iOS Software Development Kit

How to Use StarIO for ProxiPRNT

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

Tools Needed:

- Xcode 7.0 or later
- StarIO iOS Printer

Supported Device:

- BLED10-U(dedicated hardware: Bluetooth Low Energy USB Dongle)
- iOS 7.0 or later
- iPad 3rd Generation or later
- iPhone4s or later
- iPod touch 5th Generation or later

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.

StarIO SDK (ProxiPRNT) Compatibility OS : iOS 7.0 or later

StarIO SDK (ProxiPRNT) 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 Thermal POS 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:

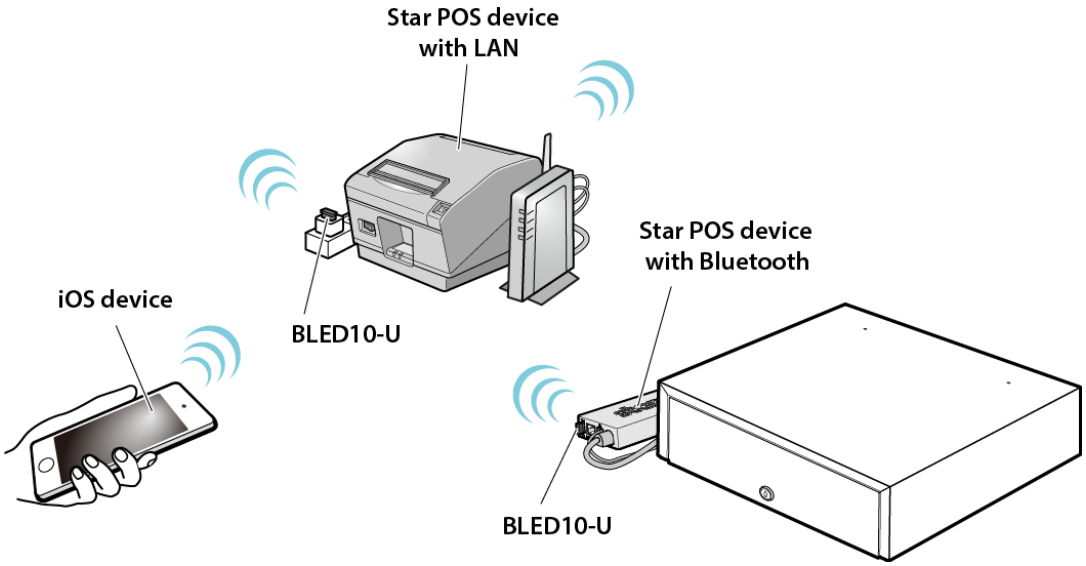
Warning		Explains potential issues
Avoid Doing This		Explains things not to do
Note		Provides important information and tips

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.

Connecting a Star Device to an iOS Device

First, connect the Star POS device (a Star POS printer or the DK-AirCash) to the iOS device.



Ethernet Interface

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

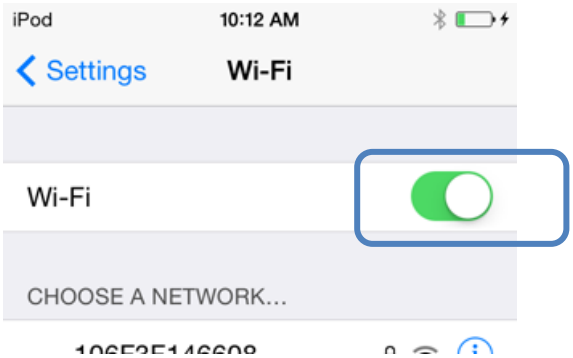
Use Star POS Devices with the #9100 Multi Session disabled.

The setting can be confirmed by Test Print which can be executed by holding down the printer's feed button while turning the printer on.

Refer to "guidelines-ethernet_en.pdf" for how to confirm and change the #9100 Multi Session setting and how to set the Static IP Address.

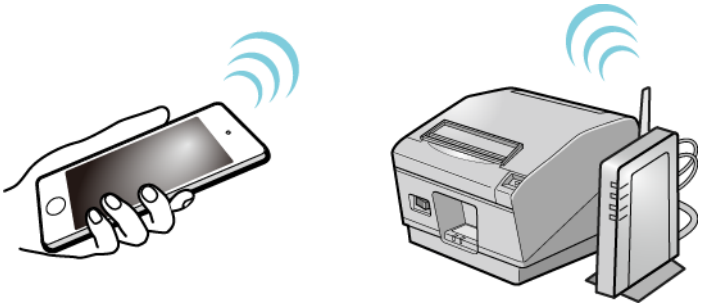
1. Assign an IP Address to the Star POS Devices and connect it to the network.

2. Tap Settings.



3. Ensure Wi-Fi is ON.

4. Connect to the same network the Star POS Device is on.



Bluetooth Interface

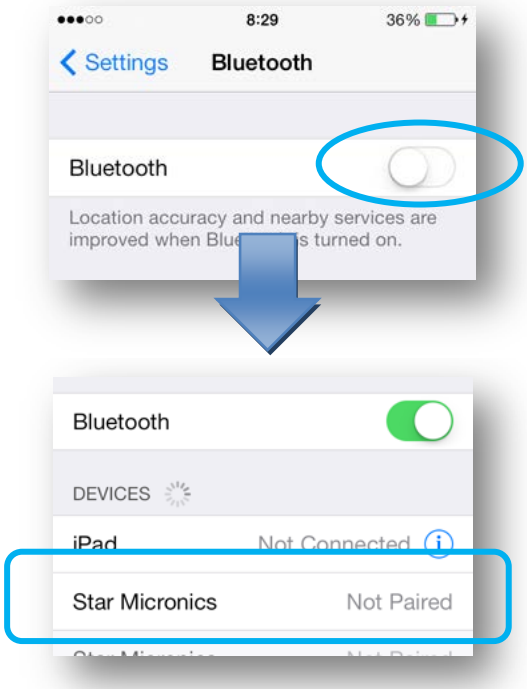
All Star Bluetooth devices have each initial device name as a factory default setting, such as "Star Micronics" and "DK-AirCash". When using multiple devices which have same device name, it is useful to change the device name for identifying each device easily.

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

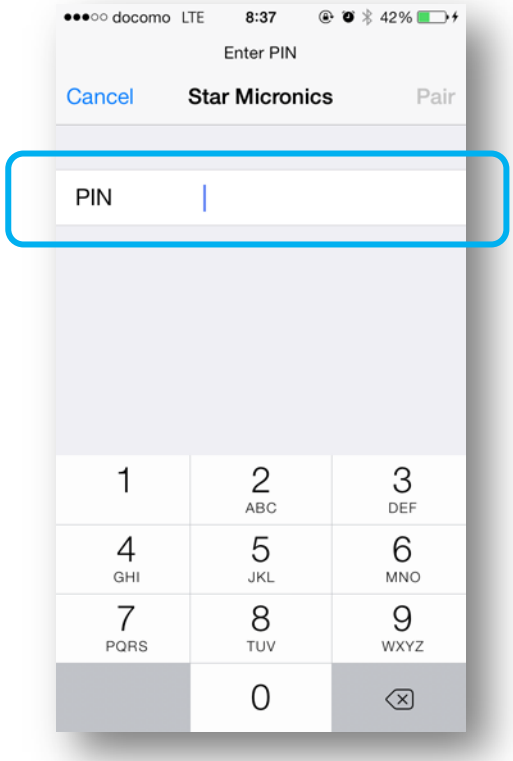
◆Pairing

1. Place your Star portable printer within connection range of the iOS device you want to pair with and turn the power on.
2. Tap Settings > General > Bluetooth.

