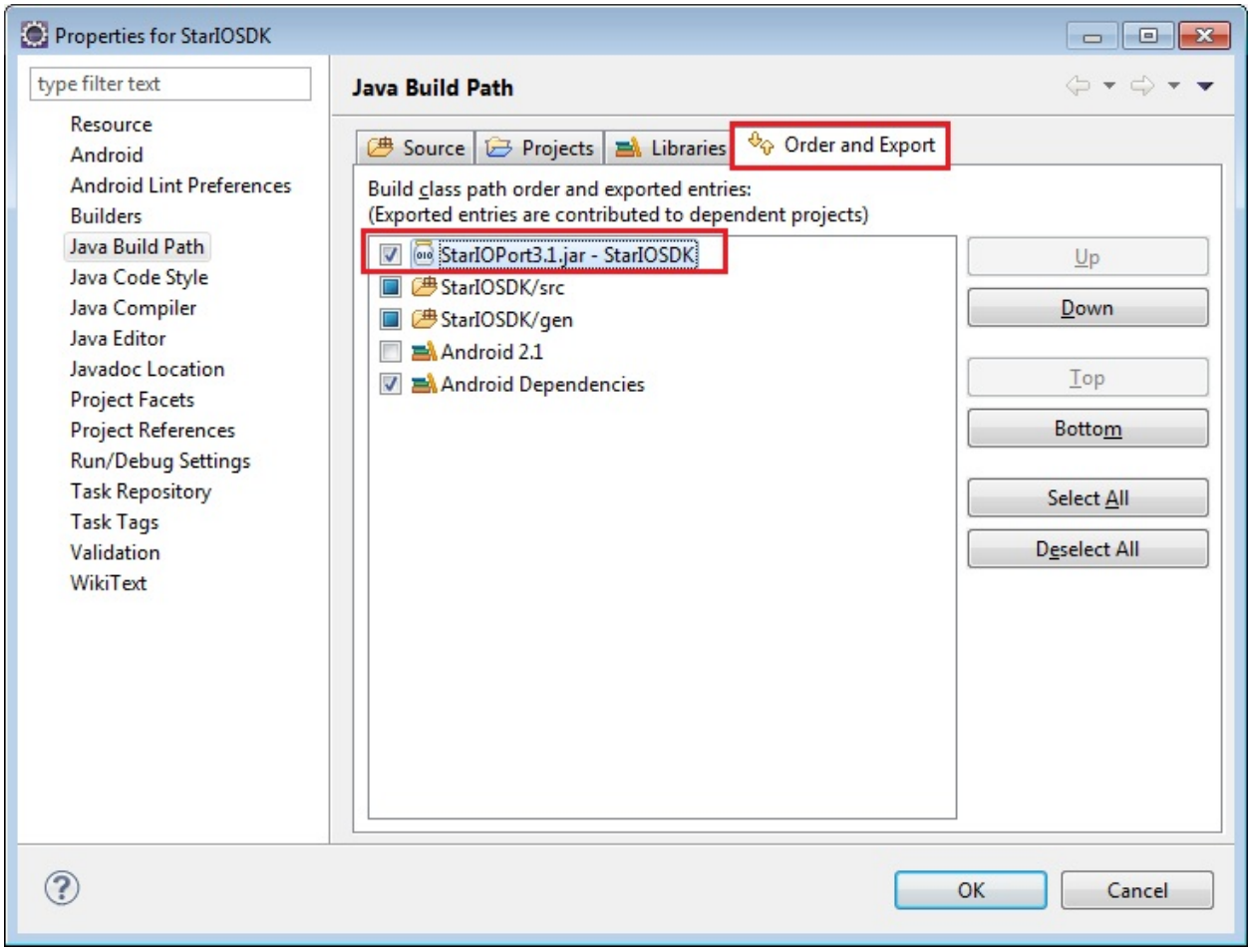
```
import com.starmicronics.stario.StarIOPort;
import com.starmicronics.stario.StarIOPortException;
import com.starmicronics.stario.StarPrinterStatus;
```

6. Now you can access all of StarIO's methods!

Android SDK Tools r17 and Higher:

Library management has been changed starting from Android SDK Tools r17. It is essential to order the StarIOPort JAR before StarIOSDK/src. This method presented no problem from Android SDK Tools r16 and lower, but caused the application to crash beginning with r17.

Star Micronics rectified this issue in the StarIO Android SDK V2.4. Please ensure the same caution is taken when using the StarIOPort JAR in your own application.



1. Right click the project name and select properties

2. Click Java Build Path and then Order and Export

3. Highlight the StarIOPort JAR and relocate it to above the application code

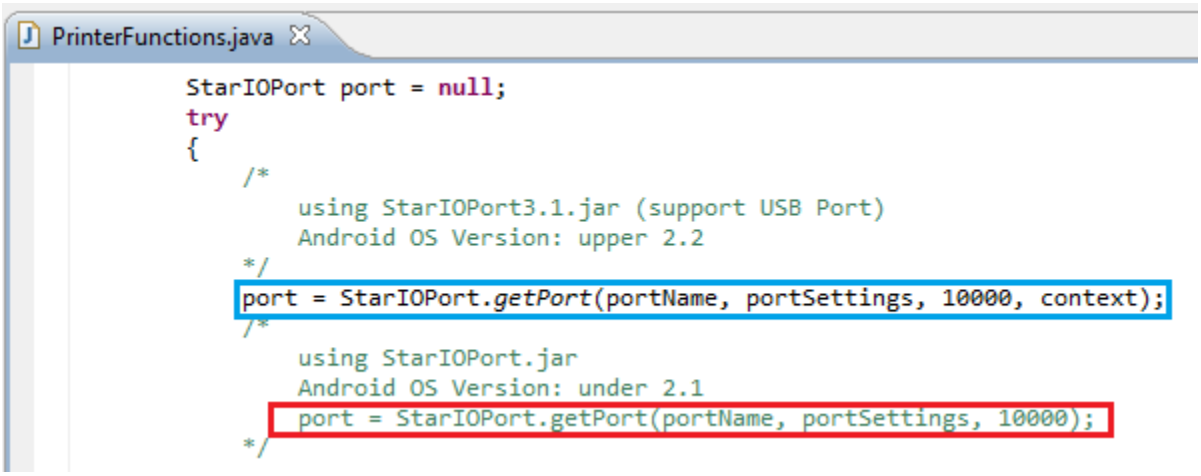
Android OS V2.2 and Higher:

Include the default library: StarIOPort3.1.jar
Upon including this library, the StarIO SDK application will successfully run. No modifications to the source code are necessary.

Android OS 3.1 and higher: Supports Bluetooth and USB connections.
Android OS 2.2, 2.3, 3.0: Supports Bluetooth onnections.

Android OS V2.1:

Include the legacy library: StarIOPort.jar
Upon including this library, the source code of the StarIO SDK application must be modified.



1. Comment this line: port = StarIOPort.getPort(portName, portSettings, 10000, context);
2. Uncomment this line: port = StarIOPort.getPort(port Name, portSettings, 10000);

This allows for the use of the legacy StarIOPort.jar library.

Android OS 2.1 supports Bluetooth connections.

StarIO Methods Overview

StarIO Port class

StarIO Port class includes 3 properties; portName (String), portSettings (String), and Timeout (int).



These 3 variables will be "read only" if accessed directly. To assign them use getPort(portName,portSettings,timeout); which will allow you to pass in variables to this methods which then assigns the 3 class variables with values.

portName is what you will be using to specify the port of communication to the printer.

Ex. "BT" / USB:

portSettings specify the following settings depending on your application.

