.NET Framework 用 バーコード作成ツール

BarCode.NET
説明書

Version 1.9.0

平成１９年２月

Pao@Office

はじめに

BarCode.NET は、Microsoft .NET Framework 上で動作する、バーコード作成ツール(クラス群)の総称です。

BarCode.NET は、次のことを念頭において開発いたしました。

１．精密なこと

単なるバーコードリーダでの検査でなく、RJS のレーザーインスペクター Model L2000 というバーコード検査機にて細かくバーコードの精度を検査しております。それにより、従来の他社のバーコード作成ツールに比べても精密なバーコードを作成することが可能です。

バーコード全体の幅を指定する方法以外にも、バーの最小幅を指定することにより、縮小することなく直接バーコードを描画し、より精度の高いバーコードを作成することが可能です。

コンビニエンスストア向けの標準料金代理収納用の EAN128 バーコード描画メソッドでは、バーコードの幅を指定するのではなく、プリンターの解像度(dpi)に合わせたドット単位での印刷を可能としました。このドット単位での印刷は、「財団法人 流通システム開発センター」の「標準料金代理収納ガイドライン」に準拠したものです。

２．用途が様々

皆様が作成されるアプリケーションから Graphics オブジェクトを渡していただいて、BarCode.NET が、その Graphics オブジェクトに対してバーコードを描画する仕組みになっておりますので、様々な用途に利用することが可能になっております。

３．使いやすいこと

わかりやすいクラスのインタフェイスになっております。

後の使用例でも書かれておりますが、2～3 Step のロジックでバーコードの印刷等を行うことができます。

４．軽いこと

何と言っても軽さが命です。BarCode.NET を利用してバーコード作成を行う場合、BarCode.NET 自体がシステムに与える負荷は微小です。ほんの数ＭＢのメモリ上で動作します。

BarCode.NET をご利用していただく皆さんが、.NET 環境でのバーコードの生成(印刷)プログラムの作成作業に、楽しさを感じていただければ幸いです。

平成１９年２月　作者

目　　次

１．BarCode.NET の動作環境・インストール方法

1-1.　動作環境

OS	Microsoft.NET Framework が正常に動作するものである事
動作に必要なメモリ	Microsoft .NET Framework が正常に動作するために必要な容量
画面解像度	特に制限なし
開発環境	Microsoft Visual Studio .NET 2002 以上 がインストールされている事

