

ePOS-Print SDK for iOS

ユーザーズマニュアル

概要

特徴および環境について説明します。

サンプルプログラム

サンプルプログラムの使い方について説明します。

プログラミングガイド

アプリケーション開発のプログラミング方法について説明します。

API リファレンス

ePOS-Print SDK for iOSで提供しているAPIについて説明します。

コマンドの送受信

コマンドを送受信するためのAPIについて説明します。

付録

ePOS-Print SDK for iOSで使用するプリンター仕様について説明します。

ご注意

- 本書の内容の一部または全部を無断で転載、複写、複製、改ざんすることは固くお断りします。
- 本書の内容については、予告なしに変更することがあります。最新の情報はお問い合わせください。
- 本書の内容については、万全を期して作成いたしましたが、万一ご不審な点や誤り、記載もれなど、お気づきの点がありましたらご連絡ください。
- 運用した結果の影響については、上項にかかわらず責任を負いかねますのでご了承ください。
- 本製品がお客様により不適切に使用されたり、本書の内容に従わずに取り扱われたり、またはエプソンおよびエプソン指定の者以外の第三者により修理・変更されたことなどに起因して生じた損害などにつきましては、責任を負いかねますのでご了承ください。
- エプソン純正品およびエプソン品質認定品以外のオプションまたは消耗品を装着してトラブルが発生した場合には、責任を負いかねますのでご了承ください。

商標について

EPSON®、ESC/POS® は、セイコーエプソン株式会社の登録商標です。

Xcode®, iPhone®, iPod touch®, iPad®, iTunes® は、米国 Apple, Inc. の米国およびその他の国で登録された商標です。

iOS® は、Cisco の米国およびその他の国のライセンスに基づき使用されています。

Wi-Fi® は、Wi-Fi Alliance® の登録商標です。



ESC/POS® コマンドシステム

EPSON は、独自の POS プリンターコマンドシステム ESC/POS により、業界のイニシアティブをとってきました。ESC/POS は特許取得済のものを含む数多くの独自のコマンドを持ち、高い拡張性で多才な POS システムの構築を実現します。EPSON POS プリンター (TM-C100 を除く) とディスプレイの全タイプに互換性を持つほか、この独自の制御システムにはフレキシビリティもあるため、将来アップグレードが行ないやすくなります。その機能と利便性は世界中で評価されています。

安全のために

記号の意味

本書では以下の記号が使われています。それぞれの記号の意味をよく理解してから製品を取り扱ってください。

	ご使用上、必ずお守りいただきたいことを記載しています。この表示を無視して誤った取り扱いをすると、製品の故障や動作不良の原因になる可能性があります。
	補足説明や知っておいていただきたいことを記載しています。

使用制限

本製品を航空機・列車・船舶・自動車などの運行に直接関わる装置・防災防犯装置・各種安全装置など機能・精度などにおいて高い信頼性・安全性が必要とされる用途に使用される場合は、これらのシステム全体の信頼性および安全維持のためにフェールセーフ設計や冗長設計の措置を講じるなど、システム全体の安全設計にご配慮いただいた上で当社製品をご使用いただくようお願いいたします。

本製品は、航空宇宙機器、幹線通信機器、原子力制御機器、医療機器など、きわめて高い信頼性・安全性が必要とされる用途への使用を意図しておりませんので、これらの用途には本製品の適合性をお客様において十分ご確認のうえ、ご判断ください。

本書について

本書の目的

本書は、ePOS-Print SDK を使った、印刷システムの構築、設計またはプリンターアプリケーションの開発、設計に必要なすべての情報を開発技術者に提供することを、その目的としています。

本書の構成

本書は次のように構成されています。

第 1 章	概要
第 2 章	サンプルプログラム
第 3 章	プログラミングガイド
第 4 章	API リファレンス
第 5 章	コマンドの送受信
付録	プリンターの仕様

もくじ

■ 安全のために.....	3
記号の意味.....	3
■ 使用制限	3
■ 本書について.....	3
本書の目的.....	3
本書の構成.....	3
■ もくじ	4

概要.....7

■ ePOS-Print SDK の概要.....	7
特徴.....	7
機能.....	8
■ 動作環境	9
iOS バージョン	9
iOS デバイス	9
プリンター	9
開発環境.....	9
■ 提供物.....	10
パッケージ	10
マニュアル.....	10
サンプルプログラム.....	10
ダウンロード	10
■ 制限事項	10

サンプルプログラム.....11

■ 機能	11
■ 使用環境	12
開発環境.....	12
プリンター	12
ターゲット デバイス	12
■ 環境構築	13
■ 使用方法	14
プリンターを検索して印刷する.....	14
プリンターの機種名を取得する.....	21

プログラミングガイド.....23

■ ePOS-Print SDK for iOS の組み込み方法.....	23
■ ePOS-Print API.....	24

印刷モードについて.....	24
プログラミングフロー.....	24
プリンターの検索.....	25
印刷ドキュメントの作成.....	26
印刷ドキュメントの送信.....	28
プリンターの状態を確認してから印刷する.....	29

■ プリンターステータスを自動的に取得.....30

イベント一覧.....	31
-------------	----

■ ステータス.....32

エラーステータス一覧.....	32
プリンターステータス一覧.....	33
バッテリーステータス.....	34

API リファレンス.....35

■ ePOS-Print API.....35

initWithPrinterModel	38
clearCommandBuffer	39
addTextAlign	40
addTextLineSpace	41
addTextRotate	42
addText.....	43
addTextLang	44
addTextFont	45
addTextSmooth.....	46
addTextDouble.....	47
addTextSize.....	48
addTextStyle.....	49
addTextPosition.....	51
addFeedUnit	52
addFeedLine.....	53
addImage(多階調印字用).....	54
addImage	57
addLogo.....	59
addBarcode	60
addSymbol.....	65
addPageBegin	70
addPageEnd	71
addPageArea.....	72
addPageDirection	74
addPagePosition	75
addCut	76
addPulse.....	77
addSound(周期鳴動設定用).....	78
addSound.....	80
addFeedPosition.....	82
addLayout	83
addCommand	85
init.....	86
openPrinter(プリンターステータス取得用).....	87
openPrinter	89

closePrinter	90
sendData	91
sendData(バッテリーステータス取得用)	93
setStatusChangeEventCallback	95
setOnlineEventCallback	96
setOfflineEventCallback	97
setPowerOffEventCallback	98
setCoverOkEventCallback	99
setCoverOpenEventCallback	100
setPaperOkEventCallback	101
setPaperNearEndEventCallback	102
setPaperEndEventCallback	103
setDrawerClosedEventCallback	104
setDrawerOpenEventCallback	105
setBatteryLowEventCallback	106
setBatteryOkEventCallback	107
setBatteryStatusChangeEventCallback	108
■ プリンター検索 API	109
start	109
stop	110
getResult	111
■ ログ設定 API	112
setLogSettings	112

コマンドの送受信115

■ プログラミング	115
プログラミングフロー	115
EpsonIo クラスの初期化	116
デバイスポートのオープン	116
データの送信	116
データの受信	117
デバイスポートのクローズ	117
■ エラー値一覧	118
■ コマンド送受信 API リファレンス	119
init	119
open	120
close	121
write	122
read	123

付録125

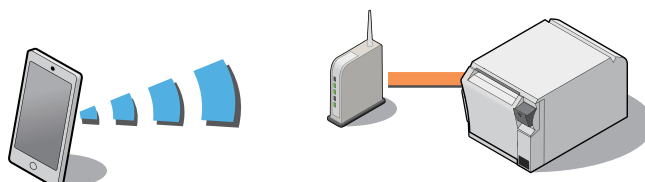
■ プリンターの仕様	125
TM-T88V	125
TM-T70	127
TM-T70II	129
TM-T90II	131
TM-P60II	133



概要

本章では、ePOS-Print SDK for iOS の特徴および仕様について説明しています。

ePOS-Print SDK の概要



ePOS-Print SDK for iOS は、EPSON TM プリンターに印刷するための iOS アプリケーションを開発する、開発者向け SDK です。ePOS-Print SDK で提供する API を使用してアプリケーションを開発します。

ePOS-Print SDK には、Android アプリケーション向けの ePOS-Print SDK for Android も用意されています。



TM プリンターにコマンドを送受信するための API も用意されています。
コマンド送受信 API は、ePOS-Print API の EposPrint クラスと同時に使用できません。
コマンド送受信 API の詳細は[コマンドの送受信 \(115 ページ\)](#)を参照してください。

特徴

- ❑ iOS アプリケーションから、TM プリンターに印刷できます。
- ❑ iOS アプリケーションから、TM プリンターのステータスを取得できます。

ePOS-Print API

- 印字の設定（位置揃え / 改行量 / 倒立印字 / ページモード）
- 文字データの設定（言語 / フォント（デバイスフォント）/ 倍角 / 倍率 / スムージング / 印字位置）
- 文字書式の設定（白黒反転 / アンダーライン / 太字）
- 紙送り設定（ドット単位 / 行単位）
- 画像の印字（ラスターイメージ / NV グラフィック）
- バーコードの印字
（機種ごと印字できるバーコードは、[プリンターの仕様（125 ページ）](#)を参照してください。）
- 2 次元シンボルの印字
（機種ごと印字できる 2 次元シンボルは、[プリンターの仕様（125 ページ）](#)を参照してください。）
- ドロアーキック機能
- ブザー機能
- 用紙レイアウトの設定
- ラベル / ブラックマーク紙の紙送り設定
- ESC/POS コマンドの送信
- プリンターからの応答を取得（印字結果 / プリンターの状態 / バッテリーの状態）

プリンター検索 API

- プリンターの検索

ログ設定 API

- ログ出力の設定
（iOS 端末のストレージや、TCP 接続できるサーバーに出力できます。）



iOS 端末に出力したログは、USB 接続で他のコンピューターにも保存できます。

動作環境

iOS バージョン

- ❑ iOS Version 4.2 ~ 4.3.5
- ❑ iOS Version 5.0 ~ 5.1.1
- ❑ iOS Version 6.0 ~ 6.1.4
- ❑ iOS Version 7.0 ~ 7.0.3



最新のバージョンは、README ファイルを参照してください。

iOS デバイス

- ❑ iPhone (3G/3GS/4/4S/5/5s/5c)
- ❑ iPod touch (第2世代 / 第3世代 / 第4世代 / 第5世代)
- ❑ iPad/iPad 2/iPad (第3世代)/iPad (第4世代)/iPad Mini

プリンター

TM プリンター	インターフェイス	
	有線 LAN	無線 LAN
TM-T88V	○	○
TM-T70	○	○
TM-T70II	○	○
TM-P60II	-	○
TM-T90II	○	○

開発環境

iOS アプリケーションを開発するには、以下が必要です。

- Xcode Version 4.2 ~ 4.6
- Xcode Version 5.0

提供物

パッケージ

ファイル名	説明
ePOS-Print.h	クラス定義、エラー値 / デバイスタイプの定数定義を含むヘッダーファイルです。
libeposprint.a	機能実行用ライブラリーです。(armv6, armv7, armv7s, arm64, i386 に対応)
ePOS-Print_Sample_iOS.zip	サンプルプログラムファイルです。
README.jp.txt	README ファイルです。
README.en.txt	README ファイルです(英語版)です。
EULA.jp.txt	SOFTWARE LICENSE AGREEMENT が記載されています。
EULA.en.txt	SOFTWARE LICENSE AGREEMENT(英語版) が記載されています。
ePOS-Print_SDK_iOS_J_Revx.pdf	本書です。
ePOS-Print_SDK_iOS_E_Revx.pdf	本書(英語版)です。

マニュアル

ePOS-Print SDK for iOS のマニュアルには以下のものがあります。

- ❑ ePOS-Print SDK for iOS ユーザーズマニュアル(本書)
- ❑ ePOS-Print SDK for iOS アプリケーション開発環境 - セットアップガイド

サンプルプログラム

ePOS-Print SDK を使用した、TM プリンター用 iOS アプリケーションのサンプルプログラムがあります。

- ❑ ePOS-Print_Sample_iOS.zip

ダウンロード

提供物は、下記エプソン販売ホームページからダウンロードできます。

<http://www.epson.jp/support/sd/>

制限事項

- ❑ ePOS-Print APIの通信API(37ページ)とコマンド送受信API(115ページ)は、同一デバイスに対して同時に使用することはできません。
- ❑ 同じアプリケーション内で同時にオープンできるデバイスポート数は 16 個です。
- ❑ 同じデバイスに対して同時に複数オープンできません。

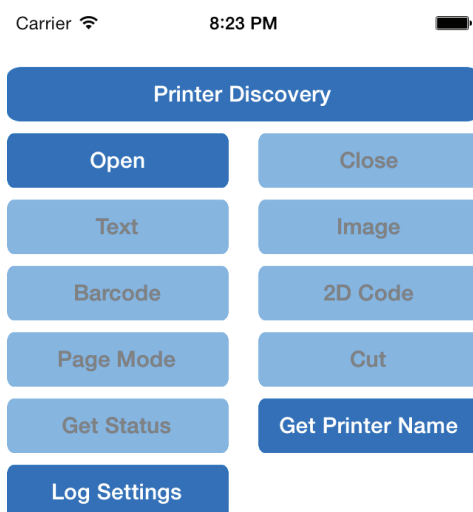
サンプルプログラム

本章では、サンプルプログラム (ePOS-Print Sample Program for iOS) の使い方について説明しています。



- サンプルプログラムは iOS アプリケーション開発者向けの、ePOS-Print for iOS API を使用した iOS アプリケーションの実装サンプルとして提供します。
- サンプルプログラムのパッケージは、Objective-C ソースファイルを含む Xcode 用 iOS アプリケーションプロジェクトとして提供しています。

機能



サンプルプログラムは、以下の機能を実装しています。

- ☐ プリンターの検索
- ☐ ポートオープン
- ☐ ポートクローズ
- ☐ テキスト印刷
- ☐ グラフィック印刷 (画像ファイルの印刷)
- ☐ バーコード印刷
- ☐ 2次元シンボル印刷
- ☐ ページモード印刷
- ☐ 用紙カット
- ☐ プリンターのステータス取得
- ☐ プリンターの機種名 / 言語情報取得
- ☐ ログ出力設定
- ☐ ステータスイベントの表示
- ☐ バッテリーステータスイベントの表示



サンプルプログラムでは、文字 / 画像 / バーコードなどを回転させる機能は実装していません。

使用環境

開発環境

- Xcode Version 4.2 ～ 4.6
- Xcode Version 5.0



開発環境構築の詳細は、「ePOS-Print SDK for iOS アプリケーション開発環境 - セットアップガイド」を参照してください。

プリンター

- ePOS-Print SDK でサポートしている TM プリンター

ターゲット デバイス

- コンピューターに USB 接続されているデバイス

環境構築

以下の手順でサンプルプログラムの環境を構築します。

- 1** サンプルプログラムの ZIP ファイルを任意のディレクトリーに展開します。
- 2** 展開したディレクトリー内に格納されている、“ePOS-Print-Sample.xcodeproj” をダブルクリックします。
- 3** Xcode が起動されます。「Scheme」としてターゲットデバイスを選択します。
- 4** 左上の [Run] ボタンをクリックします。
- 5** ターゲットデバイスにサンプルプログラムがインストールされ、サンプルプログラムが起動します。

使用方法

ここでは、以下の使い方を説明します。

- [プリンターを検索して印刷する \(14 ページ\)](#)
- [プリンターの機種名を取得する \(21 ページ\)](#)

プリンターを検索して印刷する

以下の手順で使用します。

- 1 サンプルプログラムを起動します。詳細は、[環境構築 \(13 ページ\)](#) を参照してください。
- 2 プリンターを検索します。メイン画面で [Printer Discovery] をタップします。検索されたプリンターの IP アドレスがリスト表示されます。
- 3 手順2でリスト表示されたプリンターの IP アドレスの中から、使用するプリンターを選択します。
- 4 プリンターのポートをオープンします。メイン画面で [Open] をタップします。
手順3で選択したプリンターの“Device Type”と“IP Address”が表示されます。[Printer Name] と [Language] を選択します。

- 5 [Status Monitor] を設定します。

項目	説明
Enabled	<ul style="list-style-type: none">• ON: ステータスマニターを有効にし、プリンターステータスの監視を行います。• OFF: ステータスマニターを無効にします。
Interval	(Enabled) を ON に設定した場合、ステータスの監視間隔をミリ秒単位で設定します。

- 6 [Open] をタップします。

7 以下の処理を実行します。

処理	説明
テキスト印刷	メイン画面の (Text) をタップしてください。 詳細は テキスト印刷 (16 ページ) を参照してください。
グラフィック印刷	メイン画面の (Image) をタップしてください。 詳細は グラフィック印刷 (16 ページ) を参照してください。
バーコード印刷	メイン画面の (Barcode) をタップしてください。 詳細は バーコード印刷 (17 ページ) を参照してください。
2次元シンボル印刷	メイン画面の (2D Code) をタップしてください。 詳細は 2次元シンボル印刷 (17 ページ) を参照してください。
ページモード印刷	メイン画面の (Page Mode) をタップしてください。 詳細は ページモード印刷 (17 ページ) を参照してください。
用紙カット	メイン画面の (Cut) をタップしてください。 詳細は 用紙カット (18 ページ) を参照してください。
プリンタステータスの取得	メイン画面の (Get Status) をタップしてください。
ログ出力設定	メイン画面の (Log Settings) をタップしてください。 詳細は ログ出力設定 (18 ページ) を参照してください。

8 以下の処理の実行結果が表示されます。

- 処理実行結果 (エラーステータス / プリンタステータス / バッテリーステータス)
詳細は、[処理実行結果 \(19 ページ\)](#) を参照してください。
- メソッド (API) 実行エラー
詳細は、[メソッド \(API\) 実行エラー \(20 ページ\)](#) を参照してください。

