iOS SDK for Layout Designer2

ユーザーズ・マニュアル

2023年5月31日 Rev1.0.1

このマニュアルはiOS SDK for Layout Designer2に関する、お客様がアプリケーションの構築に必要な
設計ガイドラインの情報を示しています。

三栄電機株式会社

本書の改定履歴

改訂Rev	日付	改定内容
Rev1.0.0	2022年09月13日	初版リリース （SDKバージョン1.00、エンジンバージョン1.03）
Rev1.0.1	2023年05月31日	スタンドアロンマクロに対応

ご注意

・iOS SDK for LayoutDesigner2 は三栄電機株式会社（以下三栄電機といいます。）の著作物であり、
本製品にかかる著作権その他の権利は三栄電機に帰属する。
・三栄電機はiOS SDK for LayoutDesigner2 に対応する三栄電機の製品を利用する目的で使用者に使用する
権利を許諾（コピー及び配布は自由）する。
・三栄電機は iOS SDK for LayoutDesigner2 に関して欠陥がないこと、このマニュアルに記載されている情報の
使用に起因するいかなる損害に対しても責任を負うものではありません。
・三栄電機は iOS SDK for LayoutDesigner2 の使用に関連して生じる直接的または、間接的な損失、損害など
について、いかなる場合も一切責任を負わないものとする。
・使用者は日本国政府、または該当国の政府より必要な許可等を得ることなしに、
iOS SDK for LayoutDesigner2 の全部または一部を直接または間接的に輸出することはできません。

1. はじめに

iOS SDK for LayoutDesigner2 は、LayoutDesigner2 のレイアウトファイル（SOP ファイル）をベースに API によって相互に編集を可能にする Xamarin 対応の支援ライブラリです。
本クラスライブラリファイルは下記のライブラリファイル（DLL ファイル）から構成しています。

クラスライブラリファイル本体	LayoutDesignerSDK.dll
バーコードライブラリ	BarcodeLibrary.dll

1.1. 開発環境

対応 OS:	iOS12 以降
アプリケーション開発環境:	Visual Studio for Mac

1.2 対応製品

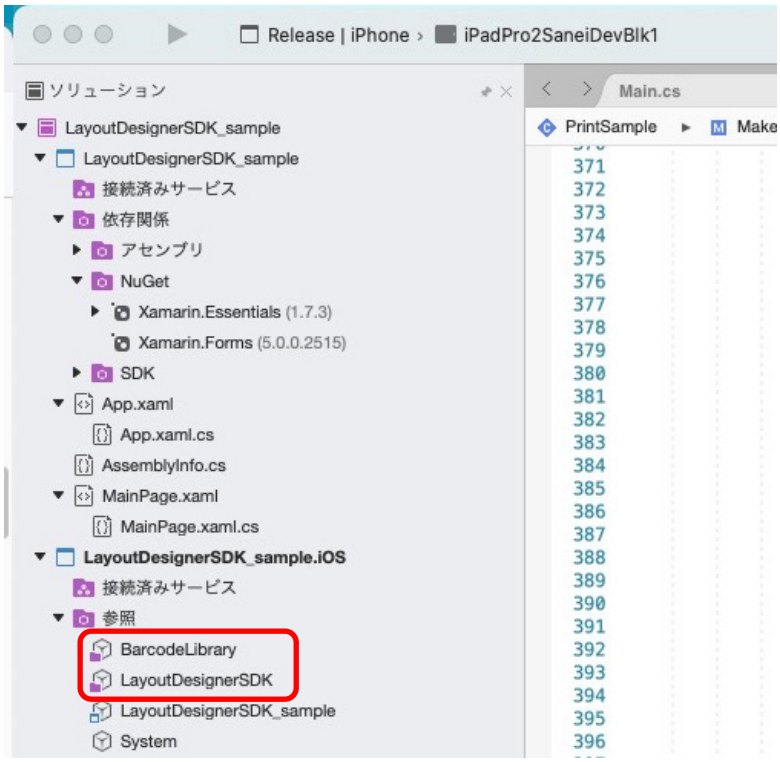
対応製品	ファームウェア	対応インターフェース
SM4-21W	V1.13 以降	無線 LAN
SM4-31W	V1.10 以降	無線 LAN

1.3. クラスライブラリの実装方法

(1) アプリケーションのプロジェクトフォルダに以下の DLL を追加します。

LayoutDesignerSDK.dll	本 SDK のライブラリ
BarcodeLibrary.dll	バーコードライブラリ

(2) ソリューションの参照に「LayoutDesignerSDK.dll」と「BarcodeLibrary.dll」を追加します。



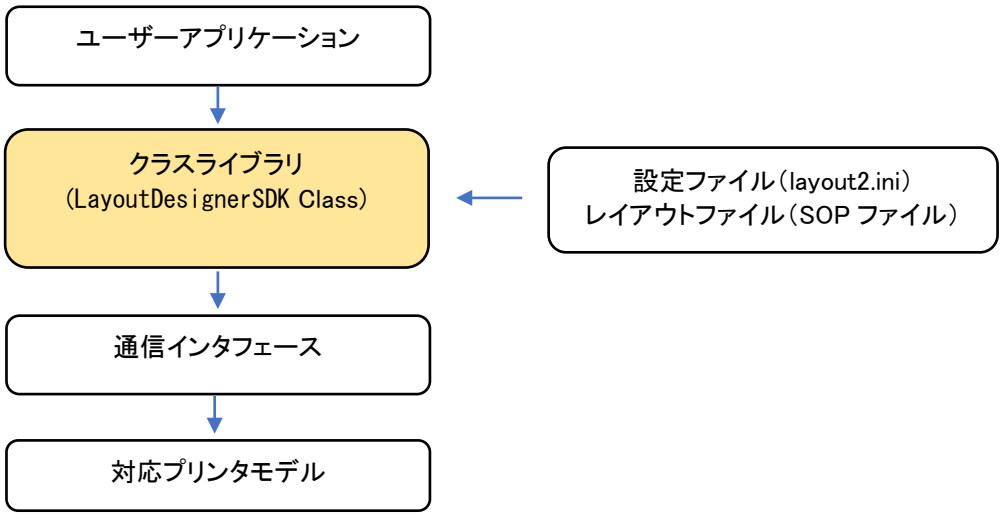
(3) ソースコードに SDK のクラスを宣言し、初期化を行います。

```
using System;
using System.IO;
using UIKit;
using Xamarin.Forms;
using System.Drawing;
using Foundation;

[assembly: Dependency(typeof(LayoutDesignerSDK_sample.iOS.PrintSample))]
namespace LayoutDesignerSDK_sample.iOS
{
    //ネイティブ側クラス（インターフェースを通してUIから呼び出せる）
    public class PrintSample : IPrintSample
    {
        private LayoutDesignerSDK.LibApi api = new LayoutDesignerSDK.LibApi();

        //SDK初期化
        public void init_SDK()
        {
            api.InitSdk();
            api.AlternativeFontNameSet(@"Hiragino Sans");
            Console.WriteLine(@"init_SDK Calling");
        }

```

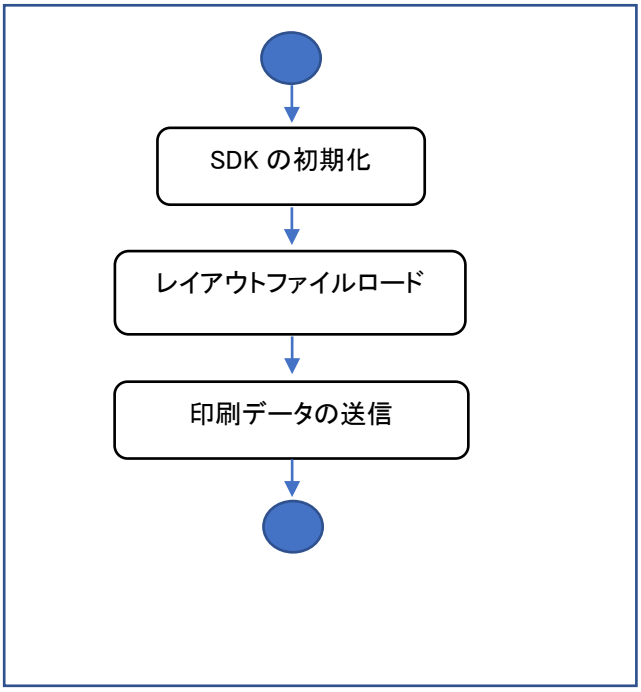
2. LayoutDesignerSDK クラス

LayoutDesignerSDK クラスを利用した全体構成及び印刷するまでのプロセスを下記に示します。

● 全体構成



● レイアウトファイルを印刷するまでのプロセス



サンプルコードのイメージ:

```
//SDK initialization.
ret = api.InitSdk();
//Initialization failure?
if (ret == false)
{
	return;
}

//File load.
int file_size = 0;
ret = api.LoadSopFile(@"sample.sop", ref file_size);
//Load failed?
if (ret == false)
{
	return;
}

//Printing.
int copies = 1;
api.WLANPrintSopFile(@"192.168.10.100", 9100, copies);
```

2.1. グローバル領域

参照可能なグローバル領域は、以下の通りです。

LibApi.ResultErrorCode　　各 API によるエラー値が格納されている。
エラー値については、4 章のエラー定数定義一覧を参照します。

2.2. 設定ファイル

設定ファイルとは、Windows 用 「LayoutDesigner2」の環境設定ファイル「layout2.ini」を示します。
本ファイルの格納先は、DLL フォルダ直下に置かれています。
お客様のアプリにおいては、LayoutDesignerSDK.dll と同じフォルダに、この設定ファイルを置きます。

