CASIO

GUI ライブラリマニュアル

このマニュアルは、GUI ライブラリの
仕様について記載します。

Ver. 1.01

ご注意

- このソフトウェアおよびマニュアルの、一部または全部を無断で使用、複製することはできません。
- このソフトウェアおよびマニュアルは、本製品の使用許諾契約書のもとでのみ使用することができます。
- このソフトウェアおよびマニュアルを運用した結果の影響については、一切の責任を負いかねますのでご了承ください。
- このソフトウェアの仕様、およびマニュアルに記載されている事柄は、将来予告なしに変更することがあります。
- このマニュアルの著作権はカシオ計算機株式会社に帰属します。
- 本書中に含まれている画面表示は、実際の画面とは若干異なる場合があります。予めご了承ください。

変更履歴

バージョン	変更日付	ページ	内容
1.00	2008.12		新規作成
1.01	2011.11	―	関数一覧に IT-9000 を追加
	2012.01	30-33	CpDecodeBarcode関数、CpSetBarcodeType関数、CpGetBarcodeType関数を追加
	2012.03	1	対象 OS に Windows Mobile を追加
	2012.03	5	IT-9000 用に CPEXTDVM 構造体の用紙とマーカ検出の指定方法を追加
	2012.03	41-44	サンプルソースを追加

目次

1. 概要

本ライブラリは、内蔵プリンタ用印刷システム/印刷ライブラリです。
本ライブラリを使用することにより、GDI 関数を用いて印刷データを作成して印刷することが可能になります。
※ ただし、GDI 関数/印刷システムの前処理が行われるため、印刷の開始まで多少時間がかかります

2. 動作環境

GUI ライブラリの動作環境を以下に示します。

対象機種

- DT-9800
- IT-9000

対象 OS

- Microsoft WindowsCE 5.0
- Microsoft WindowsCE 6.0
- Microsoft Windows Mobile 6.5

開発環境

- Microsoft embedded Visual C++ Version 4.0 + SP4
- Microsoft Visual Studio.NET 2003 +SP1
- Microsoft Visual Studio 2005 + SP1
- Microsoft Visual Studio 2008 + SP1

提供ファイル

- Cp780Lib.lib　　インポートライブラリ
- Cp780Lib.dll　　ダイナミックリンクライブラリ
- CpLib.h　　印刷ライブラリヘッダファイル
- CpExtDvm.h　　デバイスモード情報構造体定義ヘッダファイル
- Cp780LibCS.dll　C#用クラスライブラリ
- Cp780LibVB.dll　VB.NET 用クラスライブラリ

使用方法

- C++の場合は、プログラムソース内にCpLib.hをインクルードし、Cp780Lib.libを使用するライブラリとして指定してください
- VB または C#の場合は、Cp780LibCS.dll、Cp780LibVB.dll を実行モジュールと同じフォルダにコピーしてください

制約事項

VB および C#で CpPrintDlg 関数を使用する場合、「プロパティ」ボタンを押してもプリンタプロパティダイアログを表示しません。使用する場合は、フォームに「メニューコントロール」を貼り付けて使用してください。
また、本ライブラリは基本開発キットに搭載している Device Emulator では動作しません。

サポート関数

GDI 関数は、GDI 関数確認一覧に記載している関数をサポートしています。
※ CpCreateDC で作成する DC がパレットに対応していないため、パレットに関する関数は正常に動作しません。

印刷システムを使用したプリンタへの印刷

アプリケーションから印刷ライブラリ関数を使用して印刷制御(印刷データの描画)を行います。
CpCreateDC 関数で作成したデバイスコンテキストに対して、WinGDI 関数を使用して描画処理を行い、印刷を実現します。

3. 機能一覧

3.1 構造体一覧

GUI ライブラリの提供する構造体は、以下のとおりです。

構造体名	概要
CPDOCINFO	CpStartDoc 関数が使用する入力ファイル名と出力ファイル名を格納します
CPDEVMODE	プリンタの初期化と環境データに関する情報を格納します
CPEXTDVM	プリンタの詳細設定に関する情報を格納します
CPCDEVMODE	CPDEVMODE と CPEXTDVM をメンバに持つ構造体です

3.1.1 CPDOCINFO

CPDOCINFO 構造体は、CpStartDoc 関数が使用する入力ファイル名と出力ファイル名を格納します。

```
typedef struct {
   int    cbSize;                  // 構造体のサイズ(バイト単位)
   TCHAR  DocName[MAX_DOCNAME+1];  // 文書の名前(32 文字)
   TCHAR  Output[MAX_OUTPUT+1];    // 未使用(空文字列を指定)
} CPDOCINFO;
```

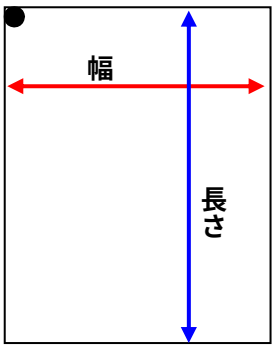
3.1.2 CPDEVMODE

CPDEVMODE 構造体は、プリンタデバイスの初期化と環境データに関する情報を格納します。

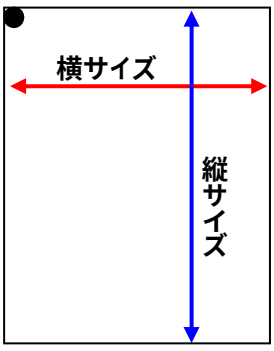
```
typedef struct {
   WORD  dmDriverVersion;  // プリンタドライバのバージョン番号
   WORD  dmSize;           // 構造体のサイズ(バイト単位)
   WORD  dmDriverExtra;    // CPEXTDVM 構造体のサイズ(バイト単位)
   WORD  dmOrientation;    // 用紙方向
                              縦 : CPDMORIENT_PORTRAIT(1)
                              横 : CPDMORIENT_LANDSCAPE(2)
   WORD  dmPaperSize;      // 印刷用紙サイズ(CPDMPAPER_FREE)
   WORD  dmPaperLength;    // 用紙の長さ(0.1mm 単位)
   WORD  dmPaperWidth;     // 用紙の幅(0.1mm 単位)
   WORD  dmCopies;         // 印刷する部数
   WORD  dmPrintQuality;   // プリンタの解像度(1mm 当たりのドット数)
   WORD  dmColor;          // カラー／モノクロ
                              モノクロ : CPDMCOLOR_MONOCHROME(1)
                              カラー   : CPDMCOLOR_COLOR(2)
   WORD  dmBitsPerPel;     // 色解像度(ピクセル当たりのビット数)
   WORD  dmLogmmPixelsX;   // 横方向の 1mm 当たりのドット数
   WORD  dmLogmmPixelsY;   // 縦方向の 1mm 当たりのドット数
   WORD  dmCollate;        // 部単位印刷の指定
                              有効 : 1
                              無効 : 0
   WORD  dmDitherType;     // ディザリングの種類
                              なし     : CPDMDITHER_NON
                              パターン : CPDMDITHER_ PATTERN
                              誤差拡散 : CPDMDITHER_ ERRORDIFFUSION
} CPDEVMODE;
```

