Java Software Development

How to Use StarIO in Java

Thermal Line Mode Printing

This SDK contains a Java NetBeans 6.9.1 Project that shows how to code StarIO commands to print to one of many Star Micronics Thermal POS Printers.

Compatible Star Printer Models:

• FVP10	(Ver.1.0 or later)
• HSP7000	(Ver.1.0 or later)
• TSP650	(Ver.2.0 or later)
• TSP700II	(Ver.2.0 or later)
• TSP800II	(Ver.1.0 or later)
• TSP800Rx	(Ver.4.3 or later)
• TSP828	(Ver.4.6 or later)
• TUP500	(Ver.1.0 or later)
• TUP900	(Ver.1.2 or later)

Supported Interfaces:

- Serial
- Parallel
- Ethernet
- USB (Printer Class & Vendor Class)

Functions include:

- Sample Receipt
- Read Printer/Drawer Status
- Code Pages
- 1D Barcodes
- 2D Barcodes
- Drawer Kick
- Check Block (ETB)
- Text Formatting
- Set Printer Font
- Line Feed & Cutting
- Text Formatting

Requirements: NetBeans 6.9.1 or later and the latest Java JDK/JRE

NOTE:

- This sample program provides source code and dependencies for 32-bit & 64-bit.
- Executable files for 64-bit cannot be executed in a 32-bit environment but 32-bit programs can run on 64-bit operating systems. This is set as 32-bit by default.
- This is not a JavaPOS sample and will not work with printers that do not support StarIO Line Mode commands.

Table of Contents

About this Manual

This manual is designed to help you understand StarIO and how to build a Java application to interact with Star Micronics Thermal Line Mode Printers. It is important to understand the basics of the Java language and have the Java JDK/JRE installed. The SDK was built in the IDE NetBeans, but can be used with other IDEs. The Java syntax used in this SDK is all within the code base Main.java which is in the “src” folder. Check the Developers section of our site for the latest updated SDKs, technical documentation, FAQs, and more.

This is not a JavaPOS example but a more robust SDK built for Star Thermal POS Printers.

Key Legend:

Warning	⚠	These will explain some potential issues.
Avoid Doing This	🚫	Explains things to avoid.
Note	💭	Provides important information and tips.

CAUTION:

- 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.

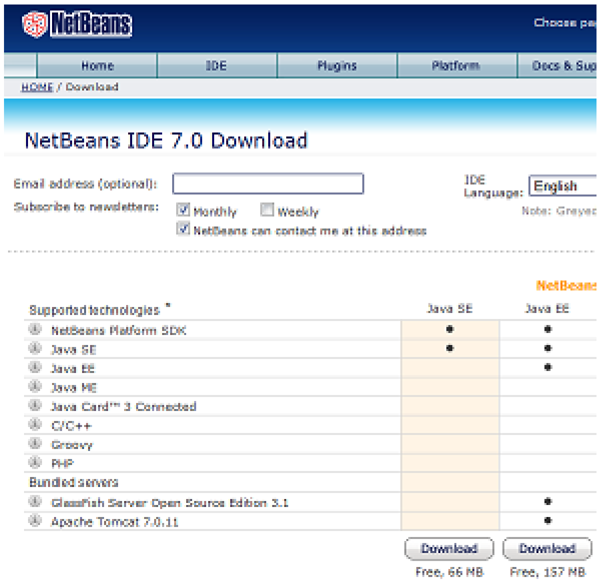
How to compile and run the Java SDK

This section will explain:

1. How to open the Java NetBeans 6.9.1 Project.
2. Compiling the project.
3. Running the project.

How to open the Java SDK project:

You must have NetBeans and Java JDK installed.

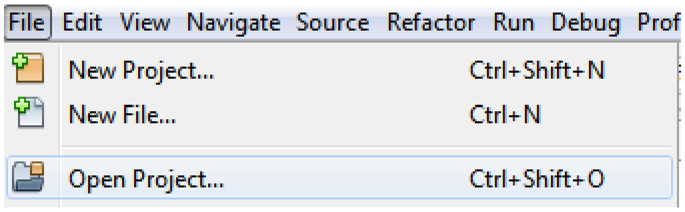


NetBeans IDE 7.0 Download

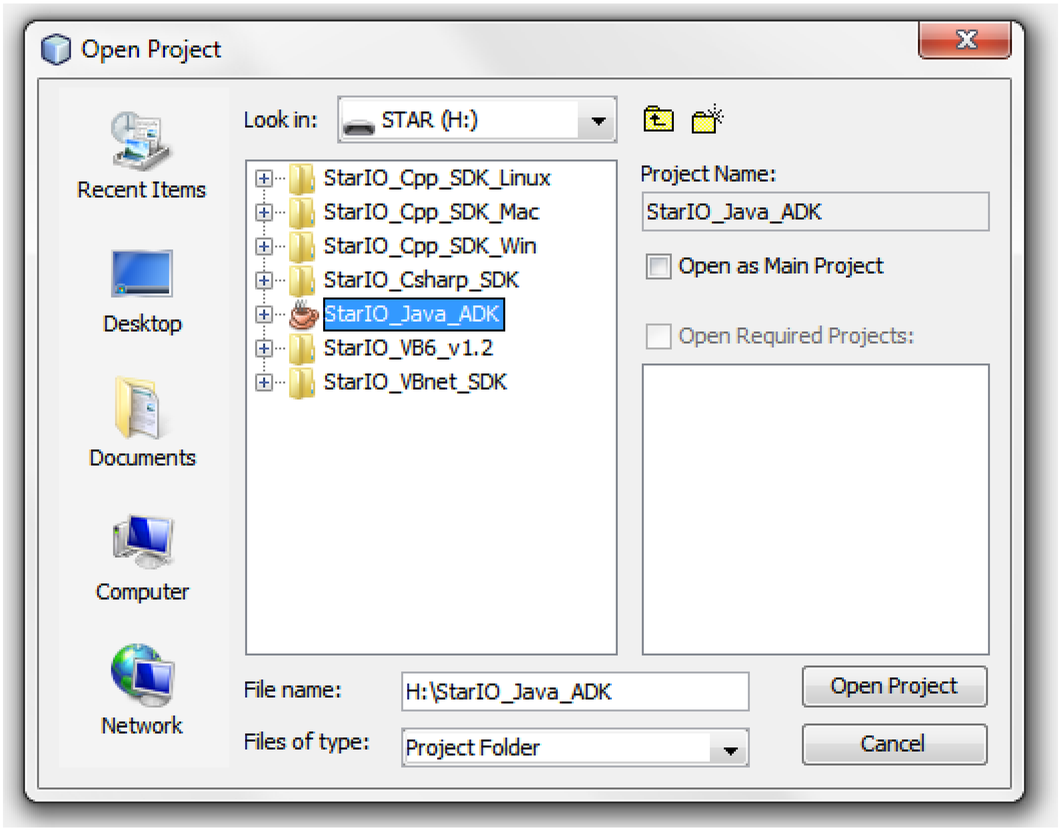
Email address (optional):
Subscribe to newsletters: [✓ Monthly] Weekly
[✓ NetBeans can contact me at this address]
IDE Language: English
Note: Greyec

