SATO
SBPL プロバイダ

Version 1.0.0

ユーザーズ ガイド

February 14, 2020

備考:

【改版履歴】

バージョン	日付	内容
1.0.0	2020-02-14	初版

【動作確認機器】

機種	注意事項
CL4NX-J	TCP/IP 接続に対応. モデル名称: CL4NX-J 305dpi メインファームウェアバージョン: 1.9.2-r1

目次

1. はじめに

本書は，SBPL[1]コマンドを通して株式会社サトー(以下，サトー社)製プリンターを制御するための CAO プロバイダのユーザーズガイドです．本プロバイダで対応しているコマンドは一部サトー社製プリンターでは未対応のものがありますので，6.2 を確認の上ご利用ください．

本書で扱う CAO プロバイダ(CaoProvSATOSBPL.dll)を SBPL プロバイダと呼びます．第 2 章以降ではプロバイダが提供する機能と変数について説明します．

[1] SBPL（SATO Barcode Printer Language）とは，サトー社のバーコードラベルプリンタを制御する共通コマンドのことです．

2. プロバイダの概要

2.1. 概要

SBPL プロバイダは, コマンドの実行方法として, CaoController::Execute による方法を提供しています. CaoController::Execute では, Ethernet インターフェースを利用して SBPL コマンドの送受信を行います.

2.2. インストール

SBPL プロバイダのファイル形式は DLL(Dynamic Link Library)であり, CAO エンジンから使用時に動的ロードされます. 使用するにあたっては ORiN2SDK をインストールするか, 表 2-1 を参照して手作業でレジストリ登録を行う必要があります.

表 2-1 SBPL プロバイダ

ファイル名	CaoProvSATOSBPL.dll
ProgID	CaoProv.SATO.SBPL
レジストリ登録	regsvr32 CaoProvSATOSBPL.dll
レジストリ登録の抹消	regsvr32 /u CaoProvSATOSBPL.dll

2.3. メソッド・プロパティ

2.3.1. CaoWorkspace::AddController メソッド

SBPL プロバイダは AddController 時に接続パラメータを参照し，通信の接続を行います．このときオプションで接続先とタイムアウトを指定します．

書式　AddController (<bstrCtrlName:VT_BSTR>,<bstrProvName:VT_BSTR>,
<bstrPcName:VT_BSTR>, [<bstrOption:VT_BSTR>])

<bstrCtrlName> : [in] コントローラ名
<bstrProvName> : [in] プロバイダ名．固定値 ="CaoProv.SATO.SBPL"
<bstrPcName> : [in] プロバイダの実行マシン名（未使用）
<bstrOption> : [in] オプション文字列

以下にオプション文字列に指定するリストを示します．

表 2-2 CaoWorkspace::AddController のオプション文字列

オプション	意味
Conn=<接続パラメータ>	必須．通信形態とその接続パラメータを設定します．(参照：2.3.1.1)
ConnTimeout[=<タイムアウト時間>]	任意．TCP 接続時のタイムアウト時間をミリ秒で指定します． （デフォルト：500）
Timeout[=<タイムアウト時間>]	任意．コマンド送受信時のタイムアウト時間をミリ秒で指定します． （デフォルト：3000）

2.3.1.1. Conn オプション

以下に Conn オプションの接続パラメータ文字列を示します．ここで角括弧("[ ]")内は省略可能を示します．また，各パラメータの解説中の下線部はオプションを指定しなかったときのデフォルト値を示します．

【Ethernet デバイス】
"Conn=ETH:<Dest IP Address>[:<Dest Port No>]"
"Conn=TCP:<Dest IP Address>[:<Dest Port No>]"
<Dest IP Address> : 接続する機器の IP アドレス．
<Dest Port No> : 接続ポート番号．9100，9101，・・・任意指定可能

使用例

```
CaoEngine caoEng;
CaoWorkspaces caoWss;
CaoWorkspace caoWs;
CaoControllers caoCtrls;
CaoController caoCtrl;

caoEng = new CaoEngine();
caoWss = caoEng.Workspaces;
caoWs = caoWss.Item(0);
caoCtrls = caoWs.Controllers;

// Connect by LAN.
caoCtrl = caoWs.AddController("SBPL_SAMPLE", "CaoProv.SATO.SBPL", null,
                              "Conn=TCP:192.168.1.2:9100,Timeout=3000");
```

2.3.2. CaoController::AddVariable メソッド

CaoController クラスの AddVariable メソッドは，変数オブジェクトを作成するためのメソッドです．変数名には，表 2-3 に示す変数名のみ使用できます．

書式　AddVariable (<bstrVariableName:VT_BSTR>, [<bstrOption:VT_BSTR>])

