Android Software Development Kit

How to Use StarIO for Portable Printers (ESC/POS)

This SDK contains a Java Eclipse project for Android devices.

Supported Versions	Supported Star Interfaces	Compatible StarIO JAR File
Android OS 2.2 and Higher	Bluetooth, Ethernet, WLAN	StarIOPort3.1.jar
Android OS 2.1	Bluetooth, Ethernet, WLAN	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

When using ESC/POS emulation:

To use the ESC/POS emulation, set the emulation setting of the printer to “ESC/POS Mode”. To change the emulation, proceed as follows.

◆ **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

Table of Contents 2
About this Manual 4
Star Printer Compatibility Chart 5
Connecting a Star Portable Printer to an Android Device 6
Configuring an Android Device for Development 9
Importing and Running the StarIO Android SDK 10
Using the SDK with Portable Star Micronics Printers 16
Overview of How This Android SDK is Designed 22
StarIO – (StarIOPort3.1.jar) 23
StarIO Methods Overview 26
StarIO Android SDK Functionality 43
- Port Discovery 53
- get StarIO Version 53
- Help 54
- Get Firmware Information 54
- Get Status 55
- Sample Receipts / JP Sample Receipt 56
- 1D Barcodes 57
- 2D Barcodes 58
- Text Formatting 60
- Japanese Kanji Text Formatting 62
- Raster Graphical Text Printing 64
- Image File Printing 65
- MSR (Magnetic Stripe Reading) 66

Tips for App Development when using StarIO 67
Additional Resources 69
ASCII Table Resource 70
SDK Package Version History 71

About this Manual

This manual is designed to help you understand StarIO and how to build an Android application to interact with Star Micronics Thermal Portable 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 Portable Printer Models supported on Android Operating Systems.

Star Printer		Port Discovery	Get Firmware Information	Status	Sample Receipts	Begin / End Checked Block	1D Barcodes	2D Barcodes	Text Formatting	Raster Graphics	Sample Images	MSR	Bluetooth Setting
Model	Interface												
SM-S200	Bluetooth	✓	✓	✓	✓	✓*1	✓	✓	✓	✓*2	✓*2	✓	
SM-S210i (Jp model only)	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*3
SM-S220i (Models for US and Europe only)	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*3
SM-S230i (Models for US and Europe only)	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SM-T300	Bluetooth	✓	✓	✓	✓	✓*1	✓	✓	✓	✓*2	✓*2	✓	
	WLAN		✓	✓	✓	✓*1	✓	✓	✓	✓*2	✓*2	✓	
SM-S300	Bluetooth	✓	✓	✓	✓	✓*1	✓	✓	✓	✓*2	✓*2	✓	
SM-T300i	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*3
SM-S400	Bluetooth	✓	✓	✓	✓	✓*1	✓	✓	✓	✓*2	✓*2	✓	
SM-T400i	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*3

(*1) F/W Ver.2.4 or later is required.

(*2) F/W Ver.2.4 or later is required.
If the F/W Ver.2.3 is used, please read the notes below.

- End Checked Block
- Raster Graphical Text Printing
- Image File Pringing

(*3) F/W Ver.3.0 or later is required.

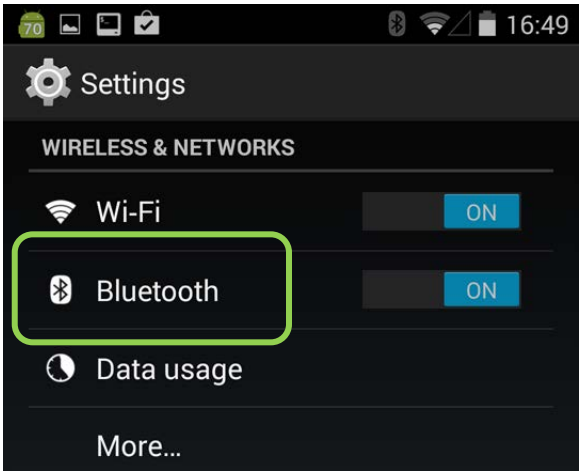
Note: This SDK offers the most popular features, but not all printer functionality has been included. The commands not included in this SDK application are available in the Star Mobile Printer Programming Manual.

Connecting a Star Portable Printer to an Android Device

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

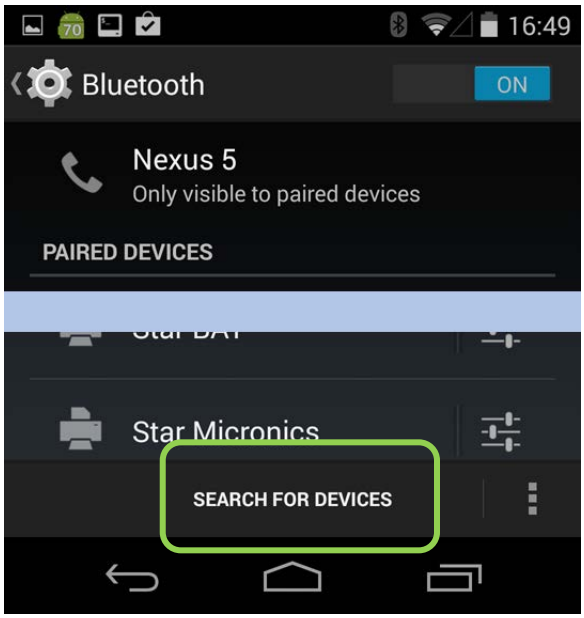
Bluetooth

1. Tap Settings.

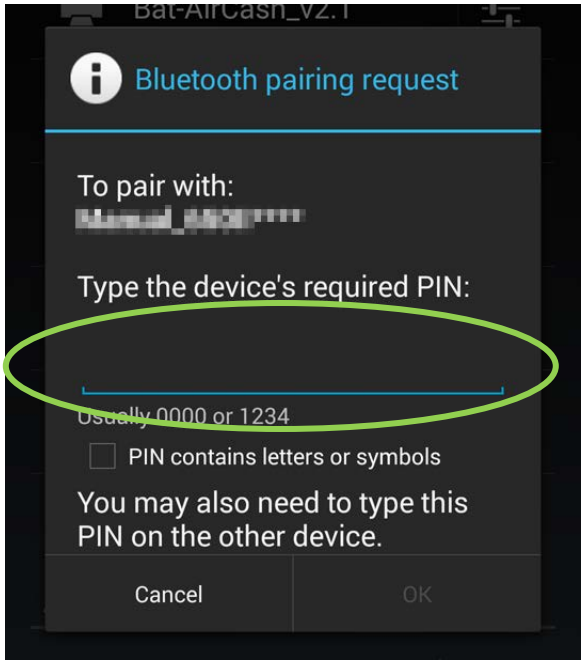


2. Ensure Bluetooth is enabled and the printer is powered on.

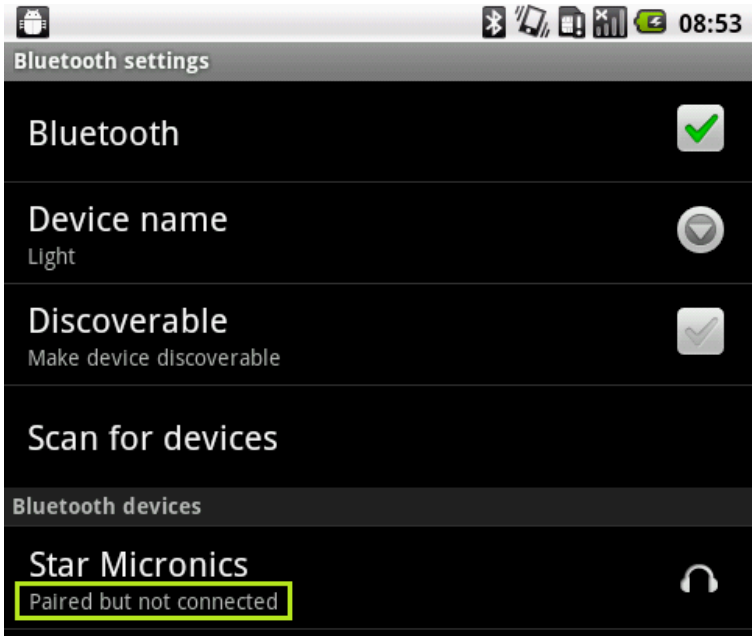
3. Tap Bluetooth to access its settings.



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



5. Enter the PIN number. The factory default for standard Star Portable Printers is "1234".

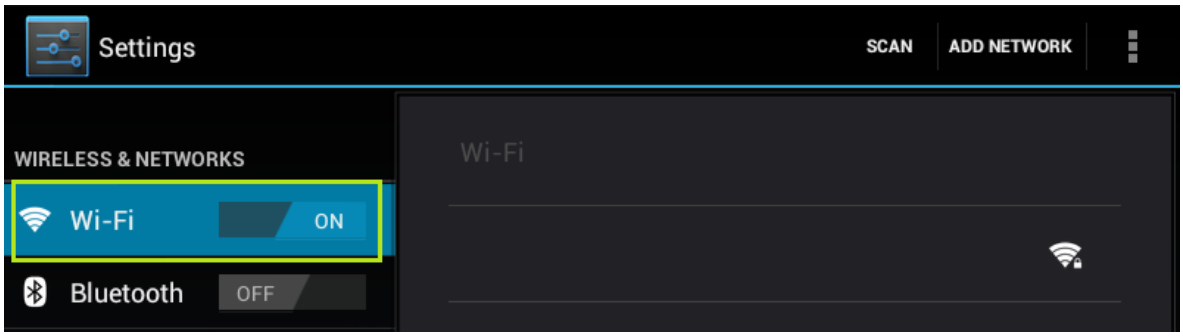


6. Older Android Operating Systems, such as V2.2 in the above screenshot, display the message "Paired but not connected". This can be neglected as connection is established at the time of printing.

Wireless LAN

1. Assign an IP Address to the Star Printer and connect it to the network. Standard Star Printers do not ship with an IP Address pre-assigned; this can set by using the WLAN Setup Utility for Mobile Printers available on Star's website.

2. Tap Settings.



3. Ensure Wi-Fi is enabled.

4. Connect to the same network the Star Printer is on.

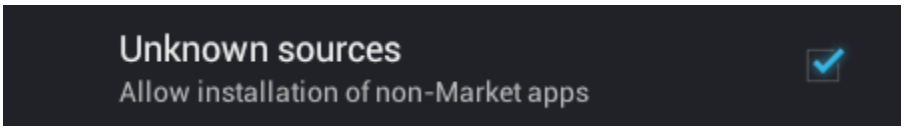
5. No pairing or further settings are necessary to connect the printer as the Android device will output data directly to the printer's TCP/IP port.