- Use printer (Required to be specified)
- Retry setting
- Data timeout function setting

Interface	Port Name	Port Settings	
USB	USB: USB:Serial Number	portable;	Use Portable printer (Required to be specified)
Bluetooth	BT: BT:Device Name BT:Printer MAC Address	portable;	Use Portable printer (Required to be specified)
		portable;l	Retry ON
		portable;d[value]	Data timeout function* ON
		portable;u	Android OS Ver.4.0 or 4.1 is used. The security setting for pairing is set to SSP.

*** Data timeout function**

In case an error occurs while printing or no data is sent to the printer during a predetermined time, the data canceling function will be executed.
To set the time value, specify from 0 to 255 (unit: second) in [value].
This function will be invalid when 0 is specified or any appropriate value is not specified.

This function prevents next print data from being printed incorrectly when the Bluetooth connection is disconnected during data transmission.

Example of Port Setting Property :

portSettings = "portable;d127" ⋯ *In case the data canceling function is executed when no data is sent for 127 seconds during printing*

timeout is a millisecond timeout controlled internally and is used for communication in the APIs (this parameter guarantees that all of the below APIs will complete in a bounded amount of time, but does NOT guarantee the exact timeout length)



When [Data timeout function setting] is used in the portSetting parameter, it should be at least 3 seconds longer than the time specified for the data timeout function. If it is set to less than 3 seconds, the timeout time of the endCheckedBlock method will be controlled internally to be 3 seconds longer automatically.

● **Method**

getPort

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

getPort is what you will be using to "open" the port to the printer. Using one of the valid inputs for portName and portSettings as mentioned previously before this, you can pass your connection string into the StarIO class so that it will correctly set its private variables.

```
//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
}
```

`IPort` is a part of StarIO and this will allow you to create a "port" handle. The above example shows the port being created and set to null then being assigned the actual port hook on the following line that contains getPort.



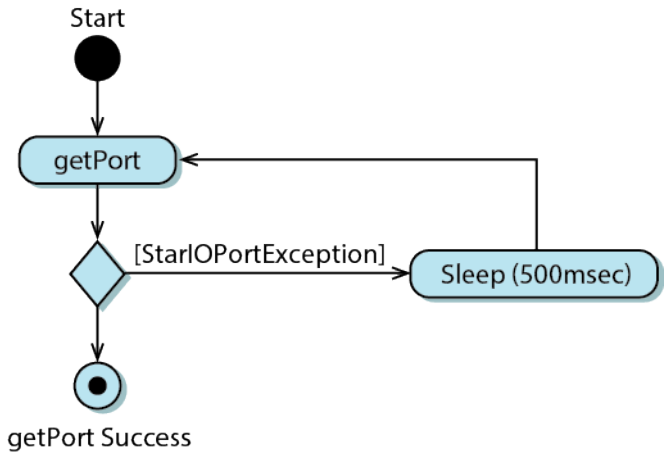
Always use a try, catch when using **getPort**. If the port cannot be opened because of connection problems, your program will crash unless you use a try, catch like the above example.

In case your application prints to bluetooth printers, the getPort method may sometimes fail to get status or print as the result of failing to connect with the bluetooth device depending on devices and a use environment.

When "portable;l" is specified for the portSetting parameter of the getPort method, a retry will be performed in case of a failure of connection with the bluetooth device.
The retry will be repeated during a time specified in the timeout parameter and the StarIOPortEX\xception will occur if a connection does succeed within a timeout time.

When getPort fails a StarIOPortException is thrown. Please repeat the operations mentioned in article 2 and 3 below (see getPort Retry Flow):

◆**getPort Retry Flow**



1. getPort fails
   (StarIOPortException is thrown).
2. Wait for certain time (e.g. 500ms)
3. Execute getPort again
4. When getPort is successful, go to procedures such as printing or getting status.

(Remark)
The number of attempts to execute getPort to obtain a port handle depends on the Host device(Android Device). Some devices have a higher failure rate than others, and may require additional attempts to getPort. Please decide based on tests conducted with your Android device.

searchPrinter

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

searchPrinter detects star printers and return search result as ArrayList<PortInfo>.

Set ”BT:” or “USB:” as Argument value target.

PortInfo class of return value includes, PortName, MAC address, Printer Model name, USB Serial Number and you can get them by String getPortName(), String getMacAddress(), String getModelName() and getUSBSerialNumber() method.

And you can use Port Name as Argument value of getPort.

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

This sample shows, get connected Bluetooth devices and output to log.



searchPrinter detects not only Star portable printers but also any Bluetooth devices. To use original name, just search specific pritners.



This API do not guarantee the discovery of devices.



When you reconnect the USB printer, tap “Search” again.

readPort

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

This method reads data from the device. Only use this if you really need to read raw bytes from the printer.



Do not use this method to read raw status.
Use RetreiveStatus for getting status.

Parameters:

`readBuffer` - A Byte Array buffer into which data is read.

`offset` - specifies where to begin writing data into the readBuffer[]

`size` - Total number of bytes to read.

Returns:

The number of bytes that were actually read. Under some interface types, this function will succeed even when no data was read in. Your application should call this function a limited number of times until the expected data has been read in or until an application determined retry threshold has been reached.

Throws:

`StarIOPortException` - when a communication failure occurs

releasePort

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

This function closes a connection to the port specified.

Parameters:

`port` – StarIOPort type representing a previously initialized port.



When do not send the data, release (close) ports.
Leaving a port open will cause future calls to open the port to fail.

writePort

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

This method writes data to the device. Use this to print to the printer, send commands, etc. The following is an example of how to use this method:

Please keep in mind this is the simplest way to send data to the printer.
The SDK has code in printToPrinter that is more complex than this but that code block will show you how to verify data transmission to the printer whereas this code is just dumping it:

```
//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
}
```

Remember to use a try, catch for safe programming practices.

Parameters:

`writeBuffer` - Contains the output data in a byte array.
`offset` - Specifies where to begin pulling data from writeBuffer .
`size` - Number of bytes to write.

Throws:

`StarIOPortException` - when a communication failure occurs

retreiveStatus

```
public StarPrinterStatus retreiveStatus ()
                                                    throws StarIOPortException
```

This method retrieves detailed status from the printer with StarIO.

Returns:

`StarPrinterStatus` structure giving the current device status

Throws:

`StarIOPortException` - when a communication failure occurs

This method uses a class structure that is included with StarIO called `StarPrinterStatus`

This structure gives the printer's status in both boolean and binary form.

Create the `StarPrinterStatus` object in your project by doing the following:

```
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.
}
```

