iOS Software Development Kit

Star プリンター (StarPRNT)向け

StarIO 使用方法

本 SDK には iOS デバイスのための Xcode Objective-C プロジェクトを含んでいます。

必要なツール:

- Xcode 7.0 以降
- StarIO iOS SDK

3.14.0 以降の StarIO.framework をご使用の際には、合わせて以下の framework を追加いただく必要があります。

・External Accessory framework

・Core Bluetooth framework

※既に 3.13.1 以前をご使用で StarIO.framework のバージョンアップを行う場合、新たに Core Bluetooth framework のプロジェクトへの追加が必要です。

詳しくはこちらをご参照ください。

❖ モバイルプリンター

StarPRNT エミュレーションを使用するには

StarPRNT エミュレーションを使用するには、プリンター本体のエミュレーションを“Star Line Mode”に設定する必要があります。（SM－L200 は切り替え不要です。）
エミュレーションの切り替えは以下の手順で行ってください。

◆ ESC/POS ⇔ StarPRNT エミュレーション切り替え方法

1. プリンターの電源を入れ、プリンターカバーを開きます。
2. 電源ボタンと FEED ボタンを同時に長押しした後、ERROR ランプが 5 回点滅したことを確認しすぐに電源ボタンと FEED ボタンから指を放します。エミュレーションの切り替えが自動的に行われます。
3. 用紙をセット後、プリンターカバーを閉めると設定されたエミュレーションが印字されます。
   ESC/POS の場合： EMU = ESC/POS Mode
   StarPRNT の場合： EMU = Star Line Mode

   エミュレーションが正しく切り替わっていない場合、再度 1～3 の手順を行ってください。
   その際、2 の手順においては、点滅中に指を放さなさず、点滅が 5 回完了したことを確認してから指を放すように注意してください。
4. エミュレーションの切り替え後は、プリンターの電源を一度オフにしてから再投入してください。
   選択したエミュレーションは、プリンターの電源を再投入することで有効になります。

StarIO SDK 対応 OS :　　iOS 7.0 以降

StarIO SDK 対応リスト

デバイス	CPU
iPad 2 *	Armv7
iPad (第 3 世代)	Armv7
iPad (第 4 世代)	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 (第 5 世代)	Armv7
iPod touch (第 6 世代)	Arm64

* BLE をサポートしていないため、SM-L200 は使用できません。

注）iPad、iPhone、iPod、iPod touch は、米国および他の国々で登録された Apple Inc.の商標です。iPad Air、iPad mini、は、Apple Inc. の商標です。“iPhone”の商標は、アイホン株式会社のライセンスにもとづき使用されています。iOS は、米国およびその他の国における Cisco 社の商標または登録商標であり、ライセンスにもとづき使用されています。

目　次

本書に関して

本マニュアルは、StarIO と Star プリンターが通信を行う、iOS アプリケーションの作成方法を解説しています。

また、このマニュアルは、アプリケーション・システム開発者を対象に作成しており、利用者は Objective-C 言語の基礎を理解していることを前提としています。

この SDK は iOS 用に作成されています。
スター精密グローバルサポートサイトの Developers セクションには、その他のオペレーティングシステムとプログラミング言語に利用可能な SDK が用意されています。最新の SDK、テクニカルドキュメント、FAQ 及び、その他の追加情報については、Developers セクションをご確認ください。

表示マークの説明:

警告		潜在的な問題について説明します。
禁止		禁止事項について説明します。
メモ		重要な情報とヒントを提供します。

注意事項:

- 本マニュアルの内容は、予告無く変更する場合があります。
- スター精密株式会社は、正確な情報を提供するためにあらゆる措置を取っていますが、誤りや不作為について責任を負うものではありません。
- スター精密株式会社は、このマニュアルに記載されている情報の使用に起因するいかなる損害に対しても責任を負うものではありません。
- 本マニュアルの一部、あるいは全部を無断で複写・複製・転載することは、固くお断りします。

Star プリンター 対応リスト

iOS における Star モバイルプリンターのモデル別対応リストを以下に記します。

✓ : Line Mode,ESC/POS 対応　　✓ : Raster Mode,ESC/POS 対応　　✓ : Raster Mode 対応

Star プリンター			Port Discovery	Get Firmware Information	Get Status	Sample Receipts	1D Barcodes	2D Barcodes	Text Formatting	Raster Graphics Text Printing	Image File Printing	MSR	AllReceipts	Bluetooth Setting
モデル	インターフェイス	F/W Ver.												
SM-S210i (JP モデルのみ)	Bluetooth	3.0 以降	✓	✓	✓	✓*2	✓	✓	✓	✓	✓	✓	✓	✓
SM-S220i (欧米モデルのみ)	Bluetooth	3.0 以降	✓	✓	✓	✓*2	✓	✓	✓	✓	✓	✓	✓	✓
SM-S230i (欧米モデルのみ)	Bluetooth	1.0 以降	✓	✓	✓	✓*2	✓	✓	✓	✓	✓	✓	✓	✓
SM-T300i	Bluetooth	3.0 以降	✓	✓	✓	✓*2	✓	✓	✓	✓	✓	✓	✓	✓
SM-T400i	Bluetooth	3.0 以降	✓	✓	✓	✓*2	✓	✓	✓	✓	✓	✓	✓	✓
SM-L200	Bluetooth	1.0	✓	✓	✓	✓*3	✓	✓*4	✓	✓	✓	✓*5	✓	✓

	Low Energy[*1]	以降												
mPOP	Bluetooth	1.0 以降	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	

[*1] Bluetooth Low Energy は SM-L シリーズを iOS 端末と接続して使用する場合の通信手段です。
[*2] Line Mode では、Russian(ロシア語)、SimplifiedChinese(簡体字)に対応しておりません。
[*3] Line Mode では、SimplifiedChinese(簡体字)はファームウェアバージョン 1.1 から対応します。
[*4] PDF417 はファームウェアバージョン 1.1 から対応します。
[*5] MSR はファームウェアバージョン 2.0 から対応します。

Star プリンターを iOS デバイスに接続するには

Bluetooth インターフェイス

Star デバイスは、工場出荷時の初期設定では各機種ごとに"Star Micronics"等、共通の Bluetooth デバイス名が設定されています。同じ Bluetooth デバイス名の機種を複数台配置して運用される場合、Bluetooth デバイス名の変更を行うと Star モバイルデバイスの判別が付けやすく便利です。

Bluetooth デバイス名の変更等、Star デバイスの Bluetooth/Bluetooth Low Energy 設定値は、スター精密の提供する Star Setting Utility を使用して変更することができます。Star Setting Utility は App Store よりダウンロードしてください。

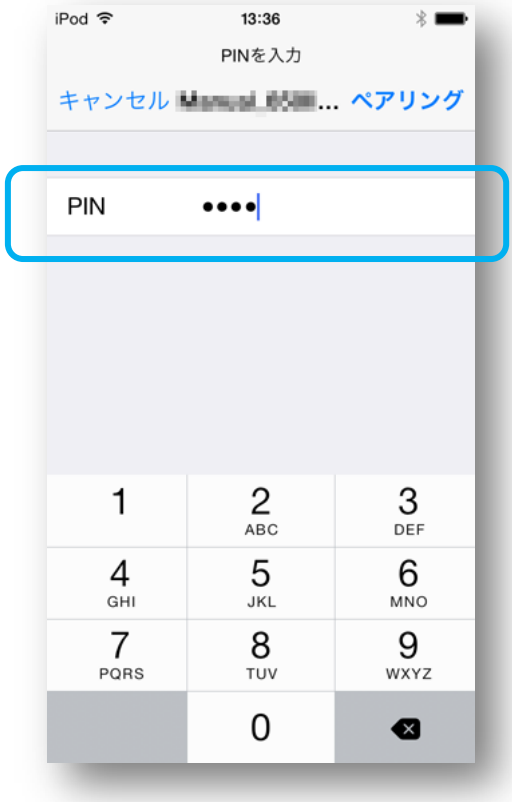
◆ペアリング

iOS デバイスと同時にペアリングする Star プリンターは、1 台のみとすることを推奨します。また、Bluetooth Low Energy を使用する SM-L200 では不要です。

1. Star プリンターを、ペアリングを行う iOS デバイスと接続が可能な範囲に設置して電源を投入します。
2. iOS の[設定]より、[Bluetooth]をタップします。

3. [Bluetooth]を ”オン”に設定すると、iOS デバイスとペアリングが可能な Bluetooth デバイスの検索を行い、表示します。ペアリングを行う Star プリンターをタップします。

4. PIN を入力します。（モバイルプリンターのみ）



5. 以下の表示が確認できればペアリング完了です。

◆Bluetooth 名称の変更

App Store から Star Bluetooth Utility をダウンロードし、iOS ポート名を変更することができます。必要に応じてご使用ください。

iOS ポート名はペアリング実行後、以下の手順で確認できます。

「設定」-「一般」-「情報」　Bluetooth アドレスの下に表示

はじめに

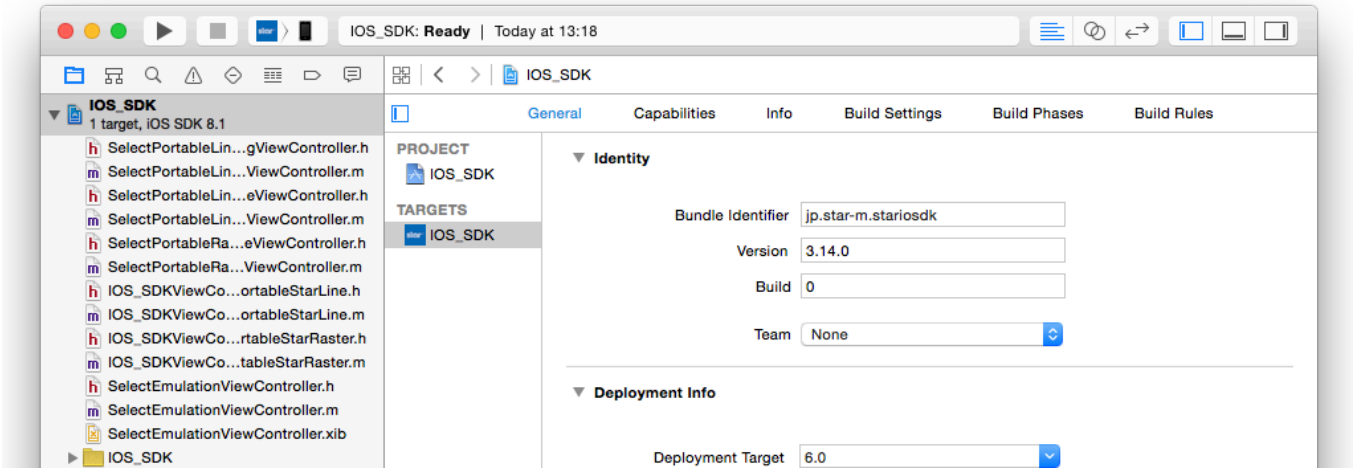
iOS のプロジェクトをビルドするには、Xcode が必要です。これらのツールは、Apple Developer サイトや Mac App Store から入手可能です。実際に iOS デバイス上で動作するアプリケーションを生成するには、Apple の Developer Program への登録が必要です。(Developer Program は年一度の更新手続きを必要とします)　Developper Program への登録を行わずに iTunes からこれらのツールを入手することは可能ですが、その場合、アプリケーションは iOS シミュレーター上で動かすことができるだけであって、実際の iOS デバイスにはインストールされません。

