iOS Software Development Kit

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

This SDK contains an Xcode Objective-C project for iOS devices.

Tools Needed:

- Xcode 7.0 or later
- StarIO iOS SDK

To use StarIO.framework version 3.14.0 or later, you need to add the following frameworks.

・External Accessory framework

・Core Bluetooth framework

- To upgrade StarIO.framework when you are using version 3.13.1 or earlier, you need to add Core Bluetooth framework into your project.

Please refer to here for more information.

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 emulation modes**

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 mode 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.

StarIO SDK Compatibility OS : iOS 7.0 or later

StarIO SDK Compatibility Chart

Device	CPU
iPad 2	Armv7
iPad (3rd Generation)	Armv7
iPad (4th Generation)	Armv7s
iPad Air	Arm64
iPad Air 2	Arm64
iPad mini	Armv7
iPad mini 2	Arm64
iPad mini 3	Arm64
iPad mini 4	Arm64
iPad Pro	Arm64
iPhone 4s	Armv7
iPhone 5	Armv7s
iPhone 5s	Arm64
iPhone 5c	Armv7s
iPhone 6	Arm64
iPhone 6 Plus	Arm64
iPhone 6s	Arm64
iPhone 6s Plus	Arm64
iPod touch (5th Generation)	Armv7
iPod touch (6th Generation)	Arm64

Note: iPad, iPhone, iPod, and iPod touch are trademarks of Apple Inc., registered in the U.S. and other countries. iPad Air and iPad mini are trademarks of Apple Inc. iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

Table of Contents

About this Manual

This manual is designed to help you understand StarIO and how to build an iOS application to interact with Star Micronics Portable Printers. It is important to understand the basics of the Objective-C language. Although this SDK is for iOS, there are SDKs available for many different operating systems and programming languages at our website in the Developers section. 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:

- 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.

Star Printer Compatibility Chart

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

Star Printer		iOS	Status	Get Firmware Information	Sample Receipts	1D Barcodes	2D Barcodes	Text Formatting	Raster Graphics	Raster Graphics Text Printing	Image File Printing	MSR	Bluetooth Setting
Model	**Interface**	**6.0 or later**											
SM-T300	Wireless LAN	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
SM-S220i (Excluding Jp model)	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓*
SM-S210i (Jp model only)	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*
SM-S230i (Excluding Jp model)	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*
SM-T300i	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*
SM-T400i	Bluetooth	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓*

* 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 Portable Printer Programming Manual.

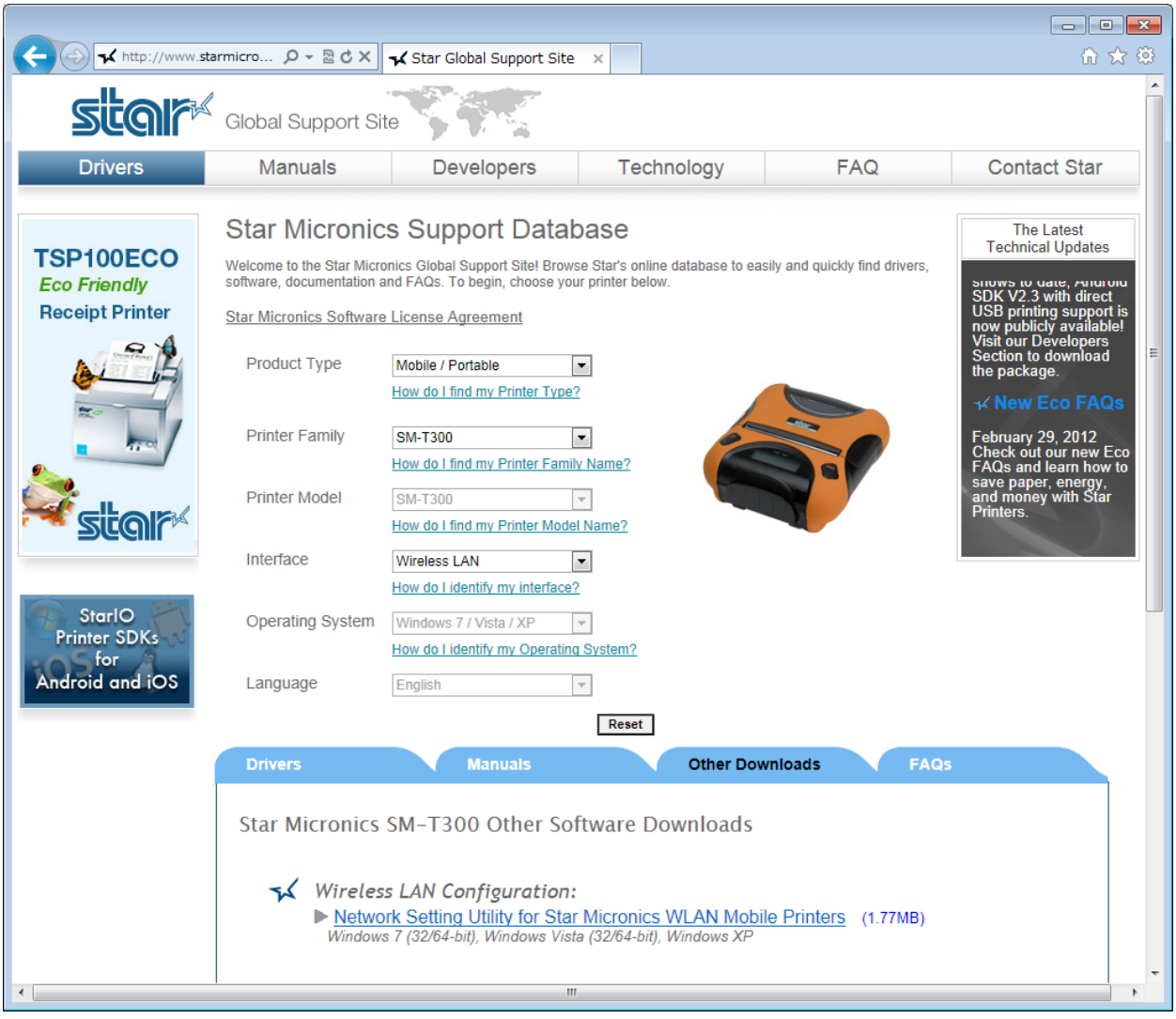
Connecting a Star Portable Printer to an iOS Device

Wireless LAN

Star Portable Printers ship with DHCP enabled by default. If your network supports DHCP, be sure to make the necessary configurations so that your Portable Printer will automatically get an IP Address.

◆How to Assign a Static IP Address

If you would like to assign a static IP Address to your Portable Printer, a Windows PC and Star's WLAN Setting Utility is necessary. The utility and its installation instructions are available from Star's Global Support Website. To download the package, select "Mobile/Portable Printer" from the dropdown and then your printer model. The package is available in the "Other Downloads" tab.



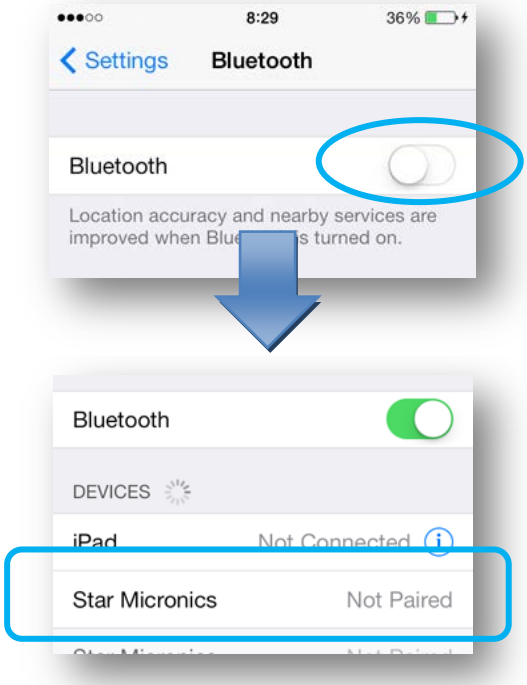
Bluetooth

◆Pairing

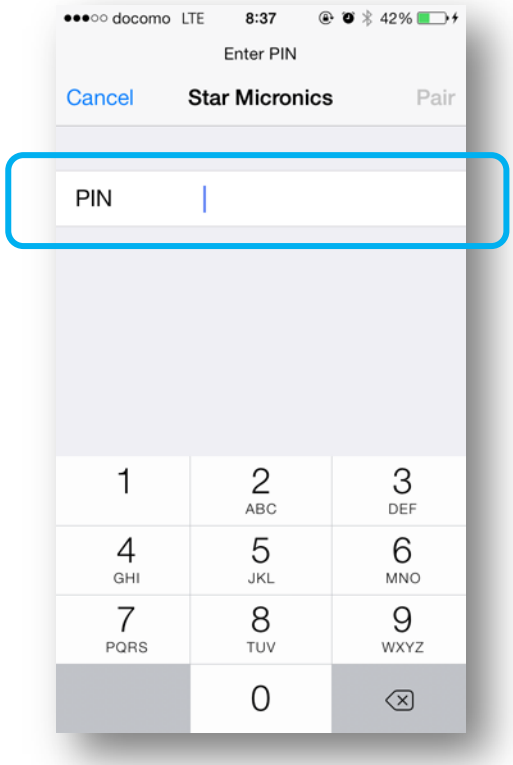
We recommend that you do not pair an iOS device with multiple Star mobile printers at the same time.

1. Place your Star POS device within connection range of the iOS device you want to pair with and turn the power on.
   If SSP is used by the Star POS device, press the PAIR button for 5 seconds or more to start paring.
2. Tap Settings > Bluetooth.

3. Tap Bluetooth to turn it on. Your iOS device searches and displays the Bluetooth devices in range. Tap the Star portable printer you want to pair with.



4. If a PIN code is used for Star Bluetooth device pairing, enter the PIN and tap Pair.



5. When the pairing is complete, you'll see this message.

◆How to change the Bluetooth Device Name

If you would like to change the Bluetooth device name of your Star Portable Printer, a Windows PC and Star's Mobile Bluetooth Setting Utility is necessary. Please contact with our dealer to get this utility.