Configuring an Android Device for Development

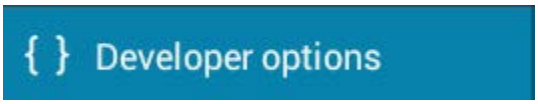
1. Tap Settings.



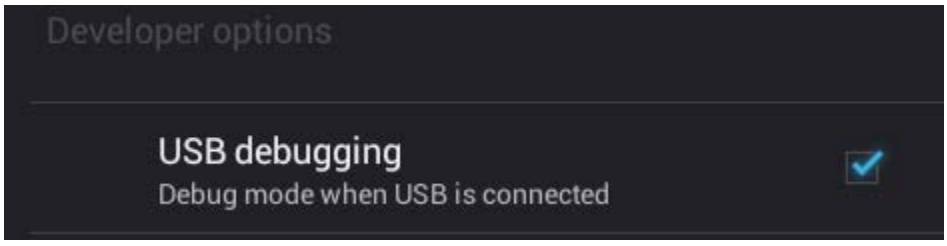
2. Tap Security.



3. Enable Unknown sources.



4. Tap Developer options.



5. Enable USB debugging.

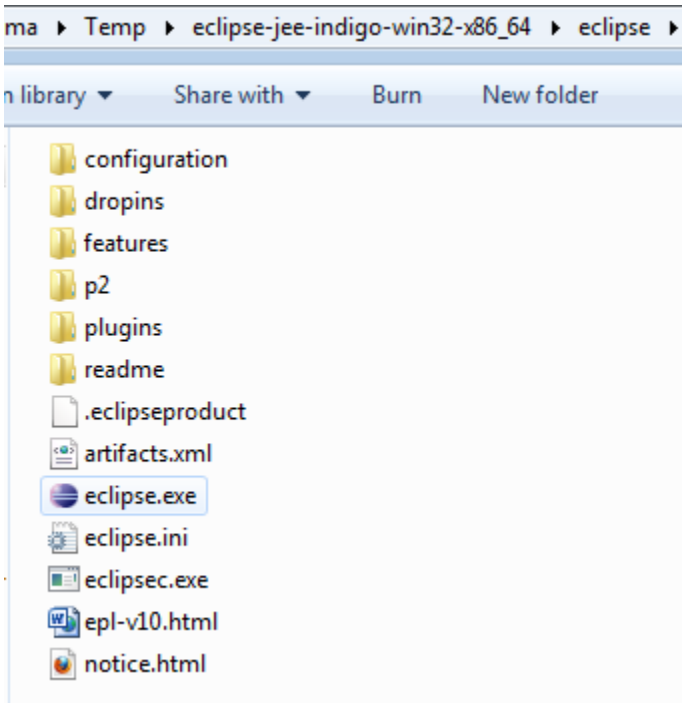
Importing and Running the StarIO Android SDK

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.

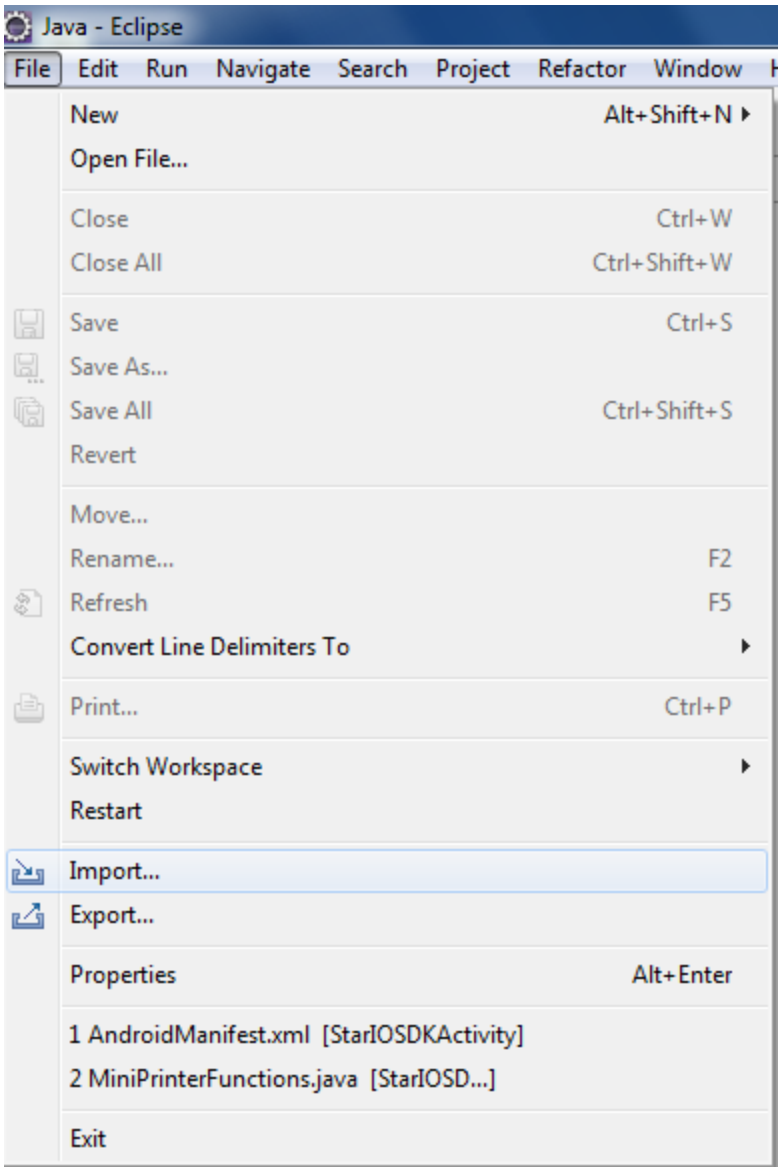
This section will explain:

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

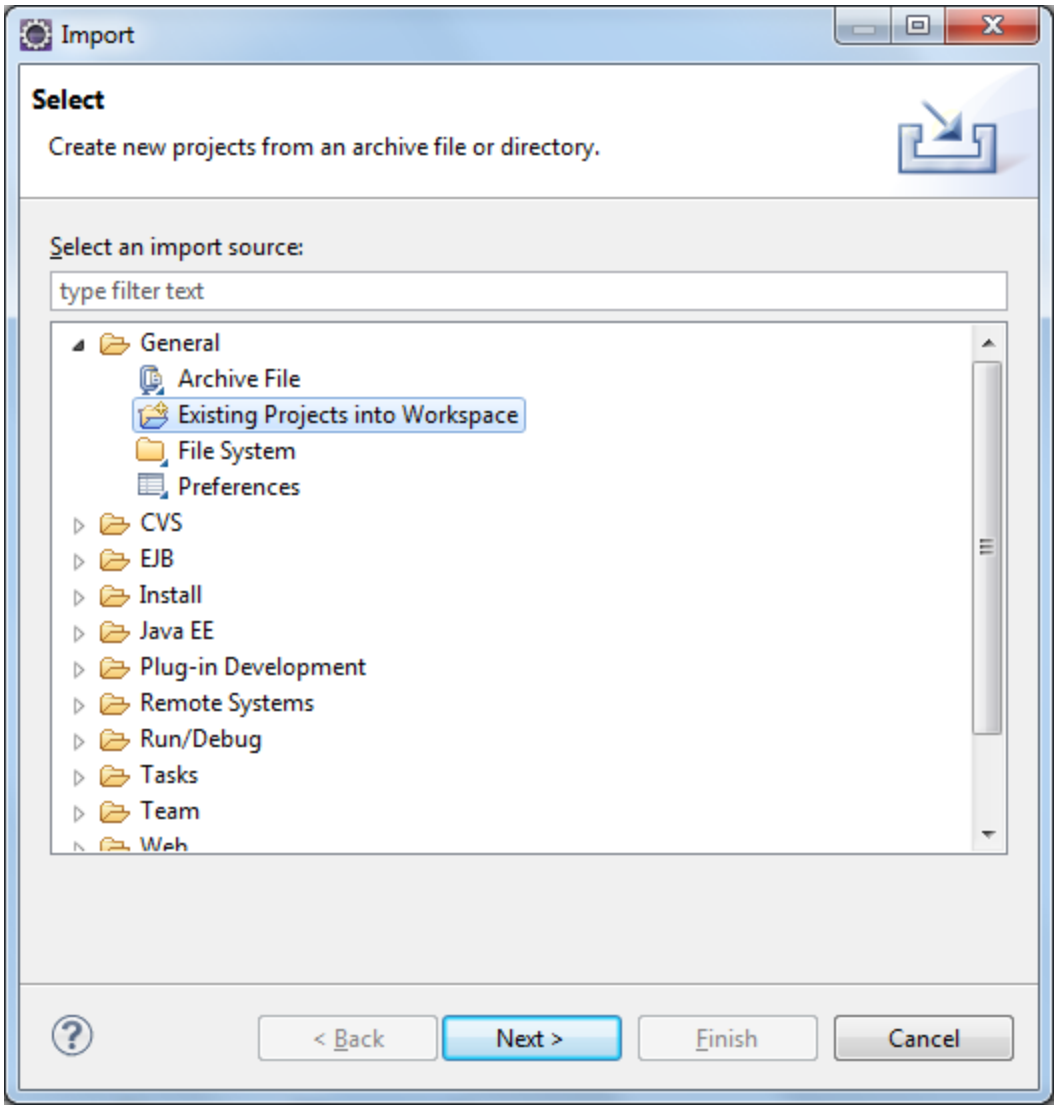
How to import the Android SDK project into Eclipse:



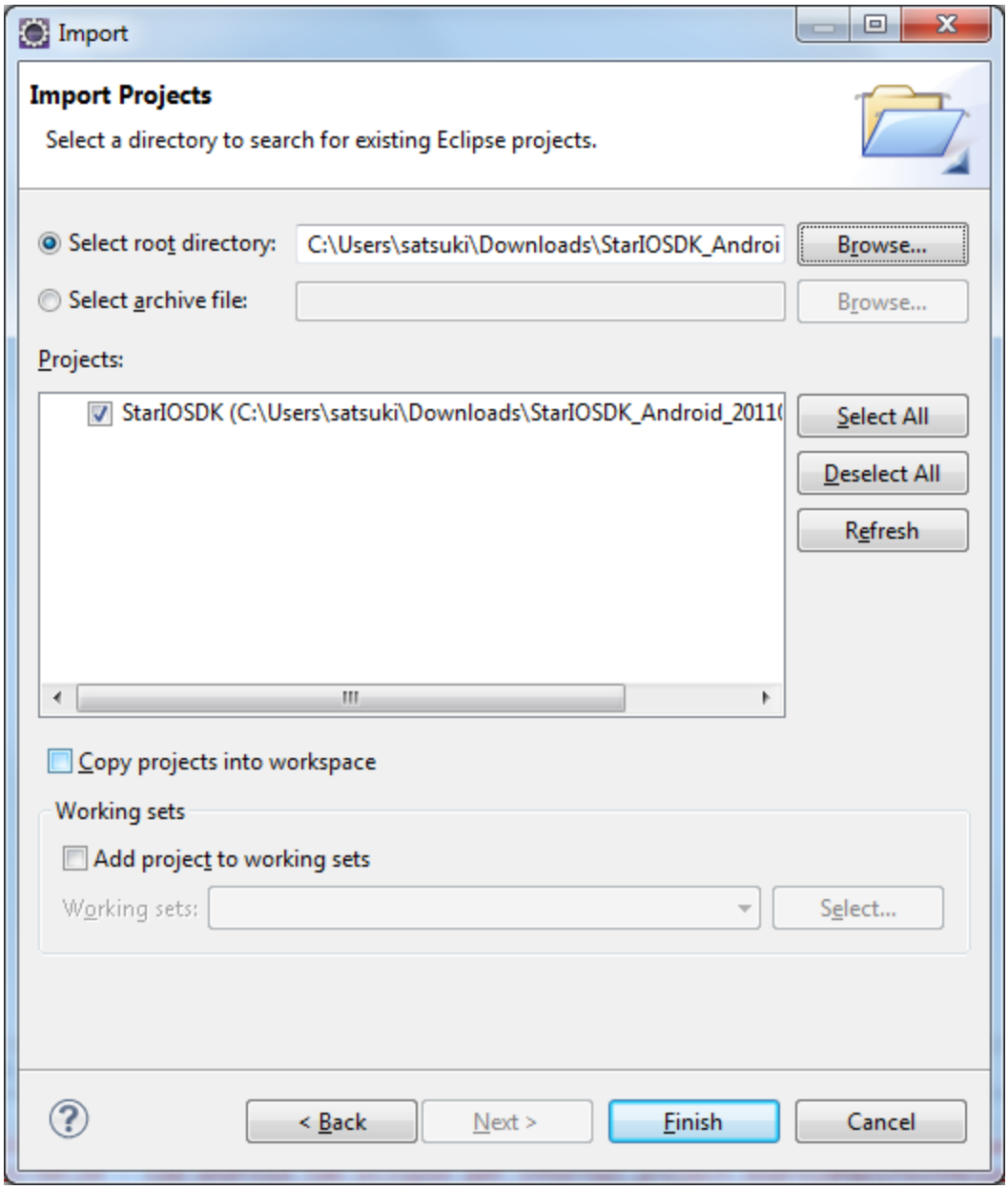
1. Execute Eclipse.



2. File > Import

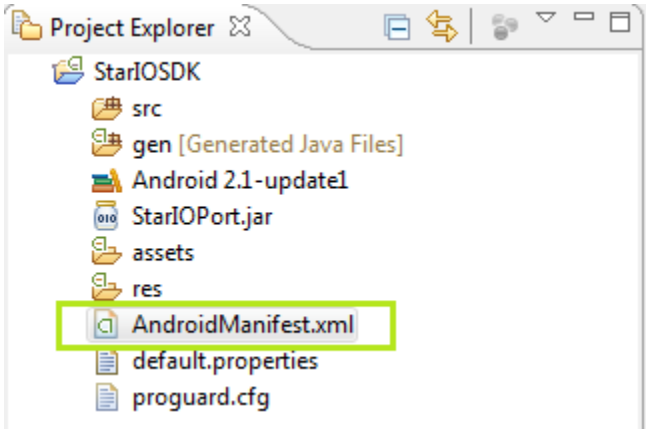


3. General > Existing Projects into Workspace

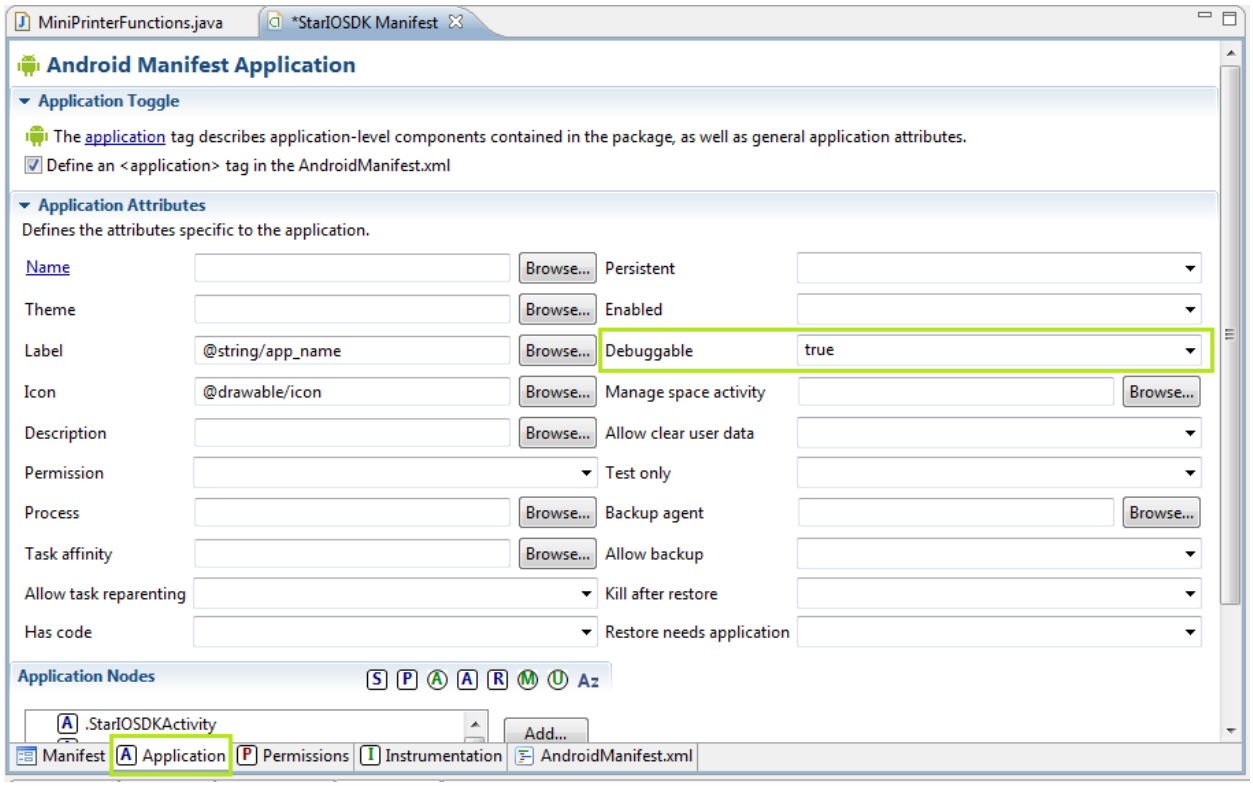


4. Browse to the location where the Star SDK is saved. Click Finish.

Enabling debug mode in Eclipse:



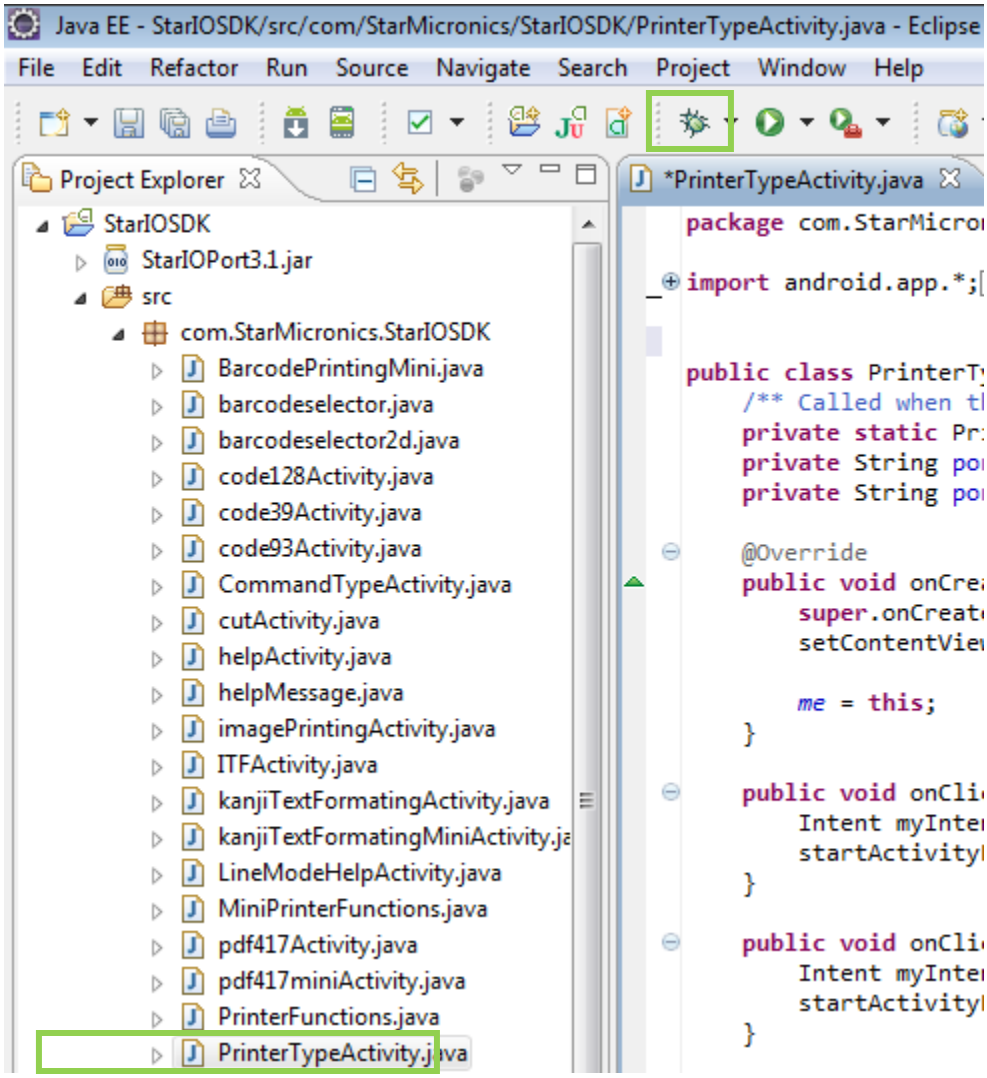
1. Click AndroidManifest.xml.