事前に、開発を行う Mac に Xcode のインストールを行ってください。万一、サポートまたは追加情報が必要な場合は、Apple Developer サイトの Resources セクションを参照してください。

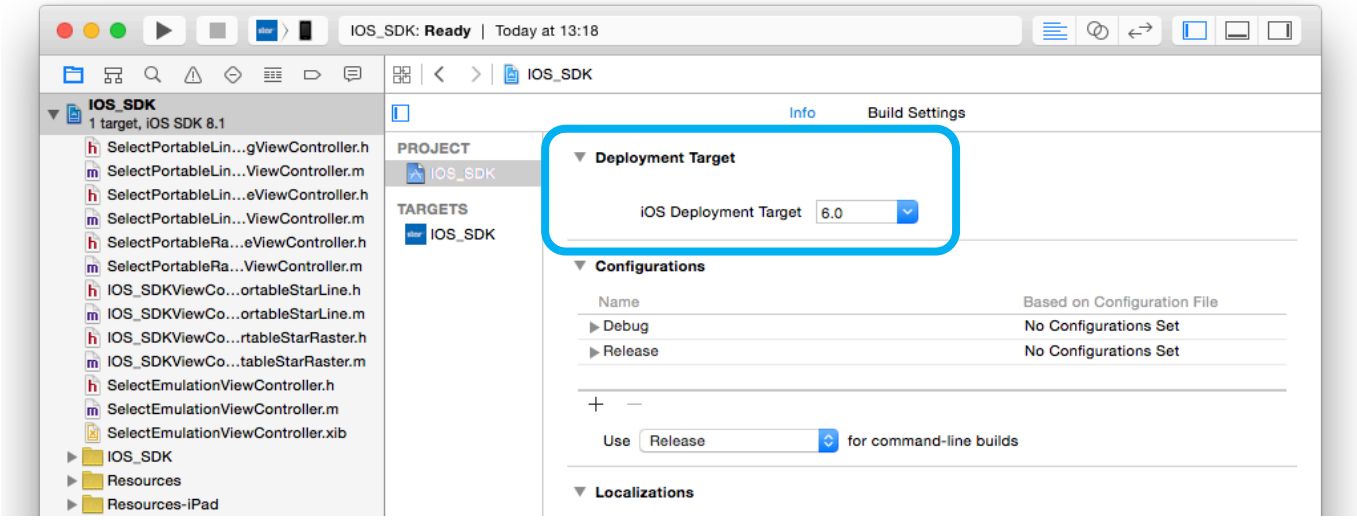
Xcode で Star iOS SDK プロジェクトを開く方法：

1）Star iOS SDK フォルダを解凍し、開きます。

2）IOS_SDK.xcodeproj を開きます。

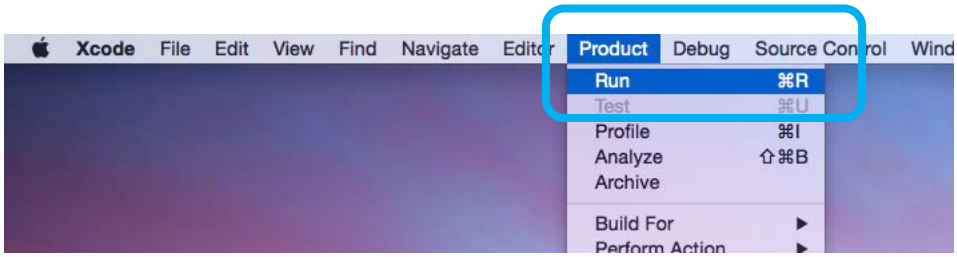


3）iOS Deployment Target の欄にて 6.0 以降を選択してください。



プロジェクトを実行する：

1） ショートカットの“⌘R”を使用するか、上部メニューバーの“Product - Run”をクリックして実行します。



Star プリンターで SDK を使用する

iOS に対応する Star プリンターがお手元にあることをご確認ください。

ポート名の設定:

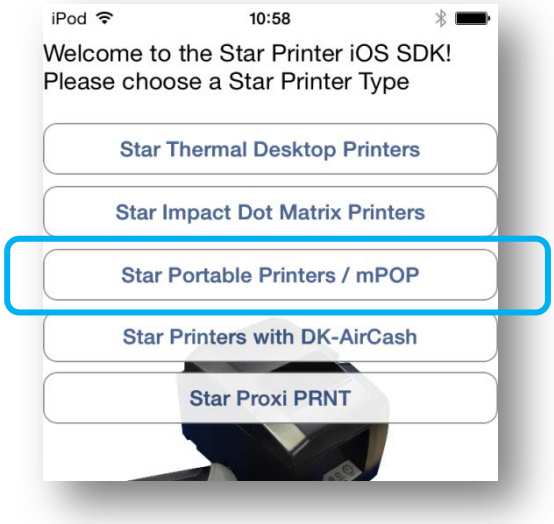
StarIO は、プリンターと通信するために特定のポート名を使用します。
ポート名は、以下の通り正しく設定しないとプリンターとの通信を行えません。また、用途に合わせて PortSettings 等の設定も必要です。詳しくはこちらをご参照ください。

Interface	Port Name
Bluetooth	BT:“iOS ポート名”
Bluetooth Low Energy	BLE:“デバイス名” * BLE:“MAC アドレス”

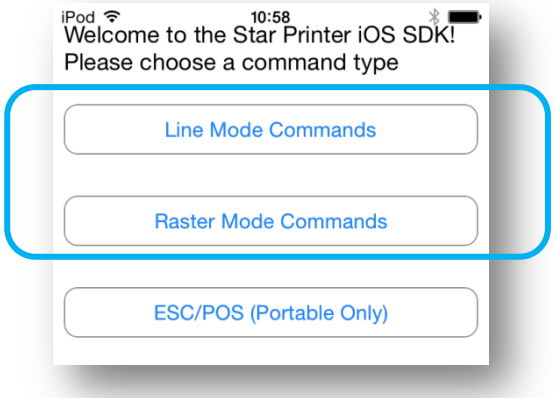
* 初めて searchPrinter メソッドでプリンタ情報を取得した際、portName が @"BLE:"となる場合があります。
この場合、一旦 getPort メソッドでプリンタに接続を行ってください。
これ以降、正しいデバイス名を取得、使用可能になります。

モバイルプリンターの設定

1) “Star Portable Printers / mPOP” をタップします。



2) “Line Mode Commands”または”Raster Mode Commands”から、使用するコマンドの種類をタップしてください。コマンドの種類によってプリンターへのデータ送信方法が変わります。



Line Mode Commands:

プリンターへのデータ送信を 1 行ごとに行います。データは小さな単位でプリンターに送信されるため、開発者が任意の場所のコマンドでカスタマイズを行うのに適しています。

このモードではプリンターにインストールされているデバイスフォントのみ利用できます。

Raster Mode Commands:

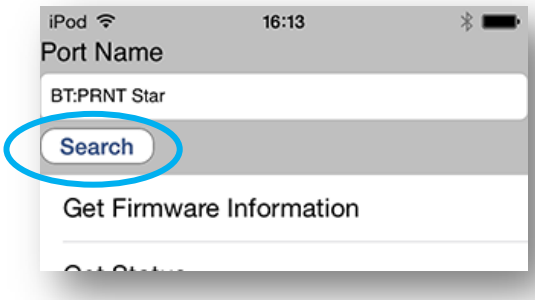
プリンターはすべての情報をグラフィックイメージで受信します。それにより、見栄えの良い TrueType フォントを利用して、高い印字速度での印刷を実現できます。

ラスターコマンドでは、印字データをプリンターに送信する前にレシート全体のグラフィックデータを生成する必要があり、ラインモードコマンドよりも複雑なコーディングが必要になります。

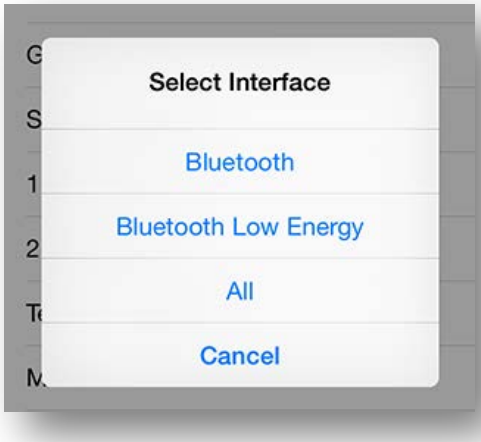
Bluetooth/Bluetooth Low Energy プリンターの設定

・接続するプリンターを検索して設定する

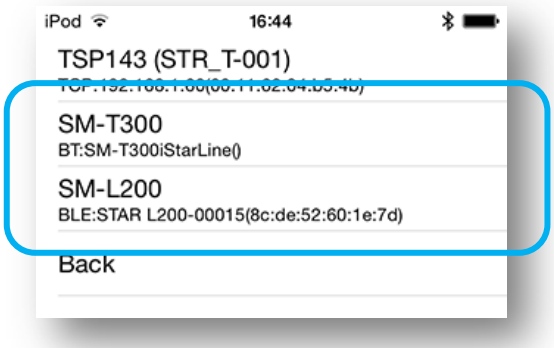
1）ネットワークに接続されているすべての Star の Bluetooth/Bluetooth Low Energy プリンターを検索するために、”Search"をタップします。



2）接続するプリンターのインターフェイスをタップします。
Bluetooth の場合、ペアリングされた接続可能な Bluetooth プリンターのポート名を表示します。Bluetooth Low Energy の場合、周辺のプリンターを検索して表示します。



3）検索されたプリンター一覧から、接続するプリンターの名前をタップしてください。



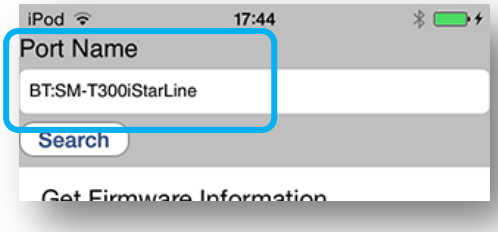
・接続するプリンターを手動で設定する

1）「PortName」に以下のように手動で入力します。

◆Bluetooth

BT: <iOS ポート名>

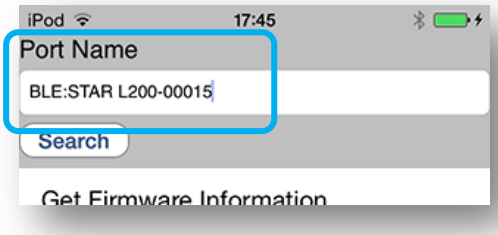
”BT:”の後に iOS ポート名を入力します。



◆Bluetooth Low Energy

BLE: <デバイス名>

”BLE:”の後にデバイス名または MAC アドレスを入力します。



iOS SDK 概要

SDK の主要コンポーネントについて簡単に説明します。

全ての機能が IOS_SDK project と IOS_SDK target に位置しています。

IOS_SDKViewController.m ファイルからプログラムを実行してください。このソース・コードがモバイル プリンターの起点となります。

他のソース・ファイルをクリックすることにより、特定の機能がどのように働くか確かめてください。例えば、“code128.m”は GUI の中の 1D barcode Code128 に相当します。

全ての機能が両方のプリンター・タイプに利用可能ではありませんので、ご注意ください。各 SDK マニュアルの最初のページには、どの機能がサポートされているか記載されています。