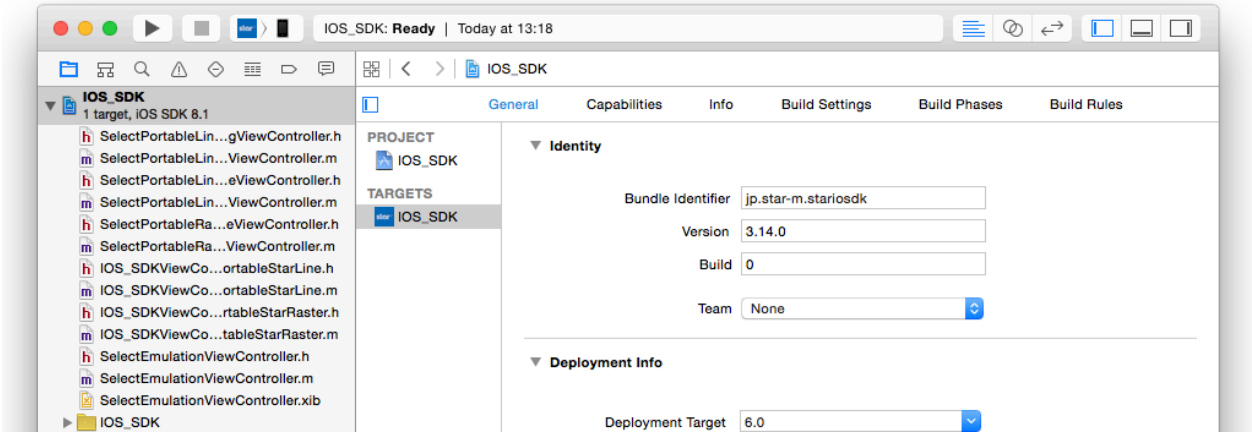
Getting Started

To build an iOS project, Xcode are needed. These tools are available in one package from the Apple Developer Site or Mac App Store. It is important to note that in order to produce applications that will actually run on an iOS device, you must be part of the Apple Developer Program, which requires a yearly subscription. While it is possible to obtain these tools from Mac App Store as stated above, your application will only be able to run in the iOS Simulator and will not install on an actual device.

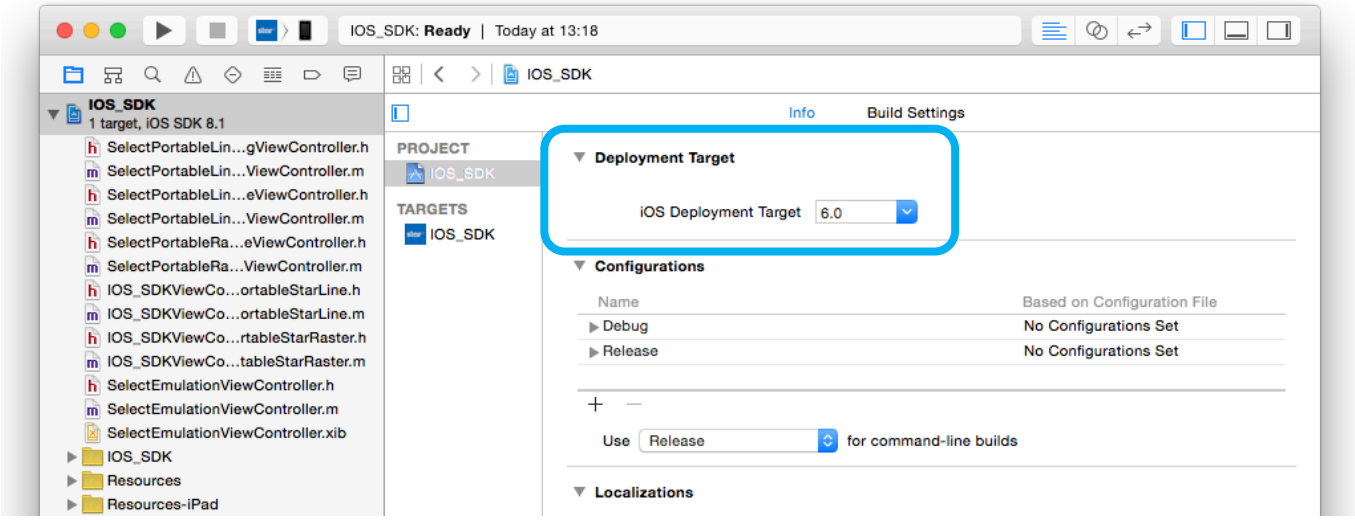
It is assumed Xcode have already been installed on your Mac at this point. Should you need assistance or additional information, visit the Resources section of the Apple Developer Site.

How to open the Star iOS SDK project in Xcode:

1. Unzip the Star iOS SDK folder and open it.

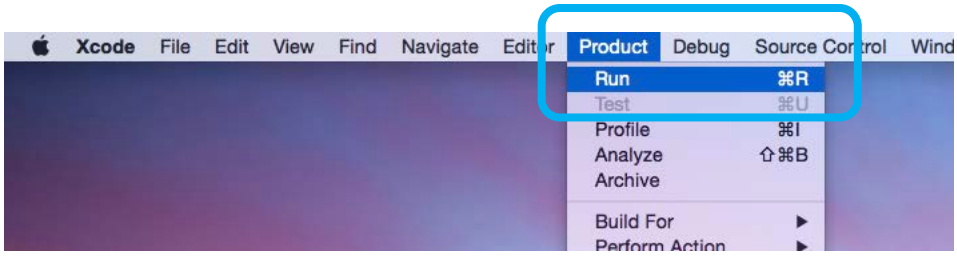


2. Open IOS_SDK.xcodeproj.



.3. Set the iOS Deployment Target to 6.0 or later.

Running the project:



1. Use the shortcut ⌘R or click Product in the top menu bar and then Run.

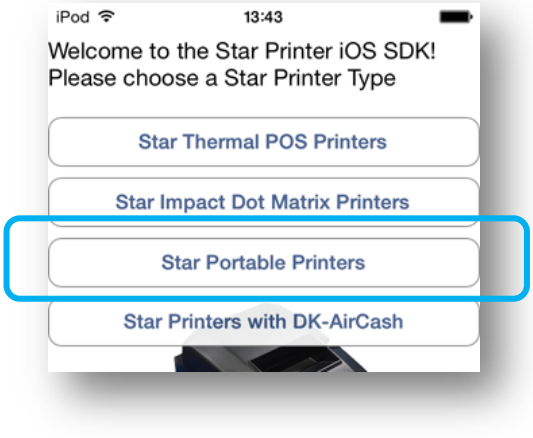
Using the SDK with Portable Star Micronics Printers

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

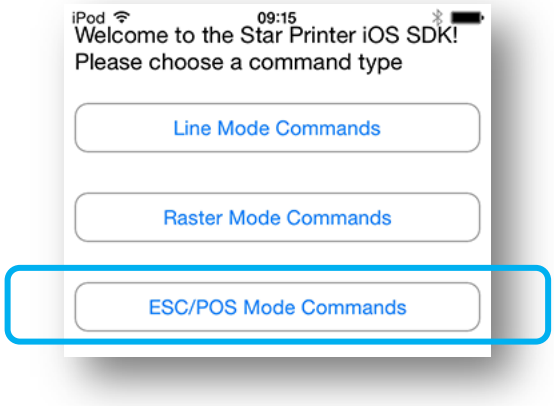
Port Name and Interface Relation:
StarIO uses specific port names to identify what port will be used. These are very important to understand because not following the naming convention correctly will fail to communicate with the printer.

Interface	Port Name	Port Settings
Wireless LAN (TCP/IP)	tcp:"IP Address"	Portable;escpos
Bluetooth	BT:	Portable;escpos

Using a Portable Printer

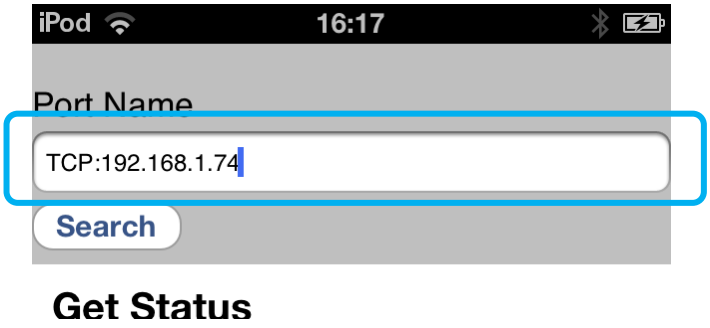


1. Tap "Star Portable Printers".



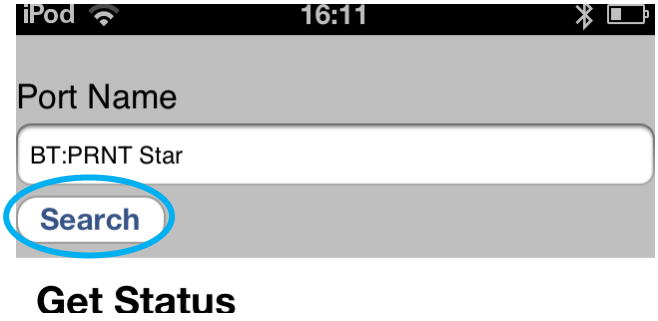
2. Tap "ESC/POS Mode Commands". The mode chosen results in which samples can be sent to the printer.