9 処理をすべて終わったら、メイン画面の [Close] をタップし、プリンターのポートをクローズします。

テキスト 印刷

以下の手順で実行します。

- 1 [Print Characters] に印字文字列を入力します。
- 2 印字文字列に文字の属性を指定します。以下の属性を指定できます。

属性	説明
Font	文字フォントを設定します。
Align	位置揃えを設定します。
Line Spacing	改行量を設定します。
Language	言語を設定します。
Size	文字の倍率（縦倍 / 横倍）を設定します。
Style	文字修飾（ボールド / アンダーライン）を設定します。
X Position	横位置の開始位置を設定します。
Feed Unit	紙送り量を設定します。

- 3 [Print] をタップし、印刷します。

グラフィック印刷

以下の手順で実行します。

- 1 [Select Image] をタップし、印刷する画像ファイルを選択します。
- 2 [Color Mode] をタップし、階調を選択します。



[Gray 16]（多階調印刷）を選択できる機種は、TM-T88V/ TM-T70II/ TM-T90II のみです。

- 3 [Halftone Method] をタップし、ハーフトーンの処理方法を選択します。
- 4 [Brightness] にタップし、数値を入力して明るさを指定します。
- 5 [Print] をタップし、印刷します。

バーコード印刷

以下の手順で実行します。

1 以下のバーコードの設定をします。

設定	説明
Type	バーコードの種類を選択します。
Data	バーコードデータを入力します。
HRI	HRI の位置を設定します。
Font	HRI フォントを設定します。
Module Size(Width, Height)	バーコードのモジュールサイズ(幅 / 高さ)を設定します。

2 [Print] をタップし、印刷します。

2 次元シンボル印刷

以下の手順で実行します。

1 [Type] から 2 次元シンボルの種類を選択します。

2 [Data] に 2 次元シンボルデータを入力します。

3 2 次元シンボルの種類ごとに、以下を設定します。

設定	説明
Error Correction Level (PDF417, QR Code, Aztec Code, DataMatrix)	エラー訂正レベルを設定します。
Module Size(Width, Height)	2 次元シンボルのモジュールサイズ(幅 / 高さ)を設定します。
Max Size	2 次元シンボルの最大サイズを設定します。

4 [Print] をタップし、印刷します。

ページモード印刷

以下の手順で実行します。

1 [Print Characters] に印字文字列を入力します。

2 [Print Area] で印字領域の設定をします。

設定	説明
X	横方向の原点を設定します。
Y	縦方向の原点を設定します。
Width	印字領域の幅を設定します。
Height	印字領域の高さを設定します。

3 [Print] をタップし、印刷します。

用紙カット

以下の手順で実行します。

- 1 [Type] で紙送りしてカットするか設定します。
- 2 [Print] をタップし、カットを実行します。

ログ出力設定

以下の手順で設定します。

- 1 [Enabled] に、ログ出力の有無と、ログの出力先を設定します。
- 2 ログの出力先に合わせて、以下を設定します。

設定	説明
IP Address	TCP 通信の IP アドレスを指定します。
Port	TCP 通信用のポート番号を指定します。
Log Size	端末のストレージに保存する、ログの最大容量を指定します。
Log Level	出力するログのレベルを設定します。

- 3 [Save Settings Permanently] に、設定の保存方法を設定します。
- 4 [Setting] をタップし、ログ出力の設定を有効にします。

実行結果

処理実行結果

以下の表示がされます。

- Result: 以下のエラーステータスが表示されます。

表示文字列	説明
SUCCESS	成功
ERR_PARAM	不正なパラメーターが渡された
ERR_ILLEGAL	不適切な方法で使用された
ERR_PROCESSING	処理を実行できなかった
ERR_TIMEOUT	処理がタイムアウトされた
ERR_CONNECT	デバイスとの通信に失敗した
ERR_MEMORY	処理に必要なメモリーが確保できなかった
ERR_OFF_LINE	オフライン状態
ERR_FAILURE	その他のエラーが発生した

- Status: 以下のプリンターステータスが表示されます。

表示文字列	説明
NO_RESPONSE	プリンター無応答
PRINT_SUCCESS	印刷完了
DRAWER_KICK	ドロアーキックコネクター 3 番ピンの状態 = "H"
BATTERY_OFFLINE	バッテリーオフライン状態
OFF_LINE	オフライン状態
COVER_OPEN	カバーが開いている
PAPER_FEED	紙送りスイッチによる紙送信中
WAIT_ON_LINE	オンライン復帰待ち中
PANEL_SWITCH	紙送りスイッチが押されている
MECHANICAL_ERR	メカニカルエラー発生
AUTOCUTTER_ERR	オートカッターエラー発生
UNRECOVER_ERR	復帰不可能エラー発生
AUTORECOVER_ERR	自動復帰エラー発生
RECEIPT_NEAR_END	ロール紙ニアエンド検出器に用紙なし
RECEIPT_END	ロール紙エンド検出器に用紙なし

- Battery Status: 以下が表示されます。

表示文字列	説明
0xnxxx	バッテリーステータス 詳細は、 バッテリーステータス (34 ページ) を参照してください。

メソッド (API) 実行エラー

以下の表示がされます。

- Error Code: 以下のエラーステータスが表示されます。

表示文字列	説明
ERR_PARAM	不正なパラメーターが渡された
ERR_OPEN	オープン処理に失敗した
ERR_CONNECT	デバイスとの通信に失敗した
ERR_TIMEOUT	指定された時間内に全てのデータを送信できなかった
ERR_MEMORY	処理に必要なメモリーが確保できなかった
ERR_ILLEGAL	不適切な方法で使用された
ERR_PROCESSING	処理を実行できなかった
ERR_UNSUPPORTED	サポートしていない機種名または言語使用が指定された」
ERR_OFF_LINE	プリンターがオフライン状態だった
ERR_FAILURE	その他のエラーが発生した

- Method: メソッド実行エラーになった API が表示されます。

プリンターの機種名を取得する



プリンターの機種名の取得は、コマンド送受信 API を使用しています。詳細は、[コマンドの送受信 \(115 ページ\)](#) を参照してください。

以下の手順で 사용합니다。

- 1 サンプルプログラムを起動します。詳細は、[環境構築 \(13 ページ\)](#) を参照してください。
- 2 プリンターを検索します。メイン画面で [Printer Discovery] をタップします。検索されたプリンターの IP アドレスがリスト表示されます。
- 3 手順 2 でリスト表示されたプリンターの IP アドレスの中から、使用するプリンターを選択します。
- 4 メイン画面の [Get Printer Name] をタップします。
- 5 [Get Printer Name] をタップします。
- 6 以下が表示されます。

表示内容	説明
Printer Name	プリンターの機種名が表示されます。
Language	プリンターの言語仕様が表示されます。



プログラミングガイド

本章では、ePOS-Print SDK を使用したアプリケーション開発のプログラミング方法について説明します。



ePOS-Print SDK for iOS を使用した、iOS アプリケーション開発環境の構築方法については、「ePOS-Print SDK for iOS アプリケーション開発環境 - セットアップガイド」を参照してください。

ePOS-Print SDK for iOS の組み込み方法

ePOS-Print SDK for iOS の組み込み方法について説明します。

以下の手順で組み込んでください。

- 1 Xcode で新しいプロジェクトを作成します。
- 2 提供されたObjective-Cヘッダー(ePOS-Print.h)をXcode の[Project Navigator]の対象プロジェクトの任意の階層にドラッグします。
- 3 提供されたスタティックライブラリー(libeposprint.a)をXcode の[Project Navigator]の対象プロジェクトの任意の階層にドラッグします。
- 4 使用したいアプリケーションの*.mソースファイルで、Objective-C ヘッダーのインポート定義を記載します。下記を参照してください。

```
#import "ePOS-Print.h"
```

ePOS-Print API

印刷モードについて

印字モードにはスタンダードモードとページモードがあります。

スタンダードモード

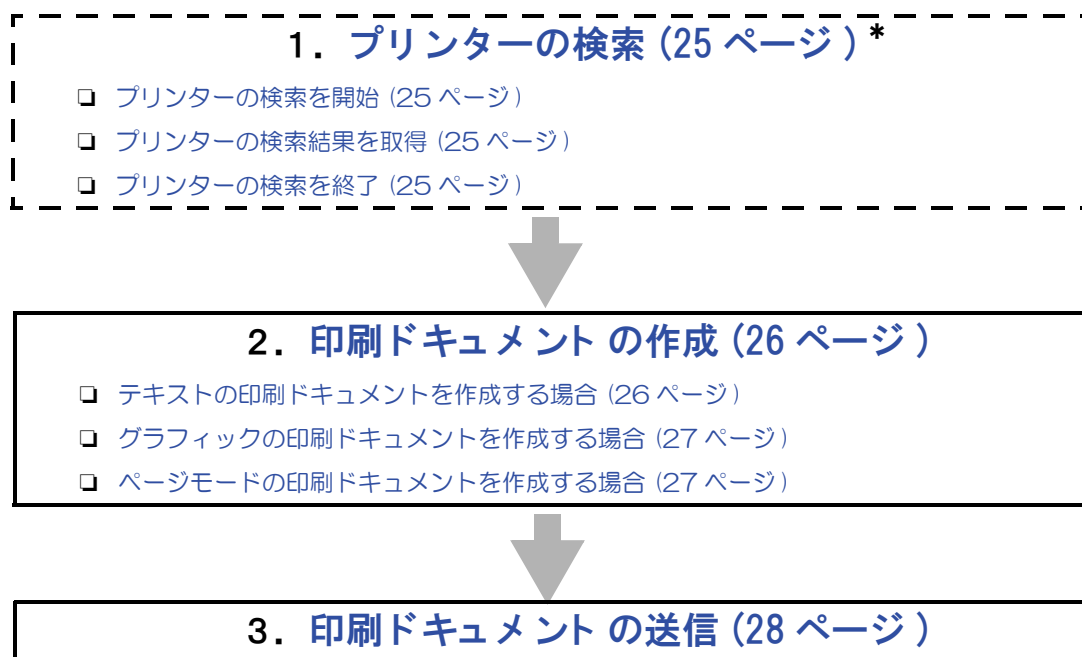
スタンダードモードとは、1行単位で印字する印字モードです。文字サイズ、画像、バーコードなどの高さに合わせて改行量が調整されます。レシートなど、印字量に合わせて用紙の長さが変化する印刷に向いています。

ページモード

ページモードとは、印字領域を設定して印字データを展開し、一括印字する印字モードです。印字位置（座標）に、文字、画像、バーコードなどを展開します。

プログラミングフロー

以下のフローでプログラミングします。



*: 任意のプロセスです。



あらかじめプリンターの状態を確認して送信するプログラミングをすると、確実に印刷できます。方法については、[プリンターの状態を確認してから印刷する \(29 ページ\)](#)を参照してください。

プリンターの検索

プリンターの検索を開始

EpsonIoFinder クラスの [start \(109 ページ\)](#) を使って、プリンターの検索を開始します。以下のプログラミングを参考にしてください。

```
// 検索開始
int errStatus =
[EpsonIoFinder start:EPSONIO_OC_DEVTYPE_TCP FindOption:@"255.255.255.255"];
```

プリンターの検索結果を取得

EpsonIoFinder クラスの [getResult \(111 ページ\)](#) を使って、プリンターの検索結果を取得します。以下のプログラミングを参考にしてください。

```
int errStatus = EPSONIO_OC_SUCCESS;

// デバイス一覧の取得
NSArray *array = [EpsonIoFinder getResult:&errStatus];
```



プリンターの検索に時間がかかるため、EpsonIoFinder クラスの start を呼んだ直後に getResult を呼んだ場合、検索結果が無いこともあります。

プリンターの検索を終了

EpsonIoFinder クラスの [stop \(110 ページ\)](#) を使って、プリンターの検索を終了します。以下のプログラミングを参考にしてください。

```
// 検索終了
int errStatus = [EpsonIoFinder stop];
```

印刷ドキュメントの作成

印刷ドキュメントは、[EposBuilder クラス \(35 ページ\)](#) で作成します。

コンストラクターで EposBuilder クラスを作成し、EposBuilder クラスの各 API で印刷ドキュメントを作成します。
以下のプログラミングを参考にしてください。

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                           Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    // 印刷ドキュメントの作成
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 3 Height: 3];
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
    [builder release];
}
```

テキストの印刷ドキュメントを作成する場合

テキストの印刷ドキュメントを作成する場合、テキストの各 API でフォントの設定を命令バッファに格納し、印刷ドキュメントを作成します。以下のプログラミングを参考にしてください。

文字列「Hello, World!」を以下の設定で印刷ドキュメントを作成する場合

- フォント: FontA
- 倍率: 幅 4 倍、高さ 4 倍
- スタイル: 太字

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                           Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    // 印刷ドキュメントの作成
    int errorStatus = EPOS_OC_SUCCESS;
    //< 印字文字の設定 >
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 4 Height: 4];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE Ul: EPOS_OC_FALSE
                                     Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    //< 印刷データを指定 >
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```

グラフィックの印刷ドキュメントを作成する場合

グラフィックの印刷ドキュメントを作成する場合、グラフィックは、UIImage クラスを EposBuilder クラスの `addImage` (57 ページ) で命令バッファに格納します。以下のプログラミングを参考にしてください。

```
UIImage * imageData = [UIImage imageNamed:@"Sample.png"];

//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                                    Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    //印刷ドキュメントの作成
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256 Height: 256
                                Color: EPOS_OC_PARAM_DEFAULT];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```



グラフィック印刷する方法には、プリンターの NV メモリーに登録されているグラフィックを印字する方法もあります。詳細は、[addLogo](#) (59 ページ) を参照してください。

ページモードの印刷ドキュメントを作成する場合

EposBuilder クラスの `addPageBegin` (70 ページ) を命令バッファに格納することで、ページモードが開始されます。印字領域 (`addPageArea` (72 ページ)) と印字開始位置 (`addPagePosition` (75 ページ)) を命令バッファに格納します。印字開始位置は、印字データに合わせて指定します。その後、各 API を命令バッファに格納し印字データを作成します。ページモードの終了は `addPageEnd` (71 ページ) を命令バッファに格納します。

文字列「Hello, World!」を以下の設定で印刷ドキュメントを作成する場合

- ページモードの印字領域 (ドット単位):
横方向原点:100, 縦方向原点:50, 幅:200, 高さ:100
- ページモードの印字位置 (ドット単位):
横方向の印字位置:0, 縦方の印字位置:42
- フォント: FontA
- 倍率: 幅 2 倍、高さ 2 倍
- スタイル: 太字

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                                    Lang: EPOS_OC_MODEL_JAPANESE];

if ( builder != nil ) {
    //印刷ドキュメントの作成
    //＜ページモード開始＞
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 100];
    errorStatus = [builder addPagePosition: 0 Y: 42];
    //＜印字文字の設定＞
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 2 Height: 2];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE U1: EPOS_OC_FALSE
                                Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    //＜印刷データを指定＞
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    //＜ページモード終了＞
    errorStatus = [builder addPageEnd];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}
```

印刷ドキュメントの送信

印刷ドキュメントは、[EposPrint クラス \(37 ページ\)](#) で送信します。

コンストラクターで EposPrint クラスを作成し、sendData で、印刷ドキュメントの命令バッファを格納した EposBuilder クラスのインスタンスを指定して送信します。以下のプログラミングを参考にしてください。

```
//EposBuilder クラスのインスタンスを初期化
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                                    Lang: EPOS_OC_MODEL_JAPANESE];

if (builder != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    // 印刷ドキュメントの作成
    // < 印字文字の設定 >
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    errorStatus = [builder addTextFont: EPOS_OC_FONT_A];
    errorStatus = [builder addTextSize: 4 Height: 4];
    errorStatus = [builder addTextStyle: EPOS_OC_FALSE Ul: EPOS_OC_FALSE
                                         Em: EPOS_OC_TRUE Color: EPOS_OC_PARAM_UNSPECIFIED];
    // < 印刷データを指定 >
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
    //EposPrint クラスのインスタンスを初期化
    id printer = [[EposPrint alloc] init];
    long status;
    // 印刷ドキュメントを送信
    if (printer != nil) {
        // < プリンターとの通信を開始 >
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP DeviceName:@"192.168.192.168"];
        // < データを送信 >
        errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
        // < プリンターとの通信を終了 >
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

プリンターの状態を確認してから印刷する

あらかじめプリンターの状態を確認してから印刷すると、確実に印刷できます。
空の印刷データを送信し、プリンターがオンライン状態の場合に印刷します。

以下を参考にしてください。

```
// 印刷ドキュメントの作成
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                     Lang: EPOS_OC_MODEL_JAPANESE];

if (builder != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    // 印刷ドキュメントの作成
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
}

// 確認用 EposBuilder クラスのインスタンスを初期化
id conBuilder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
                                     Lang: EPOS_OC_MODEL_JAPANESE];

// EposPrint クラスのインスタンスを初期化
id printer = [[EposPrint alloc] init];
long status;
if (printer != nil) {
    // <プリンターとの通信を開始>
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP DeviceName:@"192.168.192.168"];

    // <確認用データを送信>
    errorStatus = [printer sendData:conBuilder Timeout:10000 Status:&status];

    if (errorStatus == EPOS_OC_SUCCESS && (status & EPOS_OC_ST_OFF_LINE)
        != EPOS_OC_ST_OFF_LINE) {
        // <印刷データを送信>
        errorStatus = [printer sendData:builder Timeout:10000 Status:&status];
    }

    // <プリンターとの通信を終了>
    errorStatus = [printer closePrinter];
    [printer release];
}
[conBuilder release];
[builder release];
}
```

- 1 印刷データを作成します。
- 2 プリンターの状態を確認するために空の印刷データを作成します。
- 3 ②で作成した印刷データを送信します。
- 4 ②で作成した印刷データが正常に送信され、プリンターがオンライン状態の場合に、続けて①で作成した印刷データを送信します。