Supported technologies *	Java SE	Java EE
NetBeans Platform SDK	●	●
Java SE	●	●
Java EE		●
Java ME		
Java Card™ 3 Connected		
C/C++		
Groovy		
PHP		
Bundled servers		
GlassFish Server Open Source Edition 3.1		●
Apache Tomcat 7.0.11		●
	Download	Download
	Free, 66 MB	Free, 157 MB

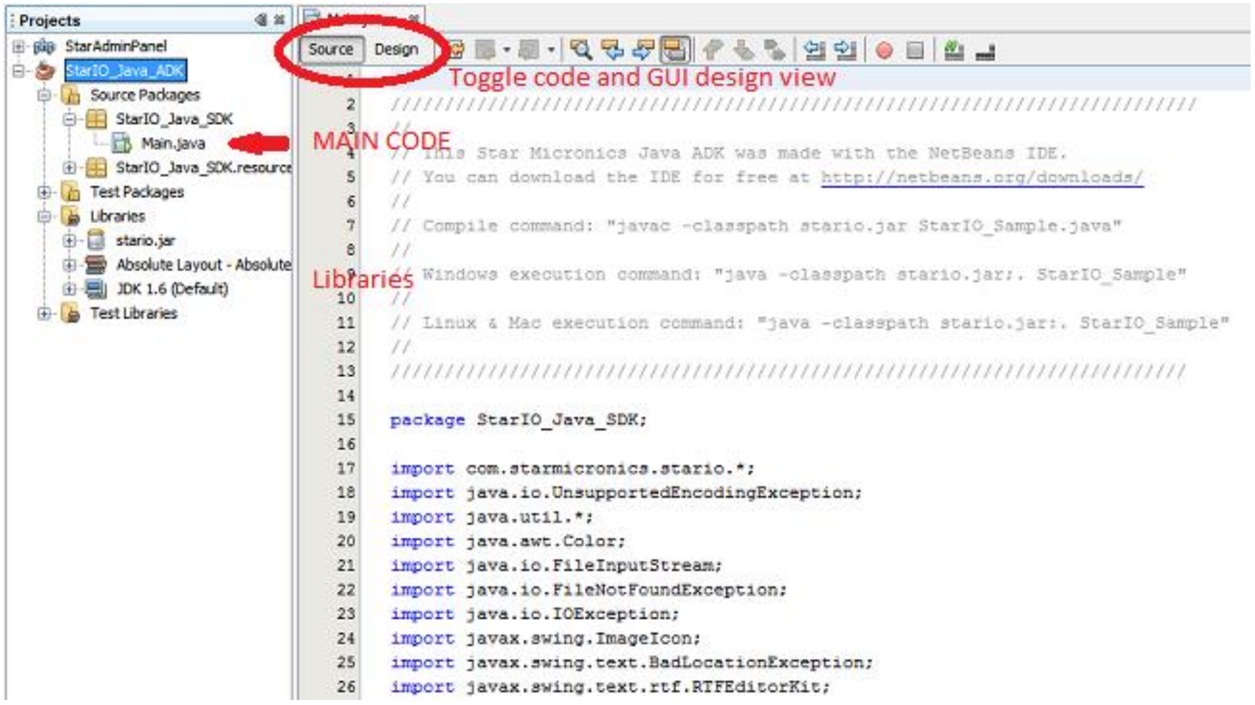
Open the NetBeans IDE and click File > Open Project…



Navigate to where you exported the StarIO_Java_SDK package, highlight the project folder, and click "Open Project".



To view the code, expand source packages/StarIO_Java_SDK and open up Main.Java

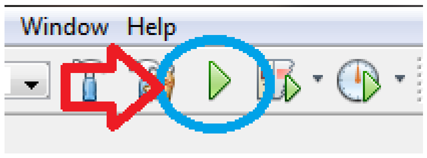


```
//////////////////////////////////////////////////////////////////////////////
//
// This Star Micronics Java ADK was made with the NetBeans IDE.
// You can download the IDE for free at http://netbeans.org/downloads/
//
// Compile command: "javac -classpath stario.jar StarIO_Sample.java"
//
// Windows execution command: "java -classpath stario.jar;. StarIO_Sample"
//
// Linux & Mac execution command: "java -classpath stario.jar:. StarIO_Sample"
//
//////////////////////////////////////////////////////////////////////////////

package StarIO_Java_SDK;

import com.starmicronics.stario.*;
import java.io.UnsupportedEncodingException;
import java.util.*;
import java.awt.Color;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import javax.swing.ImageIcon;
import javax.swing.text.BadLocationException;
import javax.swing.text.rtf.RTFEditorKit;
```

You can then use the code view or GUI view to find blocks of function code to include in your Java projects. You can double click buttons within the GUI designer to jump to a code block.

Compiling the project:

Click on the menu item “Run” and then click “Run Project” or hit F6.



You may want to do a clean build which can also be done from the menu item “Run”.

This project was built using a Windows 7 32-bit machine and although Java doesn’t need to know what operating system you’re running on, you will need to determine whether your target machine will be a 32-bit or 64-bit operating system. Once you know, you can then include the dependencies for that OS. You can find these under the folder called “dependencies” where you find respective StarIO builds for 32- and 64-bit environments.

Using the SDK with Star Micronics Printers

Please make sure you have a compatible Star Micronics Thermal Line Mode Printer model.

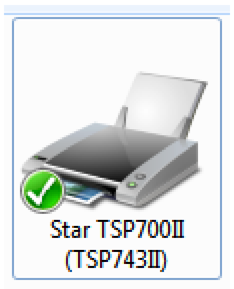
Port Name and Interface Relation:

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.

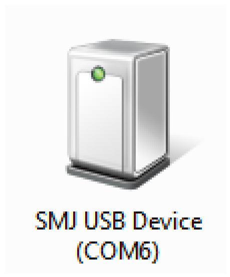
Interface	Port Name	Port Settings
Serial	COM*n*	9600,n,8,1,h
Parallel	LPT*n*	N/A
USB (Vendor Class)	usbven:COM*n*	N/A
USB (Printer Class)	usbprn:"Queue Name"	N/A
Ethernet (TCP/IP)	tcp:"IP Address"	N/A

NOTE: If using a ***USB*** interface and the printer is in ***printer class mode***, after successfully installing the printer driver, you will have a Printer Queue Name to use in the Port Name. Put ***"usbprn:Star TSP700II (TSP743II)"*** as the Port.