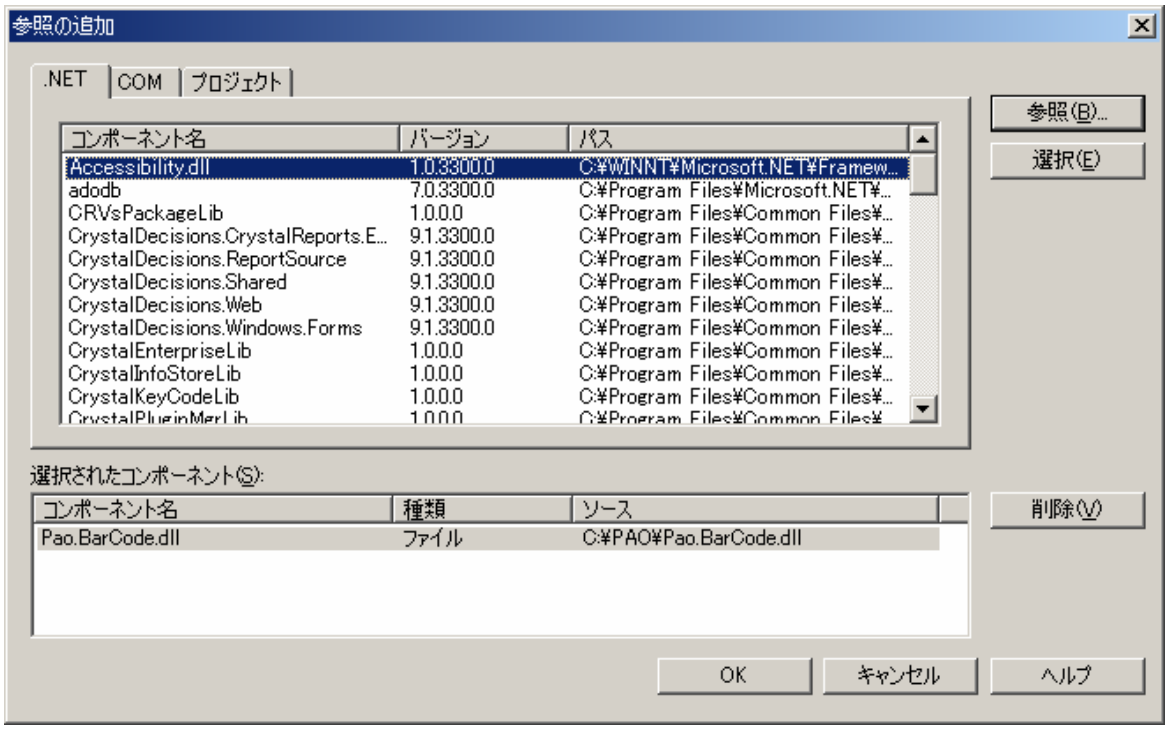
1-2. インストール方法

Pao.BarCode.dll をコピーするだけです。

http://www.pao.ac/products/barcode.net/

より、BarCode.NET(バージョン).zip をダウンロードして解凍後、任意のフォルダにコピーしてください。

BarCode.NET を使用する場合、.NET のプロジェクトに参照の追加をしてください。



２．BarCode.NET の機能

2-1. 機能概要

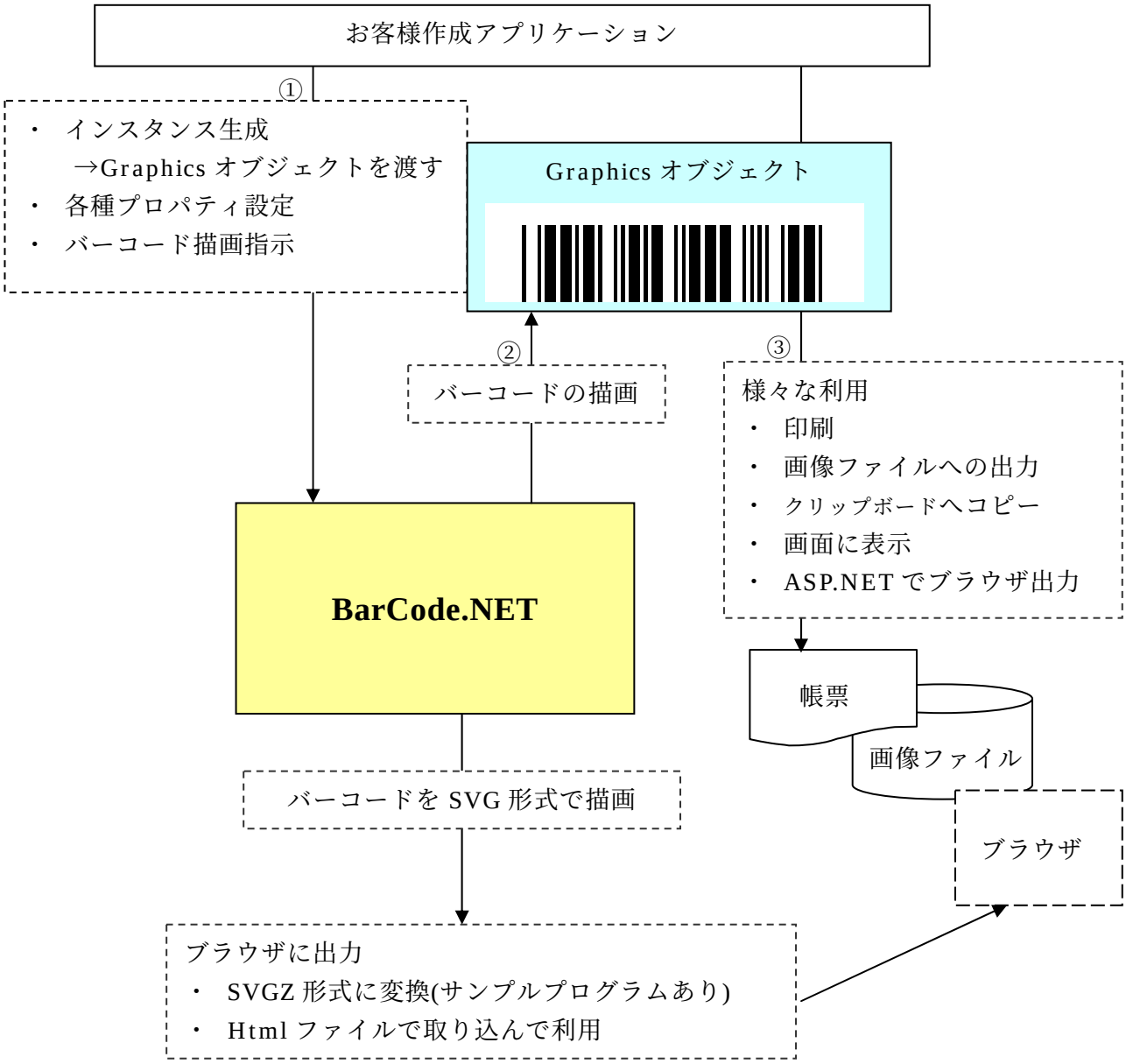
BarCode. NET は、以下 10 種類のバーコードの作成が可能です。

(1) JAN13(EAN13)
(2) JAN8(EAN8)
(3) ITF(インターリーブド 2 of 5)
(4) Matrix 2 of 5
(5) NEC 2 of 5
(6) NW7(Codebar)
(7) Code39
(8) Code128
(9) UCC/EAN128(コンビニ向け標準料金代理収納用含む)
(10) 郵便カスタマバーコード
(11) QR コード

※郵便カスタマバーコード以外は、以降総称して「一次元バーコード」と呼びます。

BarCode. NET では、上記の各バーコードを作成するために、バーコードの種類ごとに全て別々のクラスとして利用することが可能となっております。
BarCode. NET の各バーコード作成クラスは、クラスのコンストラクタで.NET の System.Drawing.Graphics オブジェクトを受け取り、Graphics オブジェクトに対してバーコードを描画します。
また、Ver1.7.1 より、SVG 形式のファイルへの出力が可能となりました。

また、Ver 1.8.0 より、コンビニ向け標準料金代理収納用バーコード(コンビニバーコード)の出力が可能となりました。
コンビニバーコードは、「財団法人流通システム開発センター」が、発行した「UCC/EAN128 による 標準料金代理収納ガイドラン」に準拠した、UCC/EAN128 のバーコードを生成する事が可能です。
バーコードの印字位置、バーコードの高さについては、mm(ミリ)単位で描画を行います。
バーコードの幅については、ガイドラインに準拠し、プリンタの解像度(dpi)に合わせて、描画を行います。


お客様作成アプリケーション
①
・ インスタンス生成
→Graphics オブジェクトを渡す
・ 各種プロパティ設定
・ バーコード描画指示
Graphics オブジェクト
②
バーコードの描画
③
様々な利用
・ 印刷
・ 画像ファイルへの出力
・ クリップボードへコピー
・ 画面に表示
・ ASP.NET でブラウザ出力
BarCode.NET
帳票
画像ファイル
ブラウザ
バーコードを SVG 形式で描画
ブラウザに出力
・ SVGZ 形式に変換(サンプルプログラムあり)
・ Html ファイルで取り込んで利用