<bstrVariableName>　:　[in] 変数名
<bstrOption>　:　[in] オプション文字列

使用例

```
// Add variable to get device version
CaoVariable val;
val = caoCtrl.AddVariable("@DEVICE_VERSION");

// GetValue
var value = val.Value;
/* example  1.9.2-r1 */
```

2.3.3. CaoController::GetVariableNames メソッド

AddVariable メソッドで指定できるシステム変数名の一覧を取得します．

2.3.4. CaoController::Execute メソッド

CaoController クラスの Execute メソッドは，コマンドを実行するためのメソッドです．各コマンドの詳細はコマンドリファレンスをご参照ください．

書式　Execute (<bstrCommandName:VT_BSTR>,[<vntParam:VT_VARIANT>])

< bstrCommandName>　:　[in] コマンド名
<vntParam>　:　[in] パラメータ

2.4. 変数一覧

2.4.1. CaoController クラス

表 2-3 CaoController クラスのシステム変数一覧

変数名	データ型	説明	属性	
			get	put
@MAKER_NAME	VT_BSTR	メーカー名="SATO"を返す.	○	–
@VERSION	VT_BSTR	プロバイダバージョンを返す.	○	–
@DEVICE_VERSION	VT_BSTR	ファームウェアのバージョンを返す.	○	–

2.5. エラーコード

SBPL プロバイダでは, 以下の固有のエラーコードを返します.

表 2-4 固有エラーコード

エラー名	エラー番号	説明
応答なし	0x80100000	プリンタからデータが受信できない場合に返します. プリンタの通信設定と接続オプションが一致しているか確認してください.
受信データ異常	0x80100001	データが欠損しているなど, 想定外のデータを受信した場合に返します.
ロック指定エラー	0x80100002	CaoController::Execute("WriteRfidPassLock") コマンドの際に, ロック領域指定パラメータで「ロック領域未指定(0,0,0,0,0)」が入力された場合に返します.
エラー応答	0x801001XX	コマンドの応答結果としてエラーを受信した場合に返します. プリンタから受信した 16 進数のエラーコード[2]が XX に挿入されます. 例)22 → 0x80100122

[2] エラーコードの詳細に関しては各プリンターの取扱説明書を参照してください.

3. 制限事項

3.1. 通信プロトコル

SBPL プロバイダは，通信プロトコルのステータス 4（ENQ 応答モード）を使用します．SBPL プロバイダ使用前にプリンタの通信プロトコルがステータス 4（ENQ 応答モード）に設定されていることを確認してください．

3.2. プロトコルコード

SBPL プロバイダは，SBPL コマンドを通して制御を行います．プリンタの SBPL 設定として，プロトコルコードが有効に設定されていることを確認してください．無効に設定されていると，SBPL プロバイダは正常に動作しません．

3.3. コマンド送信間隔

機器の仕様により，連続してコマンドを送る場合に一定の間隔を空けて送信する必要があります．

表 3-1 コマンド送信間隔の制限

処理	送信間隔(ミリ秒)
システム変数「@DEVICE_VERSION」の値取得（get_Value プロパティ）	5000
CaoController::Execute("Enq")コマンド	5

3.4. 印字フォーマットの登録

CaoController::Execute("PrintFormat")コマンドは，予めプリンタに登録されたフォーマットに対して印字データを埋め込んで印字します．そのため，CaoController::Execute("PrintFormat")コマンド使用前には必ず印字フォーマットの登録を行ってください．

印字フォーマットはプリンタ設定ツールである「All-In-OneTool[3]」の SBPL コマンド送信機能か，CaoController::Execute("Raw")コマンドで「フォーマット登録指定」「フィールド登録指定」コマンドを実行することで登録可能です．

3.5. ステータス要求

CaoController::Execute("Enq")コマンドでステータス要求を送信することが可能ですが，印字データ（STX <A>～<Z> ETX）送信中のステータス要求は行わないでください．印字データ送信中にステータス要求を送信した場合，正常にステータスが取得できない，または正常に印字されない場合があります．また，印字データの送信完了後にステータス要求を行う場合は，送信間隔を実機と調整してから利用してください．
参考：CL4NX-J 接続時は 200ms 以上の間隔で正常動作の確認をしています．

3.6. RFID 関連のコマンド

RFID 対応機種限定の機能になります． RFID 対応機種に関してはサトー社の HP を確認してください．

[3] All-In-OneTool はサトー社の製品ページよりダウンロード可能です．

4. コマンドリファレンス

4.1. CaoController クラス

表 4-1 CaoController クラス コマンド一覧

4.1.1. CaoController::Execute("PrintFormat") コマンド

登録済みフォーマットを使用した印刷を行います. 使用前に 3.4 をご参照ください.

