

CITIZEN

iOS POS Print SDK (Swift)

プログラムマニュアル

Ver. 2.12 用

シチズン・システムズ株式会社

更新履歴

年月日	バージョン	履歴
2015/08/18	1.12	新規
2015/12/10	1.13	対応端末の項に開発環境の記載を追加 対応モデルに CT-S401 を追加
2016/01/29	1.14	connect, disconnect, printerCheck メソッドの説明を修正
2016/05/23	1.15	「1.5 定義方法」に「Xcode に SDK を追加」の説明を追加 「1.5 定義方法」に「アーカイブファイルを作成する時の設定」の説明を追加
2016/12/22	1.16	2.38 MapMode プロパティを追加
2017/07/12	1.17	対応モデルに CT-S253/255 を追加 「1.4 対応モデル」の項内で、CT-S601II/651II/801II/851II 系 メモリスイッチ設定の箇所、MSW13-1、MSW13-5 の設定内容を追記
2017/11/21		「1.5 定義方法」に「SDK 種類」の説明を追加
2018/07/12	2.00	「1.5 対応モデル(周辺機器)」の説明を追加 「3.ラインディスプレイ制御」を追加 「4.バーコードスキャナー制御」を追加
2018/10/24	2.01	プリンター対応モデルに CT-S257 を追加 バーコードスキャナー対応モデルに BC-NL3000U を追加 「1.6 定義方法」の「SDK 種類」に周辺機器を利用する際の SDK バージョンの説明を追加 「3.3.2 ラインディスプレイを利用する際の SDK バージョンについて」を追加 「4.3.2 バーコードスキャナーを利用する際の SDK バージョンについて」を追加
2019/01/16	2.02	「1.5 対応モデル(周辺機器)」に Bluetooth インターフェースについての説明を追加 「3.2.3 connect メソッド」に Bluetooth 接続の説明を追加 「4.2.3 connect メソッド」に Bluetooth 接続の説明を追加
2019/02/25	2.03	プリンター対応モデルに CT-S4500 を追加 printPaddingText メソッドを追加 「2.3.2 UTF-8 エンコード文字列の印刷について」を追加
2019/04/02		SDK の種類に Swift5.0(iOS12.2)を追加
2019/10/07		SDK の種類に Swift5.1(iOS13.1/iPadOS13.1)を追加
2019/11/12		SDK の種類に Swift5.1.2(iOS13.2/iPadOS13.2)を追加
2019/12/12	2.04	searchCitizenPrinter/searchESCPOSPrinter メソッドに、CT-S2000 と CT-S4000 を CMP_PORT_WiFi 指定時に検索出来るプリンターとして追加
2020/01/16	2.05	「1.1 ドキュメント対象範囲」に自動釣銭機制御に関する説明を追加 各デバイスに「プログラム構造」の章を追加
2020/03/16	2.06	SDK の種類に Swift5.1 以降用のフレームワークを追加
2020/10/05	2.07	Lightning I/F に関する説明を追加 対応モデル(プリンター)のインターフェースに USB を追加 connect メソッドの接続タイプに USB を追加 printTextLocalFont メソッドを追加 setPrintCompletedTimeout メソッドを追加
2021/01/05		connect メソッドの USB 接続時の使用例を修正
2021/02/09	2.08	プリンター対応モデルに CT-E301 と CT-E601 を追加 「1.6 SDK 定義方法」の説明を、XCFramework 形式に変更 printBitmap メソッドのモード引き数に CMP_BM_MODE_CMD_GRAY16DOWNLOAD を追加
2021/11/01	2.09	プリンターの対応モデルに CMP シリーズを追加 LAN モデルの検索を行うための設定 (Bonjour) を追加 printTextLocalFont メソッドのイタリック指定についての説明を追加
2022/04/08	2.10	SDK の種類に Swift5.3(Xcode12.0)以降用 XCFramework を追加 printData メソッドの説明を修正 ログ機能の説明を追加

2023/06/30	2.11	SDK 定義方法の説明を修正 プリンター制御の status メソッドにニアエンプティとドロワー状態を追加 プリンター制御の printerCheckEx および openDrawerEx メソッドを追加 プリンター/ディスプレイ/スキャナー制御に ErrorCodeExtended プロパティを追加
2023/11/21		プリンター対応モデルに CT-S801III と CT-S851III を追加 ディスプレイ対応モデルに DSP01-LT2/DSP02-LS2 を追加
2024/01/10		プライバシーマニフェストに対応
2024/07/18	2.12	バージョン番号の変更 誤記訂正

ご注意

1. 本書の内容の一部、または全部を無断で転載することは、固くお断りいたします。
2. 本書の内容については、事前の予告なしに変更することがあります。
3. 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、万一誤り・お気付きの点がございましたら、ご連絡くださいますようお願いいたします。
4. 運用した結果の影響につきましては、3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。
5. 上記に同意いただけない場合は、本SDKをご使用いただけません。

商標

iOS は米国およびその他の国における Cisco の商標または登録商標です。

iPod touch、Swift、Xcode は米国およびその他の国における Apple Inc.の商標または登録商標です。

その他、記載されている会社名、製品名は、各社の商標または登録商標です。

CITIZEN は、シチズン時計株式会社の登録商標です。

目 次

1. 概要	8
1.1. ドキュメント対象範囲	8
1.2. システム概要	8
1.3. 対応端末	8
1.4. 対応モデル（プリンター）	9
1.5. 対応モデル（周辺機器）	17
1.6. SDK 定義方法	20
1.7. Bluetooth の利用方法	23
2. プリンター制御	25
2.1. プログラム構造	25
2.2. 機能一覧	26
2.3. SDK インターフェース	28
2.3.1. 戻り値	28
2.3.2. インスタンス	29
2.3.3. connect メソッド	30
2.3.4. disconnect メソッド	32
2.3.5. setEncoding メソッド	33
2.3.6. printerCheck メソッド	34
2.3.7. status メソッド	35
2.3.8. printText メソッド	37
2.3.9. printPaddingText メソッド	38
2.3.10. printTextLocalFont メソッド	40
2.3.11. printBitmap/printBitmapData メソッド	41
2.3.12. setNVBitmap メソッド	43
2.3.13. printNVBitmap メソッド	45
2.3.14. printBarCode メソッド	46
2.3.15. printPDF417 メソッド	48
2.3.16. printQRCode メソッド	49
2.3.17. printGS1DataBarStacked メソッド	50
2.3.18. cutPaper メソッド	51
2.3.19. unitFeed メソッド	52
2.3.20. markFeed メソッド	53
2.3.21. openDrawer メソッド	54
2.3.22. transactionPrint メソッド	55
2.3.23. rotatePrint メソッド	56
2.3.24. pageModePrint メソッド	57
2.3.25. clearPrintArea メソッド	59
2.3.26. clearOutput メソッド	60
2.3.27. printData メソッド	61
2.3.28. printNormal メソッド	62
2.3.29. watermarkPrint メソッド	63
2.3.30. searchCitizenPrinter メソッド	64
2.3.31. searchESCPOSPrinter メソッド	66
2.3.32. printerCheckEx メソッド	68
2.3.33. openDrawerEx メソッド	70
2.3.34. setPrintCompletedTimeout メソッド	72
2.3.35. setLog メソッド	73

2.3.36. getVersionCode メソッド	74
2.3.37. getVersionName メソッド	75
2.3.38. PageModeArea プロパティ	76
2.3.39. PageModePrintArea プロパティ	77
2.3.40. PageModePrintDirection プロパティ	78
2.3.41. PageModeHorizontalPosition プロパティ	79
2.3.42. PageModeVerticalPosition プロパティ	80
2.3.43. RecLineSpacing プロパティ	81
2.3.44. MapMode プロパティ	82
2.3.45. ErrorCodeExtended プロパティ	83
2.4. 注意事項	84
2.4.1. 印刷完了確認機能について	84
2.4.2. UTF-8 エンコード文字列の印刷について	84
2.4.3. JIS 第3、第4水準漢字の印刷について	85
2.4.4. ログ機能について	86
2.4.5. 定数定義一覧	88
3. ラインディスプレイ制御	91
3.1. プログラム構造	91
3.2. 機能一覧	92
3.3. SDK インターフェース	93
3.3.1. 戻り値	93
3.3.2. インスタンス	94
3.3.3. connect メソッド	95
3.3.4. disconnect メソッド	97
3.3.5. displayText メソッド	98
3.3.6. clearDisplay メソッド	99
3.3.7. blinkDisplay メソッド	100
3.3.8. setDisplayMode メソッド	101
3.3.9. setDisplayConfig メソッド	102
3.3.10. setCursorPosition メソッド	103
3.3.11. moveCursor メソッド	104
3.3.12. setCursorType メソッド	105
3.3.13. initializeDisplay メソッド	106
3.3.14. displayData タグ	107
3.3.15. setEncoding メソッド	108
3.3.16. setCodePage メソッド	109
3.3.17. setInternationalCharacterSet メソッド	110
3.3.18. displayCheck メソッド	111
3.3.19. setLog メソッド	112
3.3.20. getVersionCode メソッド	113
3.3.21. getVersionName メソッド	114
3.3.22. ErrorCodeExtended プロパティ	115
3.4. 注意事項	116
3.4.1. ログ機能について	116
3.4.2. 定数定義一覧	118
3.4.3. ラインディスプレイを利用する際の SDK バージョンについて	118
4. バーコードスキャナー制御	119
4.1. プログラム構造	119
4.2. 機能一覧	120

4.3. SDK インターフェース	121
4.3.1. 戻り値	121
4.3.2. インスタンス	122
4.3.3. connect メソッド	123
4.3.4. disconnect メソッド	125
4.3.5. setLog メソッド	126
4.3.6. getVersionCode メソッド	127
4.3.7. getVersionName メソッド	128
4.3.8. ErrorCodeExtended プロパティ	129
4.3.9. DataEvent イベント	130
4.3.10. StatusUpdateEvent イベント	131
4.4. 注意事項	132
4.4.1. ログ機能について	132
4.4.2. 定数定義一覧	134
4.4.3. バーコードスキャナーを利用する際の SDK バージョンについて	134

1. 概要

本ドキュメントは、iOS POS Print SDK のプログラムマニュアルです。

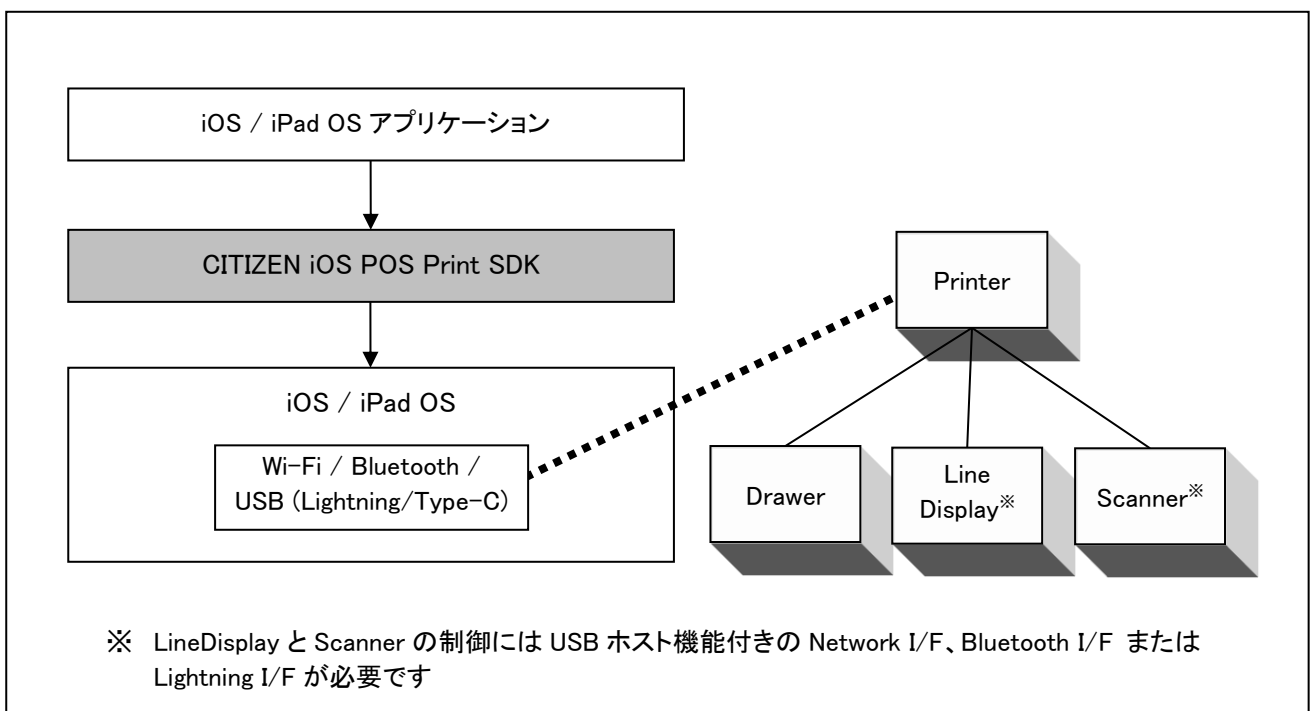
1.1. ドキュメント対象範囲

本ドキュメントは、CITIZEN POS プリンターおよびプリンターに接続された周辺機器を利用する iOS または iPad OS アプリケーション (Swift) の開発者が参照することを目的としています。

周辺機器の自動釣銭機を利用する場合は、別紙「CITIZEN iOS POS Print SDK (Swift) プログラムマニュアル (自動釣銭機用)」を参照してください。

1.2. システム概要

本 SDK は CITIZEN POS プリンターおよびプリンターに接続された周辺機器を利用する iOS または iPad OS アプリケーションから参照されることを想定しています。



SDK システム構成図

1.3. 対応端末

本 SDK が対応する端末の仕様は以下の通りです。

iOS バージョン :	対応インターフェース :	プライバシーマニフェスト対応
iOS 8.0 以上	Wi-Fi, Bluetooth	×
iOS 10.0.2 以上	USB(Lightning/Type-C)	×
iOS 12.0 以上	Wi-Fi, Bluetooth, USB(Lightning/Type-C)	○

1.4. 対応モデル(プリンター)

本 SDK の対応モデルおよびそのモデルに対応するインターフェースは以下の通りです。
各モデルの機能詳細についてはプリンターの取り扱い説明書をご参照ください。

モデル系列	対象モデル	インターフェース	プリンター機能
CT-E301 系	CT-E301	有線 LAN	標準
CT-E601 系	CT-E601	有線/無線 LAN Bluetooth USB(Lightning/Type-C)	標準
CT-S251 系	CT-S251	有線/無線 LAN Bluetooth USB(Lightning/Type-C)	標準
CT-S253 系	CT-S253	有線 LAN	標準
CT-S255 系	CT-S255	有線/無線 LAN Bluetooth	標準
	CT-S255-L	USB(Lightning/Type-C)	ラベル/ブラックマーク紙対応
CT-S257 系	CT-S257	有線/無線 LAN Bluetooth USB(Lightning/Type-C)	標準
CT-S281 系	CT-S281BD	Bluetooth	標準
	CT-S281BD-M		ブラックマーク紙対応
	CT-S281BD-L		ラベル紙対応
CT-S401 系	CT-S401	有線 LAN	標準
CT-S601/651/801/ 851 系	CT-S601/651/801/851	有線/無線 LAN	標準
	CT-S801/851-M		ブラックマーク紙対応
	CT-S801-L		ラベル紙対応
CT-S601II/651II/ 801II/851II 系	CT-S601II/651II/ 801II/851II	有線/無線 LAN Bluetooth	標準
	CT-S801II/851II-M		ブラックマーク紙対応
	CT-S801II-L		ラベル紙対応
CT-S801III/851III 系	CT-S801III/851III	有線/無線 LAN Bluetooth	標準
CT-S2000 系	CT-S2000	有線 LAN	標準
	CT-S2000-M		ブラックマーク紙対応
	CT-S2000-L		ラベル紙対応
CT-S4000 系	CT-S4000	有線 LAN	標準(表面ブラックマーク紙対応)
	CT-S4000-M		裏面ブラックマーク紙対応
	CT-S4000-L		ラベル紙対応
CT-S4500 系	CT-S4500	有線/無線 LAN Bluetooth USB(Lightning/Type-C)	標準(ラベル/ブラックマーク紙対応)
CMP-20/30/20II/30II/ 40 系	CMP-20/30/20II/ 30II/40 (ESC/POS)	無線 LAN, Bluetooth	標準

本 SDK をご使用になる際は、プリンターのメモリースイッチ設定が以下の通り設定されていることが条件となります。

CT-E301 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte

MSW No.	機能	設定
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5

CT-E601 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5-HKSCS
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

CT-S251 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字

MSW No.	機能	設定
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-3	漢字	有効 (*1)
9-4	JIS/シフト JIS	シフト JIS (*1)
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

CT-S253 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5

CT-S255 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM1000 互換モード	有効

MSW No.	機能	設定
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

CT-S257 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5-HKSCS
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

CT-S281 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM-270 互換モード	有効
3-8	印字中カバーオープン	自動復帰

4-8	強制パーシャル	無効
5-3	USB モード	プリンタクラス
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本(*1)
9-3	漢字	有効(*1)
9-4	JIS/シフト JIS	シフト JIS (*1)
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

CT-S401 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-3	漢字	有効 (*1)
9-4	JIS/シフト JIS	シフト JIS (*1)

CT-S601/651/801/851 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカット動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカット復帰	L/F 有効
3-3	パラレル 31Pin	リセット
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-3	漢字	有効 (*1)
9-4	JIS/シフト JIS	シフト JIS (*1)
10-3	ACK 出力タイミング	BUSY 前

CT-S601II/651II/801II/851II 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカッタ動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカッタ復帰	L/F 有効
3-3	パラレル 31Pin	リセット
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5
10-3	ACK 出力タイミング	BUSY 前
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

CT-S801III/851III 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカッタ動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカッタ復帰	L/F 有効
3-3	パラレル 31Pin	リセット
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5
10-3	ACK 出力タイミング	BUSY 前
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

CT-S2000 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効

MSW No.	機能	設定
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカッタ動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカッタ復帰	L/F 有効
3-3	パラレル 31Pin	リセット
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-3	漢字	有効 (*1)
9-4	JIS/シフト JIS	シフト JIS (*1)
10-3	ACK 出力タイミング	BUSY 前

CT-S4000 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカッタ動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカッタ復帰	L/F 有効
3-3	パラレル 31Pin	リセット
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-3	漢字	有効 (*1)
9-4	JIS/シフト JIS	シフト JIS (*1)
10-3	ACK 出力タイミング	BUSY 前

CT-S4500 系 メモリースイッチ設定

MSW No.	機能	設定
1-1	電源 ON 通知設定	有効
1-2	インプットバッファ	4K byte
1-3	Busy 条件	バッファフル
1-4	受信エラー文字	?文字
1-5	CR モード	無効
2-2	オートカッタ動作	有効
2-4	フル桁印字	データ待ち
3-1	オートカッタ復帰	L/F 有効
3-7	CBM1000 互換モード	有効
3-8	印字中カバーオープン	自動復帰

MSW No.	機能	設定
4-8	強制パーシャル	無効
5-2	縦基本計算ピッチ	360
5-3	USB モード	プリンタクラス
6-1	ドライバ用動作	有効
7-6	DMA 制御	有効
9-1	コードページ	Katakana (*1)
9-2	国際文字	日本 (*1)
9-4	漢字コード (*2)	SJIS(CP932) GB18030 EUC Hangul BIG5-HKSCS
13-6	再接続要求 (Bluetooth I/F 使用時)	有効

*1 MSW No.9-1～4 は、日本語使用時の設定です。ご使用環境に合わせて変更してください。

*2 CT-E301/601、CT-S253/255/257/601II/651II/801II/851II/801III/851III/4500系は、漢字コードをShift_JIS、GB18030、EUC-KR、Big5に切り替える事ができます。ご使用の環境に合わせて変更してください。

ファームウェア

CT-S601/651/801/851 系モデルにおいて本 SDK を正常に動作させるには、プリンターのファームウェアバージョンが下記の条件である必要があります。下記プリンターよりも古いプリンターをご使用の際は、ファームウェアをバージョンアップする必要があります。

機種	ファームウェアバージョン
CT-S601	DL00-2000 以降
CT-S651	DM00-2000 以降
CT-S801	DH00-2000 以降
CT-S851	DK00-2000 以降

1.5. 対応モデル(周辺機器)

本 SDK の周辺機器の対応モデルは以下の通りです。

各モデルの機能詳細については周辺機器の取扱説明書をご参照ください。

周辺機器の制御には、USB ホスト機能付きの Network I/F、Bluetooth I/F または Lightning I/F が必要です。

ラインディスプレイ

対象モデル	インターフェース	機能
DSP01-LT / DSP01-LT2	USB	TFT 型ラインディスプレイ
DSP02-LS / DSP02-LS2	USB	STN 型ラインディスプレイ

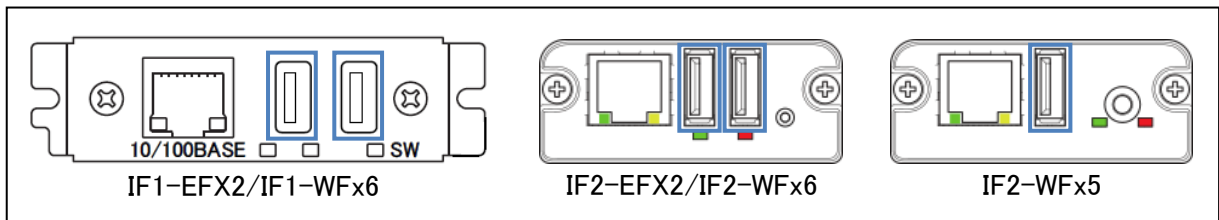
バーコードスキャナー

対象モデル	インターフェース	機能
SCN01-Z1D	USB	1 次元バーコードスキャナー
SCN02-Z2D	USB	2 次元バーコードスキャナー
BC-NL3000U	USB	2 次元バーコードスキャナー

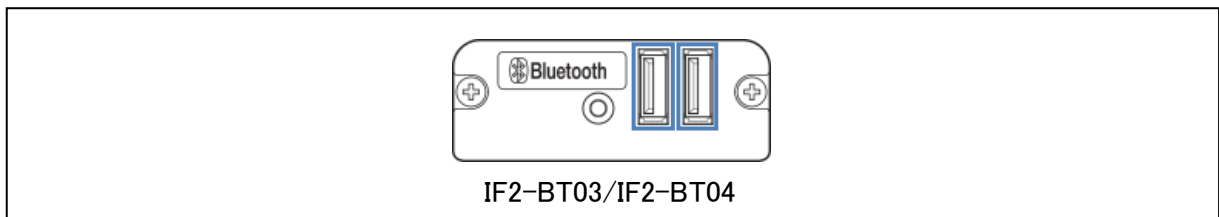
プリンターへの接続について

周辺機器の接続は、プリンター電源を一旦 OFF にしてから、下図に示すインターフェースの USB 端子に接続してご利用ください。その後、プリンター電源を ON にしてから、周辺機器が利用可能になるまで、安定動作のため、周辺機器の制御開始処理を 30 秒ほど待機してから実行してください。

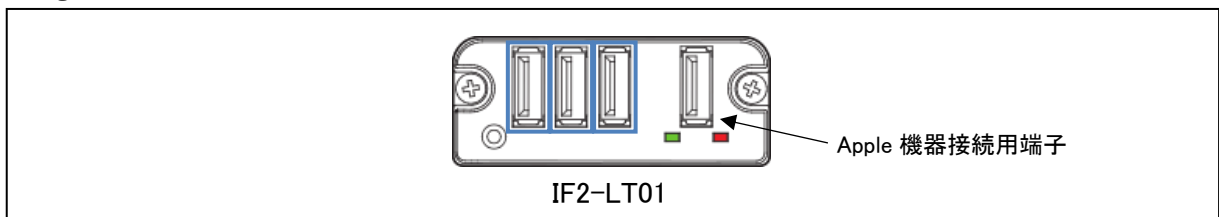
Network I/F



Bluetooth I/F



Lightning I/F



以下は、周辺機器接続に関しての、してはいけない「禁止」内容になります。

禁止事項

- 対応周辺機器以外(USB ハブやスマートフォン等)をインターフェースの USB 端子に接続
- プリンター電源が ON のまま、インターフェースの USB 端子から周辺機器のケーブルを挿抜
- 同種の周辺機器をインターフェースの USB 端子に複数接続 (例:ディスプレイを 2 台接続)
- Lightning I/F の周辺機器用 USB 端子に iPad/iPod/iPhone 接続

もし、上記事項を実施された場合、プリンターや接続周辺機器について、誤動作を招く原因や、最悪、故障の原因となりますので、おやめください。

Network I/F 設定について

Network I/F にラインディスプレイおよびバーコードスキャナーを接続して使用する場合、サービスに関する設定を変更する必要があります。基本的な操作につきましては、プリンターのインターフェースボード取り扱い説明書をご参照ください。

Web ブラウザーから各プリンターに接続して、以下の Service 画面を表示してください。プリンターが提供するサービスの設定を行います。

LAN board CITIZEN SYSTEMS

HOME | STATUS | CONFIG Logout

General Service User Account Maintenance

Media Converter

VCOM Convert ☒ Enable ☐ Disable ☐ Show configuration

HID Scanner Convert ☒ Enable ☐ Disable ☐ Show configuration

XML Print

Port Number 8080

Timeout for connect 10 5-60[Seconds]

Timeout for print 60 10-600[Seconds]

XML Device Control

Port Number 8085

Timeout for connect 10 5-180[Seconds]

Maxconnection 2

XML Device Control /Line Display

Baud rate 9600

Data 8 bit

Copyright © 2012 CITIZEN SYSTEMS JAPAN CO.,LTD. All rights reserved.