用紙サイズと用紙方向との関係

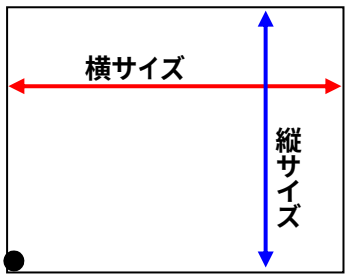
用紙サイズの指定(幅／長さ)は物理サイズの指定になります。用紙方向による論理用紙サイズは以下の通りになります。



用紙サイズ



用紙方向縦

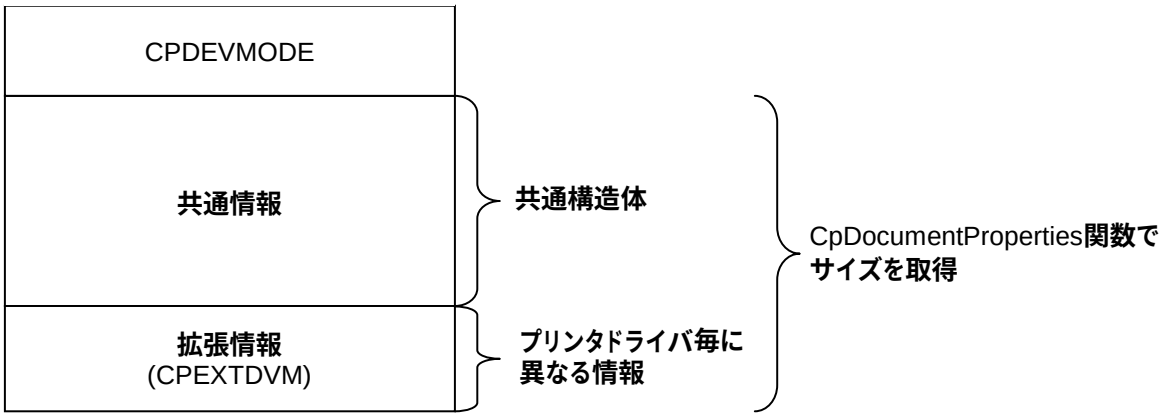


用紙方向横

3.1.3 CPEXTDVM

内蔵プリンタ専用の設定項目の構造体です。
CPDEVMODE の直後に続くプライベートなドライバデータは、プリンタドライバ毎に内容が異なります。

```
typedef struct {
   WORD  stSize;              // 構造体のサイズ(バイト単位)
   WORD  PrintSpeed;          // 印字速度
                                   高速印字          : PRINTSPEED_FAST
                                   低速印字(高品位) : PRINTSPEED_LATE
                                   グラフィック印字 : PRINTSPEED_GRAPHIC
   WORD  PrintDensity;        // 印字濃度(9 段階)
                                   Level1 : DENSITY_LV1
                                            :
                                   Level9 : DENSITY_LV9
   WORD  PaperKind;           // 用紙種類
                                   1P(高感度)   : PKIND_HIGHSENSITIVITY
                                   1P(標準)     : PKIND_STANDARD
                                   1P(長期保存) : PKIND_LONGTERMSTORAGE
                                   ラベル       : PKIND_LABEL
                                   2P           : PKIND_2P   <DT-9800 のみ>
                                   1P(薄紙)     : PKIND_THIN <IT-9000 のみ>
   WORD  bPreHeat;            // プリヒート
                                   有効 : 1
                                   無効 : 0
   WORD  bAutoLoading;        // オートローディング
                                   有効 : 1
                                   無効 : 0
   WORD  AutoLoading;         // オートローディングでのペーパーフィード(1mm 単
  位)
                                   最大 : MAX_AUTOLOADING(96)
                                   最小 : MIN_AUTOLOADING(10)
   WORD  bErrorContinuation;  // エラー時継続印字
                                   有効 : 1
                                   無効 : 0
   WORD  bMarkerDetection;    // マーカ検出(有効時はページ毎に印刷前にマーカ検
  出)
                                   <DT-9800>
                                   有効 : 1
                                   無効 : 0
                                   <IT-9000>
                                   マーカ検出有効(先端) : MARKER_BEGIN
                                   マーカ検出有効(終端) : MARKER_END
                                   マーカ検出無効 : MARKER_DISABLE
} CPEXTDVM
```


CPDEVMODE
共通情報
拡張情報
(CPEXTDVM)
共通構造体
プリンタドライバ毎に
異なる情報
CpDocumentProperties関数で
サイズを取得

3.1.4 CPCDEVMODE

CPDEVMODE 構造体と CPEXTDVM 構造体をメンバに持つ構造体です。
CPEXTDVM 構造体内の設定値を変更したい場合等に使用します。

```
typedef struct {
   CPDEVMODE  dm;   // 標準デバイスモード情報
   CPEXTDVM   dlg;  // 拡張デバイスモード情報
} CPCDEVMODE;
```

3.2 関数一覧

GUI ライブラリの提供する関数は、以下のとおりです。

表 3.1 関数一覧

関数名	概要	DT-9800	IT-9000
CpOpenPrinter	指定しプリンタを識別するハンドルを取得します	○	○
CpClosePrinter	指定したプリンタオブジェクトをクローズします	○	○
CpDocumentProperties	指定したプリンタのプリンタ構成ダイアログボックスを表示します	○	○
CpCreateDC	プリンタのデバイスコンテキスト(DC)を作成します	○	○
CpResetDC	プリンタのデバイスコンテキスト(DC)を更新します	○	○
CpDeleteDC	指定したデバイスコンテキストを削除します	○	○
CpGetDeviceCaps	プリンタの情報を取得します	○	○
CpStartDoc	印刷ジョブを開始します	○	○
CpAbortDoc	印刷ジョブを強制終了します	○	○
CpEndDoc	印刷ジョブを終了します	○	○
CpStartPage	1 ページ分の印刷を開始します	○	○
CpEndPage	1 ページ分の印刷を終了します	○	○
CpUnicodeOut	Unicode 文字列をプリンタ用コード(JIS 内部／Shift JIS)に変換して出力します	○	○
CpExtEscape	アプリケーションが直接プリンタドライバに対してアクセスできるようにします	○	○
CpPrintDlg	[プリンタ選択]ダイアログボックスを作成します	○	○
CpDecodeBarcode	裏面バーコードのデコード結果を返します	-	○
CpSetBarcodeType	裏面バーコードの読取方式を設定します	-	○
CpGetBarcodeType	裏面バーコードの読取方式を取得します	-	○