Status List of the class structure StarPrinterStatus

Member name	Contents	Type	Detail
blackMarkError	Black Mark Error	boolean	" true " : Black mark error occurs. " false " : Black mark error does not occur. When you set printer to Black mark, and print to not Black mark paper, this error occurs.
compulsionSwitch	Compulsion SW	boolean	You can check status of CashDrawer (Open or Close) " true " : Compulsion SW is pressed. " false " : Compulsion SW is not pressed.
coverOpen	Cover Status	boolean	You can check status of Cover " true " : Cover is opened. " false " : Cover is closed.
cutterError	Auto-cutter Error	boolean	You can check status of Cutter " true " : Cutter error occurs. " false " : Cutter error does not occur.
etbAvailable	ETB available or not	boolean	" true " : available to use " false " : not available to use
etbCounter	ETB Counter	int	You can get current value of ETB
headThermistorError	Head Thermistor Error	boolean	You can check status of Head Thermistor. " true " : Head thermistor detects an abnormal value. " false " : Head thermistor does not detect an abnormal value.
offline	ONLINE/OFFLINE Status	boolean	You can check status of Online or offline. " true " : Printer is Offline. " false " : Printer is Online
overTemp	Stopped by high head temperature	boolean	" true " : Printer is stopped by head temperature. " false " : Printer is not stopped by head temperature.
presenterPaperJamError	Presenter Paper Jam Error	boolean	You can check status of Paper Jam in Presenter. " true " : Paper jam occurs in presenter . " false " : Paper jam does not occur in presenter .
presenterState	Presenter Paper Position	int	You can check status of Presenter. 0 : State where there is no paper in presenter 1 : State where paper is supplied (loop state) 3 : State where paper is discharged (Can be pulled out) 6 : State where paper is recovered 7 : State where paper is pulled out.
raw	Byte column of status	byte[63]	Byte column of status (example : HEX 23 86 00 00 00 00 00 00 00)
rawLength	raw length	int	raw length
receiptPaperEmpty	Paper end	boolean	" true " : Paper end. " false " : Paper does not end.
receiptPaperNearEmptyInner	Paper Near-end (Inner Side)	boolean	" true " : Paper near-end. " false " : Paper does not near-end.
receiveBufferOverflow	Receive Buffer Overflow	boolean	You can check status of recieved Buffer. " true " : Received buffer is full. " false " : Received buffer is not full.
unrecoverableError	Non-recoverable Error	boolean	" true " : Unrecoverable error occurs. " false " : Unrecoverable error does not occur. Unrecoverable error : Head Thermistor Error, Auto-cutter Error, Electric Voltage Error and etc.)
voltageError	Electric Voltage Error	boolean	" true " : Printers detects an abnormal power supply voltage. " false " : Printers does not detect an abnormal power supply voltage.

Class structure StarPrinterStatus Supported

Member name	SM-T300	SM-S210i *	SM-S220i **	SM-S230i **	SM-T300i	SM-T400i	mPOP
blackMarkError	✔	✔	✔	✔	✔	✔	
compulsionSwitch							✔
coverOpen	✔	✔	✔	✔	✔	✔	✔
cutterError							✔
etbAvailable	✔	✔	✔	✔	✔	✔	✔
etbCounter	✔	✔	✔	✔	✔	✔	✔
headThermistorError							✔
offline	✔	✔	✔	✔	✔	✔	✔
overTemp	✔	✔	✔	✔	✔	✔	✔
presenterPaperJamError							
presenterState							
raw	✔	✔	✔	✔	✔	✔	✔
rawLength	✔	✔	✔	✔	✔	✔	✔
receiptPaperEmpty	✔	✔	✔	✔	✔	✔	✔
receiptPaperNearEmptyInner							
receiveBufferOverflow							
unrecoverableError							✔
voltageError							✔

*** JP Model only** **** Models for US and Europe only**

beginCheckedBlock

public StarPrinterStatus **beginCheckdBlock ()** throws StarIOPortException

This method is used in combination with endCheckedBlock and checks the completion of printing. To check if the whole data is completely printed, you need to run this method just before sending print data and endCheckedBlock just after sending print data.

Returns:

`StarPrinterStatus` structure giving the current device status

Throws:

`StarIOPortException` - when a communication failure occurs
- when the printer is off line

See the sample code here.

endCheckedBlock

public StarPrinterStatus **endCheckdBlock ()** throws StarIOPortException

This method is used together with the beginCheckedBlock as a pair.

This method monitors printer status and when the transferred data is printed completely, it returns control to the application.

If this method is used with other data other than print data(ex. retrieveStatus), once that data is processed by the printer, this method returns control to the application.

If printing is not completed before the timeout (*1) or a printer error occurs during printing, StarIO throws a StarIOPortException.

(*1) To set the timeout value for endCheckedBlock execution time, please call setEndCheckedBlockTimeoutMillis method of StarIOPort object.

The default timeout value is the value specified in getPort method parameter.

Please adjust the endCheckedBlockTimeoutMillis value to be longer than printing time. If printing is not complete before endCheckedBlockTimeoutMillis value is reached, a StarIOPortException will be thrown.

The minimum timeout value that can be specified in the getPort or setEndCheckedBlockTimeoutMillis methods is 10 seconds. If less than 10 seconds is specified, StarIO will default to 10 seconds.

Returns:

`StarPrinterStatus` structure giving the current device status

Throws:

`StarIOPortException`

- An error sending the command (such as Off-Line)
- No response for the completion of printing from a printer within the timeout

See the sample code here.

```
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 vold setEndCheckedBlockTimeoutMillis ( int endCheckedBlockTimeoutMillis)
```

This method sets the endCheckedBlock method's timeout value [unit: ms]

If it takes a long time to print, please increase this value to allow for enough time to complete the print job.

Default value is the timeout value designated by getPort method.



Timeout length is 10 seconds if specified less than 10 seconds.

When [Data timeout function setting] is used in the portSetting parameter of the getPort method, it should be at least 3 seconds longer than the time specified for the data timeout function. If it is set to less than 3 seconds, it will be controlled internally to be 3 seconds longer automatically.

getFirmwareInformation

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

This method gets a model name and firmware version of the printer.
It returns Map<String, String> as an acquisition result.

Returns:

Gets a model name from the return value by setting the Key to “ModelName”.
Gets a firmware version from the return value by setting the Key to “FirmwareVersion”.

Throws:

StarIOPortException - when a communication failure occurs

Note:

・If it failed to get information, it returns an empty string.

getStarIOVersion

```
public Static synchronized String getStarIOVersion ()
```

This method gets a StarIO Version of the printer.

Returns:

StarIO Version

SMBluetoothManager Class:

SMBluetoothManager Class specifies various settings of the Bluetooth interface.
It can not be used with SMPort Class.

● Constructor

StarBluetoothManager

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

Parameters:

portName – the port name for connecting to the device.
Ex. BT:StarMicronics

portSetting - the port setting for connecting to the device.
Ex. Specify ";p" when using the PIN code with Bluetooth interface.
Specify the empty string when SSP is selected.

timeoutMillis
- timeout is a millisecond timeout controlled internally and is used for communication in the APIs (this parameter guarantees that all of the below APIs will complete in a bounded amount of time, but does NOT guarantee the exact timeout length)

starDeviceType
– the type of the device to be searched
Ex. StarDeviceTypeDesktopPrinter

Throws:

StarIOPortException – when starDeviceType is set to StarDeviceTypePortablePrinter
– when portName is not set to BT:

● Method

open

```
public void open()
```

This method is used to open connection to the star Bluetooth device.
Get the current settings by loadSetting method after conducting open method.

Throws:

`StarIOPortException`	- when a communication failure occurs

loadSetting

```
public void loadSetting()
```

This method gets the value specified from the star Bluetooth device.

Throws:

`StarIOPortException`	- when a communication failure occurs

close

```
public void close()
```

This method is used to close communication with the star Bluetooth device.

Throws:

`StarIOPortException`	- when a communication failure occurs

apply

```
public void apply()
```

StarBluetooth device is set to the value specified by the following methods.

- setBluetoothDevice
- setiOSPortName
- setAutoConnect
- setPinCode
- setSecurityType

Throws:

`StarIOPortException`	- when a communication failure occurs - when SecurityType is set to PIN code and AutoConnect is set to ON

getBluetoothDeviceName

```
public String  getBluetoothDeviceName()
```

This method gets the Bluetooth device name which was acquired by the loadSetting method or specified by the setBluetoothDeviceName method. Use this method after calling the loadSetting method or the setBluetoothDeviceName method.

Returns:
BluetoothDeviceName

setBluetoothDeviceName

```
public void  setBluetoothDeviceName( string BluetoothDeviceName)
```

This method sets the Bluetooth device name of at least 1 character, with a max length of 16 characters.

To change the Bluetooth device name, execute the apply method after the setBluetoothDeviceName method.

Parameters:
BluetoothDeviceName

Throws:

StarIOPortException	– when any invalid characters are used – when the length of the set character string is not between 1 to 16 characters.