2-2. 一次元バーコード作成クラスの機能

BarCode.NET の各一次元バーコード作成クラスは、以下の機能を有します。

(1) バーコードの描画

コード、始点(左上の X/Y 座標)とバーコードの高さ・幅を指定してバーコードを描画します。幅の代わりに、バーコードの線幅の最小値を指定して描画することも可能です。その場合、より高い精度のバーコードが作成できますが、最終的に描画される幅の調整が必要になります。

座標・高さ・幅の単位は、.NET の Graphics オブジェクトが持つ全ての座標単位で指定が可能です。

少し特殊な UCC/EAN128 のコード指定については、アプリケーション識別子(AI)の前に"{FNC1}"という文字列を指定することで UCC/EAN128 のバーコード作成を可能としています。

例) (01)04912345678904 (10)0211 (3103)001528　・・・()付きがアプリケーション識別子(AI)

コード指定方法→「{FNC1} 0104912345678904{FNC1} 100211{FNC1} 3103001528」

(2) 添字の描画

バーコードの下にコードの文字列自体を描画します(既定値)。プロパティの設定で描画をしないようにすることも可能です。

添字を、コードを意味するバーの位置に描画する(既定値)か、バーコード全体の幅に均等割付するかを指定することが可能です。

※　JAN(EAN)コードの場合、既定値の状態では商品コードのバーコードのような描画を行い、均等割付にすると書籍コードのバーコードのような描画を行います。

添字のフォントをプロパティで指定することも可能です。

Code39／NW7(Codebar) のみスタート・ストップキャラクタを印字するかどうかをプロパティで指定することが可能です。既定値は印字しません。

(3) 回転描画

プロパティの設定により、バーコードの左上始点座標を中心に、

90 度／180 度／270 度　回転して描画することが可能です。

既定値は、0 度です。

2-2-1. コンビニ向け標準料金代理収納用バーコード(コンビニバーコード)

UCC/EAN128 バーコード作成クラスにコンビニバーコード描画メソッドを用意しました。コンビニバーコードメソッドは、以下の機能を有します。

コンビニ EAN128.NET のバーコード作成クラスは、以下の機能を有します。

(1) バーコードの描画

コードと、始点(左上の X/Y 座標)及び、バーコードの高さを mm(ミリ)単位で指定し、バーコードを描画します。
バーコードの幅は、プリンタの解像度(dpi)により、以下の表の通り、
自動的に決まります。

解像度	モジュール幅		バーコード部の幅
	ドット	mm	
300dpi	2	0.169	48.67mm
400dpi	3	0.190	54.72mm
480dpi	3	0.158	45.50mm
600dpi	4	0.169	48.67mm
300dpi の倍数	2 の倍数	0.169	48.67mm

アプリケーション識別子(AI)の前に”{FNC1}”という文字列を指定することで UCC/EAN128 のバーコード作成を可能としています。
例) (91)912345　・・・()付きがアプリケーション識別子(AI)
　コード指定方法→「{FNC1}91912345」

(2) 添字の描画

バーコードの下にコードの文字列自体を描画します。この添字(コード文字列)は、ガイドラインに従い、左詰で以下のように描画します。

(91)912345-123456789012345678921 1
020331-0-123456-2

なお、これは、コードで以下のように指定された場合です。
「{FNC1}9191234512345678901234567892110203310123456 2」

2-3. 郵便カスタマバーコード作成クラスの機能

BarCode.NET の郵便カスタマバーコード作成クラスは、以下の機能を有します。

(1) バーコードの描画

コード、始点(左上の X/Y 座標)と大きさとしてポイント(8～11.5)を指定してバーコードを描画します。

コードの表記は・・・

[郵便番号の数字部分 7 桁]+[郵便番号では不明部分の住所の英数字を「－」区切り]で、指定してください。

例) 〒116-0013 東京都荒川区西日暮里五丁目 37 番 5 号スタートアップオフィスＡ－207 号室

コード指定方法→「11600135-37-5-A-207」

※詳しくは、旧郵政省の web ページにマニュアルがございますのでご覧になってください。

座標の単位は、.NET の Graphics オブジェクトが持つ全ての座標単位で指定が可能です。

(2) 回転描画

プロパティの設定により、バーコードの左上始点座標を中心に、
90 度／180 度／270 度　回転して描画することが可能です。
既定値は、0 度です。

2-4. QR コード作成クラスの機能

BarCode.NET の QR コード作成クラスは、以下の機能を有します。

バーコードの描画

コード、始点(左上の X/Y 座標)とバーコードを描画する最小値を指定して描画することが可能です。

座標・バーコードを描画する最小値の単位は、.NET の Graphics オブジェクトが持つ全ての座標単位で指定が可能です。

その他に、プロパティで以下の項目を指定することが必要です。

- バージョン(1～40)
- エラー訂正レベル(L,M,Q,H)
- エンコードモード
  (N:数字モード A:英数字モード　その他:8bit byte モード)

エンコードモードに漢字モードがありませんが、漢字の入力も「その他:8bit byte モード」を指定してください。”N”/”A”以外の文字であればなんでも OK です。

３．アプリケーションプログラムから BarCode.NET の使用方法

3-1. クラス仕様

3-1-1. 概要

BarCode.NET は、以下のそれぞれバーコードごとに独立したクラスで構成されております。

Pao.BarCode
- **Pao.BarCode.Jan13**
- **Pao.BarCode.Jan8**
- **Pao.BarCode.ITF**
- **Pao.BarCode.Matrix2of5**
- **Pao.BarCode.NW7**
- **Pao.BarCode.Code39**
- **Pao.BarCode.Code128**
- **Pao.BarCode.EAN128**
- **Pao.BarCode.YubinCustomer**
- **Pao.BarCode.QRCode**