書式　PrintFormat(<PrintParam>)

<PrintParam> : [in]印字パラメータ

VT_ARRAY | VT_VARIANT			
0	VT_UI2	フォーマット番号.	必須.
1	VT_ARRAY | VT_BSTR	印字データ. テンプレートのフィールド番号順に入力. 1 つのフィールドには 1～99 文字指定可. フィールド数は 99 まで.	必須.
2	VT_UI4	任意. 印字枚数(VT_UI4). 1～999999 の数値を指定.（デフォルト:1）	任意.

戻り値 : なし

使用例1

```
var printData = new string[] { "ABCDE", "12345" };
// Print 10 sheets using format number 1
caoCtrl.Execute("PrintFormat", new object[]{ (ushort)1, printData, 10U});
```

使用例2 指定フィールドに何も印字させない場合は，フィールド値をブランクで設定してください．

```
var printData = new string[] { "", "12345" };
// Print 10 sheets using format number 1
caoCtrl.Execute("PrintFormat", new object[]{ (ushort)1, printData, 10U});
```

4.1.2. CaoController::Execute("PrintBarcodeEAN13") コマンド

バーコード(EAN13)を印字します．

書式　PrintBarcodeEAN13(<PrintParam>)

<PrintParam> ： [in]印字パラメータ

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	印字データ． 0～9 の数値を 11～13 桁で指定．	必須．
1	VT_UI2	印字位置　縦． 1～18000dot の数値を指定． （デフォルト:1）	任意．
2	VT_UI2	印字位置　横． 1～18000dot の数値を指定． （デフォルト:1）	任意．
3	VT_UI4	印字枚数． 1～999999 の数値を指定．（デフォルト:1）	任意．
4	VT_UI2	ナローバー幅． 1～12 の数値を指定．（デフォルト:4）	任意．
5	VT_UI2	バーコードの高さ． 1～999 の数値を指定．（デフォルト:120）	任意．

戻り値 ： なし

使用例

印字データ「123456789」を縦 100dot,横 100dot,ナローバー幅 3dot,バーコードの高さ 120dot で 10 枚印字

```
// Print 10 sheet with some setting
caoCtrl.Execute("PrintBarcodeEAN13", new object[]{"12345678901", (ushort)100, (ushort)100,
10U, (ushort)3, (ushort)120});
```

4.1.3. CaoController::Execute("PrintQRCode") コマンド

QR コード(モデル 2)を印字します. このコマンドで印字できるのは通常モードの QR コード(モデル 2)です.

書式　PrintQRCode (<PrintParam>)

<PrintParam> : [in]印字パラメータ

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	印字データ[4]. データ数は 1～2953.	必須.
1	VT_UI2	印字位置　縦. 1～18000dot の数値を指定.（デフォルト:1）	任意.
2	VT_UI2	印字位置　横. 1～18000dot の数値を指定.（デフォルト:1）	任意.
3	VT_UI4	印字枚数. 1～999999 の数値を指定.（デフォルト:1）	任意.
4	VT_BSTR	誤り訂正レベル.（デフォルト:L） L　:　7% M　:　15% Q　:　25% H　:　30%	任意.
5	VT_UI2	セル 1 辺のサイズ. 1～99 の数値を指定.（デフォルト:4）	任意.
6	VT_UI1	データ設定モード.（デフォルト:0） 0　:　自動設定 1　:　数字 2　:　英数字 3　:　漢字	任意.

戻り値 : なし

使用例

印字データ「123456789」を縦 100dot,横 100dot,誤り訂正レベル M(15%),セル一辺 5dot,データ設定モード 1(数字)で 10 枚印字

```
// Print 10 sheet with some setting
caoCtrl.Execute("PrintQRCode", new object[]{"123456789", (ushort)100, (ushort)100, 10U, "M",
(ushort)5, (byte)1});
```

[4] 印字データに英字, 数字, 漢字, すべてを含める場合は, データ設定モードを 0 に設定してください.

4.1.4. CaoController::Execute("PrintRfidUhf") コマンド

RFID(UHF)を印字します.

書式　PrintRfidUhf (<PrintParam>)

<PrintParam> : [in]印字パラメータ

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	EPC 領域登録データ[5]. 文字数は 4～124.	必須.
1	VT_BSTR	印字データ. 半角英数字, 漢字混合可.	必須.
2	VT_UI2	印字位置　縦. 1～18000dot の数値を指定.（デフォルト:1）	任意.
3	VT_UI2	印字位置　横. 1～18000dot の数値を指定.（デフォルト:1）	任意.
4	VT_UI4	印字枚数. 1～999999 の数値を指定.（デフォルト:1）	任意.
5	VT_BSTR	USER メモリ領域登録データ. 文字数は 4～128.	任意.