○ 関数サポート

― 関数未サポート＝ 関数を呼ぶと未サポートエラーが返ります。

3.2.1 CpOpenPrinter

指定したプリンタを識別するハンドルを取得します。

```
[C++]
BOOL CpOpenPrinter (
   LPTSTR    lpszPrinter,
   LPTSTR    lpszPort,
   LPHANDLE  lphPrinter
)
```

```
[Visual Basic]
Declare Function CpOpenPrinter (
   ByVal lpszPrinter As String, _
   ByVal lpszPort    As String, _
   ByRef lphPrinter  As Int32
) As Boolean
```

```
[C#]
Boolean CpOpenPrinter (
   String  lpszPrinter, _
   String  lpszPort, _
   Int32   lphPrinter
)
```

解説

本関数は、指定したプリンタを識別するハンドルを取得します。

パラメータ

lpszPrinter

プリンタ名を指定します。

lpszPort

出力先ポート名を指定します。

lphPrinte

オープンしたプリンタのハンドルが返ります。

戻り値

関数が正常終了した場合は TRUE を、それ以外の場合は FALSE を返します。

3.2.2 CpClosePrinter

指定したプリンタオブジェクトをクローズします。

```
[C++]
BOOL CpClosePrinter (
   HANDLE  hPrinter
)
```

```
[Visual Basic]
Declare Function CpClosePrinter (
   ByVal hPrinter As Int32
) As Boolean
```

```
[C#]
Boolean CpClosePrinter (
   Int32  hPrinter
)
```

解説

本関数は、指定したプリンタオブジェクトをクローズします。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

戻り値

関数が正常終了した場合は TRUE を、それ以外の場合は FALSE を返します。

3.2.3 CpDocumentProperties

指定したプリンタのプリンタ構成ダイアログボックスを表示します。

```
[C++]
LONG CpDocumentProperties (
   HWND         hwnd,
   HANDLE       hPrinter,
   LPCPDEVMODE  lpdmOutput,
   LPCPDEVMODE  lpdmInput,
   DWORD        fMode
)
```

```
[Visual Basic]
Declare Function CpDocumentProperties (
   ByVal hwnd      As Int32, _
   ByVal hPrinter  As Int32, _
   ByRef cdm_out   As Int16, _
   ByRef cdm_in    As Int16, _
   ByVal fMode     As Int32
) As Int32
```

```
[C#]
CpDocumentProperties (
   Int32                          hwnd, _
   Int32                          hPrinter, _
   CalibCs.Cp780LibCS.CPDEVMODE   lpdmOutput, _
   IntPtr                         NullPtr, _
   Int32                          fMode
)

CpDocumentProperties (
   Int32                          hwnd, _
   Int32                          hPrinter, _
   CalibCs.Cp780LibCS.CPDEVMODE   lpdmOutput, _
   CalibCs.Cp780LibCS.CPDEVMODE   lpdmInput, _
   Int32                          fMode
)
```

解説

本関数は、指定したプリンタのプリンタ構成ダイアログボックスを表示します。

パラメータ

hWnd

ダイアログボックスを表示するときの親ウィンドウハンドルを指定します。

hPrinter

オープンしたプリンタのハンドルを指定します。

lpdmOutput

修正したプリンタデータを指定します。

lpdmInput

元のプリンタデータを指定します。

fMode

関数が実行する操作を示す値を指定します。

0 を指定した場合は、CPDEVMODE 構造体に必要なバイト数を返します。

それ以外の場合は、次に示す値を 1 つ以上組み合わせてこのパラメータに指定します。

CPDM_IN_BUFFER	：関数への入力を指定します。
CPDM_IN_PROMPT	：プリンタ設定ダイアログを表示します。
CPDM_OUT_BUFFER	：関数からの出力を指定します。

戻り値

fMode が 0、または lpdmOutput が NULL の場合は、プリンタドライバの初期化データを設定するために必要なバッファのサイズを返します。プリンタドライバが構造体にプライベートなデータを付加している場合は、バッファが CPDEVMODE 構造体よりも大きくなります。

本関数が初期化ダイアログボックスを表示したときは、ユーザが選択したボタンに応じて、IDOK または IDCANCEL を返します。

本関数が正常に終了しなかったときは、0 未満の値を返します。

3.2.4 CpCreateDC

プリンタのデバイスコンテキスト(DC)を作成します。作成した DC に対して、各 GDI 関数が使用できます。

```
[C++]
HDC CpCreateDC (
   HANDLE       hPrinter,
   LPCPDEVMODE  lpInitData
)
```

```
[Visual Basic]
Declare Function CpCreateDC (
   ByVal hPrinter  As Int32, _
   ByRef cdm       As Int16
) As Int32
```

```
[C#]
Int32 CpCreateDC (
   Int32                          hPrinter, _
   CalibCs.Cp780LibCS.CPDEVMODE   lpInitData
)
```

解説

本関数は、プリンタのデバイスコンテキスト(DC)を作成します。
また、作成した DC に対して、各 GDI 関数が使用できます。

パラメータ

hPrinter
オープンしたプリンタのハンドルを指定します。

lpInitData
プリンタデータを指定します。

戻り値

関数が正常終了した場合は、指定したプリンタ用のデバイスコンテキストのハンドルを返します。
それ以外の場合は、NULL を返します。

3.2.5 CpResetDC

プリンタのデバイスコンテキスト(DC)を更新します。
また、1ドキュメント内での用紙方向切り替え、用紙サイズ変更時に使用します。

```
[C++]
HDC CpResetDC (
   HANDLE       hPrinter,
   HDC          hdc,
   LPCPDEVMODE  lpInitData
)
```

```
[Visual Basic]
Declare Function CpResetDC (
   ByVal hPrinter  As Int32, _
   ByVal hdc       As Int32, _
   ByRef cdm       As Int16
) As Int32
```

```
[C#]
Int32 CpResetDC (
   Int32                          hPrinter, _
   Int32                          hdc, _
   CalibCs.Cp780LibCS.CPDEVMODE   lpInitData
)
```