Wireless LAN Printers (SM-T300)



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 (SM-S210i, SM-S220i, SM-T300i, SM-T400i)

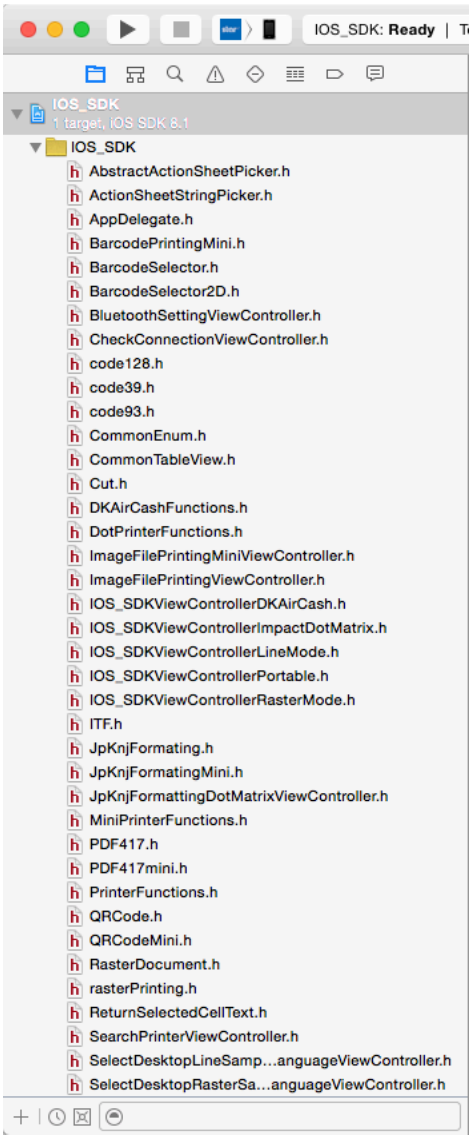


1. Tap “Search” to find all connected Star Bluetooth devices.

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

Overview of How This iOS SDK is Designed

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



All functionality is located in the IOS_SDK project and IOS_SDK target.

Run the program from the IOS_SDKViewController.m file; this source code is the starting point for both POS and Portable Printers.

See how specific functions work by clicking on the other source files. For example, “code128.m” 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

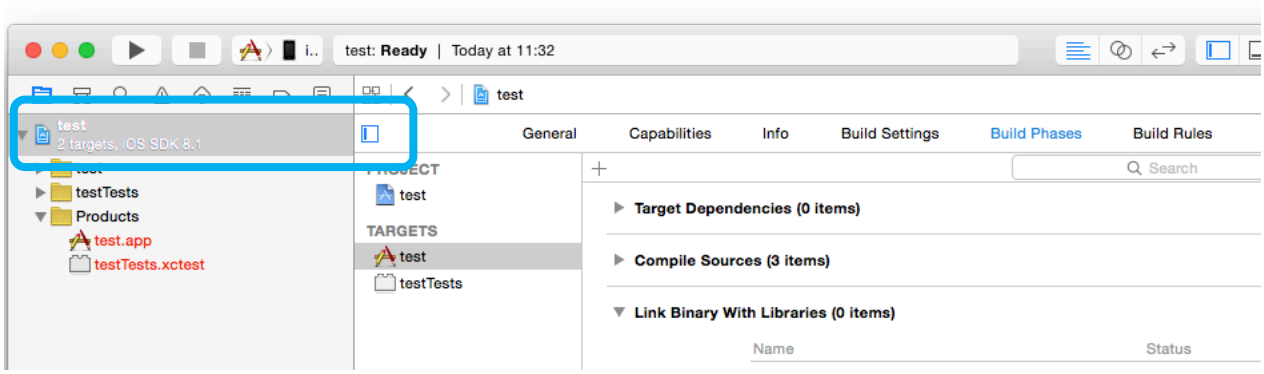
Source files containing “Mini” are sample codes for portable printer models with ESC/POS mode. StarBitmap.m applies to both printer types.

The StarIO Framework

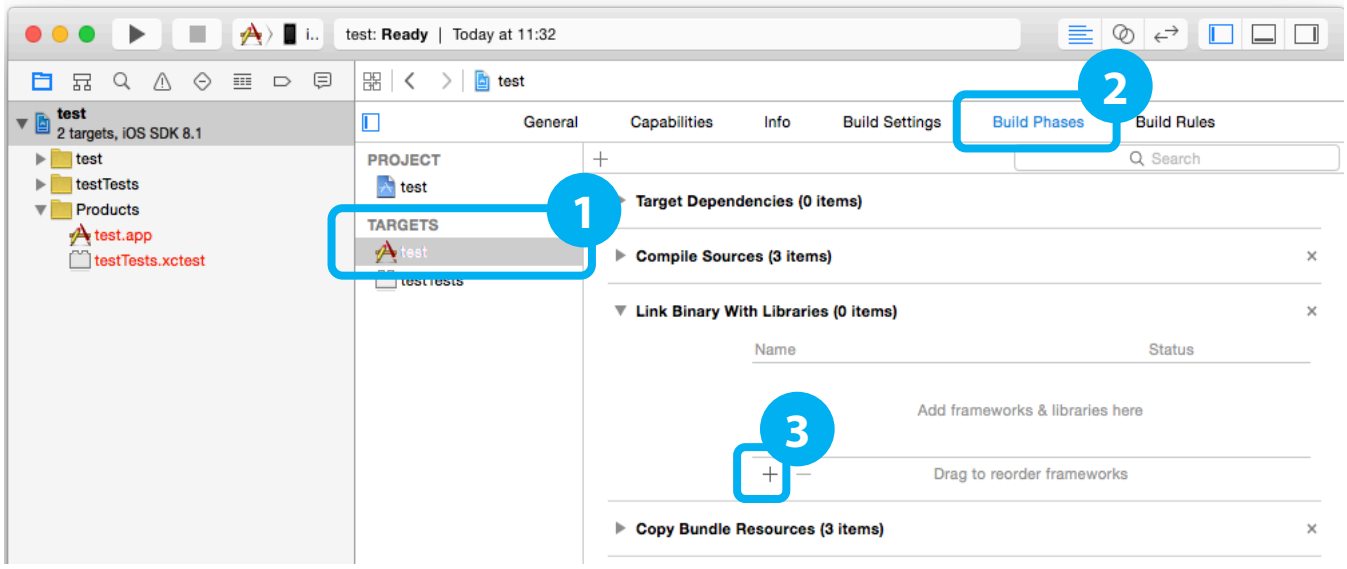
The StarIO framework is already included when the Star iOS SDK is loaded in Xcode; there's no need to include it again when testing our SDK. However, when you are building your own application, it is necessary to add the StarIO framework and the External Accessory framework into it to expose StarIO methods.

◆When building a new application

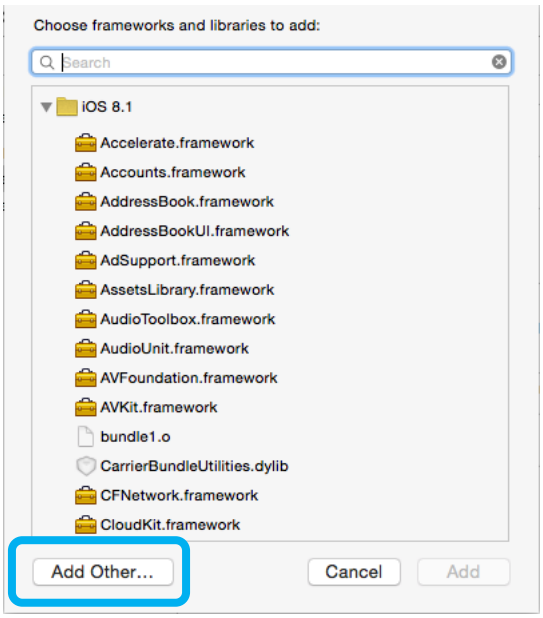
1. Add StarIO.framework into your project



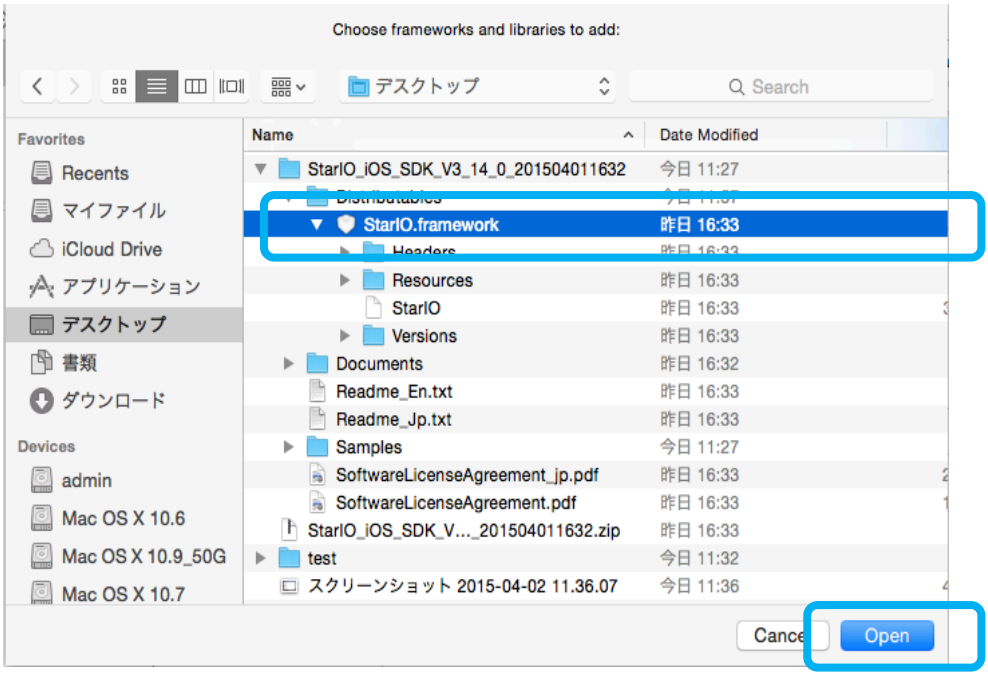
1. Click on the created project.