Printer Class Mode



Vendor Class Mode



If using a ***USB*** interface and the printer is in ***vendor class mode***,
a port number is not required. Just put "usbven:" as the Port Name.

"LPTn"	n = your port number (1, 2, 3, 4 etc)
"COMn"	n = your port number (1, 2, 3, 4 etc)
"tcp:192.168.222.244"	Enter TCP IP Address of the Ethernet printer.

Overview of how this Java SDK is designed

This overview will touch briefly on key components of the SDK and how to find them.

Focus on the file "Main.java" which contains all the business logic and StarIO commands.

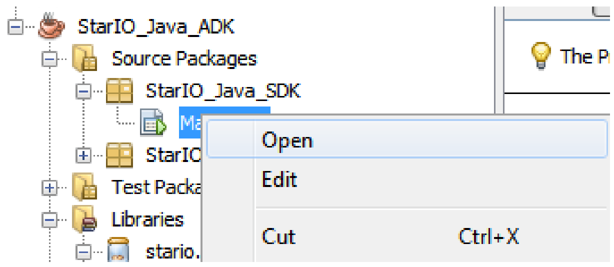
To navigate to a specific command is easy; simply right click on the file called "Main.java" in the solution explorer. Then click on the option "Open" which will open the designer view.

So let us say for example you wanted to find out what code block is executed for the "Print Sample Receipt" button.

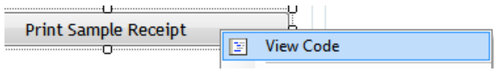
Double click on the button in the designer view. This will switch to code view and show the code block handling the mouse click event for the "Print Sample Receipt" code.

With this tip known, you can now fully explore all of the advanced StarIO commands with ease.

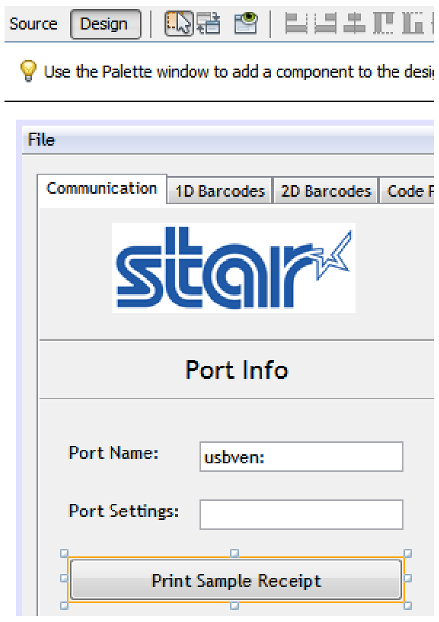
Please note that there is a stopping point in the code where everything below the line "CODE BELOW THIS POINT IS NOT USEFUL AS SDK EXAMPLES" is not really useful as code samples because they are related specifically to this SDK program. This code is mostly for the UI so please do not get confused into including this code into your business application.



Click "Open" to view the code in the designer.



Right Click Button, then click "View Code"



You can double click on a button in design mode to move to that block of code instantly.

This SDK contains a folder named "Documentation" with helper RTF documents for the program to load into the right help window. If these files / folder are moved and/or renamed from this folder, the RTF files will not get loaded correctly and the SDK will not display this helpful info.

StarIO – (StarIOPort.dll/StarIOJ.dll & StarIO.Jar)

How to include StarIO into your project:

The file StarIO.jar is a library that you can include into your Java projects to expose StarIO methods. The file StarIOJ.dll is also needed to point to StarIOPort.dll.

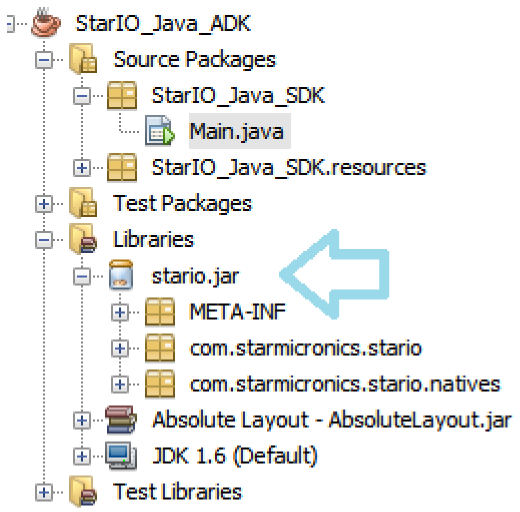
To include this library into your project:

1. Create a new NetBeans Java project
2. Choose Add JAR/Folder
3. Navigate to the Java StarIO SDK folder
4. Click the "Dependencies" folder attached with this SDK and select the folder "32" or "64" based on your target platform.
5. Click add existing item, then "StarIO.Jar"
6. To expose StarIO, add

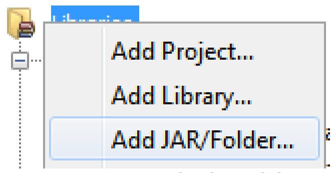
   "import com.starmicronics.stario.*;"

   at the top of your main code.
7. Now you can access all of StarIO's methods!

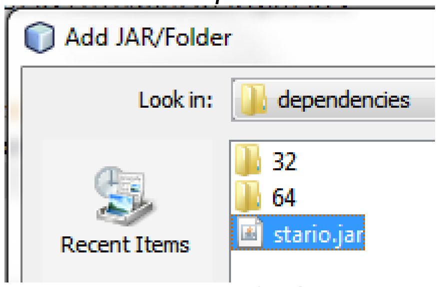
Please note that if you want to make your code portable, copy and paste the libraries into your project first. This will avoid and issues once opened on another computer.



StarIO being used as a Reference



Step 2 click Add Jar



Step 3 find stario.jar

WARNING: Make sure StarIOJ.dll and StarIOPort.dll are in the same directory as StarIO.jar. This file will link itself to StarIOJ.dll which then links to StarIOPort.dll by looking in the same folder that the StarIO.jar is located in.

Configuring your project to x64 or x86 with StarIO:

Compiling your project with the correct StarIO is very important to get the maximum speed from your CPU. Your main two choices are 32-bit (x86) and 64-bit (x64) operating systems.

1. To switch this project to a 64-bit project, remove the StarIO library from the Library folder and then right click again and select -Add Jar/Folder…
2. Navigate to the folder where the 64-bit StarIO is stored "Java_StarIO_SDK/dependencies/64/"
3. Select the file "stario.jar"
4. You now have connected the 64-bit compiled JAR StarIO into your project.