しかし、YubinCustomer/QRCode 以外の一次元バーコードのクラスは基本的に同一名のプロパティやメソッドといったメンバを所有し、それらの機能も基本的に同一です。(EAN128 クラスのコンビニバーコード描画メソッドだけ追加されています。)
そこで以降の各メンバの説明では、一次元バーコードのクラスと YubinCustomer(郵便カスタマバーコードクラス)と、QR コードのクラスの３つに分けてご説明いたします。

3-1-2．一次元バーコードクラスメンバ

3-1-2-1．コンストラクタ

初期処理を行う。
バーコードの種類別に以下のインタフェイスが存在します。

(1) **public JAN13(System.Drawing.Graphics g)**
(2) **public JAN8(System.Drawing.Graphics g)**
(3) **public ITF(System.Drawing.Graphics g)**
(4) **public Matrix2of5(System.Drawing.Graphics g)**
(5) **public NEC2of5(System.Drawing.Graphics g)**
(6) **public NW7(System.Drawing.Graphics g)**
(7) **public Code39(System.Drawing.Graphics g)**
(8) **public Code128(System.Drawing.Graphics g)**
(9) **public EAN128(System.Drawing.Graphics g)**

・引数

System.Drawing.Graphics g

バーコードの描画を行う **Graphics** を指定します。

・戻り値

なし。

・割込発生エラー

なし。

3-1-2-2．メソッド

(1) **public void Draw(string code, float x, float y, float width, float height)**

バーコードの描画を行います。指定幅を正確に合わせるためにいったん描画したバーコードを縮小して描画しなおします。そのため、多少、精度が劣化します。ドット単位で正確な精度を期待する場合は、指定した幅と正確に一緒にはなりませんが、DrawDirect / DrawDelicate メソッドを使用してください。

・引数

① **string code**

描画を行うバーコードのコードを文字列で指定します。

UCC/EAN128 のコード指定については、アプリケーション識別子(AI)の前に”{FNC1}”という文字列を指定することで UCC/EAN128 のバーコード作成を可能としています。

例) (01)04912345678904 (10)0211 (3103)001528　・・・()付きがアプリケーション識別子(AI)

コード指定方法→「{FNC1} 0104912345678904{FNC1} 100211{FNC1} 3103001528」

② **float x**

描画位置の始点(左上)の X 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float width**

バーコードの全体の幅を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

⑤ **float height**

バーコードのバーの高さを指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

・戻り値

なし

・例外の種類

予測可能割込発生エラーを以下クラス別に記述します。

○ JAN13

① **public errJAN13BadChar()**

数字以外の文字が使用されました。

使用できる文字は数字のみです。

② **public errJAN13BadLen()**

コードの桁数は、13 桁か、12 桁を指定してください。

12 桁の場合チェックキャラクタを自動付与します。

③ **public errJAN13CheckDigit()**
コード末尾のチェックデジットが誤っています。

○ JAN8

④ **public errJAN8BadChar()**
数字以外の文字が使用されました。
使用できる文字は数字のみです。

⑤ **public errJAN8BadLen()**
コードの桁数は、8 桁か、7 桁を指定してください。
12 桁の場合チェックキャラクタを自動付与します。

⑥ **public errJAN8CheckDigit()**
コード末尾のチェックデジットが誤っています。

○ ITF

⑦ **public errITFBadChar()**
数字以外の文字が使用されました。
使用できる文字は数字のみです。

○ Matrix2of5

⑧ **public errMatrix2of5BadChar()**
数字以外の文字が使用されました。
使用できる文字は数字のみです。

○ NEC2of5

⑨ **public errNEC2of5BadChar()**
数字以外の文字が使用されました。
使用できる文字は数字のみです。

○ NW7

⑩ **public errNW7BadChar()**
利用できない文字 ='？ ' が使用されました。
使用できる文字は"ABCD.+:/$-0123456789"です。

○ Code39

⑪ **public errCode39BadChar()**
利用できない文字 ='？ ' が使用されました。
使用できる文字は
"1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-. *$/+%"です。

○ Code128
予測可能割り込み発生エラーなし。

○ EAN128
予測可能割り込み発生エラーなし。

(2) **public void DrawDirect(string code, float x, float y, float width, float height)**

バーコードの描画を行います。
指定幅以内で最も広い幅でバーコードを直接描画します。
ドット単位での描画精度を実現します。

・引数

① **string code**

描画を行うバーコードのコードを文字列で指定します。
UCC/EAN128 のコード指定については、アプリケーション識別子(AI)の前に”{FNC1}”という文字列を指定することで UCC/EAN128 のバーコード作成を可能としています。

例) (01)04912345678904 (10)0211 (3103)001528　・・・()付きがアプリケーション識別子(AI)

コード指定方法→「{FNC1} 0104912345678904{FNC1} 100211{FNC1} 3103001528」

② **float x**

描画位置の始点(左上)の X 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float width**

バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

⑤ **float height**

バーコードのバーの高さを指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

・戻り値

なし

・例外の種類

Draw メソッドと同様。

(3) **public void DrawDelicate(string code, float x, float y, float minLineWidth, float height)**

バーコードの描画を行います。

Draw メソッドとの違いは、バーコード全体の幅を指定するのではなく、バーを描画する一番細い線の幅を指定します。

Draw メソッドより、精度の高いバーコードを描画することが可能です。ただし、バーコード全体の幅の調整が必要になります。

※この **DrawDelicate** メソッドを使用してバーの最小幅を指定した場合に精度が高くなる理由は、直接、Graphics オブジェクトに線を描画するためです。

Draw メソッドを使用して全体の幅を指定した場合、一旦、仮想空間に描画したバーコードを指定された全体の幅に対して当てはまるように Graphics オブジェクトに縮小描画しております。