環境設定ファイルの構成

image-print=1　　イメージデータの印字を行うか設定する。

0:　プリンタテキスト以外の印字を行わない。
1:　全ての印字を行う。

auto-save=0　　印字を行う毎に SOP ファイルを自動で保存するか設定する。

0: SOP ファイルを自動で保存しない。
1: SOP ファイルを自動で保存する。

シリアル番号を使用したレイアウトデータの場合、この設定を「1」にすることで
印字する度にシリアル番号の更新されたデータを自動的に保存します。

compress-print=1　圧縮印字を行うか設定する。

0:　圧縮印字を行わない。
1:　圧縮印字を行う。

2.3. スタンドアロンマクロについて

スタンドアロンマクロは、プリンタに行わせたい印刷処理をあらかじめ登録しておき、ボタン1つで呼び出す
仕組みです。日付やカウンタなど可変データを扱う事が出来るので今までホストが無ければ行えなかった印字も
行う事が出来ます。スタンドアロンマクロは、最大3つまで登録可能で使用時に選択する事が出来ます。

背景　:　　スタンドアロンマクロ登録時にメモリ中に存在するSOPファイルを描画して背景にします。
スタンドアロンマクロ自体にSOPファイルとの紐付けはないのであらかじめSOPファイルの
読み込みをする必要があります。

日付　:　　日付の印刷、及びQRコードへの日付埋め込みが出来ます。
日付はインターネットから取得するのであらかじめルーターと接続する設定にしてください。
ルーターと接続出来なかった場合、日付は印字されません。

カウンタ　:　カウンタの印字が行えます。
カウンタは、不揮発性メモリに保持されますので電源を切って中断しても次から続きを
印字する事が出来ます。

変数 ： QRコード内に埋め込む事の出来る文字列変数が8つ使えます。
変更したい文字列は、変数にしておく事により、マクロを見やすく出来ます。

QRコード ： QRコード内に日付、カウンタ、変数を埋め込む事が出来ます。
スタンドアロンマクロのQRコードは、SOPファイル中のQRコードと違い、
印字データを印刷直前に生成しますので可変データを扱う事が出来ます。

表示名 ： スタンドアロンマクロ選択時の表示名です。
この名前がプリンタのディスプレイに表示されます。

ファイル ： スタンドアロンマクロのファイル名は、「standalone_macro.dat」です。
フォルダは、SOPファイルの読み書きに使用するフォルダと同じになります。
その為、スタンドアロンマクロを扱う前にSOPファイルへの読み書きを先に
行う必要があります。

3. APIの詳細

LayoutDesignerSDK クラスの関数一覧は、以下の通りです。

カテゴリー	API	機能
バージョン取得	GetApiVersion	SDKと内部エンジンのバージョン値を取得する。
初期化	InitSdk	このクラスを初期化する。
SOPファイル	NewSopFile	SOP ファイルを新規作成する。
	LoadSopFile	SOP ファイルを読み込む。
	SaveSopFile	SOPファイルを保存する。
	WLANPrintSopFile	現在のSOPファイルのデータを無線LAN経由で印字する。
	PreviewSopFile	現在の SOP ファイルのデータのプレビュー画像を作成する。
シリアル番号	InitSerial	シリアル番号を初期化する。
	SetSerialNumber	シリアル番号を設定する。
日付／時刻	SetDateTime	日付または時刻を設定する。
オブジェクト編集	GetObjectList	オブジェクトリストを取得する
	DelObjectData	オブジェクトデータを削除する。
	GetTextData	テキストのオブジェクトデータを取得する。
	SetTextData	テキストのオブジェクトデータをセットする。
	GetPTextData	プリンタテキストのオブジェクトデータを取得する。
	SetPTextData	プリンタテキストのオブジェクトデータをセットする
	GetImageData	画像のオブジェクトデータを取得する。
	SetImageData	画像のオブジェクトデータをセットする。
	GetLineData	直線のオブジェクトデータを取得する。
	SetLineData	直線のオブジェクトデータをセットする。
	GetRectData	矩形のオブジェクトデータを取得する。
	SetRectData	矩形のオブジェクトデータをセットする。
	GetFillData	塗り潰しのオブジェクトデータを取得する。
	SetFillData	塗り潰しのオブジェクトデータをセットする。
	GetBarcode1dData	1次元バーコードのオブジェクトデータを取得する。
	SetBarcode1dData	1次元バーコードのオブジェクトデータをセットする。
	GetBarcode2dData	2次元バーコードのオブジェクトデータを取得する。
	SetBarcode2dData	2次元バーコードのオブジェクトデータをセットする。
	GetMacroData	ドキュメントマクロのオブジェクトデータを取得する。
	SetMacroData	ドキュメントマクロのオブジェクトデータをセットする。
スタンドアロンマクロ編集	InitStandaloneMacro	スタンドアロンマクロを初期化する。
	GetStandaloneMacroData	スタンドアロンマクロデータを取得する。
	SetStandaloneMacroData	スタンドアロンマクロデータをセットする。
	GetStandaloneMacroTitle	スタンドアロンマクロの表示名を取得する。
	SetStandaloneMacroTitle	スタンドアロンマクロの表示名をセットする。
	GetStandaloneMacroCount	スタンドアロンマクロのデータ数を取得する。
	LoadStandaloneMacro	スタンドアロンマクロファイルを読み込む。
	SaveStandaloneMacro	スタンドアロンマクロファイルを保存する。
	RegistStandaloneMacro	スタンドアロンマクロをプリンタに登録する。
その他	AlternativeFontNameSet	指定のフォント名が無かった場合の代替のフォント名を設定する。

備考：

各 API の引数については、4章の定数定義を参照下さい。

3.1. GetApiVersion

SDK と内部エンジンのバージョン値を取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetApiVersion(ref int sdk_ver, ref int eng_ver )
```

引数:

int	sdk_ver	SDK バージョン値
int	eng_ver	内部エンジンバージョン値

戻り値:

true	正常終了
false	エラー

備考:

バージョン値は 1.02 なら 102 と返します。

内部エンジンのバージョン値は、レイアウトデザイナー2本体のバージョンを返します。

3.2. InitSdk

このクラスを初期化する。

宣言:

```
bool    LayoutDesigner.LibApi.InitSdk()
```

引数: なし

戻り値:

true	正常終了
false	エラー

3.3. NewSopFile

SOP ファイルを新規作成する。

宣言:

```
bool    LayoutDesigner.LibApi.NewSopFile(int printer, int width, int height, int top, int left)
```

引数 :

int	printer	プリンタ機種番号
int	width	印字幅（単位はドットピッチ（1/8mm))
int	height	印字高（単位はドットピッチ（1/8mm))
int	top	上余白（単位はドットピッチ（1/8mm))
int	left	左余白（単位はドットピッチ（1/8mm))

戻り値:

true	正常終了
false	エラー

3.4. LoadSopFile

SOP ファイルを読み込む。

宣言:

```
bool    LayoutDesigner.LibApi.LoadSopFile(string file, ref int size)
```

引数 :

string	file	SOP ファイルのファイル名
int	size	SOP ファイルのファイルサイズ

戻り値:

true	正常終了
false	エラー

3.5. SaveSopFile

SOP ファイルを保存する。

宣言:

```
bool    LayoutDesigner.LibApi.SaveSopFile(string file)
```

引数:

string	file	SOP ファイルのファイル名

戻り値:

true	正常終了
false	エラー

3.6. WLANPrintSopFile

現在の SOP ファイルのデータを無線 LAN 経由で印字する。

宣言:

```
bool    LayoutDesigner.LibApi.WLANPrintSopFile(string ip, int port, int num)
```

引数:

string	ip	IPアドレス
int	port	ポート番号
int	num	印刷部数

戻り値:

true	正常終了
false	エラー

3.7. PreviewSoPFile

現在の SOP ファイルのデータのプレビュー画像を作成する。

宣言:

```
bool    LayoutDesigner.LibApi.PreviewSopFile(string file)
```

引数:

string	file	プレビューする画像のファイルの名称 画像フォーマットは「PNG」形式で作成する

戻り値:

true	正常終了
false	エラー

3.8. InitSerial

シリアル番号を初期化する。

宣言:

```
bool    LayoutDesigner.LibApi.InitSerial()
```

引数: なし

戻り値:

true	正常終了
false	エラー

3.9. SetSerialNumber

シリアル番号を設定する。

宣言:

```
bool    LayoutDesigner.LibApi.SetSerialNumber(int SelectData, int SerialNumberType, int SerialIncDec,
                int ObjectIncDec, int Counter, string SerialFormat, int CounterInit, int CounterMax)
```

引数:

int	SelectData	オブジェクトタイプの指定 0 : テキスト系 1 : バーコード系
int	SerialNumberType	(シリアル番号)種類の指定 0 : 10 進数
int	SerialIncDec	(シリアル番号)連番増減の指定 0 : 連番増加 1 : 連番減少
int	ObjectIncDec	同一シリアル番号による印字回数の指定
int	Counter	増減数の指定
string	SerialFormat	シリアル番号の書式 .NET Framework の ToString メソッドの書式で記述します
int	CounterInit	初期のカウント番号の指定
int	CounterMax	最大のカウント番号の指定

戻り値:

true	正常終了
false	エラー

備考:

ToString メソッド書式を 10 進数にする場合は、以下を参考にしてください。

書式指定子	名前	説明	例
D または d	10 進数	整数型のみサポート 精度指定子は最小桁数 最小桁数に満たない場合は左側に 0 が挿入される	数値が 1234 だった場合 書式:“D” 印字結果:“1234” 書式:”D6” 印字結果:”001234” 数値が-1234 だった場合 書式:”D5” 印字結果:”-01234”

3.10. SetDateTime

日付または時刻を設定する。

宣言:

bool LayoutDesigner.LibApi.SetDateTime(int DateTimeSelect, int FormatType, int Offset1, int Offset2)

引数 :

int	DateTimeSelect	日付または時刻の選択	
		0 : 日付	
		1 : 時刻	
int	FormatType	書式の指定	
		日付の書式番号: 0～24	
		時刻の書式番号: 0～19	
int	Offset1	オフセット種類の指定	
		日付書式	0 :「日」
			1 :「月」
			2 :「年」
		時刻書式	時のオフセット
int	Offset2	オフセット範囲の指定	
		日付書式	日のオフセット: 0～30
			月のオフセット: 0～11
			年のオフセット: 0～98
		時刻書式	分のオフセット

戻り値:

true	正常終了
false	エラー

備考:

オフセットなしにしたい時は、オフセットを0にしてください。

時刻の場合、負数のオフセットが指定できます。

3.11. GetObjectList

オブジェクトリストを取得する。

宣言:

bool LayoutDesigner.LibApi.GetObjectList(ref string[] object_list, ref int object_count)

引数 :

string[]	object_list	オブジェクトリスト
int	object_count;	オブジェクト数

戻り値:

true	正常終了
false	エラー

3.12. DelObjectData

オブジェクトデータを削除する。

宣言:

```
bool    LayoutDesigner.LibApi.DelObjectData(string object)
```

引数 :

string	object	オブジェクトの名称

戻り値:

true	正常終了
false	エラー

3.13. GetTextData

テキストのオブジェクトデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetTextData(string object, ref Rectangle rect, ref string TextData,
                ref int TextSource, ref string FontName, ref int FontPoint, ref int FontBold,
                ref int FontItalic, ref int FontUnderline, ref int FontStrikeout, ref int FontCharset,
                ref int TextAlign, ref bool AutoSize, ref int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の取得
string	TextData	テキストデータの取得
int	TextSource	データソースの取得
string	FontName	フォント名の取得
int	FontPoint	（フォント） ポイントサイズの取得
int	FontBold	（フォント） 強調修飾の取得
int	FontItalic	（フォント） イタリック修飾の取得
int	FontUnderline	（フォント） アンダーライン修飾の取得
int	FontStrikeout	（フォント） 取り消し線修飾の取得
int	FontCharset	（フォント） キャラクタセットの取得
int	TextAlign	テキストアライメントの取得
bool	AutoSize	テキストの自動サイズ調整の取得
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.14. SetTextData

テキストのオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetTextData(ref string object, Rectangle rect, string TextData,
                int TextSource, string FontName, int FontPoint, int FontBold, int FontItalic,
                int FontUnderline, int FontStrikeout, int FontCharset, int TextAlign, bool AutoSize,
                int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の指定
string	TextData	テキストデータの指定
int	TextSource	データソースの指定 0 : テキストデータ 1 : 日付／時刻 2 : シリアル番号
string	FontName	フォント名の指定
int	FontPoint	（フォント） ポイントサイズの指定
int	FontBold	（フォント） 強調修飾の指定 0 : 無効　1 : 有効
int	FontItalic	（フォント） イタリック修飾の指定 0 : 無効　1 : 有効
int	FontUnderline	（フォント） アンダーライン修飾の指定 0 : 無効　1 : 有効
int	FontStrikeout	（フォント） 取り消し線修飾の指定 0 : 無効　1 : 有効
int	FontCharset	（フォント） キャラクタセットの指定 0 :　欧文 128 : 日本語
int	TextAlign	テキストアライメントの指定 0 : 左詰 1 : センタリング 2 : 右詰
bool	AutoSize	テキストの自動サイズ調整の指定 true :　有効 false :　無効
int	angle	回転角度の指定 0 : 回転角度 0 度 1 : 回転角度 90 度 2 : 回転角度 180 度 3 : 回転角度 270 度

戻り値:

true	正常終了
false	エラー

備考:

TextSource が日付／時刻の時は、あらかじめ SetDateTime でフォーマットを決めてください。
TextSource がシリアル番号の時は、あらかじめ SetSerial でフォーマットを決めてください。

3.15. GetPTextData

プリンタテキストのオブジェクトデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetPTextData(string object, ref Rectangle rect, ref string TextData,
                ref int TextSource, ref int PrinterFont, ref int FontBold, ref int FontItalic,
                ref int FontUnderline, ref int WidthSize, ref int HeightSize, ref int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の取得
string	TextData	テキストデータの取得
int	TextSource	データソースの取得
int	PrinterFont	プリンタテキストのフォントタイプの取得
int	FontBold	（プリンタテキスト） 強調修飾の取得
int	FontItalic	（プリンタテキスト） イタリック修飾の取得
int	FontUnderline	（プリンタテキスト） アンダーライン修飾の取得
int	WidthSize	（プリンタテキスト） 横の倍角サイズの取得
int	HeightSize	（プリンタテキスト） 縦の倍角サイズの取得
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.16. SetPTextData

プリンタテキストのオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetPTextData(ref string object, Rectangle rect, string TextData,
                int TextSource, int PrinterFont, int FontBold, int FontItalic, int FontUnderline,
                int WidthSize, int HeightSize, int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の指定
string	TextData	テキストデータの指定
int	TextSource	データソースの指定 0：テキストデータ 1：日付／時刻 2：シリアル番号
int	PrinterFont	プリンタテキストのフォントタイプの指定 0：フォント A（24 ドット） 1：フォント B（16 ドット）
int	FontBold	（プリンタテキスト） 強調修飾の指定 0：無効　1：有効
int	FontItalic	（プリンタテキスト） イタリック修飾の指定 0：無効　1：有効
int	FontUnderline	（プリンタテキスト） アンダーライン修飾の指定 0：無効　1：有効
int	WidthSize	（プリンタテキスト） 横の倍角サイズの指定
int	HeightSize	（プリンタテキスト） 縦の倍角サイズの指定 1～8：1～8 倍
int	angle	回転角度の指定 0：回転角度 0 度 1：回転角度 90 度 2：回転角度 180 度 3：回転角度 270 度

戻り値:

true	正常終了
false	エラー

備考:

TextSource が日付／時刻の時は、あらかじめ SetDateTime でフォーマットを決めてください。
TextSource がシリアル番号の時は、あらかじめ SetSerial でフォーマットを決めてください。

3.17. GetImageData

画像のオブジェクトデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetImageData(string object, ref Rectangle rect, ref UIImage bmp,
                ref bool aspect, ref int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の取得
UIImage	bmp	イメージデータの取得
bool	aspect	アスペクト比の設定（true 又は false）の取得
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.18. SetImageData

画像のオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetImageData(ref string object, Rectangle rect, string name, int type,
                int threshold, bool aspect, int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の指定
string	name	画像ファイルの名称
int	type	2値化の変換種類の指定 0：単純 2 値化 1：ディザ変換 2：誤差拡散
int	threshold	2値化閾値（0～256）の指定
bool	aspect	アスペクト比の設定の指定 true：アスペクト比を維持する false：アスペクト比を維持しない
int	angle	回転角度の指定 0：回転角度 0 度 1：回転角度 90 度 2：回転角度 180 度 3：回転角度 270 度

戻り値:

true	正常終了
false	エラー

3.19. GetLineData

直線のオブジェクトデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetLineData(string object, ref Point st, ref Point ed, ref int width,
                                          ref int type, ref int angle)
```

引数:

string	object	オブジェクトの名称
Point	st	開始座標データの取得
Point	ed	終了座標データの取得
int	width	線幅の取得
int	type	線種の取得
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.20. SetLineData