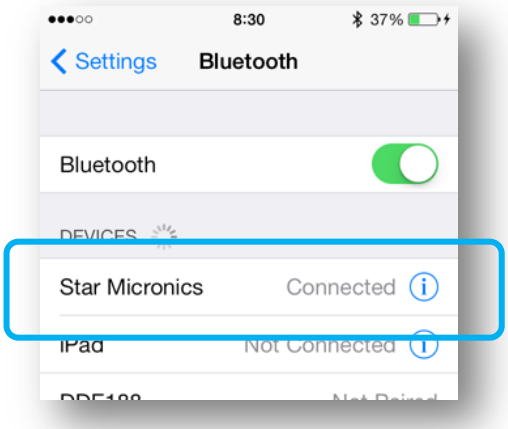
3. Tap Bluetooth to turn it on. Your iOS device searches and displays the Bluetooth devices in range. Tap the Star portable printer you want to pair with.



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**

The Star Bluetooth Utility can be downloaded from Apple App Store to change the iOS Port Name.

To confirm iOS Port Name, select [Settings]-[General]-[About] after Bluetooth pairing is established. The iOS Port Name will be shown under the Bluetooth address.

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.

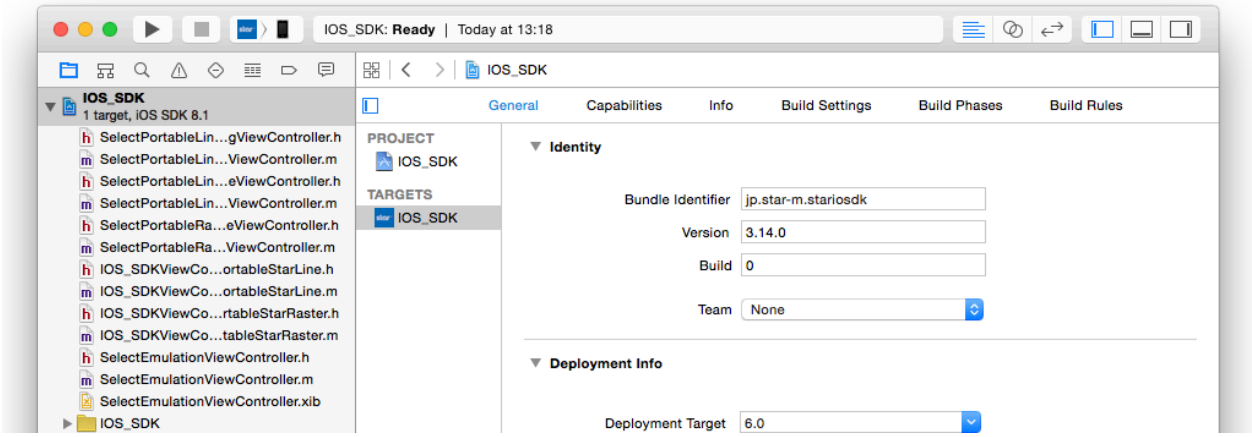
It is assumed Xcode have already been installed on your Mac at this point. Should you need assistance or additional information, visit the Resources section of the Apple Developer Site.

This documentation was created using Xcode 4.6 and the iOS Simulator.

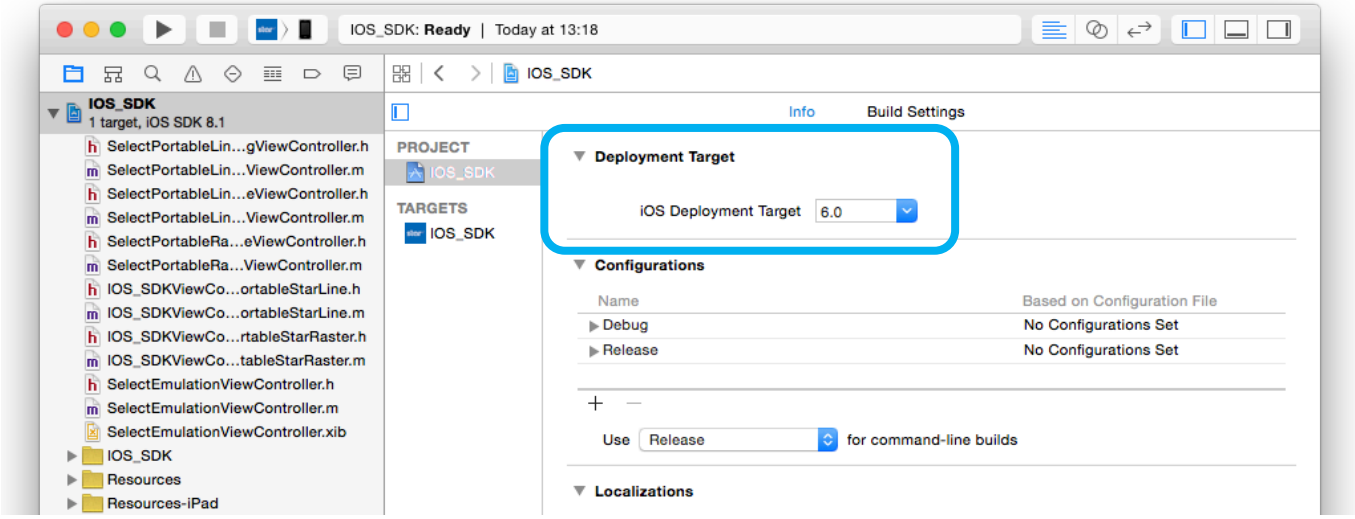
How to open the Star iOS SDK project in Xcode:

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

2. Open IOS_SDK.xcodeproj.

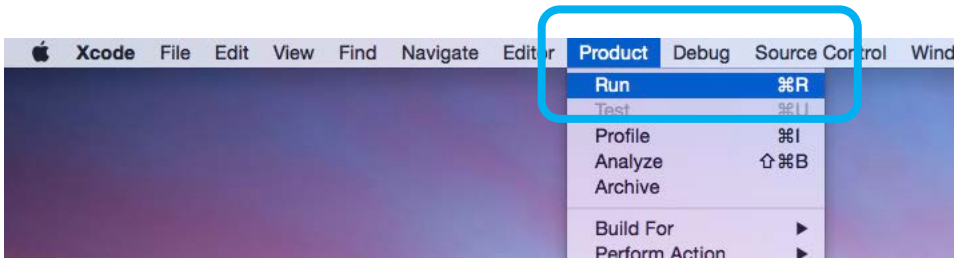


3. Set the iOS Deployment Target to 5.1.1 or later.



Running the project:

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



Using the SDK with Star Devices

About the BLED10-U

The BLED10-U uses radio waves and the radio intensity varies depending on:

- the actual radio wave environment
- an installation place of the BLED10-U
- a direction of the iOS Device
- an angle of the iOS Device
- a user's handling way of the iOS device, and so on.

When creating your App, design the threshold value setting to be conducted by holding your iOS device in the same direction as the actual use, and do not allow the auto rotation of the screen.

When you specify the threshold value for the BLED10-U, use the actual device in the actual operation environment in conformity with the use conditions of the iOS device. If you can not obtain the stable operative condition at the actual operation, the operating conditions for setting the threshold value such like a direction and an angle of the iOS device may not be reproduced.