2. Click the Application tab.

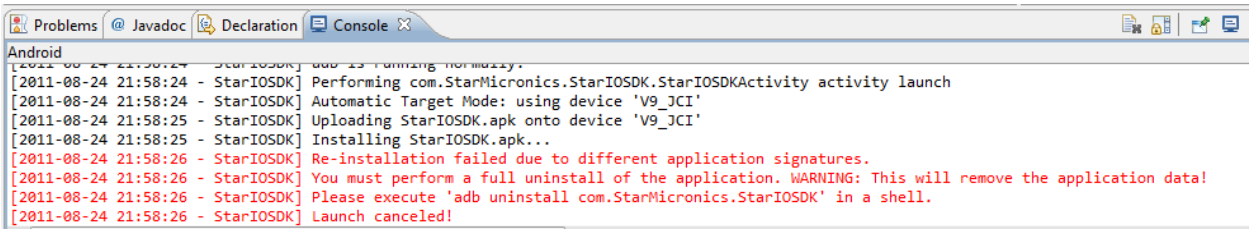
3. Set Debuggable to true.

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 Portable Star Micronics Printers

Please make sure you have a compatible Star Micronics Portable Printer Model.

Port Name and Interface Relation:

StarIO uses specific port names to identify what port will be used. These are very important to understand because not following the naming convention correctly will fail to communicate with the printer. For detailed information, please see here.

Interface	Port Name
Bluetooth	BT:
	BT:Device Name
	BT:Printer MAC Address
Wireless LAN (TCP/IP)	TCP:IP Address

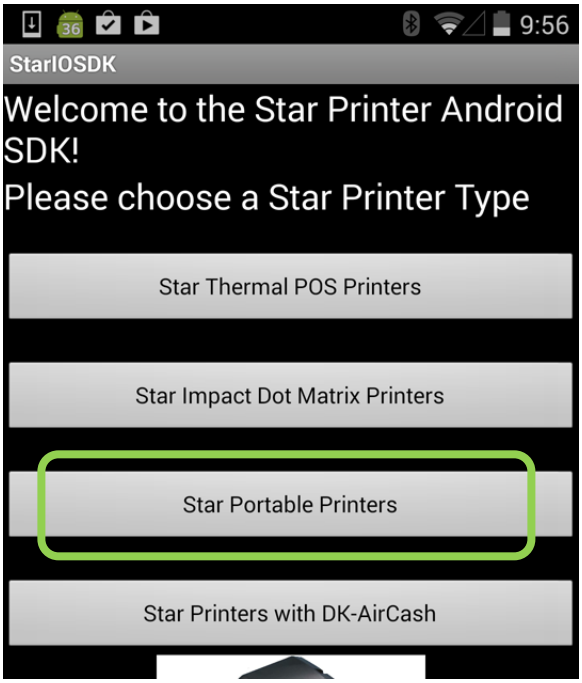
◆Notification

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.

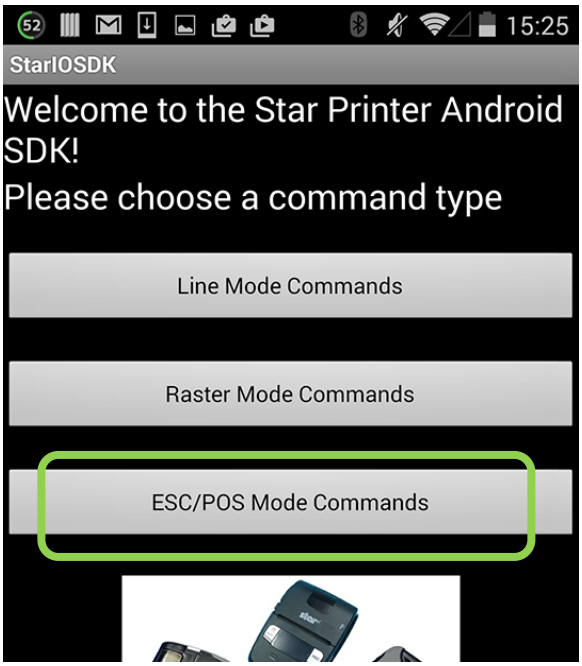
Change port settings from [portSettings = "portable;escpos";] to [portSettings = "portable;escpos;u";]

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

Using a Mobile Printer

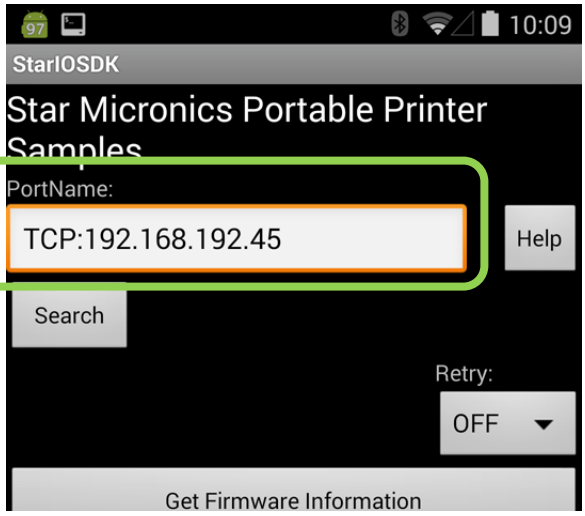


1. Tap “Star Mobile Printers”. This section contains all Mobile Printer samples.



2. Tap “ESC/POS Mode Commands”.

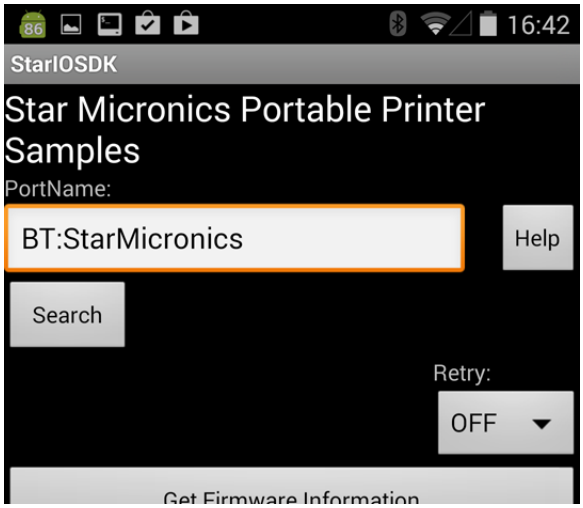
Wireless LAN Printers



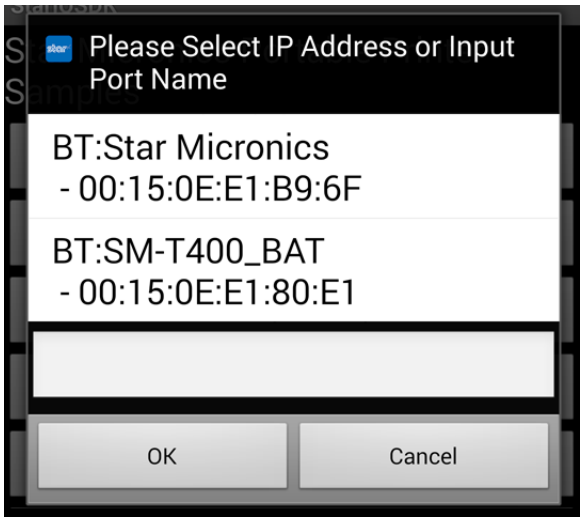
1. The IP Address must be manually entered into the “PortName” field.

   Enter TCP:<IP Address> without brackets around the printer’s IP Address.

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 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

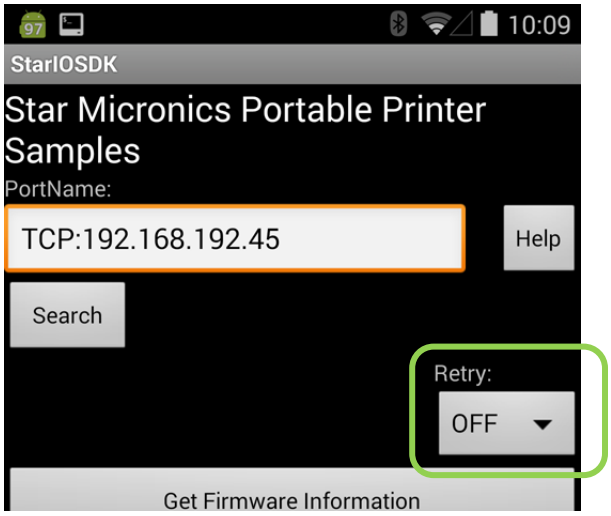
There are three types of port details that can be used for Star Bluetooth Printers:

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;escpos;I" will be specified for the the portSetting parameter of the getPort method.

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

Using an Android Virtual Device(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. They are listed again here for convenience:

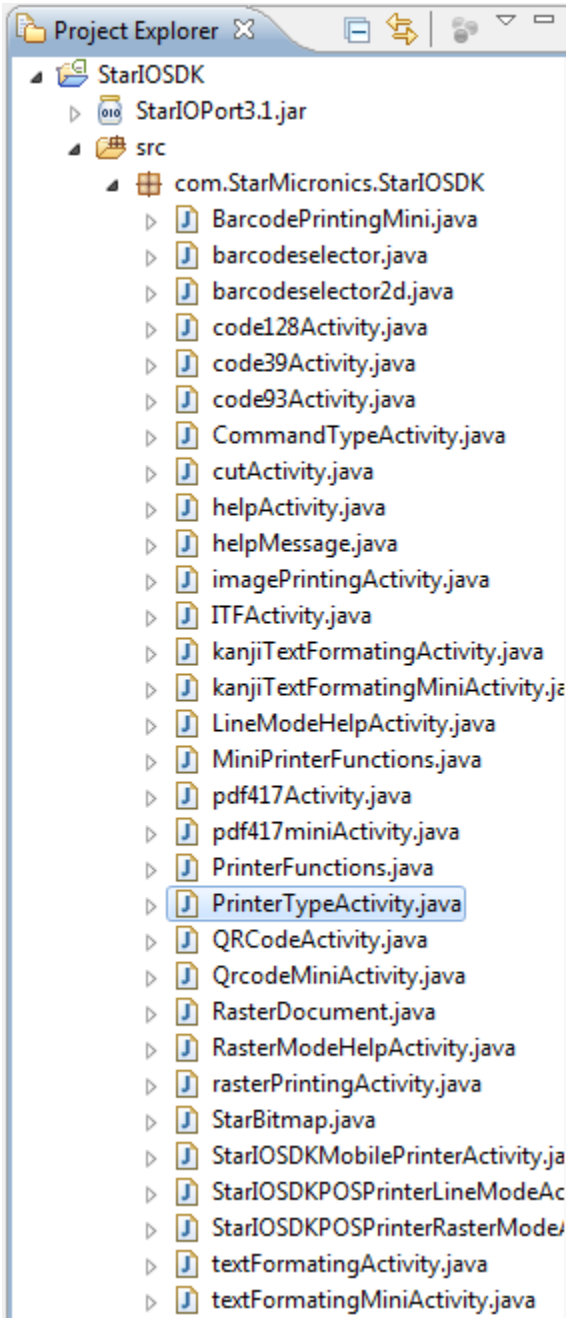
Portable Printers

- No Cash Drawer support
- No Cut Pattern support

POS Printers

- No Magnetic Stripe Reader support

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:

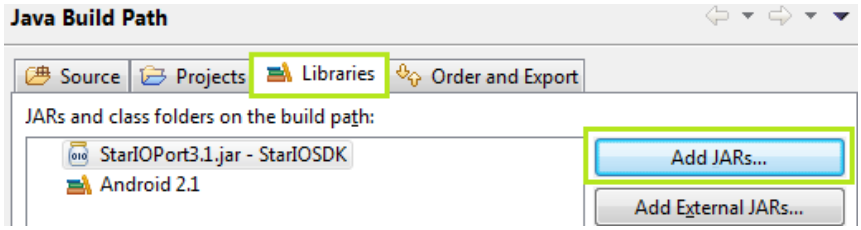
The StarIO SDK contains two library files. Use the one which corresponds to the Android OS Version your application is being developed for.