戻り値 : なし

使用例

EPC 領域登録データ 0000,印字データ「12345ABCD」,縦 1dot,横 1dot,USER メモリ領域登録データ 9999 で 5 枚印字

```
// Print 5 sheet with some setting
caoCtrl.Execute("PrintRfidUhf", new object[]{"0000", "12345ABCDE", (ushort)1, (ushort)1, 5U,
"9999"});
```

[5] EPC 領域に登録できるデータ文字数は, 印字する RF ラベルに搭載されているチップ仕様により異なります. 詳しくは印字する RF ラベル仕様を確認してください.

4.1.5. CaoController::Execute("PrintRfidUhfForFormat") コマンド

RFID ラベルの印字に登録済みフォーマットを使用して，RFID(UHF)印字を行います．使用前に 3.4 をご参照ください．

書式　PrintRfidUhf (<PrintParam>)

<PrintParam> ： [in]印字パラメータ

VT_ARRAY | VT_VARIANT			
0	VT_BSTR	EPC 領域登録データ[5]． 文字数は 4～124.	必須.
1	VT_UI2	フォーマット番号.	必須.
2	VT_ARRAY | VT_BSTR	印字データ. テンプレートのフィールド番号順に入力. 1 つのフィールドには 1～99 文字指定可. フィールド数は 99 まで.	必須.
3	VT_UI4	印字枚数. 1～999999 の数値を指定.（デフォルト:1）	任意.
4	VT_BSTR	USER メモリ領域登録データ. 文字数は 4～128.	任意.

戻り値 ： なし

使用例

EPC 領域登録データ 0000，フォーマット番号 1 のフォーマットを使用して，USER メモリ領域登録データ 9999 で 5 枚印字

```
// Print 5 sheet with some setting
var printData = new string[] { "ABCDE", "12345" };
caoCtrl.Execute("PrintRfidUhfForFormat", new object[]{"0000", (ushort)1, printData, 5U,
"9999"});
```

4.1.6. CaoController::Execute("WriteRfidPassLock") コマンド

ロック機能[6]付き RFID に対し, パスワードロックまたはアンロックを行います.

書式　WriteRfidPassLock (<WriteParam>)

<WriteParam> : [in]書き込みパラメータ

VT_ARRAY | VT_VARIANT			
0	VT_UI1	ロックタイプ指定. 0 : ロック 1 : アンロック	必須.
1	VT_ARRAY | VT_UI2	ロック領域指定. ロックまたはアンロックする領域に 1, それ以外は 0 を指定. 配列の要素番号と領域の対応は以下の通り. 0 USER メモリ領域 1 TID 領域 2 ACCESS 領域 3 KILL 4 EPC 領域	必須.
2	VT_BSTR	アクセスパスワード指定. 有効桁数 8 桁の 16 進文字列で指定.	必須.
3	VT_ARRAY | VT_BSTR	登録データ. EPC 領域, USER メモリ領域どちらか一方を必ず指定. 配列の要素番号とデータの対応は以下の通り. 0 EPC 領域登録データ[5]. 文字数は 4～124. 1 USER メモリ領域登録データ. 文字数は 4～128.	必須.

戻り値 : なし

使用例

EPC 領域データ 12345678, USER メモリ領域データ 12345678123456,USER メモリ領域をロック

```
// Lock with password
var lockBank = new ushort[] { 1, 0, 0, 0, 0 };
var writeData = new string[] { "12345678", "12345678123456" };
caoCtrl.Execute("WriteRfidPassLock", new object[] { (byte)0, lockBank, "12345678", writeData });
```

[6] ロック仕様に関しては, 使用する RF ラベルに搭載されているチップ仕様により異なります. 詳しくは RF ラベル仕様を確認してください.

4.1.7. CaoController::Execute("WriteRfidPermLock") コマンド

ロック機能[6]付き RFID に対し，パーマネントロックまたはパーマネントアンロックを行います．

書式　　WriteRfidPermLock (<WriteParam>)

<WriteParam>　：　[in]書き込みパラメータ

<table>
<tr><td colspan="4">VT_ARRAY | VT_VARIANT</td></tr>
<tr><td>0</td><td>VT_ARRAY
| VT_UI2</td><td>ロック指定．領域ごとに実行するロック種別を指定．
0　：　ロックなし
1　：　パーマネントロック
2　：　パーマネントアンロック