Before starting the actual field operation, please check sufficiently that the device does not work in the uninteded environment and adjust the threshold.

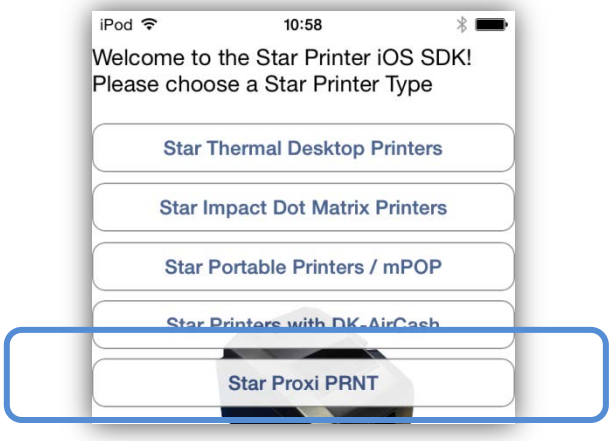
Port Name and Interface Relation:

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
Wired LAN / WirelessLAN I/F(TCP/IP)	TCP:“IP Address”
Bluetooth	BT:”iOS Port Name” ※

◆Using a ProxiPRNT:

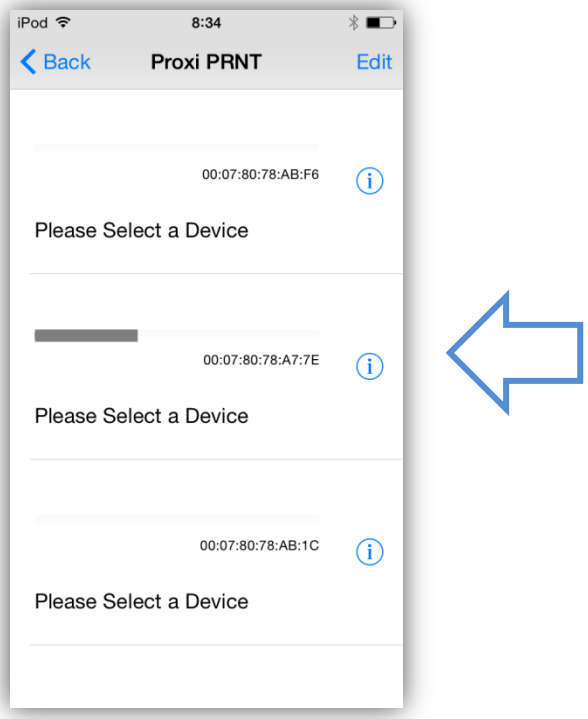
1. Tap “Star Proxi PRNT”.



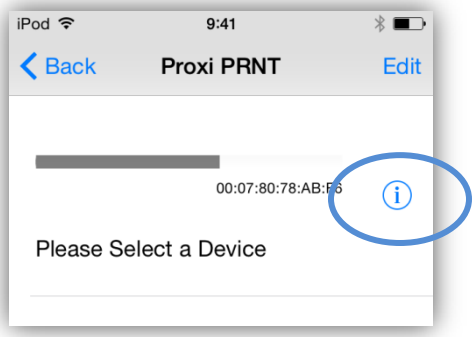
·Connecting the BLED10-U with the Star POS device

1. Connect the BLED10-U to the USB connector (TYPE-A) to supply power.

   Note:When using multiple BLED10-Us, you can identify the device by moving your iOS device close to the BLED10-U and checking that the RSSI level has increased.

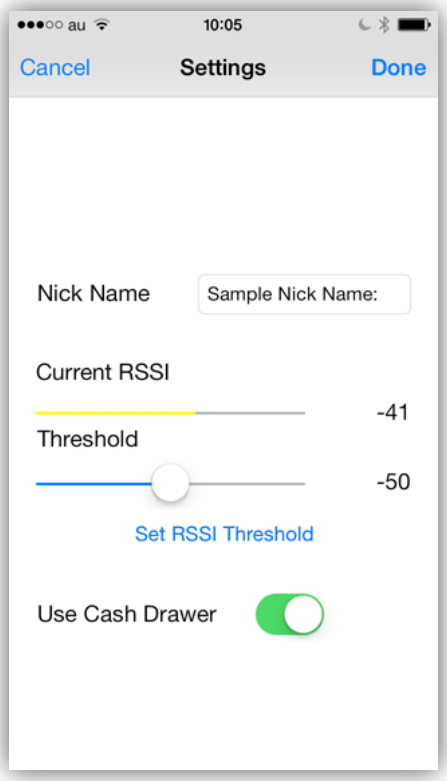


2. The RSSI bar of the BLED10-U is displayed. Tap to  go to the setting window.



3. All the Star Ethernet devices available for your iOS device and all the Star Bluetooth devices paired with your iOS device are displayed. Select the Star POS device to connect with the BLED10-U to be configured and tap it.

4. Configure the setting for the selected Star POS device. The setting contents are different for every Star POS device. After completing the configuration of the necessary settings, tap "Done".



Nick Name

Enter the Nick name to identify the connection of the BLED10-U and the Star POS device.

Current RSSI

The actual RSSI is shown in blue or yellow.

Threshold

Set the RSSI threshold.
Set the device in the actual operation environment and tap [Set RSSI Threshold].
Check the RSSI in more than one place and adjust the threshold value. See here for details.

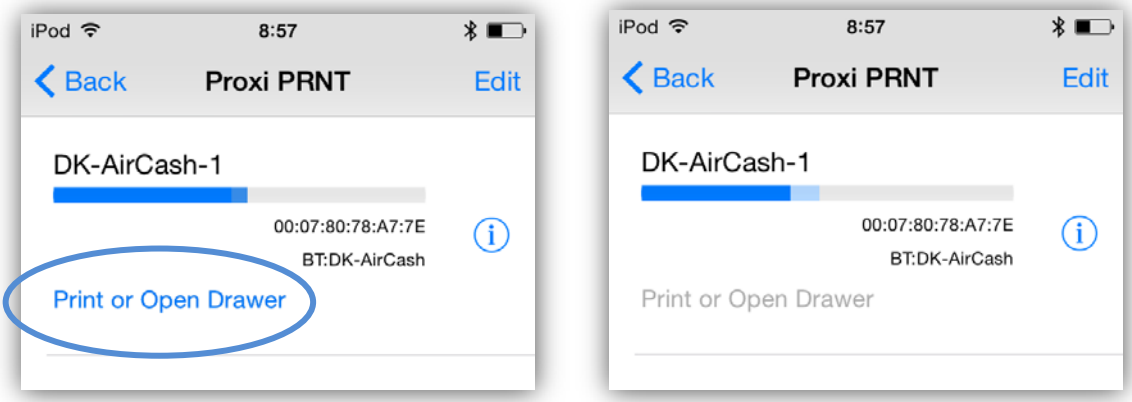
Use CashDrawer

Set to this switch On when using a cash drawer.

5. The threshold set to the RSSI bar is shown in blue and the actual signal strength is shown in light blue. If the signal strength is less than the threshold, it is displayed in deep blue.

   If the RSSI exceeds the threshold, the "Print or OpenDrawer" will be displayed and it can be operated.