上記赤枠内を参照に、「VCOM Converter」と「HID Scanner Convert」の Enable を選択します。

その後、最下部までスクロールし「Submit」ボタンを押します。

最後に「Maintenance」タブの「Save&Reboot」ボタンを押し、「Yes」を選択し、プリンターから音が鳴ると設定完了です。

上記赤枠内の「Show configuration」にチェックを入れると「Media Converter Configuration / VCOM Convert」の設定画面が表示されますが、既に対応ディスプレイに対して適切値になっているので、通常利用では変更しないでください。

各設定値は、電源を切断しても値を保持します。工場初期設定(Factory Default)の処理が行われた時には、各設定値を初期値に設定します。

バーコードスキャナーの設定について

バーコードスキャナーを使用する場合は、以下の通り設定されている必要があります。

項目	値		説明
	Network または Bluetooth I/F	Lightning I/F	
Interface	USB HID Class	USB virtual COM	通信プロトコル
Keyboard	US Keyboard	US Keyboard	キーボード言語
Terminator	Enter	<CR><LF>	データのサフィックス

本 SDK でバーコードスキャナーをご使用になる際は、プリンターのインターフェースによりバーコードスキャナーの通信プロトコル設定を変更する必要があります。設定方法は以下の通りです。



・SCNO1-Z1D

下記のバーコードを上から順番に全てスキャンし、設定を変更してください。

Network および Bluetooth I/F	Lightning I/F
 設定開始	 設定開始
 USB HID 設定	 USB virtual COM 設定
 設定終了	 設定終了

・SCNO2-Z2D

下記のバーコードをスキャンし、設定を変更してください。

Network および Bluetooth I/F	Lightning I/F
 USB HID 設定	 USB virtual COM 設定

・BC-NL3000U

下記のバーコードを上から順番に全てスキャンし、設定を変更してください。

Network および Bluetooth I/F	Lightning I/F
 設定開始 0006010	 設定開始 0006010
 USB HID 設定 1100080	 USB virtual COM 設定 1100060
 設定終了 0006000	 設定終了 0006000

1.6. SDK 定義方法

SDK 種類

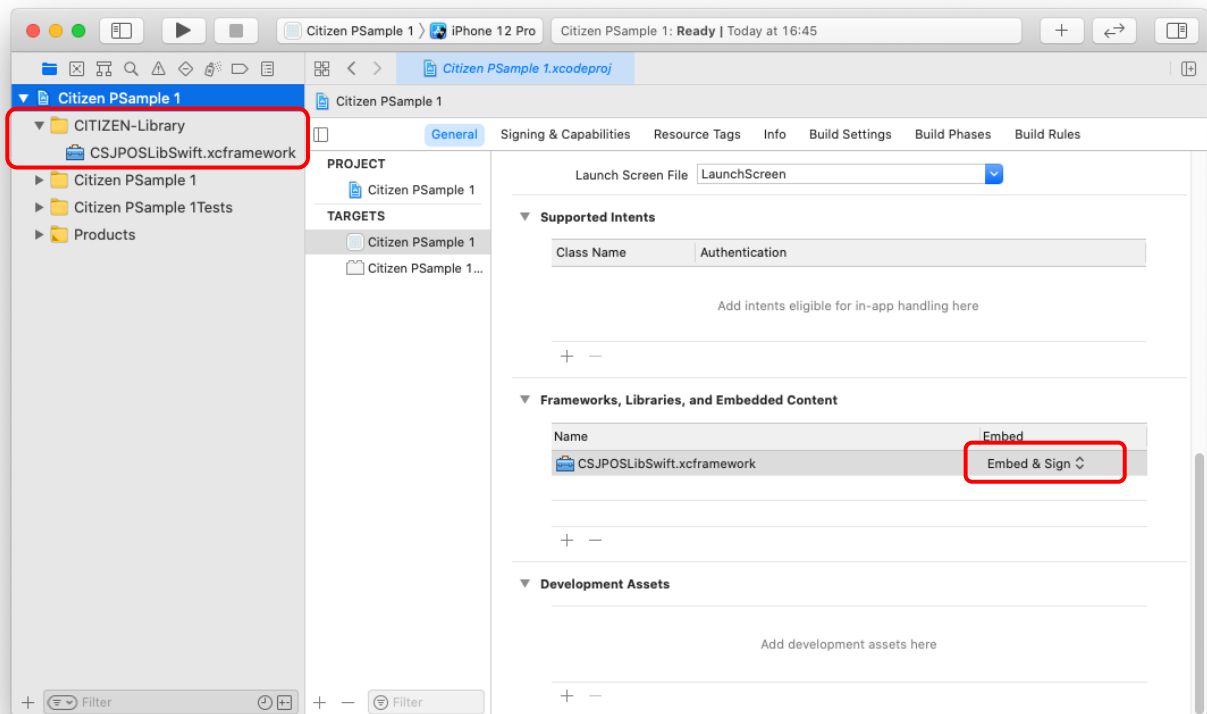
SDK(フレームワーク)を開発環境に合わせて選択してください。

SDK (Framework)	開発環境	対象 OS
Swift5.1 以降 (XCFramework)	Xcode11.0 以降	使用する XCode に依存
Swift5.3 以降 (XCFramework)	Xcode12.0 以降	使用する XCode に依存
Swift5.9 以降 プライバシーマニフェスト対応版 (XCFramework)	Xcode15.0 以降	iOS12.0 以降

※Version 2.11 より Swift5.0 以前の Framework の提供を停止しました。

Xcode に SDK を追加

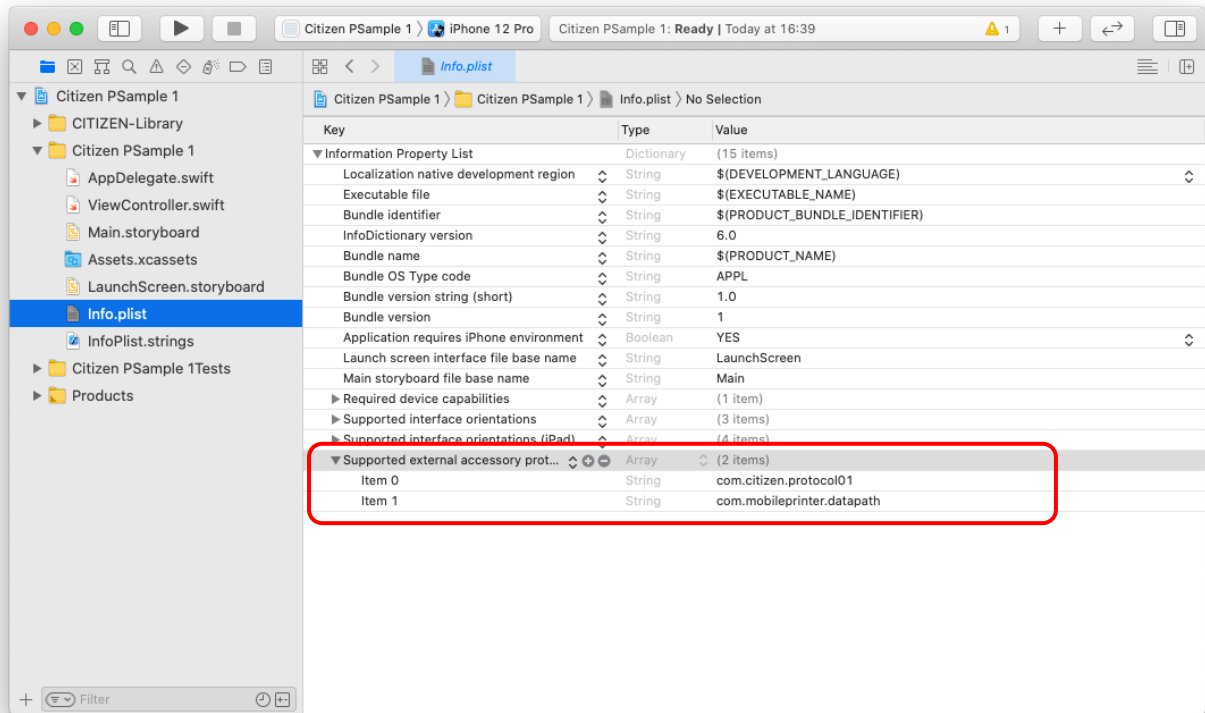
- ・開発するプロジェクトにフレームワーク (CSJPOSLibSwift.xcframework)を追加します。
- ・[Finder]から Xcode の[Project Navigator]の任意の場所へフレームワークをドラッグします。



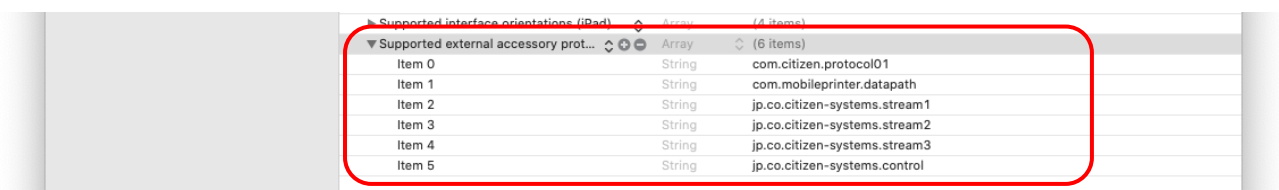
- ・ [Project Navigator]からプロジェクトファイルを選択し、[TARGET] → [General]の[Frameworks, Libraries, and Embedded Content]欄の追加したフレームワークの[Embed]を[Embed & Sign]へ変更します。

Bluetooth および USB(Lightning/Type-C)の開発を行うための設定

- Protocol name を追加します。[Project Navigator]から “Info.plist”ファイルを開きます。ポップアップメニュー (Control+クリックすると開きます)から Add Row を選び、“Supported external accessory protocols”を選択し追加します。その後“Supported external accessory protocols”を開き、“com.citizen.protocol01”と “com.mobileprinter.datapath”を追加します。



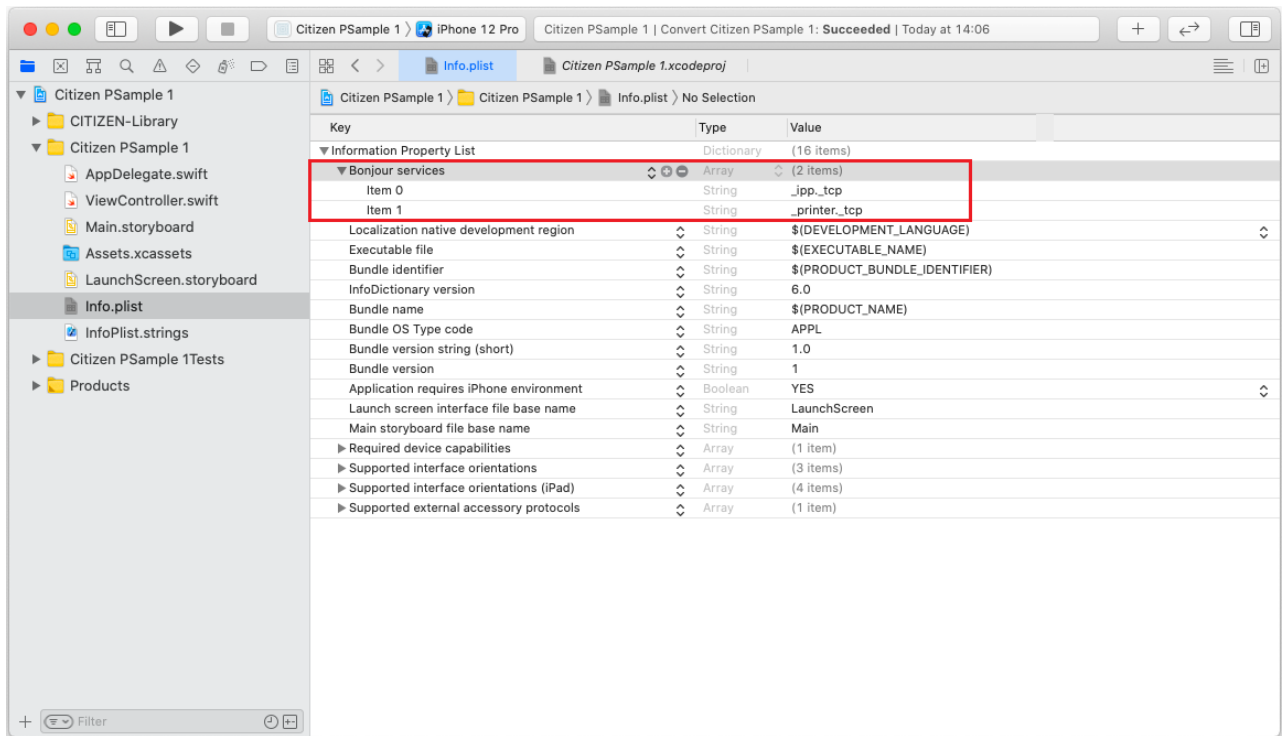
- 更に Lightning I/F で周辺機器を使用する場合は、必要に応じて、“jp.co.citizen-systems.stream1”、“jp.co.citizen-systems.stream2”、“jp.co.citizen-systems.stream3”そして“jp.co.citizen-systems.control”を追加してください。



以上で準備完了です。

LAN モデルの検索を行うための設定 (Bonjour)

- V209 以上の SDK を利用して LAN の検索を行うメソッド (searchESCPOSPrinter, searchCitizenPrinter)を使用する場合は、App に Bonjour サービスを利用するための設定を行ってください。
- [Project Navigator]から “Info.plist”ファイルを開きます。“Information Property List”をクリックして、“Bonjour service”を追加します。“Bonjour service”の “Item 0”に “_ipp_tcp”、“Item 1”に “_printer_tcp”を追加してください。



以上で準備完了です。

1.7. Bluetooth の利用方法

Bluetooth のペアリングと接続

プリンターと iOS デバイスで Bluetooth 通信をするには、iOS デバイスからプリンターを発見してペアリングをする必要があります。(以下の画面は iPod Touch/iOS7 での例です)

・プリンターの電源をオンにします。iOS デバイスの[設定]から[Bluetooth]を選び、周辺の Bluetooth 機器を検索します。Bluetooth がオフの場合、オンすると検索が始まります。



・しばらくすると周辺に有る Bluetooth 機器が表示されます。“CITIZEN SYSTEMS”と表示されている機器がプリンターです。初めて利用する場合ペアリングが必要です。既にペアリングされているプリンターは、そのまま接続が確立します。※



・ペアリングするには“CITIZEN SYSTEMS — ペアリングされていません”の箇所をタップします。しばらくすると接続が確立します。



※CT-E601, CT-S251/255/257/601II/651II/801II/851II/4500 を使用する場合
Bluetooth 機器表示名が” CITIZEN SYSTEMS”では無く、”機種名_(BD アドレスの下 2 桁)”となっています。



※再接続要求について

・プリンターの[メモリースイッチ 13-6]”再接続要求一有効” の場合、プリンター側から iOS デバイスへの接続を復元します。iOS の仕様上、Bluetooth 接続が切断される度に、Bluetooth 設定画面より接続を行う必要があります。この機能を有効すると、プリンター側が最後に接続していた iOS デバイスに対し、自動で接続を要求するため、この手間を省けます。

・複数の iOS デバイスとプリンターを交互に使用される場合は、この再接続要求によって却って利便性が低下しますので、無効の状態でご利用ください。

・iOS 以外のデバイスと接続される場合は、再接続要求を無効の状態でご利用ください。

2. プリンター制御

2.1. プログラム構造

本 SDK を使用する場合のプログラム構造は、以下の通りです。

```
var printer: ESCPOSPrinter? = CSJPOSLibSwift.ESCPOSPrinter()
var result: Int32 = 0

// プリンターへ接続
result = printer!.connect(ESCPOSConst.CMP_PORT_WiFi,
    withAddress: "192.168.0.10")

if ESCPOSConst.CMP_SUCCESS == result {
    // エンコード設定
    _ = printer!.setEncoding(String.Encoding.shiftJIS)

    // 一括処理開始設定
    _ = printer!.transactionPrint(ESCPOSConst.CMP_TP_TRANSACTION)

    // テキスト印刷
    _ = printer!.printText("- Sample Print 1 -\n",
        withAlignment: ESCPOSConst.CMP_ALIGNMENT_CENTER,
        withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
        withTextSize:
            ESCPOSConst.CMP_TXT_1WIDTH|ESCPOSConst.CMP_TXT_2HEIGHT)
    _ = printer!.printText("1234567890123456789012345678901234567890\n",
        withAlignment: ESCPOSConst.CMP_ALIGNMENT_RIGHT,
        withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
        withTextSize:
            ESCPOSConst.CMP_TXT_1WIDTH|ESCPOSConst.CMP_TXT_1HEIGHT)

    // QRCode 印刷
    _ = printer!.printQRCode("http://www.citizen-systems.co.jp",
        withModuleSize:6, withECLevel: CMP_QRCODE_EC_LEVEL_L,
        withAlignment: ESCPOSConst.CMP_ALIGNMENT_RIGHT)

    // カット位置送り後、パーシャルカット
    _ = printer!.cutPaper(ESCPOSConst.CMP_CUT_PARTIAL_PREFEED)

    // 一括処理終了設定
    result = printer!.transactionPrint(ESCPOSConst.CMP_TP_NORMAL)

    // 切断処理
    _ = printer!.disconnect()

    if ESCPOSConst.CMP_SUCCESS != result {
        messageBox("Transaction Error : \(result)", withTitle: "Error",
            withAutoDismiss: true)
    }
}else{
    // 接続失敗
    messageBox("Connect Error : \(result)", withTitle: "Error",
        withAutoDismiss: true)
}
```

} クラス定義

} 接続処理

} 印刷処理

} 切断処理

2.2. 機能一覧

本 SDK は以下の機能を提供します。

メソッド一覧

No	機能	詳細
1	クラス生成 (インスタンス)	インスタンスです。
2	プリンター接続処理 (connect メソッド)	プリンターと接続します。
3	プリンター切断処理 (disconnect メソッド)	プリンターとの接続を切断します。
4	エンコード設定処理 (setEncoding メソッド)	エンコードを設定します。
5	プリンター状態確認処理 (printerCheck メソッド)	ステータスチェックコマンドを送信します。
6	プリンター状態取得処理 (status メソッド)	プリンターのステータスを取得します。
7	印字処理 (printText メソッド)	テキストデータを印刷します。
8	空白埋め込み印字処理 (printPaddingText メソッド)	空白埋め込みテキストデータを印刷します。
9	Local フォント印字処理 (printTextLocalFont メソッド)	端末のフォントを使用してテキストデータを印刷します。
10	ビットマップ印刷処理 (printBitmap/printBitmapData メソッド)	画像ファイル[BMP/JPG/PNG/GIF ファイル]を印刷します。
11	NV ビットマップ登録処理 (setNVBitmap メソッド)	ビットマップ画像をフラッシュメモリへ登録します。
12	NV 登録ビットマップ印刷処理 (printNVBitmap メソッド)	フラッシュメモリに保存されたビットマップ画像を印刷します。
13	バーコード印刷処理 (printBarCode メソッド)	一次元バーコードを印刷します。
14	PDF-417 印刷処理 (printPDF417 メソッド)	PDF417 バーコードを印刷します。
15	QR コード印刷処理 (printQRCode メソッド)	QRCode バーコードを印刷します。
16	2次元 GS1DataBar 印刷処理 (printGS1DataBarStacked メソッド)	2次元 GS1DataBar バーコードを印刷します。
17	用紙カット処理 (cutPaper メソッド)	用紙をカットします。
18	ドット単位紙送り処理 (unitFeed メソッド)	ドット単位で紙送りします。
19	マーク紙送り処理 (markFeed メソッド)	ラベル/ブラックマーク紙用をサポートします。
20	ドロワー開処理 (openDrawer メソッド)	キャッシュドロワーを開けるコマンドを送信します。
21	一括処理開始/終了処理 (transactionPrint メソッド)	一括処理モードを開始/終了します。
22	回転印刷処理 (rotatePrint メソッド)	回転方向モード(180度)を開始/終了します。
23	ページモード開始/終了処理 (pageModePrint メソッド)	ページモードを開始/終了します。
24	ページモード印刷領域消去処理 (clearPrintArea メソッド)	ページモード印刷領域上の印刷データを消去します。
25	出力データクリア処理 (clearOutput メソッド)	処理中のデータおよびプリンターのバッファをクリアします。