WARNING: Please make sure to include the correctly compiled binary for the Java project. If you select the wrong one (64-bit even thought the project will run on 32-bit computers) and try to execute function, it may seem like a communication issue but rather it's a bad library issue.

StarIO Methods Overview:

Class Variables include 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. "usbven:" "usbprn:TSP650" "tcp:192.168.1.2" "COM4" "LPT1"

portSettings is what you will use for configuring Serial connections correctly.

Ex. "9600,n,8,1,h"

The following are the acceptable inputs from left to right:

baud: 38400, 19200, 9600, 4800, 2400

parity: n, e, o

data-bits: 8, 7

stop-bits: 1

flow-ctrl: n, h

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)

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 in Java:

```
StarIOPort port = null;
String portName = txtPortName.getText();
String portSettings = txtPortSettings.getText();
String status = "";

try {
    port = StarIOPort.getPort(portName, portSettings, TIMEOUT);
} catch (StarIOPortException e) {
    status += "Failure to print sample\nCheck printer connection.";
    // e  .printStackTrace();
} finally {
    if (port != null) {
        StarIOPort.releasePort(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.

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 try and read raw status.
Use GetOnlineStatus or GetParsedStatus 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.

Always release (close) ports that you get (open).
Leaving a port open will cause future calls to open the port to fail.

WritePort

```
public int 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.

private static int WritePortHelper(StarIOPort port, byte[] writeBuffer) throws StarIOPortException {
      int zeroProgressOccurances = 0;
      int totalSizeCommunicated = 0;

      while ((totalSizeCommunicated < writeBuffer.length) && (zeroProgressOccurances < 2))
      {
         int sizeCommunicated = port.writePort(writeBuffer, totalSizeCommunicated,
writeBuffer.length - totalSizeCommunicated);

         if (sizeCommunicated == 0) {
            zeroProgressOccurances++;
         } else {
            totalSizeCommunicated += sizeCommunicated;
            zeroProgressOccurances = 0;
         }
      }
      return totalSizeCommunicated;
   }
```

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.

Returns:

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

Throws:

`StarIOPortException` - when a communication failure occurs

BeginCheckedBlock

public StarPrinterStatus **BeginCheckedBlock**()
throws StarIOPortException

This method initiates a checked block printing operation and returns the device's detailed status.

Returns:

StarPrinterStatus structure giving the current device status - don't bother printing if the printer is offline

Throws:

`StarIOPortException` - when a communication failure occurs

EndCheckedBlock

public StarPrinterStatus **EndCheckedBlock**()
throws StarIOPortException

This method ends a checked block printing operation and returns the device's detailed status. This function does not return until either the printer has successfully printed all data or has gone offline in error. If the StarPrinterStatus structure indicates that the printer is online upon return than all data was successfully printed.

Returns:

StarPrinterStatus structure giving the current device status - if it's offline then printing failed

Throws:

`StarIOPortException` - when a communication failure occurs

Here is an **example** usage of **BeginCheckedBlock** and **EndCheckedBlock** methods:

```
try {           //Open the port
        port = StarIOPort.getPort(portName, portSettings, TIMEOUT);


        StarPrinterStatus starPrinterStatus; //Get status
```

```
            starPrinterStatus = port.beginCheckedBlock(); //Begin checked block

            if (starPrinterStatus.offline == true) {
                status += "Printer is Offline";
                return;
            }

            byte[] data = "Hello World!\n".getBytes();

            //For loop to print 50 Hello world lines and end the check block
after to make sure all the data has been received
            for (int count = 0; count < 50; count++) {
                int totalSizeCommunicated = WritePortHelper(port, data); // see
below
                if (totalSizeCommunicated != data.length) {
                    status += "Error! Could not write all data to the printer”;
                    return;
                }
            }

            //END CHECK BLOCK
            starPrinterStatus = port.endCheckedBlock();

            if (starPrinterStatus.offline == true) {
                status += "The printing failed to send all bytes";
            }

            status += "Successfully sent all data to printer!”;
        } catch (StarIOPortException e) {
            status += "Failure to Check Block";
        }
```

ResetDevice

public void **ResetDevice**()
throws StarIOPortException

This method resets the device at the hardware level.

Throws:

`StarIOPortException` - when a communication failure occurs

GetOnlineStatus

public boolean **GetOnlineStatus**()
throws StarIOPortException

This method returns a Boolean value if the printer is online or offline.

Returns:

Boolean value: true = printer is online false = printer is offline

Throws:

`StarIOPortException` - when a communication failure occurs

GetParsedStatus

public StarPrinterStatus **GetParsedStatus** ()
throws StarIOPortException

This method retrieves detailed status form 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 printerStatus = port.GetParsedStatus();

If(printerStatus.Offline == false)
{
     If(printerStatus.CompulsionSwitch == true){
          //Cash drawer is open
     }
     Else{
          //Cash drawer is closed
     }
}
Else{
     //If True, then the printer is offline.
}
```

There are different statuses that are pulled when you initialize **StarPrinterStatus**.

This is a list of statuses that are in the class structure StarPrinterStatus:

CoverOpen returns a Boolean.

Offline returns a Boolean.

CompulsionSwitch returns a Boolean.

OverTemp returns a Boolean.

UnrecoverableError returns a Boolean.

CutterError returns a Boolean.

MechanicalError returns a Boolean.

HeadThermistorError returns a Boolean.

ReceiveBufferOverflow returns a Boolean.

PageModeCommadError returns a Boolean.

BlackMarkError returns a Boolean.

PresenterPaperJamError returns a Boolean.

HeadUpError returns a Boolean.

VoltageError returns a Boolean.

ReceiptBlackMarkDetection returns a Boolean.

ReceiptPaperEmpty returns a Boolean.

ReceiptPaperNearEmptyInner returns a Boolean.

ReceiptPaperNearEmptyOuter returns a Boolean.

PresenterPaperPresent returns a Boolean.

PeelerPaperPresent returns a Boolean.

StackerFull returns a Boolean.

slipTOF returns a Boolean.

slipCOF returns a Boolean.

slipBOF returns a Boolean.

validationPaperPresent returns a Boolean.

slipPaperPresent returns a Boolean.

ETBAvailable returns a Boolean.

ETBCounter returns a Byte.

PresenterState returns a Byte.

RawStatus returns a Byte[] array.