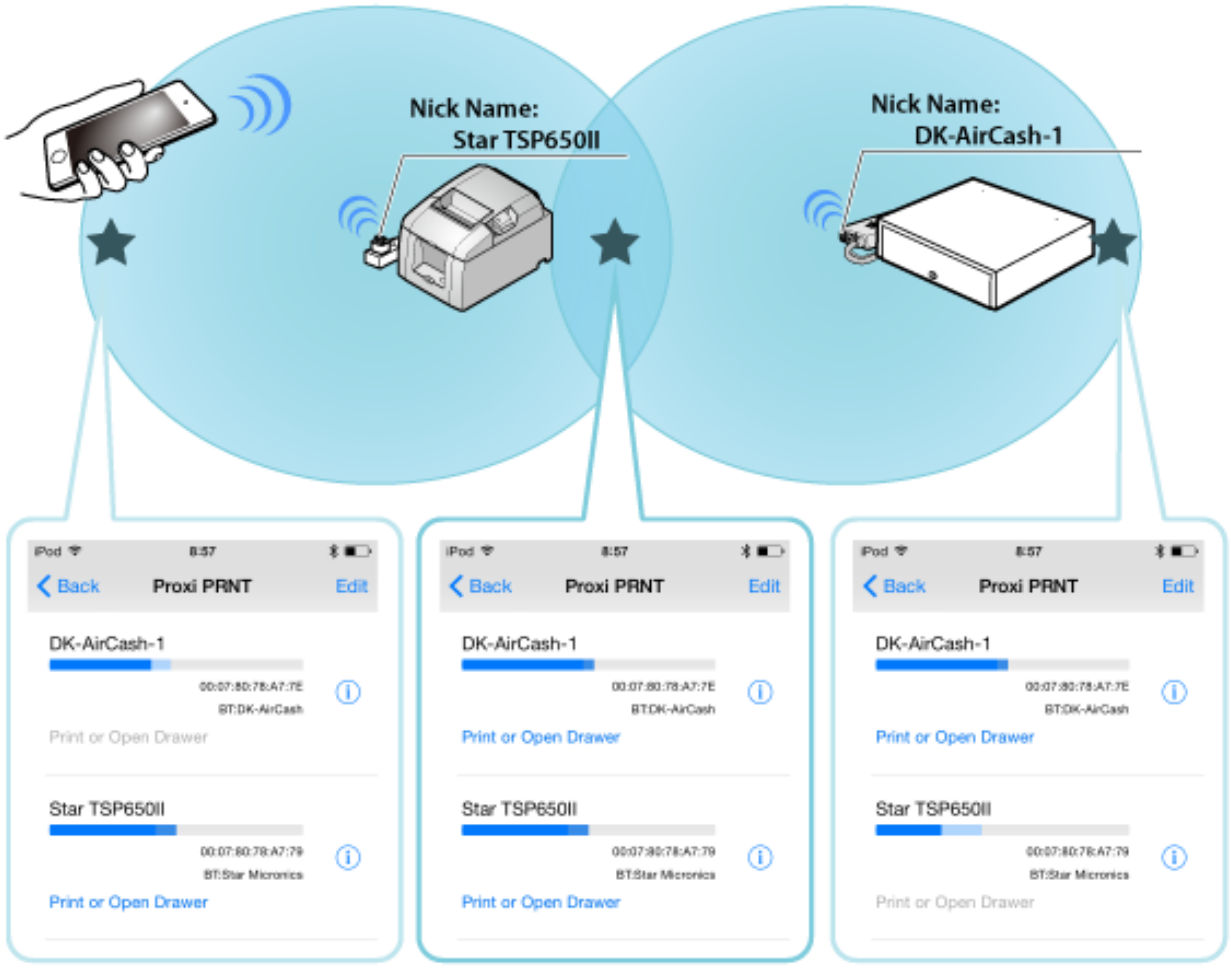
   Check that the device can not work in the unintended environment (as shown in the right screen).



Tap the "Print or OpenDrawer" to perform the test print and open the drawer (if selected).

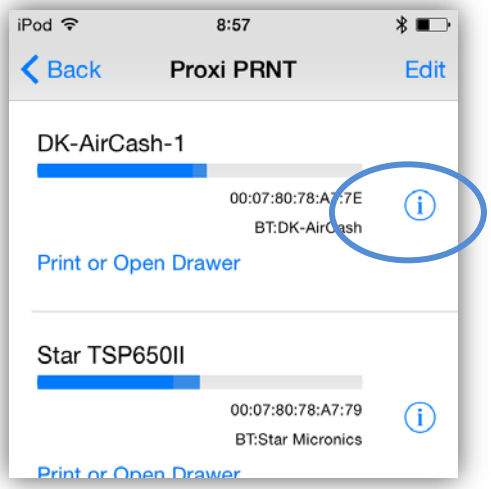
* The raster mode is used for the test print in this SDK.

Example) The RSSI level at each use position (a star mark) when using multiple BLED10-Us



Delete/edit the connection of the BLED10-U and the Star POS device

1. Tap  shown on the right of the RSSI bar to edit the settings of the connection.



When the Bluetooth device name or the IP address of the connected device is changed after connection has been configured, be sure to change the Port Name of the connection information.

Overview of How This iOS SDK is Designed

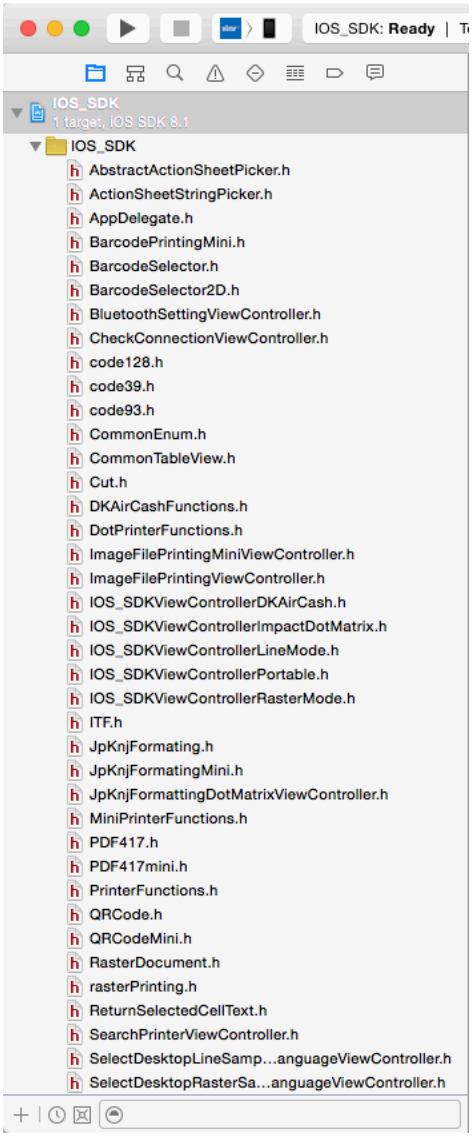
This overview will touch briefly on key components of the SDK.

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 Mobile 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.