26	データ出力処理 (printData メソッド)	データをそのままプリンターに送信します。
27	OPOS 形式印刷処理 (printNormal メソッド)	OPOS エスケープシーケンスを使用してテキストを印刷します。
28	透かし印刷開始／終了処理 (watermarkPrint メソッド)	透かし印刷を開始／終了します。
29	プリンター検索 (searchCitizenPrinter メソッド)	プリンターを検索しプリンターの情報のリストを取得します。
30	プリンター検索 (searchESCPOSPrinter メソッド)	プリンターを検索しアドレスのリストを取得します。
31	接続およびプリンター状態確認処理 (printerCheckEx メソッド)	接続とコマンド送信を行い、プリンターのステータスを取得します。
32	接続およびドロワーオープン処理 (openDrawerEx メソッド)	接続してキャッシュドロワーを開けるコマンドを送信します。
33	印刷完了通知タイムアウト設定 (setPrintCompletedTimeout メソッド)	印刷完了通知を確認するタイムアウトを設定します。
34	ログ設定 (setLog メソッド)	ログ機能を設定します。
35	バージョンコード取得 (getVersionCode メソッド)	SDK のバージョン番号を数値で取得します。
36	バージョン文字列取得 (getVersionName メソッド)	SDK のバージョン番号を文字列で取得します。

プロパティ一覧

No	機能	属性	詳細
1	ページモード領域取得 (PageModeArea プロパティ)	R	使用可能なページモード領域を示します。
2	ページモード印刷領域設定／取得 (PageModePrintArea プロパティ)	R/W	ページモード印刷領域を示します。
3	ページモード印刷方向設定／取得 (PageModePrintDirection プロパティ)	R/W	ページモード印刷領域内の印刷方向を示します。
4	印刷開始水平方向オフセット値設定／取得 (PageModeHorizontalPosition プロパティ)	R/W	ページモード印刷領域内の印刷開始位置の水平方向オフセット値を示します。
5	印刷開始垂直方向オフセット値設定／取得 (PageModeVerticalPosition プロパティ)	R/W	ページモード印刷領域内の印刷開始位置の垂直方向オフセット値を示します。
6	行間隔設定／取得 (RecLineSpacing プロパティ)	R/W	通常文字の印刷行の高さを示します。
7	マッピングモード設定／取得 (MapMode プロパティ)	R/W	プリンターのマッピングモード(長さの単位)を示します。
8	拡張エラーコード取得 (ErrorCodeExtended プロパティ)	R	接続時の拡張エラーコードを示します。

2.3. SDK インターフェース

本 SDK のインターフェースを以下に示します。

2.3.1. 戻り値

以降に示すメソッドは、下記の戻り値を返します。

戻り値	説明
CMP_SUCCESS (0)	正常終了
CMP_E_CONNECTED (1001)	プリンターへ既に接続済みです。
CMP_E_DISCONNECT (1002)	プリンターへ接続していません。
CMP_E_NOTCONNECT (1003)	プリンターへ接続できませんでした。
CMP_E_CONNECT_NOTFOUND (1004)	プリンター接続後の対応機種確認に失敗しました。
CMP_E_CONNECT_OFFLINE (1005)	プリンター接続後のプリンター状態確認に失敗しました。
CMP_E_ILLEGAL (1101)	サポートされていない処理または無効なパラメータ値です。
CMP_E_OFFLINE (1102)	プリンターがオフラインです。
CMP_E_NOEXIST (1103)	指定のファイルが存在しません。
CMP_E_FAILURE (1104)	要求された処理が実行できません。
CMP_E_TIMEOUT (1105)	所定の時間が経過してもプリンターからの応答がありません。
CMP_E_NO_LIST (1106)	プリンター検索にてプリンターが見つかりません。
CMP_EPTR_COVER_OPEN (1201)	プリンターのカバーが開いています。
CMP_EPTR_REC_EMPTY (1202)	用紙切れです。
CMP_EPTR_BADFORMAT (1203)	指定されたファイルの書式がサポートされていません。
CMP_EPTR_TOOBIG (1204)	指定されたビットマップのサイズが大きすぎます。

2.3.2. インスタンス

形式

ESCPOSPrinter

説明

クラスを生成し、初期化します。

使用例

```
var escp: ESCPOSPrinter? = CSJPOSLibSwift.ESCPOSPrinter()
```

2.3.3. connect メソッド

形式

- 1) func connect(connectType: Int32, withAddress addr: String?) -> Int32
- 2) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32) -> Int32
- 3) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32, withTimeout timeout: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
connectType	[IN]	接続タイプ	CMP_PORT_WiFi: Wi-Fi 接続 CMP_PORT_BLUETOOTH: Bluetooth 接続 CMP_PORT_USB: USB 接続 (Lightning/Type-C)
addr	[IN]	接続先の IP アドレス または Bluetooth デバイスアドレス または Bluetooth デバイス名 またはシリアル番号	WiFi : 0.0.0.0 ~ 255.255.255.255 Bluetooth : 00:00:00:00:00:00 ~ FF:FF:FF:FF:FF:FF デバイス名 USB : 空白またはシリアル番号
port	[IN]	接続先ポート番号	
timeout	[IN]	タイムアウト (msec)	

説明

このメソッドは、プリンターと接続するために使用します。プリンターの接続タイプとアドレスを指定してください。

Bluetooth 接続の場合、iOS デバイスの Bluetooth 設定画面にてペアリング済でかつ接続された状態のプリンターにだけ接続する事が出来ます(「[1.7 Bluetooth の利用方法](#)」を参照してください)。また Bluetooth デバイスアドレスを利用して接続するには iOS6 以上である必要があります。それ以外は Bluetooth デバイス名を利用して接続してください。

USB 接続の場合、アドレスに空白を指定した場合は自動的に接続されます。プリンターのシリアル番号を指定して特定のプリンターへ接続する事も可能です。

接続先ポート番号は、接続タイプに Wi-Fi を指定した場合のみ有効です。省略された場合は、9100 番で接続します。

タイムアウトは、プリンターへの接続の最大時間(ミリ秒単位)を指定します。省略された場合は、Wi-Fi/USB の場合は 4000 ミリ秒、Bluetooth の場合は 8000 ミリ秒で接続します。

プリンターと接続した際に、プリンターのステータスと対応機種を同時に確認します。

プリンターとの通信が不要になった場合は、必ず [disconnect メソッド](#) を実行し、プリンターとの接続を切断してください。切断しなかった場合は、次の接続がエラーとなります。

戻り値

成功時は CMP_SUCCESS(0)を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

エラーコード	説明
CMP_E_NOTCONNECT (1003)	デバイスへ接続できませんでした。 ①プリンターが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない

CMP_E_CONNECT_NOTFOUND (1004)	デバイス接続後に取得した対応機種に異常を確認しました。 ①対応機種でない
CMP_E_CONNECT_OFFLINE (1005)	デバイス接続後に取得したステータスに異常を確認しました。 プリンターに次のエラーが発生している可能性があります。 ①カバーが開いている ②用紙が無い ③紙ジャム等によりオートカッターエラーが発生 ④回路故障等による復帰不可能エラーが発生

使用例

```
escp!.connect(ESCPOSConst.CMP_PORT_WiFi, withAddress: "192.168.182.100",  
              withPort: 9100)  
  
escp!.connect(ESCPOSConst.CMP_PORT_BLUETOOTH, withAddress:  
              "00:01:90:F0:81:AB", withPort: 9100, withTimeout:8000)  
  
escp!.connect(ESCPOSConst.CMP_PORT_BLUETOOTH,  
              withAddress: "CITIZEN SYSTEMS", withPort: 9100, withTimeout:8000)  
  
escp!.connect(ESCPOSConst.CMP_PORT_USB, withAddress: "")
```

2.3.4. disconnect メソッド

形式

```
func disconnect() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、プリンターとの接続を切断するために使用します。

印刷の終了、あるいは、何らかのエラーが発生した場合は、本メソッドを実行して接続を切断してください。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.disconnect()
```


2.3.5. setEncoding メソッド

形式

```
func setEncoding(charset: NSStringEncoding) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
charset	[IN]	文字セット名	

説明

このメソッドは、プリンター送信データのエンコードを設定するために使用します。

インスタンスを生成時に OS のデフォルト文字セットに初期化します。

設定するエンコードはプリンターのメモリースイッチの設定に合わせてください(「[1.4 対応モデル\(プリンター\)](#)」参照)。

本 SDK は、UTF-8 でエンコードされた文字列の印刷をサポートします。詳しくは、「[2.4.2 UTF-8 エンコード文字列の印刷について](#)」を参照してください。

設定可能範囲の詳細につきましては以下の URL を参照ください。

http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSString_Class/Reference/NSString.html#//apple_ref/doc/uid/20000154-BAJJAICE

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
// 日本語
```

```
escp!.setEncoding(String.Encoding.shiftJIS)
```

```
// 中国語(Simplified Chinese)
```

```
let encode = CFStringConvertEncodingToNSStringEncoding( 0x0632 )
```

```
escp!.setEncoding(String.Encoding(rawValue:encode))
```

```
// 中国語(Traditional Chinese)
```

```
let encode = CFStringConvertEncodingToNSStringEncoding( 0x0A03 )
```

```
escp!.setEncoding(String.Encoding(rawValue:encode))
```

```
// 韓国語
```

```
let encode = CFStringConvertEncodingToNSStringEncoding( 0x0940 )
```

```
escp!.setEncoding(String.Encoding(rawValue:encode))
```

```
// UTF-8
```

```
escp!.setEncoding(String.Encoding.utf8)
```

2.3.6. printerCheck メソッド

形式

```
func printerCheck() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、プリンターのステータス取得コマンドを送信するために使用します。
本メソッドの実行結果が成功の場合は、[status メソッド](#)でプリンターのステータスを取得できます。
本メソッドの実行結果が失敗の場合は、通信異常やデバイスの異常が発生した可能性があります。この場合、[disconnect メソッド](#)および [connect メソッド](#)を使用して再接続してください。

接続後に時間を空けて印刷する場合は、必ず事前に本メソッドと [status メソッド](#)を実行してプリンターのステータスを確認してください。

ネットワーク接続の場合、長時間放置すると自動的に切断されます。接続を保持する場合は、定期的に本メソッドを実行してください。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
if ESCPOSConst.CMP_SUCCESS == escp!.printerCheck() {  
    // Success  
} else {  
    // Fail  
}
```

2.3.7. status メソッド

形式

- 1) func status() -> Int32
- 2) func status(musk: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
musk	[IN]	ステータス種類マスク	CMP_STS_PRINTEROFF CMP_STS_MSR_READ CMP_STS_PAPER_EMPTY CMP_STS_COVER_OPEN CMP_STS_BATTERY_LOW CMP_STS_PAPER_NEAREMPTY CMP_STS_DRAWER_LEVEL_H

説明

このメソッドは、[printerCheck メソッド](#)によって取得されたプリンターのステータスを確認するために使用します。

本メソッドの実行前に、[printerCheck メソッド](#)を実行する必要があります。

引き数が指定されなかった場合は、プリンターのエラーを示すステータス(CMP_STS_COVER_OPEN、CMP_STS_PAPER_EMPTY、CMP_STS_PRINTEROFF)の論理和を返します。

ステータス種類が指定された場合は、該当するステータスを返します。ステータス種類は組み合わせて指定できます。組み合わせる場合は論理和を指定してください。

戻り値

下記のステータスコードを返します。

ステータスコード	説明
CMP_STS_NORMAL (0)	プリンターは正常です。
CMP_STS_PRINTEROFF (128)	プリンターはオフラインです。
CMP_STS_MSR_READ (64)	MSR 読み取りモード状態です。(CMP-20/30/20II/30II/40 のみ)
CMP_STS_PAPER_EMPTY (32)	用紙がありません。
CMP_STS_COVER_OPEN (16)	プリンタカバーが開いています。
CMP_STS_BATTERY_LOW (8)	プリンターバッテリー容量が低下しました。(CMP-20/30/20II/30II/40 のみ)
CMP_STS_PAPER_NEAREMPTY (4)	ニアエンプティです。(type 指定時のみ)
CMP_STS_DRAWER_LEVEL_H (2)	ドロワーキックコネクタ3番ピンの状態が H です。(type 指定時のみ)

使用例

```
var status = escp!.status()
if ESCPOSConst.CMP_STS_NORMAL == status {
    // No Error
    var status2 = escp!.status(ESCPOSConst.CMP_STS_PAPER_NEAREMPTY)
    if (ESCPOSConst.CMP_STS_PAPER_NEAREMPTY & status2) > 0 {
        // Paper Near Empty
    }
} else {
    if (ESCPOSConst.CMP_STS_COVER_OPEN & status) > 0 {
        // Cover Open
    }
    if (ESCPOSConst.CMP_STS_PAPER_EMPTY & status) > 0 {
```

```
        // Paper Empty
    }
    if (ESCPOSConst.CMP_STS_PRINTEROFF & status) > 0 {
        // Printer Offline
    }
}

var status3 = escp!.status(ESCPOSConst.CMP_STS_DRAWER_LEVEL_H)
if (ESCPOSConst.CMP_STS_DRAWER_LEVEL_H & status3) > 0 {
    // Status of pin 3 of drawer kick-out connector = H
}
```

2.3.8. printText メソッド

形式

```
func printText(data: String?, withAlignment alignment: Int32, withAttribute attribute: Int32,
    withTextSize textSize: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	テキストデータ	
alignment	[IN]	テキスト配置	CMP_ALIGNMENT_LEFT: 左揃え CMP_ALIGNMENT_CENTER: 中央揃え CMP_ALIGNMENT_RIGHT: 右揃え
attribute	[IN]	テキスト属性	CMP_FNT_DEFAULT: 標準フォント CMP_FNT_FONTB: フォント B CMP_FNT_FONTC: フォント C CMP_FNT_BOLD: 太字 CMP_FNT_REVERSE: 反転 CMP_FNT_UNDERLINE: 下線
textSize	[IN]	テキストサイズ	CMP_TXT_1WIDTH: 幅 1 倍 CMP_TXT_2WIDTH: 幅 2 倍 CMP_TXT_3WIDTH: 幅 3 倍 CMP_TXT_4WIDTH: 幅 4 倍 CMP_TXT_5WIDTH: 幅 5 倍 CMP_TXT_6WIDTH: 幅 6 倍 CMP_TXT_7WIDTH: 幅 7 倍 CMP_TXT_8WIDTH: 幅 8 倍 CMP_TXT_1HEIGHT: 高さ 1 倍 CMP_TXT_2HEIGHT: 高さ 2 倍 CMP_TXT_3HEIGHT: 高さ 3 倍 CMP_TXT_4HEIGHT: 高さ 4 倍 CMP_TXT_5HEIGHT: 高さ 5 倍 CMP_TXT_6HEIGHT: 高さ 6 倍 CMP_TXT_7HEIGHT: 高さ 7 倍 CMP_TXT_8HEIGHT: 高さ 8 倍

説明

このメソッドは、テキストの配置、属性、サイズを指定して、テキストを印刷するために使用します。

テキスト属性は、フォント B、フォント C、太字、反転、下線を組み合わせて指定可能です。組み合わせる場合は論理和を指定してください。

テキストサイズは、幅と高さを組み合わせて指定可能です。組み合わせる場合は論理和を指定してください。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.printText("Print text data.\n",
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_CENTER,
    withAttribute: ESCPOSConst.CMP_FNT_BOLD,
    withTextSize:
        ESCPOSConst.CMP_TXT_2WIDTH|ESCPOSConst.CMP_TXT_2HEIGHT)
```

2.3.9. printPaddingText メソッド

形式

```
func printPaddingText(data: String?, withAttribute attribute: Int32, withTextSize textSize: Int32,
    withLength length: Int32, withSide side: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	テキストデータ	
attribute	[IN]	テキスト属性	CMP_FNT_DEFAULT: 標準フォント CMP_FNT_FONTB: フォント B CMP_FNT_FONTC: フォント C CMP_FNT_BOLD: 太字 CMP_FNT_REVERSE: 反転 CMP_FNT_UNDERLINE: 下線
textSize	[IN]	テキストサイズ	CMP_TXT_1WIDTH: 幅 1 倍 CMP_TXT_2WIDTH: 幅 2 倍 CMP_TXT_3WIDTH: 幅 3 倍 CMP_TXT_4WIDTH: 幅 4 倍 CMP_TXT_5WIDTH: 幅 5 倍 CMP_TXT_6WIDTH: 幅 6 倍 CMP_TXT_7WIDTH: 幅 7 倍 CMP_TXT_8WIDTH: 幅 8 倍 CMP_TXT_1HEIGHT: 高さ 1 倍 CMP_TXT_2HEIGHT: 高さ 2 倍 CMP_TXT_3HEIGHT: 高さ 3 倍 CMP_TXT_4HEIGHT: 高さ 4 倍 CMP_TXT_5HEIGHT: 高さ 5 倍 CMP_TXT_6HEIGHT: 高さ 6 倍 CMP_TXT_7HEIGHT: 高さ 7 倍 CMP_TXT_8HEIGHT: 高さ 8 倍
length	[IN]	半角文字相当の長さ	1～
side	[IN]	空白向け込みサイド	CMP_SIDE_RIGHT: テキストデータの右側 CMP_SIDE_LEFT: テキストデータの左側

説明

このメソッドは、テキストの属性、サイズ、半角文字相当の長さ、空白埋め込みサイドを指定して、空白が埋め込まれたテキストを印刷するために使用します。

テキストデータに、結合文字を使用する事はできません。

テキスト属性は、フォント B、フォント C、太字、反転、下線を組み合わせて指定可能です。組み合わせる場合は論理和を指定してください。

テキストサイズは、幅と高さを組み合わせて指定可能です。組み合わせる場合は論理和を指定してください。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
let nameSize = 24    // 商品名サイズ
let priceSize = 7    // 価格サイズ

// 1行目
escp!.printPaddingText("Sandwich",
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH, withLength: nameSize,
    withSide: ESCPOSConst.CMP_SIDE_RIGHT)
escp!.printPaddingText("5.00", withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH, withLength: priceSize,
    withSide: ESCPOSConst.CMP_SIDE_LEFT)
escp!.printNormal("\n")

// 2行目
escp!.printPaddingText("Hamburg steak",
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH, withLength: nameSize,
    withSide: ESCPOSConst.CMP_SIDE_RIGHT)
escp!.printPaddingText("12.00", withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH, withLength: priceSize,
    withSide: ESCPOSConst.CMP_SIDE_LEFT)
escp!.printNormal("\n")

// 3行目
escp!.printPaddingText("Coffee",
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH, withLength: nameSize,
    withSide: ESCPOSConst.CMP_SIDE_RIGHT)
escp!.printPaddingText("2.00", withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH, withLength: priceSize,
    withSide: ESCPOSConst.CMP_SIDE_LEFT)
escp!.printNormal("\n")
```

2.3.10. printTextLocalFont メソッド

形式

```
func printTextLocalFont(data: String?, withAlignment alignment: Int32,
    withFontName fontName: String?, withPoint point: Int32, withStyle style: Int32,
    withHRatio hRatio: Int32, withVRatio vRatio: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	テキストデータ	
alignment	[IN]	テキスト配置	CMP_ALIGNMENT_LEFT: 左揃え CMP_ALIGNMENT_CENTER: 中央揃え CMP_ALIGNMENT_RIGHT: 右揃え
fontName	[IN]	テキストフォント名	サポートされるフォントは、OS の実装によって異なります。
point	[IN]	テキストサイズ [単位ポイント]	1～
style	[IN]	テキストスタイル	CMP_FNT_DEFAULT: デフォルト CMP_FNT_BOLD: 太字 CMP_FNT_REVERSE: 反転 CMP_FNT_UNDERLINE: 下線 CMP_FNT_ITALIC: イタリック CMP_FNT_STRIKEOUT: 取り消し線
hRatio	[IN]	水平拡大率 [%]	1～1000
vRatio	[IN]	垂直拡大率 [%]	1～1000

説明

このメソッドは、テキストの配置、フォント、サイズ、スタイル、拡大率を指定して、端末にインストールされたフォントを使用してテキストを印刷するために使用します。

このメソッドを使用すると、入力された内容を端末で画像化し、その画像データを印刷します。

スタイルは、太字、反転、下線、イタリック、取り消し線を組み合わせて指定可能です。組み合わせる場合は論理和を指定してください。なお、イタリック指定は iOS13 以降では無視されます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.printTextLocalFont("Print local font text data.\n",
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_CENTER,
    withFontName: "Arial Hebrew",
    withPoint: 12,
    withStyle: ESCPOSConst.CMP_FNT_BOLD|ESCPOSConst.CMP_FNT_UNDERLINE,
    withHRatio: 100,
    withVRatio: 100)
```