配列の要素番号と領域の対応は以下の通り．<table><tr><td>0</td><td>USER メモリ領域</td></tr><tr><td>1</td><td>TID 領域</td></tr><tr><td>2</td><td>ACCESS 領域</td></tr><tr><td>3</td><td>KILL</td></tr><tr><td>4</td><td>EPC 領域</td></tr></table></td><td>必須．</td></tr>
<tr><td>1</td><td>VT_ARRAY
| VT_BSTR</td><td>登録データ．EPC 領域，USER メモリ領域どちらか一方を必ず指定．
配列の要素番号とデータの対応は以下の通り．<table><tr><td>0</td><td>EPC 領域登録データ[5]．
文字数は 4～124．</td></tr><tr><td>1</td><td>USER メモリ領域登録データ．
文字数は 4～128．</td></tr></table></td><td>必須．</td></tr>
</table>

戻り値　　：　なし

使用例

EPC 領域データ 12345678, USER メモリ領域データ 1234567812345678,
USER メモリ領域をパーマネントロック

```
// Put a permanent lock
var lockBank = new ushort[] { 1, 0, 0, 0, 0 };
var writeData = new string[] { "12345678", "1234567812345678" };
caoCtrl.Execute("WriteRfidPermLock", new object[] { lockBank, writeData });
```

4.1.8. CaoController::Execute("Enq") コマンド

プリンタにステータス要求コマンドを送信し，現在実行中の印字処理のステータスを取得します．プリンタステータスのコードと内容については， 6.1.1 を参照してください．

書式　Enq()

引数　：　なし
戻り値　：　プリンタステータス(VT_UI1)

使用例

印字コマンド実行後，Enq コマンドでステータスを監視

```
var printData = new string[] { "ABCDE", "12345" };
// Print 100 sheets using format number 1
caoCtrl.Execute("PrintFormat", new object[] { (ushort)1, printData, 100U });

// print status monitoring
while (true) {
    var result = caoCtrl.Execute("Enq", null);
    Byte status = Convert.ToByte(result);

    if (status == 0x30 || status == 0x41) {
        Console.WriteLine("Print complete.");
        break;
    }
    else {
        if (0x62 <= status && status <= 0x71) {
            Console.WriteLine(string.Format("Error :{0:X}", status));
            break;
        }
        else {
            // Printing
            continue;
        }
    }
}
```

4.1.9. CaoController::Execute("Raw") コマンド

引数で指定された SBPL コマンドをプリンタに送信します．SBPL コマンド文字列を指定する際は，コマンドコードを<>で囲ってください．プリンタに送信する際は，引数で指定された SBPL コマンドの先頭に STX，末尾に CRLF と ETX を付加します．

書式　Raw (<Command>)

<Command>　：　[in]SBPL コマンド(VT_BSTR)
戻り値　：　なし

使用例1

```
// [ESC + IK] Send a command to feed 120 dots in paper order
caoCtrl.Execute("Raw", new object[]{"<A><IK>0,120<Z>"});
```

使用例2 SBPLコマンド内の印字データとして<>を設定したい場合は，<<, >>と設定してください．

```
// [ESC + BG] Transmit CODE128 (128A, 128B, 128C) barcode print command
caoCtrl.Execute("Raw", new object[]{"<A><V>100<H>200<BG>02120>>GABCD123456<Q>2<Z>"});
```

4.1.10. CaoController::Execute("GetStatus") コマンド

プリンタにプリンタステータス情報取得コマンドを送信し， CaoController::Execute("Enq") コマンド よりも詳細なプリンタステータスを取得します．ステータス内容については， 6.1.2 を参照してください．

書式 GetStatus ()

引数 : なし

戻り値 : プリンタステータス

VT_ARRAY | VT_UI4		
0	VT_UI4	プリンタステータス.
1	VT_UI4	受信バッファステータス.
2	VT_UI4	リボンステータス.
3	VT_UI4	用紙ステータス.
4	VT_UI4	エラーNo..
5	VT_UI4	バッテリステータス.
6	VT_UI4	発行枚数残量. 最大 999999 枚.

使用例

```
var result = caoCtrl.Execute("GetStatus", null);
uint[] status = (uint[])result;
```

5. サンプルプログラム

以下に，本プロバイダを使用して CLNX-J シリーズと通信を行い，機器情報の取得や Execute コマンドを実行するサンプルプログラム(C#)を示します．実行前に 3 を確認してください．