また、“Mini”を含むソース・ファイルは、モバイルプリンターを ESC/POS モードで使用する場合のサンプルコードです。StarBitmap.m は、両方のプリンター・タイプに適用されます。

StarIO Framework

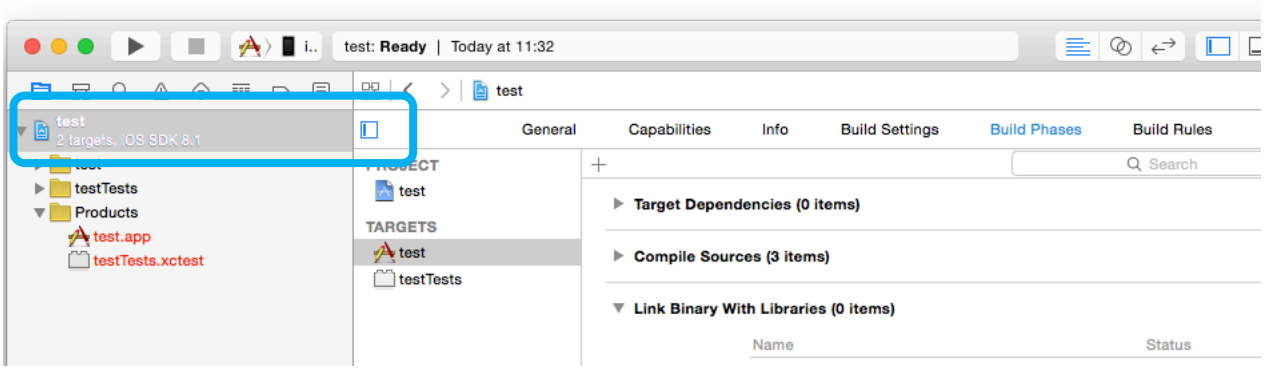
Star iOS SDK プロジェクトには、既に StarIO framework は含まれています。(この SDK をテストする場合、そのまま使用できます)

ただし、新規のアプリケーションを作成する場合、StarIO メソッドを使用するためにプロジェクトに必要な framework を追加する必要があります。

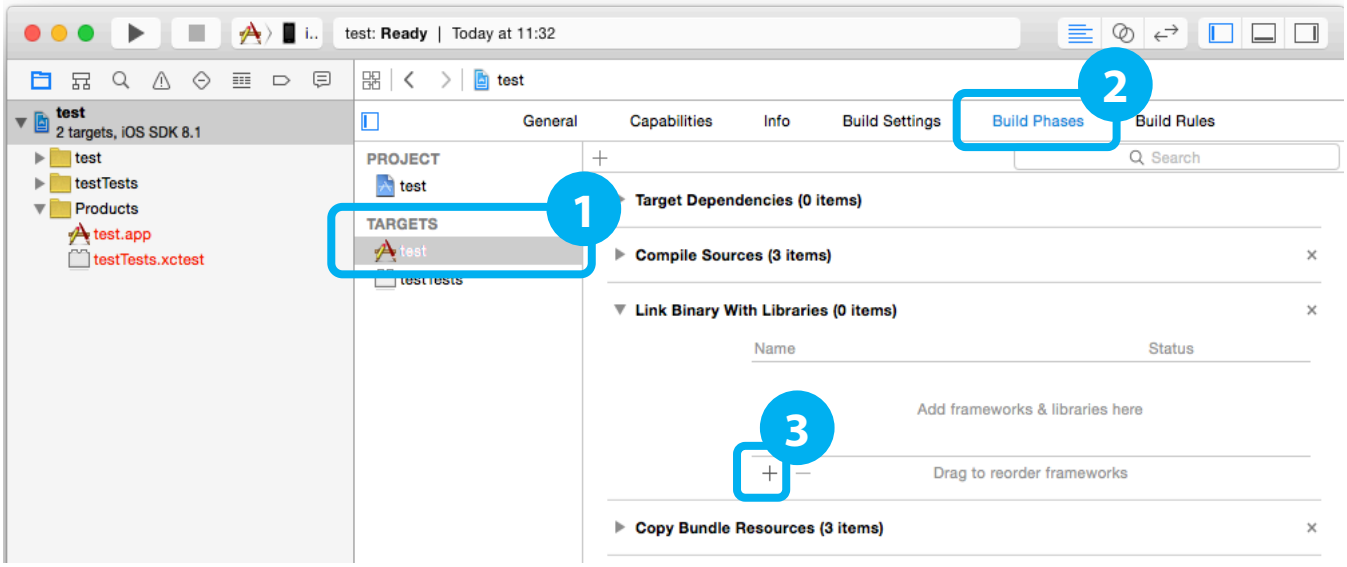
●新規のアプリケーションを作成するには

1. プロジェクトに StarIO.framework を追加する

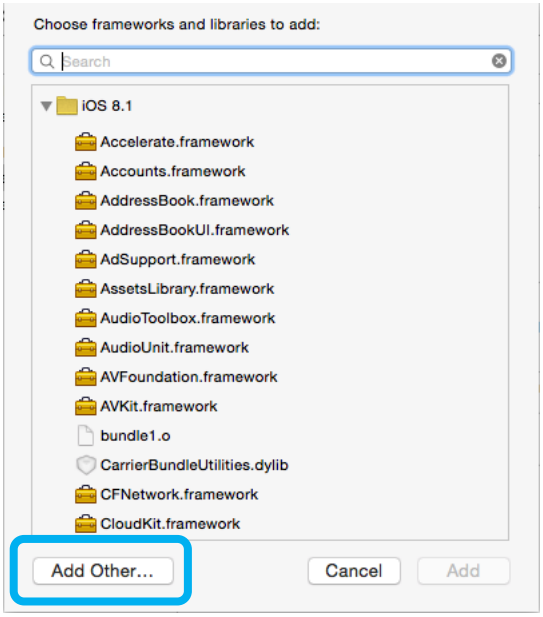
1. 新規に作成したプロジェクトをクリックします。



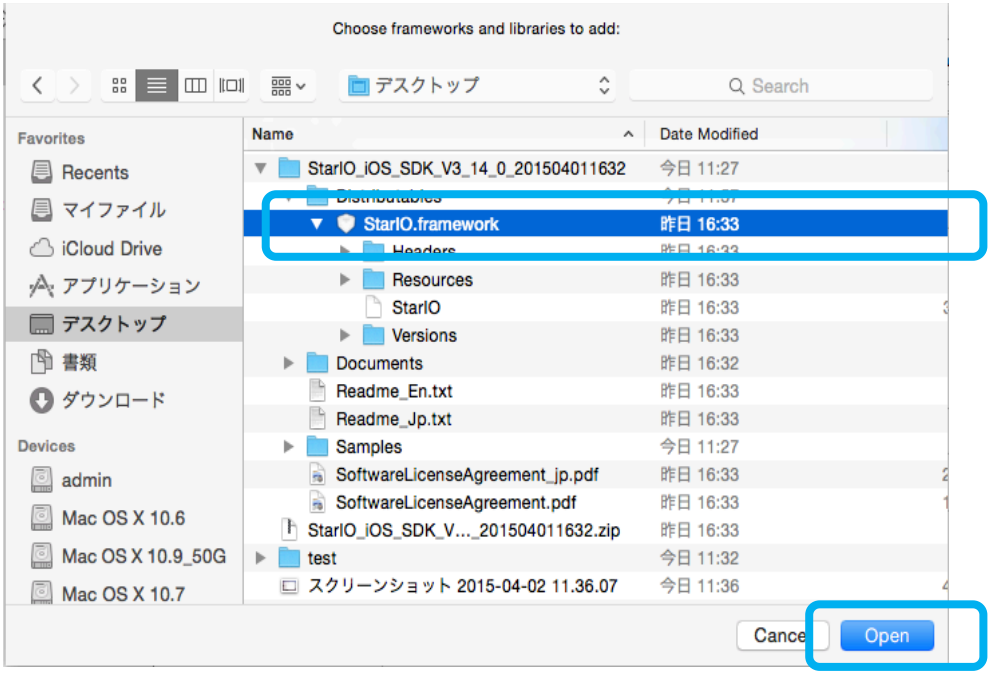
2. ターゲット → “Build Phases” → Link Binary With Libraries の“+” をクリックします。



3. “Add Other…”ボタンをクリックします。



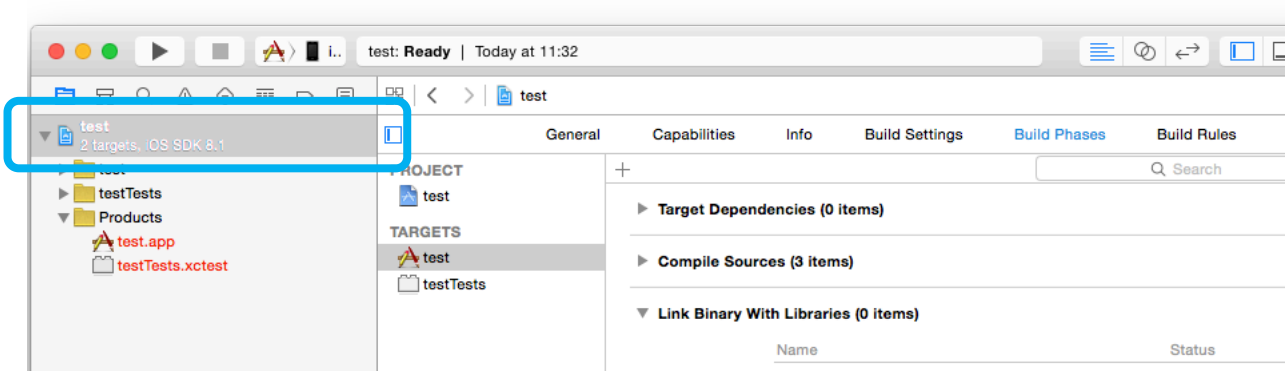
4. Star iOS SDK を解凍した場所の StarIO.framework フォルダを参照し、“Open”をクリックします。