直線のオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetLineData(ref string object, Point st, Point ed, int width, int type, int angle)
```

引数:

string	object	オブジェクトの名称
Point	st	開始座標データの指定
Point	ed	終了座標データの指定
int	width	線幅（1～10ドット）の指定
int	type	線種の指定 0：実線 1：破線 2：点線 3：一点鎖線 4：二点鎖線
int	angle	回転角度 0：回転角度 0 度 1：回転角度 90 度 2：回転角度 180 度 3：回転角度 270 度

戻り値:

true	正常終了
false	エラー

3.21. GetRectData

矩形のオブジェクトデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetRectData(string object, ref Rectangle rect, ref int width,
                                          ref int type, ref int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の取得
int	width	線幅の取得
int	type	線種の取得
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.22. SetRectData

矩形のオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetRectData(ref string object, Rectangle rect, int width, int type, int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の指定
int	width	線幅（1～10ドット）の指定
int	type	線種の指定 0：実線 1：破線 2：点線 3：一点鎖線 4：二点鎖線
int	angle	回転角度の指定 0：回転角度 0 度 1：回転角度 90 度 2：回転角度 180 度 3：回転角度 270 度

戻り値:

true	正常終了
false	エラー

3.23. GetFillData

塗り潰しのオブジェクトデータを取得する。

宣言:

bool LayoutDesigner.LibApi.GetFillData(string object, ref Rectangle rect, ref int color, ref int angle)

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の取得
int	color	塗り潰し色の取得
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.24. SetFillData

塗り潰しのオブジェクトデータをセットする。

宣言:

bool LayoutDesigner.LibApi.SetFillData(ref string object, Rectangle rect, int color, int angle)

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の指定
int	color	塗り潰し色の指定 0 : 塗り潰し色　黒 1 : 塗り潰し色　白 2 : 塗り潰し色　反転
int	angle	回転角度の指定 0 : 回転角度 0 度 1 : 回転角度 90 度 2 : 回転角度 180 度 3 : 回転角度 270 度

戻り値:

true	正常終了
false	エラー

3.25. GetBarcode1dData

1 次元バーコードのオブジェクトデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetBarcode1dData(string object, ref Rectangle rect, ref string TextData,
                ref int BarcodeSource, ref int BarcodeType, ref int HriPos, ref int BarcodeWidth,
                ref int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の取得
string	TextData	テキストデータの取得
int	BarcodeSource	1 次元バーコードのデータソースの取得
int	BarcodeType	1 次元バーコード種類の取得
int	HriPos	HRI 文字の印字位置の取得
int	BarcodeWidth	バー幅の取得
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.26. SetBarcode1dData

1 次元バーコードのオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetBarcode1dData(ref string object, Rectangle rect, string TextData,
                int BarcodeSource, int BarcodeType, int HriPos, int BarcodeWidth, int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の指定
string	TextData	テキストデータの指定
int	BarcodeSource	1 次元バーコードのデータソースの指定 0 : テキストデータ 1 : シリアル番号
int	BarcodeType	1 次元バーコード種類の指定 1 : UPC-A 2 : UPC-E 3 : JAN13 4 : JAN8 5 : CODE39 6 : ITF 7 : CODABAR 8 : CODE128 9 : CODE93
int	HriPos	HRI 文字の印字位置の指定 0 : HRI 文字を使用しない 1 : HRI 文字をバーコード上側に配置 2 : HRI 文字をバーコード下側に配置 3 : HRI 文字をバーコードの上下に配置
int	BarcodeWidth	バー幅の指定 0～3 : 1～4ドット
int	angle	回転角度の指定 0 : 回転角度 0 度 1 : 回転角度 90 度 2 : 回転角度 180 度 3 : 回転角度 270 度

戻り値:

true	正常終了
false	エラー

備考:

TextSource がシリアル番号の時は、あらかじめ SetSerial でフォーマットを決めてください。

3.27. GetBarcode2dData

2 次元バーコードのオブジェクトデータを取得する。

宣言:

bool LayoutDesigner.LibApi.GetBarcode2dData(string object, ref Rectangle rect, ref string TextData, ref int BarcodeSource, ref int BarcodeType, ref int CellSize, ref int EccLevel, ref int SymbolSize, ref int Colum, ref int angle)

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の取得
string	TextData	テキストデータの取得
int	BarcodeSource	2 次元バーコードのデータソースの取得
int	BarcodeType	2 次元バーコード種類の取得
int	CellSize	QRコードのセルサイズの取得
int	EccLevel	ECC レベルの取得
int	SymbolSize	シンボルサイズの取得
int	Colum	列数の取得（Micro PDF417 選択以外は読み捨てる）
int	angle	回転角度の取得

戻り値:

true	正常終了
false	エラー

3.28. SetBarcode2dData

2 次元バーコードのオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetBarcode2dData(ref string object, Rectangle rect, string TextData,
                int BarcodeSource, int BarcodeType, int CellSize, int EccLevel, int SymbolSize,
                int Colum, int angle)
```

引数:

string	object	オブジェクトの名称
Rectangle	rect	オブジェクト座標の指定
string	TextData	テキストデータの指定
int	BarcodeSource	2 次元バーコードのデータソースの指定 0 : テキスト文字列 1 : シリアル番号
int	BarcodeType	2 次元バーコード種類の指定 1 : QR CODE 2 : Micro QR CODE 3 : PDF417 4 : Micro PDF417 5 : DATA MATRIX 6 : MAXI CODE
int	CellSize	QRコードのセルサイズの指定 1～8 : 1 セル当たりのドット数
int	EccLevel	ECC レベルの指定 1 : ECC レベル 7%　（QR CODE /Micro QR CODE） 2 : ECC レベル 15%　（QR CODE /Micro QR CODE） 3 : ECC レベル 25%　（QR CODE /Micro QR CODE） 4 : ECC レベル 30%　（QR CODE）
int	SymbolSize	シンボルサイズの指定 QR CODE 選択時 0 : オートサイズ　1～40 : 指定サイズ Micro QR CODE 選択時 0 : オートサイズ　1～4 : 指定サイズ DATA MATRIX 選択時 10、18、22、26、32、40、48　の指定サイズ
int	Colum	列数の指定（Micro PDF417 選択時のみ値を有効） 0～3 : 列数 1～4 を指定
int	angle	回転角度 0 : 回転角度 0 度 1 : 回転角度 90 度 2 : 回転角度 180 度 3 : 回転角度 270 度

戻り値:

true	正常終了
false	エラー

備考:

TextSource がシリアル番号の時は、あらかじめ SetSerial でフォーマットを決めてください。

3.29. GetMacroData

ドキュメントマクロのオブジェクトデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetMacroData(string object, ref string[] start_cmd, ref string[] end_cmd)
```

引数:

string	object	オブジェクト名
string[]	start_cmd	ページ開始コマンドの取得
string[]	end_cmd	ページ終了コマンドの取得

戻り値:

true	正常終了
false	エラー

3.30. SetMacroData

ドキュメントマクロのオブジェクトデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetMacroData(ref string object, string[] start_cmd, string[] end_cmd)
```

引数:

string	object	オブジェクト名
string[]	start_cmd	ページ開始コマンドの指定
string[]	end_cmd	ページ終了コマンドの指定

戻り値:

true	正常終了
false	エラー

3.31. AlternativeFontNameSet

Text オブジェクトで指定したフォント名が存在しない場合、本 API で指定したフォント名が代わりに使われる。

宣言:

```
bool    LayoutDesigner.LibApi.AlternativeFontNameSet(string FontName)
```

引数:

string	FontName	代替フォント名

戻り値:

true	正常終了
false	エラー

3.32. InitStandaloneMacro

スタンドアロンマクロを初期化する。

宣言:

```
void    InitStandaloneMacro ()
```

引数:　　なし

戻り値:　　なし

3.33. GetStandaloneMacroData

スタンドアロンマクロのコマンドデータを取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetStandaloneMacroData(int macro_select, ref string[] stand_cmd)
```

引数:

int	macro_select	マクロ選択番号
string[]	stand_cmd	スタンドアロンマクロコマンド

戻り値:

true	正常終了
false	エラー

3.34. SetStandaloneMacroData

スタンドアロンマクロのコマンドデータをセットする。

宣言:

```
bool    LayoutDesigner.LibApi.SetStandaloneMacroData(int macro_select, string[] stand_cmd)
```

引数:

int	macro_select	マクロ選択番号
string[]	stand_cmd	スタンドアロンマクロコマンド

戻り値:

true	正常終了
false	エラー

3.35. GetStandaloneMacroTitle

スタンドアロンマクロの表示名を取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetStandaloneMacroTitle(int macro_select, ref string title)
```

引数:

int	macro_select	マクロ選択番号
string	title	マクロ表示名

戻り値:

true	正常終了
false	エラー

3.36. SetStandaloneMacroTitle

スタンドアロンマクロの表示名をセットする。

宣言:

```
bool    LayoutDesigner.LibApi.GetStandaloneMacroTitle(int macro_select, ref string title)
```

引数:

int	macro_select	マクロ選択番号
string	title	マクロ表示名

戻り値:

true	正常終了
false	エラー

3.37. GetStandaloneMacroCount

スタンドアロンマクロのデータ数を取得する。

宣言:

```
bool    LayoutDesigner.LibApi.GetStandaloneMacroCount (int macro_select, ref int count)
```

引数:

int	macro_select	マクロ選択番号
string	count	マクロデータ数

戻り値:

true	正常終了
false	エラー

3.38. LoadStandaloneMacro

スタンドアロンマクロファイルを読み込む。

宣言:

```
bool    LayoutDesigner.LibApi.LoadStandaloneMacro (string file, ref int size)
```

引数:

string	file	スタンドアロンマクロファイル名
int	size	ファイルサイズ

戻り値:

true	正常終了
false	エラー

3.39. SaveStandaloneMacro

スタンドアロンマクロファイルを保存する。

宣言:

```
bool    LayoutDesigner.LibApi.SaveStandaloneMacro (string file)
```

引数:

string	file	スタンドアロンマクロファイル名

戻り値:

true	正常終了
false	エラー

3.40. RegistStandaloneMacro

スタンドアロンマクロをプリンタに登録する。

宣言:

```
bool    LayoutDesigner.LibApi.RegistStandaloneMacro (string ip, int port, int macro_select)
```