2. Open a target, click the Build Phases tab, click the + of Link Binary With Libraries.



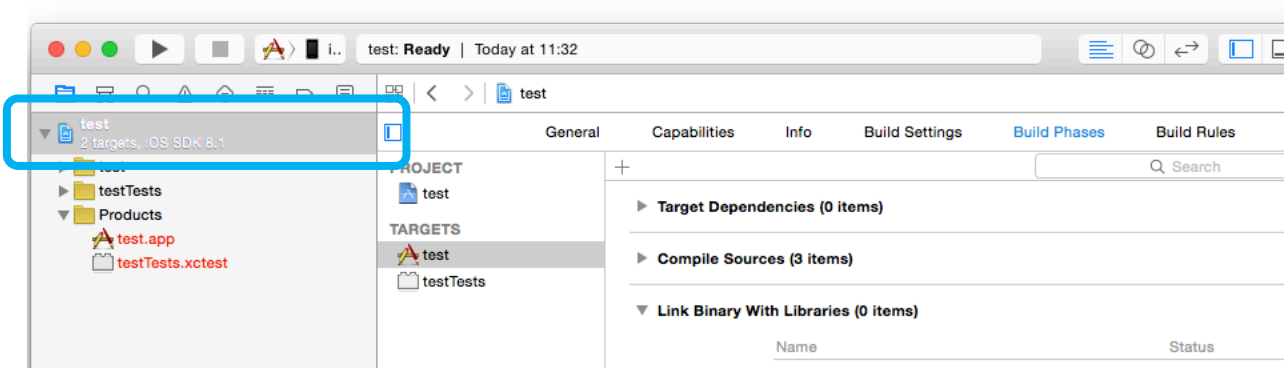
3. Click the Add Other… button.



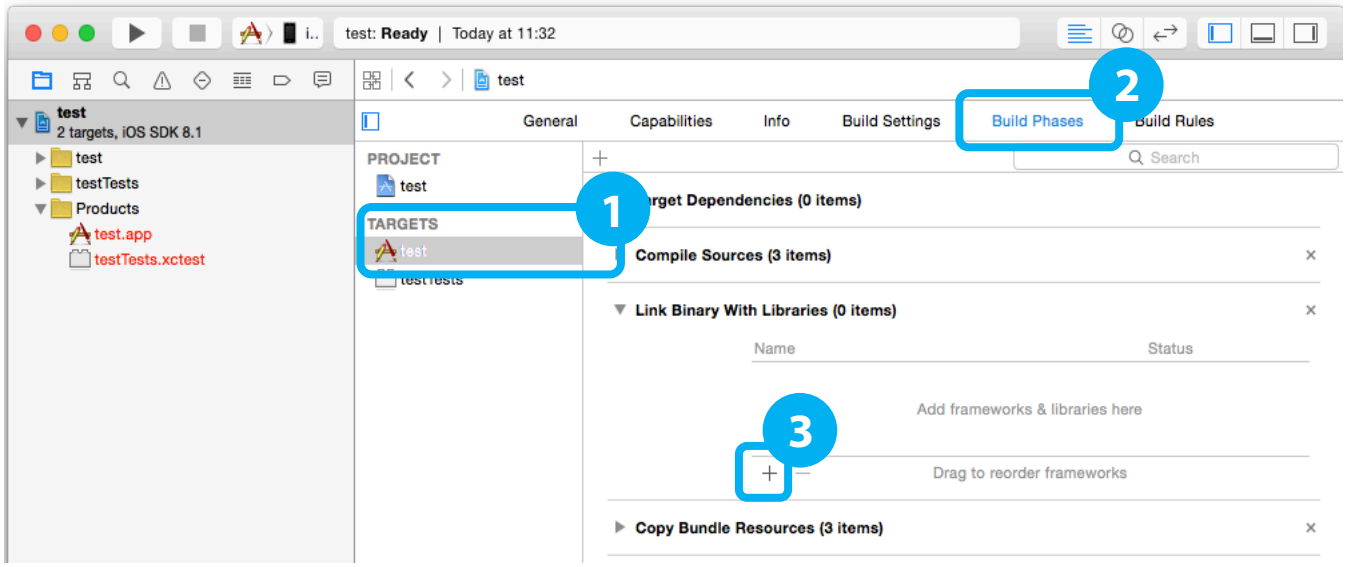
4. Browse to the location of where the Star iOS SDK was unzipped and select StarIO.framework. Then click Open.

5. The framework is added to your project and all StarIO methods are now available to you.

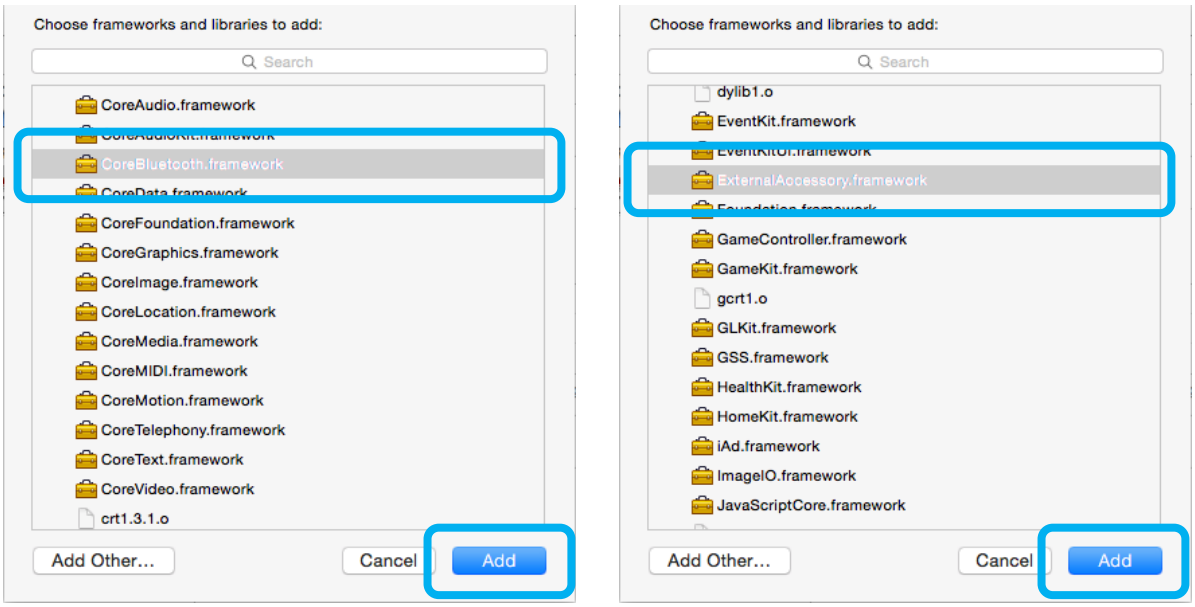
2. Add other framework into your project



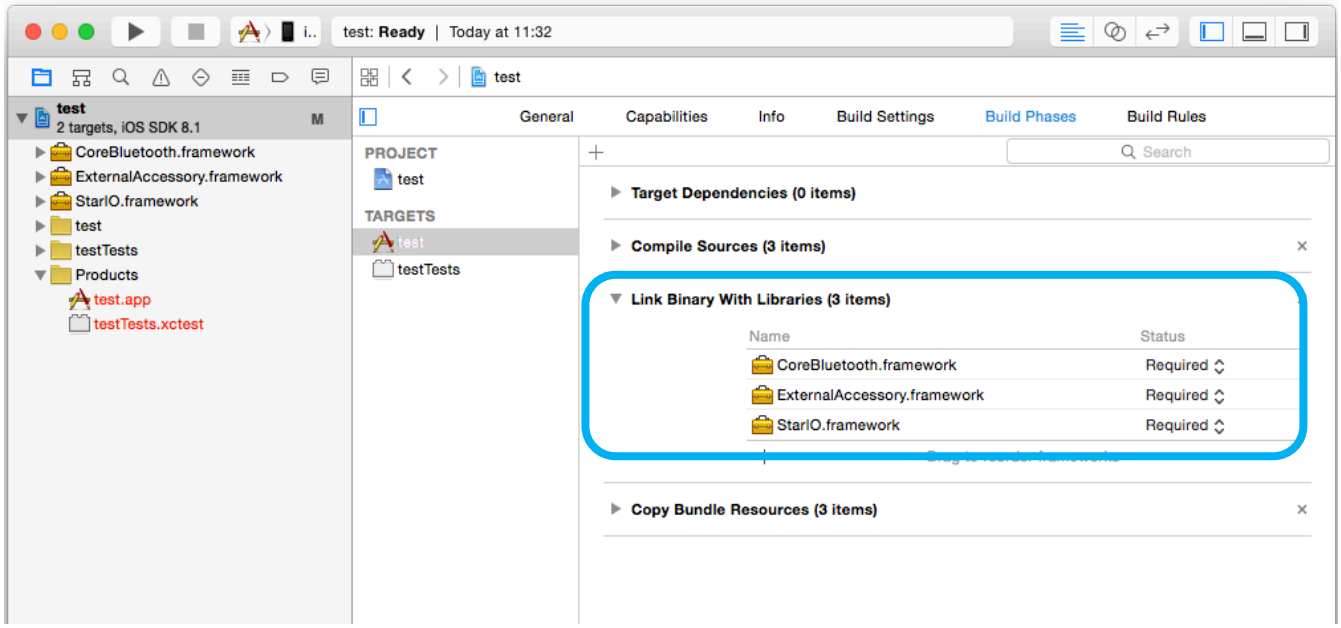
1. Click the created project.