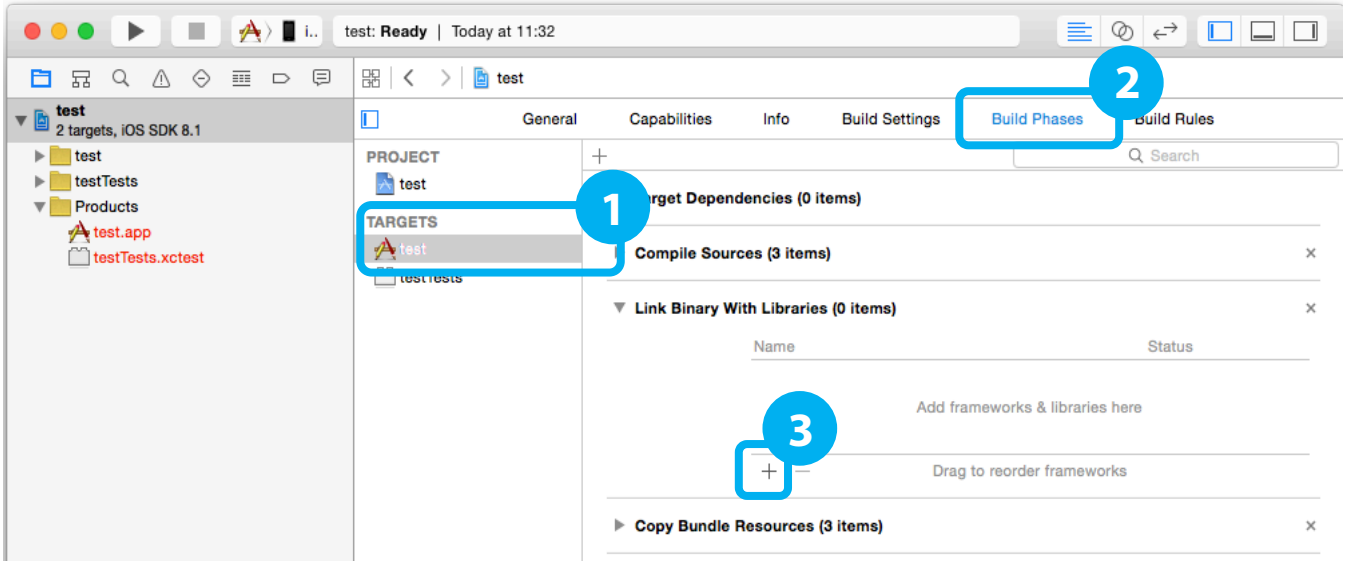
5. StarIO framework はプロジェクトに追加されます。

2. その他の framework を追加する

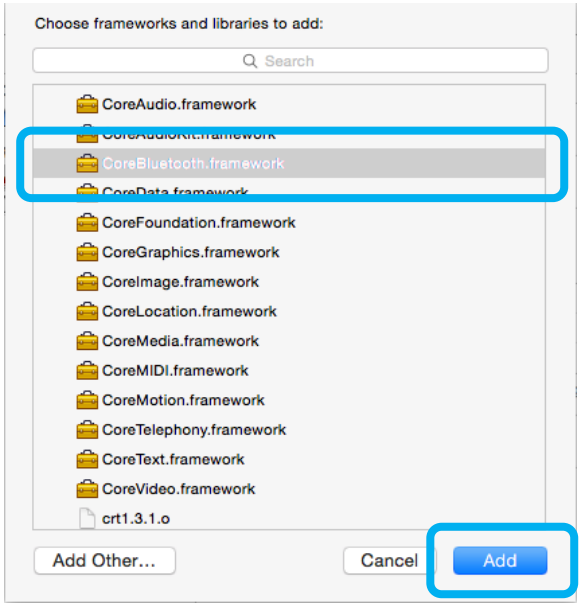
1. 新規に作成したプロジェクトをクリックします。

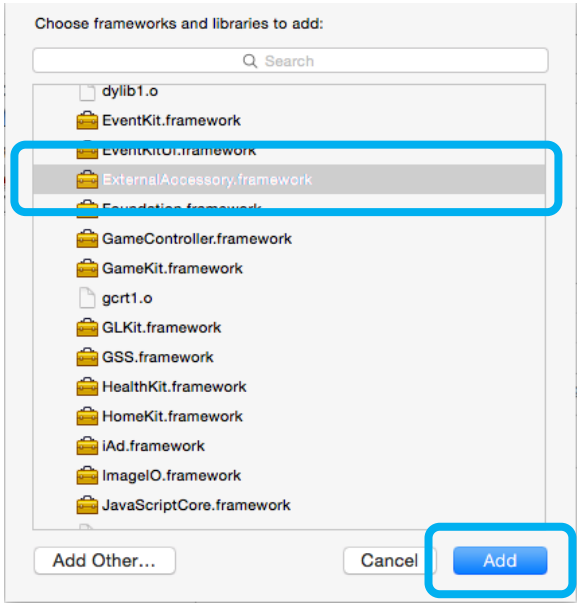


2. ターゲット → “Build Phases” → Link Binary With Libraries の“+” をクリックします。

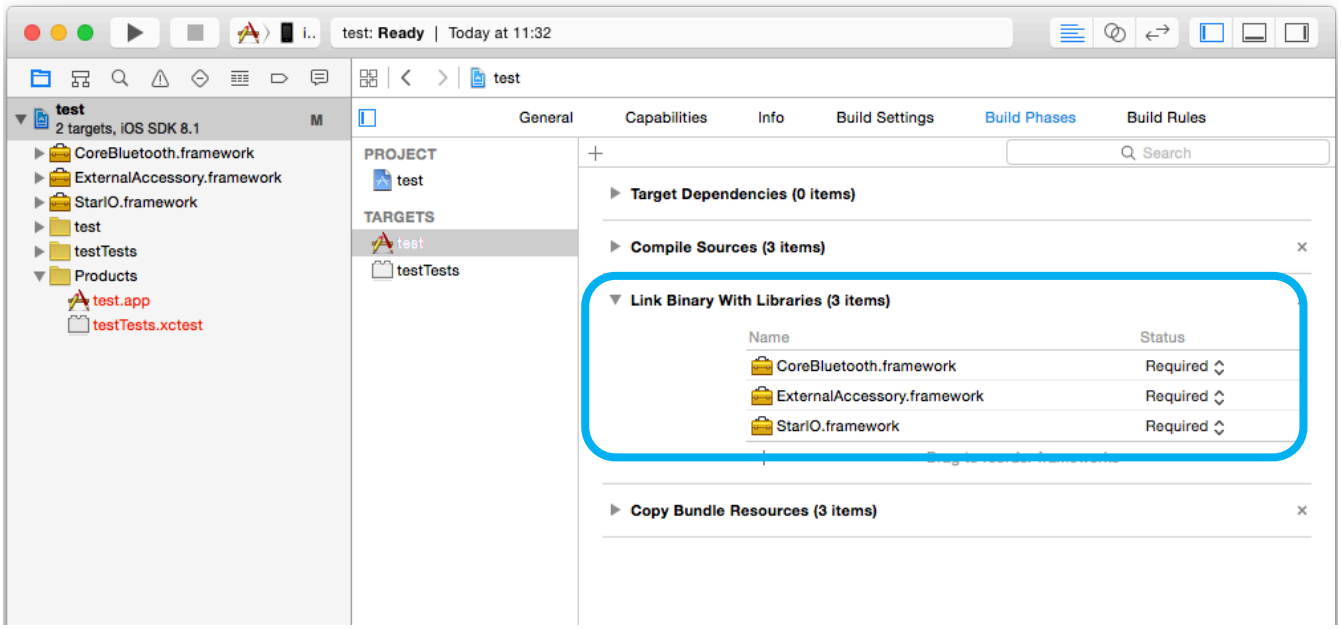


3. External Accessory framework, Core Bluetooth framework をそれぞれ追加します。framework を選択し、“Add”をクリックします。





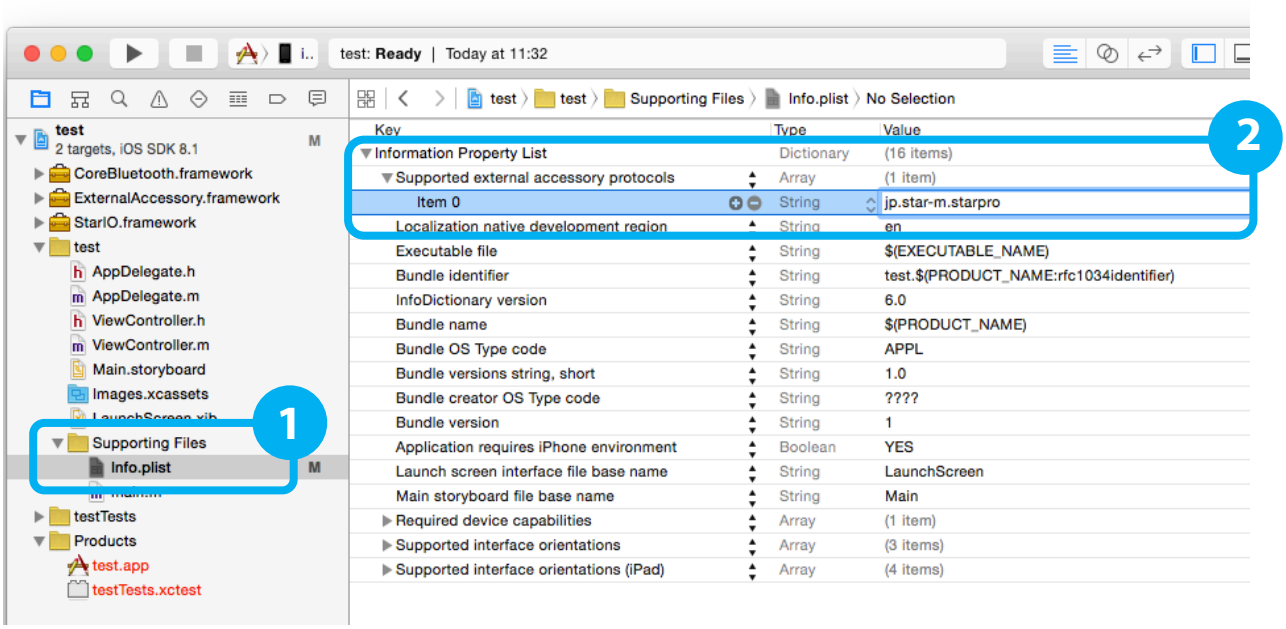
4. 必要な framework が追加されているか確認してください。



3.Information Property List へ項目を設定する（ Bluetooth を使用する場合）

※Bluetooth プリンターを使用しない場合は、この設定を行わないでください。

1. Information Property List(デフォルトでは”Info.plist”)を選択します。
2. Information Property List に項目を追加して、[Key]に“Supported external accessory protocols”を設定（入力）、項目名左側の▽をクリックして表示される”Item 0”の[Value]に “jp.star-m.starpro” を設定します。



3. Information Property List の設定は完了です。

●プロジェクトの StarIO.framework をバージョンアップするには

1. 現在使用している StarIO.framework を削除します。

2. 同じ場所に新しい StarIO.framework をコピーします。

3.Xcode プロジェクトをクリーンします。
Xcode プロジェクトを開き、メニューから[Build]-[Clean]を選択します。

4.Xcode プロジェクトをビルドします。

現在使用している StarIO.framework を削除せずに、参照先を新しい StarIO.framework に変更する場合には、必ず Xcode プロジェクトの “framework search path” の設定を確認してください。

“framework search path” の先頭に古い StarIO.framework のパスが残っていると、引き続き今までの StarIO.framework が使用されてしまいます。

StarIO メソッド概要

SMPort クラス:

●プロパティ

portName	プリンターの通信ポートを取得します。
portSettings	ポート設定を取得します。
timeoutMillis	内部制御と API のタイムアウト時間を取得、設定します。
endCheckedBlockTimeoutMillis	endCheckedBlock メソッドのタイムアウト時間を取得、設定します。

- (NSString *)**portName**

プリンターの通信ポートを取得します。

- (NSString *)**portSettings**

ポート設定を取得します。

- (u_int32_t)**timeoutMillis**

内部制御と API のタイムアウト時間を取得、設定します。（単位：ミリ秒）

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

endCheckedBlock メソッドのタイムアウト時間を取得、設定します。[単位: ミリ秒]
印刷に時間がかかる場合、この値を大きくする事で endCheckedBlock メソッドの印刷完了待ち時間を伸ばす事ができます。
初期値は、getPort メソッドで指定したタイムアウト時間となります。