解説

本関数は、プリンタのデバイスコンテキスト(DC)を更新します。
また、1ドキュメント内での用紙方向切り替え、用紙サイズ変更時に使用します。

パラメータ

hPrinter
オープンしたプリンタのハンドルを指定します。

hdc
デバイスコンテキストを指定します。

lpInitData
プリンタデータを指定します。

戻り値

関数が正常終了した場合は、元のデバイスコンテキストのハンドルと元のプリンタ制御情報を返します。それ以外の場合は、NULL を返します。

3.2.6 CpDeleteDC

指定したデバイスコンテキストを削除します。

```
[C++]
BOOL CpDeleteDC (
   HANDLE  hPrinter,
   HDC     hdc
)
```

```
[Visual Basic]
Declare Function CpDeleteDC (
   ByVal hPrinter  As Int32, _
   ByVal hDC       As Int32
) As Boolean
```

```
[C#]
Boolean CpDeleteDC (
   Int32  hPrinter, _
   Int32  hdc
)
```

解説

本関数は、指定したデバイスコンテキストを削除します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

戻り値

関数が正常終了した場合は TRUE を、それ以外の場合は FALSE を返します。

3.2.7 CpGetDeviceCaps

プリンタの情報を取得します。

```
[C++]
int CpGetDeviceCaps (
   HANDLE  hPrinter,
   HDC     hdc,
   int     nIndex
)
```

```
[Visual Basic]
Declare Function CpGetDeviceCaps (
   ByVal hPrinter  As Int32, _
   ByVal hDC       As Int32
   ByVal nIndex    As Int32
) As Boolean
```

```
[C#]
Int32 CpGetDeviceCaps (
   Int32  hPrinter, _
   Int32  hdc, _
   Int32  nIndex
)
```

解説

本関数は、プリンタの情報を取得します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

nIndex

以下の機能取得番号を指定します。

CP_HORZRES	: 印刷可能領域幅(ピクセル単位)
CP_VERTRES	: 印刷可能領域長さ(ピクセル単位)
CP_HORZSIZE	: 印刷可能領域幅(0.1mm 単位)
CP_VERTSIZE	: 印刷可能領域長さ(0.1mm 単位)
CP_LOGPIXELSX	: 水平方向のインチ当たりのピクセル数
CP_LOGPIXELSY	: 垂直方向のインチ当たりのピクセル数
CP_LOGMMPIXELSX	: 水平方向の 1mm 当たりのピクセル数
CP_LOGMMPIXELSY	: 垂直方向の 1mm 当たりのピクセル数

CP_PRINTER　　：ページ制御プリンタ/ライン制御プリンタ
CP_TEXTCAPS　　：CpUnicodeOut 描画の出力方式

戻り値

指定した機能取得番号の項目の値を返します。

3.2.8 CpStartDoc

印刷ジョブを開始します。

```
[C++]
int CpStartDoc (
   HANDLE       hPrinter,
   HDC          hdc,
   LPCPDOCINFO  lpdi
)
```

```
[Visual Basic]
Declare Function CpStartDoc (
   ByVal hPrinter  As Int32, _
   ByVal hDC       As Int32
   ByRef docinfo   As Int32
) As Boolean
```

```
[C#]
Int32 CpStartDoc (
   Int32                          hPrinter, _
   Int32                          hdc, _
   CalibCs.Cp780LibCS.CPDOCINFO   lpdi
)
```

解説

本関数は、印刷ジョブを開始します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

ldpi

文書名と出力ファイル名を指定します。

lpdi->lpszDocName　　：印刷中ダイアログに表示される文書名を指定します
lpdi->lpszOutput　　：空文字(_T(""))を指定してください

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

3.2.9 CpAbortDoc

印刷ジョブを強制終了します。

```
[C++]
int CpAbortDoc (
   HANDLE  hPrinter,
   HDC     hdc
)
```

```
[Visual Basic]
Declare Function CpAbortDoc (
   ByVal hPrinter  As Int32, _
   ByVal hDC       As Int32
) As Boolean
```

```
[C#]
Int32 CpAbortDoc (
   Int32  hPrinter, _
   Int32  hdc
)
```

解説

本関数は、印刷ジョブを強制終了します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

3.2.10 CpEndDoc

印刷ジョブを終了します。

```
[C++]
int CpEndDoc (
   HANDLE  hPrinter,
   HDC     hdc
)
```

```
[Visual Basic]
Declare Function CpEndDoc (
   ByVal hPrinter  As Int32, _
   ByVal hDC       As Int32
) As Boolean
```

```
[C#]
Int32 CpEndDoc (
   Int32  hPrinter, _
   Int32  hdc
)
```

解説

本関数は、印刷ジョブを終了します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

3.2.11 CpStartPage

1ページ分の印刷を開始します。

```
[C++]
int CpStartPage (
   HANDLE  hPrinter,
   HDC     hdc
)
```

```
[Visual Basic]
Declare Function CpStartPage (
   ByVal hPrinter  As Int32, _
   ByVal hDC       As Int32
) As Boolean
```

```
[C#]
Int32 CpStartDoc (
   Int32  hPrinter, _
   Int32  hdc
)
```

解説

本関数は、1ページ分の印刷を開始します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

3.2.12 CpEndPage

1ページ分の印刷を終了します。

```
[C++]
int CpEndPage (
   HANDLE  hPrinter,
   HDC     hdc
)
```

```
[Visual Basic]
Declare Function CpEndPage (
   ByVal hPrinter  As Int32, _
   ByVal hDC       As Int32
) As Boolean
```

```
[C#]
Int32 CpEndPage (
   Int32  hPrinter, _
   Int32  hdc
)
```

解説

本関数は、1ページ分の印刷を終了します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

3.2.13 CpUnicodeOut

Unicode 文字列をプリンタ用コード(JIS 内部／Shift-JIS)に変換して出力します。

```
[C++]
int CpUnicodeOut (
   HANDLE   hPrinter,
   HDC      hdc,
   LPCTSTR  lpszString,
   UINT     cbCount
)
```

```
[Visual Basic]
Declare Function CpUnicodeOut(
   ByVal hPrinter    As Integer, _
   ByVal hdc         As Integer, _
   ByVal lpszString  As String, _
   ByVal cbCount     As Integer
) As Integer
```

```
[C#]
Int32 CpUnicodeOut (
   Int32   hPrinter, _
   Int32   hdc, _
   String  lpszString, _
   Int32   cbCount
)
```