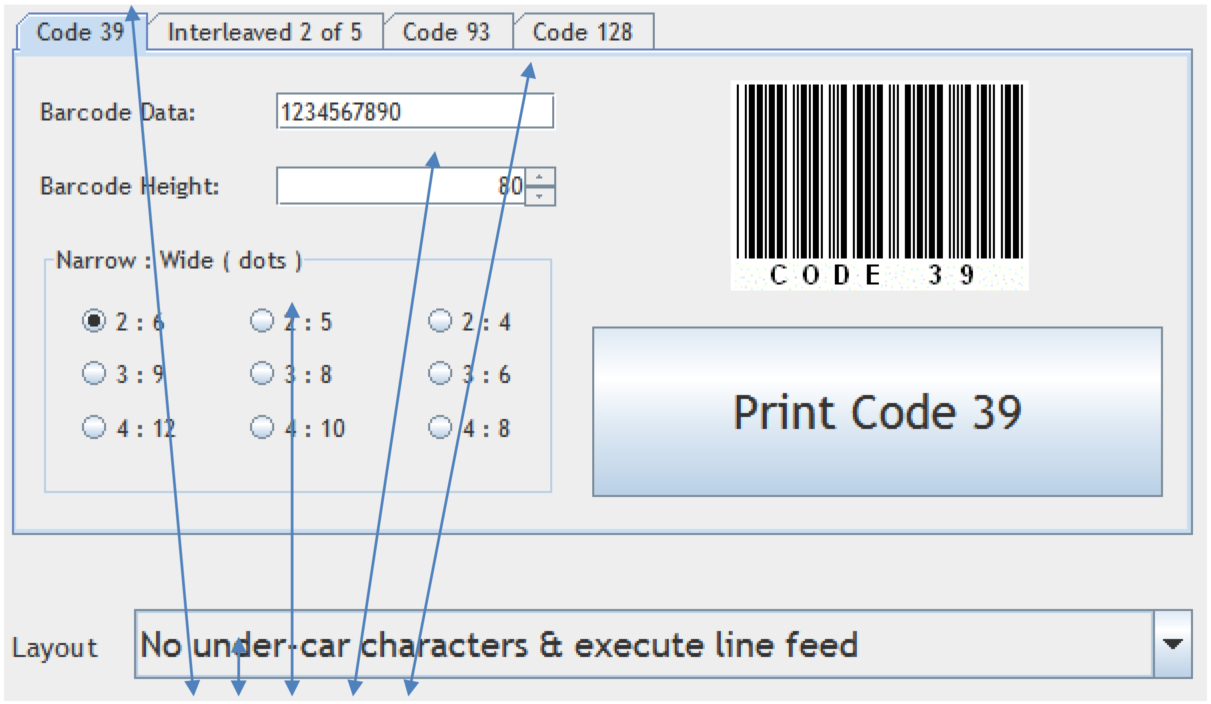
Functionality

StarIO Printer Commands

All of these commands can be found in the Star Thermal Line Mode Spec Manual.

The Java SDK also has page and section references to this document for more information so please download that manual and study it if you need more detail on a specific command.

1D Barcodes



Barcode Data: 1234567890

Barcode Height: 80

Narrow : Wide (dots)
(● 2 : 6) ○ 2 : 5 ○ 2 : 4
○ 3 : 9 ○ 3 : 8 ○ 3 : 6
○ 4 : 12 ○ 4 : 10 ○ 4 : 8

Layout: No under-car characters & execute line feed

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

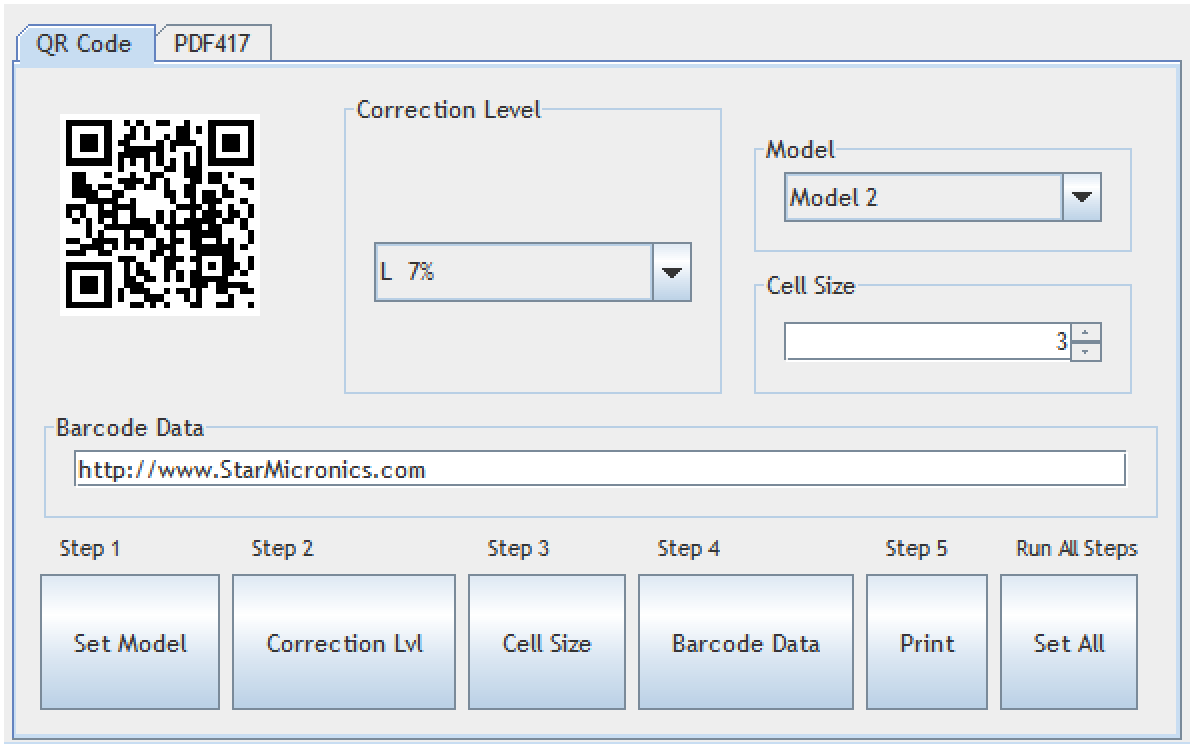
n2 = Under-bar character selection and added line feed selection
1 = No added under-bar characters & Executes line feed after printing barcode
2 = Adds under-bar characters & Executes line feed after printing barcode
3 = No added under-bar characters & doesn't line feed after printing barcode
4 = Adds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

n3 = Barcode mode selection specifies the size of the narrow and wide barcode lines

n4 = Barcode height (dot count)