プリンタステータスを自動的に取得

ePOS-Print SDK では、プリンタの状態をコールバックで自動的にアプリケーションに通知できます。

以下を参考にしてください。

```
// プリンタステータスを通知するコールバックメソッドを実装
- (void)onStatusChange:(NSString *)deviceName Status:(NSNumber *)status
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        // プリンタステータスの通知先のコールバックメソッドを登録
        [printer setStatusChangeEventCallback @selector(onStatusChange:Status:)
        Target:self];

        // プリンターとの通信、およびプリンタステータスのモニタリングを開始
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        Name:@"192.168.192.168"
        Enabled: EPOS_OC_TRUE
        Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

① / ④

②

③

1 イベント発生時の通知先のコールバックメソッドを実装します。



上記では、プリンタステータスを `openPrinter(プリンタステータス取得用)` (87 ページ) で指定した間隔で通知するコールバックメソッドを実装しています。ePOS-Print には、カバーオープンやドロアオープンのイベントといった、プリンタステータスごとのコールバックメソッドも用意されています。用途に応じて使い分けてください。ePOS-Print で使用できるコールバックメソッドは、[イベント一覧 \(31 ページ\)](#) を参照してください。

2 プリンタステータスの通知先を登録します。

3 `openPrinter` (89 ページ) を使って、プリンタステータスのモニタリングを開始します。

4 ①で実装したイベントにプリンタステータスを通知します。



プリンタステータスの通知を終了した場合、EposPrint クラスの `closePrinter` (90 ページ) で終了します。

イベント一覧



コールバックメソッドの詳細は、下記、コールバックメソッド登録 API を説明している、[API リファレンス \(35 ページ\)](#) を参照してください。

イベント	コールバックメソッド登録 API
プリンタステータスの通知	setStatusChangeEventCallback (95 ページ)
オンラインの通知	setOnlineEventCallback (96 ページ)
オフラインの通知	setOfflineEventCallback (97 ページ)
無応答の通知	setPowerOffEventCallback (98 ページ)
カバークローズの通知	setCoverOkEventCallback (99 ページ)
カバーオープンの通知	setCoverOpenEventCallback (100 ページ)
用紙ありの通知	setPaperOkEventCallback (101 ページ)
用紙残量少の通知	setPaperNearEndEventCallback (102 ページ)
用紙なしの通知	setPaperEndEventCallback (103 ページ)
ドロアークローズの通知	setDrawerClosedEventCallback (104 ページ)
ドロアーオープンの通知	setDrawerOpenEventCallback (105 ページ)
バッテリー残量なしの通知	setBatteryLowEventCallback (106 ページ)
バッテリー残量ありの通知	setBatteryOkEventCallback (107 ページ)
バッテリーステータスの通知	setBatteryStatusChangeEventCallback (108 ページ)

ステータス

ePOS-Print SDK for iOS には、以下のステータスが定義されています。

ステータス	説明
エラーステータス	各クラスの API 実行時の戻り値です。 詳細は、 エラーステータス一覧 (32 ページ) を参照してください。
プリンターステータス	印刷データ送信時のプリンターのステータスです。 プリンターステータスは、 sendData (91 ページ) を実行時に取得します。 詳細は、 プリンターステータス一覧 (33 ページ) を参照してください。
バッテリーステータス	プリンターのバッテリー残量のステータスです。 詳細は、 バッテリーステータス (34 ページ) を参照してください。

エラーステータス一覧

エラーステータス	要因
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。 < 例 > • Nil など、不正な引数を渡された。 • サポートしていない範囲の値が指定された。
EPOS_OC_ERR_OPEN	オープン処理に失敗した < 例 > 指定したプリンターに接続できなかった。
EPOS_OC_ERR_CONNECT	プリンターとの通信に失敗した。 < 例 > プリンターへのデータ送信に失敗した。
EPOS_OC_ERR_TIMEOUT	指定されたタイムアウト時間を越えた。 < 例 > 指定された時間内にすべてのデータを送信できなかった。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_ILLEGAL	不適切な方法で使用された。 < 例 > プリンターがオープンされていない状態でプリンターにコマンドを送信する API が呼び出された。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。 < 例 > 同様の処理を他のスレッドで実行中のため、処理を実行できなかった。
EPOS_OC_ERR_UNSUPPORTED	サポートしていない機種名または言語仕様が指定された。
EPOS_OC_ERR_OFF_LINE	プリンターがオフライン状態だった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

プリンターステータス一覧

プリンターステータス	要因
EPOS_OC_ST_NO_RESPONSE (0x00000001)	プリンター無応答
EPOS_OC_ST_PRINT_SUCCESS (0x00000002)	印刷完了
EPOS_OC_ST_DRAWER_KICK (0x00000004)	ドロアーキックコネクター 3 番ピンの状態 = "H"
EPOS_OC_ST_BATTERY_OFFLINE (0x00000004)	バッテリー残量によるオフライン状態
EPOS_OC_ST_OFF_LINE (0x00000008)	オフライン状態
EPOS_OC_ST_COVER_OPEN (0x00000020)	カバーが開いている
EPOS_OC_ST_PAPER_FEED (0x00000040)	紙送りスイッチによる紙送信中
EPOS_OC_ST_WAIT_ON_LINE (0x00000100)	オンライン復帰待ち中
EPOS_OC_ST_PANEL_SWITCH (0x00000200)	紙送りスイッチが押されている
EPOS_OC_ST_MECHANICAL_ERR (0x00000400)	メカニカルエラー発生
EPOS_OC_ST_AUTOCUTTER_ERR (0x00000800)	オートカッターエラー発生
EPOS_OC_ST_UNRECOVER_ERR (0x00002000)	復帰不可能エラー発生
EPOS_OC_ST_AUTORECOVER_ERR (0x00004000)	自動復帰エラー発生
EPOS_OC_ST_RECEIPT_NEAR_END (0x00020000)	ロール紙ニアエンド検出器に用紙なし
EPOS_OC_ST_RECEIPT_END (0x00080000)	ロール紙エンド検出器に用紙なし
EPOS_OC_ST_BUZZER (0x01000000)	<ul style="list-style-type: none"> • ブザーが鳴っている (対応機器のみ) • ラベル除去待ち状態 (対応機器のみ)

バッテリーステータス

バッテリーステータスは、以下の 16 ビット (0x0000) で構成されています。

ビット	説明
上位 8 ビット	共通のバッテリーステータス (詳細は、 共通のバッテリーステータス (上位 8 ビット) (34 ページ) を参照してください。)
下位 8 ビット	機種専用のバッテリーステータス (詳細は、 プリンターの仕様 (125 ページ) を参照してください。)



バッテリーステータス取得不可能状態、もしくは機種がバッテリーステータスに対応していない場合、“0x0000” を返します。

共通のバッテリーステータス (上位 8 ビット)

バッテリーステータス	要因
0x30	AC アダプターが接続されている
0x31	AC アダプターが接続されていない

APIリファレンス

本章では、ePOS-Print SDK for iOS で用意されている API について説明しています。

ePOS-Print API

ePOS-Print API は、印刷ドキュメントを作成し、印刷処理を行う API です。

以下のクラスが用意されています。

□ EposBuilder クラス (35 ページ)

□ EposPrint クラス (37 ページ)



プリンターによって、使用できる API や指定可能な設定値は異なります。
詳細は、[プリンターの仕様 \(125 ページ\)](#) を参照してください。

EposBuilder クラス

印字する文字列やグラフィックの印刷、用紙カットなどプリンターの制御命令の印刷ドキュメントを作成します。

以下の API が用意されています。

API		説明	ページ
initWithPrinterModel		EposBuilder クラスのインスタンスを初期化	38
命令バッファのクリア	clearCommandBuffer	各 API で追加した命令バッファをクリア	39
テキスト	addTextAlign	位置揃え設定を命令バッファに追加	40
	addTextLineSpace	改行量設定を命令バッファに追加	41
	addTextRotate	倒立印字設定を命令バッファに追加	42
	addText	文字印字を命令バッファに追加	43
	addTextLang	言語設定を命令バッファに追加	44
	addTextFont	文字フォント設定を命令バッファに追加	45
	addTextSmooth	文字スムージング設定を命令バッファに追加	46
	addTextDouble	文字倍角設定を命令バッファに追加	47
	addTextSize	文字倍率設定を命令バッファに追加	48
	addTextStyle	文字装飾設定を命令バッファに追加	49
	addTextPosition	文字印字位置設定を命令バッファに追加	51
紙送り	addFeedUnit	ドット単位の紙送りを命令バッファに追加	52
	addFeedLine	行単位の紙送りを命令バッファに追加	53
	addPaperFeedPosition	ラベル/ブラックマーク紙の紙送りを命令バッファに追加	82
グラフィック	addImage (多階調印字用)	多階調のラスターイメージ印字を命令バッファに追加	54
	addImage	ラスターイメージ印字を命令バッファに追加	57
	addLogo	NV ロゴ印字を命令バッファに追加	59
バーコード	addBarcode	バーコード印字を命令バッファに追加	60
	addSymbol	2次元シンボル印字を命令バッファに追加	65

API		説明	ページ
ページモード	addPageBegin	ページモード開始を命令バッファに追加	70
	addPageEnd	ページモード終了を命令バッファに追加	71
	addPageArea	ページモード印字領域設定を命令バッファに追加	72
	addPageDirection	ページモード印字方向設定を命令バッファに追加	74
	addPagePosition	ページモード印字位置設定を命令バッファに追加	75
カット	addCut	用紙カットを命令バッファに追加	76
ドロアーキック	addPulse	ドロアーキックを命令バッファに追加	77
ブザー	addSound (周期鳴動設定用)	ブザー鳴動を鳴動周期を設定し、命令バッファに追加	78
	addSound	ブザー鳴動を命令バッファに追加	80
用紙レイアウト	addLayout	用紙レイアウト情報を命令バッファに追加	83
コマンド送信	addCommand	コマンドを命令バッファに追加	85

EposPrint クラス

EposBuilder クラスで作成した印刷ドキュメントを送信してプリンターを制御したり、送信結果や通信状態を監視したりします。

API	説明	ページ
init	ePOS-Print クラスのインスタンスを初期化	86
openPrinter (プリンターステータス取得用)	プリンターとの通信とプリンターステータスのモニタリングを開始	87
openPrinter	プリンターとの通信を開始	89
closePrinter	プリンターとの通信を終了	90
sendData	プリンターにコマンドを送信	91
sendData (バッテリーステータス取得用)	プリンターにコマンドを送信	93
setStatusChangeEventCallback	プリンターステータスのコールバックメソッドを登録	95
setOnlineEventCallback	オンラインイベントのコールバックメソッドを登録	96
setOfflineEventCallback	オフラインイベントのコールバックメソッドを登録	97
setPowerOffEventCallback	無応答イベントのコールバックメソッドを登録	98
setCoverOkEventCallback	カバークローズイベントのコールバックメソッドを登録	99
setCoverOpenEventCallback	カバーオープンイベントのコールバックメソッドを登録	100
setPaperOkEventCallback	用紙ありイベントのコールバックメソッドを登録	101
setPaperNearEndEventCallback	用紙残量少イベントのコールバックメソッドを登録	102
setPaperEndEventCallback	用紙なしイベントのコールバックメソッドを登録	103
setDrawerClosedEventCallback	ドロアークローズイベントのコールバックメソッドを登録	104
setDrawerOpenEventCallback	ドロアオープンイベントのコールバックメソッドを登録	105
setBatteryLowEventCallback	バッテリー残量なしイベントの通知先を登録	106
setBatteryOkEventCallback	バッテリー残量ありイベントの通知先を登録	107
setBatteryStatusChangeEventCallback	バッテリーステータスのコールバックメソッドを登録	108

initWithPrinterModel

EposBuilder クラスのインスタンスを初期化します。

構文

```
- (id) initWithPrinterModel:(NSString *)printerModel  
                        Lang:(int)lang;
```

パラメーター

- printerModel: 対象のプリンターの機種名を指定します。

設定値	説明
"TM-T88V"	TM-T88V
"TM-T70"	TM-T70
"TM-T70II"	TM-T70II
"TM-T90II"	TM-T90II
"TM-P60II"	TM-P60II

- lang: プリンターの言語仕様を指定します。

設定値	説明
EPOS_OC_MODEL_ANK	ANK モデル
EPOS_OC_MODEL_JAPANESE	日本語モデル

戻り値

処理に成功した場合、初期化済の EposBuilder クラスインスタンスが返ります。

処理に失敗した場合、nil が返ります。処理に失敗する原因には、以下の要因があります。

- 不正なパラメーターが指定された。
- メモリーを確保できなかった。
- サポートしていない機種名または言語仕様が指定された。

例

TM-T88V 日本語モデル用の命令バッファを初期化する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"  
                Lang: EPOS_OC_MODEL_JAPANESE];  
if ( builder != nil ) {  
    ... 処理 ...  
    [builder release];  
}
```

clearCommandBuffer

EposBuilder クラスの API で使用した命令バッファをクリアします。

構文

– (int) **clearCommandBuffer**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。

例

命令バッファをクリアする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus;
    ...処理...
    errorStatus = [builder clearCommandBuffer];
    ...処理...
    [builder release];
}
```

addTextAlign

位置揃え設定を命令バッファに追加します。



本 API の設定は、バーコード / 2 次元シンボルにも適用されます。



印字モードがページモードで位置揃えを設定する場合、本 API ではなく、[addPagePosition \(75 ページ\)](#) で設定してください。

構文

– (int) **addTextAlign**: (int) align;

パラメーター

- align: 位置揃えを指定します。

設定値	説明
EPOS_OC_ALIGN_LEFT (初期値)	左揃え
EPOS_OC_ALIGN_CENTER	中央揃え
EPOS_OC_ALIGN_RIGHT	右揃え

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

中央揃えに設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextAlign: EPOS_OC_ALIGN_CENTER];
    ... 処理 ...
}
```


addTextLineSpace

改行量設定を命令バッファに追加します。

構文

– (int) **addTextLineSpace**: (long) linespc;

パラメーター

- linespc: 改行量 (ドット単位) を指定します。0 ~ 255 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

改行量を 30 ドットに設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLineSpace: 30];
    ... 処理 ...
}
```

addTextRotate

倒立印字設定を命令バッファに追加します。



本 API の設定は、バーコード / 2 次元シンボルにも適用されます。



印字モードがページモードで倒立印字を設定する場合、本 API ではなく、[addPageDirection \(74 ページ\)](#) で設定してください。

構文

```
- (int) addTextRotate: (int) rotate;
```

パラメーター

- rotate: 倒立印字の有無を指定します。

設定値	説明
EPOS_OC_TRUE	倒立印字を指定
EPOS_OC_FALSE (初期値)	倒立印字を解除

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

倒立印字を設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextRotate: EPOS_OC_TRUE];
    ... 処理 ...
}
```

addText

文字の印字を命令バッファに追加します。



テキストの印字後、テキスト以外を印字する場合、改行または紙送りを実行してください。

構文

```
- (int) addText: (NSString *)data;
```

パラメーター

- data: 印字する文字列を指定します。
水平タブ / 改行は、以下のエスケープシーケンスを使用します。

文字列	説明
\t	水平タブ (HT)
\n	改行 (LF)
\\	バックスラッシュ

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

文字列を追加する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addText: @"Hello,\t"];
    errorStatus = [builder addText: @"World!\n"];
    ... 処理 ...
}
```

addTextLang

言語設定を命令バッファに追加します。本 API で指定された言語情報に従って、[addText \(43 ページ\)](#) で指定された文字列をエンコードします。



本 API は、[addText \(43 ページ\)](#) を呼び出す前に呼び出す API です。

構文

- (int) **addTextLang**: (int) lang;

パラメーター

- lang: 対象言語を指定します。

設定値	説明
EPOS_OC_LANG_EN(初期値)	英語 (ANK 仕様)
EPOS_OC_LANG_JA	日本語

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

日本語に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextLang: EPOS_OC_LANG_JA];
    . . . 処理 . . .
}
```

addTextFont

文字のフォント設定を命令バッファに追加します。

構文

– (int) **addTextFont**: (int) font;

パラメーター

- font: フォントを指定します。

設定値	説明
EPOS_OC_FONT_A (初期値)	フォント A
EPOS_OC_FONT_B	フォント B
EPOS_OC_FONT_C	フォント C

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

フォント B を設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextFont: EPOS_OC_FONT_B];
    ... 処理 ...
}
```

addTextSmooth

スムージング設定を命令バッファに追加します。

構文

– (int) **addTextSmooth**: (int) smooth;

パラメーター

- smooth: スムージングの有無を指定します。

設定値	説明
EPOS_OC_TRUE	スムージングを指定
EPOS_OC_FALSE (初期値)	スムージングを解除

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

スムージングを有効に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
             Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextSmooth: EPOS_OC_TRUE];
    . . . 処理 . . .
}
```

addTextDouble

文字の倍角設定を命令バッファに追加します。

構文

– (int) **addTextDouble**: (int) dw Dh: (int) dh;

パラメーター

- dw: 文字の横倍角を指定します。

設定値	説明
EPOS_OC_TRUE	横倍角を指定
EPOS_OC_FALSE (初期値)	横倍角を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- dh: 文字の縦倍角を指定します。

設定値	説明
EPOS_OC_TRUE	縦倍角を指定
EPOS_OC_FALSE (初期値)	縦倍角を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



dw と dh のパラメーターの両方を EPOS_OC_TRUE にした場合、4 倍角の文字が印字されます。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

4 倍角を設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextDouble: EPOS_OC_TRUE
        Dh: EPOS_OC_TRUE];
    ... 処理 ...
}
```

addTextSize

文字の倍率設定を命令バッファに追加します。

構文

– (int) **addTextSize**: (long)width Height: (long)height;

パラメーター

- width: 文字の横倍率を指定します。

設定値	説明
1 ~ 8 の整数	横方向の倍率を指定 (初期値: 1)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- height: 文字の縦倍率を指定します。

設定値	説明
1 ~ 8 の整数	縦方向の倍率を指定 (初期値: 1)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

横倍率 4 倍、縦倍率 4 倍に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextSize: 4 Height: 4];
    ... 処理 ...
}
```


addTextStyle

文字の装飾設定を命令バッファに追加します。

構文

```
(int) addTextStyle: (int)reverse Ul: (int)ul Em: (int)em  
Color: (int)color;
```