解説

本関数は、Unicode 文字列をプリンタ用コード(JIS 内部／Shift-JIS)に変換して出力します。

パラメータ

hPrinter

オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

lpszString

文字列のアドレスを指定します。

cbCount

文字数を指定します。

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

補足

本関数は、プリンタコマンドを出力しません。そのため、ESC コマンドでの各種設定は CpExtEscape関数で出力する必要があります。

3.2.14 CpExtEscape

制御コマンドを使用して直接印刷します。

```
[C++]
int CpExtEscape (
   HANDLE  hPrinter,
   HDC     hdc,
   int     nEscape,
   int     cbInput,
   LPCSTR  lpvInData,
   int     cbOutput,
   LPVOID  lpvOutData
)
```

```
[Visual Basic]
Declare Function CpExtEscape(
   ByVal hPrinter    As Int32, _
   ByVal hdc         As Int32, _
   ByVal nEscape     As Int32, _
   ByVal cbInput     As Int32, _
   ByRef lpvInData   As Byte, _
   ByVal cbOutput    As Int32, _
   ByRef lpvOutData  As Byte
) As Int32
```

```
[C#]
Int32 CpExtEscape (
   Int32  hPrinter, _
   Int32  hdc, _
   Int32  nEscape, _
   Int32  cbInput, _
   Byte   lpvInData, _
   Int32  cbOutput, _
   Byte   lpvOutData
)
```

解説

本関数は、アプリケーションが直接プリンタドライバに対してアクセスできるようにします。
文字、バーコードのプリンタコマンドを印字位置等の制御コマンドを使用して直接印刷できます。
ただし、プリンタ毎にプリンタコマンドが異なりますので注意が必要です。

パラメータ

hPrinter
オープンしたプリンタのハンドルを指定します。

hdc

デバイスコンテキストを指定します。

nEscape

CPPASSTHROGH を指定してください。

cbInput

lpvInData のバイト数を指定します。

lpvInData

指定したエスケープに必要なデータを指定します。

cbOutput

lpvOutData のバイト数を指定します。

lpvOutData

指定したエスケープからの出力を受け取るデータを指定します。

戻り値

関数が正常終了した場合は正の値を、エスケープが実装されていない場合は、0 を返します。
それ以外の場合は CP_ERROR を返します。

3.2.15 CpPrintDlg

［プリンタ選択］ダイアログボックスを表示します。

```
[C++]
int CpPrintDlg (
   HWND         hwnd,
   LPHANDLE     lphPrinter,
   LPCPDEVMODE  lpdmOutput,
   LPTSTR       lpszPrinter,
   LPTSTR       lpszPort
)
```

```
[Visual Basic]
Declare Function CpPrintDlg(
   ByVal hwnd         As Int32, _
   ByRef lphPrinter   As Int32, _
   ByVal lpdmOutput   As Cp780LibVB.CPDEVMODE, _
   ByVal lpszPrinter  As String, _
   ByVal lpszPort     As String
) As Int32
```

```
[C#]
Int32 CpPrintDlg (
   Int32                          hwnd, _
   Int32                          lphPrinter, _
   CalibCs.Cp780LibCS.CPDEVMODE   lpdmOutput, _
   String                         lpszPrinter, _
   String                         lpszPort
)
```

解説

本関数は、［プリンタ選択］ダイアログボックスを表示します。
ユーザは［プリンタ選択］ダイアログボックスを使用して、”プリンタの選択”、”ポートの選択”とプリンタのプロパティを設定することができます。

パラメータ

hwnd

ダイアログボックスを表示する親ウィンドウハンドルを指定します。

lphPrinter

オープンしたプリンタのハンドルを指定します。

lpdmOutput

CPDEVMODE構造体を指定します。

lpszPrinter

出力するプリンタ名を指定します。

lpszPort

出力するポート名を指定します。

戻り値

lpdmOutput が NULL の場合は、プリンタドライバの初期化データを設定するために必要なバッファのサイズを返します。プリンタドライバが構造体にプライベートなデータを付加している場合は、バッファが CPDEVMODE 構造体よりも大きくなります。

関数がダイアログボックスを表示した場合は、ユーザが選択したボタンに応じて、IDOK、または IDCANCEL を返します。

関数が正常に終了しなかったときは、0 未満の値を返します。

3.2.16 CpDecodeBarcode

裏面バーコードのデコード結果を返します。

```
[C++]
int CpDecodeBarcode(
   HWND   hPrinter,
   DWORD  *pdwLength,
   TCHAR  *pszData
)
```

```
[Visual Basic]
Declare Function CpDecodeBarcode(
   ByVal hPrinter   As Int32, _
   ByRef pdwLength  As String, _
   ByRef pszData    As String
) As Int32
```

```
[C#]
Int32 CpDecodeBarcode(
   Int32      hPrinter,
   ref Int32  pdwLength,
   out string pszData
)
```

解説

本関数は、印刷中に読み取った裏面バーコードのデコード結果を返します。

パラメータ

hPrinter

プリンタのハンドルを指定します。

pdwLength

裏面バーコードのデコード結果の長さを取得します。

pszData

裏面バーコードのデコード結果を取得します。

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

注意

裏面バーコードのデコード結果は、CpEndPage関数実行後にスキャンしたデータを使用します。

CpEndPage関数実行前に、CpSetBarcodeType関数により裏面バーコードの読取方式を設定してください。

3.2.17 CpSetBarcodeType

裏面バーコードの読取方式を設定します。

```
[C++]
int CpSetBarcodeType(
   HWND   hPrinter,
   DWORD  dwType
)
```

```
[Visual Basic]
Declare Function CpSetBarcodeType(
   ByVal hPrinter  As Int32, _
   ByVal dwType    As Int32 _
) As Int32
```

```
[C#]
Int32 CpSetBarcodeType(
   Int32  hPrinter,
   Int32  dwType,
)
```

解説

本関数は、裏面バーコードの読取方式を設定します。

パラメータ

hPrinter

プリンタのハンドルを指定します。

dwType

裏面バーコードの読取に使用する方式を指定します。

0　　：無効（デフォルト）
1　　：6 桁読取
2　　：15 桁読取

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

3.2.18 CpGetBarcodeType

裏面バーコードの読取方式を取得します。

```
[C++]
int CpGetBarcodeType(
   HWND   hPrinter,
   DWORD  *pdwType
)
```

```
[Visual Basic]
Declare Function CpGetBarcodeType(
   ByVal hPrinter  As Int32, _
   ByRef pdwType   As Int32 _
) As Int32
```

```
[C#]
Int32 CpGetBarcodeType(
   Int32      hPrinter,
   ref Int32  pdwType,
)
```