Sample **Program.cs**

```
…(略)…
using ORiN2.ManagedCAO;

namespace SATO_SBPL_Sample
{
    public partial class Sample : Form
    {
        private CCaoEngine eng;
        private CCaoWorkspace ws;
        private CCaoWorkspaces wss;
        private CCaoController ctrl;
        private CCaoControllers ctrls;
        private CCaoVariable makerName;
        private CCaoVariable provVersion;
        private CCaoVariable deviceVersion;

        public Sample()
        {
            InitializeComponent();
        }

        private void Program_Load(object sender, EventArgs e)
        {
            // Create CAO Engine
            this.eng = new CCaoEngine();
            this.wss = this.eng.Workspaces;
            this.ws = this.wss[0];
            this.ctrls = this.ws.Controllers;
        }

        private void btnConnect_Click(object sender, EventArgs e)
        {
            try
            {
                if (this.ctrl != null)
                {
                    this.ws.Controllers.Remove(this.ctrl.Index);
                    this.ctrl = null;
                }
            // Connect CL4NX-J
                // option : Conn=TCP:192.168.51.20:9100, ConnTimeout=1000, TimeOut=1000
                this.ctrl = this.ws.AddController("Sample",
                                                  "CaoProv.SATO.SBPL",
                                                  null,
                                                  textConnOption.Text);

                // Add system variable
                makerName = this.ctrl.AddVariable("@MAKER_NAME", null);
                provVersion = this.ctrl.AddVariable("@VERSION", null);
                deviceVersion = this.ctrl.AddVariable("@DEVICE_VERSION", null);
            }
            catch (Exception ex)
            {
                MessageBox.Show(this, ex.Message, this.Text,
                                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
}

private void btnGetVariable_Click(object sender, EventArgs e)
{
    try
    {
        // GetValue
        textMakerName.Text = Convert.ToString(makerName.Value);
        textVersion.Text = Convert.ToString(provVersion.Value);
        textDeviceVersion.Text = Convert.ToString(deviceVersion.Value);
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void btnExecute_Click(object sender, EventArgs e)
{
    try
    {
        switch (comboBox1.SelectedIndex)
        {
            case 0:
                textResult.Text = ExecPrintFormat();
                break;
            case 1:
                textResult.Text = ExecPrintBarcodeEAN13();
                break;
            case 2:
                textResult.Text = ExecPrintQRCode();
                break;
            case 3:
                textResult.Text = ExecPrintQRCodeJoint();
                break;
            case 4:
                textResult.Text = ExecPrintRfidUhf();
                break;
            case 5:
                textResult.Text = ExecPrintRfidUhfForFormat();
                break;
            case 6:
                textResult.Text = ExecWriteRfidPassLock();
                break;
            case 7:
                textResult.Text = ExecWriteRfidPermLock();
                break;
            case 8:
                textResult.Text = ExecEnq();
                break;
            case 9:
                textResult.Text = ExecRaw();
                break;
            case 10:
                textResult.Text = ExecGetStatus();
                break;
            default:
                break;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(this, ex.Message, this.Text,
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
```

```
    }

    private string ExecPrintFormat()
    {
        var param = new object[] {(ushort)1, new string[] { "ABCDE", "12345" }, 10U};
        var result = this.ctrl.Execute("PrintFormat", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }

    private string ExecPrintBarcodeEAN13()
    {
        var param = new object[] {"12345678901", (ushort)100, (ushort)100, 10U,
                                  (ushort)3, (ushort)120};
        var result = this.ctrl.Execute("PrintBarcodeEAN13", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }

    private string ExecPrintQRCode()
    {
        var param = new object[] {"123456789", (ushort)100, (ushort)100, 10U,
                                  "M", (ushort)5, (byte)1 };
        var result = this.ctrl.Execute("PrintQRCode", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }

    private string ExecPrintQRCodeJoint()
    {
        var param = new object[] {"123456789", (ushort)1, (ushort)1, 1U,
                                  "M", (ushort)5, (byte)1, (ushort)4, (byte)1};
        var result = this.ctrl.Execute("PrintQRCodeJoint", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }

    private string ExecPrintRfidUhf()
    {
        var param = new object[] {"12345678", "12345ABCDE", (ushort)1, (ushort)1,
                                  5U, "9999"};
        var result = this.ctrl.Execute("PrintRfidUhf", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }

    private string ExecPrintRfidUhfForFormat()
    {
        var param = new object[] {"12345678", (ushort)1,
                                  new string[] {"ABCDE", "12345" }, 5U, null};
        var result = this.ctrl.Execute("PrintRfidUhfForFormat", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }

    private string ExecWriteRfidPassLock()
    {
        var lockBank = new ushort[] { 1, 0, 0, 0, 0 };
        var writeData = new string[] { "12345678", "1234567812345678" };
        var param = new object[] { (byte)0, lockBank, "12345678", writeData };
        var result = this.ctrl.Execute("WriteRfidPassLock", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }

    private string ExecWriteRfidPermLock()
    {
        var lockBank = new ushort[] { 1, 0, 0, 0, 0 };
        var writeData = new string[] { "12345678", "1234567812345678" };
        var param = new object[] { lockBank, writeData };
        var result = this.ctrl.Execute("WriteRfidPermLock", param);
        return (result != null) ? Convert.ToString(result) : string.Empty;
    }
}
```

```
        private string ExecEnq()
        {
            var result = this.ctrl.Execute("Enq", null);
            return (result != null) ? Convert.ToString(result) : string.Empty;
        }

        private string ExecRaw()
        {
            var param = "<A><V>100<H>200<BG>02120>>GABCD123456<Q>2<Z>";
            var result = this.ctrl.Execute("Raw", param);
            return (result != null) ? Convert.ToString(result) : string.Empty;
        }

        private string ExecGetStatus()
        {
            var result = this.ctrl.Execute("GetStatus", null);
            if ((result != null) && (result is Array))
            {
                return string.Join(",", (uint[])result);
            }
            return string.Empty;
        }

        private void Sample_FormClosed(object sender, FormClosedEventArgs e)
        {
            // Release CAO object
            if (this.eng != null)
            {
                this.eng.Dispose();
                this.eng = null;
            }
        }
    }

}
```