パラメーター

- reverse: 白黒反転文字を指定します。

設定値	説明
EPOS_OC_TRUE	白黒反転文字を指定
EPOS_OC_FALSE (初期値)	白黒反転文字を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- ul: アンダーラインを指定します。

設定値	説明
EPOS_OC_TRUE	アンダーラインを指定
EPOS_OC_FALSE (初期値)	アンダーラインを解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- em: 太字を指定します。

設定値	説明
EPOS_OC_TRUE	太字を指定
EPOS_OC_FALSE (初期値)	太字を解除
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- color: 色を指定します。

設定値	説明
EPOS_OC_COLOR_NONE	非印字
EPOS_OC_COLOR_1 (初期値)	第 1 色
EPOS_OC_COLOR_2	第 2 色
EPOS_OC_COLOR_3	第 3 色
EPOS_OC_COLOR_4	第 4 色
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

アンダーラインを設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextStyle: EPOS_OC_PARAM_UNSPECIFIED
        U1: EPOS_OC_TRUE Em: EPOS_OC_PARAM_UNSPECIFIED
        Color: EPOS_OC_PARAM_UNSPECIFIED];
    . . . 処理 . . .
}
```

addTextPosition

横方向の印字開始位置を命令バッファーに追加します。

構文

– (int) **addTextPosition**: (long) x;

パラメーター

- x: 横方向の印字開始位置（ドット単位）を指定します。
0 ～ 65535 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

印字位置を左端から 120 ドットの位置に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addTextPosition: 120];
    . . . 処理 . . .
}
```

addFeedUnit

ドット単位の紙送りを命令バッファに追加します。

構文

– (int) **addFeedUnit**: (long) unit;

パラメーター

- unit: 紙送り量 (ドット単位) を指定します。0 ~ 255 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

紙送りを 30 ドットする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedUnit: 30];
    ... 処理 ...
}
```

addFeedLine

行単位の紙送りを命令バッファーに追加します。

構文

– (int) **addFeedLine**: (long) line;

パラメーター

- unit: 紙送り量（行単位）を指定します。0 ～ 255 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

紙送りを 3 行する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedLine: 3];
    ... 処理 ...
}
```

addImage (多階調印字用)

ラスターイメージの印字を命令バッファーに追加します。

UIImage クラスのグラフィックを印字します。

UIImage クラスのグラフィックうち、指定範囲を本 API の設定にしたがって、ラスターイメージデータに変換します。画像の 1 ピクセルがプリンターの 1 ドットに相当します。透明色が含まれている場合、画像の背景を白とみなします。



ラスターイメージを高速に印字する場合、[addTextAlign \(40 ページ\)](#) を EPOS_OC_ALIGN_LEFT に指定し、本 API の width パラメーターの値をプリンターの用紙幅を超えない 8 の倍数に指定してください。

構文

```
(int) addImage: (UIImage *)data X:(long)x Y:(long)y  
Width:(long)width Height:(long)height  
Color:(int)color Mode:(int)mode  
Halftone:(int)halftone  
Brightness:(double)brightness;
```

パラメーター

- data : UIImage クラスのインスタンスを指定します。
- x : 印字範囲の横方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- y : 印字範囲の縦方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- width : 印字範囲の幅 (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。
- height : 印字範囲の高さ (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。
- color : 色を指定します。

設定値	説明
EPOS_OC_COLOR_NONE	非印字
EPOS_OC_COLOR_1	第 1 色
EPOS_OC_COLOR_2	第 2 色
EPOS_OC_COLOR_3	第 3 色
EPOS_OC_COLOR_4	第 4 色
EPOS_OC_PARAM_DEFAULT	既定値 (第 1 色) を選択



x/y パラメーターと width/height パラメーターで指定された領域が data パラメーターで指定した画像のサイズに収まっていない場合、戻り値に EPOS_OC_ERR_PARAM が返されます。

- mode : カラーモードを指定します。

設定値	説明	TM プリンター別設定値				
		TM-T88V	TM-T70	TM-T70II	TM-T90II	TM-P60II
EPOS_OC_MODE_MONO	モノクロ (2 階調)	○	○	○	○	○
EPOS_OC_MODE_GRAY16	多階調 (16 階調)	○	-	○	○	-
EPOS_OC_PARAM_DEFAULT	既定値を選択 (モノクロ (2 階調))	○	○	○	○	○

- halftone : ハーフトーン処理方法を指定します。

設定値	説明
EPOS_OC_HALFTONE_DITHER	ディザー (グラフィックの印刷に適しています。)
EPOS_OC_HALFTONE_ERROR_DIFFUSION	誤差拡散 (文字とグラフィックが混在する印刷に適しています。)
EPOS_OC_HALFTONE_THRESHOLD	しきい値 (文字の印刷に適しています。)
EPOS_OC_PARAM_DEFAULT	既定値 (ディザー) を選択



多階調 (16 階調) の場合、無視されます。

- brightness : 明るさの補正値を指定します。

設定値	説明
0.1 ~ 10.0 の実数	明るさ補正値 (ガンマー値)
Builder.PARAM_DEFAULT	既定値 (1.0) を選択



1.0 以外を指定した場合、印字速度が遅くなります。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
Height: 256 Color: EPOS_OC_PARAM_DEFAULT Mode: EPOS_OC_MODE_MONO
Halftone: EPOS_OC_HALFTONE_DITHER Brightness: 1.0];
    . . . 処理 . . .
}
```


ページモードで幅 256 ドット、高さ 256 ドットの画像を印字する

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPagePosition: 0 Y: 255];
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
Height: 256 Color: EPOS_OC_PARAM_DEFAULT Mode: EPOS_OC_MODE_MONO
Halftone: EPOS_OC_HALFTONE_DITHER Brightness: 1.0];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}
```


addImage

ラスターイメージの印字を命令バッファに追加します。
UIImage クラスのグラフィックを印字します。
UIImage クラスのグラフィックうち、指定範囲をディザ処理で二値化し、ラスターイメージデータに変換します。
画像の 1 ピクセルがプリンターの 1 ドットに相当します。透明色が含まれている場合、画像の背景を白とみなします。



- 多階調で印字する場合、[addImage\(多階調印字用 \) \(54 ページ\)](#) を使用してください。
- ラスターイメージを高速に印字する場合、[addTextAlign \(40 ページ\)](#) を EPOS_OC_ALIGN_LEFT に指定し、本 API の width パラメーターの値をプリンターの用紙幅を超えない 8 の倍数に指定してください。


構文

```
(int) addImage:(UIImage *)data X:(long)x Y:(long)y  
Width:(long)width Height:(long)height  
Color:(int)color;
```

パラメーター

- data : UIImage クラスのインスタンスを指定します。
- x : 印字範囲の横方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- y : 印字範囲の縦方向の開始位置 (ピクセル単位) を指定します。
0 ~ 65534 の整数値で指定します。
- width : 印字範囲の幅 (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。
- height : 印字範囲の高さ (ピクセル単位) を指定します。1 ~ 65535 の整数値で指定します。
- color : 色を指定します。

設定値	説明
EPOS_OC_COLOR_NONE	非印字
EPOS_OC_COLOR_1	第 1 色
EPOS_OC_COLOR_2	第 2 色
EPOS_OC_COLOR_3	第 3 色
EPOS_OC_COLOR_4	第 4 色
EPOS_OC_PARAM_DEFAULT	既定値 (第 1 色) を選択



x/y パラメーターと width/height パラメーターで指定された領域が data パラメーターで指定した画像のサイズに収まっていない場合、戻り値に EPOS_OC_ERR_PARAM が返されます。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
    Height: 256 Color: EPOS_OC_PARAM_DEFAULT];
    . . . 処理 . . .
}
```


ページモードで幅 256 ドット、高さ 256 ドットの画像を印字する

```
UIImage * imageData = Nil;

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    . . . 処理 . . .
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPagePosition: 0 Y: 255];
    errorStatus = [builder addImage: imageData X: 0 Y: 0 Width: 256
    Height: 256 Color: EPOS_OC_PARAM_DEFAULT];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}
```

addLogo

NV ログの印字を命令バッファに追加します。プリンターの NV メモリーに登録されているロゴを印字します。



ロゴは以下のユーティリティを使って、あらかじめプリンターにロゴの登録します。

- 機種専用ユーティリティ
- ロゴ登録ユーティリティ (TMFLogo)

構文

```
- (int) addLogo: (long)key1 Key2: (long)key2;
```

パラメーター

- key1: NV ログのキーコード 1 を指定します。32 ~ 126 の整数値で指定します。
- key2: NV ログのキーコード 2 を指定します。32 ~ 126 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

キーコード 48,48 の NV ログを印字する

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addLogo: 48 Key2: 48];
    ... 処理 ...
}
```

addBarcode

バーコード印字を命令バッファに追加します。

構文

```
(int) addBarcode: (NSString *)data Type:(int)type  
Hri:(int)hri Font:(int)font  
Width:(long)width  
Height:(long)height;
```

パラメーター

- data: バーコードデータを文字列で指定します。



type で指定するバーコードの規格に従った文字列を指定してください。規格に従っていない場合、バーコードは印刷されません。

種類	説明
UPC-A	11 桁の数字を指定した場合、チェックデジットを自動的に付加します。 12 桁の数字を指定した場合、12 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
UPC-E	最初の桁に 0 を指定してください。 2 ～ 6 桁目にメーカーコードを指定してください。 7 ～ 11 桁目にアイテムコードを右詰めで指定してください。アイテムコードの桁数はメーカーコードにより異なります。使用しない桁は 0 を指定してください。 11 桁の数字を指定した場合、チェックデジットを自動的に付加します。 12 桁の数字を指定した場合、12 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
EAN13	12 桁の数字を指定した場合、チェックデジットを自動的に付加します。
JAN13	13 桁の数字を指定した場合、13 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
EAN8	7 桁の数字を指定した場合、チェックデジットを自動的に付加します。
JAN8	8 桁の数字を指定した場合、8 桁目をチェックデジットとして処理しますが、チェックデジットの検算は行いません。
CODE39	先頭の文字が * の場合、この文字をスタートキャラクターとして処理します。それ以外の場合、スタートキャラクターを自動的に付加します。
ITF	スタートコードおよびストップコードを自動的に付加します。 チェックデジットの付加および検算は行いません。
CODABAR	スタートキャラクター (A ～ D, a ～ d) を指定してください。 ストップキャラクター (A ～ D, a ～ d) を指定してください。 チェックデジットの付加および検算は行いません。

種類	説明																		
CODE93	<p>スタートキャラクターおよびストップキャラクターを自動的に付加します。</p> <p>チェックデジットを計算して自動的に付加します。</p>																		
CODE128	<p>スタートキャラクター (CODE A, CODE B, CODE C) を指定してください。</p> <p>ストップキャラクターを自動的に付加します。</p> <p>チェックデジットを計算して自動的に付加します。</p> <p>以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。</p> <table> <tr><td>FNC1:</td><td>{1</td></tr> <tr><td>FNC2:</td><td>{2</td></tr> <tr><td>FNC3:</td><td>{3</td></tr> <tr><td>FNC4:</td><td>{4</td></tr> <tr><td>CODE A:</td><td>{A</td></tr> <tr><td>CODE B:</td><td>{B</td></tr> <tr><td>CODE C:</td><td>{C</td></tr> <tr><td>SHIFT:</td><td>{S</td></tr> <tr><td>{:</td><td>{{</td></tr> </table>	FNC1:	{1	FNC2:	{2	FNC3:	{3	FNC4:	{4	CODE A:	{A	CODE B:	{B	CODE C:	{C	SHIFT:	{S	{:	{{
FNC1:	{1																		
FNC2:	{2																		
FNC3:	{3																		
FNC4:	{4																		
CODE A:	{A																		
CODE B:	{B																		
CODE C:	{C																		
SHIFT:	{S																		
{:	{{																		
GS1-128	<p>スタートキャラクター、FNC1、チェックデジット、ストップキャラクターを自動的に付加します。</p> <p>アプリケーション識別子 (AI) とそれに続くデータのチェックデジットを計算して自動的に付加するには、チェックデジットの位置に文字 * を指定します。</p> <p>アプリケーション識別子 (AI) を括弧で囲むことができます。括弧は HRI の印字文字として使用し、データとしてエンコードしません。</p> <p>アプリケーション識別子 (AI) とデータの間に空白を挿入することができます。空白は HRI の印字文字として使用し、データとしてエンコードしません。</p> <p>以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。</p> <table> <tr><td>FNC1:</td><td>{1</td></tr> <tr><td>FNC3:</td><td>{3</td></tr> <tr><td>(:</td><td>{{</td></tr> <tr><td>):</td><td>{}</td></tr> <tr><td>*:</td><td>{*</td></tr> <tr><td>{:</td><td>{{</td></tr> </table>	FNC1:	{1	FNC3:	{3	(:	{{):	{}	*:	{*	{:	{{						
FNC1:	{1																		
FNC3:	{3																		
(:	{{																		
):	{}																		
:	{																		
{:	{{																		
GS1 DataBar Omnidirectional	アプリケーション識別子 (AI) とチェックデジットを除く 13 桁の商品識別番号 (GTIN) を指定してください。																		
GS1 DataBar Truncated																			
GS1 DataBar Limited																			

種類	説明
BARCODE_GS1_ DATABAR_EXPANDED	アプリケーション識別子 (AI) を括弧で囲むことができます。括弧は HRI の印字文字として使用し、データとしてエンコードしません。 以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。 FNC1: {1 (: {(): }

文字列で表現できないバイナリーデータを指定する場合、以下のエスケープシーケンスで指定します。

文字列	説明
\xnn	コントロールコード
\\	バックスラッシュ

- type : バーコードの種類を指定します。

設定値	説明
EPOS_OC_BARCODE_UPC_A	UPC-A
EPOS_OC_BARCODE_UPC_E	UPC-E
EPOS_OC_BARCODE_EAN13	EAN13
EPOS_OC_BARCODE_JAN13	JAN13
EPOS_OC_BARCODE_EAN8	EAN8
EPOS_OC_BARCODE_JAN8	JAN8
EPOS_OC_BARCODE_CODE39	CODE39
EPOS_OC_BARCODE_ITF	ITF
EPOS_OC_BARCODE_CODABAR	CODABAR
EPOS_OC_BARCODE_CODE93	CODE93
EPOS_OC_BARCODE_CODE128	CODE128
EPOS_OC_BARCODE_GS1_128	GS1-128
EPOS_OC_BARCODE_GS1_ DATABAR_OMNIDIRECTIONAL	GS1 DataBar Omnidirectional
EPOS_OC_BARCODE_GS1_DATABAR_TRUNCATED	GS1 DataBar Truncated
EPOS_OC_BARCODE_GS1_DATABAR_LIMITED	GS1 DataBar Limited
EPOS_OC_BARCODE_GS1_DATABAR_EXPANDED	GS1 Databar Expanded

- hri : HRI の位置を指定します。

設定値	説明
EPOS_OC_HRI_NONE (初期値)	印字しない
EPOS_OC_HRI_ABOVE	バーコードの上
EPOS_OC_HRI_BELOW	バーコードの下
EPOS_OC_HRI_BOTH	バーコードの上と下の両方
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- font : HRI フォントを指定します。

設定値	説明
EPOS_OC_FONT_A (初期値)	フォント A
EPOS_OC_FONT_B	フォント B
EPOS_OC_FONT_C	フォント C
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- width : 1 モジュールの幅をドット単位で指定します。

設定値	説明
2 ～ 6 の整数値	1 モジュールの幅 (ドット単位)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

- height : バーコードの高さをドット単位で指定します。1 ～ 255 の整数値で指定します。

設定値	説明
1 ～ 255 の整数値	バーコードの高さ (ドット単位)
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

各種バーコードを印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addBarcode: @"01234567890"
        Type: EPOS_OC_BARCODE_UPC_A Hri: EPOS_OC_HRI_BELOW
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: 2 Height: 64];
    errorStatus = [builder addBarcode: @"01234500005"
        Type: EPOS_OC_BARCODE_UPC_E Hri: EPOS_OC_PARAM_UNSPECIFIED
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
        Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"201234567890"
        Type: EPOS_OC_BARCODE_EAN13 Hri: EPOS_OC_PARAM_UNSPECIFIED
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
        Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"201234567890"
        Type: EPOS_OC_BARCODE_JAN13 Hri: EPOS_OC_PARAM_UNSPECIFIED
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
        Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"2012345" Type: EPOS_OC_BARCODE_EAN8
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"2012345" Type: EPOS_OC_BARCODE_JAN8
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"ABCDE" Type: EPOS_OC_BARCODE_CODE39
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"012345" Type: EPOS_OC_BARCODE_ITF
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"A012345A"
        Type: EPOS_OC_BARCODE_CODABAR Hri: EPOS_OC_PARAM_UNSPECIFIED
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
        Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"ABCDE" Type: EPOS_OC_BARCODE_CODE93
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"{Babcde"
        Type: EPOS_OC_BARCODE_CODE128 Hri: EPOS_OC_PARAM_UNSPECIFIED
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
        Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"(01)201234567890*"
        Type: EPOS_OC_BARCODE_GS1_128 Hri: EPOS_OC_PARAM_UNSPECIFIED
        Font: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
        Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_OMNIDIRECTIONAL
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_TRUNCATED
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"0201234567890"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_LIMITED
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addBarcode: @"(01)2012345678903"
        Type: EPOS_OC_BARCODE_GS1_DATABAR_EXPANDED
        Hri: EPOS_OC_PARAM_UNSPECIFIED Font: EPOS_OC_PARAM_UNSPECIFIED
        Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED];
    ... 処理 ...
}
```


addSymbol


2 次元シンボル印字を命令バッファに追加します。

構文