Supported Versions	Supported Star Interfaces	Compatible StarIO JAR File
Android OS 2.2 and Higher	Bluetooth, Ethernet, WLAN	StarIOPort3.1.jar
Android OS 2.1	Bluetooth, Ethernet, WLAN	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:

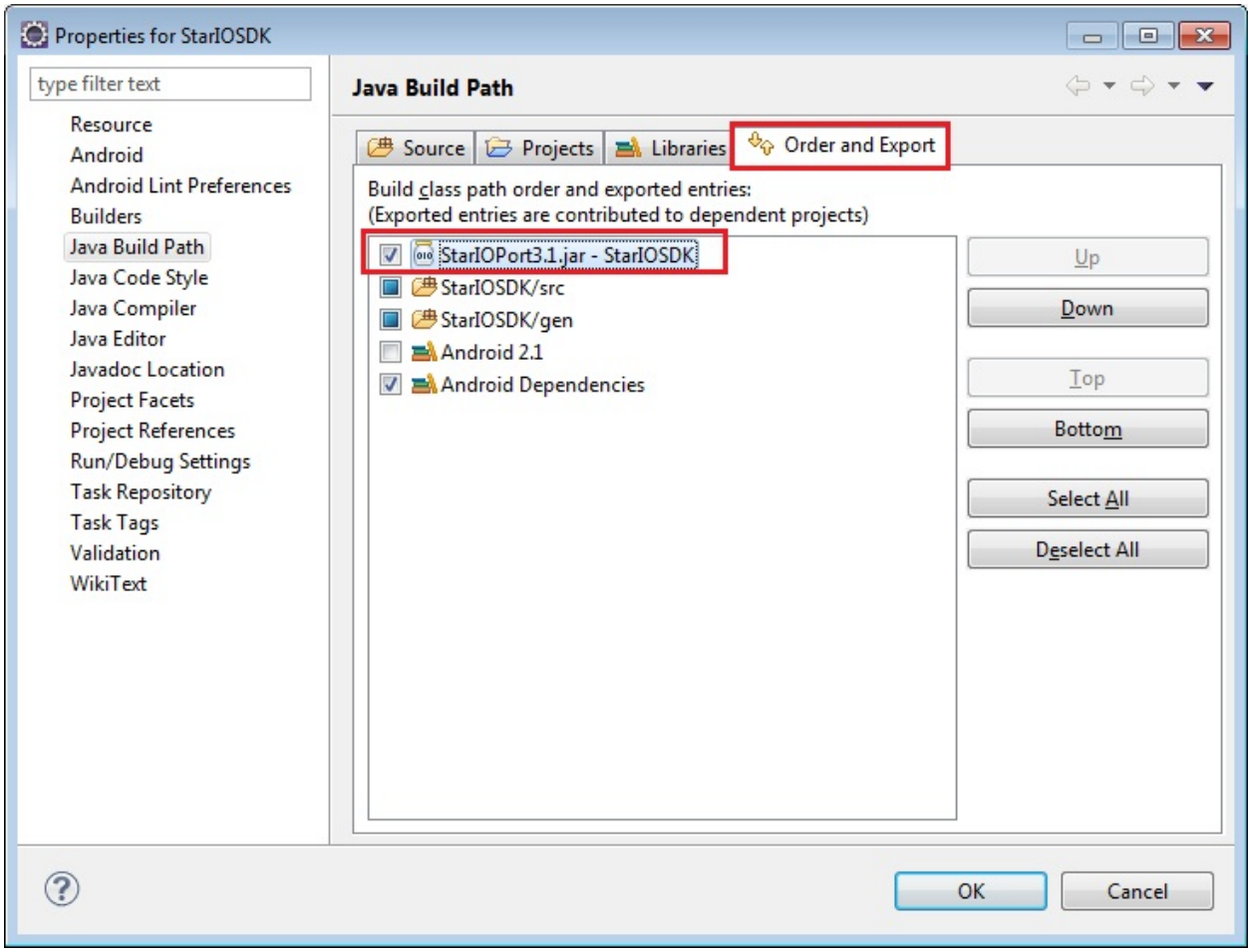
```
import com.starmicronics.stario.StarIOPort;
import com.starmicronics.stario.StarIOPortException;
import com.starmicronics.stario.StarPrinterStatus;
```

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

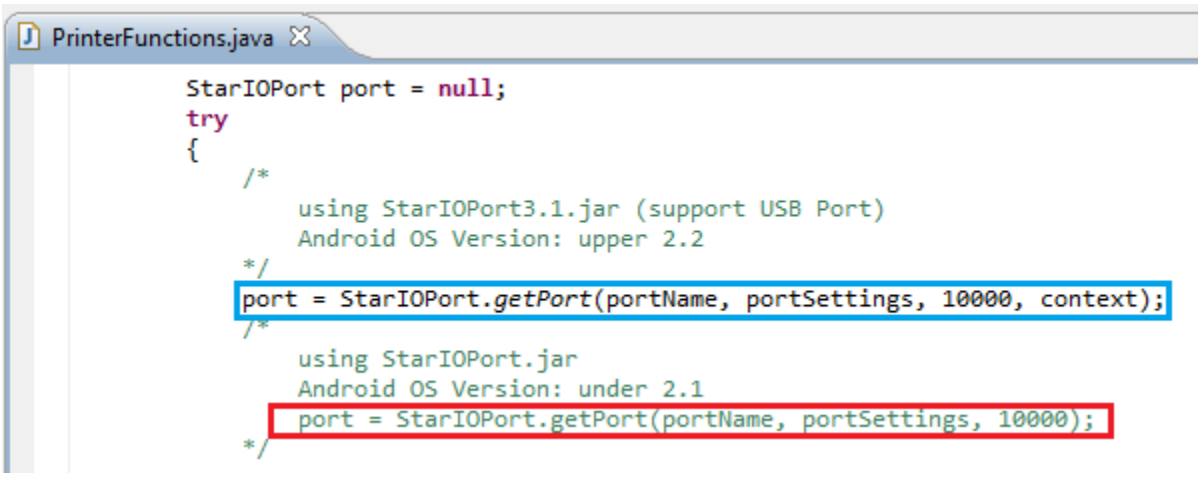
Android OS V2.2 and Higher:

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 2.2 and higher support Ethernet, WLAN, and Bluetooth connections.

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(portName, portSettings, 10000);

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

Android OS 2.1 supports Ethernet, WLAN, and 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. TCP:192.168.1.2 / BT:DeviceName

portSettings specify the following settings depending on your application.

- Portable printers setting
- Retry setting

*portSettings must be set to “mini” for portable printers.

Interface	Port Name	Port Settings	
Network I/F (TCP/IP)	TCP:IP Address	portable;escpos	Use Portable printer
Bluetooth	BT: BT:Device Name	portable;escpos	Use Portable printer
	BT:Printer MAC Address	portable;escpos;l	Retry ON

Example of Port Setting Property :

portSettings = “portable;escpos;l” In case of using the Bluetooth interface and retry ON

**@” mini” is available for backwards compatibility only.*

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)

● **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
}
```

`StarIOPort` 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.

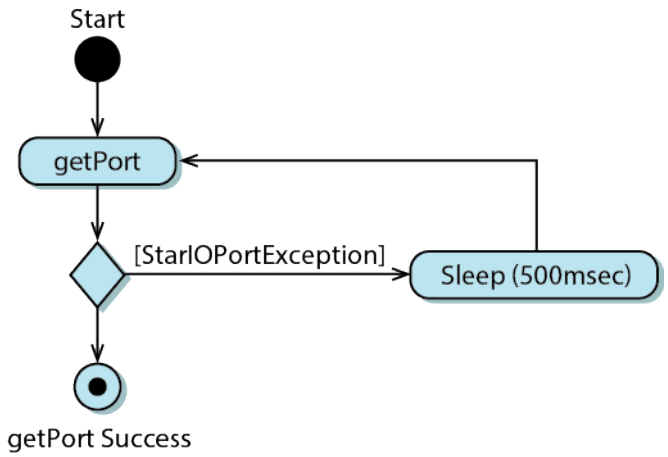


When Sleep Mode is enabled, it is necessary to execute getPort() and releasePort() before the printer physically enters Sleep Mode

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. The rate of failure depends on the host device or possibly other conditions.

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)
                                                    throws StarIOPortException
```

searchPrinter gets Bluetooth devices which connect with Android terminal, as a list.

When set “BT:” as a Argument value, return all Bluetooth devices, set “BT: with specifc string” returns Bluetooth deivces which name from specific strings.

PortInfo class of return value includes, PortName, MAC address for Bluetooth devices and you can get them by String getPortName(), and String getMacAddress() method.).

You can use PortName as a Argument value of getPort().

```
//The following would be an actual usage of searchPrinter:
try
{
   List<PortInfo> portList = StarIOPort.searchPrinter(“BT:Star”);
   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 searching the port
}
```

This sample shows, get connected Bluetooth devices which name start from “Star” 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.

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.



For Ethernet Interface:

The getPort method may fail if you run it just after running releasePort.
You must run the getPort method after 500msec from releasePort.