・引数

① **string code**

描画を行うバーコードのコードを文字列で指定します。

UCC/EAN128 のコード指定については、アプリケーション識別子(AI)の前に”{FNC1}”という文字列を指定することで UCC/EAN128 のバーコード作成を可能としています。

例) (01)04912345678904 (10)0211 (3103)001528　・・・()付きがアプリケーション識別子(AI)

コード指定方法→「{FNC1} 0104912345678904{FNC1} 100211{FNC1} 3103001528」

② **float x**

描画位置の始点(左上)の X 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float minLineWidth**

バーコードを描画するバーの最小幅の値を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

⑤ **float height**

バーコードのバーの高さを指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

・戻り値

なし

・例外の種類

Draw メソッドと同様。

(4) **public void WriteSVG**
(string code, float x, float y, float width, float height , string filePath)
SVG ファイルへのバーコードの出力を行います。

・引数

① **string code**
描画を行うバーコードのコードを文字列で指定します。
UCC/EAN128 のコード指定については、アプリケーション識別子(AI)の前に”{FNC1}”という文字列を指定することで UCC/EAN128 のバーコード作成を可能としています。
例) (01)04912345678904 (10)0211 (3103)001528　・・・()付きがアプリケーション識別子(AI)
コード指定方法→「{FNC1} 0104912345678904{FNC1} 100211{FNC1} 3103001528」

② **float x**
描画位置の始点(左上)の X 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**
描画位置の始点(左上)の Y 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float width**
バーコードの全体の幅を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
ただし、SVG ファイルは、ブラウザ表示用のため、画面の pixel とバーコードの線が一致しないといけないため、指定された幅以下のサイズに最適化されます。

⑤ **float height**
バーコードのバーの高さを指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

⑥ **string filePath**
SVG ファイルのファイル名をフルパスで指定してください。

・戻り値
なし

・例外の種類
Draw メソッドと同様。

3-1-2-3．プロパティ

(1) **public bool TextWrite**

true： 添字の描画を行う。(既定値)
false： 添字を描画しない。

(2) **public bool TextKintou**

true： 添字の描画は、バーコード全体の幅に均等割付で行う。
false： 添字を描画は、コードを意味するバーの位置に行う。(既定値)

(3) **public Font TextFont**

添字のフォント。
既定値は、”**ＭＳ ゴシック　9 ポイント　標準**”。

(4) **public float RotateAngle**

回転角度を数値で指定。左下を軸に右回転して描画を行う。
既定値は、０度。

(5) **public bool DispStartStopCode**

Code39/NW7 のみ使用可能なプロパティ
true： スタート／ストップコードの描画を行う。
false： スタート／ストップコードを描画しない。(既定値)

3-1-2-4. EAN128 コンビニバーコードメソッド

(5) **public void DrawConvenience(string code, float x, float y, float height)**

コンビニバーコードの描画を行います。

・引数

① **string code**

描画を行うバーコードのコードを文字列で指定します。
アプリケーション識別子(AI)の前に”{FNC1}”という文字列を指定することで UCC/EAN128 のバーコード作成を可能としています。
例) (91)912345　・・・()付きがアプリケーション識別子(AI)
　　コード指定方法→「{FNC1}91912345」

② **float x**

描画位置の始点(左上)の X 座標を指定します。
単位は、mm(ミリ)です。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。
単位は、mm(ミリ)です。

④ **float height**

バーコードのバーの高さを指定します。
単位は、mm(ミリ)です。

・戻り値

なし

3-1-2-5. 使用プロパティ

(1) **public Font TextFont**

添字のフォント。
既定値は、”**ＭＳ ゴシック　9 ポイント　標準**”。
(8 ポイントが妥当かもしれません)

(2) **public bool TextWrite**

true： 添字の描画を行う。(既定値)
false： 添字を描画しない。

3-1-3．郵便カスタマバーコードクラスメンバ

3-1-3-1．コンストラクタ

初期処理を行う。

public YubinCustomer (System.Drawing.Graphics g)

・引数

System.Drawing.Graphics g

バーコードの描画を行う **Graphics** を指定します。

・戻り値

なし。

・割込発生エラー

なし。

3-1-3-2. メソッド

public void Draw(string code, float x, float y, float point)

バーコードの描画を行います。

・引数

① **string code**

描画を行うバーコードのコードを文字列で指定します。

コードは・・・

[郵便番号の数字部分 7 桁]+[郵便番号では不明部分の住所の英数字を「－」区切り]

で、指定してください。

例) 〒116-0013 東京都荒川区西日暮里五丁目 37 番 5 号スタートアップオフィスＡ－207 号室

コード指定方法→「11600135-37-5-A-207」

② **float x**

描画位置の始点(左上)の X 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float point**

バーコード大きさを表すポイントを指定します。

ポイントは、8～11.5 の範囲内の数値で指定してください。

・戻り値

なし

・例外の種類

public errYubinBadChar ()

半角英数字－(ハイフォン)以外の文字が使用されました。

public void WriteSVG
(string code, float x, float y, float point , string filePath)
SVG ファイルへのバーコードの出力を行います。

・引数
② **string code**
描画を行うバーコードのコードを文字列で指定します。
コードは・・・
[郵便番号の数字部分 7 桁]+[郵便番号では不明部分の住所の英数字を「－」区切り]
で、指定してください。
例) 〒116-0013 東京都荒川区西日暮里五丁目 37 番 5 号スタートアップオフィスＡ－207 号室
コード指定方法→「11600135-37-5-A-207」
② **float x**
描画位置の始点(左上)の X 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
③ **float y**
描画位置の始点(左上)の Y 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
④ **float point**
バーコード大きさを表すポイントを指定します。
ポイントは、8～11.5 の範囲内の数値で指定してください。
⑤ **string filePath**
SVG ファイルのファイル名をフルパスで指定してください。