解説

本関数は、裏面バーコードの読取方式を取得します。

パラメータ

hPrinter

プリンタのハンドルを指定します。

pdwType

裏面バーコードの読取に使用する方式を取得します。取得する値の詳細については、CpSetBarcodeType関数を参照してください。

戻り値

関数が正常終了した場合は正の値を、それ以外の場合は CP_ERROR を返します。

3.3 プログラミング上の注意点

3.3.1 GDI 関数確認一覧

CpCreateDC で作成した DC に対して描画等を行い、確認条件下において正常に動作する関数の一覧を示します。印刷システムにおいては、以下の GDI 関数を使用してください。

AddFontResource	ExtTextOut	OffsetRgn
BitBlt	FillRect	PatBlt
CombineRgn	FillRgn	Polygon
CreateBitmap	GetBkColor	Polyline
CreateCompatibleBitmap	GetBkMode	PtInRegion
CreateCompatibleDC	GetCharWidth32	Rectangle
CreateDC	GetClipBox	RectInRegion
CreateDIBPatternBrushPt	GetClipRgn	RectVisible
CreateDIBSection	GetCurrentObject	RestoreDC
CreateFontIndirect	GetCurrentPositionEx	RoundRect
CreatePatternBrush	GetDeviceCaps	SaveDC
CreatePen	GetDIBColorTable	SelectClipRgn
CreatePenIndirect	GetNearestColor	SelectObject
CreateRectRgn	GetObject	SetBitmapBits
CreateRectRgnIndirect	GetObjectType	SetBkColor
CreateSolidBrush	GetPixel	SetBkMode
DeleteDC	GetRegionData	SetBrushOrgEx
DeleteObject	GetRgnBox	SetDIBColorTable
DrawEdge	GetStockObject	SetDIBitsToDevice
DrawFocusRect	GetSysColorBrush	SetPixel
DrawText	GetTextAlign	SetRectRgn
Ellipse	GetTextColor	SetROP2
EnableEUDC	GetTextExtentExPoint	SetTextAlign
EnumFontFamilies	GetTextFace	SetTextColor
EnumFonts	GetTextMetrics	SetViewportOrgEx
EqualRgn	IntersectClipRect	StretchBlt
ExcludeClipRect	LineTo	StretchDIBits
ExtCreateRerion	MaskBlt	TranslateCharsetInfo
ExtEscape	MoveToEx	TransparentImage

3.3.2 プリンタ名と出力ポート名へについて

プリンタ名と対応する出力ポート名を以下に示します。

	プリンタ名	出力ポート名
内蔵プリンタ(日本語版)	BuiltIn	BuiltIn
内蔵プリンタ(英語版)	BuiltIn	BuiltIn

3.3.3 用紙サイズについて

プリンタの指定可能用紙サイズを以下に示します。
※ このサイズはデバイスコンテキストに作成する描画イメージのサイズです

	幅[0.1mm]	長さ[0.1mm]
内蔵プリンタ	480 or 720	20 ～ 9999

3.3.4 ランゲージモニタについて

スプーラからの制御により、プリンタポートとの送受信を行います。
また、データ出力時の状態監視、エラー制御も行います。

インターフェースについて

印刷を開始すると、タスクトレイにプリンタアイコンを出力し、アイコンをダブルクリックすると印刷中を表すダイアログボックスを出力します。
印刷が終了すると、ダイアログボックスを出力している場合はダイアログボックスを閉じ、アイコンを出力している場合はアイコンを削除します。
また、印刷中ダイアログボックスは以下の動作が実行可能です。

キャンセルボタン押下　　印刷を中断します

印刷中にエラーが発生すると、エラーを表すダイアログボックスを出力します。エラーダイアログが出力された際は、以下の動作の実行を選択します。

継続ボタン押下　　引き続き印刷を実行します
中止ボタン押下　　印刷を中断します

各種エラーについて

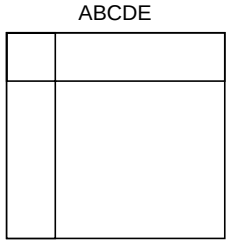
エラー表示	エラー内容	対処方法
用紙がありません	用紙切れ、用紙なし	用紙をセットしてください
プラテンがオープンしています	プラテンが開いている	プラテンを閉じてください
VDETP が発生しました	ローバッテリー状態	バッテリーを充電、交換してください
サスペンドが発生しました	サスペンド(印字中断)状態	エラー時継続印字を有効にして印刷してください
ヘッド温度エラー	ヘッド温度上昇	ヘッド温度が下がるまで印字しないでください
マーカー検出エラー	マーカーが検出できない	印刷を中止し、マーカー検出を無効にして再度印刷してください マーカー用紙で再度印刷してください
80mm へ変更できません	用紙サイズ変更エラー	印刷を中止し、用紙幅の設定を 48.0mm に設定し、58mm 用紙で印刷してください

3.3.5 プリンタへの直接印刷について

CpUnicodeOut 関数の直接プリンタ印刷サポート場合、または、CpExtEscape 関数でプリンタへ直接印刷を行う場合の印刷結果は以下のようになります。

アプリケーションの処理

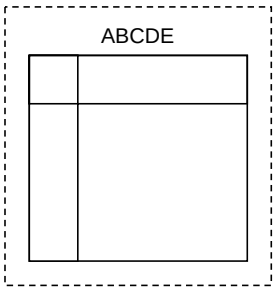
1. CpUnicodeOut 関数でテキストを描画
2. プリンタのデバイスコンテキストに対して描画



プリンタの処理

1. CpUnicodeOut 関数の内容を描画
2. デバイスコンテキストの内容を描画

   ライン単位で印刷するため、CpUnicodeOut 関数で直接プリンタへ送信すると、送信した時点で印刷データを作成します。そのため、その後のデータと重ねることはできません。