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

getiOSPortName

```
public String  getiOSPortName()
```

This method gets the iOS port name which was acquired by the loadSetting method or specified by the setiOSPortName method. Use this method after calling the loadSetting method or the setiOSPortName method.

Returns:

iOSPortName

setiOSPortName

```
public void  setBluetoothDeviceName( string iOSPortName)
```

This method sets the iOS port name of at least 1 character, with a max length of 16 characters.

To change the iOS port name, execute the apply method after the setiOSPortName method.

Parameters:

iOSPortName

Throws:

StarIOPortException	– when any invalid characters are used – when the length of the set character string is not between 1 to 16 characters.

Valid characters:

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

getPinCode

```
public String getPinCode()
```

This method gets the PIN code which was specified by the setPinCode method. Use this method after calling the setPinCode method.

Returns:

PinCode



mPOP is not supported.

setPinCode

```
public void setPinCode( string pinCode)
```

The method sets the PIN code of 4 characters for SM-L200 and that of at least 4 character, with a max length of 16 characters for other Star Bluetooth devices.

To change the PIN code, execute the apply method after the setPinCode method.

Parameters:

PinCode *Ex.* 1234

Throws:

StarIOPortException – when any invalid characters are used
– when the length of the set character string is not between 4 to 16 characters.

Valid characters: 0-9 a-z A-Z (except for SM-L200)
0-9 (SM-L200)



mPOP is not supported.

getAutoConnect

```
public boolean getAutoConnect()
```

This method gets the value specified of Auto Connection which was acquired by the loadSetting method or specified by the setAutoConnect method. Use this method after calling the loadSetting method or the setAutoConnect method.

Returns:

true - when AutoConnection setting is ON
false - when AutoConnection setting is OFF



When using Android devices, set the AutoConnection to OFF.

setAutoConnect

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

This method sets the Auto Connection of Star Bluetooth Device.

To change the Auto Connection, execute the apply method after the setAutoConnect method.

Parameters:

autoConnect *Ex.* false



When using Android devices, set the AutoConnection to OFF.

getSecurityType

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

This method gets the Bluetooth security type(SSP or PIN Code) which was acquired by the loadSetting method or specified by the setSecurityType method. Use this method after calling the loadSetting method or the setSeurityType method.

Returns:

StarBluetoothManager.StarBluetoothSecurity

setSecurityType

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

This method sets the Bluetooth security type(SSP or PIN Code) of star Bluetooth device.

To change the Bluetooth security type, execute the apply method after the setSecurityType method.

Parameters:

securityType *Ex.* StarBluetoothManager.StarBluetoothSecurity.SSP

getPortName

```
public String  getPortName()
```

This method gets the port name specified by the StarBluetoothManager constructor.

Returns:

portName

getPortSetting

```
public String  getPortSetting()
```

This method gets the port setting specified by the StarBluetoothManager constructor.

Returns:

portSetting

getTimeoutMillis

```
public String getTimeoutMillis()
```

This method gets the TimeoutMillis specified by the StarBluetoothManager constructor.

Returns:
timeoutMillis

getDeviceType

```
public String getDeviceType()
```

This method gets the star Device type specified by the StarBluetoothManager constructor.

Returns:
StarBluetoothManager.StarDeviceType

isOpened

```
public boolean isOpened()
```

This method acquires the port status.

Returns:
true - The port is opened.

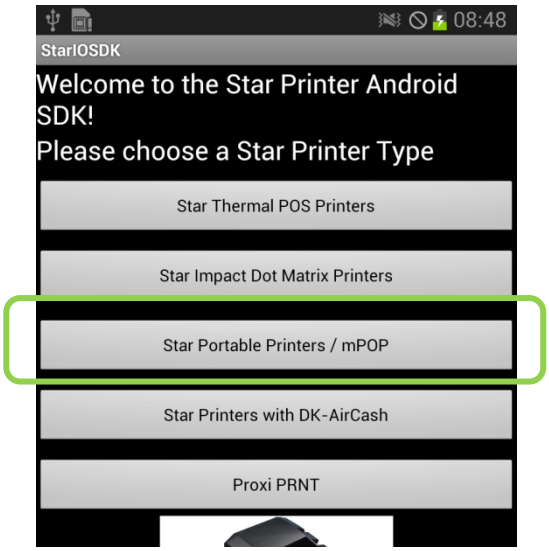
StarIO Android SDK Functionality

Overview of this SDK functionality and StarIO Printer Commands
All of these commands can be found in the Star Line Mode Command Manual.

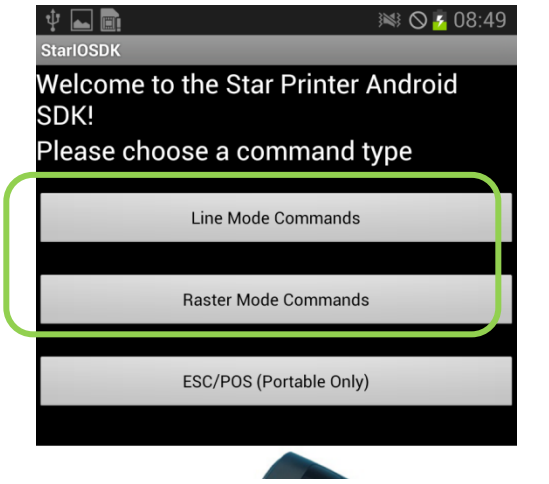
This SDK also has page and section references to the Line Mode Manual for more information so please download and study it if you need more detail on a specific command.

Choosing a Printer and Communication Type

1. Tap "Star Portable Printers / mPOP" to use a thermal printer and "Star Impact Dot Matrix Printers" to use a dot printer.



2. Select "Line Mode Commands" or "Raster Mode Commands". When a dot printer is used, the screen will jump to the corresponding Line Mode command page.
   The difference is detailed here.



Supported Samples by Command Type

Line Mode Command Samples Include:

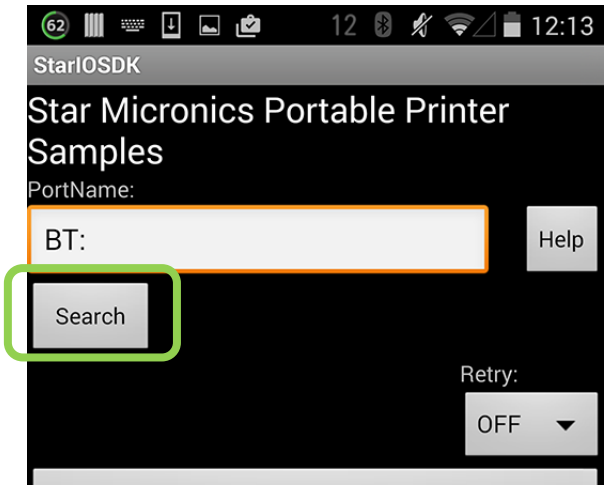
- Port Discovery
- Get Firmware Information
- Get Status
- Sample Receipt
- Open Cash Drawer
- 1D Barcodes
- 2D Barcodes
- Text Formatting
- MSR
- Bluetooth Setting
- USB Setting

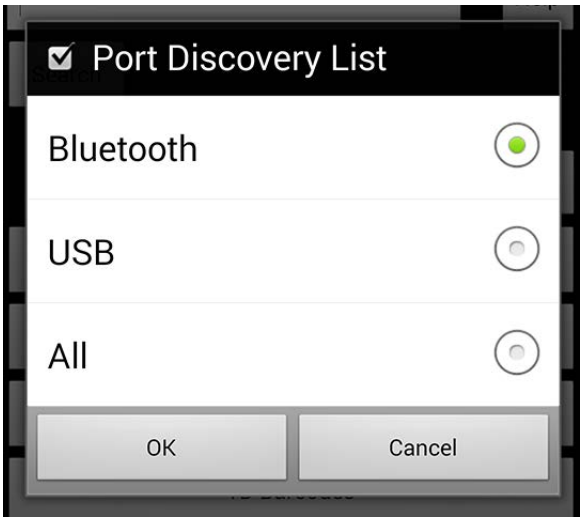
Raster Mode Command Samples Include:

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

Port Discovery

◆Line ◆Raster



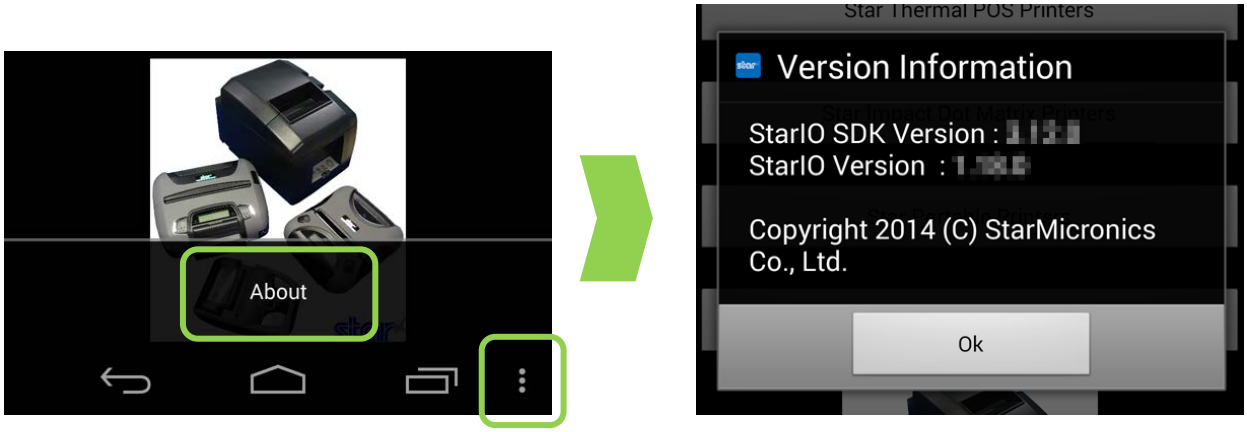


Automatically detects which Star Micronics Printers are connected to the network. Tap the printer to connect to it. [This feature is documented in greater detail here](). USB printers do not support this feature.

---

get StarIO Version

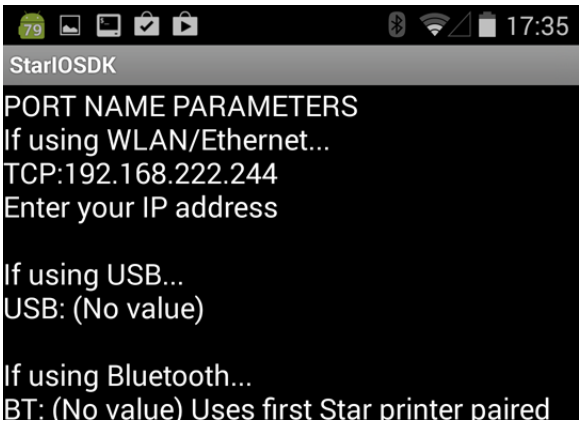
◆Line ◆Raster



This method gets a StarIO Version of the printer.

Help

◆Line ◆Raster



Displays rules for manually entering the printer's port data into the "PortName" field. Manually entering port data is unnecessary if Port Discovery is used.

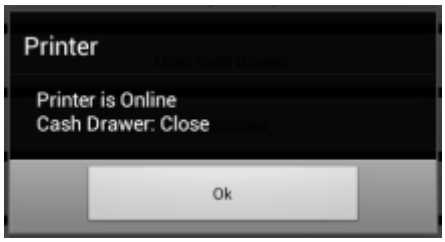
Get Firmware Information

◆Line ◆Raster



Displays firmware information of the printer specified by Port Name.

Get Status ◆Line ◆Raster

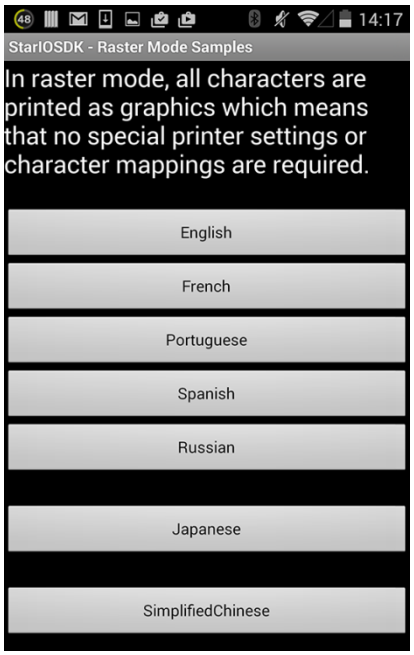


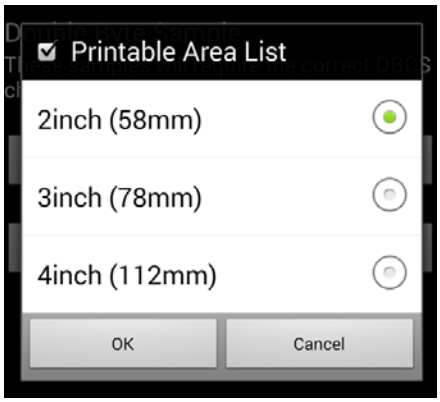
StarPrinterStatus

public boolean retreiveStatus()	See status return values here
offline	false = printer online; true = printer offline
other	See status return values here

Sample Receipt ◆Line ◆Raster

Screen transition at Raster mode:





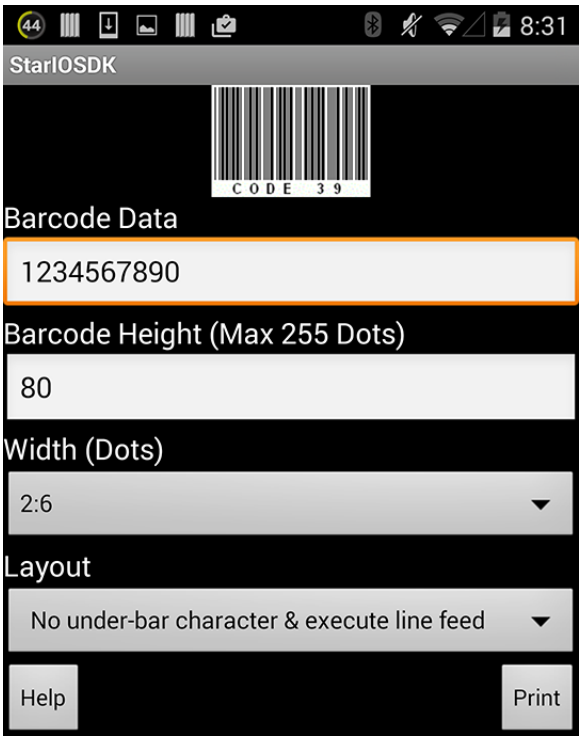
Prints a premade sample receipt in the chosen command type and the chosen language.

Select the sample's width and tap "OK" to print it. This part of the source code is heavily commented to demonstrate how receipts can be fully customized.

When using Line Mode commands, printing in Russian and Simplified Chinese is not supported with SM-210i, SM-S220i, SM-S230i, SM-T300i and SM-T400i.

When using Line Mode commands, Simplified Chinese will be supported from firmware version 1.1 for SM-L200.