```
(int) addSymbol:(NSString *)data Type:(int)type
      Level:(int)level Width:(long)width
      Height:(long)height Size:(long)size;
```

パラメーター

- data : 2 次元シンボルデータを文字列で指定します。



type で指定する 2 次元シンボルの規格に従った文字列を指定してください。規格に従っていない場合、2 次元シンボルは印刷されません。

文字列	説明
Standard PDF417	文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。 データ領域の最大コードワード数は 928 個、1 段あたりのデータ領域の最大コードワード数は 30 個、最大段数は 90 段です。
Truncated PDF417	
QR Code Model 1	文字列をシフト JIS に変換後、エスケープシーケンスの処理を行い、データの種類を以下の中から選択してエンコードします。 数字： 0 ～ 9 英数字： 0 ～ 9, A ～ Z, スペース, \$, %, *, +, -, ., /, : 漢字： シフト JIS 値 8 ビットバイトデータ： 0x00 ～ 0xff
QR Code Model 2	

文字列	説明
MaxiCode Mode 2	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>モード 2 およびモード 3 の場合、最初のデータが 0>\x1e01\x1dyy (yy は 2 桁の数字) の場合、これをメッセージヘッダーとして処理し、次のデータからプライマリメッセージとして処理します。それ以外の場合、最初のデータからプライマリメッセージとして処理します。</p> <p>モード 2 の場合、プライマリメッセージを以下の形式で指定してください。</p> <p>郵便コード (1 ~ 9 桁の数字) GS:(\x1d) ISO 国名コード (1 ~ 3 桁の数字) GS:(\x1d) サービスクラスコード (1 ~ 3 桁の数字)</p> <p>モード 3 の場合、プライマリメッセージを以下の形式で指定してください。</p> <p>郵便コード (1 ~ 6 個のコードセット A で変換可能なデータ) GS(\x1d) ISO 国名コード (1 ~ 3 桁の数字) GS(\x1d) サービスクラスコード (1 ~ 3 桁の数字)</p>
MaxiCode Mode 3	
MaxiCode Mode 4	
MaxiCode Mode 5	
MaxiCode Mode 6	
GS1 DataBar Stacked	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>アプリケーション識別子 (AI) とチェックデジットを除く 13 桁の商品識別番号 (GTIN) を指定してください。</p>
GS1 DataBar Stacked Omnidirectional	
GS1 DataBar Expanded Stacked	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>アプリケーション識別子 (AI) を括弧で囲むことができます。括弧は HRI の印字文字として使用し、データとしてエンコードしません。</p> <p>以下の文字をエンコードするには、文字 { で始まる 2 文字を指定してください。</p> <p>FNC1: {1 (: {(): }</p>
Aztec Code Full-Range モード	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>最大でテキスト 3067 文字、数字 3832 文字、バイナリーデータ 1914 バイトを指定できます。</p>
Aztec Code Compact モード	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>最大でテキスト 89 文字、数字 110 文字、バイナリーデータ 53 バイトを指定できます。</p>
DataMatrix 正方形	<p>文字列を UTF-8 に変換後、エスケープシーケンスの処理を行い、エンコードします。</p> <p>シンボルは 10 行 x10 列 ~ 144 行 x144 列の正方形、または行数 8、行数 12、行数 16 の長方形です。</p> <p>最大で英数字 2335 文字、数字 3116 文字、バイナリーデータ 1556 バイトを指定できます。</p>
DataMatrix 長方形、行数 8	
DataMatrix 長方形、行数 12	
DataMatrix 長方形、行数 16	

文字列で表現できないバイナリーデータを指定する場合、以下のエスケープシーケンスで指定します。

文字列	説明
\xnn	コントロールコード
\\	バックスラッシュ

- type : 2次元シンボルの種類を指定します。

設定値	種類
EPOS_OC_SYMBOL_PDF417_STANDARD	Standard PDF417
EPOS_OC_SYMBOL_PDF417_TRUNCATED	Truncated PDF417
EPOS_OC_SYMBOL_QRCODE_MODEL_1	QR Code Model 1
EPOS_OC_SYMBOL_QRCODE_MODEL_2	QR Code Model 2
EPOS_OC_SYMBOL_MAXICODE_MODE_2	MaxiCode Mode 2
EPOS_OC_SYMBOL_MAXICODE_MODE_3	MaxiCode Mode 3
EPOS_OC_SYMBOL_MAXICODE_MODE_4	MaxiCode Mode 4
EPOS_OC_SYMBOL_MAXICODE_MODE_5	MaxiCode Mode 5
EPOS_OC_SYMBOL_MAXICODE_MODE_6	MaxiCode Mode 6
EPOS_OC_SYMBOL_GS1_DATABAR_STACKED	GS1 DataBar Stacked
EPOS_OC_SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL	GS1 DataBar Stacked Omnidirectional
EPOS_OC_SYMBOL_GS1_DATABAR_EXPANDED_STACKED	GS1 DataBar Expanded Stacked
EPOS_OC_SYMBOL_AZTECCODE_FULLRANGE	Aztec Code Full-Range モード
EPOS_OC_SYMBOL_AZTECCODE_COMPACT	Aztec Code Compact モード
EPOS_OC_SYMBOL_DATAMATRIX_SQUARE	DataMatrix 正方形
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_8	DataMatrix 長方形、行数 8
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_12	DataMatrix 長方形、行数 12
EPOS_OC_SYMBOL_DATAMATRIX_RECTANGLE_16	DataMatrix 長方形、行数 16

- level : エラー訂正レベルを指定します。

設定値	説明
EPOS_OC_LEVEL_0	PDF417 エラー訂正レベル 0
EPOS_OC_LEVEL_1	PDF417 エラー訂正レベル 1
EPOS_OC_LEVEL_2	PDF417 エラー訂正レベル 2
EPOS_OC_LEVEL_3	PDF417 エラー訂正レベル 3
EPOS_OC_LEVEL_4	PDF417 エラー訂正レベル 4
EPOS_OC_LEVEL_5	PDF417 エラー訂正レベル 5
EPOS_OC_LEVEL_6	PDF417 エラー訂正レベル 6
EPOS_OC_LEVEL_7	PDF417 エラー訂正レベル 7
EPOS_OC_LEVEL_8	PDF417 エラー訂正レベル 8
EPOS_OC_LEVEL_L	QR Code エラー訂正レベル L
EPOS_OC_LEVEL_M	QR Code エラー訂正レベル M
EPOS_OC_LEVEL_Q	QR Code エラー訂正レベル Q

設定値	説明
EPOS_OC_LEVEL_H	QR Code エラー訂正レベル H
5 ~ 95 の整数	Aztec Code エラー訂正レベル (パーセント単位)
EPOS_OC_LEVEL_DEFAULT	既定レベル
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



- 2次元シンボルの種類に合わせて選択してください。
- MaxiCode/2次元GS1 DataBar//DataMatrixの場合、EPOS_OC_LEVEL_DEFAULTを選択してください。

- width : モジュールの幅を指定します。

設定値	説明
1 ~ 255 の整数値	モジュールの幅
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



MaxiCode は無視されます。

- height : モジュールの高さを指定します。

設定値	説明
1 ~ 255 の整数値	モジュールの高さ
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



QR Code/MaxiCode/2次元GS1 DataBar/Aztec Code/DataMatrix は無視されます。

- size : 2次元シンボルの最大サイズを指定します。

設定値	説明
0 ~ 65535 の整数値	2次元シンボルの最大サイズ
EPOS_OC_PARAM_UNSPECIFIED	設定を変更しない



QR Code/MaxiCode/Aztec Code/DataMatrix は無視されます。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

各種2次元シンボルを印字する場合

```

id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSymbol: @"ABCDE"
                               Type: EPOS_OC_SYMBOL_PDF417_STANDARD Level:
                               EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                               Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"ABCDE"
                               Type: EPOS_OC_SYMBOL_QRCODE_MODEL_2 Level: EPOS_OC_LEVEL_Q
                               Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED
                               Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"908063840\\x1d850\\x1d001\\x1d\\x04"
                               Type: EPOS_OC_SYMBOL_MAXICODE_MODE_2 Level: EPOS_OC_PARAM_UNSPECIFIED
                               Width: EPOS_OC_PARAM_UNSPECIFIED Height: EPOS_OC_PARAM_UNSPECIFIED
                               Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"0201234567890"
                               Type: EPOS_OC_SYMBOL_GS1_DATABAR_STACKED
                               Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                               Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"0201234567890"
                               Type: EPOS_OC_SYMBOL_GS1_DATABAR_STACKED_OMNIDIRECTIONAL
                               Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                               Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    errorStatus = [builder addSymbol: @"(01)02012345678903"
                               Type: EPOS_OC_SYMBOL_GS1_DATABAR_EXPANDED_STACKED
                               Level: EPOS_OC_PARAM_UNSPECIFIED Width: EPOS_OC_PARAM_UNSPECIFIED
                               Height: EPOS_OC_PARAM_UNSPECIFIED Size: EPOS_OC_PARAM_UNSPECIFIED];
    ... 処理 ...
}

```

addPageBegin

ページモード開始を命令バッファに追加します。ページモードの処理が開始します。



本 API は [addPageEnd \(71 ページ\)](#) と一緒にお使いください。

構文

- (int) **addPageBegin**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

ページモードで文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}
```

addPageEnd

ページモード終了を命令バッファに追加します。ページモードの処理が終了します。



本 API は [addPageBegin \(70 ページ\)](#) と一緒にお使いください。

構文

- (int) **addPageEnd**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

ページモードで文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}
```

addPageArea

ページモード印字領域を命令バッファに追加します。

ページモード印字領域（座標）を指定します。本 API に続けて、addText など印刷データの API を指定します。



印字内容に合わせて印字領域を指定してください。印字データが印字領域をはみ出した場合、印字データが途中で切れた印字結果になります。



本 API は [addPageBegin \(70 ページ\)](#) と [addPageEnd \(71 ページ\)](#) に挟んでお使いください。

構文

```
(int) addPageArea: (long)x Y: (long)y Width: (long)width  
Height: (long)height;
```

パラメーター

- x: 横方向の原点（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。0 はプリンターの印字可能領域の左端になります。
- y: 縦方向の原点（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。0 は紙送りをしていない位置です。
- width: 印字領域の幅（ドット単位）を指定します。1 ～ 65535 の整数値で指定します。
- height: 印字領域の高さ（ドット単位）を指定します。1 ～ 65535 の整数値で指定します。



印字領域の幅と高さは、印字方向の設定に合わせて確定してください。
印字データが切れてしまう場合があります。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

原点 (100, 50)、幅 200 ドット、高さ 30 ドットの印字領域を指定して、文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 30];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    . . . 処理 . . .
}
```

addPageDirection

ページモード印字方向設定を命令バッファに追加します。ページモードの印字方向を指定します。回転させない場合は、省略できます。



本 API は [addPageBegin \(70 ページ\)](#) と [addPageEnd \(71 ページ\)](#) に挟んでお使いください。

構文

– (int) **addPageDirection**: (int) dir;

パラメーター

- dir: ページモードの印字方向を指定します。

設定値	説明
EPOS_OC_DIRECTION_LEFT_TO_RIGHT (初期値)	回転しない (左上を始点に右方向へ印字)
EPOS_OC_DIRECTION_BOTTOM_TO_TOP	反時計回り 90 度回転 (左下を始点に上方向へ印字)
EPOS_OC_DIRECTION_RIGHT_TO_LEFT	180 度回転 (右下を始点に左方向へ印字)
EPOS_OC_DIRECTION_TOP_TO_BOTTOM	時計回り 90 度回転 (右上を始点に下方向へ印字)

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

時計回りに 90 度回転させて、文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
               Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 30 Height: 200];
    errorStatus = [builder addPageDirection:
                  EPOS_OC_DIRECTION_TOP_TO_BOTTOM];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ... 処理 ...
}
```

addPagePosition

ページモードの印字位置設定領域を命令バッファに追加します。

addPageArea で指定したエリア内での、印字開始位置（座標）を指定します。



本 API は [addPageBegin \(70 ページ\)](#) と [addPageEnd \(71 ページ\)](#) に挟んでお使いください。

構文

```
(int) addPagePosition: (long)x Y: (long)y;
```

パラメーター

- x: 横方向の印字位置（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。
- y: 縦方向の印字位置（ドット単位）を指定します。0 ～ 65535 の整数値で指定します。



印字開始位置（座標）は、印字内容に合わせて指定してください。以下を参考にしてください。

- * 文字列を印字する場合
最初の文字のベースライン左端を指定します。
標準の大きさで左詰めで印字する場合は省略可能です。高さが 2 倍の文字を印刷する場合は、y を 42 以上に指定します。
- * バーコードを印字する場合
シンボルの左下を指定します。y にバーコードの高さを指定してください。
- * グラフィック / ロゴを印字する場合
グラフィックデータの左下を指定します。y にグラフィックデータの高さを指定してください。
- * 2 次元シンボルを印字する場合
シンボルの左上を指定します。左上から印字する場合は、省略可能です。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

addPageArea で指定したエリア内の印字開始位置を (50, 30) に指定して、文字「ABCDE」を印字する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPageBegin];
    errorStatus = [builder addPageArea: 100 Y: 50 Width: 200 Height: 100];
    errorStatus = [builder addPagePosition: 50 Y: 30];
    errorStatus = [builder addText: @"ABCDE"];
    errorStatus = [builder addPageEnd];
    ... 処理 ...
}
```

addCut

用紙カットを命令バッファに追加します。用紙カットを設定します。



ページモードでは使用できません。

構文

- (int) **addCut**: (int) type;

パラメーター

- type: 用紙カット方法を指定します。

設定値	説明
EPOS_OC_CUT_NO_FEED	フィード無しカット (紙送りせずにカット)
EPOS_OC_CUT_FEED	フィードカット (紙送り後カット)
EPOS_OC_CUT_RESERVE	カット予約 (後に続く印字を実行後、カット位置でカット)
EPOS_OC_PARAM_DEFAULT	フィードカット (紙送り後カット)

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

フィードカットする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
             Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addCut: EPOS_OC_CUT_FEED];
    ... 処理 ...
}
```

addPulse

ドロアーキックを命令バッファーに追加します。ドロアーキックを設定します。



- ページモードでは使用できません。
- ドロアーは、ブザーと一緒に使用できません。

構文

- (int) **addPulse**: (int) drawer Time: (int) time;

パラメーター

- drawer : ドロアーキックコネクタを指定します。

設定値	説明
EPOS_OC_DRAWER_1	ドロアーキックコネクタ 2 番ピン
EPOS_OC_DRAWER_2	ドロアーキックコネクタ 5 番ピン
EPOS_OC_PARAM_DEFAULT	ドロアーキックコネクタ 2 番ピン

- time : ドロアーキック信号のオン時間を指定します。

設定値	説明
EPOS_OC_PULSE_100	100 ミリ秒の信号
EPOS_OC_PULSE_200	200 ミリ秒の信号
EPOS_OC_PULSE_300	300 ミリ秒の信号
EPOS_OC_PULSE_400	400 ミリ秒の信号
EPOS_OC_PULSE_500	500 ミリ秒の信号
EPOS_OC_PARAM_DEFAULT	100 ミリ秒の信号

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

ドロアーキックコネクタ 2 番ピンに 100 ミリ秒のパルス信号を出力する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addPulse: EPOS_OC_DRAWER_1 Time: EPOS_OC_PULSE_100];
    ... 処理 ...
}
```

addSound (周期鳴動設定用)

ブザーの鳴動を命令バッファに追加します。ブザーを設定します。



- ページモードでは使用できません。
- ブザーの機能は、ドロアーと一緒に使用できません。
- 本 API はプリンターにブザーが付いてなければ使用できません。

構文

```
(int) addSound: (int)pattern Repeat: (long)repeat  
Cycle: (long)cycle;
```

パラメーター

- pattern : ブザーの音色を指定します。

設定値	説明
EPOS_OC_PATTERN_A	パターン A
EPOS_OC_PATTERN_B	パターン B
EPOS_OC_PATTERN_C	パターン C
EPOS_OC_PATTERN_D	パターン D
EPOS_OC_PATTERN_E	パターン E
EPOS_OC_PATTERN_ERROR	エラー鳴動パターン
EPOS_OC_PATTERN_PAPER_END	紙無し鳴動パターン
EPOS_OC_PATTERN_1	パターン 1
EPOS_OC_PATTERN_2	パターン 2
EPOS_OC_PATTERN_3	パターン 3
EPOS_OC_PATTERN_4	パターン 4
EPOS_OC_PATTERN_5	パターン 5
EPOS_OC_PATTERN_6	パターン 6
EPOS_OC_PATTERN_7	パターン 7
EPOS_OC_PATTERN_8	パターン 8
EPOS_OC_PATTERN_9	パターン 9
EPOS_OC_PATTERN_10	パターン 10
EPOS_OC_PARAM_DEFAULT	パターン A

- repeat : 繰り返し回数を指定します。

設定値	説明
1 ~ 255	1 ~ 255 回
EPOS_OC_PARAM_DEFAULT	1 回

- cycle : ブザーを鳴らす周期（ミリ秒単位）を指定します。

設定値	説明
1000 ~ 25500	1000 ~ 25500 ミリ秒
EPOS_OC_PARAM_DEFAULT	1000 ミリ秒



パターン A ~ E/ エラー鳴動パターン / 紙無し鳴動パターンは無視されます。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

パターン 1 を 1000 ミリ秒周期で 3 回鳴らす場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSound: EPOS_OC_PATTERN_1 Repeat: 3 Cycle: 1000];
    ... 処理 ...
}
```

addSound

ブザーの鳴動を命令バッファに追加します。ブザーを設定します。



ブザーを鳴らす周期（ミリ秒単位）を任意で設定したい場合、`addSound(周期鳴動設定用)` (78 ページ) を使用してください。



- ページモードでは使用できません。
- ブザーの機能は、ドロアーと一緒に使用できません。
- 本 API はプリンターにブザーが付いてなければ使用できません。

構文