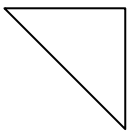
4. サンプルソースコード

4.1 イメージおよび文字の印刷

本サンプルプログラムではライブラリの関数を使用して、GDI 関数である"LineTo"によるイメージを 1 ページ目に印刷し、用紙サイズを変更して文字列を 2 ページ目に印刷します。

```
TCHAR szUni[]   = _T("UnicodeOut  テスト");
char  szESC_E[] = "¥x1b""E";

void Sample(HWND  hWnd) // hWnd   アプリケーションのウィンドウハンドル
{
   HANDLE       hPrinter;
   HDC          hDC, hDCOld;
   LPCPDEVMODE  pInCDM = NULL;  // プリンタデータ入力値
   LPCPDEVMODE  pCDM   = NULL;  // プリンタデータ領域
   CPDOCINFO    doc;
   int fMode = CPDM_OUT_BUFFER; // プリンタデータ出力要求
   int size;

   /***** CpOpenPrinter:指定プリンタをオープンしてハンドルを取得します  *****/
   // プリンタ名：BuiltIn(内蔵プリンタ) 出力ポート名：BuiltIn(内蔵プリンタ)
   if (CpOpenPrinter(_T("BuiltIn"),  _T("BuiltIn"),  &hPrinter)  !=  TRUE)
      return;

   /***** CpDocumentProperties：プリンタデータを取得します  *****/
   // プリンタデータ保存用領域として必要なサイズを取得
   size = CpDocumentProperties(hWnd,  hPrinter,  pCDM, pInCDM,  fMode);
   if (size  < 0) {
      CpClosePrinter(hPrinter);
      return;
   }

   // 取得された必要サイズ分の領域を確保
   pCDM  = (LPCPDEVMODE)malloc(size);

   // CPDEVMODE のデフォルト情報を取得
   if (CpDocumentProperties(hWnd,  hPrinter,  pCDM,  pInCDM,  fMode)  < 0){
      CpClosePrinter(hPrinter);
      free(pCDM);
      return;
   }

   /***** 描画サイズ(長さ)を変更     *****/
   // 描画サイズ(長さ)を 200(20[mm])に変更
   pCDM->dmPaperLength  = 200;

   // CpDocumentProperties の引数 fMode の CPDM_IN_PROMPT プリンタプロパティに
```

```
よる
// 値の変更以外にも上記コードのように値を直接指定することも可能です

/***** プリンタ毎の拡張情報を変更したい場合 *****/
// プリンタ拡張情報定義のヘッダファイル(CpExtDvm.h)のインクルードが必要です
// 内蔵プリンタの【印字速度】を「高速印字」に変更する場合
// ((LPCPCDEVMODE)pCDM)->dlg.PrintSpeed  = PRINTSPEED_FAST;

/***** CpCreateDC：描画用のデバイスコンテキストハンドルを取得します  *****/
hDC = CpCreateDC(hPrinter,  pCDM);
if (hDC  == NULL) {
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpSartDoc：印刷ドキュメントの開始設定を行います  *****/
// CPDOCINFO  設定
doc.cbSize  = sizeof(CPDOCINFO);
wcscpy((LPWSTR)doc.DocName,  (LPWSTR)_T("ドキュメント名"));
wcscpy((LPWSTR)doc.Output,  (LPWSTR)_T(""));

if (CpStartDoc(hPrinter,  hDC,  &doc)  == CP_ERROR) {
   CpDeleteDC(hPrinter,  hDC); CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpStartPage：ページの開始設定を行います  *****/
if (CpStartPage(hPrinter,  hDC) == CP_ERROR)
{
   CpEndDoc(hPrinter,  hDC);
   CpDeleteDC(hPrinter,  hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** イメージ描画：デバイスコンテキストハンドルに以下のイメージを描画して
います *****/

LineTo(hDC,  100,  100); //
LineTo(hDC,  100,  0);   //
LineTo(hDC,    0,  0);   //
```



```
/***** CpEndPage：ページの終了設定を行います  *****/
if (CpEndPage(hPrinter,  hDC) == CP_ERROR)  {
   CpEndDoc(hPrinter,  hDC);
   CpDeleteDC(hPrinter,  hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}
```

```
/***** CpResetDC：用紙サイズを変更します　*****/
hDCOld  = hDC;
pCDM->dmPaperLength  = 100;  // 描画サイズ(長さ)を 200 から 100(10[mm])に変
更
hDC  = CpResetDC(hPrinter,  hDC, pCDM);
if (hDC  == NULL) {
   CpEndDoc(hPrinter,  hDCOld);
   CpDeleteDC(hPrinter,  hDCOld);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpStartPage：ページの開始設定を行います　*****/
if (CpStartPage(hPrinter,  hDC) == CP_ERROR) {
   CpEndDoc(hPrinter,  hDC);
   CpDeleteDC(hPrinter,  hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpUnicodeOut：文字列を送ります　*****/
if (CpUnicodeOut(hPrinter,  hDC,  szUni,  lstrlen(szUni))  == CP_ERROR) {
   CpEndPage(hPrinter,  hDC);
   CpEndDoc(hPrinter,  hDC);
   CpDeleteDC(hPrinter,  hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpExtEscape：[ESC]E　(未印字データの吐き出し)　コマンドを送ります
*****/
if (CpExtEscape(hPrinter,  hDC, CPPASSTHROGH,  (int)sizeof(szESC_E),
(LPCSTR)szESC_E,  0, NULL)  == CP_ERROR) {
   CpEndPage(hPrinter,  hDC);
   CpEndDoc(hPrinter,  hDC);
   CpDeleteDC(hPrinter,  hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpEndPage：ページの終了設定を行います　*****/
if (CpEndPage(hPrinter,  hDC) == CP_ERROR) {
   CpEndDoc(hPrinter,  hDC);
   CpDeleteDC(hPrinter,  hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}
```

```
   /***** CpEndDoc：ドキュメントの終了設定を行います  *****/
   // CpEndDoc 実行により、印刷がスタートします。
   if (CpEndDoc(hPrinter,  hDC) == CP_ERROR) {
      CpDeleteDC(hPrinter,  hDC);
      CpClosePrinter(hPrinter);
      free(pCDM);
      return;
   }

   /***** CpDeleteDC：デバイスコンテキストを削除します  *****/
   if (CpDeleteDC(hPrinter,  hDC)  != TRUE) {
      CpClosePrinter(hPrinter);
      free(pCDM);
      return;
   }

   /***** CpClosePrinter：指定されたプリンタを閉じます  *****/
   if (CpClosePrinter(hPrinter)  !=  TRUE) {
      free(pCDM);
      return;
   }

   // プリンタデータ保存領域を解放
   if(pCDM)
      free(pCDM);
}
```