2D Barcodes

QR Code



Correction Level: L 7%

Model: Model 2

Cell Size: 3

Barcode Data: http://www.StarMicronics.com

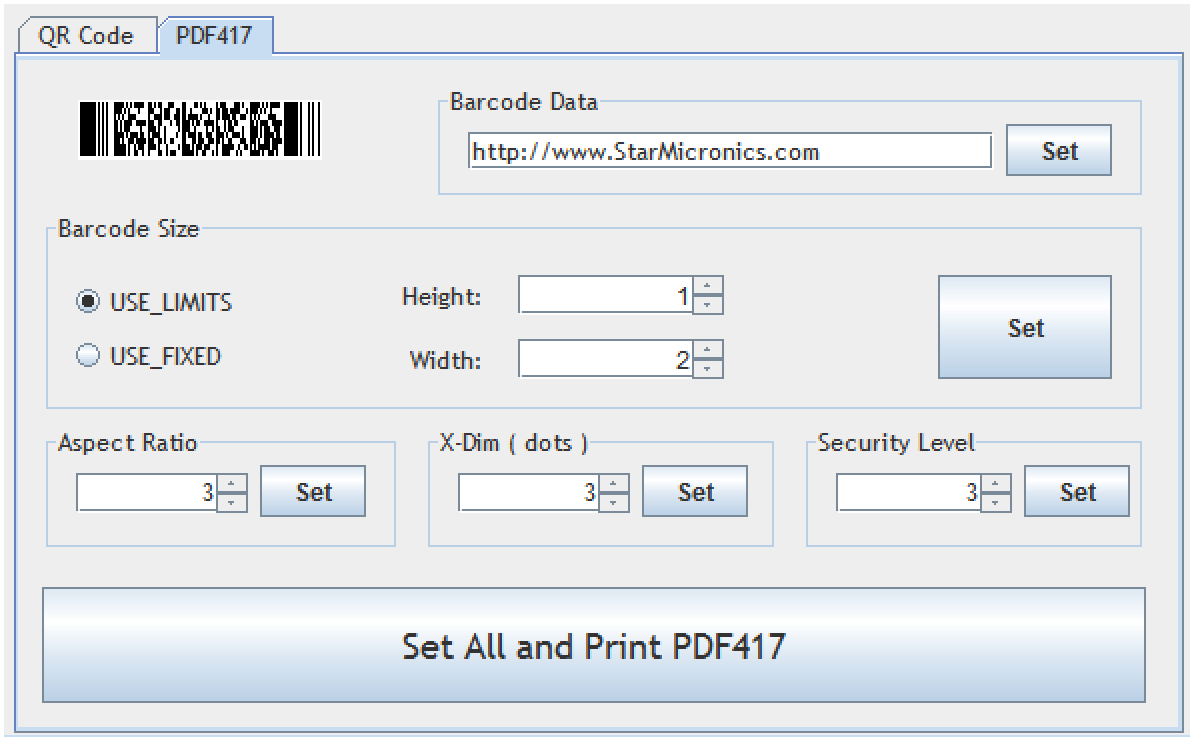
There are 5 commands below that are very important to printing a good QR Code.

(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 *n*L *n*H *d1…dk*
(5) Print the QR Code	ESC GS y P

Here is the order in which commands need to be sent to the printer for it to print the QR code.

QR Model + QR Correction Level + QR Cell Size + QR Data + Print QR Code

PDF417



Barcode Data: http://www.StarMicronics.com

Barcode Size: [✓ USE_LIMITS] USE_FIXED

Height: 1

Width: 2

Aspect Ratio: 3

X-Dim (dots): 3

Security Level: 3

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

Here is the order in which commands need to be sent to the printer for it to print the PDF417.

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

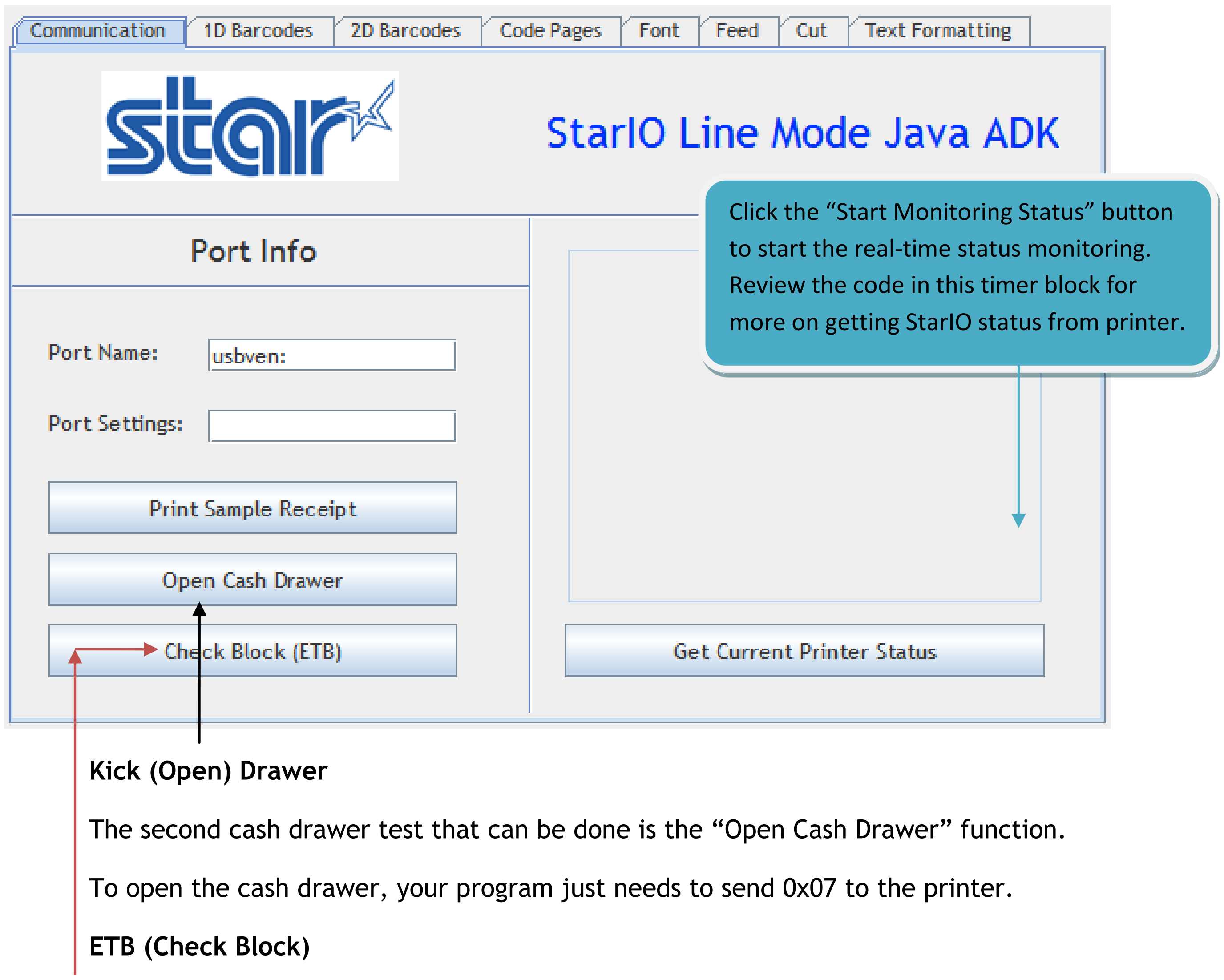
Cash Drawer

There are 2 examples that involve the cash drawer.