```
- (int) addSound: (int)pattern Repeat: (long)repeat;
```

パラメーター

- pattern : ブザーの音色を指定します。

設定値	説明
EPOS_OC_PATTERN_A	パターン A
EPOS_OC_PATTERN_B	パターン B
EPOS_OC_PATTERN_C	パターン C
EPOS_OC_PATTERN_D	パターン D
EPOS_OC_PATTERN_E	パターン E
EPOS_OC_PATTERN_ERROR	エラー鳴動パターン
EPOS_OC_PATTERN_PAPER_END	紙無し鳴動パターン
EPOS_OC_PATTERN_1	パターン 1
EPOS_OC_PATTERN_2	パターン 2
EPOS_OC_PATTERN_3	パターン 3
EPOS_OC_PATTERN_4	パターン 4
EPOS_OC_PATTERN_5	パターン 5
EPOS_OC_PATTERN_6	パターン 6
EPOS_OC_PATTERN_7	パターン 7
EPOS_OC_PATTERN_8	パターン 8
EPOS_OC_PATTERN_9	パターン 9
EPOS_OC_PATTERN_10	パターン 10
EPOS_OC_PARAM_DEFAULT	パターン A

- repeat : 繰り返し回数を指定します。

設定値	説明
1 ~ 255	1 ~ 255 回
EPOS_OC_PARAM_DEFAULT	1 回

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

パターン A を 3 回鳴らす場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addSound: EPOS_OC_PATTERN_A Repeat: 3];
    . . . 処理 . . .
}
```

addFeedPosition

ラベル / ブラックマーク紙の紙送りを命令バッファーに追加します。

構文

```
- (int) addFeedPosition: (int) position;
```

パラメーター

- position: 紙送りする位置を指定します。

設定値	説明
EPOS_OC_FEED_PEELING	剥離位置まで紙送り
EPOS_OC_FEED_CUTTING	カット位置まで紙送り
EPOS_OC_FEED_CURRENT_TOF	現在のラベル頭出し位置まで紙送り
EPOS_OC_FEED_NEXT_TOF	次のラベル頭出し位置まで紙送り

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

ラベル紙を剥離位置まで紙送りする場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addFeedPosition: EPOS_OC_FEED_PEELING];
    ... 処理 ...
}
```

addLayout

ラベル / ブラックマーク紙の用紙レイアウト情報を命令バッファに追加します。

構文

```
- (int) addLayout: (int) type
                        Width: (long) width Height: (long) height
                        MarginTop: (long) marginTop
                        MarginBottom: (long) marginBottom
                        OffsetCut: (long) offsetCut
                        OffsetLabel: (long) offsetLabel;
```

パラメーター

- type: 用紙種類を指定します。

設定値	説明
EPOS_OC_LAYOUT_RECEIPT	レシート紙 (ブラックマークなし)
EPOS_OC_LAYOUT_LABEL	ラベル紙 (ブラックマークなし)
EPOS_OC_LAYOUT_LABEL_BM	ラベル紙 (ブラックマークあり)
EPOS_OC_LAYOUT_RECEIPT_BM	レシート紙 (ブラックマークあり)

- width: 用紙幅 (0.1mm 単位) を指定します。1 ~ 10000 の整数値で指定します。
- height: 印字基準から次の印字基準までの距離 (0.1mm 単位) を指定します。
1 ~ 10000 の整数値で指定します。
0 を指定した場合、印字基準位置から次の印字基準位置までの距離を自動検出します。
- marginTop: 印字基準から頭出し位置までの距離 (0.1mm 単位) を指定します。
-9999 ~ 10000 の整数値で指定します。
- marginBottom: 排出基準から印刷可能領域の下端までの距離 (0.1mm 単位) を指定します。
-9999 ~ 10000 の整数値で指定します。
- offsetCut: 排出基準からカット位置までの距離 (0.1mm 単位) を指定します。
-9999 ~ 10000 の整数値で指定します。
- offsetLabel: 排出基準からラベル下端までの距離 (0.1mm 単位) を指定します。
0 ~ 10000 の整数値で指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

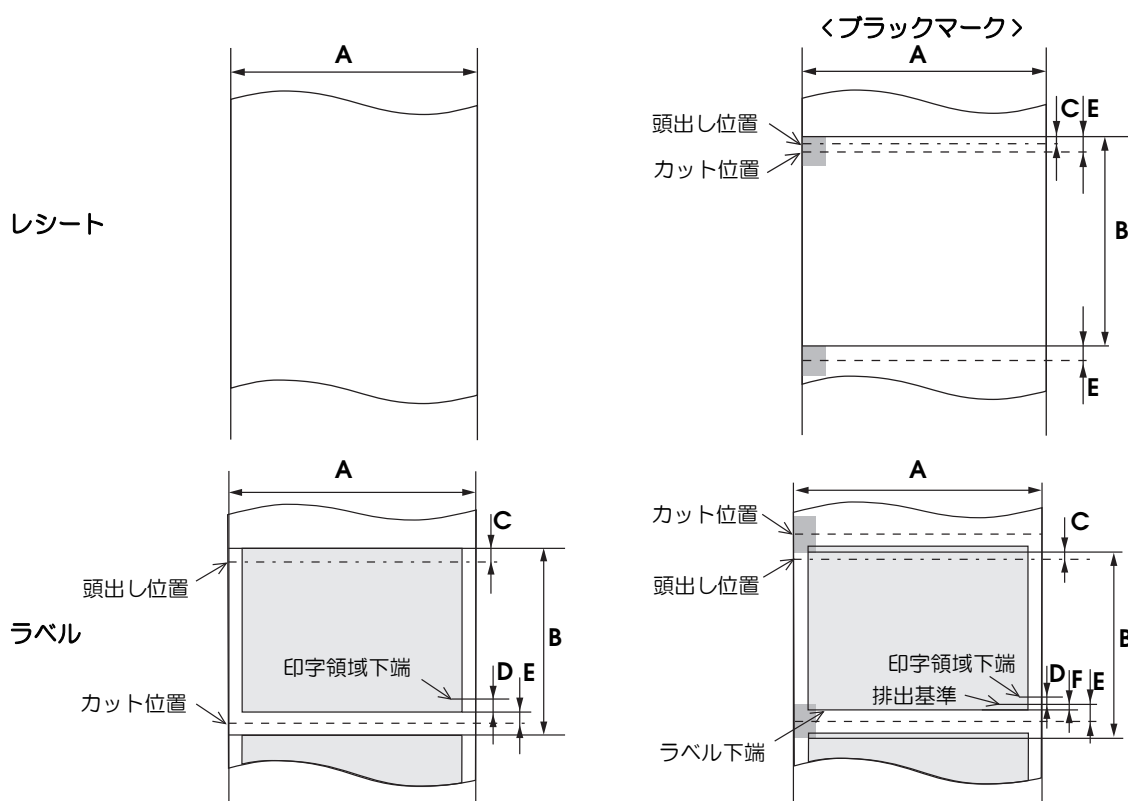
例

60mm ラベル紙（ブラックマークあり）に設定する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-P60II"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [builder addLayout: EPOS_OC_PAPER_TYPE_LABEL_BM
        Width:600 Height:0 MarginTop:15 MarginBottom:-15
        OffsetCut:15 OffsetLabel:0];
    ... 処理 ...
}
```

詳細説明

□ 用紙ごと指定可能なパラメーターと、パラメーターの位置は以下を参照してください。



記号	パラメーター	設定値			
		レシート	レシート (ブラックマーク)	ラベル	ラベル (ブラックマーク)
A	width	1 ~ 10000	1 ~ 10000	1 ~ 10000	1 ~ 10000
B	height	0	0 ~ 10000	0 ~ 10000	0 ~ 10000
C	marginTop	0	-9999 ~ 10000	0 ~ 10000	-9999 ~ 10000
D	marginBottom	0	0	-9999 ~ 0	-9999 ~ 10000
E	offsetCut	0	-9999 ~ 10000	0 ~ 10000	0 ~ 10000
F	offsetLabel	0	0	0	0 ~ 10000

addCommand

コマンドを命令バッファに追加します。ESC/POS コマンドを送信します。



ESC/POS コマンドは、一般公開されていません。詳細については、ご購入元にご相談ください。

構文

- (int) **addCommand:** (NSData *)data;

パラメーター

- data: ESC/POS コマンドをバイナリーデータで指定します。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
              Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    NSData* data = Nil;
    ... 処理 ...
    errorStatus = [builder addCommand: data];
}
```

init

EposPrint クラスのインスタンスを初期化します。

構文

– (id) ***init***;

戻り値


初期化済みの EposPrint クラスインスタンスが返ります。

例


```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    ... 処理 ...
    [printer release];
}
```

openPrinter (プリンターステータス取得用)

プリンターとの通信・プリンターステータスのモニタリングを開始します。



- プリンターステータスは、EposPrint クラスで登録したコールバックメソッドに通知されます。詳細は、[プリンターステータスを自動的に取得 \(30 ページ\)](#) を参照してください。
- プリンターステータスのモニタリングをやめたい場合、[closePrinter \(90 ページ\)](#) を呼び出してください。



プリンターとの通信が不要になった場合、必ず [closePrinter \(90 ページ\)](#) を呼び出し、プリンターとの通信を終了してください。

構文

```
- (int) openPrinter: (int)deviceType
                        DeviceName: (NSString *)deviceName
                        Enabled: (int)enabled
                        Interval: (long)interval;
```

パラメーター

- deviceType: 通信を開始するデバイスの種別を指定します。

設定値	説明
EPOS_OC_DEVTYPE_TCP	Wi-Fi/Ethernet デバイス

- deviceName: 対象デバイスを特定するための識別子を指定します。
deviceType ごとに以下を指定します。

deviceType	設定値
EPOS_OC_DEVTYPE_TCP	IPv4 形式の IP アドレス

- enabled: プリンターステータスのモニタリングの有効・無効を指定します。

設定値	設定値
EPOS_OC_TRUE	有効
EPOS_OC_FALSE	無効
EPOS_OC_PARAM_DEFAULT	無効

- interval: プリンターステータスを更新する間隔 (ミリ秒単位) を指定します。

設定値	設定値
1000 ~ 60000 の整数	プリンターステータスを更新する間隔 (ミリ秒単位)
EPOS_OC_PARAM_DEFAULT	1000 ミリ秒

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_OPEN	オープン処理に失敗した。
EPOS_OC_ERR_ILLEGAL	既に通信が開始されているデバイスを再度通信開始しようとした。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。


例

IP アドレスが“192.168.192.168”のプリンターと Wi-Fi/Ethernet でプリンターステータスのモニタリングを有効にして通信を開始する場合


```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        Name:@"192.168.192.168" Enabled: EPOS_OC_TRUE
        Interval:EPOS_OC_PARAM_DEFAULT];
    . . . 処理 . . .
}
```


openPrinter

プリンターとの通信を開始します。



プリンターステータスを自動的に取得したい場合、openPrinter(プリンターステータス取得用) (87 ページ) を使用してください。



プリンターとの通信が不要になった場合、必ず closePrinter (90 ページ) を呼び出し、プリンターとの通信を終了してください。

構文

```
- (int) openPrinter: (int)deviceType
                        DeviceName: (NSString *)deviceName;
```

パラメーター

- deviceType: 通信を開始するデバイスの種別を指定します。

設定値	説明
EPOS_OC_DEVTYPE_TCP	Wi-Fi/Ethernet デバイス

- deviceName: 対象デバイスを特定するための識別子を指定します。
deviceType ごとに以下を指定します。

deviceType	設定値
EPOS_OC_DEVTYPE_TCP	IPv4 形式の IP アドレス

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_OPEN	オープン処理に失敗した。
EPOS_OC_ERR_ILLEGAL	既に通信が開始されているデバイスを再度通信開始しようとした。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

IP アドレスが“192.168.192.168”のプリンターと Wi-Fi/Ethernet で通信を開始する場合

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                DeviceName:@"192.168.192.168"];
    ... 処理 ...
}
```

closePrinter

プリンターとの通信、およびプリンターステータスのモニタリングを終了します。

構文

– (int) **closePrinter**;

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_ILLEGAL	通信が開始されていない状態で、本 API が呼び出された。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

```
id printer = [[EposPrint alloc] init];
if ( printer != nil) {
    int errorStatus = EPOS_OC_SUCCESS;
    errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
        DeviceName:@"192.168.192.168"];
    ... 処理 ...
    errorStatus = [printer closePrinter];
}
```

sendData

EposBuilder クラスで作成した印刷ドキュメントを送信します。



印刷ドキュメント 送信時に、バッテリーステータスを取得したい場合、`sendData(バッテリーステータス取得用) (93 ページ)` を使用してください。

構文

```
- (int) sendData: (EposBuilder *)builder
               Timeout: (long)timeout
               Status: (unsigned long *)status;
```

パラメーター

- `builder` : EposBuilder クラスのインスタンスを指定します。
EposBuilder クラスの詳細は、[EposBuilder クラス \(35 ページ\)](#) を参照してください。
- `timeout` : 送受信待ちのタイムアウト時間を指定します。
0 ～ 6000000(ミリ秒単位) の整数値を指定します。
- `status` : コマンド送信終了時のプリンターステータスがセットされます。プリンターステータスの設定値の組み合わせがセットされます。詳細は、[プリンターステータス一覧 \(33 ページ\)](#) を参照してください。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_ILLEGAL	通信が開始していない状態で本 API が呼び出された。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_TIMEOUT	指定された時間内に全データを送信できなかった。
EPOS_OC_ERR_CONNECT	通信エラーが発生した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_OFF_LINE	プリンターがオフライン状態だった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

タイムアウトに 10 秒を指定し、プリンターにコマンドを送信する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
             Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    unsigned long status = 0;

    errorStatus = [builder addText:@"ABCDE"];

    id printer = [[EposPrint alloc] init];

    if ( printer != nil ) {
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                         DeviceName:@"192.168.192.168"];
        errorStatus = [printer sendData:builder Timeout:10000
                     Status:&status];
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

sendData (バッテリーステータス取得用)

EposBuilder クラスで作成した印刷ドキュメントを送信します。

構文

```
- (int) sendData: (EposBuilder *)builder
    Timeout: (long) timeout
    Status: (unsigned long *) status
    Battery: (unsigned long *) battery;
```

パラメーター

- builder : EposBuilder クラスのインスタンスを指定します。
EposBuilder クラスの詳細は、[EposBuilder クラス \(35 ページ\)](#) を参照してください。
- timeout : 送受信待ちのタイムアウト時間を指定します。
0 ～ 600000 (ミリ秒単位) の整数値を指定します。
- status : コマンド送信終了時のプリンターステータスがセットされます。プリンターステータスの設定値の組み合わせがセットされます。詳細は、[プリンターステータス一覧 \(33 ページ\)](#) を参照してください。
- battery : バッテリーステータスがセットされます。
詳細は、[プリンターの仕様 \(125 ページ\)](#) を参照してください。

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_ILLEGAL	通信が開始していない状態で本 API が呼び出された。
EPOS_OC_ERR_PROCESSING	処理を実行できなかった。
EPOS_OC_ERR_TIMEOUT	指定された時間内に全データを送信できなかった。
EPOS_OC_ERR_CONNECT	通信エラーが発生した。
EPOS_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPOS_OC_ERR_OFF_LINE	プリンターがオフライン状態だった。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

タイムアウトに 10 秒を指定し、プリンターにコマンドを送信する場合

```
id builder = [[EposBuilder alloc] initWithPrinterModel: @"TM-T88V"
    Lang: EPOS_OC_MODEL_JAPANESE];
if ( builder != nil ) {
    int errorStatus = EPOS_OC_SUCCESS;
    unsigned long status = 0;
    unsigned long battery = 0;

    errorStatus = [builder addText:@"ABCDE"];

    id printer = [[EposPrint alloc] init];

    if ( printer != nil ) {
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
            DeviceName:@"192.168.192.168"];
        errorStatus = [printer sendData:builder Timeout:10000
            Status:&status Battery:&battery];
        errorStatus = [printer closePrinter];
        [printer release];
    }
    [builder release];
}
```

setStatusChangeEventCallback

プリンタステータスのイベントのコールバックメソッドを登録します。



- 本 API は、`openPrinter(プリンタステータス取得用)` (87 ページ) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

- (void) **setStatusChangeEventCallback:** (SEL) method
Target: (NSObject*) target;

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

- (void) **メソッドの名称:** (NSString *)deviceName
Status: (NSNumber *)status;

パラメーター

- deviceName: プリンタステータスを通知した、デバイスの識別子 (IPv4形式のIPアドレス/MACアドレス) がセットされます。
- status: プリンタステータスがセットされます。

例

```
- (void)onStatusChange:(NSString *)deviceName Status:(NSNumber *)status
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setStatusChangeEventCallback @selector(onStatusChange:Status:)
                                         Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                         Name:@"192.168.192.168"
                                         Enabled: EPOS_OC_TRUE
                                         Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setOnlineEventCallback

オンラインイベントのコールバックメソッドを登録します。プリンタステータスがオンライン時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setOnlineEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: オンラインイベントを通知した、デバイスの識別子 (IPv4形式のIPアドレス/MACアドレス) がセットされます。

例

```
- (void)onOnline:(NSString *)deviceName  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setOnlineEventCallback @selector(onOnline:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```


setOfflineEventCallback

オフラインイベントのコールバックメソッドを登録します。プリンタステータスがオフライン時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
- (void) setOfflineEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
- (void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: オフラインイベントを通知した、デバイスの識別子 (IPv4形式のIPアドレス/MACアドレス) がセットされます。