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 (printerStatus.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	Portable printers are not supported.
compulsionSwitch	Compulsion SW	boolean	Portable printers are not supported.
coverOpen	Cover Status	boolean	You can check status of Cover " true " : Cover is opened. " false " : Cover is closed.
cutterError	Auto-cutter Error	boolean	Portable printers are not supported.
etbAvailable	ETB available or not	boolean	Portable printers are not supported.
etbCounter	ETB Counter	int	Portable printers are not supported.
headThermistorError	Head Thermistor Error	boolean	Portable printers are not supported.
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	Portable printers are not supported.
presenterPaperJamError	Presenter Paper Jam Error	boolean	Portable printers are not supported.
presenterState	Presenter Paper Position	int	Portable printers are not supported.
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	Portable printers are not supported.
receiveBufferOverflow	Receive Buffer Overflow	boolean	Portable printers are not supported.
unrecoverableError	Non-recoverable Error	boolean	Portable printers are not supported.
voltageError	Electric Voltage Error	boolean	Portable printers are not supported.

Class structure StarPrinterStatus Supported by Portable Printers

Member name	SM-T300	SM-S200	SM-S300	SM-S400	SM-S210i *	SM-S220i **	SM-S230i **	SM-T300i	SM-T400i
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**

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 setEndCheckedBlockTimeoutMills 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.



When using SM-S200, SM-S300, SM-S400 or SM-T300 with firmware Ver 2.3 or earlier, the following limitation exists.

- You can confirm the completion of transmission of print data but cannot confirm the completion of printing.

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

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

    //Printing
    byte[] command = MiniPrinterFunctions.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) {}
    }
}
```

compressRasterData / generateBitImageCommand

```
public byte[] compressRasterData  (int width, int height, byte[] imageData,
                                   String portSettings)

public byte[] generateBitImageCommand  (int width, int height, byte[] imageData,
                                   String portSettings)[Deprecated]