2. Open a target, click the Build Phases tab, click the + of Link Binary With Libraries.



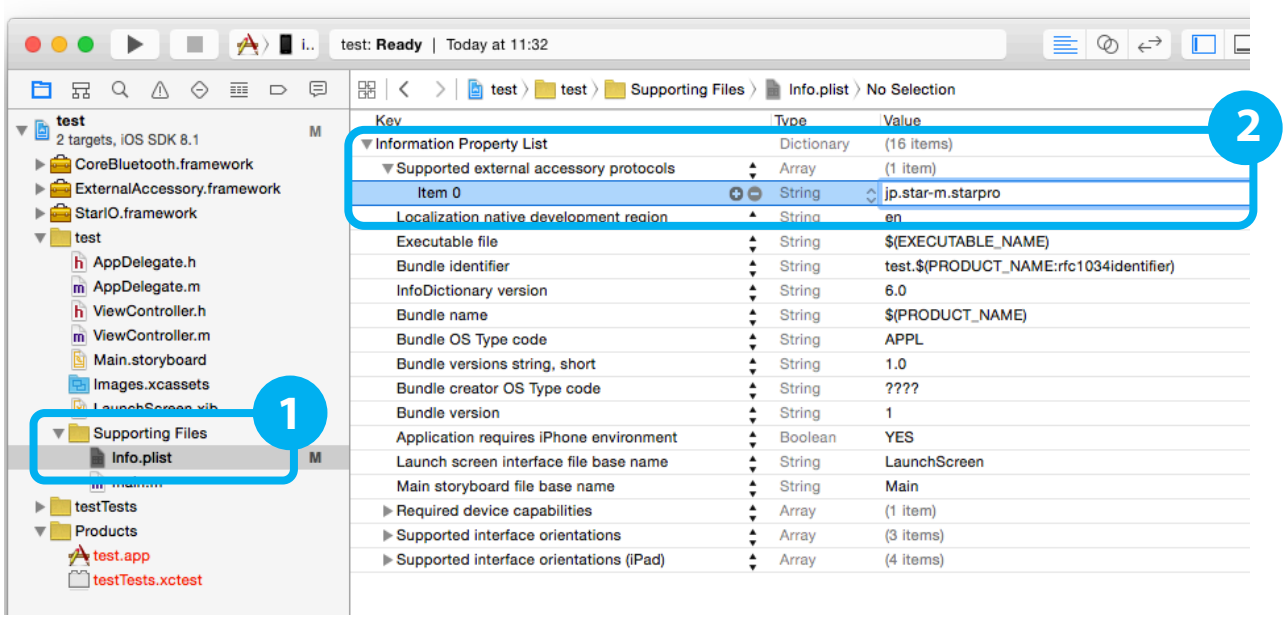
3. " External Accessory framework" and "Core Bluetooth framework" are added respectively. Select the framework, click the Add button.



4. Check if the necessary framework has been added.

3. Edit information property list (Bluetooth printer only)

Note: **Please do not apply this, if you are not using Bluetooth ineterface.**



1. Click on the information property list file (default: Info.plist file).
2. Add the Supported external accessory protocols key. Click the triangle of this key and set the value for the Item 0 to jp.star-m.starpro.
4. You have finished editing the information property list.

◆Version up of StarIO.framework

1. Delete StarIO.framework from your project.
2. Copy new StarIO.framework
3. Clean the Xcode project.
   -Open the Xcode project and select [Build]-[Clean] from the menu.
4. Build the Xcode project.

To refer to the new StarIO.framework without deleting the existing StarIO.framework, surely confirm the “framework search path” setting of the Xcode project.

If the old path of the StarIO.framework remains in front of the “framework serach path”, the previous StarIO.framework will be used.

The StarIO Methods Overview

SMPort Class:

● Property

portName	Acquires the printer port name.
portSettings	Acquires the port settings.
timeoutMillis	Acquires and specifies the timeout time for internal control and API.
endCheckedBlockTimeoutMillis	Acquires and specifies the timeout time for endCheckedBlock method.
connected	Acquires the connected condition of the iOS device with the specified Bluetooth printer.

- (NSString *)**portName**

Specifies the port of the printer.

- (NSString *)**portSettings**

Acuqires the port settings.

- (u_int32_t)**timeoutMillis**

Acquires and specifies the timeout time for internal control and API. (unit: millisecond)

@property(assign, readwrite) u_int32_t **endCheckedBlockTimeoutMillis**

It obtains and sets endCheckedBlock method timeout value [unit: ms]
If it takes long time to print, stand-by time for print completion in endCheckedBlock method can be extended by increasing this value.
Default value is the timeout value designated by getPort method.

(BOOL)**connected**

This property is supported by the Bluetooth interface only.
It always returns “Yes” when the printer is connected to your iOS device and “No” if it is not.

This property always returns “Yes” if the interface does not support it.

This property has a 5-second delay after the Bluetooth is disconnected because of limitations on the iOS device.

● Method

getPort

```
+ (SMPort *) getPort: (NSString *) portName : (NSString *) portSettings : (u_int32_t) TimeoutMillis)
```

GetPort is what you will be using to “open” the port to the printer.

Parameters:

portName – Specify the communication port to the printer.

Ex. *@”TCP:192.168.1.2” (In Ethernet)*
@”BT:PRNT Star” (In Bluetooth)
@”BT:00:11:62:1b:4d:f4”(To specify the MAC address in Bluetooth)
Note: iOS6 is required to specify the MAC address in Bluetooth. Other iOS versions cannot use this function.

portSettings – Specify the communication port to the portable printer(ESC/POS Mode).

Ex. *@”portable;escpos”*
**@”mini” is available for backwards compatibility only.*

timeoutMillis – Specify the timeout time for internal control and API.

Note: 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).

Returns:

An instance of SMPort class. It returns "nil" if it fails to generate communication port.

After executing getPort, please do not forget releasePort before executing the next getPort.
Otherwise the communication may return nil.

The following would be an actual usage of GetPort:

```
SMPort *port = nil;

     @try
     {
          port = [SMPort getPort: portName : portSettings: 10000];
     }
     @catch(PortException)
     {
          //There was an error opening the port
     }
```

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

For the Bluetooth I/F, close a port when it is not in communication with a printer for 30 seconds or more.
It is recommended to open and close a port per transaction.

searchPrinter

```
+ (NSArray *) searchPrinter;

+ (NSArray *) searchPrinter: (NSString *) target
```

searchPrinter detects printers in LAN and paired Bluetooth printers and returns search result as NSArray..

NSArray of return value includes instance of PortInfo Class.
PortInfo class of return value includes, PortName, MAC address(Ethernet model only), ModelName and you can get them by portName, macAddress, and modelName property.

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

When the Argument value of target is specified, it detects either Ethernet printers or Bluetooth printers.

Parameters:

Target – When @"TCP:" is specified, Ethernet printers will be detected.

When @"BT:" is specified, Bluetooth printers will be detected.

This API do not guarantee the discovery of devices.
iOS6 is required to specify the MAC address in Bluetooth.
Other iOS versions cannot use this function.

```
//The following would be an actual usage of searchPrinter:

NSArray *portArray = [[SMPort searchPrinter] retain];
for (int  i = 0; i < portArray.count; i++) {
   PortInfo *port = [portArray objectAtIndex:i];
   NSLog(@"Port Name: %@", port.portName);
   NSLog(@"MAC Address : %@", port.macAddress);
   NSLog(@"Model Name: %@", port.modelName);
}
 [portArray release];
```

The above example shows both of printers in LAN and Bluetooth printers being detected and search result being output to the log.

readPort

```
- (u_int32_t) readPort: (u_int8_t *) readBuffer : (u_int32_t *) offset : (u_int_32_t) size;
```

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:

`PortException` - when a communication failure occurs

releasePort

```
+ (void) releasePort: (SMPort *) port;
```

This function closes a connection to the port specified.

Parameters:
port – SMPort type representing a previously initialized port.

After executing getPort, please do not forget releasePort before executing the next getPort.
Otherwise the communication may return nil.

writePort

```
-(u_int32_t) writePort: (u_int8_t const *) writeBuffer: (u_int32_t) offset: (u_int32_t) size;
```

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 PrintTextWithPortName 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}
unsigned char command = {0x41, 0x42, 0x43, 0x44, 0x1B, 0x7A, 0x00, 0x1B, 0x64, 0x02};

uint bytesWritten = 0;

@try
{
   while (bytesWritten < (sizeof command))
   {
     bytesWritten += [port writePort: command : bytesWritten : sizeof command - bytesWritten];
   }
}
@catch (PortException)
{
  //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.

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:

`PortException` - when a communication failure occurs

getParsedStatus

```
-(void) getParsedStatus: (void *) starPrinterStatus: (u_int32_t) level;
```

This method retrieves detailed status from the printer with StarIO.

Returns:
StarPrinterStatus structure giving the current device status

Throws:

`PortException` - 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: &printerStatus : 2];

if (printerStatus.offline == true)
{
     if (printerStatus.coverOpen == true) {
          //There was a cover open error
     }
     else if (printerStatus.receiptPaperEmpty == true) {
          //There was a receipt paper empty error
     }
     else {
          //There was a offline error
     }

}
else {
     //If False, then the printer is online.
}
```

Status List of the class structure StarPrinterStatus

mamber name	contents	Type	Detail (*1)
blackMarkError	Black Mark Error	SM_BOOLEAN	Portable printers are not supported.
compulsionSwitch	Compulsion SW	SM_BOOLEAN	Portable printers are not supported.
coverOpen	Cover Status	SM_BOOLEAN	You can check status of Cover SM_TRUE : Cover is opened. SM_FALSE : Cover is closed.
cutterError	Auto-cutter Error	SM_BOOLEAN	Portable printers are not supported.
etbAvailable	ETB available or not	SM_BOOLEAN	Portable printers are not supported.
etbCounter	ETB Counter	UCHAR	Portable printers are not supported.
headThermistorError	Head Thermistor Error	SM_BOOLEAN	Portable printers are not supported.
offline	ONLINE/OFFLINE Status	SM_BOOLEAN	You can check status of Online or offline. SM_TRUE : Printer is Offline. SM_FALSE : Printer is Online
overTemp	Stopped by high head temperature	SM_BOOLEAN	Portable printers are not supported.
presenterPaperJamError	Presenter Paper Jam Error	SM_BOOLEAN	Portable printers are not supported.
presenterState	Presenter Paper Position	UCHAR	Portable printers are not supported.
raw	Byte column of status	UCHAR[63]	Byte column of status (example : HEX 23 86 00 00 00 00 00 00 00)
rawLength	raw length	CHAR	raw length
receiptPaperEmpty	Paper end	SM_BOOLEAN	SM_TRUE : Paper end. SM_FALSE : Paper exist.
receiptPaperNearEmptyInner	Paper Near-end (Inner Side)	SM_BOOLEAN	Portable printers are not supported.
receiveBufferOverflow	Receive Buffer Overflow	SM_BOOLEAN	Portable printers are not supported.
unrecoverableError	Non-recoverable Error	SM_BOOLEAN	Portable printers are not supported.
voltageError	Electric Voltage Error	SM_BOOLEAN	Portable printers are not supported.

Class Structure StarPrinterStatus Supported by Portable Printer

mamber name	SM-T300	SM-S210i (Jp model only)	SM-S220i (Excluding Jp model)	SM-S230i (Excluding Jp model)	SM-T300i	SM-T400i
blackMarkError						
compulsionSwitch						
coverOpen	✓	✓	✓	✓	✓	✓
cutterError						
etbAvailable						
etbCounter						
headThermistorError						
offline	✓	✓	✓	✓	✓	✓
overTemp						
presenterPaperJamError						
presenterState						
raw	✓	✓	✓	✓	✓	✓
rawLength	✓	✓	✓	✓	✓	✓
receiptPaperEmpty	✓	✓	✓	✓	✓	✓
receiptPaperNearEmptyInner						
receiveBufferOverflow						
unrecoverableError						
voltageError						

beginCheckedBlock