2.3.11. printBitmap/printBitmapData メソッド

形式

- 1) func printBitmap(fileName: String?, withAlignment alignment: Int32) -> Int32
- 2) func printBitmap(fileName: String?, withWidth width: Int32, withAlignment alignment: Int32) -> Int32
- 3) func printBitmap(fileName: String?, withWidth width: Int32, withAlignment alignment: Int32, withMode mode: Int32) -> Int32
- 4) func printBitmapData(imageData: UIImage?, withAlignment alignment: Int32) -> Int32
- 5) func printBitmapData(imageData: UIImage?, withWidth width: Int32, withAlignment alignment: Int32) -> Int32
- 6) func printBitmapData(imageData: UIImage?, withWidth width: Int32, withAlignment alignment: Int32, withMode mode: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
fileName	[IN]	ビットマップファイル名	
imageData	[IN]	ビットマップデータ	
width	[IN]	ビットマップ印刷幅	CMP_BM_ASIS: プリンターのドット当たり1ビットマップピクセルでビットマップを印刷します。 上記定数以外の1以上の値: ビットマップ幅を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
alignment	[IN]	ビットマップ配置位置	CMP_ALIGNMENT_LEFT: 左揃え CMP_ALIGNMENT_CENTER: 中央揃え CMP_ALIGNMENT_RIGHT: 右揃え 上記定数以外の3以上の値: ビットマップ印刷を開始する左からの距離を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
mode	[IN]	ビットマップモード	CMP_BM_MODE_HT_THRESHOLD: ハーフトーン しきい値 CMP_BM_MODE_HT_DITHER: ハーフトーン ディザー CMP_BM_MODE_CMD_RASTER: モノクロラスタ ー コマンド出力 CMP_BM_MODE_CMD_BITIMAGE: モノクロビットイメージ コマンド出力 CMP_BM_MODE_CMD_GRAY16: グレースケール(16 階調)出力 CMP_BM_MODE_CMD_GRAY16DOWNLOAD: グレースケール(16 階調)ダウンロードグラフィックスコマンド出力

説明

このメソッドは、ビットマップのファイル名あるいはデータ、印刷幅、配置位置、モードを指定して、ビットマップを印刷するために使用します。

ビットマップファイル名はフルパスを指定します。

印刷可能なビットマップ形式は、BMP/JPG/PNG/GIF です。

ビットマップの印刷幅を省略された場合は、CMP_BM_ASIS で印刷します。

ビットマップモードは、ハーフトーンと出力方法を組み合わせて指定可能です。組み合わせる場合は、論理和を指定してください。モードが省略された場合は、CMP_BM_MODE_HT_THRESHOLD | CMP_BM_MODE_CMD_RASTER で印刷します。モードに関する詳細は、次の通りです。

ハーフトーン モノクロ印刷／グレースケール印刷時のハーフトーン処理方法を指定します。

設定値	説明
CMP_BM_MODE_HT_THRESHOLD	しきい値 文字の印刷に適します。
CMP_BM_MODE_HT_DITHER	ディザー グラフィックの印刷に適します。

出力 印刷時の出力方法を指定します。

設定値	説明
CMP_BM_MODE_CMD_RASTER	モノクロラスター コマンド出力 小さなデータの印刷に適します。データを一括して出力するため、高さ制限(2,304ドット 約 28cm)があります。
CMP_BM_MODE_CMD_BITIMAGE	モノクロビットイメージ コマンド出力 大きなデータの印刷に適します。データを分割して出力するため、高さ制限はありません。
CMP_BM_MODE_CMD_GRAY16	グレースケール出力 CT-E601, CT-S251/255/257/601II/651II/801II/851II/801III/851III 系で利用可能です。より綺麗なグラフィックの印刷が行えます。
CMP_BM_MODE_CMD_GRAY16DOWNLOAD	グレースケール ダウンロードグラフィックス コマンド出力 CT-E601, CT-S251/255/257/601II/651II/801II/851II/801III/851III 系で利用可能です。より綺麗なグラフィックの印刷を行います。データを一括して出力するため、16 階調のサイズで 384KB の制限があります。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

•Resource 内ファイル指定の場合

```
let FileName = NSBundle.mainBundle().pathForResource("sample_1.jpg",
    ofType: nil)
var errCode = escp!.printBitmap(FileName,
    withWidth: ESCPOSConst.CMP_BM_ASIS,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT)
```

•Image フォルダ内ファイル指定の場合

```
// Documents フォルダパス取得
let docDir = NSSearchPathForDirectoriesInDomains(.DocumentDirectory,
    .UserDomainMask, true)[0] as! String
// Images フォルダパス取得
let image_path = docDir + "/Images"
let FileName = image_path + "/sample_1.jpg"
var errCode = escp!.printBitmap(FileName,
    withWidth: ESCPOSConst.CMP_BM_ASIS,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT)
```

2.3.12. setNVBitmap メソッド

形式

- 1) func setNVBitmap(number: Int32, withFileName fileName: String?, withWidth width: Int32) -> Int32
- 2) func setNVBitmap(number: Int32, withFileName fileName: String?, withWidth width: Int32, withMode mode: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
number	[IN]	プリンターのフラッシュメモリ内に格納する画像番号	1～20
fileName	[IN]	登録するビットマップファイル名	
width	[IN]	ビットマップ登録幅	CMP_BM_ASIS: プリンターのドット当たり1ビットマップピクセルでビットマップを印刷します。 上記定数以外の1以上の値: ビットマップ幅を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
mode	[IN]	ビットマップモード	CMP_BM_MODE_HT_THRESHOLD: ハーフトーン しきい値 CMP_BM_MODE_HT_DITHER: ハーフトーン ディザー CMP_BM_MODE_CMD_MONO モノクロ登録 CMP_BM_MODE_CMD_GRAY16: グレイ(16階調)登録

説明

このメソッドは、画像番号、ビットマップのファイル名、印刷幅、配置位置、モードを指定して、プリンターのフラッシュメモリにビットマップ画像(ロゴ)を保存するために使用します。保存したロゴは、[printNVBitmap メソッド](#)や [watermarkPrint メソッド](#)を使用して印刷できます。

登録するビットマップファイル名はフルパスを指定します。

登録可能なビットマップファイル形式は、BMP/JPG/PNG/GIF です。

ビットマップの登録幅を省略された場合は、CMP_BM_ASIS で登録します。

ビットマップモードは、ハーフトーンと登録方法を組み合わせて指定可能です。組み合わせる場合は、論理和を指定してください。モードが省略された場合は、CMP_BM_MODE_HT_THRESHOLD | CMP_BM_MODE_CMD_MONO で登録します。モードに関する詳細は、次の通りです。

ハーフトーン モノクロ登録／グレースケール登録時のハーフトーン処理方法を指定します。

設定値	説明
CMP_BM_MODE_HT_THRESHOLD	しきい値 文字の印刷に適します。
CMP_BM_MODE_HT_DITHER	ディザー グラフィックの印刷に適します。

登録

登録方法を指定します。

設定値	説明
CMP_BM_MODE_CMD_MONO	モノクロ登録

CMP_BM_MODE_CMD_GRAY16	<p>グレースケール登録</p> <p>CT-E601, CT-S251/255/257/601II/651II/801II/851II/801III/851III 系で利用可能です。より綺麗なグラフィックの登録が行えます。</p>
------------------------	--

[CT-S281, CMP-20/30/20II/30II/40 系]

画像番号1から順に登録する必要があります。接続後、新たに登録すると前に登録した画像は消去されます。

[CT-E301/601, CT-S251/253/255/257/401/601/651/801/851/601II/651II/801II/851II/801III/851III/2000/4000/4500 系]

画像番号に関係なく登録できます。また、fileName パラメータに空文字列を設定することによって登録済みの画像を消去できます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
let FileName = NSBundle.mainBundle().pathForResource("sample_1.jpg",
    ofType: nil)
var errCode = escp!.setNVBitmap(1, withFileName: FileName,
    withWidth: ESCPOSConst.CMP_BM_ASIS,
    withMode:
        ESCPOSConst.CMP_BM_MODE_HT_DITHER|ESCPOSConst.CMP_BM_MODE_CMD_GRAY16
)
```

2.3.13. printNVBitmap メソッド

形式

```
func printNVBitmap(nvImageNumber: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
nvImageNumber	[IN]	プリンターのフラッシュメモリ内に格納されている画像番号	1～20

説明

このメソッドは、プリンターのフラッシュメモリに保存されているビットマップ画像(ロゴ)を印刷するために使用します。

このメソッドを使用するためには、事前にロゴの登録が必要です。ロゴ登録は、[setNVBitmap メソッド](#)で登録するか、プリンター用ユーティリティソフトウェアの「POS プリンターユーティリティ」を使用してください。

「POS プリンターユーティリティ」でロゴを登録する場合は、プリンターのモデルによって、登録モードが異なります。次の通り登録してください。

[CT-S281, CMP-20/30/20II/30II/40 系]

キーコード未使用モードでロゴ登録してください。

使用する画像番号に合わせて、順にロゴ登録する必要があります。

[CT-E301/601, CT-S251/253/255/257/401/601/651/801/851/601II/651II/801II/851II/801III/851III/2000/4000/4500 系]

キーコードモードでロゴ登録してください。

使用する画像番号に合わせて、キーコードを指定して登録する必要があります。

画像番号に対応するキーコードは次の通りです。

画像番号	キーコード (文字列)
1	"01"
2	"02"
3	"03"
.	.
.	.
.	.
19	"19"
20	"20"

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.printNVBitmap(1)
```

2.3.14. printBarCode メソッド

形式

```
func printBarCode(data: String?, withSymbology symbology: Int32, withHeight height: Int32,
    withWidth width: Int32, withAlignment alignment: Int32, withTextPosition textPosition: Int32)
    -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	印刷するバーコードデータ	
symbology	[IN]	バーコードタイプ	CMP_BCS_UPCA: UPC-A CMP_BCS_UPCE: UPC-E CMP_BCS_EAN8: EAN8(=JAN8) CMP_BCS_JAN8: JAN8(=EAN8) CMP_BCS_EAN13: EAN13(=JAN13) CMP_BCS_JAN13: JAN13(=EAN13) CMP_BCS_ITF: Interleaved 2 of 5 CMP_BCS_Codabar: Codabar CMP_BCS_Code39: コード 39 CMP_BCS_Code93: コード 93 CMP_BCS_Code128: コード 128 CMP_BCS_GS1DATABAR: GS1 DataBar Omnidirectional CMP_BCS_GS1DATABAR_E: GS1 DataBar Expanded CMP_BCS_GS1DATABAR_T: GS1 DataBar Truncated CMP_BCS_GS1DATABAR_L: GS1 DataBar Limited
height	[IN]	バーコード高さ	1～255(ドット単位時) バーコードの高さを MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
width	[IN]	バーコード横サイズ(倍率)	2～6(ドット単位時) バーコードのモジュール幅を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
alignment	[IN]	バーコード配置位置	CMP_ALIGNMENT_LEFT: 左揃え CMP_ALIGNMENT_CENTER: 中央揃え CMP_ALIGNMENT_RIGHT: 右揃え 上記定数以外の 3 以上の値: バーコード印刷を開始する左からの距離を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
textPosition	[IN]	可視コード印字位置	CMP_HRI_TEXT_NONE: 印刷しない CMP_HRI_TEXT_ABOVE: バーコードの上 CMP_HRI_TEXT_BELOW: バーコードの下

説明

このメソッドは、一次元バーコードを印刷するために使用します。

GS1 DataBar(CMP_BCS_GS1DATABAR、CMP_BCS_GS1DATABAR_E、CMP_BCS_GS1DATABAR_T、CMP_BCS_GS1DATABAR_L)は、CT-E301/601. CT-S251/253/255/257/401/601/651/801/851/601II/651II/

801II/851II/801III/851III/4500 系のプリンターのみ使用できます。

ページモード中のバーコード配置位置の CMP_ALIGNMENT_CENTER と CMP_ALIGNMENT_RIGHT の指定は無視されます。

註: データには、使える文字種、桁数の制約があったり、Code Set 文字の追加が必要なものがあります。
詳しくは、プリンターのコマンドリファレンスを参照ください。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.printBarcode("123456789012", withSymbology: ESCPOSConst.CMP_BCS_UPCA,  
    withHeight: 50, withWidth: 2,  
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,  
    withTextPosition: ESCPOSConst.CMP_HRI_TEXT_ABOVE)
```

2.3.15. printPDF417 メソッド

形式

```
func printPDF417(data: String?, withDigits digits: Int32, withSteps steps: Int32, withModuleWidth moduleWidth: Int32, withStepHeight stepHeight: Int32, withECLevel ECLevel: Int32, withAlignment alignment: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	印刷データ	
digits	[IN]	桁数	0: 自動 1～30
steps	[IN]	段数	0: 自動 3～90
moduleWidth	[IN]	モジュール幅	2～8(ドット単位時) バーコードのモジュール幅を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
stepHeight	[IN]	段の高さ	2～8
ECLevel	[IN]	エラー訂正レベル	CMP_PDF417_EC_LEVEL_0: レベル 0 CMP_PDF417_EC_LEVEL_1: レベル 1 CMP_PDF417_EC_LEVEL_2: レベル 2 CMP_PDF417_EC_LEVEL_3: レベル 3 CMP_PDF417_EC_LEVEL_4: レベル 4 CMP_PDF417_EC_LEVEL_5: レベル 5 CMP_PDF417_EC_LEVEL_6: レベル 6 CMP_PDF417_EC_LEVEL_7: レベル 7 CMP_PDF417_EC_LEVEL_8: レベル 8
alignment	[IN]	バーコード配置位置	CMP_ALIGNMENT_LEFT: 左揃え CMP_ALIGNMENT_CENTER: 中央揃え CMP_ALIGNMENT_RIGHT: 右揃え 上記定数以外の 3 以上の値: バーコード印刷を開始する左からの距離を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。

説明

このメソッドは、PDF417 バーコードを印刷するために使用します。

各パラメータの詳細は、各プリンターのコマンドリファレンスを参照してください。

ページモード中のバーコード配置位置の CMP_ALIGNMENT_CENTER と CMP_ALIGNMENT_RIGHT の指定は無視されます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.printPDF417(
    "http://www.citizen-systems.co.jp/prINTER/tps/index.html",
    withDigits: 0, withSteps: 0, withModuleWidth: 3, withStepHeight: 3,
    withECLevel: ESCPOSConst.CMP_PDF417_EC_LEVEL_0,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT)
```


2.3.16. printQRCode メソッド

形式

```
func printQRCode(data: String?, withModuleSize moduleSize: Int32, withECLevel ECLevel: Int32,
    withAlignment alignment: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	印刷データ	
moduleSize	[IN]	モジュール幅	1～16(ドット単位時) バーコードのモジュールサイズを MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
ECLevel	[IN]	エラー修正レベル	CMP_QRCODE_EC_LEVEL_L: レベル L(7%) CMP_QRCODE_EC_LEVEL_M: レベル M(15%) CMP_QRCODE_EC_LEVEL_Q: レベル Q(25%) CMP_QRCODE_EC_LEVEL_H: レベル H(30%)
alignment	[IN]	バーコード配置位置	CMP_ALIGNMENT_LEFT: 左揃え CMP_ALIGNMENT_CENTER: 中央揃え CMP_ALIGNMENT_RIGHT: 右揃え 上記定数以外の 3 以上の値: バーコード印刷を開始する左からの距離を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。

説明

このメソッドは、QR コードを印刷するために使用します。

各パラメータの詳細は、各プリンターのコマンドリファレンスを参照してください。

ページモード中のバーコード配置位置の CMP_ALIGNMENT_CENTER と CMP_ALIGNMENT_RIGHT の指定は無視されます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.printQRCode(
    "http://www.citizen-systems.co.jp/prINTER/tps/index.html",
    withModuleSize: 4,
    withECLevel: ESCPOSConst.CMP_QRCODE_EC_LEVEL_L,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT)
```

2.3.17. printGS1DataBarStacked メソッド

形式

```
func printGS1DataBarStacked(data: String?, withSymbology symbology: Int32, withModuleSize
    moduleSize: Int32, withMaxWidth maxWidth: Int32, withAlignment alignment: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	印刷データ	
symbology	[IN]	バーコードタイプ	CMP_BCS_GS1DATABAR_S: GS1 DataBar Stacked CMP_BCS_GS1DATABAR_E_S: GS1 DataBar Expanded Stacked CMP_BCS_GS1DATABAR_S_O: GS1 DataBar Stacked Omnidirectional
moduleSize	[IN]	モジュール幅	2～8(ドット単位時) GS1 DataBar Expanded Stacked の最大幅を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
maxWidth	[IN]	最大幅	106～39528 GS1 DataBar Expanded Stacked の最大幅を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
Alignment	[IN]	バーコード配置位置	CMP_ALIGNMENT_LEFT: 左揃え CMP_ALIGNMENT_CENTER: 中央揃え CMP_ALIGNMENT_RIGHT: 右揃え 上記定数以外の 3 以上の値: バーコード印刷を開始する左からの距離を MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。

説明

このメソッドは、2次元の GS1 DataBar を印刷するために使用します。

このメソッドは、CT-E301/601, CT-S251/253/255/257/401/601/651/801/851/601II/651II/801II/851II/801III/851III/4500 系のプリンターのみ使用できます。

各パラメータの詳細は、各プリンターのコマンドリファレンスを参照してください。

ページモード中のバーコード配置位置の CMP_ALIGNMENT_CENTER と CMP_ALIGNMENT_RIGHT の指定は無視されます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.printGS1DataBarStacked(
    "0123456789012",
    withSymbology: ESCPOSConst.CMP_BCS_GS1DATABAR_S,
    withModuleSize: 4,
    withMaxWidth: 400,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT)
```

2.3.18. cutPaper メソッド

形式

```
func cutPaper(percentage: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
type	[IN]	カット種別	CMP_CUT_FULL: フルカット CMP_CUT_PARTIAL: パーシャルカット CMP_CUT_FULL_PREFEED: カット位置紙送り後、フルカット CMP_CUT_PARTIAL_PREFEED: カット位置紙送り後、パーシャルカット

説明

このメソッドは、用紙をカットするために使用します。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.cutPaper(ESCPOSConst.CMP_CUT_PARTIAL_PREFEED)
```

2.3.19. unitFeed メソッド

形式

```
func unitFeed(IfConunt: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
ufCount	[IN]	紙送り数	MapMode プロパティ で定義された単位 (デフォルトはドット単位)で指定します。

説明

このメソッドは、ドット単位で紙送りするために使用します。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.unitFeed(200)
```

2.3.20. markFeed メソッド

形式

```
func markFeed(type: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
type	[IN]	ラベル用紙／ブラックマーク用紙のハンドリングの種類	CMP_MF_TO_CUTTER: 自動カッターのカット位置の上まで フィードし、さらにカット CMP_MF_TO_NEXT_TOF: 次の先頭印字位置までフィード

説明

このメソッドは、ラベル用紙／ブラックマーク用紙を利用するために使用します。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.markFeed(ESCPOSConst.CMP_MF_TO_CUTTER)
```

2.3.21. openDrawer メソッド

形式

```
func openDrawer(drawer: Int32, withPulseLength pulsLen: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
drawer	[IN]	キャッシュドローワー番号	CMP_DRAWER_1: ドローワー1 CMP_DRAWER_2: ドローワー2
pulseLen	[IN]	シグナルの長さ	1～8 オン時間/オフ時間をそれぞれ値 × 100ms に指定

説明

このメソッドは、プリンターに接続されたキャッシュドローワーをオープンするために使用します。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.openDrawer(ESCPOSConst.CMP_DRAWER_1, withPulseLength: 1)
```

2.3.22. transactionPrint メソッド

形式

```
func transactionPrint(control: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
control	[IN]	一括処理を開始／終了を指定	CMP_TP_TRANSACTION: 一括処理の開始 CMP_TP_NORMAL: バッファデータを印刷して一括処理を終了

説明

このメソッドは、一括処理モードを開始または終了するために使用します。

control が CMP_TP_TRANSACTION ならば、一括処理モードに入ります。これ以降のメソッド呼び出しは、印刷データをバッファリングします。一括処理モードに該当するメソッドは以下の通りです。

```
printText, printBitmap, printNVBitmap, printBarCode, printPDF417, printQRCode, cutPaper, unitFeed,
markFeed, openDrawer, rotatePrint, pageModePrint, clearePrintArea, printData, printNormal
```

control が CMP_TP_NORMAL ならば、一括処理モードを抜けます。データがバッファリングされていたならば、そのデータは印刷されます。一括処理全体は、一つのメッセージとして処理されます。

clearOutput メソッドを呼び出すことによって、一括処理モードは取り消されます。バッファされた印刷行も削除されます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.transactionPrint(ESCPOSConst.CMP_TP_TRANSACTION)
escp!.printNVBitmap(1)
escp!.printBarCode("123456789012", withSymbology: ESCPOSConst.CMP_BCS_UPCA,
    withHeight: 50, withWidth: 2,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withTextPosition: ESCPOSConst.CMP_HRI_TEXT_ABOVE)
escp!.printText("Line 1\n", withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH)
escp!.printText("Line 2\n", withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH)
escp!.printText("Line 3\n", withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH )
escp!.printBarCode("123456789012", withSymbology: ESCPOSConst.CMP_BCS_UPCA,
    withHeight: 50, withWidth: 2,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withTextPosition: ESCPOSConst.CMP_HRI_TEXT_ABOVE)
escp!.printNVBitmap(1)
escp!.cutPaper(ESCPOSConst.CMP_CUT_PARTIAL_PREFEED)
escp!.transactionPrint(ESCPOSConst.CMP_TP_NORMAL)
```

2.3.23. rotatePrint メソッド

形式

```
func rotatePrint(rotation: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
rotation	[IN]	回転方向を指定	CMP_RP_ROTATE180: 180 度回転印刷、つまり倒立印字の開始 CMP_RP_BARCODE: 回転バーコード印刷の開始、この値は上記の回転印刷開始の値との論理和 CMP_RP_BITMAP: 回転ビットマップ印刷の開始、この値は上記の回転印刷開始の値との論理和 CMP_RP_NORMAL: 回転印刷の終了

説明

このメソッドは、回転印刷モードを開始または終了するために使用します。

rotation に CMP_RP_ROTATE180 が含まれている場合は、倒立印刷モードに入ります。回転印刷モードに該当するメソッドは以下の通りです。

printText、printNormal

更に CMP_RP_BARCODE あるいは CMP_RP_BITMAP が含まれている場合は、次のメソッドも回転して印刷されます。

printBarcode, printPDF417, printQRCode あるいは printBitmap

rotation が CMP_RP_NORMAL ならば、回転印刷モードを抜けます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.rotatePrint(ESCPOSConst.CMP_RP_ROTATE180|ESCPOSConst.CMP_RP_BARCODE|
    ESCPOSConst.CMP_RP_BITMAP)
escp!.printBitmap("samplebitmap.bmp", withWidth: ESCPOSConst.CMP_BM_ASIS,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_CENTER)
escp!.printBarCode("123456789012", withSymbology: ESCPOSConst.CMP_BCS_UPCA,
    withHeight: 50, withWidth: 2,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withTextPosition: ESCPOSConst.CMP_HRI_TEXT_ABOVE)
escp!.printText("Line 1\n", withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH)
escp!.printText("Line 2\n", withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH)
escp!.printText("Line 3\n", withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize: ESCPOSConst.CMP_TXT_1WIDTH)
escp!.cutPaper(ESCPOSConst.CMP_CUT_PARTIAL_PREFEED)
escp!.rotatePrint(ESCPOSConst.CMP_RP_NORMAL)
```


形式

パラメータ

値	[IN/OUT]	意味	設定可能範囲
control	[IN]	ページモードコントロール	CMP_PM_PAGE_MODE: ページモードの開始 CMP_PM_PRINT_SAVE: ページモード印刷領域の印刷データを印刷し、その印刷データを保存 CMP_PM_NORMAL: ページモード印刷領域の印刷データを印刷し、その印刷データを消去し、ページモードを終了 CMP_PM_CANCEL: ページモード印刷領域の印刷データを消去し、何も印刷せずにページモードを終了

説明

戻り値

使用例