4.2 裏面バーコードの読み取り

本サンプルプログラムではライブラリの関数を使用して、GDI 関数である"LineTo"によるイメージを印刷し、裏面バーコードの読み取りを行います。

裏面バーコードを読み取る場合の注意点

- ヘッダファイルは Cp780Lib.h と CpExtDvm.h の両方をインクルードしてください。
- 裏面バーコードの読み取り設定関数 CpSetBarcodeType を CpStartDoc 関数実行前に呼び、バーコードタイプを設定してください。
- 印刷の描画サイズ(長さ)を 42mm 以上に設定してください。
- テキストのみを印字する場合は、印刷長が 42mm 以上となるように、改行を入れてください。
- デフォルトの文字サイズ(縦 2mm)･改行ピッチ(1mm)の場合、(2+1)×14 行=42mm。
- マーカ検出設定に「マーカ検出(終端)」を設定してください。
- バーコード読み取りデータ取得関数 CpDecodeBarcode は、印刷が完了してから実行してください。印刷完了のタイミングはメッセージ CP_WM_PRINTDONE により通知されます。

```
void Sample(HWND hWnd)      // hWnd  アプリケーションのウィンドウハンドル
{
   HANDLE hPrinter;
   HDC hDC;
   LPCPDEVMODE pInCDM = NULL;  // プリンタデータ入力値
   LPCPDEVMODE pCDM = NULL; // プリンタデータ領域
   CPDOCINFO doc;

   int fMode = CPDM_OUT_BUFFER;// プリンタデータ出力要求
   int size;
   DWORD dwBarType;

   /***** CpOpenPrinter：指定プリンタをオープンしてハンドルを取得します *****/
   // プリンタ名：BuiltIn(内蔵プリンタ)  出力ポート名：BuiltIn(内蔵プリンタ)
   if (CpOpenPrinter(_T("BuiltIn"), _T("BuiltIn"), &hPrinter) != TRUE)
      return;

   /***** CpSetBarcodeType：裏面バーコードの読み取り方式を指定します *****/
   //dwBarType = 1;     // 裏面バーコードタイプ 1
   dwBarType = 2;       // 裏面バーコードタイプ 2
   CpSetBarcodeType( hPrinter, dwBarType);

   /***** CpDocumentProperties：プリンタデータを取得します *****/
   // プリンタデータ保存用領域として必要なサイズを取得
   size = CpDocumentProperties(hWnd, hPrinter, pCDM, pInCDM, fMode);
   if (size < 0)
   {
      CpClosePrinter(hPrinter);
      return;
   }
```

```
// 取得された必要サイズ分の領域を確保
pCDM = (LPCPDEVMODE)malloc(size);
// CPDEVMODE のデフォルト情報を取得

if (CpDocumentProperties(hWnd, hPrinter, pCDM, pInCDM, fMode) < 0)
{
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** 描画サイズ(長さ)を変更 *****/
// 裏面バーコードを読み取る場合は、描画サイズ(長さ)を 420(42[mm])以上に設定
してください
pCDM->dmPaperLength = 420;

/**** 裏面バーコード読み取り時は「マーカ検出(終端)」を設定してください ****/
((LPCPCDEVMODE)pCDM)->dlg.bMarkerDetection = MARKER_END;

/***** CpCreateDC：描画用のデバイスコンテキストハンドルを取得します *****/
hDC = CpCreateDC(hPrinter, pCDM);
if (hDC == NULL)
{
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}
/***** CpSartDoc：印刷ドキュメントの開始設定を行います *****/
// CPDOCINFO 設定
doc.cbSize = sizeof(CPDOCINFO);
wcscpy((LPWSTR)doc.DocName, (LPWSTR)_T("ドキュメント名"));
wcscpy((LPWSTR)doc.Output, (LPWSTR)_T(""));

if (CpStartDoc(hPrinter, hDC, &doc) == CP_ERROR)
{
   CpDeleteDC(hPrinter, hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpStartPage：ページの開始設定を行います *****/
if (CpStartPage(hPrinter, hDC) == CP_ERROR)
{
   CpEndDoc(hPrinter, hDC);
   CpDeleteDC(hPrinter, hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** イメージ描画：デバイスコンテキストハンドルに右のイメージを描画してい
```

```
ます *****/

LineTo(hDC, 100, 100);          //
LineTo(hDC, 100, 0);            //
LineTo(hDC, 0, 0);              //

/***** CpEndPage：ページの終了設定を行います *****/
if (CpEndPage(hPrinter, hDC) == CP_ERROR)
{
   CpEndDoc(hPrinter, hDC);
   CpDeleteDC(hPrinter, hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpEndDoc：ドキュメントの終了設定を行います *****/
// CpEndDoc 実行により、印刷がスタートします。
if (CpEndDoc(hPrinter, hDC) == CP_ERROR)
{
   CpDeleteDC(hPrinter, hDC);
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** CpDeleteDC：デバイスコンテキストを削除します *****/
if (CpDeleteDC(hPrinter, hDC) != TRUE)
{
   CpClosePrinter(hPrinter);
   free(pCDM);
   return;
}

/***** 印字完了を待って裏面バーコード読み取り結果を取得する *****/
// 印字完了メッセージ CP_WM_PRINTDONE を受け取った後、
// 裏面バーコード読み取り結果を取得します
// 以下の処理は通常はウィンドウプロシージャ内で行ってください
DWORD tick = GetTickCount();
while(( GetTickCount() - tick) < 10000)
{
   MSG winmsg;
   if ( PeekMessage( &winmsg, hWnd, CP_WM_PRINTDONE, CP_WM_PRINTDONE,
      PM_REMOVE))
   {
      DWORD len;
      TCHAR szbarcode[ 32], szmes[ 64];
      len = 0;
      ZeroMemory( szbarcode, sizeof( szbarcode));
      if ( CpDecodeBarcode( hPrinter, &len, szbarcode) != CP_ERROR)
      {
         wsprintf( szmes, L"len=%u¥r¥n%s", len, szbarcode);
         MessageBox( hWnd, szmes, L"Barcode", MB_OK);
```

```
            }
            break;
        }
        Sleep( 100);
    }

    /***** CpClosePrinter：指定されたプリンタを閉じます *****/
    if (CpClosePrinter(hPrinter) != TRUE)
    {
        free(pCDM);
        return;
    }

    // プリンタデータ保存領域を解放
    if(pCDM)
    {
        free(pCDM);
    }
}
```

カシオ計算機お問い合わせ窓口

製品に関する最新情報

●製品サポートサイト（カシオペア・ハンディターミナル）

http://casio.jp/support/ht/

製品の取扱い方法のお問い合わせ

●情報機器コールセンター

0570-022066
市内通話料金でご利用いただけます。
携帯電話・PHS 等をご利用の場合、**048-233-7241**

カシオ計算機株式会社
〒151-8543　東京都渋谷区本町 1-6-2
TEL 03-5334-4638（代）