1D Barcodes ◆Line



Configure 1D Barcode

ESC b n1 n2 n3 n4 d1 … dk RS

n1 = Barcode Type
0 = UPC-E * 1 = UPC-A * 2 = JAN/EAN8 * 3 = JAN/EAN13 *
4 = Code39 5 = ITF 6 = Code128 7 = Code93 8 = NW-7 *

n2 = Under-bar character selection and added line feed selection
1 = No added under-bar characters & Executes line feed after printing barcode
2 = Adds under-bar characters & Executes line feed after printing barcode
3 = No added under-bar characters & doesn't line feed after printing barcode
4 = Adds under-bar characters & doesn't line feed after printing barcode

n3 = Specifies the size of the narrow and wide barcode lines

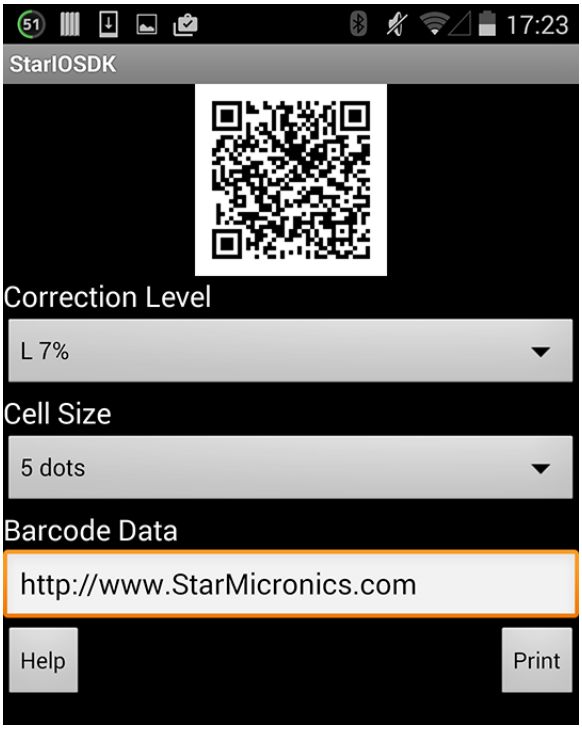
n4 = Barcode height (dot count)

* These barcodes are supported by Star Printers, but no example is in the sample application.

Note: 1D Barcode samples are not available for Raster Mode in this application. It is recommended to send barcodes graphically to the printer when using Raster Mode.

2D Barcodes ◆Line

QR Code



Select QR Code

There are 5 commands below that are very important to printing a good QR Code.

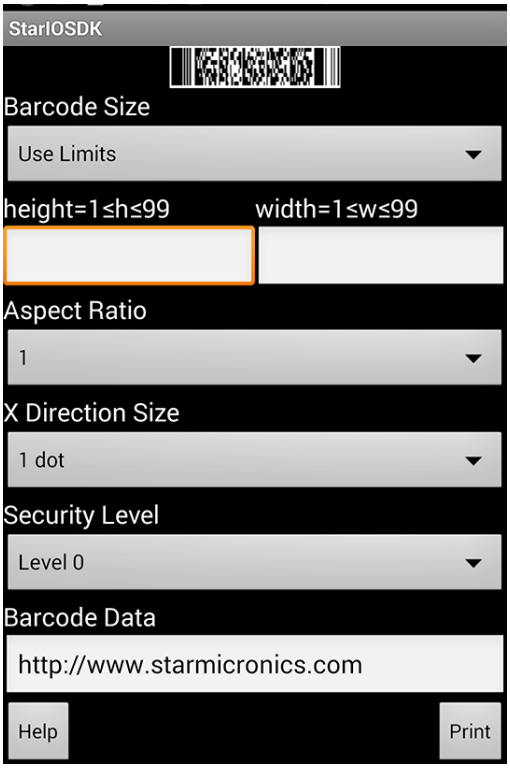
(1) Set QR Code Model #	ESC GS y S 0 n
(2) Set QR Code Correction Level	ESC GS y S 1 n
(3) Set QR Code Cell Size	ESC GS y S 2 n
(4) Set QR Code Data	ESC GS y D 1 NUL nL nH d1…dk
(5) Print the QR Code	ESC GS y P

This is the order in which commands need to be sent to print the QR Code:
QR Model + QR Correction Level + QR Cell Size + QR Data + Print QR Code

Refer to the Line Mode Programming Manual for a listing of all QR Code commands.

Note: 2D Code samples are not available for Raster Mode in this application. It is recommended to send barcodes graphically to the printer when using Raster Mode.

PDF417



Select PDF417

Please visit page 3-120 in the Line Mode Spec Manual for more details on PDF417

(1) Set PDF417 barcode size	ESC GS x S 0 n p1 p2
(2) Set PDF417 ECC (Security Level)	ESC GS x S 1 n
(3) Set PDF417 module X direction size	ESC GS x S 2 n
(4) Set PDF417 module aspect ratio	ESC GS x S 3 n
(5) Set PDF417 barcode data	ESC GS x D nL nH d1 d2 … dk
(6) Print PDF417 barcode	ESC GS x P

This is the order in which commands need to be sent to print the PDF417 barcode:

PDF417 Size + PDF417 ECC + PDF417 X-dim + PDF417 Ratio + PDF417 Data + Print PDF417

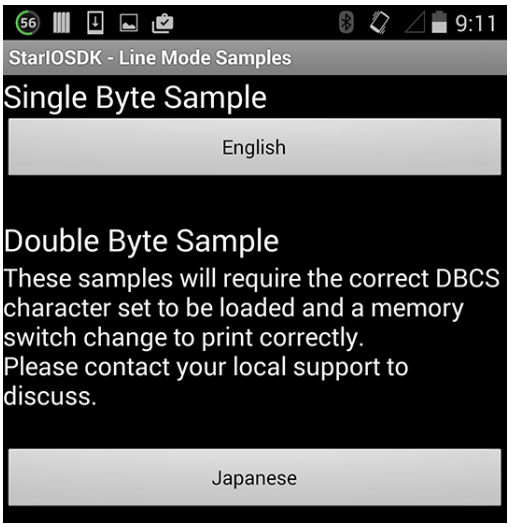
Refer to the Line Mode Programming Manual for a listing of all PDF417 commands.

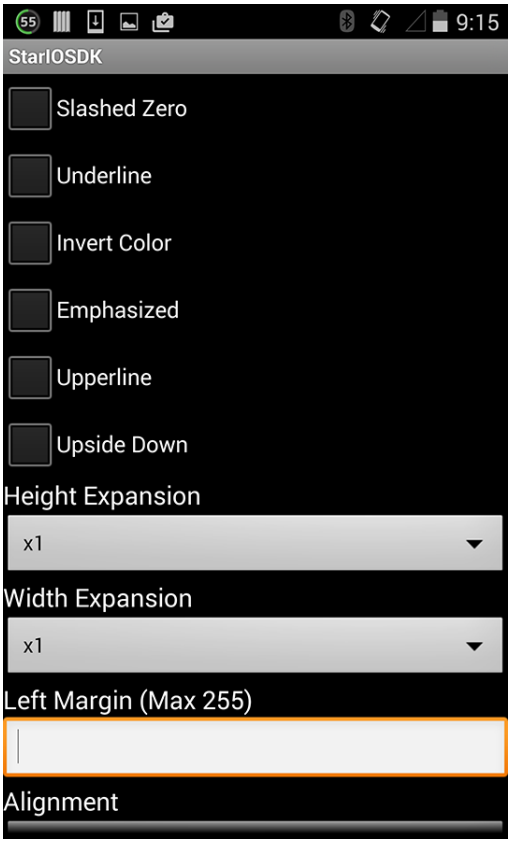
Note: 2D Code samples are not available for Raster Mode in this application. It is recommended to send barcodes graphically to the printer when using Raster Mode.
PDF417 are not supported by dot printers.