```

9012345 67890123456789012345678901234567890\n",
withAlignment: ESCPOSConst.CMP_ALIGNMENT_RIGHT,
withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
withTextSize:
    ESCPOSConst.CMP_TXT_1WIDTH|ESCPOSConst.CMP_TXT_1HEIGHT)
// Start of Page Mode
escp!.pageModePrint(ESCPOSConst.CMP_PM_PAGE_MODE)
// Set offset of Page Mode
escp!.setPageModeVerticalPosition(0)
escp!.setPageModeHorizontalPosition(0)
// Set direction of Page Mode
escp!.setPageModePrintDirection(ESCPOSConst.CMP_PD_TOP_TO_BOTTOM)
// Set print area of Page Mode
escp!.setPageModePrintArea("308,0,76,800")
var receipt = String(format: "%c|4C- Receipt -\n", 27)
escp!.printNormal(receipt)
// Set print area of Page Mode
escp!.setPageModePrintArea("184,0,120,800")
escp!.printText("$ 299.99- \n", withAlignment:
    ESCPOSConst.CMP_ALIGNMENT_CENTER,
    withAttribute:
        ESCPOSConst.CMP_FNT_UNDERLINE|ESCPOSConst.CMP_FNT_BOLD,
    withTextSize:
        ESCPOSConst.CMP_TXT_4WIDTH|ESCPOSConst.CMP_TXT_4HEIGHT)
// Set print area of Page Mode
escp!.setPageModePrintArea("88,0,88,560")
escp!.printText("CITIZEN SYSTEMS\n",
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_RIGHT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize:
        ESCPOSConst.CMP_TXT_2WIDTH|ESCPOSConst.CMP_TXT_3HEIGHT)
// Set print area of Page Mode
escp!.setPageModePrintArea("0,0,98,480")
escp!.printBarcode("123456789012", withSymbology: ESCPOSConst.CMP_BCS_UPCA,
    withHeight: 64, withWidth: 4,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withTextPosition: ESCPOSConst.CMP_HRI_TEXT_BELOW)
// Set print area of Page Mode
escp!.setPageModePrintArea("0,600,192,192")
escp!.printQRCode("http://www.citizen-systems.co.jp/", withModuleSize: 5,
    withECLevel: ESCPOSConst.CMP_QRCODE_EC_LEVEL_L,
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT)
// End of Page Mode
escp!.pageModePrint(ESCPOSConst.CMP_PM_NORMAL)

```

印刷イメージ



2.3.25. clearPrintArea メソッド

形式

```
func clearPrintArea() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、PageModePrintArea プロパティで定義したページモード印刷領域上の印刷データを消去するために使用します。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.clearPrintArea()
```

2.3.26. clearOutput メソッド

形式

```
func clearOutput() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、transactionPrint や pageModePrint メソッドでバッファリングされている全ての送信データをクリアするために使用します。同時にプリンター上の印刷中データをクリアするコマンドを送信します。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.clearOutput()
```

2.3.27. printData メソッド

形式

```
func printData(data: UnsafeMutablePointer<Int8>, withLength length: UInt) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
data	[IN]	送信データ	
length	[IN]	送信データサイズ	

説明

このメソッドは、バイトデータをそのままプリンターに送信するために使用します。
通常は必要ありませんが、プリンターの ESC コマンドなどを直接指定したい場合に使用します。
ご使用の際は、他のメソッドに影響を与えない様に注意する必要があります。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
// ブザーを鳴らす(ブザー対応プリンターが必要です)
var data: [Int8] = [0x1b, 0x1e]
res = escp!.printData(&data, withLength: 2)
```

2.3.28. printNormal メソッド

形式

```
func printNormal(data: String?) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
Data	[IN]	印字データ (OPOS エスケープシーケンス対応)	

説明

このメソッドは、OPOS で定義されているエスケープシーケンスを使用して印字する場合に使用します。
OPOS に精通している場合に、ご使用ください。
本 SDK で対応するエスケープシーケンスは、下記の通りです。
詳細は OPOS の仕様書を参照ください。

エスケープシーケンス		注意事項
用紙カット	ESC #P	パーシャルカット(1-99)、フルカット(0,100)
フィードと用紙カット	ESC #FP	パーシャルカット(1-99)、フルカット(0,100)
ビットマップ印刷	ESC #B	1-20(プリンターに登録された Bitmap 番号を指定) Bitmap 印字後の印字位置は初期状態(左寄せ)にもどります
複数行フィード	ESC #F	
単位フィード	ESC #uF	
バーコード印刷	ESC #R	
フォントタイプ指定	ESC #FT	
ボールド	ESC bC	
アンダーライン	ESC #uC	
カスタムカラー	ESC #rC	専用の 2 色紙を使用時のみ有効
赤色	ESC rC	専用の 2 色紙を使用時のみ有効
反転文字	ESC rvC	
縦横 1 倍角	ESC 1C	
横倍角	ESC 2C	
縦倍角	ESC 3C	
縦横倍角	ESC 4C	
横倍率	ESC #hC	1-8
縦倍率	ESC #vC	1-8
中央揃え	ESC cA	
右寄せ	ESC rA	
ノーマル	ESC N	

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
var receipt = String(format: "%c|2vC Rceipt -\n", 27)
escp!.printNormal(receipt)
```

2.3.29. watermarkPrint メソッド

形式

```
func watermarkPrint(start: Int32, withNvImageNumber nvImageNumber: Int32, withPass pass: Int32,
    withFeed feed: Int32, withRepeat repeat: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
start	[IN]	開始／終了	CMP_WM_START: 透かし印刷を開始します CMP_WM_STOP: 透かし印刷を終了します。
nvImageNumber	[IN]	プリンターのフラッシュメモリ内に格納されている画像番号	1～20
pass	[IN]	透かし印刷初回開始位置(縦方向)	0～65,535(ドット単位時) MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
feed	[IN]	透かし印刷間隔(縦方向)	0～65,535(ドット単位時) MapMode プロパティ で定義された単位(デフォルトはドット単位)で指定します。
repeat	[IN]	透かし印刷繰り返し回数	0: 無限 1～65,535: 繰り返し回数

説明

このメソッドは、透かし印刷を行うために使用します。

CT-E601, CT-S251/255/257/601II/651II/801II/851II 系のプリンターで利用可能です。

プリンターのフラッシュメモリに保存されているビットマップ画像を重ねて印刷します。

このメソッドを使用するためには、事前に画像の登録が必要です。画像登録は、[setNVBitmap メソッド](#)で登録するか、プリンター用ユーティリティソフトウェアの「POS プリンターユーティリティ」を使用してください。

CMP_WM_STOP で透かし印刷終了を指定した時は、他の全ての引数は無視されます。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

```
escp!.watermarkPrint(ESCPOSConst.CMP_WM_START, withNvImageNumber: 1,
    withPass: 0, withFeed: 0, withRepeat: 0)
```

2.3.30. searchCitizenPrinter メソッド

形式

```
class func searchCitizenPrinter(ifType: Int32, withSearchTime searchTime: Int32, result:
    UnsafeMutablePointer<Int32>) -> [AnyObject]?
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
ifType	[IN]	接続タイプ	CMP_PORT_WiFi: Wi-Fi 接続 CMP_PORT_BLUETOOTH: Bluetooth 接続 CMP_PORT_USB: USB 接続
searchTime	[IN]	検索時間 (秒)	1～30
result	[OUT]	エラーコード	---

result には成功時は CMP_SUCCESS(0)を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「2.3.1 戻り値」を参照してください。

エラーコード	説明
CMP_E_ILLEGAL (1101)	無効なパラメータ値です。 ①未対応の接続タイプを指定した ②範囲外の検索時間を指定した
CMP_E_FAILURE (1104)	検索処理の途中でエラーが発生し、検索に失敗しました
CMP_E_NO_LIST (1106)	検索の結果、プリンターが発見出来ませんでした

説明

このメソッドは、プリンターを検索し、プリンターの情報リストを取得するために使用します。接続タイプと検索時間を指定して実行してください。

シミュレーター利用時は Wi-Fi 接続のみ使用可能です。

検索時間経過後に、結果を result パラメータにセットし、検索出来たプリンターの情報を NSArray 型で返します。

接続タイプが CMP_PORT_WiFi の場合、CT-E301/601, CT-S251/253/255/257/401/601/651/801/851/601II/651II/801II/851II/801III/851III/2000/4000/4500 系のプリンターのみ検索可能です。検索時間の推奨値は 3 秒以上です。これより短い時間の場合、ネットワークの状態によっては検索漏れが発生する場合があります。

接続タイプが CMP_PORT_BLUETOOTH の場合、CT-E601, CT-S251/255/257/281/601II/651II/801II/851II/801III/851III/4500 が検索可能です。検索時間である searchTime に関係無く、検索は直ぐに完了します。

接続タイプが CMP_PORT_USB の場合、CT-E601, CT-S251/255/257/4500 が検索可能です。検索時間である searchTime に関係無く、検索は直ぐに完了します。

戻り値

成功した時は検索されたプリンターの情報リストが返されます。失敗した時は空のリストが返されます。プリンターの情報リストは CitizenPrinterInfo 型で格納されており、接続タイプによって取得出来る情報が異なります。

接続タイプ	CitizenPrinterInfo	取得出来る情報
CMP_PORT_WiFi	ipAddress	IP アドレス
	macAddress	MAC アドレス
	deviceName	(空)
	bdAddress	(空)

	usbSerialNo	(空)
CMP_PORT_BLUETOOTH	ipAddress	(空)
	macAddress	(空)
	deviceName	(空)／Bluetooth デバイス名 ※プリンタ機種に依存
	bdAddress	(空)／BD アドレス ※iOS6 以上
	usbSerialNo	(空)
CMP_PORT_USB	ipAddress	(空)
	macAddress	(空)
	deviceName	モデル名
	bdAddress	(空)
	usbSerialNo	USB シリアル番号

使用例

```

var result = Int32(0)
let array = ESCPOSPrinter.searchCitizenPrinter(ESCPOSConst.CMP_PORT_WiFi,
        withSearchTime: 4, result: &result)!
for var i = 0; i < array.count; i++ {
    let prninfo = array[i] as? CitizenPrinterInfo
    println("IP Address : \(prninfo.ipAddress)")
    println("MAC Address : \(prninfo.macAddress)")
}

```

2.3.31. searchESCPOSPrinter メソッド

形式

```
class func searchESCPOSPrinter(ifType: Int32, withSearchTime searchTime: Int32, result:
    UnsafeMutablePointer<Int32>) -> [AnyObject]?
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
ifType	[IN]	接続タイプ	CMP_PORT_WiFi: Wi-Fi 接続 CMP_PORT_BLUETOOTH: Bluetooth 接続 CMP_PORT_USB: USB 接続
searchTime	[IN]	検索時間 (秒)	1~30
result	[OUT]	エラーコード	---

result には成功時は CMP_SUCCESS(0)を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

エラーコード	説明
CMP_E_ILLEGAL (1101)	無効なパラメータ値です。 ①未対応の接続タイプを指定した ②範囲外の検索時間を指定した
CMP_E_FAILURE (1104)	検索処理の途中でエラーが発生し、検索に失敗しました
CMP_E_NO_LIST (1106)	検索の結果、プリンターが発見出来ませんでした

説明

このメソッドは、プリンターを検索し、アドレスのリストを取得するために使用します。接続タイプと検索時間を指定して実行してください。

シミュレーター利用時は Wi-Fi 接続のみ使用可能です。

検索時間経過後に、結果を result パラメータにセットし、検索出来たプリンターの情報を NSArray 型で返します。

接続タイプが CMP_PORT_WiFi の場合、CT-E301/601, CT-S251/253/255/257/401/601/651/801/851/601II/651II/801II/851II/801III/851III/2000/4000/4500 系のプリンターのみ検索可能です。検索時間の推奨値は 3 秒以上です。これより短い時間の場合、ネットワークの状態によっては検索漏れが発生する場合があります。

接続タイプが CMP_PORT_BLUETOOTH の場合、CT-E601, CT-S251/255/257/281/601II/651II/801II/851II/801III/851III/4500 が検索可能です。検索時間である searchTime に関係無く、検索は直ぐに完了します。

接続タイプが CMP_PORT_USB の場合、CT-E601, CT-S251/255/257/4500 が検索可能です。検索時間である searchTime に関係無く、検索は直ぐに完了します。

戻り値

接続タイプが CMP_PORT_WiFi の場合、成功した時はプリンターの IP アドレスのリストが返されます。失敗した時は空のリストが返されます。

接続タイプが CMP_PORT_BLUETOOTH で iOS6 以上の場合、成功した時はプリンターの Bluetooth デバイスアドレスのリストが返されます。失敗した時は空のリストが返されます。iOS6 未満の場合、Bluetooth デバイスアドレスの情報に対応していませんので、空のリストが返されます。

接続タイプが CMP_PORT_USB の場合、成功した時はプリンターのシリアル番号のリストが返されます。失敗した時は空のリストが返されます。

使用例

```
var result = Int32(0)
let array = ESCPOSPrinter.searchESCPOSPrinter(ESCPOSConst.CMP_PORT_WiFi,
    withSearchTime: 4, result: &result)!
```

2.3.32. printerCheckEx メソッド

形式

- 1) func printerCheckEx(status: UnsafeMutablePointer<Int32>, withConnectType connectType: Int32, withAddress addr: String?) -> Int32
- 2) func printerCheckEx(status: UnsafeMutablePointer<Int32>, withConnectType connectType: Int32, withAddress addr: String?, withPort port: Int32) -> Int32
- 3) func printerCheckEx(status: UnsafeMutablePointer<Int32>, withConnectType connectType: Int32, withAddress addr: String?, withPort port: Int32, withTimeout timeout: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
status	[OUT]	ステータスコード	
connectType	[IN]	接続タイプ	CMP_PORT_WiFi: Wi-Fi 接続 CMP_PORT_BLUETOOTH: Bluetooth 接続 CMP_PORT_USB: USB 接続 (Lightning/Type-C) CMP_PORT_SNMP: SNMP 接続 (Network 接続のみ)
addr	[IN]	接続先の IP アドレス または Bluetooth デバイスアドレス または Bluetooth デバイス名 またはシリアル番号	WiFi/SNMP : 0.0.0.0 ~ 255.255.255.255 Bluetooth : 00:00:00:00:00:00 ~ FF:FF:FF:FF:FF:FF デバイス名 USB : 空白またはシリアル番号
port	[IN]	接続先ポート番号	
timeout	[IN]	タイムアウト (msec)	

説明

本メソッドは、プリンターへ接続し、プリンターのステータスを取得するために使用します。処理が完了した後は接続を切断します（接続タイプの CMP_PORT_SNMP は除く）。

接続タイプの CMP_PORT_SNMP は、Network 接続のプリンターのみ使用できます。本接続タイプを利用する事により、他の接続に関係なくステータスを取得する事ができます。本接続タイプを使用するには、本機能に対応したプリンターが必要です。

ステータスの取得が成功の場合は、下記のステータスコードを論理和で status パラメータにセットします。

ステータスコード	説明
CMP_STS_NORMAL (0)	プリンターは正常です。
CMP_STS_PRINTEROFF (128)	プリンターはオフラインです。
CMP_STS_PAPER_EMPTY (32)	用紙がありません。
CMP_STS_COVER_OPEN (16)	プリンタカバーが開いています。
CMP_STS_PAPER_NEAREMPTY (4)	ニアエンptyです。
CMP_STS_DRAWER_LEVEL_H (2)	ドロワーキックコネクタ3番ピンの状態が H です。

※ CMP-20/30 用の CMP_STS_BATTERY_LOW と CMP_STS_MSR_READ の取得は対応していません

戻り値

成功時は CMP_SUCCESS(0) を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「[2.4.1 戻り値](#)」を参照してください。

エラーコード	説明
CMP_E_NOTCONNECT (1003)	プリンターへ接続できませんでした。 ①プリンターが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない ④プリンターは別の接続で使用中 (SNMP を除く)
CMP_E_CONNECT_NOTFOUND (1004)	プリンター接続後の対応機種確認に失敗しました。 ①対応機種でない

使用例

```

var status = Int32(0)
var result = escp!.printerCheckEx(&status,
                                   withConnectType: ESCPOSConst.CMP_PORT_WiFi,
                                   withAddress: "192.168.182.100")
if ESCPOSConst.CMP_SUCCESS == result {
    if ESCPOSConst.CMP_STS_NORMAL == status {
        // No Error
    } else {
        if (ESCPOSConst.CMP_STS_COVER_OPEN & status) > 0 {
            // Cover Open
        }
        if (ESCPOSConst.CMP_STS_PAPER_EMPTY & status) > 0 {
            // Paper Empty
        }
        if (ESCPOSConst.CMP_STS_PRINTEROFF & status) > 0 {
            // Printer Offline
        }
        if (ESCPOSConst.CMP_STS_PAPER_NEAREMPTY & status) > 0 {
            // Paper Near Empty
        }
        if (ESCPOSConst.CMP_STS_DRAWER_LEVEL_H & status) > 0 {
            // Pin 3 of drawer kick-out connector = H
        }
    }
} else {
    // printerCheckEx Error
}

```

2.3.33. openDrawerEx メソッド

形式

- 1) func openDrawerEx(drawer: Int32, withPulseLength pulsLen: Int32, connectType: Int32, withAddress addr: String?) -> Int32
- 2) func openDrawerEx(drawer: Int32, withPulseLength pulsLen: Int32, connectType: Int32, withAddress addr: String?, withPort port: Int32) -> Int32
- 3) func openDrawerEx(drawer: Int32, withPulseLength pulsLen: Int32, connectType: Int32, withAddress addr: String?, withPort port: Int32, withTimeout timeout: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
drawer	[IN]	キャッシュドロー番号	CMP_DRAWER_1: ドロー1 CMP_DRAWER_2: ドロー2
pulseLen	[IN]	シグナルの長さ	1~8 (× 100) msec
connectType	[IN]	接続タイプ	CMP_PORT_WiFi: Wi-Fi 接続 CMP_PORT_BLUETOOTH: Bluetooth 接続 CMP_PORT_USB: USB 接続 (Lightning/Type-C)
addr	[IN]	接続先の IP アドレス または Bluetooth デバイスアドレス または Bluetooth デバイス名 またはシリアル番号	WiFi : 0.0.0.0~ 255.255.255.255 Bluetooth : 00:00:00:00:00:00~FF:FF:FF:FF:FF:FF デバイス名 USB : 空白またはシリアル番号
port	[IN]	接続先ポート番号	
timeout	[IN]	タイムアウト (msec)	

説明

本メソッドは、プリンターへ接続し、プリンターに接続されたキャッシュドローをオープンするために使用します。処理が完了した後は接続を切断します。

このメソッドは、プリンターが異常な場合(カバーオープンや用紙無し)でも処理を実行できます。

戻り値

成功時は CMP_SUCCESS(0) を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「[2.4.1 戻り値](#)」を参照してください。

エラーコード	説明
CMP_E_NOTCONNECT (1003)	プリンターへ接続できませんでした。 ①プリンターが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない ④プリンターは別の接続で使用
CMP_E_CONNECT_NOTFOUND (1004)	プリンター接続後の対応機種確認に失敗しました。 ① 対応機種でない

使用例

```
escp!.openDrawerEx(ESCPOSConst.CMP_DRAWER_1,
    withPulseLength: 1,
```

```
withConnectType: ESCPOSConst.CMP_PORT_WiFi,  
withAddress: "192.168.182.100")
```

2.3.34. setPrintCompletedTimeout メソッド

形式

```
func setPrintCompletedTimeout(timeout: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
timeout	[IN]	印刷完了通知確認タイムアウト (msec)	0: タイムアウトを自動調整します。 上記定数以外の 1 以上の値: タイムアウトを msec 単位で指定します。

説明

このメソッドは、印刷完了通知を確認するタイムアウトを設定するために使用します。

タイムアウトは、インスタンス生成時に 0 に初期化します。

0 を指定した場合は、印刷データに合わせてタイムアウトを自動的に調整されます。

1 以上の値を指定した場合は、指定されたタイムアウトに固定されます。

印刷完了確認処理の詳細は、「[2.4.1 印刷完了確認機能について](#)」を参照してください。

戻り値

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

使用例

// 自動調整

```
escp!.setPrintCompletedTimeout(0)
```

// 90sec 固定

```
escp!.setPrintCompletedTimeout(90000)
```


2.3.35. setLog メソッド

形式

```
func setLog(mode: Int32, withPath path: String?, withMaxSize maxSize: Int32)
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
mode	[IN]	ログモード	0 : 記録なし 1 : アクセス履歴の記録 2 : エラーのみ記録
path	[IN]	格納フォルダ	アプリケーション共有フォルダ内のフォルダ
maxSize	[IN]	ログサイズ	0: サイズ制限なし 1~: 最大サイズ(MB)

説明

このメソッドは、ログ機能を設定するために使用します。ログ機能の詳細は「[2.4.4 ログ機能について](#)」を参照してください。

戻り値

ありません。

使用例

```
escp!.setLog(1, withPath: "/", withMaxSize: 10)
```

2.3.36. getVersionCode メソッド

形式

```
func getVersionCode() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、SDK のバージョン番号を数値で取得するために使用します。

戻り値

SDK のバージョン番号を数値 (Ver1.00 の場合: 100) で返します。

使用例

```
escp!.getVersionCode()
```

2.3.37. getVersionName メソッド

形式

```
func getVersionName() -> String?
```

パラメータ

ありません。

説明

このメソッドは、SDK のバージョン番号を文字列で取得するために使用します。

戻り値

SDK のバージョン番号を文字列 (Ver1.00 の場合: “1.00”) で返します。

使用例