引数:

string	ip	IPアドレス
int	port	ポート番号
int	macro_select	マクロ選択番号

戻り値:

true	正常終了
false	エラー

4. 定数定義

LibApi クラスのエラー定数定義の一覧は、以下の通りです。

名称	値	項目
エラー種類		
ERROR_CODE_OK	0	正常終了
ERROR_CODE_VER_GET	1	バージョンの取得に失敗
ERROR_CODE_SDK_INIT	2	SDK の初期化に失敗
ERROR_CODE_FILE_NEW	3	SOP ファイルの新規作成に失敗
ERROR_CODE_PAPER_SETTING	4	用紙の設定に失敗
ERROR_CODE_FILE_READ	5	SOP ファイルの読み込みに失敗
ERROR_CODE_FILE_WRITE	6	SOP ファイルの書き込みに失敗
ERROR_CODE_FILE_PRINT	7	SOP ファイルの印刷に失敗
ERROR_CODE_FILE_PREVIEW	8	SOP ファイルのプレビュー作成に失敗
ERROR_CODE_OBJECT_LIST_GET	9	オブジェクトリストの取得に失敗
ERROR_CODE_OBJECT_DATA_GET	10	オブジェクトデータの取得に失敗
ERROR_CODE_OBJECT_DATA_SET	11	オブジェクトデータの設定に失敗
ERROR_CODE_OBJECT_DATA_DEL	12	オブジェクトデータの削除に失敗
ERROR_CODE_SERIAL_INIT	13	シリアル番号の初期化に失敗
ERROR_CODE_SERIAL_SET	14	シリアル番号の設定に失敗
ERROR_CODE_DATE_TIME_SET	15	日付／時間データの設定に失敗
ERROR_CODE_TEXT_OBJECT_GET	16	テキストデータの取得に失敗
ERROR_CODE_TEXT_OBJECT_SET	17	テキストデータの設定に失敗
ERROR_CODE_PTEXT_OBJECT_GET	18	プリンタテキストデータの取得に失敗
ERROR_CODE_PTEXT_OBJECT_SET	19	プリンタテキストデータの設定に失敗
ERROR_CODE_IMAGE_OBJECT_GET	20	画像データの取得に失敗
ERROR_CODE_IMAGE_OBJECT_SET	21	画像データの設定に失敗
ERROR_CODE_LINE_OBJECT_GET	22	直線データの取得に失敗
ERROR_CODE_LINE_OBJECT_SET	23	直線データの設定に失敗
ERROR_CODE_RECT_OBJECT_GET	24	矩形データの取得に失敗
ERROR_CODE_RECT_OBJECT_SET	25	矩形データの設定に失敗
ERROR_CODE_FILL_OBJECT_GET	26	塗り潰しデータの取得に失敗
ERROR_CODE_FILL_OBJECT_SET	27	塗り潰しデータの設定に失敗
ERROR_CODE_BARCODE_1D_OBJECT_GET	28	1 次元バーコードデータの取得に失敗
ERROR_CODE_BARCODE_1D_OBJECT_SET	29	1 次元バーコードデータの設定に失敗
ERROR_CODE_BARCODE_2D_OBJECT_GET	30	2 次元バーコードデータの取得に失敗
ERROR_CODE_BARCODE_2D_OBJECT_SET	31	2 次元バーコードデータの設定に失敗
ERROR_CODE_MACRO_OBJECT_GET	32	ドキュメントマクロデータの取得に失敗
ERROR_CODE_MACRO_OBJECT_SET	33	ドキュメントマクロデータの設定に失敗
ERROR_CODE_STANDALONE_MACRO_GET	34	スタンドアロンマクロデータの取得に失敗
ERROR_CODE_STANDALONE_MACRO_SET	35	スタンドアロンマクロデータの設定に失敗
ERROR_CODE_STANDALONE_MACRO_LOAD	36	スタンドアロンマクロデータの読み込みに失敗
ERROR_CODE_STANDALONE_MACRO_SAVE	37	スタンドアロンマクロデータの保存に失敗
ERROR_CODE_STANDALONE_MACRO_REGIST	38	スタンドアロンマクロデータの登録に失敗

LibApi クラスの API で指定できる定数定義の一覧は、以下の通りです。

名称	値	項目
プリンタ機種番号		
PRINTER_SM4_21	13	SM4-21
PRINTER_SM4_31	14	SM4-31
オブジェクトタイプ		
OBJECT_TYPE_NONE	0	無効オブジェクト
OBJECT_TYPE_TEXT	1	テキストオブジェクト
OBJECT_TYPE_PTEXT	2	プリンタテキストオブジェクト
OBJECT_TYPE_RECT	3	矩形オブジェクト
OBJECT_TYPE_FILL	4	塗り潰しオブジェクト
OBJECT_TYPE_LINE	5	直線オブジェクト
OBJECT_TYPE_IMAGE	6	画像オブジェクト
OBJECT_TYPE_BAR1	7	1次元バーコードオブジェクト
OBJECT_TYPE_BAR2	8	2次元バーコードオブジェクト
OBJECT_TYPE_MACRO	9	ドキュメントマクロオブジェクト
回転角度		
OBJECT_ANGLE0	0	回転角度 0 度
OBJECT_ANGLE90	1	回転角度 90 度
OBJECT_ANGLE180	2	回転角度 180 度
OBJECT_ANGLE270	3	回転角度 270 度
テキスト又はプリンタテキストのデータソース		
TEXT_SOURCE_TEXT	0	テキストデータ
TEXT_SOURCE_DATE	1	日付／時刻
TEXT_SOURCE_SERIAL	2	シリアル番号
テキストフォント又はプリンタテキストの強調修飾		
TEXT_BOLD_ENABLE	0	強調修飾を有効にする
TEXT_BOLD_DISABLE	1	強調修飾を無効にする
テキストフォント又はプリンタテキストのイタリック修飾		
TEXT_ITALIC_ENABLE	0	イタリック修飾を有効にする
TEXT_ITALIC_DISABLE	1	イタリック修飾を無効にする
テキストフォント又はプリンタテキストのアンダーライン修飾		
TEXT_UNDERLINE_ENABLE	0	アンダーライン修飾を有効にする
TEXT_UNDERLINE_DISABLE	1	アンダーライン修飾を無効にする
テキストフォントの取り消し線修飾		
TEXT_STRIKE_ENABLE	0	取り消し線修飾を有効にする
TEXT_STRIKE_DISABLE	1	取り消し線修飾を無効にする
テキストフォントのキャラクタセット		
CHARACTER_SET_JAPAN	128	キャラクタセット日本語
CHARACTER_SET_EURO	0	キャラクタセット欧文

テキストフォントのアライメント		
TEXT_ALIGN_LEFT	0	左詰
TEXT_ALIGN_CENTER	1	センタリング
TEXT_ALIGN_RIGHT	2	右詰
テキストフォントの自動サイズ調整		
TEXT_AUTO_SIZE_DISABLE	false	自動サイズ調整を無効にする
TEXT_AUTO_SIZE_ENABLE	true	自動サイズ調整を有効にする
プリンタテキストのフォントタイプ		
PTEXT_FONTA	0	フォントA（24 ドット）
PTEXT_FONTB	1	フォントB（16 ドット）
プリンタテキストの倍角サイズ		
PTEXT_ZOOM_SIZE1	1	1 倍
PTEXT_ZOOM_SIZE2	2	2 倍
PTEXT_ZOOM_SIZE3	3	3 倍
PTEXT_ZOOM_SIZE4	4	4 倍
PTEXT_ZOOM_SIZE5	5	5 倍
PTEXT_ZOOM_SIZE6	6	6 倍
PTEXT_ZOOM_SIZE7	7	7 倍
PTEXT_ZOOM_SIZE8	8	8 倍
2 値化変換種類		
IMAGE_CONVERT_SIMPLE	0	単純 2 値化
IMAGE_CONVERT_DITHER	1	ディザ変換
IMAGE_CONVERT_ERROR_DIFFUSION	2	誤差拡散
2 値化閾値		
IMAGE_THRESHOLD_MIN	0	閾値最小値
IMAGE_THRESHOLD_MAX	256	閾値最大値
アスペクト比		
ASPECT_RATIO_NON_KEEP	false	アスペクト比を維持しない
ASPECT_RATIO_KEEP	true	アスペクト比を維持する
線種		
LINE_TYPE_SOLID	0	線種 実線
LINE_TYPE_DASH	1	線種 破線
LINE_TYPE_DOT	2	線種 点線
LINE_TYPE_1DOT_CHAIN	3	線種 １点鎖線
LINE_TYPE_2DOT_CHAIN	4	線種 ２点鎖線
線幅		
LINE_WIDTH_1DOT	1	線幅 1ドット
LINE_WIDTH_2DOT	2	線幅 2ドット
LINE_WIDTH_3DOT	3	線幅 3ドット
LINE_WIDTH_4DOT	4	線幅 4ドット
LINE_WIDTH_5DOT	5	線幅 5ドット
LINE_WIDTH_6DOT	6	線幅 6ドット
LINE_WIDTH_7DOT	7	線幅 7ドット
LINE_WIDTH_8DOT	8	線幅 8ドット
LINE_WIDTH_9DOT	9	線幅 9ドット
LINE_WIDTH_10DOT	10	線幅 10 ドット
塗り潰し色		
FILL_COLOR_BLACK	0	塗り潰し色 黒
FILL_COLOR_WHITE	1	塗り潰し色 白
FILL_COLOR_REVERSE	2	塗り潰し色 反転