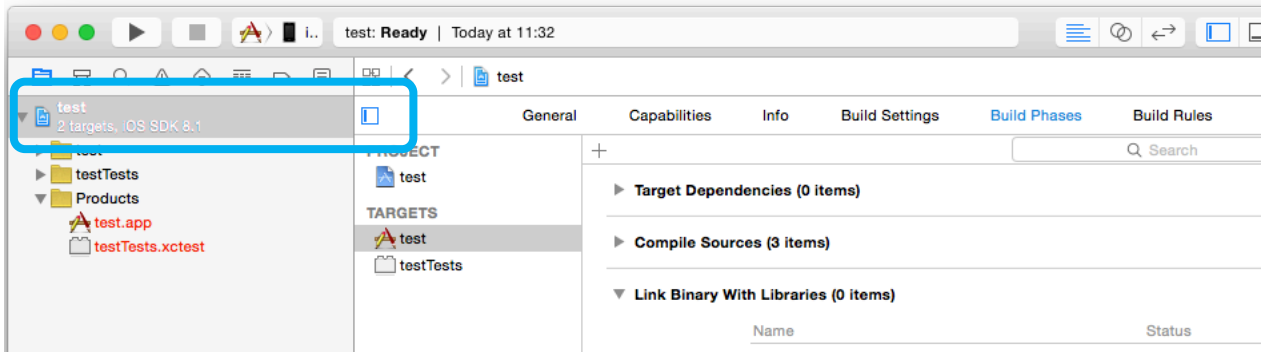
The StarIO Framework

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 into it to utilize the StarIO methods.

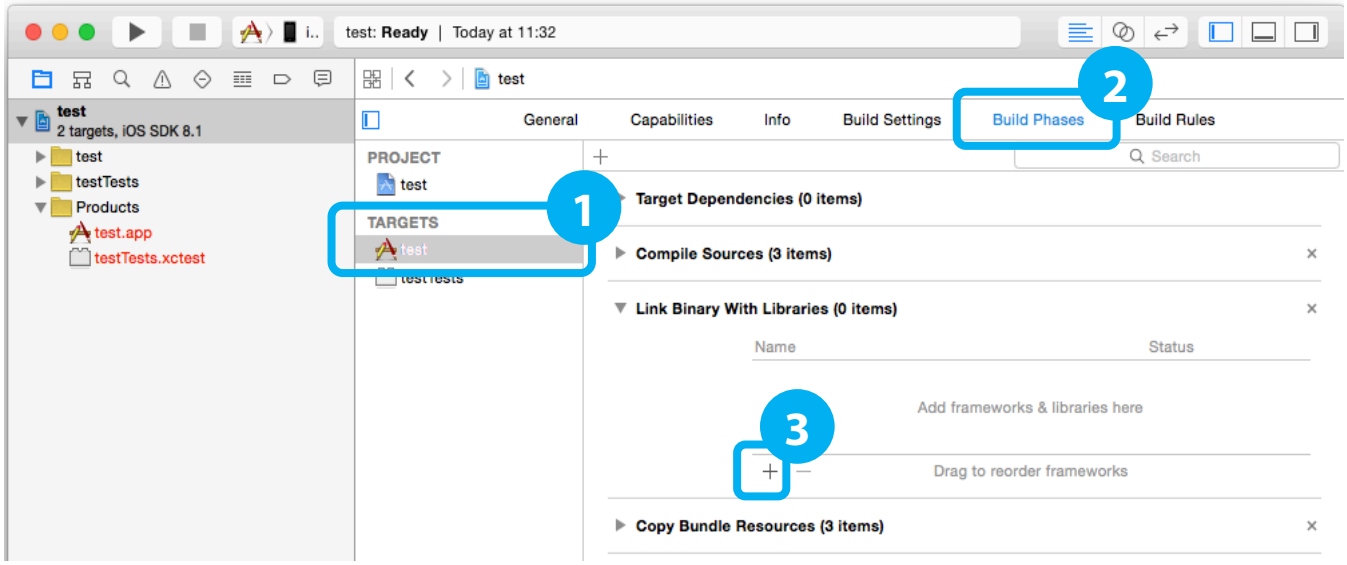
◆When building a new application

1. Add StarIO.framework into your project

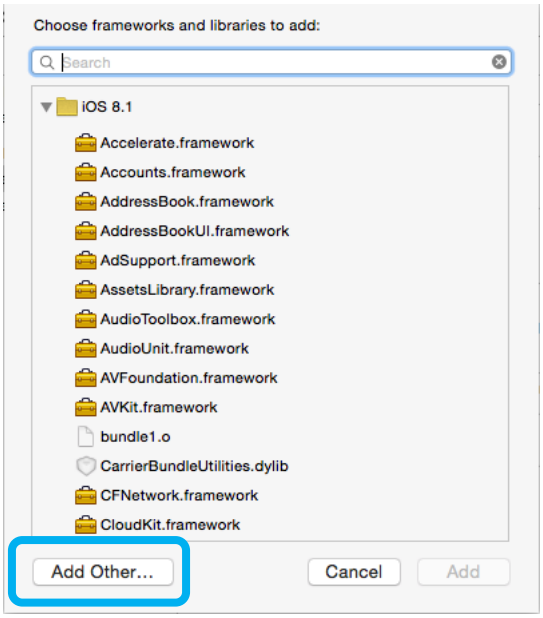
1. Click the created project.



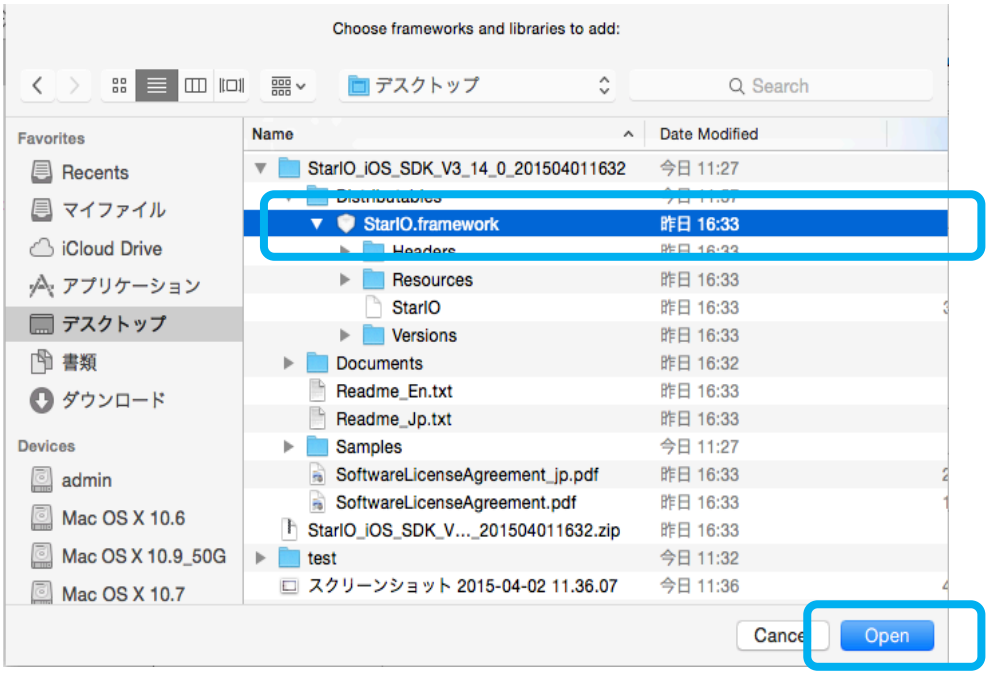
2. Open a target, click the Build Phases tab, click the + of Link Binary With Libraries.