PDF417 will be supported from firmware version 1.1 for SM-L200.

Text Formatting

◆Line





*English screen

Text Formatting (continued from above)

Slashed Zero (only English)

ESC / 1 = on	ESC / 0 = off

Shift JIS Mode (only Japanese)

ESC $ 1 = on	ESC $ 0 = off [Default]

Underline

ESC – 1 = on	ESC – 0 = off [Default]

Invert Color (B/W) (Thermal printer)

ESC 4 = on	ESC 5 = off [Default]

Emphasized (Bold)

ESC E = on	ESC F = off [Default]

Upperline

ESC _ 1 = on	ESC _ 0 = off [Default]

Upside-Down

SI = on	DC2 = off [Default]

Character Expansion

Height Expansion	ESC h n	0 ≤ n ≤ 5 (except for SM-L200) 0 ≤ n ≤ 2 (SM-L200)
Width Expansion	ESC W n	0 ≤ n ≤ 5

Left Margin

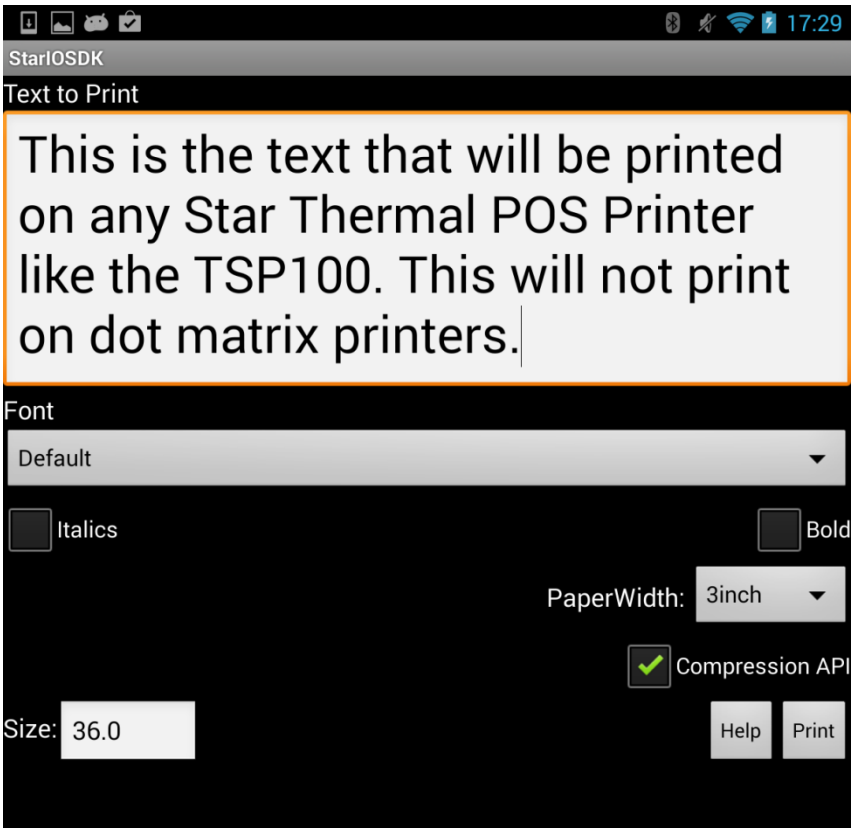
ESC l n	0 ≤ n ≤ 255

Alignment

Left [Default]	ESC GS a 0
Center	ESC GS a 1
Right	ESC GS a 2

Note: Raster Mode receives graphical data only so this sample is not compatible. For a data formatting example in this mode, refer to Raster Graphical Text Printing.

Raster Graphical Text Printing ◆Raster



Raster Mode converts all print data into image data and then outputs it to the printer. This enables Star Printers to print at high speeds, regardless of outputting receipts with only text or text and logos/coupons. As there are many options on how to customize output in Raster Mode, refer to the StarPRNT Programming Manual for a listing of all Raster commands. These commands are also conveniently listed right on the Android device by tapping the Help button on the screen.

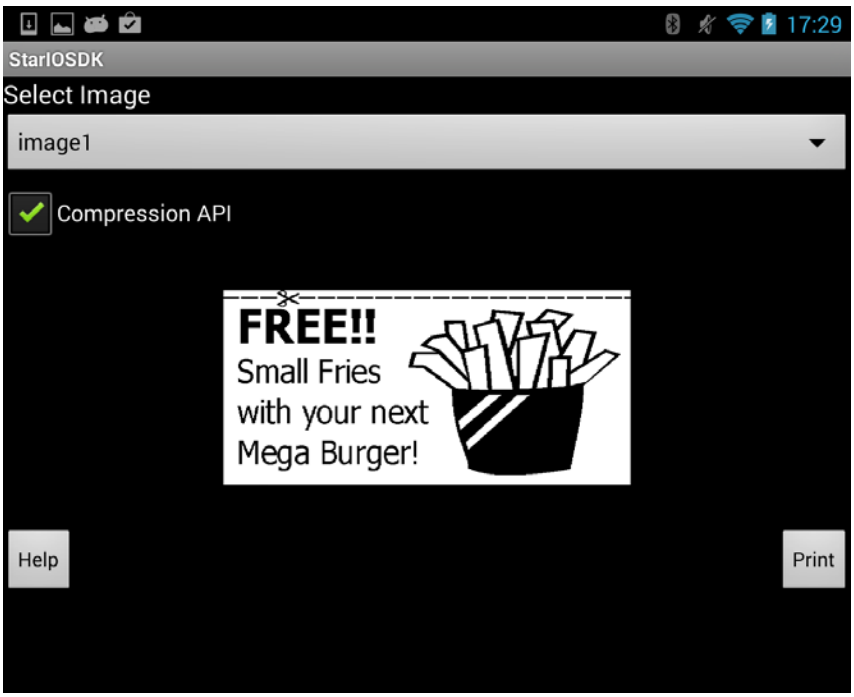
Using ”Compression API” method may improve through put.

Note1: Line Mode cannot process graphical data so this sample is not compatible. For a data formatting example in this mode, refer to Text Formatting.

Note2: If send large amounts of raster data command, use beginCheckedBlock / endCheckBlock method “before / after” sending data by writePort method for preventing “data detective”.

Detail refer to the **"printBitmap"** method in the **"PrinterFunctions.java"**.

Image File Printing ◆Raster



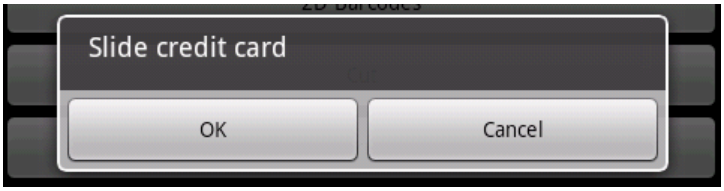
Use the dropdown box to select one of four different sample images to print via Raster Graphics. Note: The images in this sample are pre-formatted for 80mm wide receipts. If the printer in use is smaller or wider than 80mm, the image will not be automatically scaled.

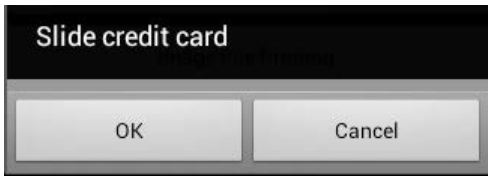
Raster Mode converts all print data into image data and then outputs it to the printer. This enables Star Printers to print at high speeds, regardless of outputting receipts with only text or text and logos/coupons. As there are many options on how to customize output in Raster Mode, refer to the StarPRNT Programming Manual for a listing of all Raster commands. These commands are also conveniently listed right on the Android device by tapping the Help button on the screen.