指定した時間が 10 秒未満の場合、タイムアウトは 10 秒になります。

●メソッド

getPort

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

getPort は、プリンターポートのオープンに使用されます。

引数:

portName　　– プリンターへの通信ポートを指定

例）*@"BT:PRNT Star" (Bluetooth の場合)*
@"BT:00:11:62:1b:4d:f4" (Bluetooth で MAC アドレスを指定する場合)
@"BLE:STAR L200-00001" (Bluetooth Low Energy の場合)
@"BLE:8C:DE:52:60:30:C6" (Bluetooth Low Energy で MAC アドレスを指定する場合)

Note: Bluetooth の MAC アドレス指定は iOS6 のみ使用可能で、iOS7 以降では使用できません。

portSettings　　- @"portable"を指定

timeoutMillis　　– 内部制御と API の通信タイムアウト値を指定

Note: timeoutMillis は API が制限された時間内で完了することを保証しますが、正確なタイムアウトの長さを保証するものではありません。

戻り値:

SMPort クラスインスタンス。ポートオープンに失敗した場合、既にオープン済みであった場合は、nil が返されます。

getPort を実行した後は、必ず releasePort してから次の getPort を行ってください。releasePort をせずに次の getPort を行うと、nil が返されます。

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

SMPort *port = nil;
NSString *portName = @"BT:PRNT Star";
NSString *portSettings = @"portable";
@try
{
    port = [SMPort getPort:portName :portSettings   :10000];
}
@catch (PortException)
{
    //There was an error opening the port
}
@finally
{
    [SMPort releasePort:port];
}
```

getPort を使用する場合は、常に try catch を使用してください。上記の例のような try catch を使用しなければ、通信エラーでポートオープンができなかった場合、プログラムはクラッシュする可能性があります。

Bluetooth インターフェイスの場合、プリンターとの通信を 30 秒以上行わない場合は一度ポートをクローズしてください。
1 トランザクションごとにポートオープン・クローズすることを推奨します。

注意事項（SM-L シリーズのみ）

iOS 端末で Bluetooth Low Energy にて通信する場合、プリンターとの接続に時間がかかることがあります。 接続に失敗した場合は、接続が成功するまでリトライを行ってください。
また、接続時間を短縮したい場合は、プリンターとの接続が常に維持されるようアプリケーションを設計してください。

* ただし、接続中のプリンターは他のアプリケーション、および他の端末から検出することができなくなります。

searchPrinter

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

searchPrinter は、LAN 上のプリンターとペアリングされた Bluetooth プリンターを検索し、検索結果を NSArray 型で返します。
戻り値の NSArray には、PortInfo クラスのインスタンスが含まれます。
戻り値の PortInfo クラスは、ポート名、プリンターの MAC アドレス、プリンターモデル名を持ち、それぞれ portName, macAddress, modelName プロパティにて取得することができます。

portName は、getPort の引数として使用することができます。

引数 target を指定すると、Bluetooth もしくは Bluetooth Low Energy プリンターのいずれかのみを検索することができます。

引数:

target　– @”BT:”を指定した場合：　Bluetooth プリンターを検索
　　@”BLE:”を指定した場合：　Bluetooth Low Energy プリンターを検索

本 API はデバイスを確実に検出する事を保証するものではありません。
Bluetooth の MAC アドレス取得は iOS6 のみ使用可能で、iOS7 以降では使用できません。

初めて searchPrinter メソッドでプリンタ情報を取得した際、portName が@"BLE:"となる場合があります。この場合、一旦 getPort メソッドでプリンタに接続を行ってください。これ以降、正しいデバイス名を取得、使用可能になります。

```
//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];
```

上記の例では、Bluetooth プリンターと Bluetooth Low Energy プリンターの両方を検索して一覧を取得し、その内容をログに出力します。

readPort

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

このメソッドは、デバイスからデータを読み込みます。プリンターから Raw byte を読み取る必要のある場合のみ、ご使用ください。

Raw Status の取得にこのメソッドを使用しないでください。
Status の取得は、getParsedStatus::メソッドを使用してください。

引数:

readbuffer	– データが読み込まれるバイト配列のバッファ
offset	- ReadBuffer にデータを書き込み始める場所を指定
size	– 読み込むバイト数の合計

戻り値:

実際に読み込まれたバイト数。データが全て読み取れなかった時でも、この関数は成功します。アプリケーションは、期待されるデータが全て読み取れるまで、この関数を複数回、呼び出す必要があります。または、しきい値に達するまで再試行をするようにします。

例外:

PortException – 通信エラーが発生したとき

releasePort

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

このメソッドは、指定されたポートへの接続をクローズします。

引数:

port – 以前に初期化されたポートを表すポートタイプ

getPort を実行した後は、必ず releasePort してから次の getPort を行ってください。releasePort をせずに次の getPort を行うと、nil が返されます。

writePort

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

このメソッドは、デバイスにデータを書き込みます。コマンドや印刷データの送信に使用します。

印字終了の確認を行うため、このメソッドの前後で beginCheckedBlock/endCheckedBlock を使用してください。

サンプルコードはこちらをご参照ください。

※安全なプログラミングをするために try catch を使用してください。

SDK の“PrintTextWithPortName”には、プリンターにデータが送信されたことを確認する方法を示すコードが記述されています。

引数:

writeBuffer	– 出力データを格納する Byte 配列のバッファ
offset	– writeBuffer からデータを読み込み始める場所を指定
size	– 書き込むバイト数の合計

戻り値:

Bluetooth :
実際に書き込まれたバイト数。データが全て書き込めなかった時でも、この関数は成功します。アプリケーションは、期待されるデータが全て書き込まれるまで、この関数を複数回、呼び出す必要があります。または、しきい値に達するまで再試行をするようにします。

Bluetooth Low Energy :
成功時には、送信データのサイズが返ります。失敗時は 0 が返ります。

例外:

PortException – 通信エラーが発生したとき

getParsedStatus

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

このメソッドは、StarIO で詳細なステータスをプリンターから取得します。

戻り値:

StarPrinterStatus　構造体は、現在のデバイスのステータスを保持します。

例外:

PortException　– 通信エラーが発生したとき

このメソッドは StarPrinterStatus と呼ばれる StarIO の構造体を使用します。
この構造体は、ブーリアン型とバイナリーの両方の形式でプリンターステータスを保持します。

下記を行うことにより、プロジェクトで StarPrinterStatus オブジェクトを作成してください。

```
StarPrinterStatus_2 printerStatus;
[port getParsedStatus: &printerStatus : 2];

if (printerStatus.offline == SM_TRUE)
{
    if (printerStatus.coverOpen == SM_TRUE) {
        //There was a cover open 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.
}
```

StarPrinterStatus 構造体　ステータスリスト

メンバ名	説明	型	詳細
blackMarkError	ブラックマークエラー	SM_BOOLEAN	ブラックマークエラー(ブラックマーク設定時に非ブラックマーク用紙を使って印刷した場合等に発生)の時に SM_TRUE となる。通常時は SM_FALSE。
compulsionSwitch	コンパルジョン SW	SM_BOOLEAN	ドロワのコンパルジョン SW が押されていると SM_TRUE となる。通常は SM_FALSE。
coverOpen	カバーの状態	SM_BOOLEAN	カバーが開いている場合に SM_TRUE となる。閉じている場合は SM_FALSE。
cutterError	オートカッターエラー	SM_BOOLEAN	カッターエラー発生時に SM_TRUE となる。通常は SM_FALSE。
etbAvailable	ETB 使用可否	SM_BOOLEAN	ETB が使用可能な場合に SM_TRUE となる。使用できない場合は SM_FALSE。
etbCounter	ETB カウンタ	UCHAR	現在の ETB カウンタの値。
headThermistorError	ヘッドサーミスタエラー	SM_BOOLEAN	ヘッドサーミスタ異常値検出時に SM_TRUE となる。通常は SM_FALSE。
offline	ON-LINE/OFF-LINE 状態	SM_BOOLEAN	オフラインの場合に SM_TRUE となる。 オンライン時は SM_FALSE。
overTemp	印字ヘッド高温による停止中	SM_BOOLEAN	ヘッドが高温になり印刷停止している状態で SM_TRUE となる。通常は SM_FALSE。
presenterPaperJamError	プレゼンター紙ジャムエラー	SM_BOOLEAN	プレゼンター装着時、プレゼンターで用紙ジャムが発生すると SM_TRUE となる。通常は SM_FALSE。
presenterState	プレゼンタ用紙位置	UCHAR	以下のいずれかの数値。 0: プレゼンタ内部に用紙がない 1: 用紙を給紙した状態(ループ状態) 3: 用紙を排出した状態 6: 用紙回収状態 7: 用紙が引き抜かれた状態
raw	ステータスのバイト列	UCHAR[63]	ステータスのバイト列 (例: HEX 23 86 00 00 00 00 00 00)
rawLength	raw の長さ	CHAR	raw の長さ
receiptPaperEmpty	用紙エンド	SM_BOOLEAN	用紙切れの場合は SM_TRUE となる。 通常は SM_FALSE。
receiptPaperNearEmptyInner	用紙ニアエンド(内側)	SM_BOOLEAN	用紙ニアエンド状態の時に SM_TRUE となる。通常は SM_FALSE。
receiveBufferOverflow	受信バッファオーバーフロー	SM_BOOLEAN	受信バッファフルの時に SM_TRUE となる。通常は SM_FALSE。
unrecoverableError	復帰不可能エラー	SM_BOOLEAN	復帰不可能エラー(ヘッドサーミスタエラー、オートカッターエラー、電源電圧エラー等)が発生した場合に SM_TRUE となる。通常は SM_FALSE。
voltageError	電源電圧エラー	SM_BOOLEAN	電源電圧異常値検出時に SM_TRUE となる。通常は SM_FALSE。

StarPrinterStatus 構造体　機種別対応リスト

メンバ名	SM-L200	SM-S210i (JP モデルのみ)	SM-S220i (欧米モデルのみ)	SM-S230i (欧米モデルのみ)	SM-T300i	SM-T400i	mPOP
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;
```

このメソッドは、endCheckedBlock メソッドとセットで使用して印字終了の確認を行います。印刷データ送信の直前に beginCheckedBlock を実行します。

引数:

starPrinterStatus	– StarPrinterStatus 構造体へのポインタ （StarPrinterStatus,StarPrinterStatus_1,StarPrinterStatus_ 2 の指定が可能だが、通常は StarPrinterStatus_2 を指定）
level	– StarPrinterStatus 構造体のレベル （0,1,2 の指定が可能だが、通常は 2 を指定）

サンプルコードはこちらをご参照ください。

endCheckedBlock

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

このメソッドは、beginCheckedBlock メソッドとセットで使用します。

プリンタの状態を監視し、送信した印刷データの印刷が完了すると制御を返します。印刷データ以外を送信した場合は、そのコマンドがプリンタに処理されると制御を返します。

タイムアウト時間(*1) 内に印刷が完了しなかった場合や、印刷中にプリンタエラーが発生した場合は、例外 PortException をスローします。

(*1) タイムアウト時間は、endCheckedBlockTimeoutMillis プロパティの値が使用されます。
初期値は getPort で指定したタイムアウト時間となります。
endCheckedBlockTimeoutMillis の値は、印刷時間より長くなるよう調整してください。
また、10 秒未満の値が設定された場合にはタイムアウトは 10 秒になります。

引数:

starPrinterStatus	– StarPrinterStatus 構造体へのポインタ （StarPrinterStatus,StarPrinterStatus_1,StarPrinterStatus_ 2 の指定が可能だが、通常は StarPrinterStatus_2 を指定）
level	– StarPrinterStatus 構造体のレベル （0,1,2 の指定が可能だが、通常は 2 を指定）