First is the real-time monitoring function located on the bottom right.

Second is the Open Cash Drawer function located on the bottom left.

Check Drawer Status


Communication
1D Barcodes
2D Barcodes
Code Pages
Font
Feed
Cut
Text Formatting
StarIO Line Mode Java ADK
Click the "Start Monitoring Status" button to start the real-time status monitoring. Review the code in this timer block for more on getting StarIO status from printer.
Port Info
Port Name:
usbven:
Port Settings:
Print Sample Receipt
Open Cash Drawer
Check Block (ETB)
Get Current Printer Status

Kick (Open) Drawer

The second cash drawer test that can be done is the "Open Cash Drawer" function.

To open the cash drawer, your program just needs to send 0x07 to the printer.

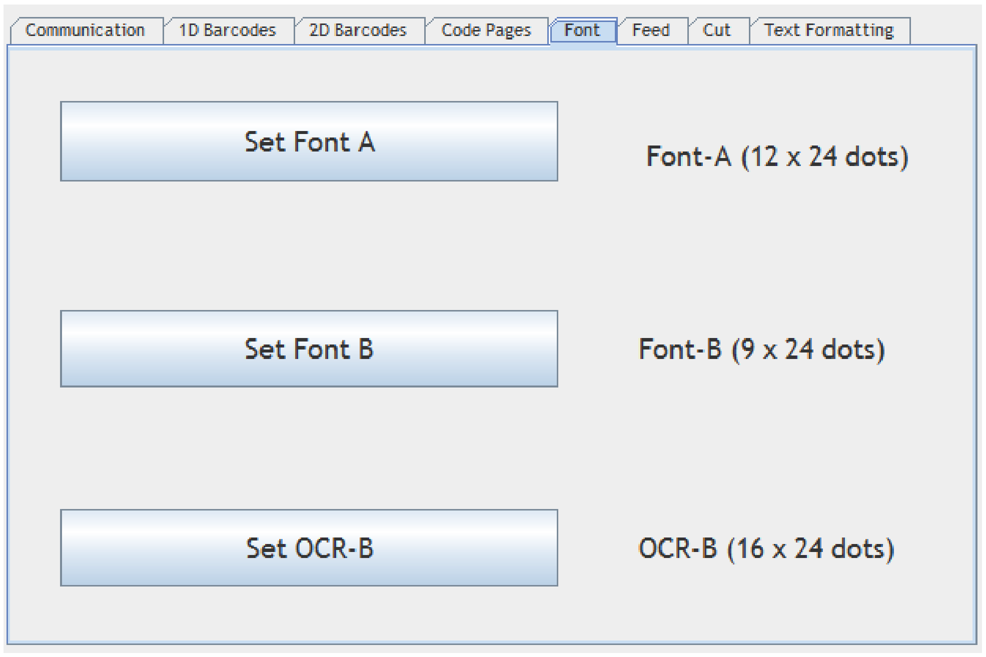
ETB (Check Block)

The ETB will return 0 from the printer once the print job has been completed.

Change Font

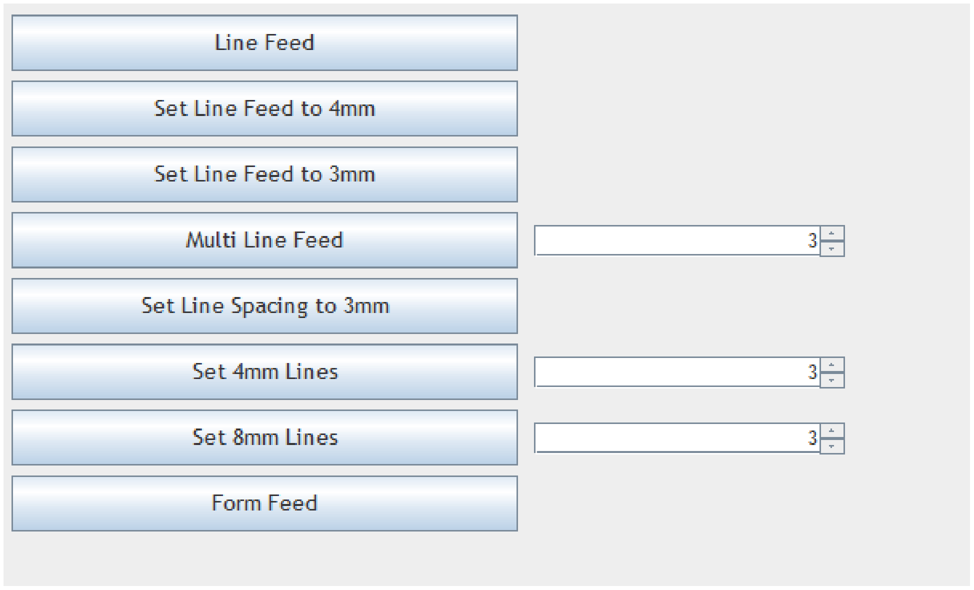
Changing the font on the printer can be done with the following commands.

ESC RS F n n = 0 for A, 1 for B, 10 for OCR-B

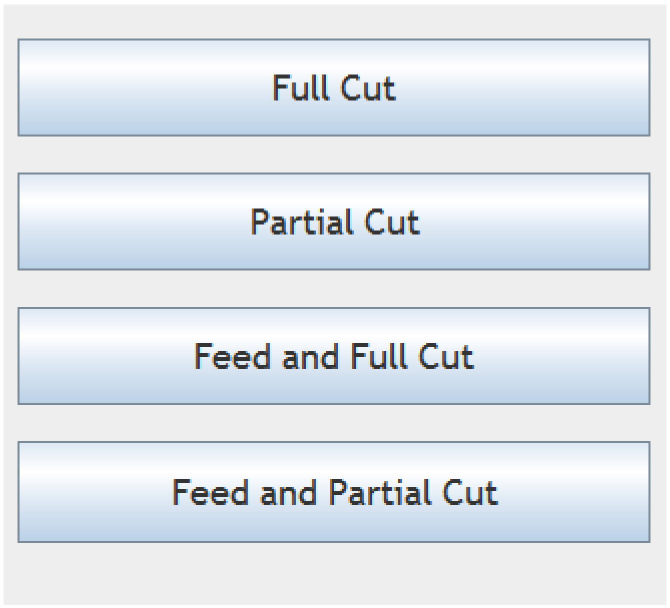


Feed

The feed commands are very straight forward. Use LF for best results.