```
-(void) beginCheckdBlock: (void *) starPrinterStatus: (u_int32_t) level;
```

This method is used in combination with endCheckedBlock and checks the completion of printing. beginCheckedBlock must be run just before sending print data.

Parameters:

starPrinterStatus – a pointer to StarPrinterStatus structure

(Possible to specify StarPrinterStatus, StarPrinterStatus_1 of StarPrinterStatus_2. Normally StarPrinterStatus_2 is specified.)

level – the level of StarPrinterStatus structure

(Possible to specify a value of 0,1 or 2. Normally 2 is specified.)

See the sample code here.

endCheckedBlock

```
-(void) endCheckdBlock: (void *) starPrinterStatus: (u_int32_t) level;
```

This method is used together with the beginCheckedBlock method in a set.

It monitors printer status and when the transferred data is printed completely, returns control. In case of being transferred other kind of data than print data, when its command is processed in the printer, it returns the control.

In case that printing is not completed before the timeout (*1) or printer error occurs during printing, it returns PortException.

(*1) To timeout value, endCheckedBlockTimeoutMillis property is applied. Default value is the timeout value designated by getPort. Please adjust the endCheckedBlockTimeoutMillis value to be longer than printing time.
Timeout length is specified by getPort, endCheckedBlockTimeoutMillis or is 10 seconds if specified less than 10 seconds.

Parameters:

starPrinterStatus – a pointer to StarPrinterStatus structure

(Possible to specify StarPrinterStatus, StarPrinterStatus_1 of StarPrinterStatus_2. Normally StarPrinterStatus_2 is specified.)

level – the level of StarPrinterStatus structure

(Possible to specify a value of 0,1 or 2. Normally 2 is specified.)

Returns:

StarPrinterStatus structure giving the current device status

Throws:

`PortException` - when a communication failure* occurs

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

When using 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.

```
unsigned char command = {0x41, 0x42, 0x43, 0x44, 0x1B, 0x7A, 0x00, 0x1B, 0x64, 0x02};
uint bytesWritten = 0;

StarPrinterStatus_2 starPrinterStatus;

SMPort *port = nil;

@try
{
    port = [SMPort getPort:@"BT:" :@"MINI" :10000 ];

    //Start checking the completion of printing
    [port beginCheckedBlock:&starPrinterStatus :2];

    if (starPrinterStatus.offline == SM_TRUE)
    {
        //There was an error writing to the port
    }
    while (bytesWritten < sizeof (command))     {
        bytesWritten += [port writePort: command : bytesWritten : sizeof command - bytesWritten];
    }

    //End checking the completion of printing
    [port endCheckedBlock:&starPrinterStatus :2];

    if (starPrinterStatus.offline == SM_TRUE)
    {
        //There was an error writing to the port
    }
}
@catch (PortException)
{
    //There was an error writing to the port
}
@finally
{
    [SMPort releasePort:port];
}
```

compressRasterData / generateBitImageCommand

```
+ (NSMutableData *) compressRasterData: (int32_t) width :(int32_t) height (u_int8_t *) imageData : (NSString *)portSettings

+ (NSMutableData *) generateBitImageCommand: (int32_t) width :(int32_t) height (u_int8_t *) imageData : (NSString *)portSettings [Deprecated]
```

This method compressed the raster data. It will improve throughput.

Parameters:

width	– Width of image data (pixel)
height	– Height of image data (pixel)
imageData	– Raster data before compression
portSettings	– Printer options (@"portable;escpos")
	*@"mini" is available for backwards compatibility only.

Returns:

Success: Compressed Raster data

Failure: nil

getFirmwareInformation

```
-(NSDictionary *) getFirmwareInformation:
```

This method gets a firmware Information of the printer.

Returns:

It returns NSDictionary as an acquisition result.
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.

StarIOVersion

```
+(NSString *) StarIOVersion
```

This method gets the StarIO version.

Returns:
StarIO version

SMBluetoothManager Class:

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

SMBluetoothManager Class is supported by SM-S210i, SM-T300i and SM-T400i with F/W version 3.0 or later.

● Property

portName	Acquires the portName of the device to be connected.
deviceType	Acquires the type of the device to be connected.
opened	Shows whether the port is opened.
deviceName	Acquires and specifies the current Bluetooth device name.
iOSPortName	Acquires and specifies the port name to be used with the StarIO.
autoConnect	Acquires and specifies the setting (Valid or Invalid) of the autoconnection function.
security	Acquires the Bluetooth security setting (SPP or PIN Mode)
pinCode	Specifies the PIN Code to be used for pairing.

```
@property(nonatomic, readonly) NSString *portName
```

Creates an instance of SMBluetoothManager.

```
@property(nonatomic, readonly) SMDeviceType deviceType
```

Acquires the type of the device to be connected.

```
@property(nonatomic, readonly) BOOL opened
```

Shows whether the port is opened.
It returns YES if the open method was successful.
Then it will return NO when the close method is called.

@property(nonatomic, retain) NSString *__deviceName__

Acquires and specifies the current Bluetooth device name.
The current setting is read when the loadSetting method is called.
To set it, run the apply method after changing this property.

Valid number of characters: 1 to 16
Valid characters: 0-9, a-z, A-Z
; : ! ? # $ % & , . @ _ - = Space / * + ~ ^ [{ ( ] }) | \

@property(nonatomic, retain) NSString *__iOSPortName__

Acquires and specifies the iOS port name to be used with the StarIO.
The current setting is read when the loadSetting method is called.
To set it, run the apply method after changing this property.

Valid number of characters: 1 to 16
Valid characters: 0-9, a-z, A-Z
; : ! ? # $ % & , . @ _ - = Space / * + ~ ^ [{ ( ] }) | \

@property(nonatomic, assign) BOOL __autoConnect__

Acquires and specifies the setting of the auto connection function.
The current setting is read when the loadSetting method is called.
To set it, run the apply method after changing this property.

Set to NO when the security setting is set to PIN code mode.

@property(nonatomic, assign) SMBluetoothSecurity __security__

Acquires and specifies the Bluetooth security setting(SSP or PIN code mode).
The current setting is read when the open method is called.
To set it, run the apply method after changing this property.

Set the autoConnect property to NO when specifies the PIN code mode.

```
@property(nonatomic, retain) NSString  *pinCode
```

Specifies the PIN code of the Bluetooth interface.
It can not acquire the current setting.
Set to nil when the PIN code is not changed.

Valid number of characters: 4 to 16

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

● **Method**

initWithPortName : deviceType

```
-(id) initWithPortName: (NSString *) portName deviceType: (SMDeviceType) deviceType
```

This method is used to create an instance of SMBluetoothManager.

Parameters:

portName – the port name of the device to be connected
Ex. “BT:Star Micronic”

deviceType – the type of the device to be connected
SMDeviceTypePortablePrinter

Returns:

It returns Instance of SMBluetoothManager when succeeded.
It returns nil when failed.

open

```
-(BOOL) open
```

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

Returns:

It returns YES when succeeded and NO when failed.

loadSetting

```
-(BOOL) loadSetting
```

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

Returns:

It returns YES when succeeded and NO when failed.

apply

```
-(BOOL) apply
```

This method is used to apply the property values of deviceName, iOSPortName, autoConnect, security and pinCode.

Returns:
It returns YES when succeeded and NO when failed.

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

close