1 次元バーコードの種類		
BARCODE1D_TYPE_UPCA	1	UPC-A
BARCODE1D_TYPE_UPCE	2	UPC-E
BARCODE1D_TYPE_JAN13	3	JAN13
BARCODE1D_TYPE_JAN8	4	JAN8
BARCODE1D_TYPE_CODE39	5	CODE39
BARCODE1D_TYPE_ITF	6	ITF
BARCODE1D_TYPE_CODABAR	7	CODABAR
BARCODE1D_TYPE_CODE128	8	CODE128
BARCODE1D_TYPE_CODE93	9	CODE93
2 次元バーコードの種類		
BARCODE2D_TYPE_QRCODE	1	QR CODE
BARCODE2D_TYPE_MICRO_QRCODE	2	Micro QR CODE
BARCODE2D_TYPE_PDF417	3	PDF417
BARCODE2D_TYPE_MICRO_PDF417	4	Micro PDF417
BARCODE2D_TYPE_DATA_MATRIX	5	DATA MATRIX
BARCODE2D_TYPE_MAXI_CODE	6	MAXI CODE
1 次元バーコード及び 2 次元バーコードのデータソース		
BARCODE_SOURCE_TEXT	0	テキストデータ
BARCODE_SOURCE_SERIAL	1	シリアル番号
HRI 文字の印字位置		
HRI_LOCATE_NONE	0	HRI 文字を使用しない
HRI_LOCATE_UPPER	1	HRI 文字をバーコード上側に配置
HRI_LOCATE_BOTTOM	2	HRI 文字をバーコード下側に配置
HRI_LOCATE_BOTH	3	HRI 文字をバーコードの上下に配置
バー幅		
BAR_WIDTH1	0	バー幅1ドット
BAR_WIDTH2	1	バー幅2ドット
BAR_WIDTH3	2	バー幅3ドット
BAR_WIDTH4	3	バー幅4ドット
QR コードのセルサイズ		
QRCODE_CELL_SIZE1	1	QR CODE のセルサイズ 1
QRCODE_CELL_SIZE2	2	QR CODE のセルサイズ 2
QRCODE_CELL_SIZE3	3	QR CODE のセルサイズ 3
QRCODE_CELL_SIZE4	4	QR CODE のセルサイズ 4
QRCODE_CELL_SIZE5	5	QR CODE のセルサイズ 5
QRCODE_CELL_SIZE6	6	QR CODE のセルサイズ 6
QRCODE_CELL_SIZE7	7	QR CODE のセルサイズ 7
QRCODE_CELL_SIZE8	8	QR CODE のセルサイズ 8

シンボルサイズ		
QRCODE_SYMBOL_MIN_SIZE	0	QR CODE オートサイズ
QRCODE_SYMBOL_MAX_SIZE	40	QR CODE 最大サイズ
MICRO_QRCODE_SYMBOL_MIN_SIZE	0	MICRO QR CODE オートサイズ
MICRO_QRCODE_SYMBOL_MAX_SIZE	4	MICRO QR CODE 最大サイズ
DATAMATRIX_SYMBOL_SIZE10	10	DATA MATRIX サイズ 10
DATAMATRIX_SYMBOL_SIZE18	18	DATA MATRIX サイズ 18
DATAMATRIX_SYMBOL_SIZE22	22	DATA MATRIX サイズ 22
DATAMATRIX_SYMBOL_SIZE26	26	DATA MATRIX サイズ 26
DATAMATRIX_SYMBOL_SIZE32	32	DATA MATRIX サイズ 32
DATAMATRIX_SYMBOL_SIZE40	40	DATA MATRIX サイズ 40
DATAMATRIX_SYMBOL_SIZE48	48	DATA MATRIX サイズ 48
ECC レベル		
QRCODE_ECC_L	1	QR CODE ECC レベル 7%
QRCODE_ECC_M	2	QR CODE ECC レベル 15%
QRCODE_ECC_Q	3	QR CODE ECC レベル 25%
QRCODE_ECC_H	4	QR CODE ECC レベル 30%
列数		
MICRO_PDF417_COLUMNS_SIZE1	0	Micro PDF417 の列数 1
MICRO_PDF417_COLUMNS_SIZE2	1	Micro PDF417 の列数 2
MICRO_PDF417_COLUMNS_SIZE3	2	Micro PDF417 の列数 3
MICRO_PDF417_COLUMNS_SIZE4	3	Micro PDF417 の列数 4
シリアル番号のオブジェクトタイプ		
OBJECT_TYPE_TEX	0	テキスト
OBJECT_TYPE_BARCODE	1	バーコード
シリアル番号の種類		
SERIAL_TYPE_DECIMAL	0	10 進数
シリアル番号の連番増減		
SERIAL_NUMBER_INC	0	連番増加
SERIAL_NUMBER_DEC	1	連番減少
日付又は時刻の選択		
SELECT_TYPE_DATE	0	日付選択
SELECT_TYPE_TIME	1	時刻選択
日付のオフセット種類		
OFFSET_TYPE_DAY	0	日
OFFSET_TYPE_MONTH	1	月
OFFSET_TYPE_YEAR	2	年
日付のオフセット範囲		
OFFSET_DAY_MIN	0	日のオフセット最小値
OFFSET_DAY_MAX	30	日のオフセット最大値
OFFSET_MONTH_MIN	0	月のオフセット最小値
OFFSET_MONTH_MAX	11	月のオフセット最大値
OFFSET_YEAR_MIN	0	年のオフセット最小値
OFFSET_YEAR_MAX	98	年のオフセット最大値

日付の書式		
DATE_FORMAT_TYPE01	0	2021/04/13(Year/Month/Day)
DATE_FORMAT_TYPE02	1	21/04/13(Year/Month/Day)
DATE_FORMAT_TYPE03	2	2021-04-13(Year-Month-Day)
DATE_FORMAT_TYPE04	3	21-04-13(Year-Month-Day)
DATE_FORMAT_TYPE05	4	2021,04,13(Year,Month,Day)
DATE_FORMAT_TYPE06	5	21,04,13(Year,Month,Day)
DATE_FORMAT_TYPE07	6	2021.04.13(Year.Month.Day)
DATE_FORMAT_TYPE08	7	21.04.13(Year,Month.Day)
DATE_FORMAT_TYPE09	8	2021 年 04 月 13 日
DATE_FORMAT_TYPE10	9	21 年 04 月 13 日
DATE_FORMAT_TYPE11	10	2021 年 04 月 13 日 火曜日
DATE_FORMAT_TYPE12	11	21 年 04 月 13 日 火曜日
DATE_FORMAT_TYPE13	12	令和 03 年 04 月 13 日
DATE_FORMAT_TYPE14	13	R03 年 04 月 13 日
DATE_FORMAT_TYPE15	14	R03/04/13(Year/Month/Day)
DATE_FORMAT_TYPE16	15	R03-04-13(Year-Month-Day)
DATE_FORMAT_TYPE17	16	R03,04,13(Year,Month,Day)
DATE_FORMAT_TYPE18	17	R03.04.13(Year.Month.Day)
DATE_FORMAT_TYPE19	18	210413(YearMonthDay)
DATE_FORMAT_TYPE20	19	20210413(YearMonthDay)
DATE_FORMAT_TYPE21	20	火(DayOfWeek)
DATE_FORMAT_TYPE22	21	21(Year)
DATE_FORMAT_TYPE23	22	2021(Year)
DATE_FORMAT_TYPE24	23	04(Month)
DATE_FORMAT_TYPE25	24	13(Day)
時刻の書式		
TIME_FORMAT_TYPE01	0	02:22 PM(12 Hour Format)
TIME _FORMAT_TYPE02	1	02:22(12 Hour Format)
TIME _FORMAT_TYPE03	2	02:22:19(12 Hour Format)
TIME _FORMAT_TYPE04	3	02:22:19(12 Hour Format:2Digits)
TIME _FORMAT_TYPE05	4	02:22:19 PM(12 Hour Format)
TIME _FORMAT_TYPE06	5	14:22 PM(24 Hour Format)
TIME _FORMAT_TYPE07	6	14:22(24 Hour Format)
TIME _FORMAT_TYPE08	7	14:22:19(24 Hour Format)
TIME _FORMAT_TYPE09	8	14:22:19(24 Hour Format:2Digits)
TIME _FORMAT_TYPE10	9	02 時 22 分(12 Hour Format)
TIME _FORMAT_TYPE11	10	14 時 22 分(24 Hour Format)
TIME _FORMAT_TYPE12	11	02 時 22 分 19 秒(12 Hour Format)
TIME _FORMAT_TYPE13	12	14 時 22 分 19 秒(24 Hour Format)
TIME _FORMAT_TYPE14	13	午後 02 時 22 分(12 Hour Format)
TIME _FORMAT_TYPE15	14	午後 14 時 22 分(24 Hour Format)
TIME _FORMAT_TYPE16	15	PM
TIME _FORMAT_TYPE17	16	02 時 00 分(12 Hour Format)
TIME _FORMAT_TYPE18	17	14 時 00 分(24 Hour Format)
TIME _FORMAT_TYPE19	18	02:00(12 Hour Format)
TIME _FORMAT_TYPE20	19	14:00(24 Hour Format)

5. ドキュメントマクロで使用出来るコマンド

ドキュメントマクロで使用出来るコマンドは、以下の通りです。

名称	コマンド文字列	引数
フィード	FEED	紙送り量
バックフィード	BACK_FEED	紙送り量
フルカット	FULL_CUT	なし
パーシャルカット	PARTIAL_CUT	なし
プリンタの初期化	PRT_INIT	なし
印字濃度設定	PRT_DENSITY	印字濃度
マーキングポジションの検知	MARK_DETECTION	なし
マーキングポジションの再検知	RE-MARK_DETECTION	なし
不揮発性メモリへの登録開始	REG_BEGINNING	メモリ番号
不揮発性メモリへの登録終了	REG_END	メモリ番号

ドキュメントマクロで実際に設定するには、以下の様に記述します。

```
string[] start = { @"PRT_INIT", @" PRT_DENSITY,120"};   //ページ開始マクロ
string[] end = { @"FEED,120"};                          //ページ終了マクロ
string item = @"";                                      //オブジェクト名

SetMacroData(ref item, start, end);                     //マクロの設定
```