Using ”Compression API” method may improve through put.

Note: This sample is not available in Line Mode.

MSR (Magnetic Stripe Reading)





Read One Track

ESC M C	Read Track 1 or Read Track 2

Read One Track

ESC M D	Read Track 2 or Read Track 3

Read Two Tracks

ESC M E	Read Track 1&2 or Read Tracks 2&3

Set JIS-II Mode

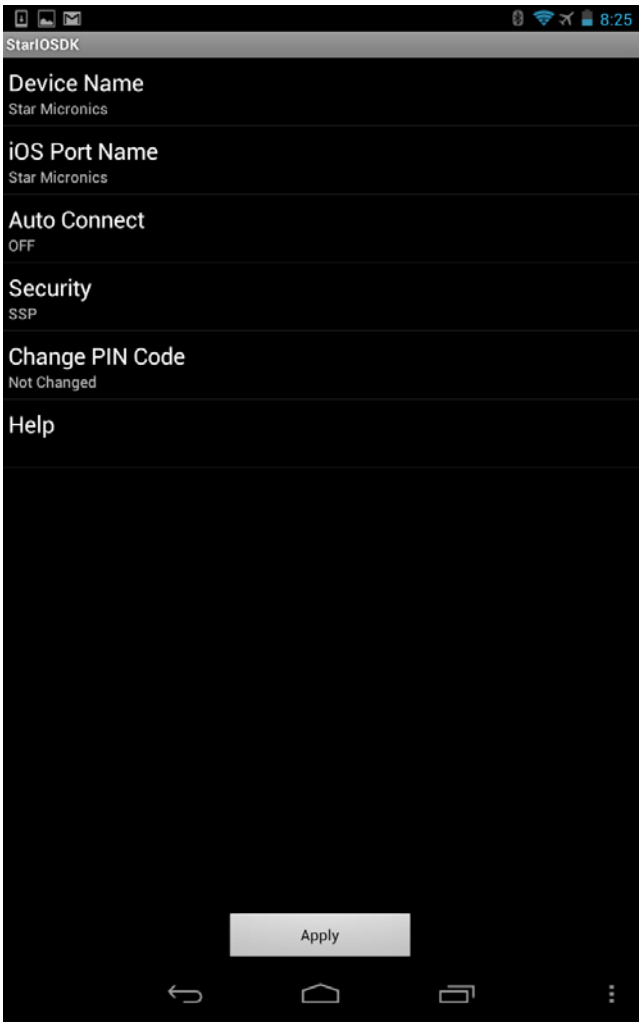
ESC M J	Read Japanese Characters

Cancel MSR Mode

EOT	Return to standard print mode

Bluetooth Setting

◆Line ◆Raster

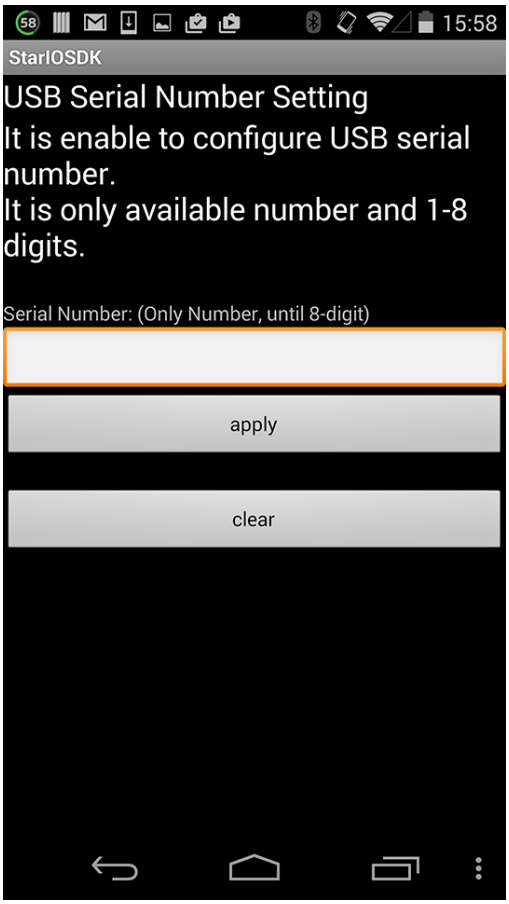


Connects to the Bluetooth device which is specified for PortName and changes various settings of the Bluetooth interface.

The values applied with this method are effective after turning the device off and on and paring again.

USB Setting

◆Line ◆Raster



Connects to the USB device which is specified for PortName and changes various settings of the USB interface.

The values applied with this method are effective after turning the device off and on.

Tips for App Development when using StarIO

Star Micronics prides itself as the industry leader in great POS products and with great power comes great responsibility. Below is a tips section just to help you get on the fast track to software development with StarIO.

TIP #1: If you are going to be coding a large project, create a class to abstract all the printing methods into class(s) instead of having the code reside in the main code block. This will help with code reusability and will also save you time in the long run from having to find one line of code in the main code. By having StarIO only reside in the class(s), you will be fully taking advantage of object oriented programming.

TIP #2: Know what the differences and definitions of (ASCII & Unicode), (Hex & Decimal), and (Byte & Char) are. A byte is normally 8-bits long which would be 8 digits of binary (1s and 0s). These bytes are just 8 bits of binary data but bytes can also be int or char. The three different variable types basically hold the data in the same way but there are slight differences. Try to code with Bytes instead of Chars, ints, or strings when choosing a variable to contain your print job data. ASCII to Unicode and vice versa conversions are sometimes unsecure so make sure you know what and how the encoding class works with these. Big mistakes made in Unicode are culture-sensitive search and casing, surrogate pairs, combining characters, and normalization.

TIP #3: HEX DUMP MODE! If you are debugging and your application seems to have a bug in it use hex dump mode on the printer. This is the best way to verify what is being sent out of the computer is being received correctly. To put the printer in hex dump mode, turn the printer off, open the cover to the paper, hold the feed button down, turn the printer back on, close the cover, let go of the feed button. Hex dump mode is a sure fire way to verify hex data is sent correctly. When in hex dump mode, printer functions will not work.

TIP #4: Do not waste time trying to reverse engineer StarIO command codes. All the available StarIO commands are available in the Thermal Line Mode Spec Manual and that is the best resource to use when researching a specific StarIO command. This SDK & Manual was built to help you (The Developer) have a very easy job ahead of you to program for Star Printers.

TIP #5: If there is a command that is not covered in this SDK but you wish to see a code snippet of that command in use then visit our Developers section for a possible code block that matches your needs.

TIP #6: Looking for an iOS printing SDK? Visit our Developers section to get access to Star developer tools for these environments.

Additional Resources

This section will share resources that will help you develop good software with StarIO.

Please get the programmers manual for Star Portable Printers from the link below.

Star Micronics Developers Network

Browse Star Micronics' FAQs, ask a question, look up information, etc.

The Developers Network gets you access to:

- Updated Versions of this Manual and Source Code
- Getting Started Advice and Industry Information
- Star Micronics Printer Drivers
- Technical Questions/Support

Android Developers Resource

The official Android development resource.

System Requirements for Android Development

See requirements for Windows, Linux, and Mac.

Unicode.org

The Unicode Consortium – Good place to learn more about Unicode.

1D Barcodes

Barcode Island is a great resource for specs on 1D barcodes.

2D Barcodes

Great place for information on 2D Barcodes, QR Codes, and PDF417

Code Pages

Learn about Code Pages here.

ASCII Table Resource

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
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	?

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
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	□

Use this to compare hex values to symbol (ASCII) values.

SDK Package Version History

Release Date	SDK Package Version	Update
Apr. 17 2015	3.14.0	- Initial Release
Aug. 31 2015	3.15.0	- Added mPOP support.

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/