3. Click the Add Other… button.



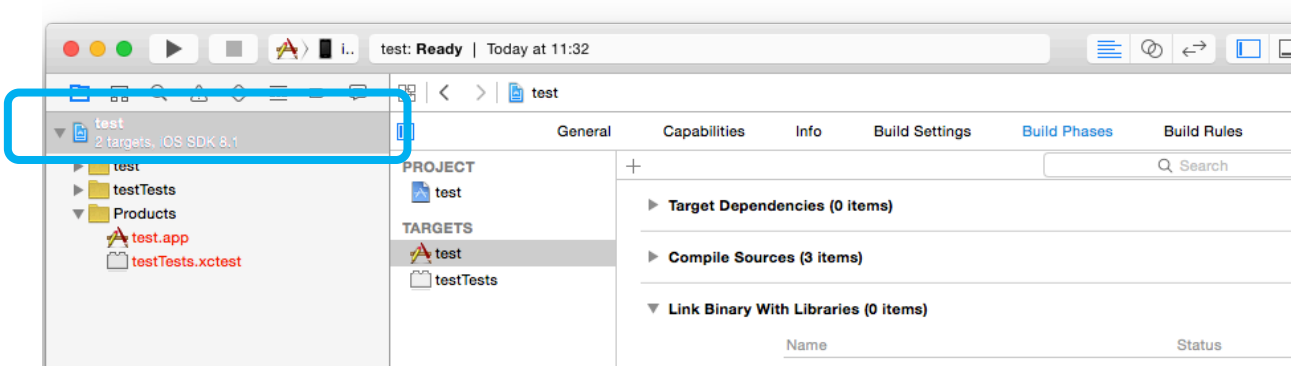
4. Browse to the location of where the Star iOS SDK was unzipped and select StarIO.framework. Then click Open.



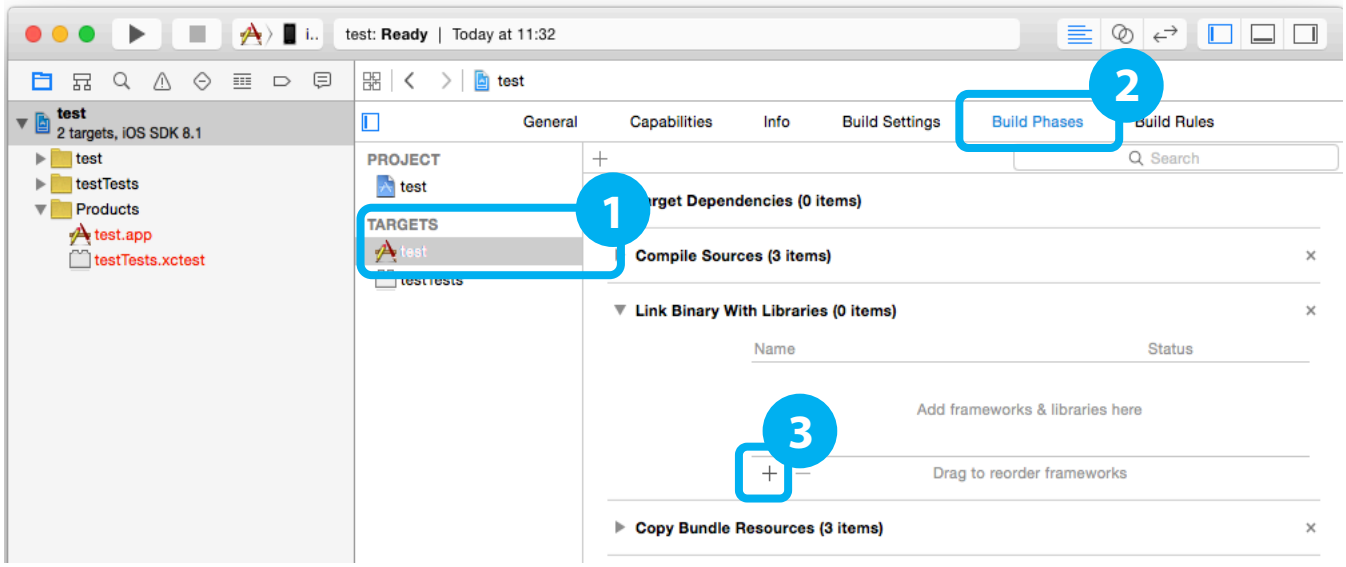
5. The framework is added to your project and all StarIO methods are now available to you.

2. Add other framework into your project.

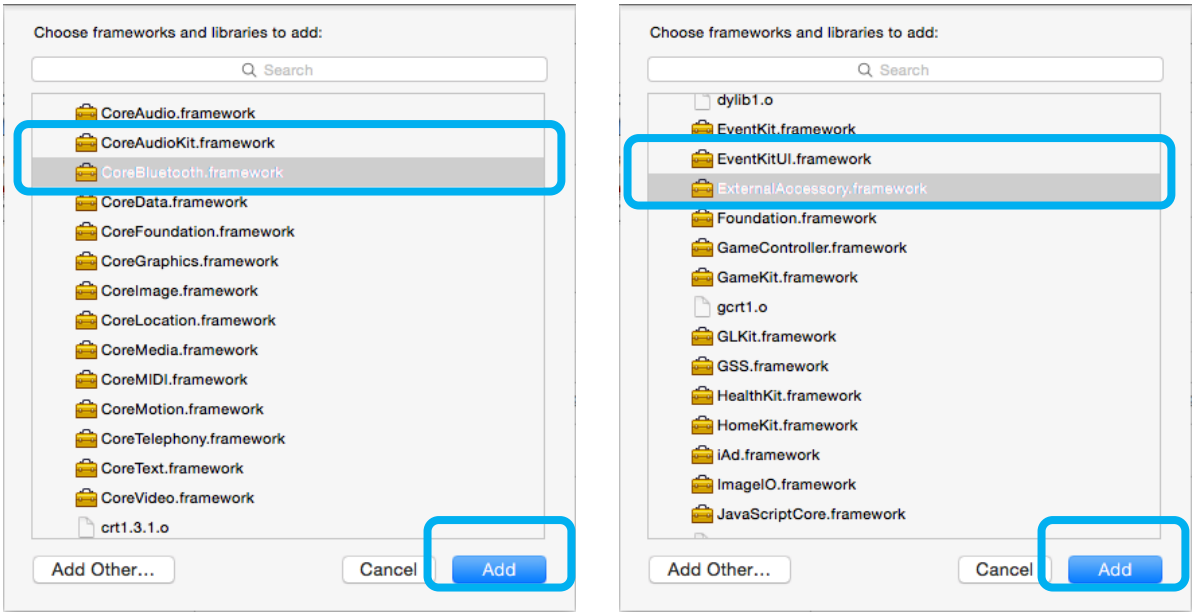
1. Click the created project.