例

```
- (void)onOffline:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setOfflineEventCallback @selector(onOffline:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setPowerOffEventCallback

無応答イベントのコールバックメソッドを登録します。プリンタステータスが無応答時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

- (void) **setPowerOffEventCallback**: (SEL) method
Target: (NSObject*) target;

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

- (void) **メソッドの名称**: (NSString *)deviceName

パラメーター

- deviceName: 無応答イベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onPowerOff:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setPowerOffEventCallback @selector(onPowerOff:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                   Name:@"192.168.192.168"
                                   Enabled: EPOS_OC_TRUE
                                   Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setCoverOkEventCallback

カバークローズイベントのコールバックメソッドを登録します。プリンタステータスがカバークローズ時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setCoverOkEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: カバークローズイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onCoverOk:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setCoverOkEventCallback @selector(onCoverOk:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setCoverOpenEventCallback

カバーオープンイベントのコールバックメソッドを登録します。プリンタステータスがカバーオープン時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setCoverOpenEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: カバーオープンイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onCoverOpen:(NSString *)deviceName  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil ) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setCoverOpenEventCallback @selector(onCoverOpen:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```

setPaperOkEventCallback

用紙ありイベントのコールバックメソッドを登録します。プリンタステータスが用紙あり時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setPaperOkEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: 用紙ありイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onPaperOk:(NSString *)deviceName
{
    ...処理...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setPaperOkEventCallback @selector(onPaperOk:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                   Name:@"192.168.192.168"
                                   Enabled: EPOS_OC_TRUE
                                   Interval:EPOS_OC_PARAM_DEFAULT];

        ...処理...
    }
}
```

setPaperNearEndEventCallback

用紙残量少イベントのコールバックメソッドを登録します。プリンタステータスが用紙残量少時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setPaperNearEndEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: 用紙残量少イベントを通知した、デバイスの識別子 (IPv4形式のIPアドレス/MACアドレス) がセットされます。

例

```
- (void)onPaperNearEnd:(NSString *)deviceName  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setPaperNearEndEventCallback @selector(onPaperNearEnd:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```

setPaperEndEventCallback

用紙なしイベントのコールバックメソッドを登録します。プリンタステータスが用紙なし時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setPaperEndEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: 用紙なしイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onPaperEnd:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setPaperEndEventCallback @selector(onPaperEnd:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setDrawerClosedEventCallback

ドロアークローズイベントのコールバックメソッドを登録します。プリンタステータスがドロアークローズ時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setDrawerClosedEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: ドロアークローズイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onDrawerClosed:(NSString *)deviceName  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setDrawerClosedEventCallback @selector(onDrawerClosed:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```


setDrawerOpenEventCallback

ドロアーオープンイベントのコールバックメソッドを登録します。プリンタステータスがドロアーオープン時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setDrawerOpenEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: ドロアーオープンイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onDrawerOpen:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setDrawerOpenEventCallback @selector(onDrawerOpen:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setBatteryLowEventCallback

バッテリー残量なしイベントのコールバックメソッドを登録します。プリンタステータスがバッテリー残量によるオフライン時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setBatteryLowEventCallback: (SEL) method  
Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: バッテリー残量なしを通知した、デバイスの識別子 (IPv4形式のIPアドレス/MACアドレス) がセットされます。

例

```
- (void)onBatteryLow:(NSString *)deviceName  
{  
    ...処理...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setBatteryLowEventCallback @selector(onBatteryLow:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ...処理...  
    }  
}
```

setBatteryOkEventCallback

バッテリー残量ありイベントのコールバックメソッドを登録します。プリンタステータスがバッテリー残量によるオフラインから復帰時に通知されるメソッドです。



- 本 API は、[openPrinter\(プリンタステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setBatteryOkEventCallback: (SEL) method
                                Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName
```

パラメーター

- deviceName: バッテリー残量ありイベントを通知した、デバイスの識別子 (IPv4 形式の IP アドレス / MAC アドレス) がセットされます。

例

```
- (void)onBatteryOk:(NSString *)deviceName
{
    ... 処理 ...
}

- (void)openPrinter
{
    id printer = [[EposPrint alloc] init];

    if ( printer != nil) {
        int errorStatus = EPOS_OC_SUCCESS;

        [printer setBatteryOkEventCallback @selector(onBatteryOk:) Target:self];

        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP
                                Name:@"192.168.192.168"
                                Enabled: EPOS_OC_TRUE
                                Interval:EPOS_OC_PARAM_DEFAULT];

        ... 処理 ...
    }
}
```

setBatteryStatusChangeEventCallback

バッテリーステータスのイベントのコールバックメソッドを登録します。



- 本 API は、[openPrinter\(プリンターステータス取得用 \) \(87 ページ\)](#) の実行後でも実行できます。
- 本 API を複数回実行した場合、後に指定されたコールバックメソッドで上書きされます。

構文

```
(void) setBatteryStatusChangeEventCallback:  
      (SEL) method Target: (NSObject*) target;
```

パラメーター

- method: コールバックメソッドのセレクターを指定します。
- target: コールバックメソッドを持つオブジェクトを指定します。



method、target のいずれかに Nil を指定した場合、コールバックメソッドが解除されます。

コールバックメソッドの定義

```
(void) メソッドの名称: (NSString *)deviceName  
                        Battery: (NSNumber *)battery;
```

パラメーター

- deviceName: バッテリーステータスを通知した、デバイスの識別子 (IPv4形式のIPアドレス/MACアドレス) がセットされます。
- battery: バッテリーステータスがセットされます。

例

```
- (void)onBatteryStatusChange:(NSString *)deviceName Battery:(NSNumber *)battery  
{  
    ... 処理 ...  
}  
  
- (void)openPrinter  
{  
    id printer = [[EposPrint alloc] init];  
  
    if ( printer != nil) {  
        int errorStatus = EPOS_OC_SUCCESS;  
  
        [printer setBatteryStatusChangeEventCallback  
                @selector(onBatteryStatusChange:Battery:) Target:self];  
  
        errorStatus = [printer openPrinter:EPOS_OC_DEVTYPE_TCP  
                        Name:@"192.168.192.168"  
                        Enabled: EPOS_OC_TRUE  
                        Interval:EPOS_OC_PARAM_DEFAULT];  
  
        ... 処理 ...  
    }  
}
```

プリンター検索 API

プリンターを検索するための API です。以下のクラスが用意されています。

□ EpsonIoFinder クラス (109 ページ)

EpsonIoFinder クラス

プリンターを検索するクラスです。以下の API が用意されています。

API	説明	ページ
start	プリンター検索を開始	109
stop	プリンターとの通信を終了	110
getResult	プリンターの検索結果を取得	111

start

指定されたデバイス種別のプリンター検索を開始します。



- 本 API を使用したら、必ず stop (110 ページ) で検索終了してください。
- 既にプリンター検索を開始している状態で、本 API を呼び出すことはできません。

構文

```
+ (int) start:(int)deviceType  
FindOption:(NSString *)findOption;
```

パラメーター

- deviceType: 検索するデバイス種別を指定します。以下の値を指定します。

deviceType	説明
EPSONIO_OC_DEVTYPE_TCP	ネットワークに接続された TM デバイスを検索します。

- findOption: 対象デバイスを検索する際の設定値を指定します。

deviceType	指定する値
EPSONIO_OC_DEVTYPE_TCP	検索範囲のブロードキャストアドレス

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_ILLEGAL	既に検索を開始した状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

stop

プリンター検索を終了します。

構文

```
+ (int) stop;
```

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_ILLEGAL	検索を開始していない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

getResult

本 API を呼び出した時点までの、プリンターの検索結果を取得します。

構文

```
+ (NSArray *) getResult:(int *)errorStatus;
```

パラメーター

- errorStatus： エラーステータスが返ります。

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_ILLEGAL	検索を開始していない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

戻り値

検索したデバイスのリストが返されます。
リスト内には、検索したデバイスの識別情報が、文字列 (String 型) で格納されています。
デバイス種別 (deviceType) によって、格納される結果が異なります。

deviceType	取得するリスト
EPSONIO_OC_DEVTTYPE_TCP	プリンターの IP アドレスのリスト

ログ設定 API

ログ出力の設定をします。以下のクラスが用意されています。

□ EposLog クラス (112 ページ)

EposLog クラス

ログの出力機能を設定します。

API	説明	ページ
setLogSettings	ログ出力機能の設定	112

setLogSettings

ログ出力機能を設定します。

構文

```
+ (int) setLogSettings: (int) period
                        Enabled: (int) enabled
                        IPAddress: (NSString *) ipAddress
                        Port: (int) port LogSize: (int) logSize
                        LogLevel: (int) logLevel;
```

パラメーター

- period: ログ出力機能の設定方法を指定します。

設定値	説明
EPOS_OC_LOG_TEMPORARY	アプリケーションを終了すると、本 API の設定は無効になります。
EPOS_OC_LOG_PERMANENT	アプリケーションを終了させても、本 API の設定を有効にします。

- enabled: ログ出力機能の有効 / 無効、およびログの出力先を指定します。


設定値	説明
EPOS_OC_LOG_DISABLE	ログ出力機能を無効にする。
EPOS_OC_LOG_STORAGE	端末のストレージに出力する。
EPOS_OC_LOG_TCP	TCP で出力する。



enabled を EPOS_OC_LOG_STORAGE に指定する場合、iTunes のファイル共有を可能にしてください。以下の手順で設定します。

- アプリケーションの info.plist に “UIFileSharingEnabled” を追加します。自動で “Application supports iTunes file sharing” に変更されます。
- “Application supports iTunes file sharing” の Value を、“YES” に設定します。


- ipAddress： TCP 通信の IP アドレス (IPv4 形式) を指定します。



enabled が以下の値の場合、“nil” も指定できます。

- * EPOS_OC_LOG_DISABLE
- * EPOS_OC_LOG_STORAGE


- port： TCP 通信のポート番号を指定します。0 ～ 65535 の整数値を指定します。



enabled に以下の値を指定した場合も、範囲内の任意の値を指定してください。

- * EPOS_OC_LOG_DISABLE
- * EPOS_OC_LOG_STORAGE

- logSize： 端末のストレージへ保存する、ログの最大容量を指定します。
1 ～ 50 (MB 単位) の整数値を指定します。



enabled に以下の値を指定した場合も、範囲内の任意の値を指定してください。

- * EPOS_OC_LOG_DISABLE
- * EPOS_OC_LOG_TCP

- logLevel： ログの出力レベルを指定します。

設定値	説明
EPOS_OC_LOG_LOW	低レベル

戻り値

エラーステータス	説明
EPOS_OC_SUCCESS	処理に成功した。
EPOS_OC_ERR_PARAM	不正なパラメーターが渡された。
EPOS_OC_ERR_FAILURE	その他のエラーが発生した。

例

TCP で、IP アドレス 192.168.192.168 の 8080 番ポートにログを出力する場合

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
                Enabled:EPOS_OC_LOG_TCP IPAddress:@"192.168.192.168"
                Port:8080 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
    . . . 処理 . . .
}
```

端末のストレージにログを出力する場合

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
                Enabled: EPOS_OC_LOG_STORAGE IPAddress:nil
                Port:0 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
    . . . 処理 . . .
}
```

ログ出力機能を無効にする場合

```
errorStatus = [EposLog setLogSettings: EPOS_OC_LOG_PERMANENT
                Enabled: EPOS_OC_LOG_DISABLE IPAddress:nil
                Port:0 LogSize:10 LogLevel: EPOS_OC_LOG_LOW ];
    . . . 処理 . . .
}
```

コマンドの送受信

本章では、コマンド (ESC/POS コマンドなど) を送受信するための API について説明しています。

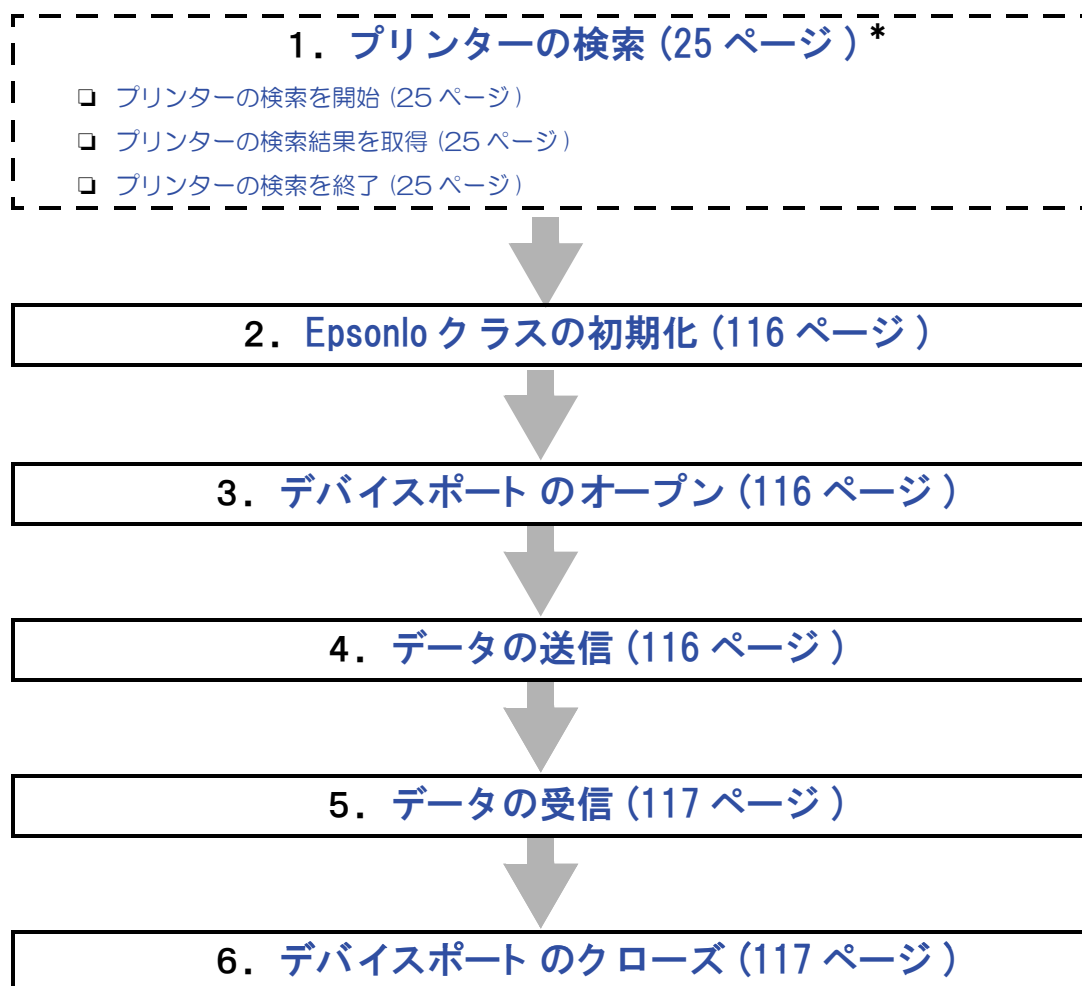


コマンド送受信 API は、ePOS-Print API の [EposPrint クラス \(37 ページ\)](#) と同時に使用できません。

プログラミング

プログラミングフロー

以下のフローでプログラミングします。



*: 任意のプロセスです。

Epsonlo クラスの初期化

[init \(119 ページ\)](#) を使って、Epsonlo クラスを初期化します。以下のプログラミングを参考にしてください。

```
//Epsonlo クラスの初期化
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
    - 処理 -
    [port release];
}
```

デバイスポート のオープン

Epsonlo クラスの [open \(120 ページ\)](#) を使って、デバイスポートをオープンします。以下のプログラミングを参考にしてください。

```
//Epsonlo クラスの初期化
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
    int errorStatus = EPSONIO_OC_SUCCESS;
    // デバイスポートのオープン
    errorStatus = [port open:EPSONIO_OC_DEVTYPE_TCP DeviceName:
        @"192.168.192.168" DeviceSettings:nil];
    if (EPSONIO_OC_SUCCESS == errorStatus ) {
        - 処理 -
    }
}
```

データの送信

Epsonlo クラスの [write \(122 ページ\)](#) を使って、プリンターにデータを送信します。以下のプログラミングを参考にしてください。

文字列「Hello, World!」を印刷する場合

```
// 送信設定
long sizeWritten;
int errStatus;
NSString *str = @"Hello, World!\r\n";
NSData *data = [str dataUsingEncoding:NSUTF8StringEncoding];

// データの送信
errStatus = [port write:data Offset:0 Size:[data length]
    Timeout:100 SizeWritten:& sizeWritten];
```

データの受信

EpsonIo クラスの [read \(123 ページ\)](#) を使って、プリンターからのデータを受信します。以下のプログラミングを参考にしてください。

```
// 受信設定
NSMutableData *data;
long sizeRead;
int errStatus;
data = [[NSMutableData alloc] initWithLength:256];

// データの受信
errStatus =
[port read:data Offset:0 Size:256 Timeout:100 SizeRead:& sizeRead];
```

デバイスポートのクローズ

EpsonIo クラスの [close \(121 ページ\)](#) を使って、デバイスポートをクローズします。以下のプログラミングを参考にしてください。

```
//EpsonIo クラスの初期化
id port = [[EpsonIo alloc] init];
if ( port != nil ) {
    int errorStatus = EPSONIO_OC_SUCCESS;
    // デバイスポートのオープン
    errorStatus = [port open:EPSONIO_OC_DEVTYPE_TCP DeviceName:
@"192.168.192.168" DeviceSettings:nil];
    if (EPSONIO_OC_SUCCESS == errorStatus ) {
        - 処理 -
    }
    // デバイスポートのクローズ
    errorStatus = [port close];
}
```