・戻り値
なし

・例外の種類
public errYubinBadChar ()
半角英数字－(ハイフォン)以外の文字が使用されました。

3-1-3-3. プロパティ

public float RotateAngle
回転角度を数値で指定。左下を軸に右回転して描画を行う。
既定値は、０度。

3-1-4. QR コードクラスメンバ

3-1-4-1. コンストラクタ

初期処理を行う。

public QRCode (System.Drawing.Graphics g)

・引数

System.Drawing.Graphics g

バーコードの描画を行う **Graphics** を指定します。

・戻り値

なし。

・割込発生エラー

なし。

3-1-4-2．メソッド

(1) **public void Draw (string code, float x, float y, float width, float height)**

バーコードの描画を行います。指定幅を正確に合わせるためにいったん描画したバーコードを縮小して描画しなおします。そのため、多少、精度が劣化します。ドット単位で正確な精度を期待する場合は、指定した幅と正確に一緒にはなりませんが、DrawDirect / DrawDelicate メソッドを使用してください。

・ 引数

① **string code**

描画を行うバーコードのコードを文字列で指定します。

② **float x**

描画位置の始点(左上)の X 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float width**

バーコードの全体の幅を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
通常、height と、同じ値を指定します。

⑤ **float height**

バーコードのバーの高さを指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
通常、width と、同じ値を指定します。

・戻り値

なし

・例外の種類

public errQRCodeOverLenght ()

指定されたコードの文字数が、指定されたバージョンの QR コードに格納できる文字数をオーバーした。

(2) **public void DrawDirect (string code, float x, float y, float width, float height)**

バーコードの描画を行います。
指定幅以内で最も広い幅でバーコードを直接描画します。
ドット単位での描画精度を実現します。

引数

① **string code**
描画を行うバーコードのコードを文字列で指定します。

② **float x**
描画位置の始点(左上)の X 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**
描画位置の始点(左上)の Y 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float width**
バーコードの全体の幅を指定します。
指定した幅以内で最も広い幅のバーコードを描画します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
通常、height と、同じ値を指定します。

⑤ **float height**
バーコードのバーの高さを指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
通常、width と、同じ値を指定します。

・戻り値
なし

・例外の種類
public errQRCodeOverLenght ()
指定されたコードの文字数が、指定されたバージョンの QR コードに格納できる文字数をオーバーした。

(3) **public void DrawDelicate (string code, float x, float y, float minLineWidth)**

バーコードの描画を行います。

Draw メソッドとの違いは、バーコード全体の幅を指定するのではなく、バーを描画する一番細かい単位を指定します。

Draw メソッドより、精度の高いバーコードを描画することが可能です。ただし、バーコード全体の幅の調整が必要になります。

※この **DrawDelicate** メソッドを使用してバーの最小幅を指定した場合に精度が高くなる理由は、直接、Graphics オブジェクトに描画するためです。

Draw メソッドを使用して全体の幅を指定した場合、一旦、仮想空間に描画したバーコードを指定された全体の幅に対して当てはまるように Graphics オブジェクトに縮小描画しております。

・ 引数

① **string code**

描画を行うバーコードのコードを文字列で指定します。

② **float x**

描画位置の始点(左上)の X 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**

描画位置の始点(左上)の Y 座標を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float minLineWidth**

バーコードを描画するの最小値を指定します。

単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

・戻り値

なし

・例外の種類

public errQRCodeOverLenght ()

指定されたコードの文字数が、指定されたバージョンの QR コードに格納できる文字数をオーバーした。

(3) **public void WriteSVG**
(string code, float x, float y, float width, float height, string filePath)
SVG ファイルへのバーコードの出力を行います。

・ 引数

① **string code**
描画を行うバーコードのコードを文字列で指定します。

② **float x**
描画位置の始点(左上)の X 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

③ **float y**
描画位置の始点(左上)の Y 座標を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。

④ **float width**
バーコードの全体の幅を指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
ただし、SVG ファイルは、ブラウザ表示用のため、画面の pixel とバーコードの線が一致しないといけないため、指定された幅以下のサイズに最適化されます。
通常、height と、同じ値を指定します。

⑤ **float height**
バーコードのバーの高さを指定します。
単位は、本メソッド呼び出し時の **Grahics.PageUnit** の値に依存します。
ただし、SVG ファイルは、ブラウザ表示用のため、画面の pixel とバーコードの線が一致しないといけないため、指定された幅以下のサイズに最適化されます。
通常、width と、同じ値を指定します。

⑥ **string filePath**
SVG ファイルのファイル名をフルパスで指定してください。

・戻り値
なし

・例外の種類
public errQRCodeOverLenght ()
指定されたコードの文字数が、指定されたバージョンの QR コードに格納できる文字数をオーバーした。

3-1-4-3．プロパティ

(1) **public int Version**

バージョン 1～40 を指定・取得。

(2) **public string ErrorCorrect**

エラー訂正レベル：”L”/”M”/”Q”/”H” のいずれかを指定・取得。

(2) **public string EncodeMode**

エンコードモードを指定・取得します。

“N”:数字モード

“A”:英数字モード

その他:8bit byte モード

エンコードモードに漢字モードがありませんが、漢字の入力も「その他:8bit byte モード」を指定してください。”N”/”A”以外の文字であればなんでも OK です。

3-2. C#での使用例(Code39 の例)

ここでは、Code39 のバーコード印刷を例にして簡単な使用方法を説明します。

まず、フォーム上に Button コントロールと PrintDocument コントロールを貼り付けてください。
次に、それぞれの各イベントに以下のようにコードを入れてください。

```
private void button1_Click(object sender, System.EventArgs e)
{
        printDocument1.Print();
}

private void printDocument1_PrintPage(object sender,
                                      System.Drawing.Printing.PrintPageEventArgs e)
{
        Pao.BarCode.Code39 cd39 = new Pao.BarCode.Code39(e.Graphics);
        cd39.Draw("12345", 50, 50, 200, 50);
}
```