成功時:

StarPrinterStatus のステータスを最新のものに更新して終了します。

例外:

PortException – 通信エラー*が発生したとき

*例） -コマンド送信自体の失敗（オフライン等）
-タイムアウト時間内にプリンタからの終了の応答がない

```
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:" :@"" :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];
}
```

getFirmwareInformation

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

このメソッドは、プリンターからファームウェア情報を取得します。

戻り値:

モデル名とファームウェアバージョンを NSDictionary 型で返します。
Key に@modelName を設定することで戻り値からモデル名を取得します。
Key に@firmwareVersion を設定することで戻り値からファームウェアバージョンを取得します。

例外:

PortException – 取得に失敗したとき

Note:

・取得に失敗した場合、空文字を返します。

StarIOVersion

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

このメソッドは、StarIO のバージョンを取得します。

戻り値:

StarIO バージョン

SMBluetoothManager クラス:

Bluetooth および Bluetooth Low Energy インターフェイスの各種設定を行うためのクラスです。
SMPort クラスと同時に使用しないでください。

●プロパティ

portName	接続先デバイスの portName を取得します。
deviceType	接続先デバイスの種類を取得します。
opened	ポートがオープンされているかを示します。
deviceName	現在の Bluetooth デバイス名を取得、設定します。
iOSPortName	StarIO で使用するポート名を取得、設定します。
autoConnect	自動接続機能の有効/無効を取得、設定します。
security	Bluetooth のセキュリティ（SSP もしくは PIN コードモード）を取得、設定します。
pinCode	Bluetooth ペアリング時に使用する PIN コードを設定します。

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

SMBluetoothManeger のインスタンスを作成します。

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

接続先デバイスの種類を取得します。

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

ポートがオープンされているかを示します。
open メソッドが成功すると YES になります。
その後 close メソッドを呼び出すと NO になります。

@property(nonatomic, retain) NSString ***deviceName**

現在の Bluetooth デバイス名を取得、設定します。
この名前は、Bluetooth ではペアリング時に表示されます。
Bluetooth Low Energy では、通信時の接続ポート名として使用されます。
loadSetting メソッドを呼び出した際に現在の設定値が読み込まれます。
設定するには、本プロパティを変更後 apply メソッドを実行します。

設定可能文字数：1~16
使用可能文字列：　0-9, a-z, A-Z,
; : ! ? # $ % & , . @ _ - = スペース / * + ~ ^ [{ ( ] }) | \

Bluetooth Low Energy の場合、変更した Bluetooth 名称は、デバイスの電源再投入・再接続を行った後に有効になります。

@property(nonatomic, retain) NSString ***iOSPortName**

StarIO で Bluetooth 通信時に使用するポート名を取得、設定します。
Bluetooth Low Energy では使用されません。
loadSetting メソッドを呼び出した際に現在の設定値が読み込まれます。
設定するには、本プロパティを変更後 apply メソッドを実行します。

設定可能文字数：1~16
使用可能文字列：
0-9 a-z A-Z ; : ! ? # $ % & , . @ _ - = スペース / * + ~ ^ [{ ( ] }) | \

@property(nonatomic, assign) BOOL **autoConnect**

自動接続機能の有効/無効を取得、設定します。
本機能は Bluetooth のみ対応しています。
loadSetting メソッドを呼び出した際に現在の設定値が読み込まれます。
設定するには、本プロパティを変更後 apply メソッドを実行します。

security プロパティが PIN コード設定の場合は、この値に NO を設定してください。

@property(nonatomic, assign) SMBluetoothSecurity **security**

Bluetooth のセキュリティ（No Security もしくは PIN コード）を取得、設定します。
本機能は Bluetooth のみ対応しています。
loadSetting メソッドを呼び出した際に現在の設定値が読み込まれます。
設定するには、本プロパティを変更後 apply メソッドを実行します。

PIN コード設定を使用する場合は、autoConnect プロパティに NO を設定してください。

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

Bluetooth インターフェイスの PIN コードを設定します。
本機能は Bluetooth のみ対応しています。
現在の設定値を取得することはできません。
現在の PIN コードから値を変更しない場合は nil を指定します。

設定可能文字数：4~16 [*1]

[*1] SM-L200 は、4 文字固定です。

使用可能文字列： 0-9, a-z, A-Z [*2]

[*2] SM-L200 は、0-9 のみ使用可能です。

●メソッド

initWithPortName : deviceType

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

SMBluetoothManager のインスタンスを作成します。

引数:

portName　　– 接続するデバイスのポート名
　　例）”BT:PRNT Star”
deviceType　　– 接続するデバイスの種類
　　SMDeviceTypePortablePrinter（モバイルプリンター指定）

戻り値:

成功時は SMBluetoothManager のインスタンスを返します。
失敗時は nil を返します。

open

```
- (BOOL) open
```

Bluetooth もしくは Bluetooth Low Energy プリンターとの接続を開きます。
open メソッドの実行後は、必ず loadSetting メソッドで現在の設定を取得してください。

戻り値:

成功した場合は YES を、失敗した場合は NO を返します。

loadSetting

```
- (BOOL) loadSetting:
```

Bluetooth インターフェイスカードの設定情報を取得します。

戻り値:

成功した場合は YES を、失敗した場合は NO を返します。

apply

```
- (BOOL) apply
```

deiceName, iOSPortName, autoConnect, security, pinCode プロパティの値をデバイスに適用します。

戻り値:

成功した場合は YES を、失敗した場合は NO を返します。

apply メソッドで適用した値は、デバイスの電源再投入・再ペアリングを行った後に有効になります。

close

```
- (void) close
```

Bluetooth/Bluetooth Low Energy プリンターとの接続を閉じます。

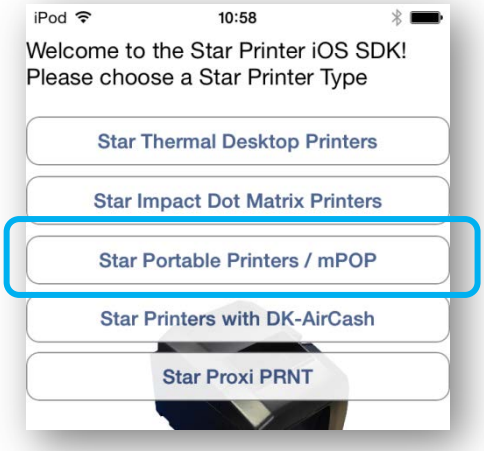
StarIO iOS SDK 機能

SDK 機能と StarIO のプリンターコマンドについての概要
これらのコマンドは、全て StarPRNT コマンド仕様書に記載されています。

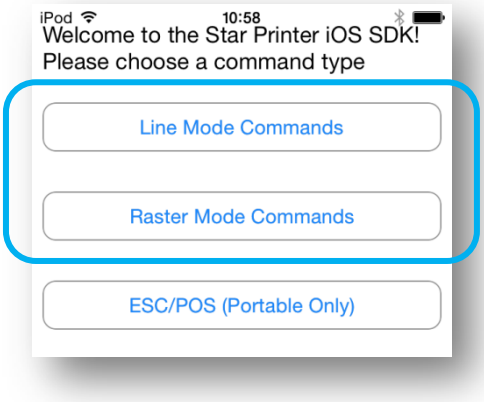
特定のコマンドに関する詳細な情報が必要な場合は、StarPRNT コマンド仕様書をダウンロードし参照してください。

プリンターとコマンド種類の選択

1) “Star Portable Printers / mPOP”をタップします。



2) “Line Mode Commands”または”Raster Mode Commands” をタップします。
コマンドの種類によってプリンターへのデータ送信方法が変わります。
“Line Mode”、”Raster Mode”の詳細についてはこちらを参照してください。



コマンド別サンプル機能対応リスト

Line Mode Command Samples Include:

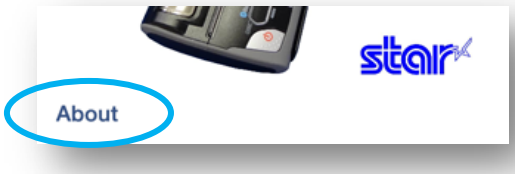
- Port Discovery
- Get Firmware Information
- Get Status
- Sample Receipt
- 1D Barcodes
- 2D Barcodes
- Text Formatting
- Bluetooth Setting

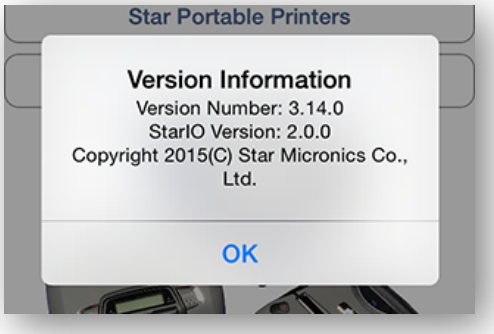
Raster Mode Command Samples Include:

- Port Discovery
- Get Firmware Information
- Get Status
- Sample Receipt
- Raster Graphics Text Printing
- ImageFile Printing
- AllReceipts
- Bluetooth Setting