```
-(void) close
```

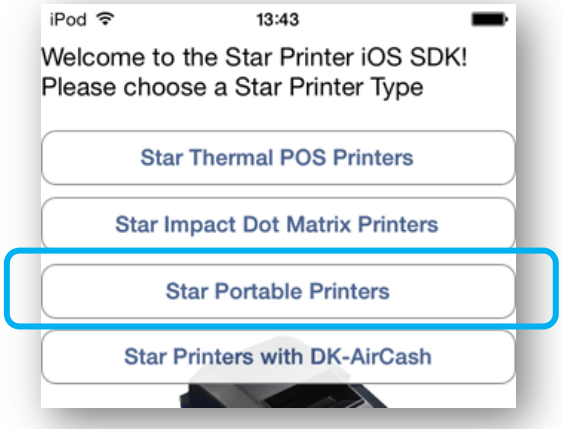
This method is used to close communication with the printer.

StarIO iOS SDK Functionality

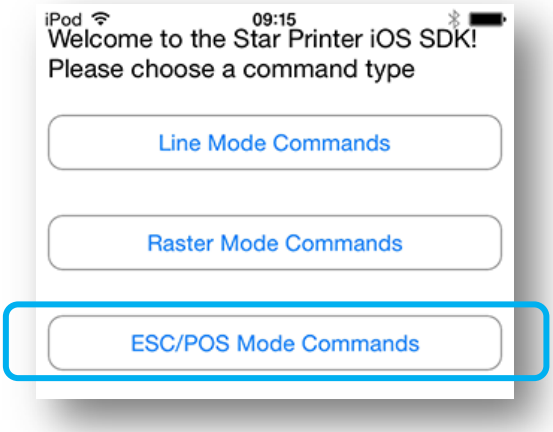
Overview of this SDK functionality and StarIO Printer Commands
All of these commands can be found in the Star Portable Printer Command Manual.

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

Choosing Portable Printer Commands



1. Tap “Star Portable Printers”.

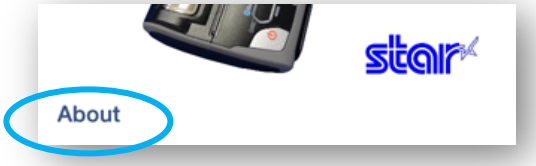


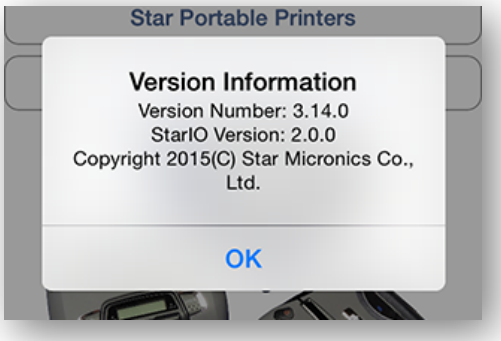
2. Tap “ESC/POS Mode Commands”.
   The mode chosen results in which samples can be sent to the printer.

Supported Samples

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

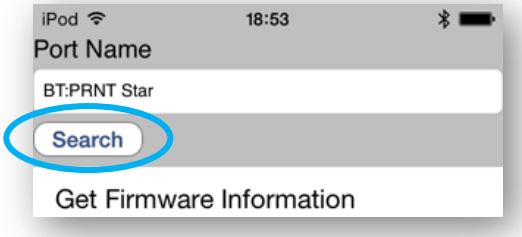
Get StarIO Version





Tap “About”, displays StarIO version.

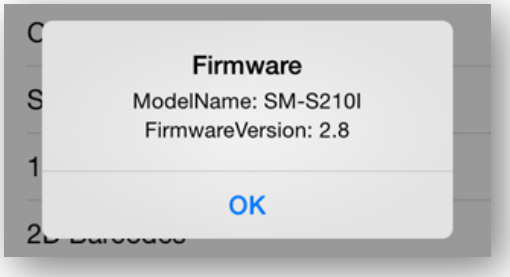
Port Discovery



Automatically detects which Star Micronics Bluetooth Printers are connected to the network. Tap the printer to connect to it. This feature is documented in greater detail here.

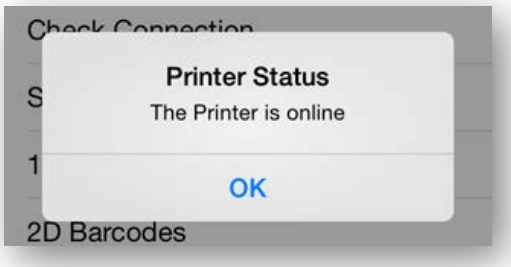
USB printers and LAN printer do not support this feature.

Get Firmware Information



Displays firmware information of the printer specified by Port Name.

Get Status



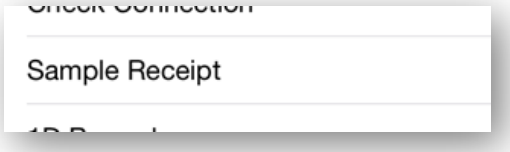
StarPrinterStatus

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

Help

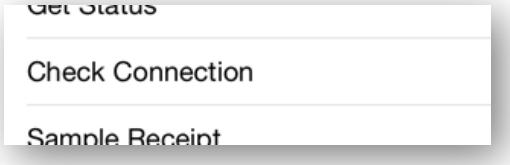
Help produces information on WLAN Parameters and StarIO Port Settings.

Sample Receipts

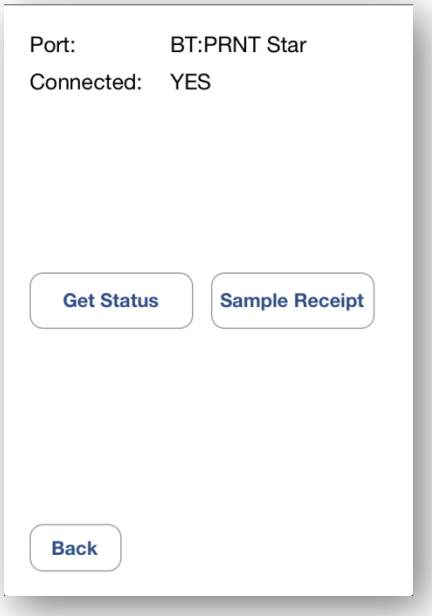


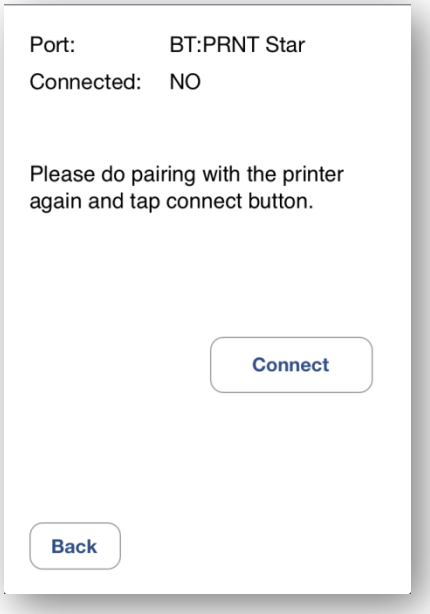
Prints a premade sample receipt in the chosen command type. “Sample Receipt” outputs a receipt in English, while “JP Sample Receipt” outputs one in Japanese.

Check Connection



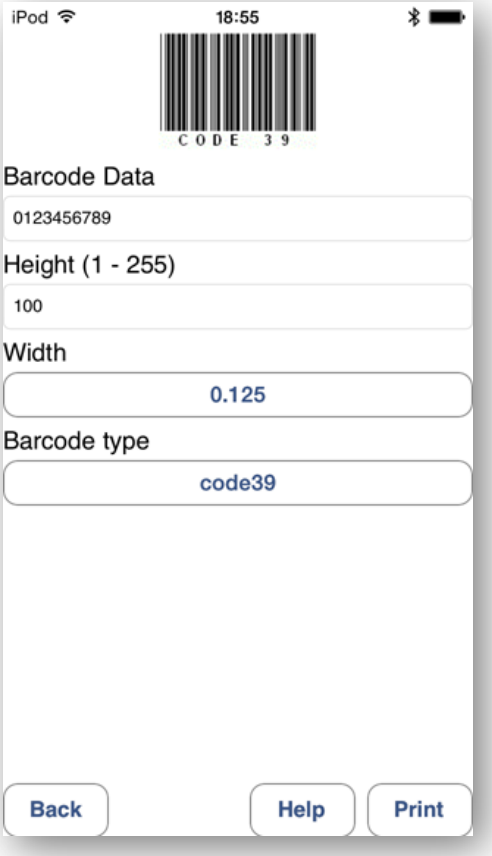
Checks Bluetooth connection using the “connected” property.





It gives “Yes” while connected.
The property changes to “No” when Bluetooth is disconnected. To connect again, do pairing with the printer following the Bluetooth connecting procedure.

1D Barcodes



Set Barcode Height

GS h n 0 ≤ n ≤ 255

Set Barcode Width

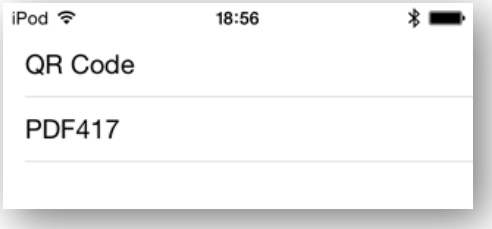
GS w n 1 ≤ n ≤ 8

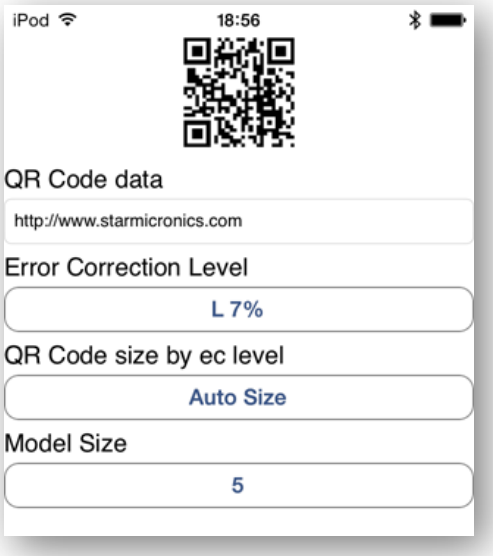
Print Barcode

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

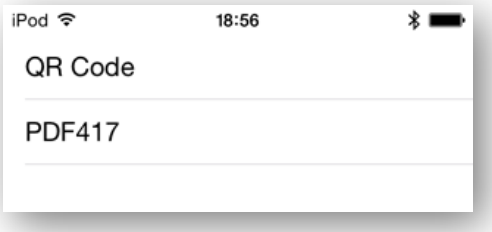
a = Correction Level L = 7%, M = 15%, Q = 25%, H = 30%

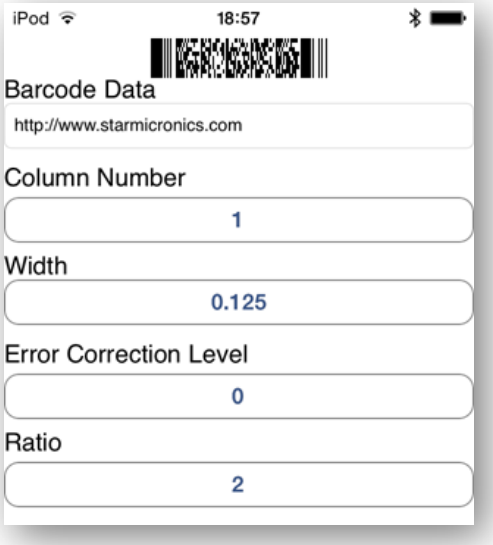
k = Model Size k = 1~8

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

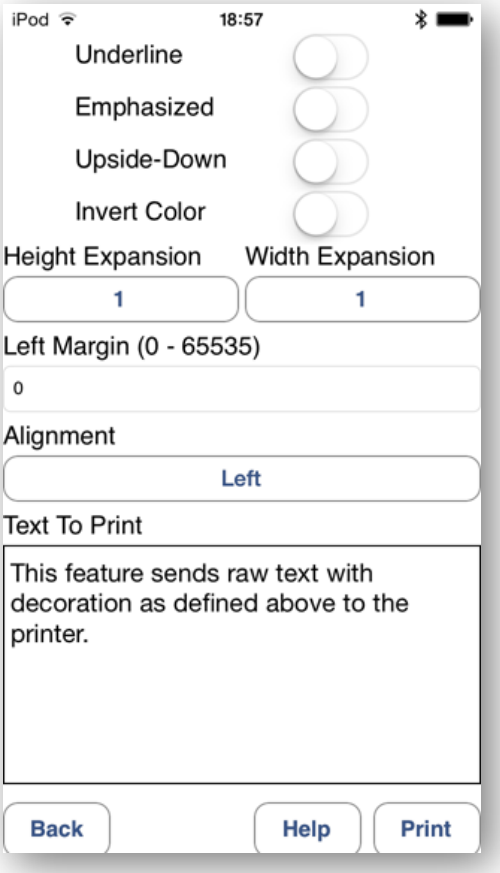
a = Correction Level 0 ≤ n ≤ 8

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 exactly the same as that of Text Formatting, except Japanese Kanji is supported. As shown above, it is easy to switch between Shift-JIS and JIS.

Japanese Kanji Text Formatting (continued from above)

Shift-JIS Kanji

FS C n	1 = On	0 = Off [Default]

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

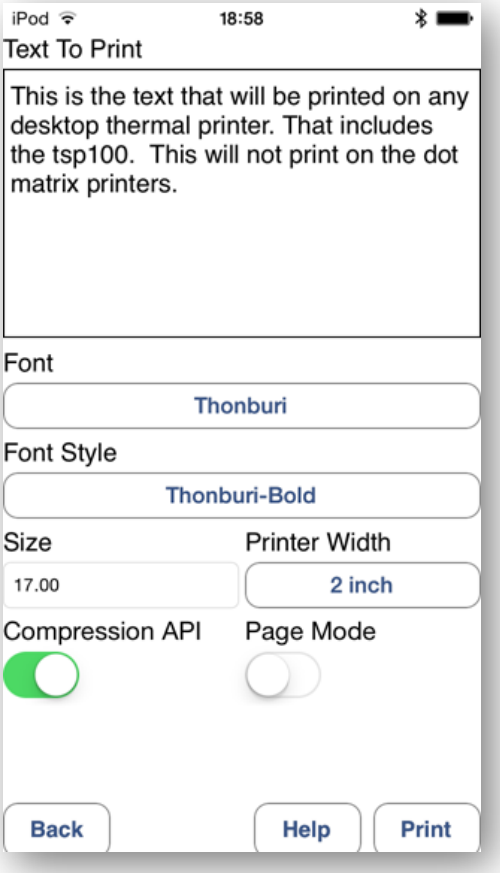
Left Margin

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



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.m"**.

[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

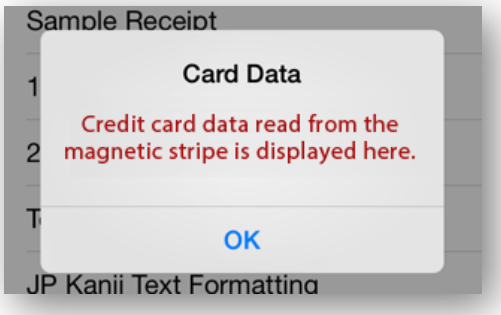
Use the dropdown box to select one of four different sample images to print via Raster Graphics. Note: The images in this sample are pre-formatted for 80mm wide receipts. If the printer in use is smaller or wider than 80mm, the image will not be automatically scaled.

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

Using ”Compression API” method may improve through put.

Note: This sample is not available in Line Mode.

MSR (Magnetic Stripe Reading)



Read One Track

ESC M C Read Track 1 or Read Track 2

Read One Track

ESC M D Read Track 2 or Read Track 3

Read Two Tracks

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