```
escp!.getVersionName()
```

2.3.38. PageModeArea プロパティ

形式

String

属性

Read only

説明

このプロパティは、ページモード印刷領域で設定可能な最大値である、ページ領域を [MapMode プロパティ](#) で定義された単位（デフォルトはドット単位）で保持します。

このプロパティの構成は、カンマ区切りの 2 つの ASCII 数字で構成され、幅と高さの順に列挙します。

ページ領域は、プリンターのハードウェア的な能力で決定されます。

[CT-S251 系]: "432,1662"

[CT-S281 系]: "384,938"

[CT-E301/601, CT-S253/255/257/401/601/651/801/851/601II/651II/801II/851II/2000 系]:
"576,1662"

[CT-S4000/4500 系]: "832,1662"

[CMP-20/20II 系]: "384,938"

[CMP-30/30II 系]: "576,938"

[CMP-40 系]: "832,938"

例えば、文字列が "384,938" であれば、ページモード領域は幅 384 単位、高さ 938 単位です。このページモード領域は、左上隅 (0,0) と右下隅 (383,937) で囲まれる長方形を示します。

このプロパティにアクセスする前に [connect メソッド](#) を完了してください。このプロパティは、[connect メソッド](#) で設定されます。

設定方法

なし。

取得方法

```
func getPageModeArea() -> String?
```

戻り値として、ページ領域を返します。

2.3.39. PageModePrintArea プロパティ

形式

String

属性

Read/Write

説明

このプロパティは、ページモード印刷領域を [MapMode プロパティ](#) で定義された単位（デフォルトはドット単位）で保持します。ページモード印刷領域のサイズは、ページ領域より大きく取ることはできません。

このプロパティは、カンマ区切りの 4 つの ASCII 数字のみで構成され、空白文字を含めることはできません。構成は、始点の水平方向座標、始点の垂直方向座標、水平方向の幅、垂直方向の高さ、の順に列挙します。

ページモード印刷領域の右端を超える文字は、次の行に印刷されます。ページモード印刷領域の下端を超える文字と画像は、印刷されません。

例えば文字列が、“50,100,200,400” である場合、ページモード印刷領域の座標は、左上隅（50,100）と右下隅（249,499）で囲まれる長方形を示します。

このプロパティにアクセスする前に connect メソッドを完了してください。このプロパティは、connect メソッドで “0,0,0,0” に初期化されます。

設定方法

```
func setPageModePrintArea(area: String?) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

成功時は CMP_SUCCESS(0) を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

取得方法

```
func getPageModePrintArea() -> String?
```

戻り値として、設定されているページモード印刷領域を返します。

2.3.40. PageModePrintDirection プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、現在編集集中のページモード印刷領域内の印刷方向を保持します。値は次の通りです。

値	意味
CMP_PD_LEFT_TO_RIGHT	ページモード印刷領域の左上隅を始点に、左から右方向へ印刷します。通常印刷方向です。
CMP_PD_BOTTOM_TO_TOP	ページモード印刷領域の左下隅を始点に、下から上方向へ印刷します。左 90 度回転印刷です。
CMP_PD_RIGHT_TO_LEFT	ページモード印刷領域の右下隅を始点に、右から左方向へ印刷します。180 度回転印刷です。
CMP_PD_TOP_TO_BOTTOM	ページモード印刷領域の右上隅を始点に、上から下方向へ印刷します。右 90 度回転印刷です。

このプロパティを変更すると、PageModeHorizontalPosition プロパティと PageModeVerticalPosition プロパティで示される印刷開始点の補正方向も変化します。

更にページモード印刷領域を切り替えることで、文字の回転方向を組み合わせたレシートや単票を印刷することができます。

このプロパティにアクセスする前に、connect メソッドを完了してください。このプロパティは、connect メソッドで CMP_PD_LEFT_TO_RIGHT に初期化されます。

設定方法

```
func setPageModePrintDirection(direction: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

取得方法

```
func getPageModePrintDirection() -> Int32
```

戻り値として、設定されているページモード印刷領域の印刷方向を返します。

2.3.41. PageModeHorizontalPosition プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、ページモード印刷領域内の印刷開始位置を水平方向に補正するためのオフセット値を [MapMode プロパティ](#) で定義された単位 (デフォルトはドット単位) で保持します。

水平方向とは、PageModePrintDirection プロパティで設定した印刷方向と同じ方向を指します。

このプロパティは、現在位置ではなくて、最後に指定した水平方向のオフセット値の設定となります。

このプロパティにアクセスする前に、connect メソッドを完了してください。このプロパティは、connect メソッドでゼロ(0)に初期化されます。

設定方法

```
func setPageModeHorizontalPosition(position: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

取得方法

```
func getPageModeHorizontalPosition() -> Int32
```

戻り値として、設定されているページモード印刷領域の水平方向オフセットを返します。

2.3.42. PageModeVerticalPosition プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、ページモード印刷領域内の印刷開始位置を垂直方向に補正するためのオフセット値を [MapMode プロパティ](#) で定義された単位 (デフォルトはドット単位) で保持します。

垂直方向とは、PageModePrintDirection プロパティで設定した印刷方向に対して垂直な方向です。

このプロパティは、現在位置ではなくて、最後に指定した垂直方向のオフセット値の設定となります。

このプロパティにアクセスする前に、connect メソッドを完了してください。このプロパティは、connect メソッドでゼロ(0)に初期化されます。

設定方法

```
func setPageModeVerticalPosition(position: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

取得方法

```
func getPageModeVerticalPosition() -> Int32
```

戻り値として、設定されているページモード印刷領域の垂直方向オフセットを返します。

2.3.43. RecLineSpacing プロパティ

形式

Int32

属性

Read/Write

説明

このプロパティは、通常文字の印刷行の高さを [MapMode プロパティ](#) で定義された単位 (デフォルトはドット単位) で保持します。すなわち、印字行の高さと行間スペースの高さの両方を加えたものです。

現在の行間の値によっては、縦倍角文字はこの値を超える場合があります。この場合の行間スペースは無しになります。

このプロパティにアクセスする前に、connect メソッドを完了してください。このプロパティは、connect メソッドで 34 に初期化されます。

設定方法

```
func setRecLineSpacing(spacing: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

成功時は CMP_SUCCESS(0) を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

取得方法

```
func getRecLineSpacing() -> Int32
```

戻り値として、設定されている通常文字の印刷行の高さを返します。

2.3.44. MapMode プロパティ

形式

Int MapMode

属性

Read/Write

説明

このプロパティは、プリンターのマッピングモードを保持します。マッピングモードはメソッドや他のプロパティで使用されるラインの高さや行間を示す様な尺度の単位を定義します。値は次の通りです。

値	意味
CMP_MM_DOTS	POS プリンターのドット幅
CMP_MM_TWIPS	1 インチの 1/1440
CMP_MM_ENGLISH	0.001 インチ
CMP_MM_METRIC	0.01 ミリメートル

MapMode プロパティの影響を受けるメソッドの引数およびプロパティは以下の通りです。

[printBitmap メソッド](#) の width, alignment
[printBitmapData メソッド](#) の width, alignment
[setNVBitmap メソッド](#) の width
[printBarcode メソッド](#) の height, width, alignment
[printPDF417 メソッド](#) の moduleWidth, alignment
[printQRCode メソッド](#) の moduleSize, alignment
[printGS1DataBarStacked メソッド](#) の moduleSize, maxSize, alignment
[unitFeed メソッド](#) の ufCount
[watermarkPrint メソッド](#) の pass, feed
[PageModeArea プロパティ](#)
[PageModePrintArea プロパティ](#)
[PageModeHorizontalPosition プロパティ](#)
[PageModeVerticalPosition プロパティ](#)
[RecLineSpacing プロパティ](#)

このプロパティにアクセスする前に、connect メソッドを完了してください。このプロパティは、connect メソッドで CMP_MM_DOTS に初期化されます。

設定方法

```
func setMapMode(number: Int32) -> Int32
```

パラメータに、設定したいプロパティ値を指定してください。

成功時は CMP_SUCCESS(0)を返します。それ以外のエラーコードは「[2.3.1 戻り値](#)」を参照してください。

取得方法

```
func getMapMode() -> Int32
```

戻り値として、設定されているマッピングモードを返します。

2.3.45. ErrorCodeExtended プロパティ

形式

Int32

属性

Read only

説明

このプロパティは、プリンターへの接続がエラーになった際の拡張エラーコードを保持します。値は以下の通りです。

拡張エラーコード	説明
CMP_EX_DEV_NO_PRINTER (62000)	プリンターへ接続できませんでした。(Bluetooth/USB/SNMP 接続) ①プリンターが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない
CMP_EX_DEV_NO_SERIALNO (62012)	プリンターシリアル番号が一致しません。(USB 接続のみ) ①指定シリアル番号の間違い
CMP_EX_DEV_OPEN_ERROR (62100)	ソケットまたはストリーム接続できませんでした。 ①通信異常が発生した ②WiFi 接続時のアドレス指定間違い
CMP_EX_DEV_NO_RESPONSE (62102)	プリンターからのコマンド応答がありません。 ①他アプリでプリンターを使用中

このプロパティは、インスタンス時と接続処理開始時に 0 に初期化されます。[connect メソッド](#)、[printerCheckEx メソッド](#)、[openDrawerEx メソッド](#)実行後に参照してください。

設定方法

なし。

取得方法

```
func getResultCodeExtended() -> Int32
```

戻り値として、拡張エラーコードを返します。

2.4. 注意事項

本 SDK の注意事項を以下に示します。

2.4.1. 印刷完了確認機能について

本 SDK は、プリンターへのデータ送信後、印刷完了通知を待ち、メソッドの成功／失敗を判断しています。印刷完了確認機能は次のタイミングで処理されます。

- (1) 一括処理(transactionPrint メソッド)の完了時
- (2) ページモード(pageModePrint メソッド)の完了時
- (3) 一括処理またはページモードのバッファリング中以外のメソッドのデータ送信時

印刷完了確認機能は、プリンターの応答を待つため時間がかかります。複数のメソッドを連続して印刷する場合は、一括処理(transactionPrint メソッド)を使用する事によりスムーズな印刷が可能です。

印刷完了通知を確認するタイムアウトは、印刷データに合わせて自動的に調整されます。

印刷データによっては、毎回タイムアウトエラーが発生する場合があります。その場合は、実際の印刷時間に合わせて [setPrintCompletedTimeout メソッド](#) でタイムアウトを設定してください。

2.4.2. UTF-8 エンコード文字列の印刷について

本 SDK は、UTF-8 でエンコードされた文字列の印刷をサポートします。

使用例

```
escp!.setEncoding(String.Encoding.utf8);
```

対応機種

機種	ファームウェアバージョン	制限事項
CT-S251	EM01-0304 以降	※1
CT-S401	DT00-1000 以降 DT10-1100 以降	
CT-S601II	EE00-0200 以降	※2
CT-S651II	EA00-0200 以降	
CT-S801II	ED00-0200 以降	
CT-S851II	DY00-0200 以降	
CT-E301/601	全バージョン	※3
CT-S253/255/257/801III/851III 4500		

注意

- ※1 プリンターは日本語、韓国語、簡体字中国語、繁体字中国語の同時印刷に未対応です。印刷可能な言語はプリンターの仕向け毎に決められた 1 つの言語のみとなります。
- ※2 プリンターは日本語、韓国語、簡体字中国語、繁体字中国語の同時印刷に未対応です。印刷可能な言語はプリンターの MSW9-4 で指定される 1 つの言語のみとなります。
- ※3 プリンターは日本語、韓国語、簡体字中国語、繁体字中国語の同時印刷に対応します。複数の言語を印刷する場合、プリンターは MSW9-4 の設定に割り当てられた言語に基づいて、使用する文字を順次検索します。その際、字体、字形、書体が一様とならない場合があります。ご注意ください。

言語と書体 (CT-S253/255/801III/851III 系)

言語	書体
日本語フォント 韓国語フォント	ゴシック体

簡体中国語フォント 繁体中国語フォント	明朝体
------------------------	-----

言語と書体 (CT-E301/601, CT-S257/4500 系)

言語	書体
日本語フォント 韓国語フォント 簡体中国語フォント 繁体中国語フォント	ゴシック体

2.4.3. JIS 第3、第4水準漢字の印刷について

本 SDK は、JIS 第3、第4水準漢字の印刷をサポートしています。

JIS 第3、第4水準漢字を印刷には、プリンター送信データのエンコードを、UTF-8 に設定する必要があります。
詳しくは、「[2.4.2 UTF-8 エンコード文字列の印刷について](#)」を参照してください。

使用例

```
// エンコード設定
escp!.setEncoding(String.Encoding.utf8);

// テキスト印刷
escp!.printText("啞焰鷗摑麴嘘俠頰軀俱繫妍鹼嚙攢繡蔣醬蟬搔瘦¥n",
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize:
        ESCPOSConst.CMP_TXT_1WIDTH|ESCPOSConst.CMP_TXT_1WIDTH)
escp!.printText("驪簞塤顛鄧禱瀆吞囊剝潑醱屏并麵萊屢蔞蠟汙¥n",
    withAlignment: ESCPOSConst.CMP_ALIGNMENT_LEFT,
    withAttribute: ESCPOSConst.CMP_FNT_DEFAULT,
    withTextSize:
        ESCPOSConst.CMP_TXT_1WIDTH|ESCPOSConst.CMP_TXT_1WIDTH);
```

2.4.4. ログ機能について

本 SDK は、メソッドの実行やプロパティの読み書きを記録するログ機能をサポートしています。ログ機能を設定する際は、[setLog メソッド](#)を使用するか、次の書式の設定ファイル「CSJPOSLib.cfg」をアプリケーションの共有フォルダに配置してください。

<CSJPOSLib.cfg の例>

[LogSetting]	・・・セクション名(固定)
LogMode=1	・・・ログモードを指定
LogPath=/	・・・ログファイルを格納するアプリケーション共有フォルダ内のフォルダを指定
LogMaxSize=10	・・・ログファイルの最大サイズを MB 単位で指定

設定項目

・ログモード

ログを記録するモードを指定します。

- 0: 記録なし
- 1: アクセス履歴の記録
- 2: エラーのみ記録

・格納フォルダ

ログファイルを格納するフォルダを指定します。本設定が指定されていない場合は、アプリケーション共有フォルダに格納されます。

・ログサイズ

ログファイルの最大容量を MB 単位で指定します。0 を指定した場合は容量制限が解除され可能な限り記録されます。

ログファイル名

ログファイルの拡張子は「.log」です。ファイル名は「CSJPOSLib」の後ろに曜日を表す数字が追加されます。曜日は日曜日を 0、月曜日を 1 として 0 から 6 のいずれかの数字が加えられます。

例) CSJPOSLib_1.log

ログファイルが既に存在し、それが当日以外の場合は、古いファイルを削除してからログを記録します。

ログフォーマット

ログ機能は、メソッド、プロパティの日付、時間、結果のアクセス情報を記録します。

--- メソッドの例1 (connect) ---

```
2022/04/06 11:54:13.548 001 METHOD call    connect(0, 192.168.234.100, 9100, 4000)
2022/04/06 11:54:13.962 001 METHOD result connect() -> Success(0)
```

--- メソッドの例2 (printText) ---

```
2022/04/06 11:54:16.672 001 METHOD call    printText([See below], 1, 0, 0)
-----Parameter Detail-----
Print text 1
Print text 2
-----
2022/04/06 11:54:16.948 001 METHOD result printText() -> Success(0)
```

--- プロパティの設定例 (RecLineSpacing) ---

```
2022/04/06 11:54:16.950 001 PROPERTY set  RecLineSpacing <- 24 : Success(0)
```

--- プロパティの参照例(RecLineSpacing) ---

```
2022/04/06 11:54:16.954 001 PROPERTY get RecLineSpacing -> 24
```

※ログ機能を使用する場合、全てのメソッドとプロパティのアクセス時にログファイルが更新されますので、SDK の処理が低下してしまうことがあります。

※次のような理由などでファイルの書き込みができない場合はログファイルの記録が行われません。このような場合エラーメッセージなどは表示されませんので、ご注意ください。

- ・書き込み禁止デバイスを指定した場合
- ・出力先に十分な領域が残っていない場合
- ・書き込み禁止のログファイルがある場合
- ・ファイルやフォルダのアクセス権がない場合
- ・他のアプリケーションがログファイルを使用している場合

2.4.5. 定数定義一覧

No	項目	定数名	型	値	説明
1	処理結果	CMP_SUCCESS	Int32	0	正常終了
		CMP_E_CONNECTED	Int32	1001	接続済み
		CMP_E_DISCONNECT	Int32	1002	未接続
		CMP_E_NOTCONNECT	Int32	1003	接続失敗
		CMP_E_CONNECT_NOTFOUND	Int32	1004	未対応機種
		CMP_E_CONNECT_OFFLINE	Int32	1005	プリンター状態異常
		CMP_E_ILLEGAL	Int32	1101	未対応処理または無効パラメータ
		CMP_E_OFFLINE	Int32	1102	オフライン
		CMP_E_NOEXIST	Int32	1103	ファイルが存在しない
		CMP_E_FAILURE	Int32	1104	処理異常
		CMP_E_TIMEOUT	Int32	1105	書き込みタイムアウト
		CMP_E_NO_LIST	Int32	1106	プリンターが見つからない
		CMP_EPTR_COVER_OPEN	Int32	1201	プリンターのカバーオープン
		CMP_EPTR_REC_EMPTY	Int32	1202	用紙切れ
		CMP_EPTR_BADFORMAT	Int32	1203	画像フォーマット異常
		CMP_EPTR_CMP_EPTR_TOOBIG	Int32	1204	画像サイズが大きすぎる
2	接続インターフェース	CMP_PORT_WiFi	Int32	0	ネットワーク接続
		CMP_PORT_BLUETOOTH	Int32	1	Bluetooth 接続
		CMP_PORT_USB	Int32	3	USB 接続 (Lightning/Type-C)
		CMP_PORT_SNMP	Int32	6	SNMP 接続
3	ステータス	CMP_STS_NORMAL	Int32	0	ステータス正常
		CMP_STS_DRAWER_LEVEL_H	Int32	2	ドロワーキックコネクタ3番ピン状態 H
		CMP_STS_PAPER_NEAREMPTY	Int32	4	ニアエンプティ
		CMP_STS_BATTERY_LOW	Int32	8	バッテリー容量低下
		CMP_STS_COVER_OPEN	Int32	16	カバーオープン
		CMP_STS_PAPER_EMPTY	Int32	32	用紙切れ
		CMP_STS_MSR_READ	Int32	64	MSR 読み取りモード状態
		CMP_STS_PRINTEROFF	Int32	128	オフライン
4	配置	CMP_ALIGNMENT_LEFT	Int32	0	左揃え
		CMP_ALIGNMENT_CENTER	Int32	1	中揃え
		CMP_ALIGNMENT_RIGHT	Int32	2	右揃え
5	文字属性	CMP_FNT_DEFAULT	Int32	0	標準フォント
		CMP_FNT_FONTB	Int32	1	フォント B
		CMP_FNT_FONTC	Int32	2	フォント C
		CMP_FNT_BOLD	Int32	8	太字
		CMP_FNT_REVERSE	Int32	16	反転
		CMP_FNT_UNDERLINE	Int32	128	下線
		CMP_FNT_ITALIC	Int32	256	イタリック
		CMP_FNT_STRIKEOUT	Int32	512	打ち消し
6	文字サイズ	CMP_TXT_1WIDTH	Int32	0	幅 1 倍
		CMP_TXT_2WIDTH	Int32	16	幅 2 倍
		CMP_TXT_3WIDTH	Int32	32	幅 3 倍
		CMP_TXT_4WIDTH	Int32	48	幅 4 倍
		CMP_TXT_5WIDTH	Int32	64	幅 5 倍
		CMP_TXT_6WIDTH	Int32	80	幅 6 倍
		CMP_TXT_7WIDTH	Int32	96	幅 7 倍

		CMP_TXT_8WIDTH	Int32	112	幅 8 倍
		CMP_TXT_1HEIGHT	Int32	0	高さ 1 倍
		CMP_TXT_2HEIGHT	Int32	1	高さ 2 倍
		CMP_TXT_3HEIGHT	Int32	2	高さ 3 倍
		CMP_TXT_4HEIGHT	Int32	3	高さ 4 倍
		CMP_TXT_5HEIGHT	Int32	4	高さ 5 倍
		CMP_TXT_6HEIGHT	Int32	5	高さ 6 倍
		CMP_TXT_7HEIGHT	Int32	6	高さ 7 倍
		CMP_TXT_8HEIGHT	Int32	7	高さ 8 倍
7	サイド	CMP_SIDE_RIGHT	Int32	0	右側
		CMP_SIDE_LEFT	Int32	1	左側
8	画像幅	CMP_BM_ASIS	Int32	-11	1ドット当たり1ピクセル
9	画像モード	CMP_BM_MODE_CMD_RASTER	Int32	1	モノクロ印刷(ラスターコマンド)
		CMP_BM_MODE_CMD_BITIMAGE	Int32	2	モノクロ印刷(ビットイメージコマンド)
		CMP_BM_MODE_CMD_MONO	Int32	8	モノクロ登録
		CMP_BM_MODE_CMD_GRAY16	Int32	8	グレースケール印刷/登録
		CMP_BM_MODE_HT_THRESHOLD	Int32	16	ハーフトーン(しきい値)
		CMP_BM_MODE_HT_DITHER	Int32	32	ハーフトーン(ディザー)
		CMP_BM_MODE_CMD_GRAY16DOWNLOAD	Int32	256	グレースケール印刷(ダウンロードグラフィックスコマンド)
10	バーコード種類	CMP_BCS_UPCA	Int32	101	UPC-A
		CMP_BCS_UPCE	Int32	102	UPC-E
		CMP_BCS_EAN8	Int32	103	EAN8
		CMP_BCS_EAN13	Int32	104	EAN13
		CMP_BCS_JAN8	Int32	105	JAN8
		CMP_BCS_JAN13	Int32	106	JAN13
		CMP_BCS_ITF	Int32	107	Interleaved 2 of 5
		CMP_BCS_Codabar	Int32	108	Codabar
		CMP_BCS_Code39	Int32	109	Code39
		CMP_BCS_Code93	Int32	110	Code93
		CMP_BCS_Code128	Int32	111	Code128
		CMP_BCS_GS1DATABAR	Int32	131	GS1 DataBar Omnidirectional
		CMP_BCS_GS1DATABAR_E	Int32	132	GS1 DataBar Expanded
		CMP_BCS_GS1DATABAR_S	Int32	133	GS1 DataBar Stacked
		CMP_BCS_GS1DATABAR_E_S	Int32	134	GS1 DataBar Expanded Stacked
		CMP_BCS_GS1DATABAR_T	Int32	135	GS1 DataBar Truncated
		CMP_BCS_GS1DATABAR_L	Int32	136	GS1 DataBar Limited
		CMP_BCS_GS1DATABAR_S_O	Int32	137	GS1 DataBar Stacked Omnidirectional
11	可視コード	CMP_HRI_TEXT_NONE	Int32	0	なし
		CMP_HRI_TEXT_ABOVE	Int32	1	バーコードの上
		CMP_HRI_TEXT_BELOW	Int32	2	バーコードの下
12	エラー修正レベル(PDF417)	CMP_PDF417_EC_LEVEL_0	Int32	48	レベル 0
		CMP_PDF417_EC_LEVEL_1	Int32	49	レベル 1
		CMP_PDF417_EC_LEVEL_2	Int32	50	レベル 2
		CMP_PDF417_EC_LEVEL_3	Int32	51	レベル 3
		CMP_PDF417_EC_LEVEL_4	Int32	52	レベル 4
		CMP_PDF417_EC_LEVEL_5	Int32	53	レベル 5
		CMP_PDF417_EC_LEVEL_6	Int32	54	レベル 6
		CMP_PDF417_EC_LEVEL_7	Int32	55	レベル 7