Get StarIO Version

◆Line　◆Raster

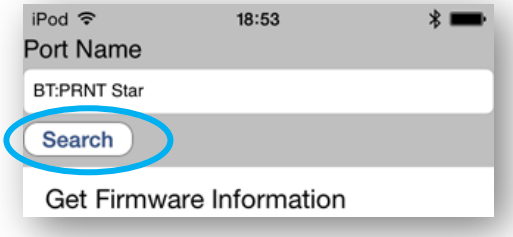


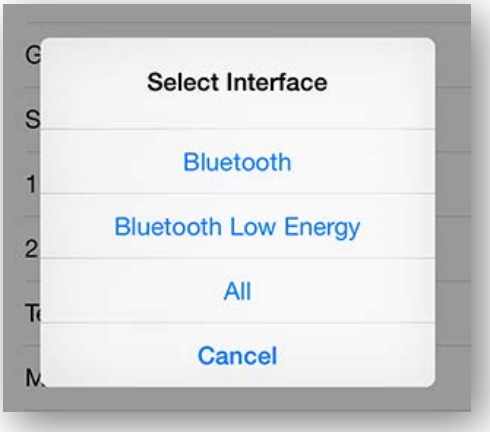


About をタップすると、StarIO のバージョンを表示します。

Port Discovery

◆Line　◆Raster

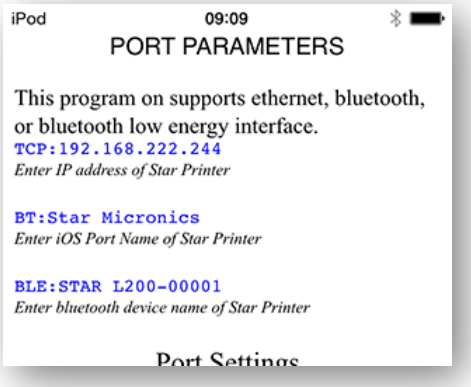




ネットワークに接続されている Star のプリンターを自動的に検索します。検索結果より、接続したいプリンターを選択してください。詳細はこちらをご参照ください。

Help

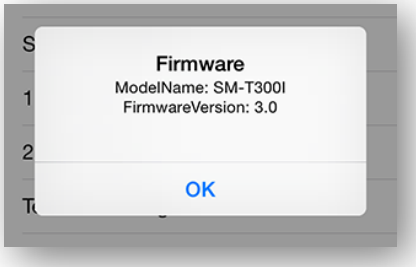
◆Line　　◆Raster



Help は、StarIO ポートの設定に関する情報を表示します。

Get Firmware Information

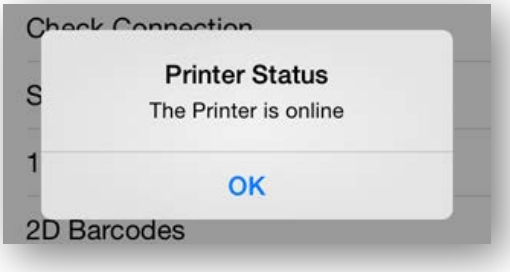
◆Line　　◆Raster



Port Name に設定されたプリンターのファームウェア情報を表示します。

Get Status

◆Line　　◆Raster



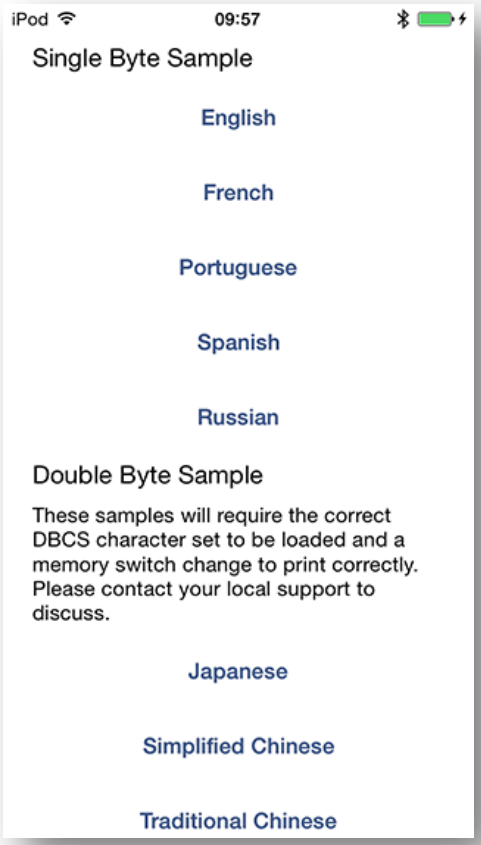
StarPrinterStatus

SM_BOOLEAN getParsedStatus::	ステータスの戻り値はここを参照してください
offline	SM_FALSE = プリンターオンライン SM_TRUE = プリンターオフライン
other	ステータスの戻り値はここを参照してください

Sample Receipt

◆Line　◆Raster

※Raster の場合の画面遷移



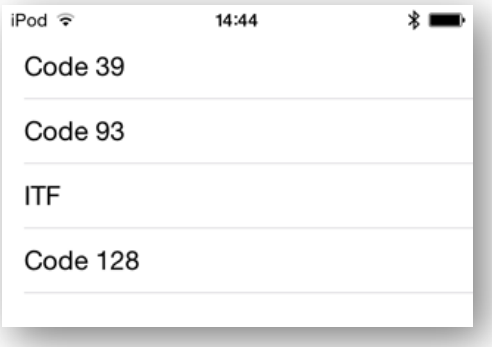
選択したコマンド種類で、選択した言語でのサンプルレシートの印刷をします。
レシート幅をタップして印刷を開始してください。
ソースコードには、レシートをカスタマイズする方法が詳しく説明されています。

Line Mode では、SM-S210i, SM-S220i, SM-S230i, SM-T300i, SM-T400i でロシア語(Russian)、簡体字(Simplified Chinese)の印刷に対応しておりません。

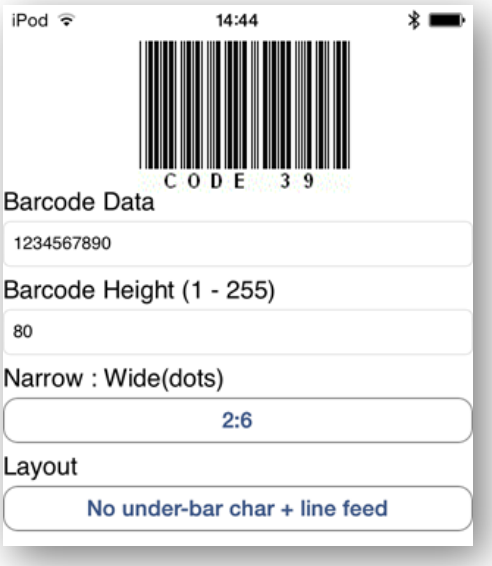
Line Mode では、SM-L200 での簡体字(Simplified Chinese)の印刷にはファームウェアバージョン 1.1 から対応いたします。

1D Barcodes

◆Line



<<CODE39>>



ESC b n1 n2 n3 n4 d1 ... dk RS

n1 = バーコード種選択

0 = UPC-E *1　1 = UPC-A *1　2 = JAN/EAN8 *1　3 = JAN/EAN13 *1

4 = Code39　5 = ITF　6 = Code128　7 = Code93　8 = NW-7 *1

n2 = バー下文字選択、及び、改行付加選択

1 = バー下文字を付加しない。バーコード印字後、改行を実行する。

2 = バー下文字を付加する。バーコード印字後、改行を実行する。

3 = バー下文字を付加しない。バーコード印字後、改行を実行しない。

4 = バー下文字を付加する。バーコード印字後、改行を実行しない。

n3 = バーコードモード選択

n4 = バーコード高さ(ドット数)

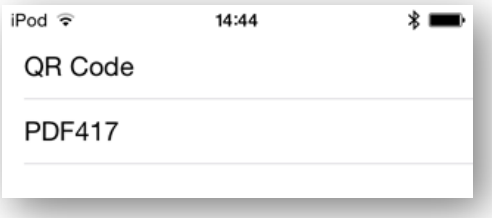
*1: これらのバーコードは、Star モバイル プリンターではサポートされていますが、SDK では実装されていません。

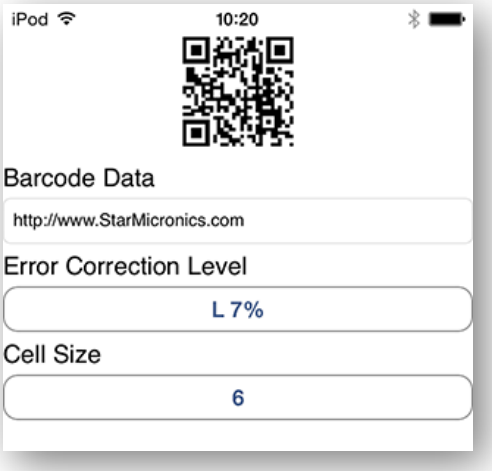
注記：このサンプルアプリケーションは Raster Mode での 1D Barcodes 印刷に対応していません。Raster Mode ではバーコードを画像データとしてプリンターに送信する必要があります。

2D Barcodes

◆Line

＜＜QR Code＞＞





QR Code に関するコマンドは、その機能により以下の 5 つに分類される。

(1) モデルの指定	ESC GS y S 0 n
(2) エラー訂正レベルの指定	ESC GS y S 1 n
(3) セルサイズの指定	ESC GS y S 2 n
(4) QR Code データの指定	ESC GS y D 1 NUL nL nH d1…dk
(5) QR Code の印刷	ESC GS y P

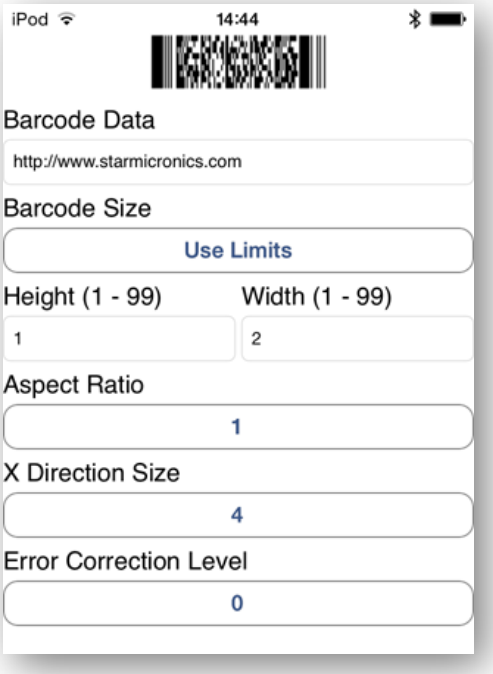
以下の順番でコマンドを送信することにより、QR Code が印刷できます。

モデル指定 + エラー訂正レベル + セルサイズ + QR Code データ + QR Code 印刷

※詳細は、StarPRNT コマンド仕様書を参照してください。

注記：このサンプルアプリケーションは Raster Mode での 2D Barcodes 印刷に対応していません。Raster Mode ではバーコードを画像データとしてプリンターに送信する必要があります。

＜＜PDF417＞＞



PDF417 に関するコマンドは、その機能により以下の 7 つに分類される。

(1) サイズの指定	ESC GS x S 0 n p1 p2
(2) エラー訂正レベルの指定	ESC GS x S 1 n
(3) モジュールサイズ(X 方向)の指定	ESC GS x S 2 n
(4) モジュールサイズ(Y 方向)の指定	ESC GS x S 3 n
(5) PDF417 データの指定	ESC GS x D nL nH d1 d2…dk
(6) PDF417 の印刷	ESC GS x P
(7) PDF417 展開情報取得	ESC GS x I

以下の順番でコマンドを送信することにより、PDF417 が印刷できます。

サイズ + エラー訂正レベル + モジュール(X) + モジュール(Y) + PDF417 データ + PDF417 印刷

※詳細は、StarPRNT コマンド仕様書を参照してください。

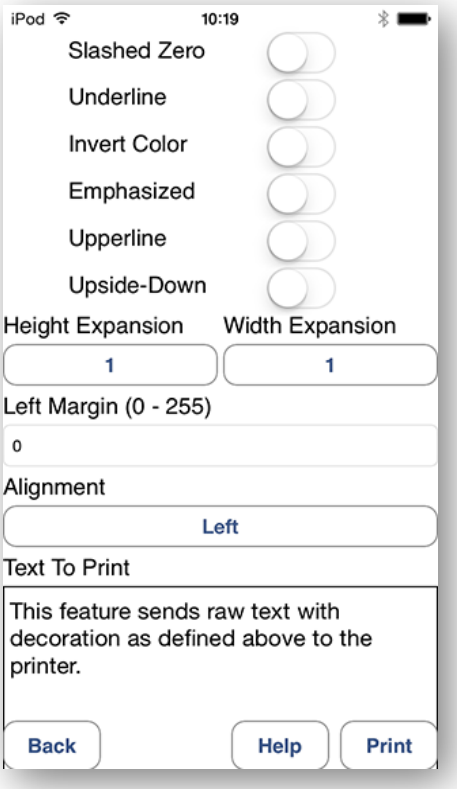
注記：このサンプルアプリケーションは Raster Mode での 2D Barcodes 印刷に対応していません。
Raster Mode ではバーコードを画像データとしてプリンターに送信する必要があります。