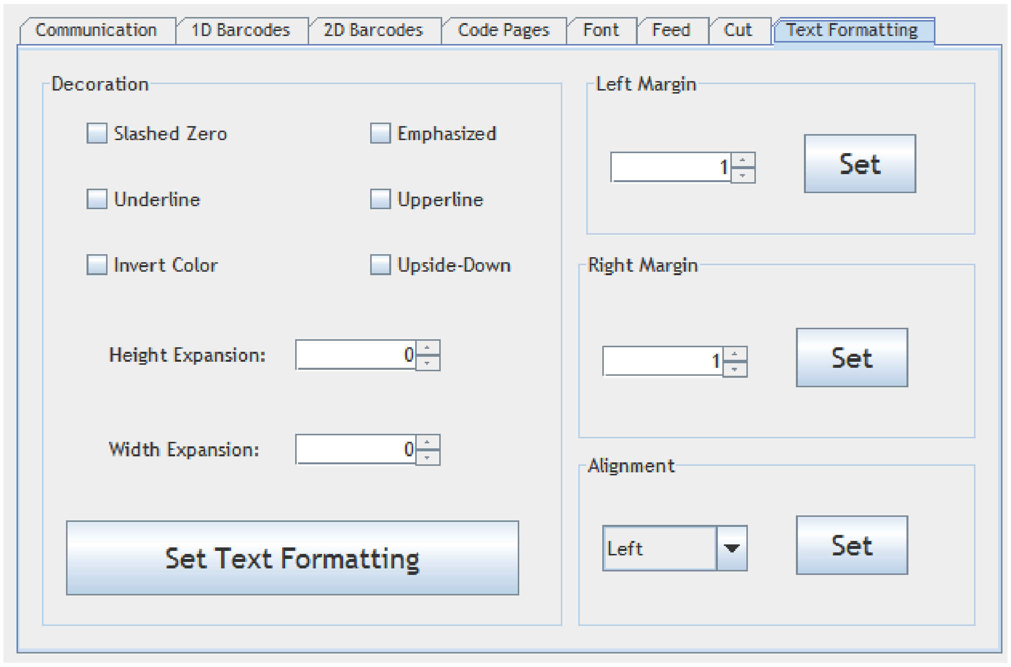
Cut



Partial Cut	ESC d 1 or 3
Full Cut	ESC d 0 or 2

Text Formatting

The following are all Text Decoration or formatting related.



Height Expansion: 0

Width Expansion: 0

Left Margin: 1

Right Margin: 1

Alignment: Left

Slashed Zero	ESC / n	
Underline	ESC - n	
Upperline	ESC _ n	
Invert Color (B/W)	ESC 4	
Emphasized (Bold)	ESC E = on	ESC F = off
Upside-Down	0F = Start	DC2 = off

Character Expansion

Width	ESC W n	0 ≤ n ≤ 255
Height	ESC h n	0 ≤ n ≤ 255

Set Left Margin

ESC l n	0 ≤ n ≤ 255

Set Right Margin

ESC Q n	0 ≤ n ≤ 255

Alignment

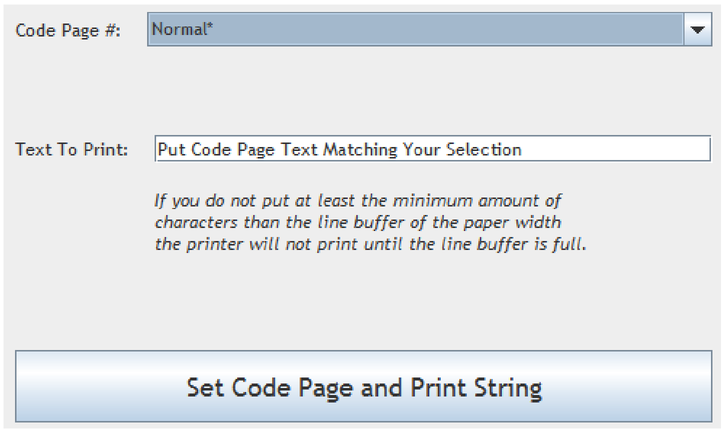
Left	ESC GS a 0
Center	ESC GS a 1
Right	ESC GS a 2

Horizontal Tabbing

This is covered in the sample receipt that can be printed, please review that code block or page 3-29 of the Thermal Line Mode Manual.

Set Horizontal Tab	ESC D n1 n2 ... nk NUL
Clear Horizontal Tab	ESC D NUL
Move Horizontal Tab	HT

SBCS Code Pages


Code Page #:
Normal*
Text To Print:
Put Code Page Text Matching Your Selection
Set Code Page and Print String

If you do not put at least the minimum amount of characters than the line buffer of the paper width the printer will not print until the line buffer is full.

Currently this SDK only supports fonts that are built into the Java virtual machine. There is a list of fonts that the JVM supports but luckily for you we have already done the hard work by comparing Star supported character sets to the Java sets.

If you hit the drop down box you will see certain code pages that show that the JVM do not support, so in those cases you will not be able to copy and paste these code page character sets into any Java program.

To set a code page on the printer:

ESC GS 74 n

n = The Code Page Selection Index

Stored Logo Printing

Stored logo printing is done in the sample receipt. Please review that and the Thermal Line Mode Manual for more information.

ESC FS p 1 0

Getting Online Status of the Printer

Visit the function code block called **printToPrinter** and there you will see the StarIO method **GetOnlineStatus** being used to retrive a boolean value for online status.

True = Online
False = Offline

Getting Parsed Status of the Printer

Review the function code block called **timerGetStatus_Tick** and there you will see the StarIO method **GetParsedStatus** being used to pull a struct down of all the potential status flags the printer can throw. Click here for the full list of statuses.

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: StarIO, ESC/POS, UPOS: JavaPOS, POS for .NET, & OPOS are all different ways to communicate with the printer. Visit our Developers' section for more info on these. This SDK covers StarIO only.

Additional Resources

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

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
- Code Snippets
- Star Micronics Printer Drivers
- Technical Questions/Support

Download the Star Thermal Line Mode Command Spec Manual

Use it as your reference for all StarIO Line Mode commands!

Character Encoding in Java

If you don't know what ASCII and Unicode is, this is a good place to start.

Oracle Java Internationalization

Good resource for more detail on internationalization.

New to Java Developer Center

Great place to learn more about the Java language.

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.

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!

Version	Release Date
1.0.0	Jul 2011

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.
1150 King Georges Post Road
Edison, NJ 08837
USA
1-800-782-7636
+1-732-623-5500
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-(0)-2-631-1161 x 2
http://www.starmicronics.co.th/