                                                    throws StarIOPortException
```

This method generates the print command from image data.

Parameters:

width	– Width of image data (pixel)
height	– Height of image data (pixel)
imageData	– Raster data before compression
portSettings	– Port options ”portable;escpos” is specified for a portable printer

Returns:

Success: Compressed Raster data

Failure: null

Throws:

`StarIOPortException` - When failed to generate the print command

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.

getFirmwareInformation

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

This method gets a model name and firmware version of the printer and 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

setPinCode

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

This method sets the PIN code of Star Bluetooth Device of at least 4 character, with a max length of 16 characters.

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

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.

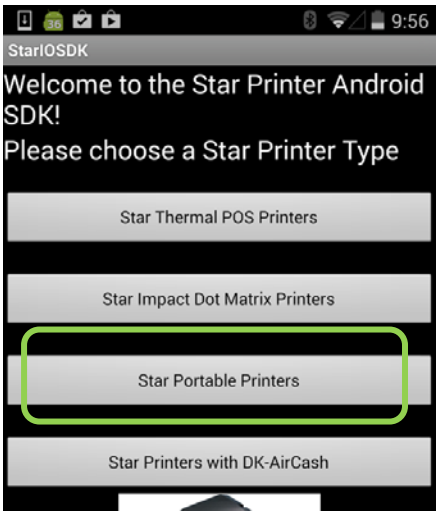
StarIO Android SDK Functionality

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

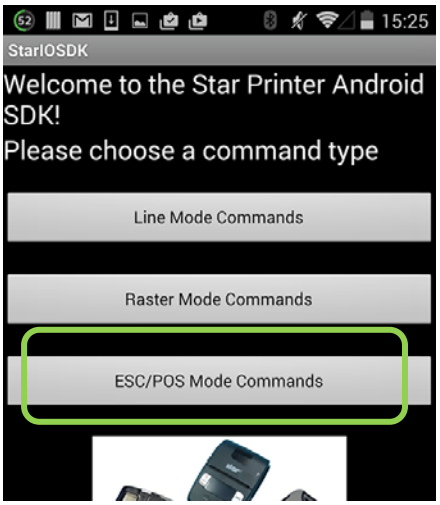
This SDK also has page and section references to the Mobile Command Manual for more information so please download and study it if you need more detail on a specific command.

Upon running the SDK, the above buttons are displayed to demonstrate different functionality. This section describes what each does and how the command can be tailored to fit your needs.

Choosing Portable Printer Commands



1. Tap “Star Portable Printers”.

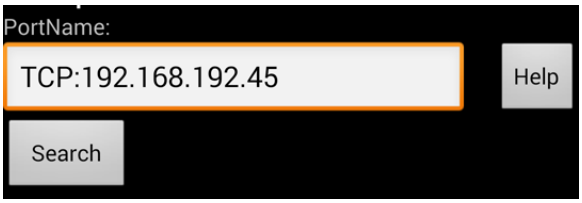


2. Tap “ESC/POS Mode Commands”.

Supported Samples

Port Discovery
Get Firmware Information
Get Status
Sample Receipt
JP Sample Receipt
1D Barcodes
2D Barcodes
Text Formatting
JP Kanji Text Formatting
Raster Graphics Text Printing
Image File Printing
Magnetic Stripe Reading
Bluetooth Setting

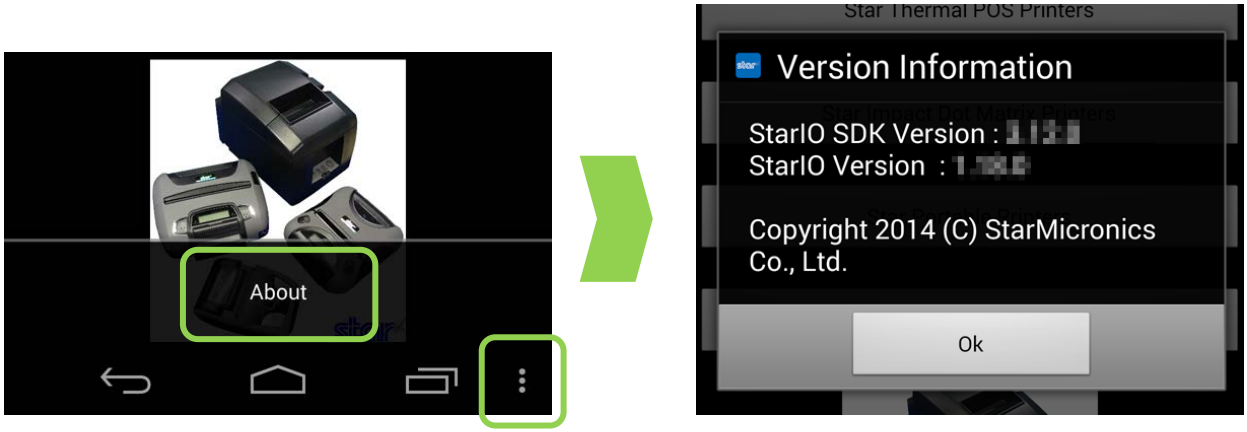
Port Discovery



Automatically detects which Star Micronics Bluetooth Printers are paired. Tap the printer to connect to it. This feature is documented in greater detail here. Wireless portable printers currently do not support this feature.

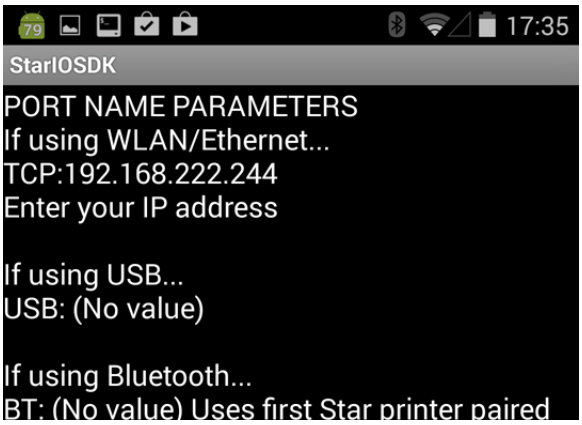
---

get StarIO Version



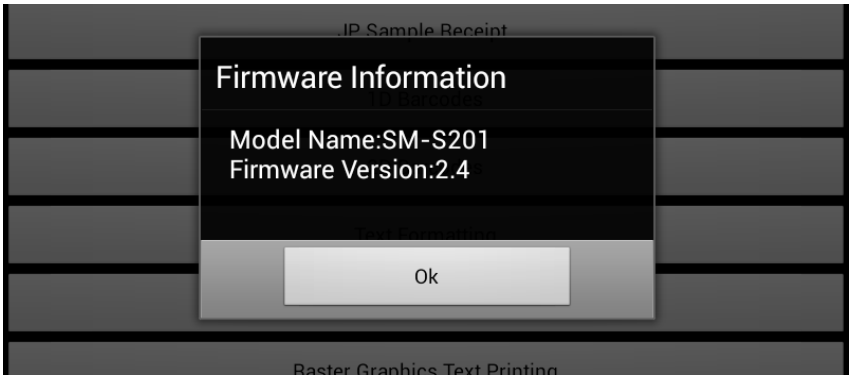
This method gets a StarIO Version of the printer.

Help



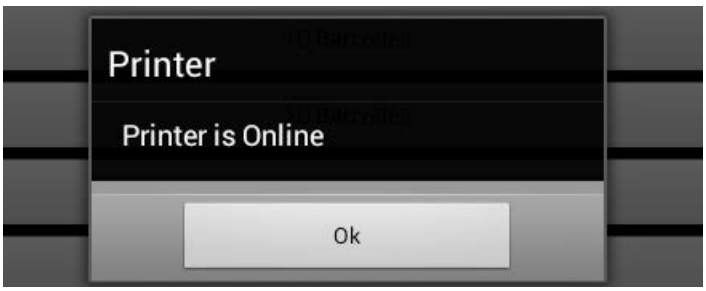
Displays rules for manually entering the printer's port data into the "PortName" field. Manually entering port data is unnecessary if Port Discovery is used.

Get Firmware Information



Displays firmware information of the printer specified by Port Name.

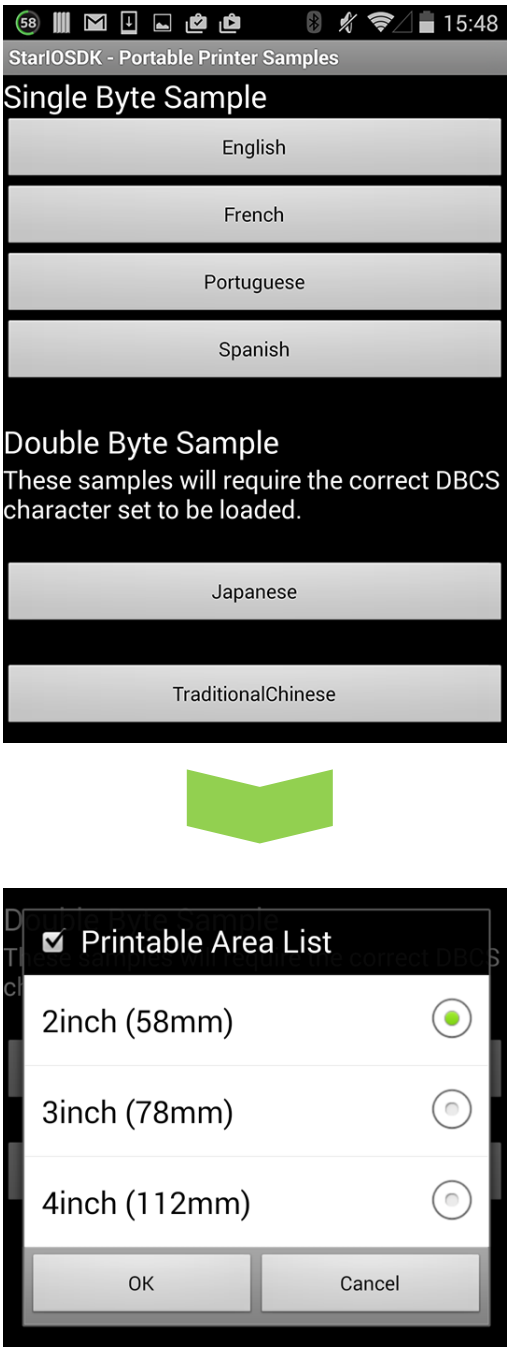
Get Status



StarPrinterStatus

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

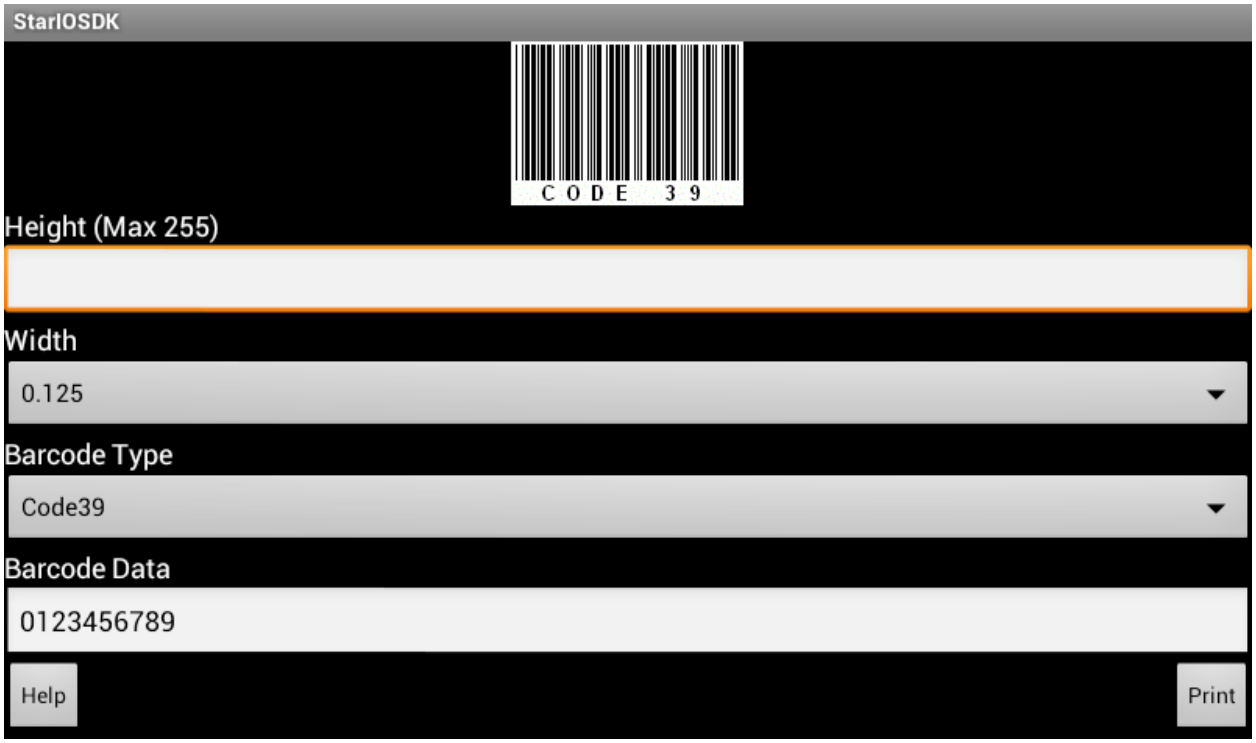
Sample Receipts



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.

1D Barcodes



Set Barcode Height

GS h n 0 ≤ n ≤ 255

Set Barcode Width

GS w n 1 ≤ n ≤ 8

Print Barcode

GS k m n d1…dn NUL m = Barcode Type
n = Number of data
d1…dn = Barcode Data

2D Barcodes

QR Code



Select QR Code

GS Z n n = 2 (QR Code)

Print QR Code

ESC Z m a k nL nH d1…dn

m = Version of the symbol m = 1~40, 0 = Auto size

a = Correction Level L = 7%, M = 15%, Q = 25%, H = 30%

k = Model Size k = 1~8

nL = Lower Byte Value; nH = Higher Byte Value

d1…dn = QR Code Data

PDF417



Select PDF417

GS Z n n = 0 (PDF417)

Set PDF417 Size

GS w n 1 ≤ n ≤ 8

Print PDF417

ESC Z m a k nL nH d1…dn

m = Column Number 1 ≤ m ≤ 30

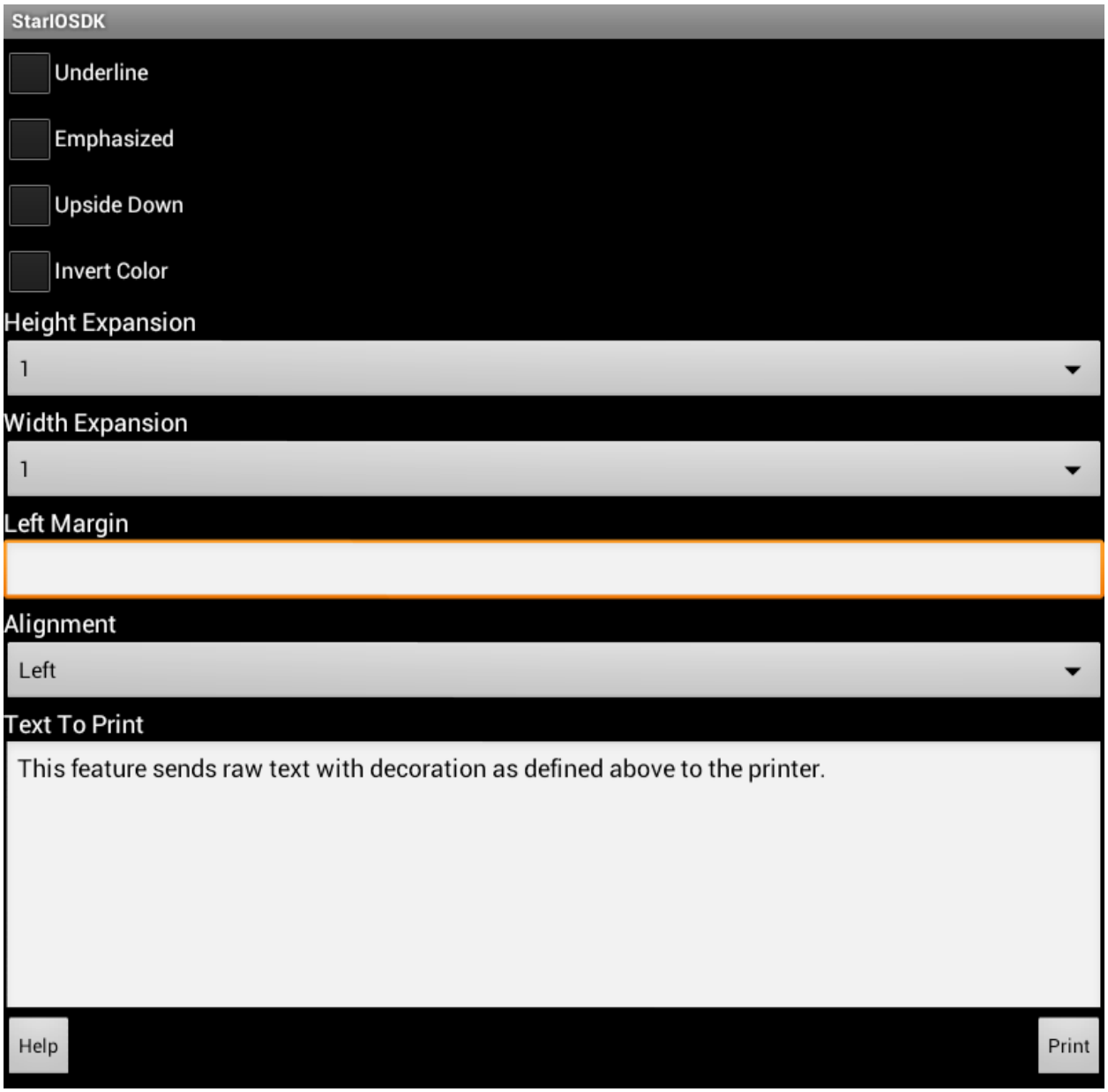
a = Correction Level 0 ≤ n ≤ 8

k = Horizontal/Vertical Ratios 2 ≤ k ≤ 5

nL = Lower Byte Value; nH = Higher Byte Value

d1…dn = Barcode Data

Text Formatting



Text Formatting (continued from above)

Underline

ESC – n	1 = Underline On, 1 dot thick	
	2 = Underline On, 2 dots thick	
	0 = Underline Off [Default]	

Emphasized

ESC E n	1 = On	0 = Off [Default]

Upside Down Text

ESC { n	1 = On	0 = Off [Default]

Invert Color

GS B n	1 = On	0 = Off [Default]

Character Expansion

GS ! n	0 ≤ n ≤ 8

Left Margin

GS L nL nH	0 ≤ nL ≤ 255	0 ≤ nH ≤ 255

Alignment

ESC a n	0 = Left [Default]	1 = Center	2 = Right

Japanese Kanji Text Formatting



This functionality is the same as that of Text Formatting, except Japanese Kanji is supported.

Japanese Kanji Text Formatting (continued from above)

Underline

ESC – n	1 = Underline On, 1 dot thick 2 = Underline On, 2 dots thick 0 = Underline Off [Default]	

Emphasized

ESC E n	1 = On	0 = Off [Default]

Upside Down Text

ESC { n	1 = On	0 = Off [Default]

Invert Color

GS B n	1 = On	0 = Off [Default]

Character Expansion

GS ! n	0 ≤ n ≤ 8