ボタンをクリックすると、これだけで Code39 のバーコードは、出力されるでしょう。
しかし、指定した x=50/y=50/width=200/height=50 の単位はいったい何なんでしょう？
GDI+で使用できる単位系は複数あり、Graphics オブジェクトのプロパティである、PageUnit の値のいずれかで指定することが可能です。
そこで以下のように予めミリメートルを長さの単位に指定しますと、以下の例では x=1cm/y=1cm の位置から、幅=5cm/高さ=1.5cm のバーコードを出力することになります。

```
        e.Graphics.PageUnit = GraphicsUnit.Millimeter;
        Pao.BarCode.Code39 cd39 = new Pao.BarCode.Code39(e.Graphics);
        cd39.Draw("12345", 10, 10, 50, 15);
```

また、以下の例のように各種プロパティを指定して、添字を均等割付にし、フォントを変更し 270 度回転して印刷すること等も試してみてください。

```
        e.Graphics.PageUnit = GraphicsUnit.Millimeter;
        Pao.BarCode.Code39 cd39 = new Pao.BarCode.Code39(e.Graphics);
        cd39.TextKintou = true;
        cd39.TextFont = new Font("Times New Roman",12,FontStyle.Bold);
        cd39.RotateAngle = Pao.BarCode.RotateAngle.Angle270;
        cd39.Draw("12345", 100, 100, 50, 15);
```

詳しくは、サンプルプログラムを 8 本ご用意させていただきましたので、じっくり弄繰り回してください。

3-3. サンプルプログラム

C#.NET / VB.NET で作成した BarCode.NET を利用したサンプルプログラムを６本ご用意いたしました。

http://www.pao.ac/products/barcode.net/

より、BarCode.NET(バージョン).zip をダウンロードして解凍すると、Pao.BarCode.dll と一緒に６つのフォルダが作成されると思います。この６つのフォルダ内にそれぞれ、C#／VB という２つのフォルダがあり、それぞれに、C#.NET／VB.NET を利用したサンプルプログラムが入っております。Pao.BarCode.dll を６つのサンプルプログラムで参照設定して、起動してみてください。

(1) バーコードの印刷・プレビューサンプル

BarApp(通常のサンプル)¥C# 又は BarApp(通常のサンプル)¥VB フォルダ内にある BarApp.sln ファイルを起動する事により、このサンプルプログラムを開くことができます。

サンプルプログラムは、BarCode.NET の機能をフルに利用した印刷・プレビュー処理を実現しています。サンプルプログラムの割には、市販のバーコード作成ソフトにも見劣りしない多くの機能を実装しております。このままでも色々な用途があると思いますが、是非、弄繰り回して改造して遊んでください。

(2) クリップボードに貼り付け・画像ファイルに保存するサンプル

BarApp2(クリップボード・画像処理)¥C# 又は BarApp2(クリップボード・画像処理)¥VB フォルダ内にある BarApp2.sln ファイルを起動する事により、このサンプルプログラムを開くことができます。

バーコードをフォーム上のピクチャーボックスに表示後、クリップボードに貼り付けたり、Bitmap や JPEG 等のファイルに保存するサンプルプログラムです。

(3) QR コードの描画・印刷・プレビューサンプル

QrApp(QR コードサンプル)¥C# 又は QrApp(QR コードサンプル)¥VB フォルダ内にある QRApp.sln ファイルを起動する事により、このサンプルプログラムを開くことができます。

サンプルプログラムは、BarCode.NET の機能をフルに利用した印刷・プレビュー処理を実現しています。

(4) SVG / SVGZ 出力・ブラウザ表示サンプル

BarSVG(SVG･SVGZ サンプル)¥C# 又は BarSVG(SVG･SVGZ サンプル)¥VB フォルダ内にある SVG.sln ファイルを起動する事により、このサンプルプログラムを開くことができます。

QR コード以外のバーコードの SVG /SVGZ 出力とブラウザ表示をを行うサンプルです。

(5) QR コード、SVG / SVGZ 出力・ブラウザ表示サンプル
QrSVG(SVG・SVGZ サンプル)¥C# 又は QrSVG(SVG・SVGZ サンプル)¥VB フォルダ内にある QrSVG.sln ファイルを起動する事により、このサンプルプログラムを開くことができます。
QR コードの SVG /SVGZ 出力とブラウザ表示をを行うサンプルです。

(6) ASP.NET を使って QR コードブラウザ出力するサンプル
QRWeb(ASP.NETサンプル)フォルダ内にある QRWeb.sln ファイルを起動する事により、このサンプルプログラムを開くことができます。
ただし、このプロジェクトは、ASP.NET アプリケーションですので、C:¥Inetpub¥wwwroot¥ 等の IIS の WEB サイトフォルダにコピーして、ASP.NET で動く設定を行ってから、ソリューションを開いてください。
今のところ、C#版のみです。VB.NET 版は、作成していません。
QR コードを ASP.NET を使用してブラウザ表示をを行うサンプルです。