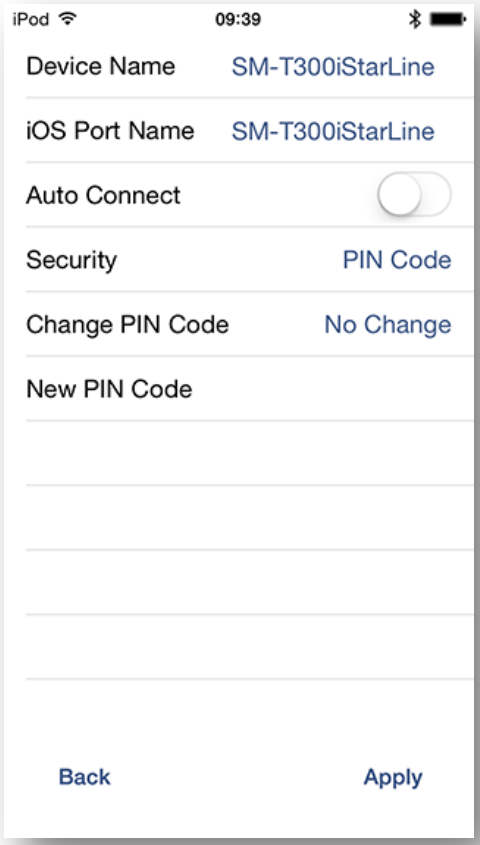
Set JIS-II Mode

ESC M J Read Japanese Characters

Cancel MSR Mode

EOT Return to standard print mode

Bluetooth Setting



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 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 Android 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, look up information, etc.

The Developers Network gets you access to:

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

Apple Developer Site

The official Apple development resource.

Apple Developer Site Resources

Peruse Apple's library of documentation for developers.

Unicode.org

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

1D Barcodes

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

2D Barcodes

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

Code Pages

Learn about Code Pages here.

ASCII Table Resource

ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol	ASCII	Hex	Symbol
0	0	NUL	16	10	DLE	32	20	(space)	48	30	0
1	1	SOH	17	11	DC1	33	21	!	49	31	1
2	2	STX	18	12	DC2	34	22	"	50	32	2
3	3	ETX	19	13	DC3	35	23	#	51	33	3
4	4	EOT	20	14	DC4	36	24	$	52	34	4
5	5	ENQ	21	15	NAK	37	25	%	53	35	5
6	6	ACK	22	16	SYN	38	26	&	54	36	6
7	7	BEL	23	17	ETB	39	27	'	55	37	7
8	8	BS	24	18	CAN	40	28	(	56	38	8
9	9	TAB	25	19	EM	41	29	)	57	39	9
10	A	LF	26	1A	SUB	42	2A	*	58	3A	:
11	B	VT	27	1B	ESC	43	2B	+	59	3B	;
12	C	FF	28	1C	FS	44	2C	,	60	3C	<
13	D	CR	29	1D	GS	45	2D	-	61	3D	=
14	E	SO	30	1E	RS	46	2E	.	62	3E	>
15	F	SI	31	1F	US	47	2F	/	63	3F	?
ASCII	**Hex**	**Symbol**	**ASCII**	**Hex**	**Symbol**	**ASCII**	**Hex**	**Symbol**	**ASCII**	**Hex**	**Symbol**
64	40	@	80	50	P	96	60	`	112	70	p
65	41	A	81	51	Q	97	61	a	113	71	q
66	42	B	82	52	R	98	62	b	114	72	r
67	43	C	83	53	S	99	63	c	115	73	s
68	44	D	84	54	T	100	64	d	116	74	t
69	45	E	85	55	U	101	65	e	117	75	u
70	46	F	86	56	V	102	66	f	118	76	v
71	47	G	87	57	W	103	67	g	119	77	w
72	48	H	88	58	X	104	68	h	120	78	x
73	49	I	89	59	Y	105	69	i	121	79	y
74	4A	J	90	5A	Z	106	6A	j	122	7A	z
75	4B	K	91	5B	[	107	6B	k	123	7B	{
76	4C	L	92	5C	\	108	6C	l	124	7C	|
77	4D	M	93	5D	]	109	6D	m	125	7D	}
78	4E	N	94	5E	^	110	6E	n	126	7E	~
79	4F	O	95	5F	_	111	6F	o	127	7F	

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

SDK Package Version History

Release Date	SDK Package Version	Update
Apr. 2016	3.16.0	- Add support devices
Apr. 2015	3.14.0	- Added SMBluetooth Manager Class - iOS 5.x End of support
Mar. 18.2015	3.13.1	- Deleted a 32-bit separate build settings.
Sep. 8 2014	3.12.0	- Added StarIO Version method - Added Image File Printing function
Jul. 24 2014	3.10.3	- Fixed a problem with multiple execution of getPort method in Bluetooth connection.
Mar. 10 2014	3.10.0	- Added getFirmwareInformation method
Nov.26 2013	3.9.0	-Added endCheckedBlockTimeoutMillis Property -iPad Air, iPad mini(2nd Generation), iPhone5S, iPhone 5C support -iOS 4.3 End of support -Added 64-bit build
Sep. 19 2013	3.8.0	- Added iOS 7 support
Jul. 3 2013	3.7.1	- Added Star Printer Status List
Jan. 29 2013	3.4.0	-Added SM-T300i/SM-T400i
Jan. 21 2013	3.3.0	-Error correction
Oct. 2 2012	3.2.0	- Added Bluetooth support - Added Check Connection and Begin/End Checked Block support
Oct. 2 2012	3.1.0	- Added iOS 6 support - Added iPhone 5 support
Aug. 31 2012	3.0.0	- Changed SDK UI to tree structure - Added Port Discovery support (POS Printers Only) - Added Apple AirPort Express support - Added Sample Receipt Printing support

Release Date	SDK Package Version	Update
May. 30 2012	2.3.0	- Fixed hang up in application when Portable Printer Sleep Mode is enabled (Portable Printers Only)
Apr. 27 2012	2.2.0	- Updated default StarIO Port Class name - Added support for Retina Display - Added ARC support
Jan. 12 2012	2.1.0	- Added Japanese Text Formatting Sample - Added Japanese Manuals
Dec. 9 2011	2.0.0	- Updated Sample Printing
Jun. 3 2011	1.2.1	- 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/