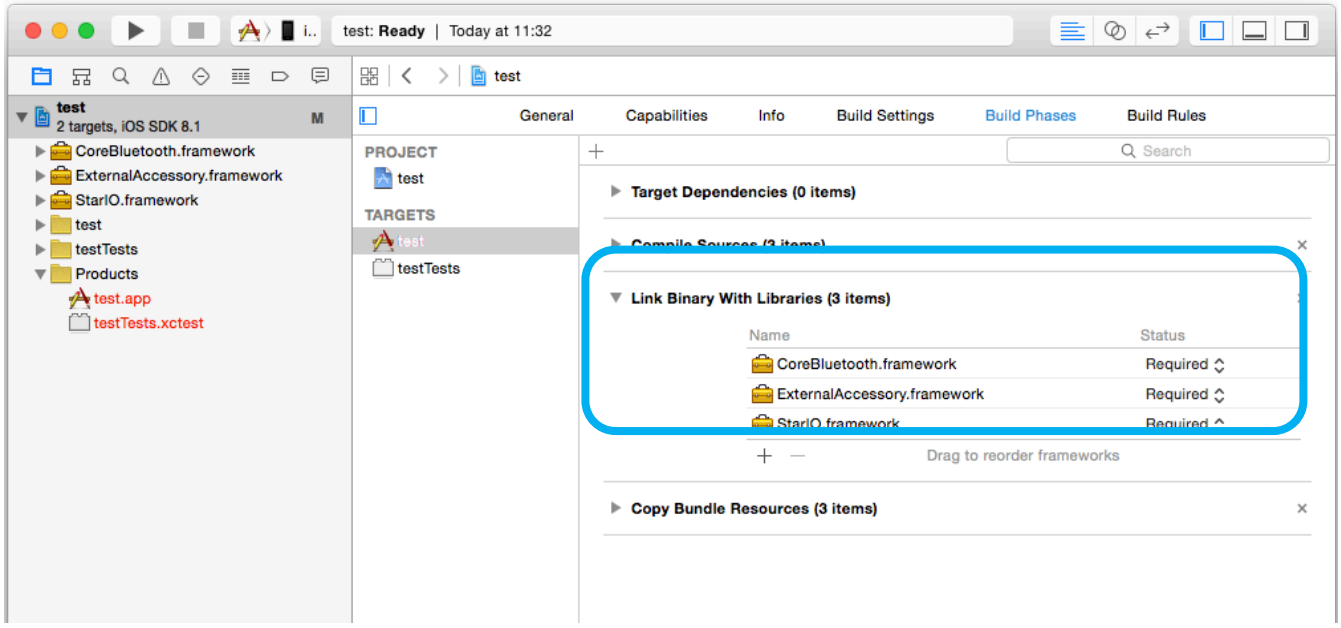
2. Open a target, click the Build Phases tab, click the + of Link Binary With Libraries.



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



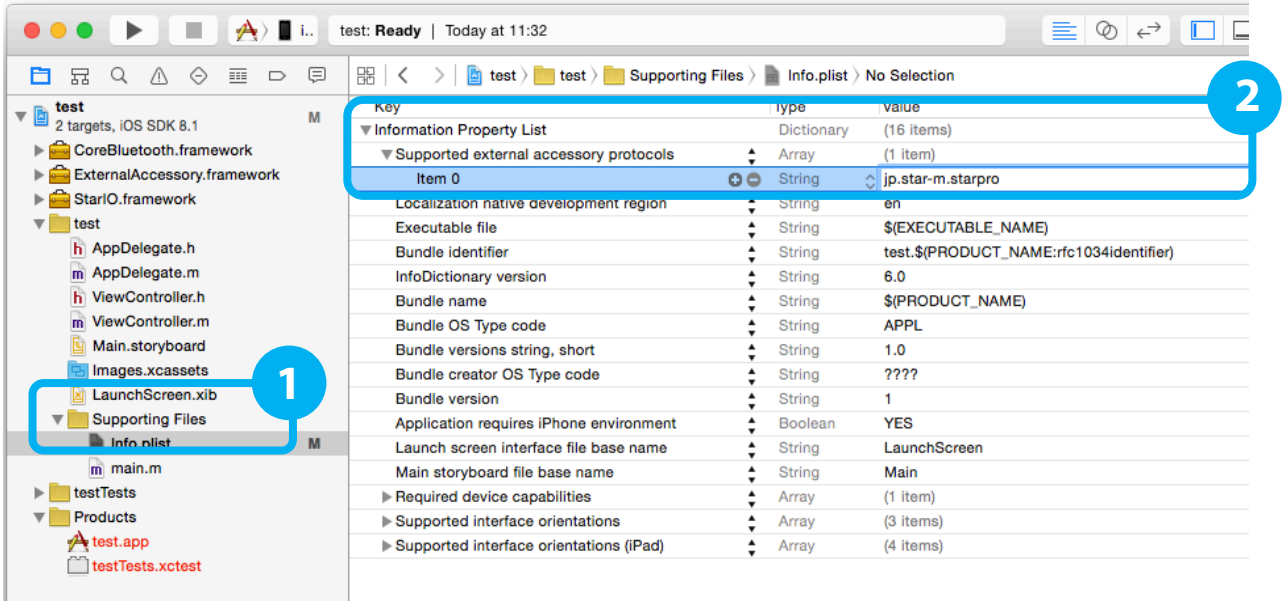
4. Check if the necessary framework has been added.



3. Edit information property list (Bluetooth Interface 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.

```
- (NSString *)portName
```

Specifies the port of the printer.

```
- (NSString *)portSettings
```

Acquires 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.

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

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

● 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 Wired LAN / Wireless LAN)*
@”BT:StarMicronics” (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.

[Use share printer function with Apple AirPort Express]
Set AirPort Express IP Address for **portName**.
Ex. @”TCP:192.168.1.2”

portSettings – when using a wireless LAN, specify @”;wl”.
- when using a connection other than a wireless LAN, specify a empty string (@””).

Ex. *@”” (for Wired LAN or Bluetooth)*
@”wl” (for Wireless LAN)

- Specify @”d[value]” for portSettings to set the Data timeout function* ON.

*** Data timeout function**
In case an error occurs while printing or no data is sent to the printer during a predetermined time, the data canceling function will be executed.
To set the time value, specify from 0 to 255 (unit: second) in [value].
The default value is 3 seconds. If a value outside of the range is specified, it will be ignored.

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

The data timeout function is supported by TSP650II and TSP800 with F/W version 2.0 or later and TSP700II with F/W version 5.0.

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

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).

[Use share printer function with Apple AirPort Express]

Set port number for **portSettings**.

Increase the port number in sequential order from 9100 to 9109 until communication is successful.

Ex. @"9100"

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

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 afterward may fail.

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

SMPort *port = nil;
NSString *portName = @"TCP:192.168.0.5";
NSString *portSettings = @"";
@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.

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 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:

`PortException` - when a communication failure occurs

releasePort

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

This function closes a connection to the port specified.

Parameters:

port – StarIOPort type representing a previously initialized port.

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

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:

To check the completion of printing, run beginCheckedBlock before and endCheckedBlock after this method.

See the sample code here.

※Remember to use a Try, Catch for safe programming practices.

The SDK has code in “PrintTextWithPortName” that will show you how to verify data transmission to the printer.

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_2 printerStatus;
[port getParsedStatus:&printerStatus :2];
if (printerStatus.offline == SM_TRUE)
{
    if (printerStatus.coverOpen == SM_TRUE) {
        //There was a cover opne error
    }
    else if (printerStatus.receiptPaperEmpty == SM_TRUE) {
        //There was a receipt paper empty error
    }
    else {
        //There was a offline error
    }
}
else {
    //If False, then the printer is online.
}
```

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.

F/W Version 3.0 or later is required for TSP650 and TUP500.

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

F/W Version 3.0 or later is required for TSP650 and TUP500.

```
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];
}
```

disconnect

```
-(BOOL) disconnect
```

This method disconnects the specified Bluetooth device.
After the disconnection, the Bluetooth device can be connected by other iOS terminals.

This method fails in the following cases:

- when the disconnection has not been completed within the timeout specified by getPort
- when the disconnection function is not supported by a printer (such like portable printers).

This method has no effect on Ethernet devices.

Returns:

It returns YES when succeeded and NO when failed.
It always returns YES when it was run with the Ethernet device.

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.
- It is impossible to get the firmware version of TSP100U, TSP100GT, TSP100LAN and TSP100ECO.
- When using Apple AirMac Express with a USB printer, it returns an empty string.

SMProxiPRNTManager Class:

StarProxiPRNTManager Class utilizes the RSSI broadcast by the dongle (BLED10-U) which supports Bluetooth Low Energy to use a POS printer or a DK-AirCash placed nearby.

● Property

delegate	Specifies the delegate.
settings	Gets the setting information of the ProxiPRNT.