		CMP_PDF417_EC_LEVEL_8	Int32	56	レベル 8
13	エラー修正レベル(QR Code)	CMP_QRCODE_EC_LEVEL_L	Int32	48	レベル L(7%)
		CMP_QRCODE_EC_LEVEL_M	Int32	49	レベル M(15%)
		CMP_QRCODE_EC_LEVEL_Q	Int32	50	レベル Q(25%)
		CMP_QRCODE_EC_LEVEL_H	Int32	51	レベル H(30%)
14	カット種類	CMP_CUT_FULL	Int32	-1	フルカット
		CMP_CUT_PARTIAL	Int32	-2	パーシャルカット
		CMP_CUT_FULL_PREFEED	Int32	-3	カット位置送り後フルカット
		CMP_CUT_PARTIAL_PREFEED	Int32	-4	カット位置送り後パーシャルカット
15	マークフィード種類	CMP_MF_TO_CUTTER	Int32	2	フィードしてカット
		CMP_MF_TO_NEXT_TOF	Int32	8	次の用紙までフィード
16	ドロワー番号	CMP_DRAWER_1	Int32	1	ドロワー1
		CMP_DRAWER_2	Int32	2	ドロワー2
17	一括処理コントロール	CMP_TP_TRANSACTION	Int32	11	一括処理の開始
		CMP_TP_NORMAL	Int32	12	一括処理の印刷
18	回転方向コントロール	CMP_RT_NORMAL	Int32	0x0001	回転印刷の終了
		CMP_RT_ROTATE180	Int32	0x0103	倒立印刷の開始
		CMP_RP_BARCODE	Int32	0x1000	バーコード回転の開始
		CMP_RP_BITMAP	Int32	0x2000	画像回転の開始
19	ページモードコントロール	CMP_PM_PAGE_MODE	Int32	1	ページモードの開始
		CMP_PM_PRINT_SAVE	Int32	2	印刷とデータ保持
		CMP_PM_NORMAL	Int32	3	印刷とページモード終了
		CMP_PM_CANCEL	Int32	4	ページモードキャンセル
20	ページモード印刷方向	CMP_PD_LEFT_TO_RIGHT	Int32	1	通常印刷方向
		CMP_PD_BOTTOM_TO_TOP	Int32	2	左 90 度回転印刷
		CMP_PD_RIGHT_TO_LEFT	Int32	3	180 度回転印刷
		CMP_PD_TOP_TO_BOTTOM	Int32	4	右 90 度回転印刷
21	透かし印刷コントロール	CMP_WM_STOP	Int32	0	透かし印刷の終了
		CMP_WM_START	Int32	1	透かし印刷の開始
22	マップモード種類	CMP_MM_DOTS	Int32	1	POS プリンターのドット幅
		CMP_MM_TWIPS	Int32	2	1 インチの 1/1440
		CMP_MM_ENGLISH	Int32	3	0.001 インチ
		CMP_MM_METRIC	Int32	4	0.01 ミリメートル
23	拡張エラーコード	CMP_EX_DEV_NO_PRINTER	Int32	62000	プリンター未接続
		CMP_EX_DEV_NO_SERIALNO	Int32	62012	プリンターシリアル番号一致なし
		CMP_EX_DEV_OPEN_ERROR	Int32	62100	ソケットまたはストリーム接続失敗
		CMP_EX_DEV_NO_RESPONSE	Int32	62102	コマンド無応答

3. ラインディスプレイ制御

3.1. プログラム構造

本 SDK を使用する場合のプログラム構造は、以下の通りです。

<pre> var display: LineDisplay? = CSJPOSLibSwift.LineDisplay() var result: Int32 = 0 // ディスプレイへ接続 result = display!.connect(LineDisplayConst.CDP_PORT_WiFi, withAddress: "192.168.0.10") if LineDisplayConst.CMP_SUCCESS == result { // エンコード設定 _ = display!.setEncoding(String.Encoding.shiftJIS) // 文字を消去 _ = display!.clearDisplay(LineDisplayConst.CDP_AREA_ALL) // テキスト表示 _ = display!.displayText("123456") // カーソルの位置指定 _ = display!.setCursorPosition(1, y: 2) // テキスト表示(反転) result = display!.displayText("123456", reverseFlag: true) // 切断処理 _ = printer!.disconnect() if LineDisplayConst.CMP_SUCCESS != result { messageBox("DisplayText Error : \(result)", withTitle: "Error", withAutoDismiss: true) } }else{ // 接続失敗 messageBox("Connect Error : \(result)", withTitle: "Error", withAutoDismiss: true) } </pre>	<div style="margin-bottom: 10px;">} クラス定義</div> <div style="margin-bottom: 10px;">} 接続処理</div> <div style="margin-bottom: 10px;">} ディスプレイ 処理</div> <div style="margin-bottom: 10px;">} 切断処理</div>
---	---

3.2. 機能一覧

本 SDK は以下の機能を提供します。

メソッド一覧

No	機能	詳細
1	クラス生成 (インスタンス)	インスタンスです。
2	接続 (connect メソッド)	ディスプレイと接続します。
3	切断 (disconnect メソッド)	ディスプレイとの接続を切断します。
4	文字列表示 (displayText メソッド)	文字を表示します。
5	表示クリア (clearDisplay メソッド)	表示文字を消去します。
6	点滅表示 (blinkDisplay メソッド)	点滅表示させます。
7	スクリーンモード設定 (setDisplayMode メソッド)	ディスプレイモードを設定します。
8	ディスプレイ設定 (setDisplayConfig メソッド)	各種設定を行います。
9	カーソル設定 (setCursorPosition メソッド)	カーソルの位置指定をします。
10	カーソル移動 (moveCursor メソッド)	カーソルを移動します。
11	カーソル型の設定 (setCursorType メソッド)	カーソル位置を表示します。
12	初期化 (initializeDisplay メソッド)	ディスプレイを初期化します。
13	コマンド送信 (displayData メソッド)	コマンドを送信します
14	文字エンコード指定 (setEncoding メソッド)	文字エンコードを指定します。
15	コードページ指定 (setCodePage メソッド)	コードページを指定します。
16	国際文字指定 (setInternationalCharacterSet メソッド)	国際文字を指定します。
17	ディスプレイ状態確認 (displayCheck メソッド)	ディスプレイの接続状態を確認します。
18	ログ設定 (setLog メソッド)	ログ機能を設定します。
19	SDK バージョン番号取得 (getVersionCode メソッド)	バージョン番号を取得します。
20	SDK バージョン文字列取得 (getVersionName メソッド)	バージョン文字列を取得します。

プロパティ一覧

No	機能	属性	詳細
1	拡張エラーコード取得 (ErrorCodeExtended プロパティ)	R	接続時の拡張エラーコードを示します。

3.3. SDK インターフェース

本 SDK のインターフェースを以下に示します。

3.3.1. 戻り値

以降に示すメソッドは、下記の値を返します。

戻り値	説明
CDP_SUCCESS (0)	正常終了
CDP_E_CONNECTED (1001)	デバイスへ既に接続済みです。
CDP_E_DISCONNECT (1002)	デバイスへ接続していません。
CDP_E_NOTCONNECT (1003)	デバイスへ接続できませんでした。
CDP_E_CONNECT_NOTFOUND (1004)	デバイス接続後の対応機種確認に失敗しました。
CDP_E_ILLEGAL (1101)	サポートされていない処理または無効なパラメータ値です。
CDP_E_OFFLINE (1102)	デバイスがオフラインです。
CDP_E_FAILURE (1104)	要求された処理が実行できません。

3.3.2. インスタンス

形式

LineDisplay

説明

クラスを生成し、初期化します。

使用例

```
var display: LineDisplay? = CSJPOSLibSwift.LineDisplay()
```

3.3.3. connect メソッド

形式

- 1) func connect(connectType: Int32, withAddress addr: String?) -> Int32
- 2) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32) -> Int32
- 3) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32, withTimeout timeout: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
connectType	[IN]	接続されているプリンターの接続タイプ	CDP_PORT_WiFi: Wi-Fi 接続 CDP_PORT_BLUETOOTH: Bluetooth 接続 CDP_PORT_USB: USB 接続 (Lightning/Type-C)
addr	[IN]	接続先の IP アドレス または Bluetooth デバイスアドレス または Bluetooth デバイス名 またはシリアル番号	WiFi : 0.0.0.0 ~ 255.255.255.255 Bluetooth : 00:00:00:00:00:00 ~ FF:FF:FF:FF:FF:FF デバイス名 USB : 空白またはシリアル番号
port	[IN]	接続先ポート番号 または USB ストリーム番号	WiFi : ポート番号 USB : 1 ~ 3
timeout	[IN]	タイムアウト (msec)	

説明

このメソッドは、ラインディスプレイと接続するために使用します。ラインディスプレイが接続されているプリンターの接続タイプとアドレスを指定してください。

Bluetooth 接続の場合、iOS デバイスの Bluetooth 設定画面にてペアリング済でかつ接続された状態のプリンターにだけ接続する事が出来ます(「[1.7 Bluetooth の利用方法](#)」を参照してください)。また Bluetooth デバイスアドレスを利用して接続するには iOS6 以上である必要があります。それ以外は Bluetooth デバイス名を利用して接続してください。

USB 接続の場合、アドレスに空白を指定した場合は自動的に接続されます。プリンターのシリアル番号を指定して特定のプリンターへ接続する事も可能です。

接続先ポート番号は、接続タイプに Wi-Fi または USB を指定した場合のみ有効です。省略された場合は、Wi-Fi の場合はポートの 9200 番、USB の場合はストリームの 1 番で接続します。

タイムアウトは、ラインディスプレイへの接続の最大時間(ミリ秒単位)を指定します。省略された場合は、Wi-Fi/USB の場合は 4000 ミリ秒、Bluetooth の場合は 8000 ミリ秒で接続します。

ラインディスプレイとの通信が不要になった場合は、必ず [disconnect メソッド](#) を実行し、ラインディスプレイとの接続を切断してください。切断しなかった場合は、次の接続がエラーとなります。

戻り値

成功時は CDP_SUCCESS(0)を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「[3.3.1. 戻り値](#)」を参照してください。

エラーコード	説明
CDP_E_NOTCONNECT (1003)	デバイスへ接続できませんでした。 ①ラインディスプレイが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない

使用例

```
display!.connect(LineDisplayConst.CDP_PORT_WiFi,  
    withAddress: "192.168.182.100", withPort: 9200)  
  
display!.connect(LineDisplayConst.CDP_PORT_BLUETOOTH,  
    withAddress: "00:01:90:F0:81:AB", withPort: 9200,  
    withTimeout:8000)  
  
display!.connect(LineDisplayConst.CDP_PORT_USB, withAddress: "")
```


3.3.4. disconnect メソッド

形式

```
func disconnect() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、ラインディスプレイの接続を切断するために使用します。

印刷の終了、あるいは、何らかのエラーが発生した場合は、本メソッドを実行して接続を切断してください。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.disconnect()
```

3.3.5. displayText メソッド

形式

```
func displayText(data: String?) -> Int32  
func displayText(data:String?, reverseFlag flag: Bool) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
data	テキストデータ	String
reverseFlag	反転指定フラグ	false: 標準 true: 反転 引数省略時には、false として扱います。

説明

このメソッドは、現在のカーソル位置からテキストを表示するために使用します。
テキスト属性は、反転を指定可能です。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.displayText("Hello, World!")
```

3.3.6. clearDisplay メソッド

形式

```
func clearDisplay(displayArea: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
displayArea	消去領域	CDP_AREA_ALL(0): 全領域 CDP_AREA_CURSORLINE(1): カーソル行 引数省略時には、AREA_ALL として扱います。

説明

このメソッドは、表示中の文字を消去します。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.clearDisplay(LineDisplayConst.CDP_AREA_ALL)
```

3.3.7. blinkDisplay メソッド

形式

```
func blinkDisplay(intervalBlinkg: UInt32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
intervalBlink	点滅間隔(ミリ秒)	0～

説明

このメソッドは、表示画面全体を点滅させます。

点滅間隔(ミリ秒)は、点灯と消灯の間隔を指定します。点滅間隔に、0 を指定すると、点滅は解除されます。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.blinkDisplay(1000)
```

3.3.8. setDisplayMode メソッド

形式

```
func setDisplayMode(displayMode: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
displayMode	ディスプレイモード	CDP_OVERWRITE(1): 上書きモード CDP_VERTICALSCROLL(2): 垂直スクロールモード CDP_HORIZONTALSCROLL(3): 水平スクロールモード

説明

このメソッドは、以下のディスプレイのモードを設定します。

DisplayMode	概要
Overwrite	カーソル位置の文字を上書きし、カーソルを右に移動。 (カーソルが上端右端のときの入力は、カーソルを下端左端に移動、 カーソルが下端右端のときの文字入力は、カーソルを上端左端に移動)
VerticalScroll	カーソルが上端のときのカーソル上移動(または、左端での左移動)で、上端の表示行を下端にスクロール カーソルが下端のときのカーソル下移動(または、右端での右移動)で、下端の表示行を上端にスクロール
HorizontalScroll	カーソルが右端でのカーソル右移動(または、文字入力)で、現在のカーソル行に対して、左方向に文字をスクロール カーソル左端のカーソル左移動で、現在のカーソル行に対して、右方向に文字をスクロール

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.setDisplayMode(LineDisplayConst.CDP_VERTICALSCROLL)
```

3.3.9. setDisplayConfig メソッド

形式

```
func setDisplayConfig(brightness: UInt32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
brightness	輝度(%)	0~100

説明

このメソッドは、表示画面の輝度を変更します。

輝度は、数値が大きいほど明るくなります。0を指定すると、画面を消灯します(表示内容は保持されます)。

設定後、表示画面全体の点滅は解除されます。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.setDisplayConfig(40)
```

3.3.10. setCursorPosition メソッド

形式

```
func setPosition(x: UInt32, y: UInt32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
x	桁位置	1～
y	行位置	1～

説明

このメソッドは、カーソル位置を設定するために使用します。

カーソル位置は、カーソルの移動座標で、桁位置と行位置を指定します。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.setPosition(1, y:2)
```

3.3.11. moveCursor メソッド

形式

```
func moveCursor(dx: Int32, y dy: Int32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
dx	左右方向移動量	-128～127
dy	上下方向移動量	-128～127

説明

このメソッドは、カーソルを移動するために使用します。

現在のカーソル位置からの移動となります。カーソル移動量は、左右方向移動量(-:左方向, +:右方向)と上下方向移動量(-:上方向, +:下方向)を指定します。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.moveCursor(2, dy:0)
```


3.3.12. setCursorPosition メソッド

形式

```
func setCursorPosition(cursorType: UInt32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

要素	意味	設定可能範囲
cursorType	カーソル型指定	CDP_TYPE_NONE(0): カーソル非表示 CDP_TYPE_UNDERLINE(1): カーソル表示 (省略可能要素, 省略時は TYPE_UNDERLINE)

説明

現在のカーソル位置をディスプレイに表示します。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.setCursorPosition(LineDisplayConst.CDP_TYPE_UNDERLINE)
```

3.3.13. initializeDisplay メソッド

形式

```
func initializeDisplay() -> Int32
```

パラメータ

ありません。

説明

デバイスを初期化します。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.initializeDisplay()
```

3.3.14. displayData タグ

形式

```
func displayData(data: UnsafeMutablePointer<UInt8>, withLength length: UInt) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

要素	意味	設定可能範囲
data	送信データ	UnsafeMutablePointer<Int8>

説明

バイトデータをそのままデバイスに送信するために使用します。
ご使用の際は、他のメソッドに影響を与えない様に注意する必要があります。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
// セルフテスト実行
var data: [Int8] = [0x1f, 0x40]
res = display!.displayData(&data)
```

3.3.15. setEncoding メソッド

形式

```
func setEncoding(charset: String.Encoding) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
charset	文字セット名	NSStringEncoding

説明

このメソッドは、プリンター送信データのエンコードを設定するために使用します。

インスタンスを生成時に OS のデフォルト文字セットに初期化します。

日本語で使用する場合は、NSStringEncoding 指定する必要があります。

設定可能範囲の詳細につきましては以下の URL を参照ください。

http://developer.apple.com/library/mac/#documentation/Cocoa/Reference/Foundation/Classes/NSString_Class/Reference/NSString.html#//apple_ref/doc/uid/20000154-BAJJAICE

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
// 日本語
display!.setEncoding(NSShiftJISStringEncoding)
// 中国語(Simplified Chinese)
let encode = CFStringConvertEncodingToNSStringEncoding( 0x0632 )
display!.setEncoding(encode)
// 中国語(Traditional Chinese)
let encode = CFStringConvertEncodingToNSStringEncoding( 0x0A03 )
display!.setEncoding(encode)
// 韓国語
let encode = CFStringConvertEncodingToNSStringEncoding( 0x0940 )
display!.setEncoding(encode)
```

3.3.16. setCodePage メソッド

形式

```
func setCodePage(codePage: UInt32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

要素	意味	設定可能範囲
codePage	コードページ指定	0～255

説明

設定値につきましては、利用デバイスのコマンドリファレンス”ESC t” コマンドを参照してください。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
display!.setCodePage(1)
```

3.3.17. setInternationalCharacterSet メソッド

形式

```
func setInternationalCharacterSet(characterSet: UInt32) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	意味	設定可能範囲
characterSet	国際文字指定	0～16

説明

以下の国際文字セットを設定します。

CharacterSet	国際文字セット	CharacterSet	国際文字セット
0	アメリカ	9	ノルウェー
1	フランス	10	デンマークⅡ
2	ドイツ	11	スペインⅡ
3	イギリス	12	ラテンアメリカ
4	デンマークⅠ	13	韓国
5	スウェーデン	14	クロアチア
6	イタリア	15	中国
7	スペインⅠ	16	ベトナム
8	日本		

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.2.1.戻り値](#)」を参照してください。

使用例

```
display!.setInternationalCharacterSet(8)
display!.displayText("Total:¥¥1,010")
```

3.3.18. displayCheck メソッド

形式

```
func displayCheck () -> Int32
```

パラメータ

ありません。

説明

このメソッドは、ディスプレイの接続状態を確認するために使用します。

本メソッドの実行結果が成功の場合は、ラインディスプレイが接続されている事が確認できます。

本メソッドの実行結果が失敗の場合は、通信異常やデバイスの異常が発生した可能性があります。この場合、[disconnect メソッド](#)および [connect メソッド](#)を使用して再接続してください。

ネットワーク接続の場合、長時間放置すると自動的に切断されます。接続を保持する場合は、定期的に本メソッドを実行してください。

戻り値

成功時は CDP_SUCCESS(0)を返します。それ以外のエラーコードは「[3.3.1.戻り値](#)」を参照してください。

使用例

```
var result = display!.displayCheck()
```

3.3.19. setLog メソッド

形式

```
func setLog(mode: Int32, withPath path: String?, withMaxSize maxSize: Int32)
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
mode	[IN]	ログモード	0 : 記録なし 1 : アクセス履歴の記録 2 : エラーのみ記録
path	[IN]	格納フォルダ	アプリケーション共有フォルダ内のフォルダ
maxSize	[IN]	ログサイズ	0: サイズ制限なし 1~: 最大サイズ(MB)

説明

このメソッドは、ログ機能を設定するために使用します。ログ機能の詳細は「[3.4.1 ログ機能について](#)」を参照してください。

戻り値

ありません。

使用例

```
display!.setLog(1, withPath: "/", withMaxSize: 10)
```


3.3.20. getVersionCode メソッド

形式

```
func getVersionCode() -> Int32
```

戻り値

バージョン番号 : Number

パラメータ

ありません。

説明

このメソッドは、SDK のバージョン番号を数値で取得するために使用します。

戻り値

SDK のバージョン番号を数値 (Ver1.23 の場合:123) で取得します。

使用例

```
var vno = display!.getVersionCode()  
alert("SDK Version:" + vno/100)
```

3.3.21. getVersionName メソッド

形式

```
func getVersionName() -> String?
```

戻り値

バージョン文字列 : String

パラメータ

ありません。

説明

このメソッドは、SDK のバージョン番号を数値で取得するために使用します。

戻り値

SDK のバージョン番号を文字列 (Ver1.00 の場合 : "1.00") で取得します。

使用例

```
var vname = display!.getVersionName()  
alert("SDK Version:" + vname)
```

3.3.22. ErrorCodeExtended プロパティ

形式

Int32

属性

Read

説明

このプロパティは、ディスプレイへの接続がエラーになった際の拡張エラーコードを保持します。値は以下の通りです。

拡張エラーコード	説明
CDP_EX_DEV_NO_PRINTER (62000)	ディスプレイが接続されているプリンターへ接続できませんでした。 ①プリンターが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない
CDP_EX_DEV_NO_DEVICE (62001)	ディスプレイが接続されていません。 ①ディスプレイが未接続 ②インターフェースポートの設定が正しくない
CDP_EX_DEV_USING (62002)	ディスプレイが使用中です。 ①他アプリでディスプレイを使用中
CDP_EX_DEV_CONFIRM (62003)	ディスプレイの接続後確認に失敗しました。 ①指定ポート番号の間違い
CDP_EX_DEV_NO_STREAMNO (62011)	ストリーム番号が一致しません。(USB 接続のみ) ①指定ポート番号の間違い
CDP_EX_DEV_NO_SERIALNO (62012)	プリンターシリアル番号が一致しません。(USB 接続のみ) ①指定シリアル番号の間違い
CDP_EX_DEV_STATUS_GET_ERROR (62013)	デバイス情報取得に失敗しました。(USB 接続のみ) ①通信異常が発生した
CDP_EX_DEV_OPEN_ERROR (62100)	ソケットまたはストリーム接続できませんでした。 ①通信異常が発生した
CDP_EX_DEV_SEND_ERROR (62101)	ディスプレイへのコマンド送信が失敗しました。 ①通信異常が発生した
CDP_EX_DEV_NO_RESPONSE (62102)	ディスプレイからのコマンド応答がありません。 ①対応機種でないディスプレイが接続されている ②インターフェースボードの設定が正しくない

このプロパティは、インスタンス時と接続処理開始時に 0 に初期化されます。[connect メソッド](#)実行後に参照してください。

設定方法

なし。

取得方法