エラー値一覧

コマンド送受信 API には、以下のエラー値が定義されています。

エラー値	要因
EPSONIO_OC_SUCCESS	処理に成功した。
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された。 < 例 > <ul style="list-style-type: none">• Nil など、不正な引数を渡された。• サポートしていない範囲の値が指定された。
EPSONIO_OC_ERR_OPEN,	オープン処理に失敗した < 例 > TCP 通信用の Socket の作成に失敗した。
EPSONIO_OC_ERR_CONNECT	デバイスとの通信に失敗した。 < 例 > <ul style="list-style-type: none">• タイムアウト以外の要因で、対象デバイスへのデータ送信に失敗した。• タイムアウト以外の要因で、対象デバイスからのデータ受信に失敗した。
EPSONIO_OC_ERR_TIMEOUT	指定されたタイムアウト時間を越えた。 < 例 > <ul style="list-style-type: none">• 指定されたサイズのデータを指定時間内に送信できなかった。• 指定された時間内に 1 バイトも受信できなかった。
EPSONIO_OC_ERR_MEMORY	処理に必要なメモリーが確保できなかった。
EPSONIO_OC_ERR_ILLEGAL	不適切な方法で使用された。 < 例 > <ul style="list-style-type: none">• デバイSPORTがオープンされていない状態でデータ送受信 API が呼び出された。• 既にプリンター検索が開始されている状態で、再度検索開始 API が呼び出された。
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった。 < 例 > 同様の処理を他のスレッドで実行中のため、共有リソースのロック権限を取得できなかった。
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した。

コマンド送受信 API リファレンス

コマンド送受信 API には以下のクラスが用意されています。

Epsonlo クラス

データ送受信用のクラスです。以下の API が用意されています。

API	説明	ページ
init	Epsonlo クラスのインスタンスを初期化	119
open	デバイスポートのオープン	120
close	デバイスポートのクローズ	121
write	データ送信	122
read	データ受信	123

init

生成された Epsonlo クラスのインスタンスを初期化します。

構文

```
- (id) init;
```

戻り値

初期化済みの Epsonlo クラスのインスタンスが返されます。

open

指定されたデバイスポートをオープンします。

構文

```
(int) open: (int)deviceType  
DeviceName: (NSString *)deviceName  
DeviceSettings: (NSString *)deviceSettings;
```

パラメーター

- deviceType: オープンするデバイス種別を指定します。以下の値を指定します。

設定値	説明
EPSONIO_OC_DEVTYPE_TCP	オープンするプリンターが Wi-Fi/Ethernet の時に指定します。

- deviceName: 対象デバイスを特定するための識別子を指定します。以下の値を指定します。

deviceType	設定値
EPSONIO_OC_DEVTYPE_TCP	IPv4 形式の IP アドレス

- deviceSettings (Reserved):
"nil" を指定します。

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_OPEN	オープン処理に失敗した
EPSONIO_OC_ERR_ILLEGAL	既にオープンされているデバイスを再度オープンしようとした
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

close

指定されたデバイスポートをクローズします。

構文

```
- (int) close;
```

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_ILLEGAL	オープンしていない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

write

データをデバイスポートへ送信します。

構文

```
(int) write: (NSData *)data  
          Offset: (size_t)offset  
          Size: (size_t)size  
          Timeout: (long)timeout  
          SizeWritten: (size_t *)sizeWritten;
```

パラメーター

- data : 送信データのバッファです。送信するデータを格納します。
- offset : 送信開始位置を指定します。
送信データバッファの先頭からのオフセット値を指定してください。
- size : 送信したいデータのバイト数を指定します。



size に 0 が指定された場合、送信されません。この場合、sizeWritten に 0 が返ります。

- timeout : 送信待ちのタイムアウト時間を、msec 単位で指定します。
指定可能な最長時間は 600000 msec (10 分) です。



- timeout は、伝送速度、送信データ量などを考慮して指定してください。
- timeout が短い場合、正常にデータが送信できている間、全データを送信し終えるまで timeout を超えても送信処理を継続します。

- sizeWritten : 送信を終了したデータのバイト数が格納されます。



sizeWritten で返されるサイズのデータが、実際にプリンターが受信しているとは限りません。



timeout で指定した時間を過ぎた場合、その時点までに送信を終了したバイト 数を sizeWritten に格納します。

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_ILLEGAL	オープンしていない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_TIMEOUT	指定された時間内に全データを送信できなかった
EPSONIO_OC_ERR_CONNECT	通信エラーが発生した
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した

read

データをデバイスポートから受信します。



本 API は、受信エラーが発生するまでデータを受信し続けますが、timeout で指定された時間内に 1 バイトもデータが受信できなかった場合、処理が終了します。

構文

```
- (int) read: (NSMutableData *)data
             Offset: (size_t)offset
             Size: (size_t)size
             Timeout: (long)timeout
             SizeRead: (size_t *)sizeRead;
```

パラメーター

- data: 受信データの格納先バッファです。
- offset: 格納先バッファの格納開始位置を指定します。
受信データバッファの先頭からのオフセット値を指定します。
- size: 受信可能なバイト数を指定します。



size に 0 が指定された場合、受信されません。この場合、sizeRead に 0 が返ります。

- timeout: データ受信する時間を、msec 単位で指定します。
指定可能な最長時間は 600000 msec(10 分)です。
- sizeRead: 受信したデータのバイト数が格納されます。

戻り値

戻り値	説明
EPSONIO_OC_SUCCESS	処理に成功した
EPSONIO_OC_ERR_ILLEGAL	オープンしていない状態で本 API が呼び出された
EPSONIO_OC_ERR_PROCESSING	処理を実行できなかった
EPSONIO_OC_ERR_PARAM	不正なパラメーターが渡された
EPSONIO_OC_ERR_TIMEOUT	指定された時間内に 1 バイトもデータが受信できなかった
EPSONIO_OC_ERR_CONNECT	通信エラーが発生した
EPSONIO_OC_ERR_MEMORY	メモリーを確保できなかった
EPSONIO_OC_ERR_FAILURE	その他のエラーが発生した



付録

プリンターの仕様

TM-T88V

		58mm 仕様	80mm 仕様
インターフェイス		Ethernet/ 無線 LAN	
解像度		180 x 180 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		360 ドット	512 ドット
印字桁数	フォント A	ANK 30 桁 / 漢字 15 桁	ANK 42 桁 / 漢字 21 桁
	フォント B	ANK 40 桁	ANK 56 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 9 x 17 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 16 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		360 x 831 ドット	512 x 831 ドット
ページモード最大領域		360 x 1662 ドット	512 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked (Composit Symbology 非サポート)	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		オプション	
バッテリー		非サポート	

ePOS-Print API サポート 一覧

ePOS-Print API	ページ	ePOS-Print API	ページ
EposBuilder クラス			
initWithPrinterModel	38	clearCommandBuffer	39
addTextAlign	40	addTextLineSpace	41
addTextRotate	42	addText	43
addTextLang	44	addTextFont	45
addTextSmooth	46	addTextDouble	47
addTextSize	48	addTextStyle	49
addTextPosition	51	addFeedUnit	52
addFeedLine	53	addImage(多階調印字用)	54
addImage	57	addLogo	59
addBarcode	60	addSymbol	65
addPageBegin	70	addPageEnd	71
addPageArea	72	addPageDirection	74
addPagePosition	75	addCut	76
addPulse	77	addSound(周期鳴動設定用)	78
addSound	80	addCommand	85
EposPrint クラス			
init	86	openPrinter (プリンターステータス取得用)	87
openPrinter	89	closePrinter	90
sendData	91	setStatusChangeEventCallback	95
setOnlineEventCallback	96	setOfflineEventCallback	97
setPowerOffEventCallback	98	setCoverOkEventCallback	99
setCoverOpenEventCallback	100	setPaperOkEventCallback	101
setPaperNearEndEventCallback	102	setPaperEndEventCallback	103
setDrawerClosedEventCallback	104	setDrawerOpenEventCallback	105



コマンド送受信 API は全てサポートしています。

TM-T70

		58mm 仕様	80mm 仕様
インターフェイス		Ethernet/ 無線 LAN	
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		416 ドット	576 ドット
印字桁数	フォント A	ANK 34 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 52 桁 / 漢字 26 桁	ANK 72 桁 / 漢字 36 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 8 x 16 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		416 x 1662 ドット	576 x 1662 ドット
ページモード最大領域		416 x 1662 ドット	576 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128	
2 次元シンボル		QR Code	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		非サポート	
バッテリー		非サポート	

ePOS-Print API サポート 一覧

ePOS-Print API	ページ	ePOS-Print API	ページ
EposBuilder クラス			
initWithPrinterModel	38	clearCommandBuffer	39
addTextAlign	40	addTextLineSpace	41
addTextRotate	42	addText	43
addTextLang	44	addTextFont	45
addTextSmooth	46	addTextDouble	47
addTextSize	48	addTextStyle	49
addTextPosition	51	addFeedUnit	52
addFeedLine	53	addImage(多階調印字用)	54
addImage	59	addLogo	59
addBarcode	65	addSymbol	65
addPageBegin	71	addPageEnd	71
addPageArea	74	addPageDirection	74
addPagePosition	76	addCut	76
addPulse	77	addCommand	85
EposPrint クラス			
init	86	openPrinter (プリンターステータス取得用)	87
openPrinter	89	closePrinter	90
sendData	91	setStatusChangeEventCallback	95
setOnlineEventCallback	96	setOfflineEventCallback	97
setPowerOffEventCallback	98	setCoverOkEventCallback	99
setCoverOpenEventCallback	100	setPaperOkEventCallback	101
setPaperNearEndEventCallback	102	setPaperEndEventCallback	103
setDrawerClosedEventCallback	104	setDrawerOpenEventCallback	105



コマンド送受信 API は全てサポートしています。

TM-T70II

		58mm 仕様	80mm 仕様
インターフェイス		Ethernet/ 無線 LAN	
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		416 ドット	576 ドット
印字桁数	フォント A	ANK 34 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 52 桁 / 漢字 26 桁	ANK 72 桁 / 漢字 36 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 9 x 17 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		416 x 1662 ドット	576 x 1662 ドット
ページモード最大領域		416 x 1662 ドット	576 x 1662 ドット
バーコード		Codabar, Code39, ITF, JAN13(EAN), JAN8(EAN), UPC-A, UPC-E, Code93, Code128, GS1-128, GS1 DataBar Omni-directional, GS1 DataBar Truncated, GS1 DataBar Expanded, GS1 DataBar Limited	
2 次元シンボル		PDF417, QRCode, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omni-directional, GS1 DataBar Expanded Stacked	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		サポート	
バッテリー		非サポート	

ePOS-Print API サポート 一覧

ePOS-Print API	ページ	ePOS-Print API	ページ
EposBuilder クラス			
initWithPrinterModel	38	clearCommandBuffer	39
addTextAlign	40	addTextLineSpace	41
addTextRotate	42	addText	43
addTextLang	44	addTextFont	45
addTextSmooth	46	addTextDouble	47
addTextSize	48	addTextStyle	49
addTextPosition	51	addFeedUnit	52
addFeedLine	53	addImage(多階調印字用)	54
addImage	59	addLogo	59
addBarcode	65	addSymbol	65
addPageBegin	71	addPageEnd	71
addPageArea	74	addPageDirection	74
addPagePosition	76	addCut	76
addPulse	77	addSound(周期鳴動設定用)	78
addSound	80	addCommand	85
EposPrint クラス			
init	86	openPrinter (プリンターステータス取得用)	87
openPrinter	89	closePrinter	90
sendData	91	setStatusChangeEventCallback	95
setOnlineEventCallback	96	setOfflineEventCallback	97
setPowerOffEventCallback	98	setCoverOkEventCallback	99
setCoverOpenEventCallback	100	setPaperOkEventCallback	101
setPaperNearEndEventCallback	102	setPaperEndEventCallback	103
setDrawerClosedEventCallback	104	setDrawerOpenEventCallback	105



コマンド送受信 API は全てサポートしています。

TM-T90II

		58mm 仕様	80mm 仕様
インターフェイス		Ethernet/ 無線 LAN	
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> 日本語モデル 	
印字幅		416 ドット	576 ドット
印字桁数	フォント A	ANK 35 桁 / 漢字 17 桁	ANK 48 桁 / 漢字 24 桁
	フォント B	ANK 42 桁 / 漢字 21 桁	ANK 57 桁 / 漢字 28 桁
	フォント C	ANK 52 桁 / 漢字 26 桁	ANK 72 桁 / 漢字 36 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 10 x 24 ドット / 漢字 20 x 24 ドット	
	フォント C	ANK 8 x 16 ドット / 漢字 16 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 15 ドット目	
	フォント C	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		416 x 1662 ドット	576 x 1662 ドット
ページモード最大領域		416 x 1662 ドット	576 x 1662 ドット
バーコード		Codabar, Code39, ITF, JAN13(EAN), JAN8(EAN), UPC-A, UPC-E, Code93, Code128, GS1-128, GS1 DataBar Omni-directional, GS1 DataBar Truncated, GS1 DataBar Expanded, GS1 DataBar Limited	
2 次元シンボル		PDF417, QRCode, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omni-directional, GS1 DataBar Expanded Stacked	
用紙のカット		カット / フィードカット	
ドロアーキック		サポート	
ブザー		サポート	
バッテリー		非サポート	

ePOS-Print API サポート 一覧

ePOS-Print API	ページ	ePOS-Print API	ページ
EposBuilder クラス			
initWithPrinterModel	38	clearCommandBuffer	39
addTextAlign	40	addTextLineSpace	41
addTextRotate	42	addText	43
addTextLang	44	addTextFont	45
addTextSmooth	46	addTextDouble	47
addTextSize	48	addTextStyle	49
addTextPosition	51	addFeedUnit	52
addFeedLine	53	addImage(多階調印字用)	54
addImage	59	addLogo	59
addBarcode	65	addSymbol	65
addPageBegin	71	addPageEnd	71
addPageArea	74	addPageDirection	74
addPagePosition	76	addCut	76
addPulse	77	addCommand	85
EposPrint クラス			
init	86	openPrinter (プリンターステータス取得用)	87
openPrinter	89	closePrinter	90
sendData	91	setStatusChangeEventCallback	95
setOnlineEventCallback	96	setOfflineEventCallback	97
setPowerOffEventCallback	98	setCoverOkEventCallback	99
setCoverOpenEventCallback	100	setPaperOkEventCallback	101
setPaperNearEndEventCallback	102	setPaperEndEventCallback	103
setDrawerClosedEventCallback	104	setDrawerOpenEventCallback	105



コマンド送受信 API は全てサポートしています。

TM-P60II

		58mm 仕様	60mm 仕様
インターフェイス		無線 LAN/ Bluetooth	
解像度		203 x 203 dpi	
言語		<ul style="list-style-type: none"> • ANK モデル • 日本語モデル 	
印字幅		420 ドット	432 ドット
印字桁数	フォント A	ANK 35 桁 / 漢字 17 桁	ANK 36 桁 / 漢字 18 桁
	フォント B	ANK 42 桁	ANK 43 桁
	フォント C	ANK 52 桁	ANK 54 桁
文字サイズ	フォント A	ANK 12 x 24 ドット / 漢字 24 x 24 ドット	
	フォント B	ANK 10 x 24 ドット	
	フォント C	ANK 8 x 16 ドット	
文字のベースライン	フォント A	文字の上端から 21 ドット目	
	フォント B	文字の上端から 16 ドット目	
	フォント C	文字の上端から 15 ドット目	
初期改行量		30 ドット	
色指定		第 1 色	
ページモード初期領域		420 x 831 ドット	576 x 831 ドット
ページモード最大領域		420 x 1662 ドット	576 x 1662 ドット
バーコード		UPC-A, UPC-E, EAN13, JAN13, EAN8, JAN8, CODE39, ITF, CODABAR, CODE93, CODE128, GS1-128, GS1 DataBar Omnidirectional, GS1 DataBar Truncated, GS1 DataBar Limited, GS1 Databar Expanded	
2 次元シンボル		PDF417, QR Code, MaxiCode, GS1 DataBar Stacked, GS1 DataBar Stacked Omnidirectional, GS1 DataBar Expanded Stacked, Composit Symbology	
用紙のカット		カット / フィードカット	
ドロアーキック		非サポート	
ブザー		オプション	
バッテリー		サポート	

ePOS-Print API サポート 一覧

ePOS-Print API	ページ	ePOS-Print API	ページ
EposBuilder クラス			
initWithPrinterModel	38	clearCommandBuffer	39
addTextAlign	40	addTextLineSpace	41
addTextRotate	42	addText	43
addTextLang	44	addTextFont	45
addTextSmooth	46	addTextDouble	47
addTextSize	48	addTextStyle	49
addTextPosition	51	addFeedUnit	52
addFeedLine	53	addFeedPosition	83
addImage(多階調印字用)	54	addImage	57
addLogo	59	addBarcode	60
addSymbol	65	addPageBegin	70
addPageEnd	71	addPageArea	72
addPageDirection	74	addPagePosition	75
addCut	76	addSound(周期鳴動設定用)	78
addSound	80	addLayout	83
addCommand	85		
EposPrint クラス			
init	86	openPrinter (プリンターステータス取得用)	87
openPrinter	89	closePrinter	90
sendData	91	sendData (バッテリーステータス取得用)	93
setStatusChangeEventCallback	95	setOnlineEventCallback	96
setOfflineEventCallback	97	setPowerOffEventCallback	98
setCoverOkEventCallback	99	setCoverOpenEventCallback	100
setPaperOkEventCallback	101	setPaperNearEndEventCallback	102
setPaperEndEventCallback	103	setBatteryLowEventCallback	106
setBatteryOkEventCallback	107	setBatteryStatusChangeEventCallback	108



コマンド送受信 API は全てサポートしています。

バッテリーステータス

上位 8 ビット

バッテリーステータス	要因
0x30	AC アダプターが接続されている
0x31	AC アダプターが接続されていない

下位 8 ビット

バッテリーステータス	要因
0x30	バッテリー残量 0(リアルエンド)
0x31	バッテリー残量 1(ニアエンド)
0x32	バッテリー残量 2
0x33	バッテリー残量 3
0x34	バッテリー残量 4
0x35	バッテリー残量 5
0x36	バッテリー残量 6



バッテリーステータス取得不可能状態の場合、“0x0000”を返します。