SM-L200 はファームウェアバージョン 1.1 から PDF417 に対応します。

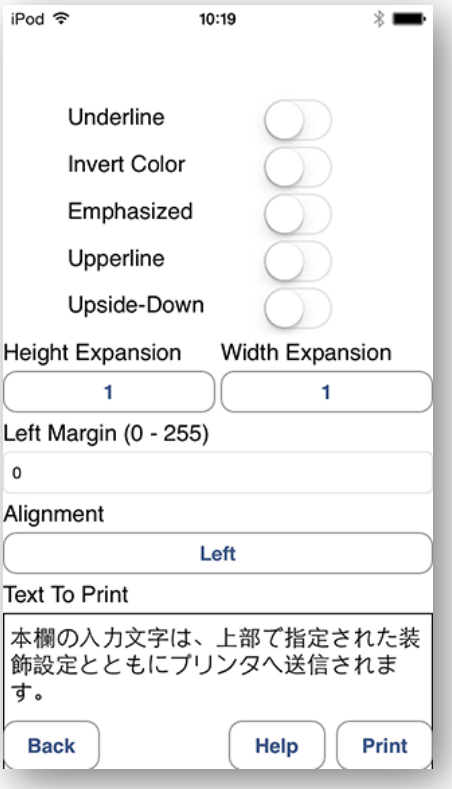
Text Formatting

◆Line

English



Japanese



文字の書式設定

・**Slashed Zero：　　スラッシュゼロ（ASCII のみ）**
　ESC/ n　　１＝設定　　0=解除[デフォルト]

・**Underline：　　アンダーライン**
　ESC – n　　１＝設定　　0=解除[デフォルト]

・**Invert Color：　　白黒反転印字**
　ESC 4 ＝設定　　ESC 5 ＝解除[デフォルト]

・**Emphasized：　　強調印字**
　ESC E ＝設定　　ESC F ＝解除[デフォルト]

・**Upperline：　　アッパーライン**
　ESC _ n　　１＝設定　　0=解除[デフォルト]

・**Upside-Down：　　倒立印字**
　SI ＝設定　　DC2 ＝解除[デフォルト]

・**文字サイズの拡大**
　Height Expansion: 高さ　　ESC h n　　0 ≦ n ≦ 5 (SM-L200 以外)
　　0 ≦ n ≦ 2 (SM-L200)
　Width Expansion: 幅　　ESC W n　　0 ≦ n ≦ 5

・**Left Margin (0 – 255):　　左マージンの設定**
　GS l n　　0 ≦ n ≦ 255

・**Alignment:　　位置揃え**
　ESC GS a 0 = 左揃え[デフォルト]
　ESC GS a 1 = 中央寄せ
　ESC GS a 2 = 右揃え

注記：Raster Mode を使用されている場合は、Raster Graphical Text Printing をご参照ください。

Raster Graphical Text Printing　　◆Raster

◆Raster

Raster Mode では、すべての印刷データをイメージデータに変換してプリンターへ送信します。これにより、文字データおよびロゴ/クーポンの付加された印刷を高速で行うことができます。

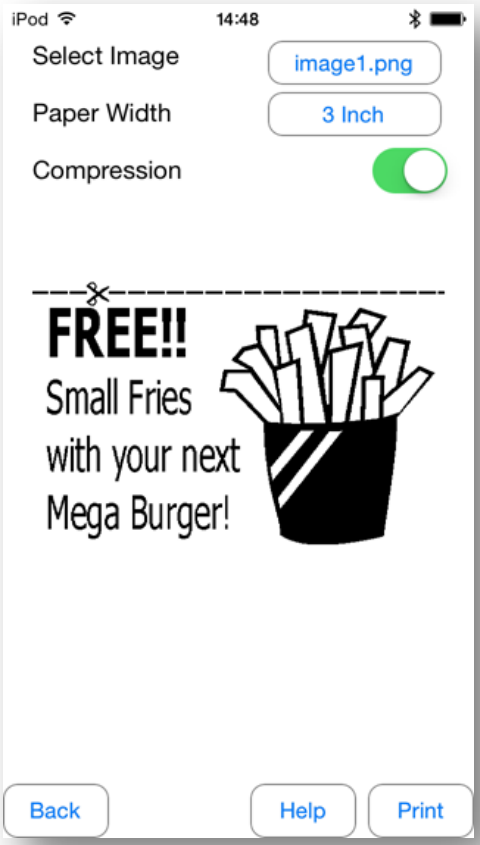
Raster コマンドの詳細については、「StarPRNT コマンド仕様書」をご参照ください。
また、画面右下に表示される[Help]をタップすることにより、Raster コマンドについて確認できます。

注記：
・Line Mode を使用されている場合は、Text Formatting をご参照ください。

Image File Printing

◆Raster

◆Raster



イメージデータの Raster 印刷に使用する画像を Select Image から選択してください。

Raster Mode では、すべての印刷データをイメージデータに変換してプリンターへ送信します。これにより、文字データおよびロゴ/クーポンの付加された印刷を高速で行うことができます。

“Compression”を使用することにより、スループットの向上が望めます。

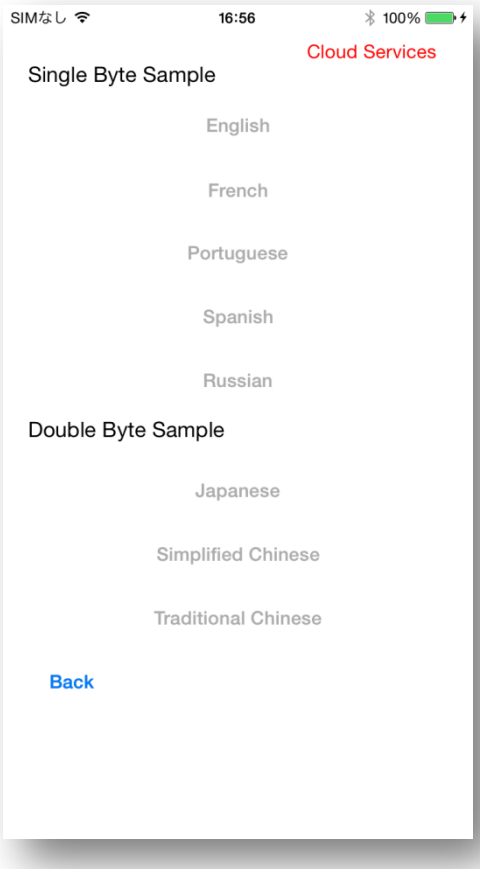
Raster コマンドの詳細については、「StarPRNT コマンド仕様書」をご参照ください。
また、画面右下に表示される[Help]をタップすることにより、Raster コマンドについて確認できます。

注記：このイメージデータは 80mm幅のレシートに合わせて作成されています。
80mm幅以外の用紙を使用されている場合、自動的にリサイズされません。

AllReceipts

◆Raster

◆Raster

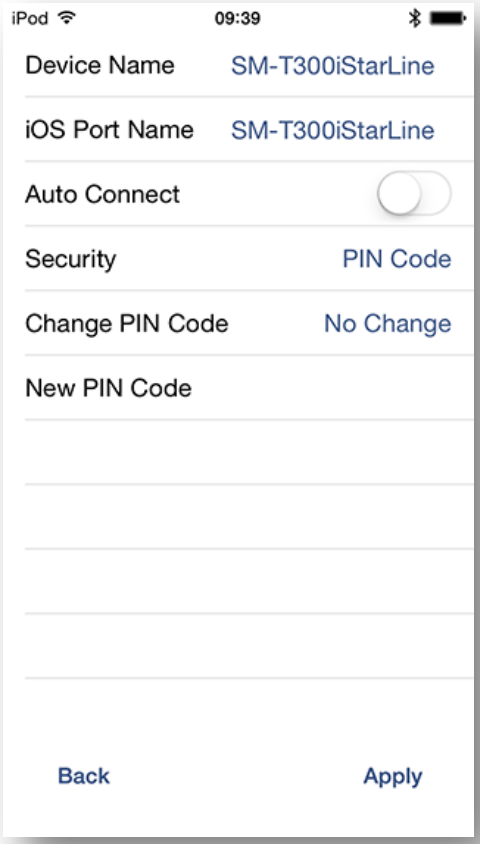


Cloud Services をタップすると、スター精密クラウドサービスへのデバイス登録を行うための Registration View を表示します。

AllReceipts、スター精密クラウドサービスについて詳しくは、Star_AllReceipts_iOS_SDK マニュアルをご参照ください。

Bluetooth Setting

◆Line　　◆Raster



PortName に指定された Bluetooth デバイスに接続して、Bluetooth インターフェイスの各種設定を変更します。

変更された値は、デバイスの電源再投入・再ペアリングを行った後に有効になります。また、Bluetooth Low Energy の場合は、デバイスの電源再投入・再接続を行った後に有効になります。

StarIO を使用するアプリケーション開発のために

#1: 大規模なプロジェクトをコーディングしている場合、全ての印刷メソッドを抽象化してクラスを作成してください。これはソース・コードの再利用に役立ちます。また、特定のコードを見つけるのが容易になり、時間の節約にもなります。StarIO を唯一のクラスに存在させることによって、オブジェクト指向プログラミングを実現できます。

#2: ASCII と Unicode、16 進と 10 進、及び、Byte と Char の違いと定義が何であるかを確認してください。1 バイトは、通常、8 桁の 2 進数(1 と 0)で 8 ビットです。これらのバイトは、ちょうど 8 ビットのバイナリーデータです。しかし、byte は int または char でもありえます。3 つの異なる変数の型は、基本的に同じ方法でデータを保持しますが、わずかな違いがあります。印刷ジョブのデータを格納する変数を選択する際、byte の代わりに char、int、または string で試してみてください。
ASCII から Unicode(また逆も同様に)への変換は、時として安全ではありません。そのため、Encoding クラスがどのように機能するのかを確認してください。Unicode で見られる間違いの例として、”Culture-sensitive searching and casing”、“Surrogate pairs”、“Combining characters”、“Normalization”などがあります。

#3: StarIO コマンド・コードをリバース・エンジニアリングしないでください。全ての StarIO コマンドは、コマンド仕様書より参照可能です。また、本 SDK を活用することで、アプリケーション作成時の工数を大幅に削減可能です。

#4: 本 SDK に記載されていないコマンドについては、SDK のコードサンプルを参照してください。また、**スター精密グローバルサポートサイト**の Developers セクションにアクセスすることで、より詳細な情報を入手可能です。

#5: iOS 以外の OS(例: Android) に対応した SDK をお探しの場合には、**スター精密グローバルサポートサイト**の Developers セクションをご覧ください。

追加リソース

以下リンクよりプログラマーマニュアルを入手してください。

スター精密グローバルサポートサイト

FAQ を参照してください。

グローバルサポートサイトより、以下の情報を入手できます。

- 最新バージョンの SDK マニュアル／ソース・コード
- アドイス／業界情報
- スター精密プリンタードライバ
- 技術的な質問／サポート

Apple Developer Site

Apple の公式開発リソース

Apple Developer Site Resources

Apple ライブラリ - 開発者のためのドキュメントに関する情報の入手

Unicode.org

ユニコードコンソーシアム – Unicode の詳細について

1D Barcodes

Barcode Island - バーコードの詳細について

2D Barcodes

QR Codes、PDF417 に関する情報について

Code Pages

コード・ページに関する情報について

ASCII コード表

10 進	16 進	文字	10 進	16 進	文字	10 進	16 進	文字	10 進	16 進	文字
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	?

10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字
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	□

SDK パッケージ改訂履歴

改訂年月	SDK パッケージバージョン	更新内容
Apr. 2015	3.14.0	・マニュアル新規発行
Aug. 2015	3.15.0	・mPOP 対応追加
Mar. 2016	3.16.0	・AllReceipts 対応追加

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/