```
func getResultCodeExtended() -> Int32
```

戻り値として、拡張エラーコードを返します。

3.4. 注意事項

3.4.1. ログ機能について

本 SDK は、メソッドの実行やプロパティの読み書きを記録するログ機能をサポートしています。ログ機能を設定する際は、[setLog メソッド](#)を使用するか、次の書式の設定ファイル「CSJPOSLibD.cfg」をアプリケーションの共有フォルダに配置してください。

<CSJPOSLibD.cfg の例>

[LogSetting]	・・・セクション名(固定)
LogMode=1	・・・ログモードを指定
LogPath=/	・・・ログファイルを格納するアプリケーション共有フォルダ内のフォルダを指定
LogMaxSize=10	・・・ログファイルの最大サイズを MB 単位で指定

設定項目

・ログモード

ログを記録するモードを指定します。

- 0: 記録なし
- 1: アクセス履歴の記録
- 2: エラーのみ記録

・格納フォルダ

ログファイルを格納するフォルダを指定します。本設定が指定されていない場合は、アプリケーション共有フォルダに格納されます。

・ログサイズ

ログファイルの最大容量を MB 単位で指定します。0 を指定した場合は容量制限が解除され可能な限り記録されます。

ログファイル名

ログファイルの拡張子は「.log」です。ファイル名は「CSJPOSLibD」の後ろに曜日を表す数字が追加されます。曜日は日曜日を 0、月曜日を 1 として 0 から 6 のいずれかの数字が加えられます。

例) CSJPOSLibD_1.log

ログファイルが既に存在し、それが当日以外の場合は、古いファイルを削除してからログを記録します。

ログフォーマット

ログ機能は、メソッド、プロパティの日付、時間、結果のアクセス情報を記録します。

--- メソッドの例1 (connect) ---

```
2022/04/06 12:54:21.165 001 METHOD call    connect(0, 192.168.234.100, 9200, 4000)
2022/04/06 12:54:25.970 001 METHOD result connect() -> Success(0)
```

--- メソッドの例2 (displayText) ---

```
2022/04/06 13:13:03.795 001 METHOD call    displayText([See below], 0)
-----Parameter Detail-----
Thank you so much!
-----
2022/04/06 13:13:03.798 001 METHOD result displayText() -> Success(0)
```

※ログ機能を使用する場合、全てのメソッドとプロパティのアクセス時にログファイルが更新されますので、SDK の処理が低下してしまうことがあります。

※次のような理由などでファイルの書き込みができない場合はログファイルの記録が行われません。このような場合エラーメッセージなどは表示されませんので、ご注意ください。

- ・書き込み禁止デバイスを指定した場合
- ・出力先に十分な領域が残っていない場合
- ・書き込み禁止のログファイルがある場合
- ・ファイルやフォルダのアクセス権がない場合
- ・他のアプリケーションがログファイルを使用している場合

3.4.2. 定数定義一覧

No	項目	定数名	型	値	説明
1	処理結果	CDP_SUCCESS	Int32	0	正常終了
		CDP_E_CONNECTED	Int32	1001	接続済み
		CDP_E_DISCONNECT	Int32	1002	未接続
		CDP_E_NOTCONNECT	Int32	1003	接続失敗
		CDP_E_ILLEGAL	Int32	1101	未対応処理または無効パラメータ
		CDP_E_OFFLINE	Int32	1102	オフライン
		CDP_E_FAILURE	Int32	1104	処理異常
2	接続インターフェース	CDP_PORT_WiFi	Int32	0	ネットワーク接続
		CDP_PORT_BLUETOOTH	Int32	1	Bluetooth 接続
		CDP_PORT_USB	Int32	3	USB 接続(Lightning/Type-C)
3	消去領域	CDP_AREA_ALL	Int32	0	全領域
		CDP_AREA_CURSORLINE	Int32	1	カーソル行
4	ディスプレイモード	CDP_OVERWRITE	Int32	1	上書きモード
		CDP_VERTICALSCROLL	Int32	2	垂直スクロールモード
		CDP_HORIZONTALSCROLL	Int32	3	水平スクロールモード
5	カーソル型指定	CDP_TYPE_NONE	Int32	0	カーソル非表示
		CDP_TYPE_UNDERLINE	Int32	1	カーソル表示
6	拡張エラーコード	CDP_EX_DEV_NO_PRINTER	Int32	62000	プリンター未接続
		CDP_EX_DEV_NO_DEVICE	Int32	62001	周辺機器デバイス未接続
		CDP_EX_DEV_USING	Int32	62002	周辺機器デバイス使用中
		CDP_EX_DEV_CONFIRM	Int32	62003	周辺機器デバイス接続後確認失敗
		CDP_EX_DEV_NO_STREAMNO	Int32	62011	ストリーム番号一致なし
		CDP_EX_DEV_NO_SERIALNO	Int32	62012	シリアル番号一致なし
		CDP_EX_DEV_STATUS_GET_ERROR	Int32	62013	デバイス情報取得失敗
		CDP_EX_DEV_OPEN_ERROR	Int32	62100	ソケットまたはストリーム接続失敗
		CDP_EX_DEV_SEND_ERROR	Int32	62101	コマンド送信失敗
		CDP_EX_DEV_NO_RESPONSE	Int32	62102	コマンド無応答

3.4.3. ラインディスプレイを利用する際の SDK バージョンについて

ラインディスプレイの機能を利用する場合、Swift4.0 以降の SDK をご使用ください。

4. バーコードスキャナー制御

4.1. プログラム構造

本 SDK を使用する場合は、以下の通りです。

var scanner: CSJPOSLibSwift.Scanner?	} クラス定義
override func viewDidLoad() { scanner = CSJPOSLibSwift.Scanner() _ = scanner?.setStatusUpdateEventCallback(self) _ = scanner?.setDataEventCallback(self) }	} インスタンス } コールバック登録
// データイベントコールバック設定 func dataEventCallback(_ data: Data!) { messageBox(String(data: data, encoding: .utf8)!, withTitle: "Data Event", withAutoDismiss: false) }	} コールバック処理
// ステータスイベントコールバック設定 func statusUpdateEventCallback(_ status: Int32) { messageBox("Status : \(status)", withTitle: "Status Update Event", withAutoDismiss: true) }	
// スキャナーへ接続 func connectScanner() { let result = scanner!.connect(ScannerConst.CSC_PORT_WiFi, withAddress: "192.168.0.10") if ScannerConst.CSC_SUCCESS != result { // 接続失敗 messageBox("Connect Error : \(result)", withTitle: "Error", withAutoDismiss: true) } }	} 接続処理
// スキャナーを切断 func disconnectScanner() { scanner!.disconnect() }	} 切断処理

4.2. 機能一覧

本 SDK は以下の機能を提供します。

メソッド一覧

No	機能	詳細
1	クラス生成 (インスタンス)	インスタンスです。
2	接続 (connect メソッド)	スキャナーと接続します。
3	切断 (disconnect メソッド)	スキャナーとの接続を切断します。
4	ログ設定 (setLog メソッド)	ログ機能を設定します。
5	SDK バージョン番号取得 (getVersionCode メソッド)	バージョン番号を取得します。
6	SDK バージョン文字列取得 (getVersionName メソッド)	バージョン文字列を取得します。

プロパティ一覧

No	機能	属性	詳細
1	拡張エラーコード取得 (ErrorCodeExtended プロパティ)	R	接続時の拡張エラーコードを示します。

イベント一覧

No	機能	詳細
1	スキャンデータ通知 (DataEvent イベント)	バーコード読み取りデータを通知します。
2	ステータス変化通知 (StatusUpdateEvent イベント)	デバイスのステータス変化を通知します。

4.3. SDK インターフェース

本 SDK のインターフェースを以下に示します。

4.3.1. 戻り値

以降に示すメソッドは、下記の値を返します。

戻り値	説明
CSC_SUCCESS (0)	正常終了
CSC_E_CONNECTED (1001)	デバイスへ既に接続済みです。
CSC_E_DISCONNECT (1002)	デバイスへ接続していません。
CSC_E_NOTCONNECT (1003)	デバイスへ接続できませんでした。
CSC_E_CONNECT_NOTFOUND (1004)	デバイス接続後の対応機種確認に失敗しました。
CSC_E_ILLEGAL (1101)	サポートされていない処理または無効なパラメータ値です。
CSC_E_OFFLINE (1102)	デバイスがオフラインです。
CSC_E_FAILURE (1104)	要求された処理が実行できません。

4.3.2. インスタンス

形式

Scanner

説明

クラスを生成し、初期化します。

使用例

```
var scanner: Scanner? = CSJPOSLibSwift.Scanner()
```

4.3.3. connect メソッド

形式

- 1) func connect(connectType: Int32, withAddress addr: String?) -> Int32
- 2) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32) -> Int32
- 3) func connect(connectType: Int32, withAddress addr: String?, withPort port: Int32, withTimeout timeout: Int32) -> Int32

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
connectType	[IN]	接続されているプリンターの接続タイプ	CDP_PORT_WiFi: Wi-Fi 接続 CDP_PORT_BLUETOOTH: Bluetooth 接続 CDP_PORT_USB: USB 接続 (Lightning/Type-C)
addr	[IN]	接続先の IP アドレス または Bluetooth デバイスアドレス または Bluetooth デバイス名 またはシリアル番号	WiFi : 0.0.0.0 ~ 255.255.255.255 Bluetooth : 00:00:00:00:00:00 ~ FF:FF:FF:FF:FF:FF デバイス名 USB : 空白またはシリアル番号
port	[IN]	接続先ポート番号 または接続先ストリーム番号	WiFi : ポート番号 USB : 1 ~ 3
timeout	[IN]	タイムアウト (msec)	

説明

このメソッドは、バーコードスキャナーと接続するために使用します。バーコードスキャナーが接続されているプリンターの接続タイプとアドレスを指定してください。

Bluetooth 接続の場合、iOS デバイスの Bluetooth 設定画面にてペアリング済でかつ接続された状態のプリンターにだけ接続する事が出来ます(「[1.7 Bluetooth の利用方法](#)」を参照してください)。また Bluetooth デバイスアドレスを利用して接続するには iOS6 以上である必要があります。それ以外は Bluetooth デバイス名を利用して接続してください。

USB 接続の場合、アドレスに空白を指定した場合は自動的に接続されます。プリンターのシリアル番号を指定して特定のプリンターへ接続する事も可能です。

接続先ポート番号は、接続タイプに Wi-Fi または USB を指定した場合のみ有効です。省略された場合は、Wi-Fi の場合はポートの 9210 番、USB の場合はストリームの 2 番で接続します。

タイムアウトは、バーコードスキャナーへの接続の最大時間(ミリ秒単位)を指定します。省略された場合は、Wi-Fi/USB の場合は 4000 ミリ秒、Bluetooth の場合は 8000 ミリ秒で接続します。

バーコードスキャナーとの通信が不要になった場合は、必ず [disconnect メソッド](#)を実行し、接続を切断してください。切断しなかった場合は、次の接続がエラーとなります。

戻り値

成功時は CSC_SUCCESS(0)を返します。失敗時は下記のエラーコードの説明を確認してください。

それ以外のエラーコードは「[4.3.1 戻り値](#)」を参照してください。

エラーコード	説明
CSC_E_NOTCONNECT (1003)	バーコードスキャナーへ接続できませんでした。 ①バーコードスキャナーが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない

使用例

```
scanner!.connect(ScannerConst.CSC_PORT_WiFi,  
    withAddress: "192.168.182.100", withPort: 9210)  
  
scanner!.connect(ScannerConst.CSC_PORT_BLUETOOTH,  
    withAddress: "00:01:90:F0:81:AB", withPort: 9210,  
    withTimeout:8000)  
  
scanner!.connect(ScannerConst.CSC_PORT_USB, withAddress: "")
```

4.3.4. disconnect メソッド

形式

```
func disconnect() -> Int32
```

パラメータ

ありません。

説明

このメソッドは、スキャナーとの接続を切断するために使用します。

印刷の終了、あるいは、何らかのエラーが発生した場合は、本メソッドを実行して接続を切断してください。

戻り値

成功時は CSC_SUCCESS(0)を返します。それ以外のエラーコードは「[4.3.1 戻り値](#)」を参照してください。

使用例

```
scanner!.disconnect()
```

4.3.5. setLog メソッド

形式

```
func setLog(mode: Int32, withPath path: String?, withMaxSize maxSize: Int32)
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味	設定可能範囲
mode	[IN]	ログモード	0 : 記録なし 1 : アクセス履歴の記録 2 : エラーのみ記録
path	[IN]	格納フォルダ	アプリケーション共有フォルダ内のフォルダ
maxSize	[IN]	ログサイズ	0: サイズ制限なし 1~: 最大サイズ(MB)

説明

このメソッドは、ログ機能を設定するために使用します。ログ機能の詳細は「[4.4.1 ログ機能について](#)」を参照してください。

戻り値

ありません。

使用例

```
scanner!.setLog(1, withPath: "/", withMaxSize: 10)
```

4.3.6. getVersionCode メソッド

形式

```
func getVersionCode() -> Int32
```

戻り値

バージョン番号 : Number

パラメータ

ありません。

説明

SDK のバージョン番号を数値 (Ver1.23 の場合:123) で取得します。

使用例

```
var vno = scanner!.getVersionCode();  
alert("SDK Version:" + vno/100);
```

4.3.7. getVersionName メソッド

形式

```
func getVersionName() -> String?
```

戻り値

バージョン文字列 : String

パラメータ

ありません。

説明

SDK のバージョン番号を文字列で取得します。

使用例

```
var vname = scanner!.getVersionName();  
alert("SDK Version:" + vname);
```


4.3.8. ErrorCodeExtended プロパティ

形式

Int32

属性

Read

説明

このプロパティは、スキャナーへの接続がエラーになった際の拡張エラーコードを保持します。値は以下の通りです。

拡張エラーコード	説明
CSC_EX_DEV_NO_PRINTER (62000)	スキャナーが接続されているプリンターへ接続できませんでした。 ①プリンターが未接続 ②プリンターの電源が入っていない ③インターフェースポートのハンドルを取得できない
CSC_EX_DEV_NO_DEVICE (62001)	スキャナーが接続されていません。 ①スキャナーが未接続 ②インターフェースポートの設定が正しくない
CSC_EX_DEV_USING (62002)	スキャナーが使用中です。 ①他アプリでスキャナーを使用中
CSC_EX_DEV_CONFIRM (62003)	スキャナーの接続後確認に失敗しました。 ①指定ポート番号の間違い
CSC_EX_DEV_NO_STREAMNO (62011)	ストリーム番号が一致しません。(USB 接続のみ) ①指定ポート番号の間違い
CSC_EX_DEV_NO_SERIALNO (62012)	プリンターシリアル番号が一致しません。(USB 接続のみ) ①指定シリアル番号の間違い
CSC_EX_DEV_STATUS_GET_ERROR (62013)	デバイス情報取得に失敗しました。(USB 接続のみ) ①通信異常が発生した
CSC_EX_DEV_OPEN_ERROR (62100)	ソケットまたはストリーム接続できませんでした。 ①通信異常が発生した

このプロパティは、インスタンス時と接続処理開始時に 0 に初期化されます。[connect メソッド](#)実行後に参照してください。

設定方法

なし。

取得方法

```
func getResultCodeExtended() -> Int32
```

戻り値として、拡張エラーコードを返します。

4.3.9. DataEvent イベント

形式

```
func dataEventCallback(_ data: Data!)
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味
data	[IN]	スキャンデータ

説明

バーコードスキャナーがバーコードから読み取った情報を通知します。

コールバックメソッドは、バーコードスキャナーがバーコードから読み取った情報として、Data 型の引き数を受け取ります。

設定方法

```
func setDataEventCallback(callback: Any!) -> Int32
```

バーコードスキャナーがバーコードから読み取った情報を受け取るコールバックメソッドを登録します。

setDataEventCallback メソッドを複数回実行した場合は、最後に指定されたコールバックメソッドに上書きされます。null が設定された場合は、コールバックを解除します。

使用例

```
// Register the scan data callback method
scanner!.setDataEventCallback(self)

// callback method
func dataEventCallback(_ data: Data!) {
    // Data receive process
}
```

4.3.10. StatusUpdateEvent イベント

形式

```
func statusUpdateEventCallback(callback: Any!) -> Int32
```

パラメータ

パラメータの意味と設定可能な値は以下の通りです。

値	[IN/OUT]	意味
status	[IN]	デバイスのステータス変化

説明

ステータスが変化を通知します。

コールバックメソッドは、デバイスのステータス変化の情報として、ステータスを示す Int32 型の引き数を受け取ります。

ステータスコード	説明
CSC_SUE_POWER_ONLINE (2001)	デバイスはレディ状態です
CSC_SUE_POWER_OFF (2002)	通信異常または本体に接続されていません

設定方法

```
func setStatusUpdateEventCallback(_ callback: Any!) -> Int32
```

デバイスのステータス変化情報を受け取るコールバックメソッドを登録します。

setStatusUpdateEventCallback メソッドを複数回実行した場合は、最後に指定されたコールバックメソッドに上書きされます。null が設定された場合は、コールバックを解除します。

使用例

```
// Register the status update callback method
scanner!.setStatusUpdateEventCallback(self)

// callback method
func statusUpdateEventCallback(_ status: Int32) {
    if ScannerConst.CSC_SUE_POWER_ONLINE == status {
        // Online
    } else {
        // Offline
    }
}
```

4.4. 注意事項

4.4.1. ログ機能について

本 SDK は、メソッドの実行やプロパティの読み書きを記録するログ機能をサポートしています。ログ機能を設定する際は、[setLog メソッド](#)を使用するか、次の書式の設定ファイル「CSJPOSLibS.cfg」をアプリケーションの共有フォルダに配置してください。

<CSJPOSLibS.cfg の例>

[LogSetting]	・・・セクション名(固定)
LogMode=1	・・・ログモードを指定
LogPath=/	・・・ログファイルを格納するアプリケーション共有フォルダ内のフォルダを指定
LogMaxSize=10	・・・ログファイルの最大サイズを MB 単位で指定

設定項目

・ログモード

ログを記録するモードを指定します。

- 0: 記録なし
- 1: アクセス履歴の記録
- 2: エラーのみ記録

・格納フォルダ

ログファイルを格納するフォルダを指定します。本設定が指定されていない場合は、アプリケーション共有フォルダに格納されます。

・ログサイズ

ログファイルの最大容量を MB 単位で指定します。0 を指定した場合は容量制限が解除され可能な限り記録されます。

ログファイル名

ログファイルの拡張子は「.log」です。ファイル名は「CSJPOSLibS」の後ろに曜日を表す数字が追加されます。曜日は日曜日を 0、月曜日を 1 として 0 から 6 のいずれかの数字が加えられます。

例) CSJPOSLibS_1.log

ログファイルが既に存在し、それが当日以外の場合は、古いファイルを削除してからログを記録します。

ログフォーマット

ログ機能は、メソッド、プロパティの日付、時間、結果のアクセス情報を記録します。

--- メソッドの例(connect) ---

```
2022/04/06 12:54:30.450 001 METHOD call    connect(0, 192.168.234.100, 9210, 4000)
2022/04/06 12:54:30.888 001 METHOD result connect() -> Success(0)
```

--- イベントの例(DataEvent) ---

```
2022/04/06 13:13:21.720 015 EVENT          DataEvent -> <31323334 35363738 39303132>
```

※ログ機能を使用する場合、全てのメソッドとプロパティのアクセス時にログファイルが更新されますので、SDK の処理が低下してしまうことがあります。

※次のような理由などでファイルの書き込みができない場合はログファイルの記録が行われません。このよう

な場合エラーメッセージなどは表示されませんので、ご注意ください。

- ・書き込み禁止デバイスを指定した場合
- ・出力先に十分な領域が残っていない場合
- ・書き込み禁止のログファイルがある場合
- ・ファイルやフォルダのアクセス権がない場合
- ・他のアプリケーションがログファイルを使用している場合

4.4.2. 定数定義一覧

No	項目	定数名	型	値	説明
1	処理結果	CSC_SUCCESS	Int32	0	正常終了
		CSC_E_CONNECTED	Int32	1001	接続済み
		CSC_E_DISCONNECT	Int32	1002	未接続
		CSC_E_NOTCONNECT	Int32	1003	接続失敗
		CSC_E_ILLEGAL	Int32	1101	未対応処理または無効パラメータ
		CSC_E_OFFLINE	Int32	1102	オフライン
		CSC_E_FAILURE	Int32	1104	処理異常
2	接続インターフェース	CSC_PORT_WiFi	Int32	0	ネットワーク接続
		CSC_PORT_BLUETOOTH	Int32	1	Bluetooth 接続
		CSC_PORT_USB	Int32	3	USB 接続 (Lightning/Type-C)
3	ステータス	CSC_SUE_POWER_ONLINE	Int32	2001	レディ状態
		CSC_SUE_POWER_OFF	Int32	2002	通信異常または本体に接続されていない
4	拡張エラーコード	CSC_EX_DEV_NO_PRINTER	Int32	62000	プリンター未接続
		CSC_EX_DEV_NO_DEVICE	Int32	62001	周辺機器デバイス未接続
		CSC_EX_DEV_USING	Int32	62002	周辺機器デバイス使用中
		CSC_EX_DEV_CONFIRM	Int32	62003	周辺機器デバイス接続後確認失敗
		CSC_EX_DEV_NO_STREAMNO	Int32	62011	ストリーム番号一致なし
		CSC_EX_DEV_NO_SERIALNO	Int32	62012	シリアル番号一致なし
		CSC_EX_DEV_STATUS_GET_ERROR	Int32	62013	デバイス情報取得失敗
		CSC_EX_DEV_OPEN_ERROR	Int32	62100	ソケットまたはストリーム接続失敗

4.4.3. バーコードスキャナーを利用する際の SDK バージョンについて

バーコードスキャナーの機能を利用する場合、Swift4.0 以降の SDK をご使用ください。

iOS POS Print SDK (Swift) プログラムマニュアル

2024/07/18 Ver.2.12 用

シチズン・システムズ株式会社

<https://csj.citizen.co.jp/>