6. スタンドアロンマクロで使用出来るコマンド

スタンドアロンマクロで使用出来るコマンドは、以下の通りです。

名称	コマンド文字列	引数	その他
フィード	FEED	紙送り量	
バックフィード	BACK_FEED	紙送り量	
プリンタの初期化	PRT_INIT	なし	
印字濃度設定	PRT_DENSITY	印字濃度	
改行	CR	なし	
スペース	SPACE	なし	
寄せ配置	ALIGNMENT	指定位置	
プリンタフォントサイズ	FONT_SIZE	なし	
時間印字	TIME_PRINT	書式番号	
時間読み込み	TIME_READ	なし	
オフセット付き時間印字	OFFSET_TIME_PRINT	オフセット種類、オフセット値、書式	
カウンタ印字	COUNTER_PRINT	なし	
ゼロサプレス設定	ZERO_SUPPRESS	桁数、ゼロサプレス設定	
カウンタ増減設定	INCDEC_SETTING	カウンタ増減タイミング、増減数、増減実行までの回数、増加 or 減算	
漢字コード選択	KANJI_SELECT	漢字コード体系	
QR コードの印字	QR_CODE	サイズ、ECC レベル	QR 文字列使用
フルカット	FULL_CUT	なし	
パーシャルカット	PARTIAL_CUT	なし	
マーキングポジションの検知	MARK_DETECTION	なし	
登録イメージ印刷	BATCH_PRINT	なし	
文字列変数設定	STRVAL_SET	変数番号	変数文字列使用
セルサイズ設定	CELL_SET	セルサイズ	

スタンドアロンマクロで指定するコマンド文字列の形式は、次の様になります。

“コマンド, 引数1, 引数 2, 引数 3, 引数 4, 引数 5, 引数 6, QR 文字列, 変数文字列, コマンド個数”
コマンド　　: FEED や CR のようなコマンド文字列です。
引数　　: コマンドの引数で 0 から 255 までの数値を指定します。
　　引数の指定がない場合でも 0 を指定しておきます。
QR 文字列　: QR コードのデータを設定する場合に使用します。
　　QR 文字列の設定以外では何も指定しません。
変数文字列　: 文字列変数を設定する場合に使用します。
　　文字列変数の設定以外では何も指定しません。
コマンド個数 : コマンドの数を指定します。
　　範囲は、1～30 までです。

スタンドアロンマクロのコマンド詳細

・フィード
　機能: 指定したドット数のフィードを行います。
　引数: 紙送りのドット数(1～255)を指定します。

・バックフィード
　機能: 指定したドット数のバックフィードを行います。
　引数: 紙送りのドット数(1～255)を指定します。

・プリンタの初期化
　機能: プリンタの初期化を行います。

・印字濃度設定
　機能: プリンタの印字濃度を設定します。
　引数: 印字濃度 (50～150%) を指定します。

・改行
　機能: 1 行分の改行を行います。

・スペース
　機能: 半角 1 桁のスペースを挿入します。

・位置揃え
　機能: 1 行の印字データを指定位置に揃えます。
　引数: 指定位置 (0～2) を指定します。
　　0:左寄せ
　　1:中央寄せ
　　2:右寄せ

・文字サイズ

機能： 文字の縦・横倍を指定します。

引数： 文字サイズを指定します。

0：縦横 1 倍

17：縦横 2 倍

34：縦横 3 倍

51：縦横 4 倍

68：縦横 5 倍

85：縦横 6 倍

102：縦横 7 倍

119：縦横 8 倍

・日付/時刻の印字

機能： 日付/時刻の印字を行います。

引数： 日付/時刻の印字フォーマットを指定します。

0：YYYY/MM/DD hh:mm

1：YY/MM/DD hh:mm

2：YYYY 年 MM 月 DD 日 hh 時 mm 分

3：YY 年 MM 月 DD 日 hh 時 mm 分

4：YYYY/MM/DD

※ “年月日”を印字する場合、事前に漢字コード選択マクロを指定します。

日付/時刻はプリンタの WLAN/ROUTER モードを有効にしておく事で動作します。

・時刻の読み込み

機能： 日付/時刻の読み込みを行います。

プリンタのマクロ動作時に自動で日付/時刻の読み込みを行うため通常設定は不要です。

・オフセット時刻の印字

機能： 指定した日付と時刻のオフセットを加算して印字します。

引数 1： オフセットのタイプを指定します。

100：月

101：日

102：時

引数 2： オフセットする数値を指定します。

オフセットタイプ月：0～12

オフセットタイプ日：0～31

オフセットタイプ時：0～24

引数 3： 印字する時刻フォーマットを指定します。

0：YYYY/MM/DD hh:mm

1：YY/MM/DD hh:mm

2：YYYY 年 MM 月 DD 日 hh 時 mm 分

3：YY 年 MM 月 DD 日 hh 時 mm 分

4：YYYY/MM/DD

・カウンタの印字

機能： インクリメント機能にあるカウンタメモリを印字します。

・ゼロサプレス設定

機能： インクリメント機能の印字桁数とゼロサプレスを設定します。
引数 1： 印字桁数(1～6)を指定します。
引数 2： ゼロサプレス(0～2)を指定します。
0:ゼロサプレスの桁位をスペースに変換します。
1:ゼロサプレスを行いません。
2:ゼロサプレスの桁数を左詰めにします。

・カウンタ増減設定

機能： インクリメントの増減を設定します。
引数 1： インクリメントのイベントを設定します。
('0' 固定:カウンタ印字の実行時に行います。)
引数 2： インクリメントの増減数(1～255)を指定します。
引数 3： インクリメントを行うまでの印字回数(1～255)を指定します。
引数 4： カウンタの増減を指定します。
0:インクリメントを増加します。
1:インクリメントを減算します。

・漢字コード選択

機能： 漢字モードを指定します。

・QR コード

機能： QR コードを印字します。
引数 1： シンボルサイズ(1～40)を指定します。
引数 2： ECC LV(1=L,2=M,3=Q,4=H)を指定します。
※ 文字列の入力データは「QR 文字列」にセットします。

・フルカット

機能:フルカットを行います。

・パーシャルカット

機能： パーシャルカットを行います。

・マーキングポジションの検知

機能： マーキングポジションの検知を行います。

・登録データの一括印字

機能： プリンタに登録されているレイアウトデータを印字します。

・文字列変数の設定

機能： データストリングの文字列変数を設定します。
データストリングは最大 8 個まで登録可能です。
引数： 登録番号(0～7)を指定します。
※ 文字列の入力データは「変数文字列」にセットします。

・セルサイズの設定

機能： QR コードのセルサイズを設定します。
引数： セルサイズ(3～20)を指定します。

QR 文字列中で使用可能なコマンドは、以下の通りです。

名称	コマンド文字列	引数	機能
時刻のセット 0	$TIM0	なし	「YYYYMMDDhhmm」の形式で QR データ中に時刻をセット
時刻のセット 1	$TIM1	なし	「YYMMDDhhmm」の形式で QR データ中に時刻をセット
時刻のセット 2	$TIM2	なし	「YYYYMMDDhhmm」の形式で QR データ中に時刻をセット
時刻のセット 3	$TIM3	なし	「YYMMDDhhmm」の形式で QR データ中に時刻をセット
時刻のセット 4	$TIM4	なし	「YYYYMMDD」の形式で QR データ中に時刻をセット
時刻のセット 5	$TIM5	なし	EPOCH 形式で QR データ中に時刻をセット
オフセット付き時刻のセット 0	$TIMd	月のオフセット、書式番号	書式番号は、0～4 までで時刻のセット 0～4 に対応する
オフセット付き時刻のセット 1	$TIMe	日のオフセット、書式番号	書式番号は、0～4 までで時刻のセット 0～4 に対応する
オフセット付き時刻のセット 2	$TIMf	時のオフセット、書式番号	書式番号は、0～4 までで時刻のセット 0～4 に対応する
カウンタインクリメント	$INC	なし	カウンタを更新する
文字列変数のセット 0	$STR0	なし	文字列変数 0 の内容をセットする
文字列変数のセット 1	$STR1	なし	文字列変数 1 の内容をセットする
文字列変数のセット 2	$STR2	なし	文字列変数 2 の内容をセットする
文字列変数のセット 3	$STR3	なし	文字列変数 3 の内容をセットする
文字列変数のセット 4	$STR4	なし	文字列変数 4 の内容をセットする
文字列変数のセット 5	$STR5	なし	文字列変数 5 の内容をセットする
文字列変数のセット 6	$STR6	なし	文字列変数 6 の内容をセットする
文字列変数のセット 7	$STR7	なし	文字列変数 7 の内容をセットする
チェックサム	$CHK	なし	データ末尾にチェックサムを追加する

スタンドアロンマクロを実際に登録するには、以下の様に記述します。

```
string stand_cmd1 = { @" QR_CODE,8,2,0,0,0,0,SANEI ELEC,,1"};      //QR コードを印字する
string stand_cmd1 = { @" CR,0,0,0,0,0,0,,,10"};                    //改行を10回行う

InitStandaloneMacro();                                             //スタンドアロンマクロの初期化
SetStandaloneMacroTitle(0, @"1 QR TEST");                          //表示名を設定
RegistStandaloneMacro(@"192.168.10.100", 9100, 0);                 //プリンタにスタンドアロンマクロを登録
```