6. 付録

6.1. プリンタステータス一覧表

6.1.1. ステータス要求

内容			10 進
オフライン状態	エラーなし		48
	リボン/ラベルニアエンド		49
	バッファニアフル		50
	リボン/ラベルニアエンド＆バッファニアフル		51
	印字停止中（エラーなし）		52
	（未使用）バッテリニアエンド		53
	（未使用）バッテリニアエンド＆リボン/ラベルニアエンド		54
	（未使用）バッテリニアエンド＆バッファニアフル		55
	（未使用）バッテリニアエンド＆リボン/ラベルニアエンド＆バッファニアフル		56
オンライン状態	受信待ち	エラーなし	65
		リボン/ラベルニアエンド	66
		バッファニアフル	67
		リボン/ラベルニアエンド＆バッファニアフル	68
		印字停止中（エラーなし）	69
		（未使用）バッテリニアエンド	33
		（未使用）バッテリニアエンド＆リボン/ラベルニアエンド	34
		（未使用）バッテリニアエンド＆バッファニアフル	35
		（未使用）バッテリニアエンド＆リボン/ラベルニアエンド＆バッファニアフル	36
	印字中	エラーなし	71
		リボン/ラベルニアエンド	72
		バッファニアフル	73
		リボン/ラベルニアエンド＆バッファニアフル	74
		印字停止中（エラーなし）	75
		（未使用）バッテリニアエンド	37
		（未使用）バッテリニアエンド＆リボン/ラベルニアエンド	38
		（未使用）バッテリニアエンド＆バッファニアフル	39
		（未使用）バッテリニアエンド＆リボン/ラベルニアエンド＆バッファニアフル	40
	待機中（ハクリ	エラーなし	77
		リボン/ラベルニアエンド	78

	・カット待ち)	バッファニアフル	79
		リボン/ラベルニアエンド＆バッファニアフル	80
		印字停止中(エラーなし)	81
		(未使用)バッテリニアエンド	41
		(未使用)バッテリニアエンド＆リボン/ラベルニアエンド	42
		(未使用)バッテリニアエンド＆バッファニアフル	43
		(未使用)バッテリニアエンド＆リボン/ラベルニアエンド＆バッファニアフル	44
	解析 ・編集中	エラーなし[7]	83
		リボン/ラベルニアエンド [7]	84
		バッファニアフル [7]	85
		リボン/ラベルニアエンド＆バッファニアフル [7]	86
		印字停止中(エラーなし)[7]	87
		(未使用)バッテリニアエンド	45
		(未使用)バッテリニアエンド＆リボン/ラベルニアエンド	46
		(未使用)バッテリニアエンド＆バッファニアフル	47
		(未使用)バッテリニアエンド＆リボン/ラベルニアエンド＆バッファニアフル	64
エラー検出	ヘッドオープン		98
	ペーパーエンド		99
	リボンエンド		100
	メディアエラー(印字飛びエラー)		101
	センサエラー/用紙詰まりエラー		102
	バーコード読取り／照合エラー		102
	バーコードリーダ接続確認エラー		102
	ヘッドエラー		103
	(未使用)カバーオープン		104
	カッタオープンエラー		104
	(未使用)リボンコアノンロックエラー		104
	カードエラー		105
	カッタエラー		106
	その他のエラー		107
	(未使用)カッタセンサエラー		108
	(未使用)スタッカ or リワインダフル 巻き取りフル		109

[7] 編集・解析のタイミングにより印字枚数がセットされない場合があります.