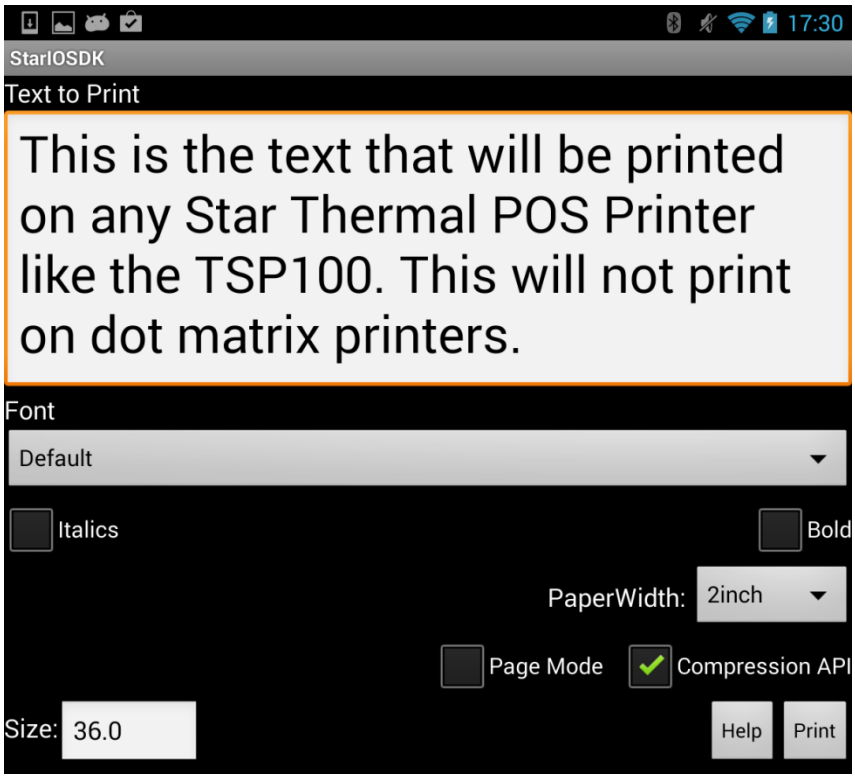
Left Margin

GS L nL nH	0 ≤ nL ≤ 255	0 ≤ nH ≤ 255

Alignment

ESC a n	0 = Left [Default]	1 = Center	2 = Right

Raster Graphical Text Printing



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 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.

Note: If send large amounts of raster data command, refer to the following in order to prevent "data defective". Detail refer to the **"printBitmap"** method in the **"MiniPrinterFunctions.java"**.

[Printer Device Firmware version 2.4 later]

Use beginCheckedBlock / endCheckBlock method “before / after” sending data by writePort method.

[Printer Device Firmware version 2.3 earlier]

For confirming the end of printing, send query commands after writePort data.

Store Print Job as Raster Graphic

ESC X 4

Print Stored Raster Graphic Data

ESC X 2

Image File Printing

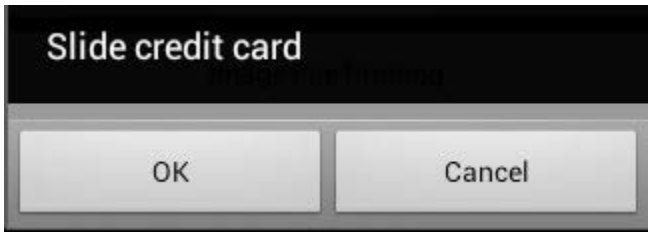


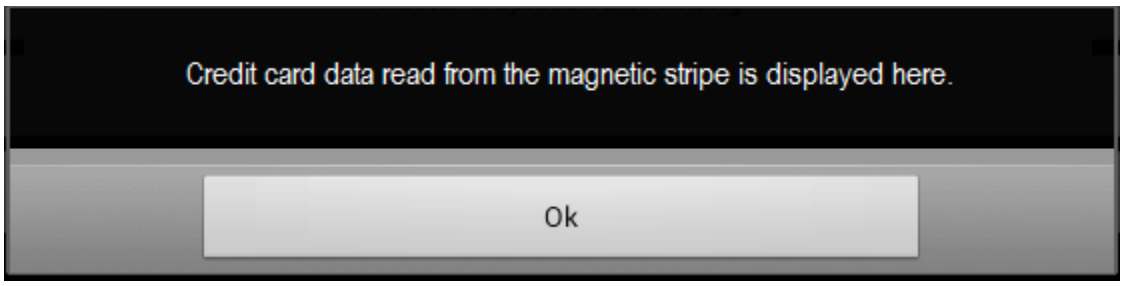
Use the dropdown box to select one of four different sample images to print via Raster Graphics and choose the desired paper width.

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 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.

Note: If send large amounts of raster data command, refer to the following in order to prevent "data defective". Detail refer to the **"printBitmapImage"** method in the **"MiniPrinterFunctions.java"**.

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

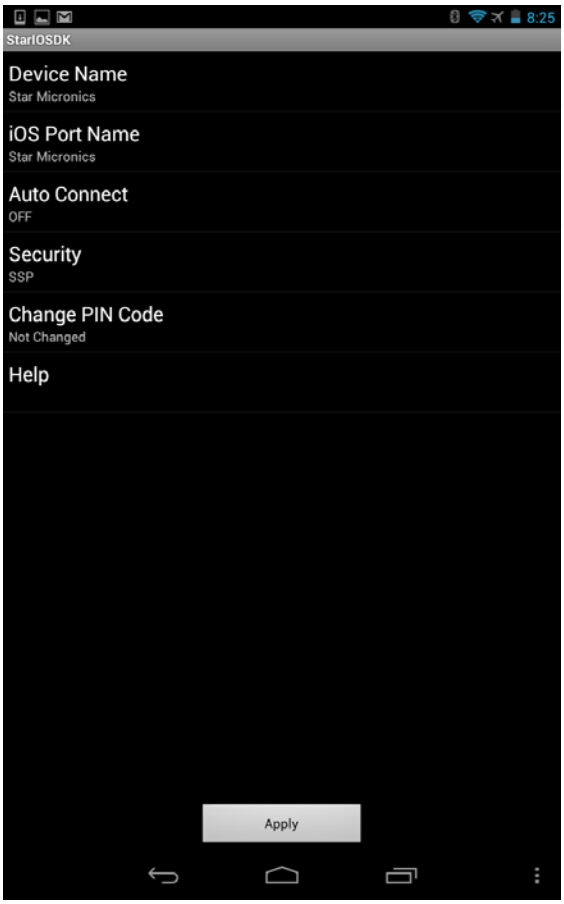
Set JIS-II Mode

ESC M J	Read Japanese Characters

Cancel MSR Mode

EOT	Return to standard print mode

Bluetooth Setting



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.

F/W Version 3.0 or later is required for SM-S210i, SM-S220i, SM-T300i and SM-T400i.

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 which are answered here.

TIP #3: 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 #4: 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 #5: Looking for an iOS printing SDK? Visit our Developers section to get access to Star developer tools for these environments.

Additional Resources

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
- 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
0	0	NUL
1	1	SOH
2	2	STX
3	3	ETX
4	4	EOT
5	5	ENQ
6	6	ACK
7	7	BEL
8	8	BS
9	9	TAB
10	A	LF
11	B	VT
12	C	FF
13	D	CR
14	E	SO
15	F	SI

ASCII	Hex	Symbol
16	10	DLE
17	11	DC1
18	12	DC2
19	13	DC3
20	14	DC4
21	15	NAK
22	16	SYN
23	17	ETB
24	18	CAN
25	19	EM
26	1A	SUB
27	1B	ESC
28	1C	FS
29	1D	GS
30	1E	RS
31	1F	US

ASCII	Hex	Symbol
32	20	(space)
33	21	!
34	22	"
35	23	#
36	24	$
37	25	%
38	26	&
39	27	'
40	28	(
41	29	)
42	2A	*
43	2B	+
44	2C	,
45	2D	-
46	2E	.
47	2F	/

ASCII	Hex	Symbol
48	30	0
49	31	1
50	32	2
51	33	3
52	34	4
53	35	5
54	36	6
55	37	7
56	38	8
57	39	9
58	3A	:
59	3B	;
60	3C	<
61	3D	=
62	3E	>
63	3F	?

ASCII	Hex	Symbol
64	40	@
65	41	A
66	42	B
67	43	C
68	44	D
69	45	E
70	46	F
71	47	G
72	48	H
73	49	I
74	4A	J
75	4B	K
76	4C	L
77	4D	M
78	4E	N
79	4F	O

ASCII	Hex	Symbol
80	50	P
81	51	Q
82	52	R
83	53	S
84	54	T
85	55	U
86	56	V
87	57	W
88	58	X
89	59	Y
90	5A	Z
91	5B	[
92	5C	\
93	5D	]
94	5E	^
95	5F	_

ASCII	Hex	Symbol
96	60	`
97	61	a
98	62	b
99	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6A	j
107	6B	k
108	6C	l
109	6D	m
110	6E	n
111	6F	o

ASCII	Hex	Symbol
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x
121	79	y
122	7A	z
123	7B	{
124	7C	|
125	7D	}
126	7E	~
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	-Added Bluetooth Setting -Updated PortSettings
Jul.14 2014	3.12.0	- Added getStarIOVersion method
Mar. 17 2014	3.10.0	- Added getFirmwareInformation method - Added setEndCheckedBlockTimeoutMillis method
Dec. 24 2013	3.8.0	- Added compressRasterData method - Depricated generateBitImageCommand method
Jul. 3 2013	3.7.0	- Added Star Printer Status List
Feb. 25 2013	3.4.0	- Added generateBitmapCommand support - Added SM-T300i, SM-T400i
Jan. 18 2013	3.3.0	- Added SM-S210i - Updated Raster Graphical Text Printing, - Updated Begin / End Checked Block method
Nov. 30 2012	3.2.0	- Added Bluetooth support - Added Begin/End Checked Block support
Jul. 27 2012	3.1.0	- Changed SDK UI to tree structure - Added Port Discovery support - Added Apple AirPort Express support (POS Printers Only) - Added Sample Receipt Printing support
May. 31 2012	2.4.0	- Updated default StarIO Port Class name - Revised default order of StarIOPort JAR to support Android SDK r17 and higher
Mar. 15 2012	2.3.0 Beta	- Added USB Printing Support (POS Printers Only)
Jan. 18 2012	2.2.0 Beta	- Added Raster Image Printing Sample
Jan. 12 2012	2.1.0	- Added Japanese Text Formatting Sample - Added Japanese Manuals
Oct. 14 2011	2.0.0	- Updated Sample Printing
Jun. 03 2011	1.0.0	- Initial Release

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/