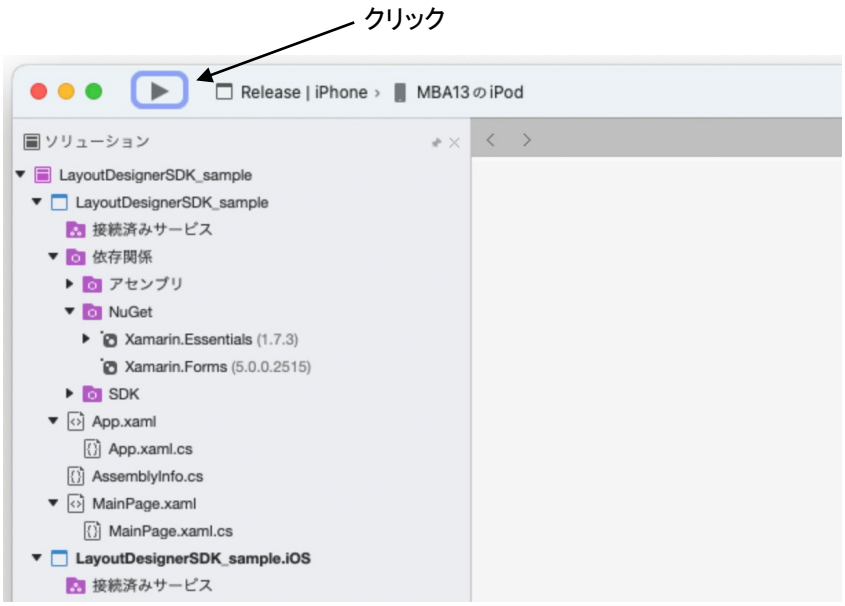
スタンドアロンマクロ登録時は、事前に背景データとして SOP ファイルを読み込んでおいてください。
背景データが存在しないと登録に失敗します。

付録 1. サンプルプログラムについて

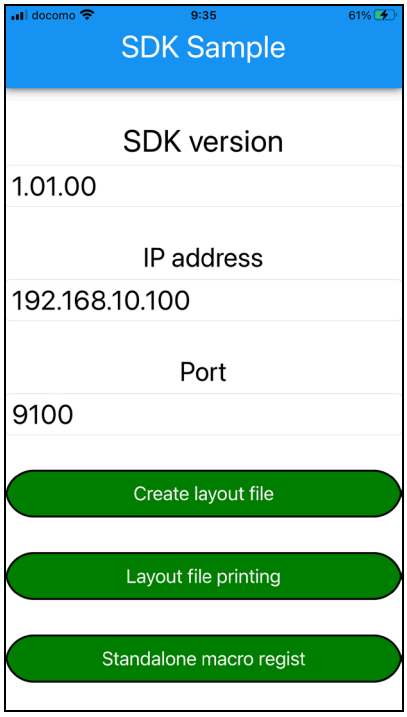
SDK の中に、下図のラベルイメージを作成するサンプルプログラムを付属しています。
API の使い方などの参考にしてください。

インストール（ビルド）について

1. LayoutDesignerSDK_sample.sln ファイルをダブルクリックします。
2. VisualStudio が立ち上がります。
3. 対象の iOS デバイスと Mac を USB ケーブルで接続します。
4. 実行アイコンをクリックします。

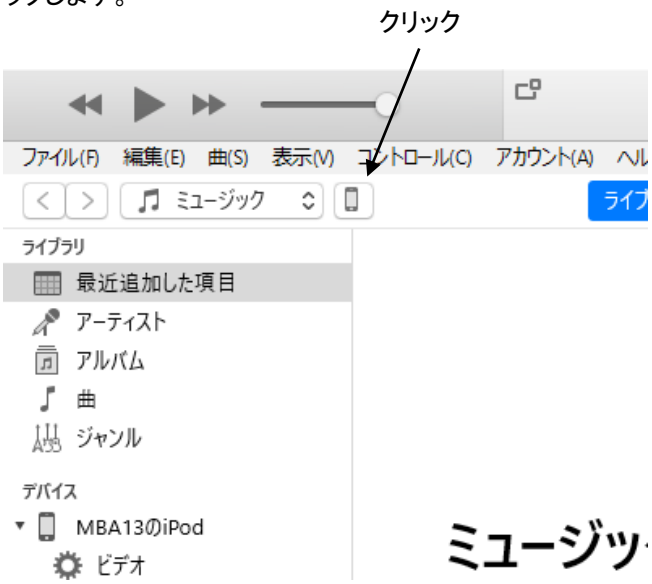


5. iOS デバイスに下のような画面が表示されればインストール成功です。

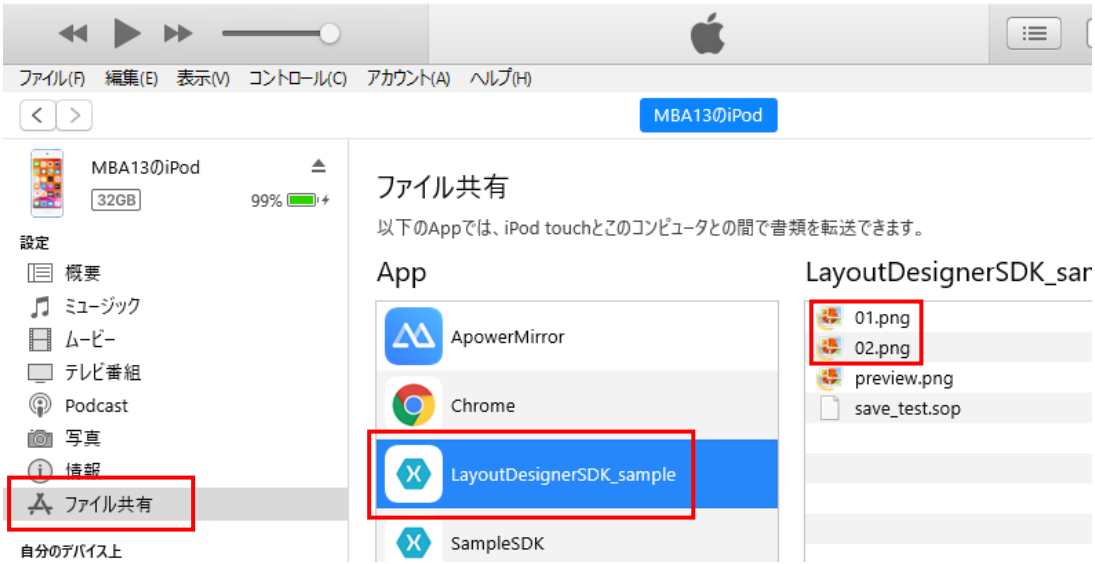


サンプルアプリ画面

6. iTunes を立ち上げます。
7. デバイスアイコンをクリックします。



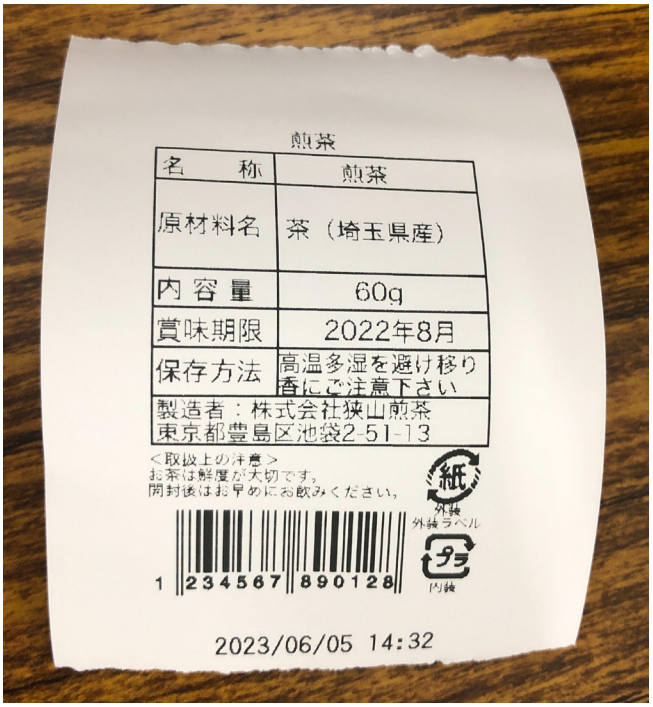
8. ファイル共有を選択します。
9. LayoutDesignerSDK_sample を選択します。
10. C#_sample フォルダ内の 01.png と 02.png を追加します。



11. 以上でサンプルアプリを動かす準備が出来ました。

サンプルアプリの動かし方について

1. iOS デバイスと対象プリンタが同一ネットワークになるようにあらかじめ設定しておきます。
2. サンプルアプリを立ち上げます。
3. IP Address と Port を対象プリンタに合わせます。
4. Create layout file をクリックします。
5. Layout file printing をクリックします。
6. 正常に動作すると下のような印字となります。



印字イメージ

付録 2. ファイルの共有について

Windows 側で作成した SOP ファイルを iOS 側で読んだり、逆に iOS 側で作成した SOP ファイルを Windows 側に取り出したりする場合は、iTunes からファイルの共有を使ってアプリケーション単位でやり取りします。
その場合、info.plist に以下のキーをセットしてください。

```
<key>UIFileSharingEnabled</key>
<true/>
```

このキーは、Visual Studio からではセットできないのでエディタを使って書き込む必要があります。

付録 3. 機能の制限について

レイアウトファイルは、異なる OS であっても相互に利用できますが機能の制限も存在します。
iOS の場合、以下の制限があります。

・1 つのフォントファイルに対してゴシックやイタリック修飾を掛ける事が出来ない。
　iOS の場合、フォント名でゴシックフォントやイタリックフォントを指定して描画します。
・Windows のフォント名が使用できない。
　iOS では、Windows のフォントを使用できないので AlternativeFontNameSet を使用して代わりのフォント名を指定する事で対処します。