	RFID タグエラー	110
	RFID プロテクトエラー	112
	(未使用)バッテリエラー	113

6.1.2. プリンタステータス情報

No.	意味	プリンタステータス情報名称	プリンタステータス情報データ
1	プリンタステータス	PS	0:待機状態(受信待ち) 1:剥離待ち 2:解析中 3:印字中 4:オフライン 5:エラー発生中
2	受信バッファステータス	RS	0:バッファ空きあり 1:バッファニアフル 2:バッファフル
3	リボンステータス ※印字・フィード中に監視可能 動作停止中は正確な値が取得できません.	RE	0:リボンあり 1:リボンニアエンド 2:リボンなし 3:サーマル仕様
4	用紙ステータス ※印字・フィード中に監視可能 動作停止中は正確な値が取得できません.	PE	0:用紙あり(起動時含む) 1:ラベルニアエンド 2:用紙なし
5	エラーNo.[8]	EN	00:オンライン * エラーでないが返送 01:オフライン * エラーでないが返送 02:マシンエラー 03:メモリエラー 04:プログラムエラー 05:設定情報エラー(FLASH-ROM エラー) 06:設定情報エラー(EE-PROM エラー) 07:ダウンロードエラー 08:パリティエラー

[8] エラーNo.には, 本製品では発生しないエラーも記載されています.

			09:オーバーラン 10:フレーミングエラー 11:LAN タイムアウトエラー 12:バッファオーバー 13:ヘッドオープン 14:ペーパーエンド 15:リボンエンド 16:メディアエラー 17:センサエラー 18:ヘッドエラー 19:カバーオープンエラー 20:メモリ/カードタイプエラー 21:メモリ/カード読込み/書込みエラー 22:メモリ/カードフルエラー 23:メモリ/カードバッテリ無しエラー 24:リボンセーバーエラー 25:カッターエラー 26:カッターセンサエラー 27:スタッカーフルエラー 28:コマンドエラー 29:電源投入時のセンサエラー 30:RFID タグエラー 31:インタフェースカードエラー 32:リワインダーエラー 33:その他のエラー 34:RFID 制御エラー 35:ヘッド密度エラー 36:漢字データエラー 37:カレンダエラー 38:アイテム№.エラー 39:BCC エラー 40:カッターカバーオープンエラー 41:リボン巻取りノンロックエラー 42:通信タイムアウトエラー 43:リッドラッチオープンエラー 44:電源投入時の用紙無しエラー

			45:SD カードアクセスエラー 46:SD カードフルエラー 47:ヘッドリフトエラー 48:ヘッド温度異常エラー 49:SNTP 時刻補正エラー 50:CRC エラー 51:カッタモータエラー 53:スキャナ読取りエラー 54:スキャナ照合エラー 55:スキャナ接続エラー 56:Bluetooth モジュールエラー 57:EAP Authentication Error(EAP failed) 58:EAP Authentication Error(TimeOut) 59:バッテリエラー 60:ローバッテリエラー 61:ローバッテリエラー(充電中) 62:バッテリ未装着エラー 63:バッテリ温度エラー 64:バッテリ劣化エラー 65:モータ温度エラー 66:筐体内温度エラー 67:ジャムエラー 68:SIPL フィールドフルエラー 69:充電時パワーオフエラー 70:WLAN モジュールエラー 71:オプション不一致エラー 72:バッテリ劣化エラー(注意) 73:バッテリ劣化エラー(警告) 74:未電源オフエラー 75:NonRFID 警告エラー 76:バーコードリーダー接続エラー 77:バーコード読み取りエラー 78:バーコード読み取りエラー(検証開始位置異常) 79:バーコード照合エラー 80:NFC モジュールエラー

			81:NFC コマンドエラー
6	バッテリステータス	BT	0:正常 1:バッテリニアエンド 2:バッテリエラー
7	発行枚数残量	Q	000000～999999:発行残枚数 6 桁

6.2. コマンド対応表

変数名	get_Value	set_Value
@DEVICE_VERSION	DC2(12H)+PC	

Execute のメソッド名	コマンド名
PrintFormat	ESC(1BH)+YR, ESC(1BH)+/D
PrintBarcodeEAN13	ESC(1BH)+B
PrintQRCode	ESC(1BH)+2D30
PrintQRCodeJoint	ESC(1BH)+2D30
PrintRfidUhf	ESC(1BH)+IP0
PrintRfidUhfForFormat	ESC(1BH)+IP0, ESC(1BH)+YR, ESC(1BH)+/D
WriteRfidPassLock	ESC(1BH)+IP0
WriteRfidPermLock	ESC(1BH)+IP0
Enq	ENQ(05H)
GetStatus	DC2(12H)+PG
Raw	-