```
@property(retain, nonatomic) id<SMProxiPRNTManagerDelegate> delegate
```

Specifies the delegate.

```
@property(retain, readwrite) NSDictionary* settings
```

Gets the setting information of the ProxiPRNT.
Use the following methods to add or delete the setting.

- addSettingForPrinterPortName:portSettings:withDrawer:MACAddr:RSSIthreshould:nickName:
- addSettingForDKAirCashPortName:portSettings: MACAddr:RSSIthreshould:nickName:
- removeSettingWithPortName:

● Method

sharedManager

```
+(SMProxiPRNTManager *) sharedManager
```

Gets an instance of SMProxiPRNTManager

addSettingForDKAirCashPortName

```
- (void) addSettingForDKAirCashPortName:(NSString *)portName portSettings:(NSString *)
                portSettings dongleMACAddr:(NSString *)MACAddr RSSIthreshold:(NSNumber *)Threshold
                nickName:(NSString *)nickName
```

Adds the settings of the DK-AirCash to be connected with the BLED10-U.

Parameters:

portName	– Port name of the connect printer
portSettings	– The settings for the port (See here for details)
dongleMACAddr	– Mac address of the BLED10-U
RSSI threshold	– Threshold
nickName	– Nick name to connect the BLED-10U and the DK-AirCash

Throws:

StarIOPortException

- when portName or dongleMACAddr is nil
- when the same portName has been specified
- when the same dongleMACAddr has been specified

addSettingForPrinterPortName

```
- (void) addSettingForPrinterPortName:(NSString *)portName portSettings:(NSString *)
                portSettings withDrawer:(BOOL)useDrawer dongleMACAddr:(NSString *)MACAddr
                RSSIthreshold(NSNumber *)Threshold nickName:(NSString *)nickName
```

Adds the settings of the POS Printer to be connected with the BLED10-U.

Parameters:

portName	– Port name of the connect printer
portSettings	– The settings for the port (See here for details)
withDrawer	– Presence/absence of a cash drawer
dongleMACAddr	– MAC address of the BLED10-U
RSSI threshold	– Threshold
nickName	– Nick name to connect the BLED-10U and the POS Printer

Throws:

StarIOPortException

- when portName or dongleMACAddr is nil
- when the same portName has been specified
- when the same dongleMACAddr has been specified

getRSSI

```
- (int) getRSSI:(NSString *)MACAddr
```

Gets the RSSI value of the connected printer.

Parameters:

MACAddr	- UUID of the BLED10-U

Return:

RSSI value

Returns 127 when the MACAddr device is not found.

removeSettingsWithPortName

```
- (void)removeSettingsWithPortName:(NSString *)portName
```

This method removes the setting information connected to the BLED10-U by the port name.

deserializeSetting

```
- (BOOL)deserializeSetting:(NSData *)data
```

Restores the setting information connected with the BLED10-U and sets them.

Returns:

YES	- when the restoration is succeeded
NO	- when the restoration is failed

serializedSettings

```
- (NSData *)serializedSettings
```

This method acquires the data of byte string of the setting connected with the BLED10-U.

startScan

```
- (BOOL)startScan
```

This method starts a scan of the Bluetooth Low Energy device.

Returns:

YES	- when a scan is operated correctly
NO	- when a scan is not operated correctly

Do not execute this method when printing data or opening the drawer. Execute stopScan method and stop scanning BLE device before printing data or opening the drawer.

startScan

```
- (BOOL)startScan:(NSUInteger)samplingNumber;
```

This method starts a scan of the Bluetooth Low Energy device.

Parameters:

samplingNumber	- set 3, 7 or 10

Returns:

YES	- when a scan is operated correctly
NO	- When the sampligNumber is spedified to 0

Do not execute this method when printing data or opening the drawer. Execute stopScan method and stop scanning BLE device before printing data or opening the drawer.

stopScan

```
- (void)stopScan
```

This method stops a scan of the Bluetooth Low Energy device.

Execute stopScan method and stop scanning BLE device before printing data or opening the drawer.

calibrateActionArea

```
- (int)calibrateActionArea:(NSString *)MACAddr
```

This method returens the maximum RSSI value approximately five seconds later.
It returns 127 when acquisition of the maximum value fails.

Parameters:

MACAddr - MAC Address of the BLED10-U

Execute stopScan method and stop scanning BLE device before executing this method.

SMProxiPRNTManager Delegate

● Delegate

didUpdateState

```
@optional
- (void) didUpdateState:(NSString *)portName MACAddr:(NSString *)MACAddr
```

Notifies the Mac address of the BLED10-U and the port name of the printer or the drawer connected with the BLED10-U when the current RSSI value cannot be acquired for 5 seconds or more during a scan of the Bluetooth Low Energy device started by the startScan method.

Parameters:

portName	– Port name of the Printer or the drawer connected with the BLED10-U
MACAddr	– MACAddr of the BLED10-U

didDiscoverPort

```
@optional
- (void) didDiscoverPort:(NSString *)portName deviceType:(SMDeviceType)deviceType
                  MACAddr:(NSString *) MACAddr RSSI:(NSNumber *)RSSI
```

Notifies the Bluetooth Low Energy device discovered during a device scan started by the startScan method.

Parameters:

portName	– port name of the printer or the drawer connected with the BLED10-U The port name is an empty string if no device is connected.
deviceType	– Type of the device which the current RSSI value is acquired for.
MACAddr	– MACAddr of the BLED10-U
RSSI	– RSSI value

Status List of the class structure StarPrinterStatus

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

Class structure StarPrinterStatus Supported

Member name	TSP 100 LAN	TSP 100 U	TSP 100 GT	TSP 100 IIU	FVP 10	TSP 650	TSP 650 II	TSP 700 II	TSP 800 II	TUP 500	DK-Air Cash
blackMarkError					✔			✔	✔	✔	
compulsionSwitch	✔	✔	✔	✔	✔	✔	✔	✔	✔		✔
coverOpen	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
cutterError	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
etbAvailable	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
etbCounter	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
headThermistorError										✔	
offline	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
overTemp	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
presenterPaperJamError										✔	
presenterState										✔	
raw	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
rawLength	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
receiptPaperEmpty	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	
receiptPaperNearEmptyInner					✔	✔	✔	✔	✔	✔	
receiveBufferOverflow					✔	✔	✔	✔	✔	✔	
unrecoverableError	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
voltageError										✔	

Tips for App Development when using StarIO

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

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

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

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

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

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

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

Additional Resources

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

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

Star Micronics Developers Network

Browse Star Micronics' FAQs, 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
Jul. 2014	3.11.0	- Initial Release
Aug. 2015	3.15.0	- The sampling frequency of startScan method can be changed. - The identification method of the BLE dongle is changed. - DK-AirCash Wireless LAN support.
Mar. 2016	3.16.0	- Add devices support.

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

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

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

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

Star Micronics Worldwide

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

Star Micronics America, Inc.
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
+66-2-631-1161 x 2
http://www.starmicronics.co.th/