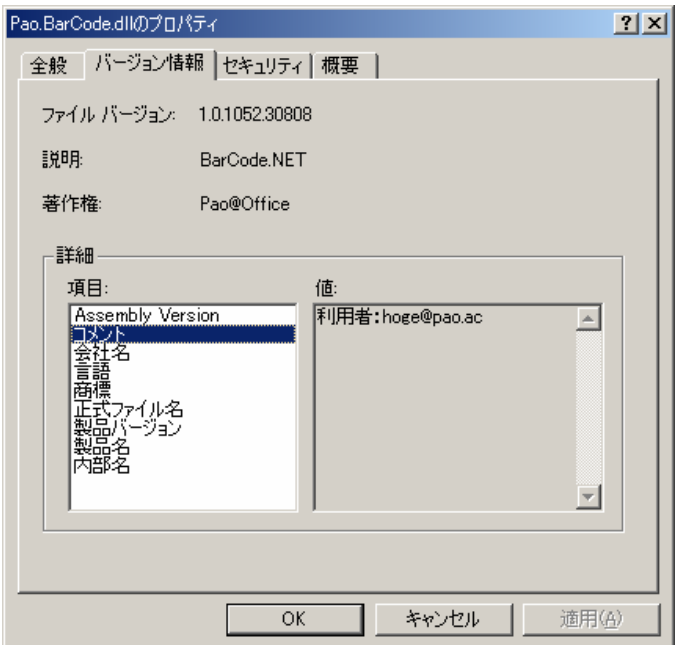
(7) コンビニバーコードの印刷・プレビューサンプル
コンビニ EAN128 サンプル¥C# 又は コンビニ EAN128 サンプル¥VB フォルダ内にある BarApp.sln ファイルを起動する事により、このサンプルプログラムを開くことができます。
サンプルプログラムは、コンビニバーコードを伝票に描画し、印刷・プレビュー処理を実現しています。

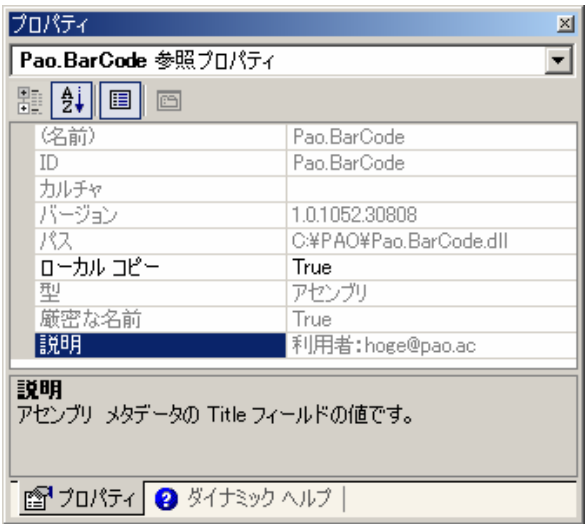
４．使用条件等

4-1. お試し版と製品版

BarCode. NET は、いつでもどこでも誰でも試用できるように、
http://www.pao.ac/products/barcode.net/
にお試し版として最新バージョンを常にご用意させていただいております。
お試し版の制限は、バーコードに控えめに「SAMPLE」という文字が入ります。

QR コードのお試し版の制限は、指定したコードの先頭に半角の「9」という文字が入ります。
製品版は、BarCode.NET を購入(ユーザ登録)された方全員にオーダーメイドで作成し、メールでお送りいたします。
エクスプローラ等でファイルを右クリックして、「プロパティ－バージョン情報－コメント」を選択すると、購入(ユーザ登録)された方のメールアドレスが表示されます。また、Visual Studio .NET プロジェクト内の参照プロパティにもメールアドレスは表示されます。





バージョンアップの際は、Web サイトにてお知らせいたします。
お客様には、WebSite にてバージョンアップ依頼をしていただきます。
すぐにバージョンアップ版をメールにてお送りいたします。

4-2. 使用許諾

BarCode.NET の使用について、BarCode.NET の使用者(以下「利用者様」と称します)と有限会社パオ・アット・オフィス(以下「弊社」と称します)は、以下の各項目についての内容に同意するものとします。

1.BarCode.NET の使用に関する使用許諾書

この使用許諾書は、利用者様がお使いのパソコンにおいて、BarCode.NET を使用する場合に同意しなければならない契約書です。

2.使用許諾書の同意

利用者様が BarCode.NET を使用する時点で、本使用許諾書に同意されたものとします。同意されない場合は、BarCode.NET を使用する事はできません。

3.ライセンス(使用権)の購入

利用者様が BarCode.NET の製品版を使用して開発を行う場合には、1 人または 1 台のコンピュータで BarCode.NET を使用するにあたり、1 ライセンスを購入する必要があります。同時に複数の人が複数のコンピュータで使用する場合はどちらか少ない方のライセンスを購入しなければなりません。

4.著作権

BarCode.NET及の著作権は、いかなる場合においても弊社に帰属いたします。

5.免責

BarCode.NET の使用によって、直接的、あるいは、間接的に生じた、いかなる損害に対しても、弊社は補償賠償の責任を負わないものとします。

6.禁止事項

BarCode.NET 及びその複製物を第三者に譲渡・貸与する事は出来ません。BarCode.NET を開発ツールとして再販/再配布することを禁止します。なお、モジュールとして組み込みを行い再販/再配布する場合は、開発ツールとしての再販/再配布には含まれませんので、OK です。

7.保証の範囲

弊社は BarCode.NET の仕様を予告無しに変更することがあります。その場合の利用者様に対する情報提供は、弊社 WebSite にて行う事とします。

8.適用期間

本使用許諾条件は利用者様が BarCode.NET を使用した日より有効です。利用者様が本使用許諾条件のいずれかの条項に違反した場合、又は、本許諾条件に同意出来ない場合は、利用者様は BarCode.NET を一切使用出来ないものとします。

4-3. 代金支払い方法(ユーザ登録の方法)

BarCode.NET の製品版をご利用頂ける場合は、ライセンスを購入して頂く必要があります。ライセンス形態及び代金支払方法は以下のとおりです。

- 必要なライセンス数の数え方
  - BarCode.NET を利用する人数、もしくは、BarCode.NET をインストールするパソコンの台数で、いずれか少ない方。
    - 例 1)　4 台のパソコンにインストールしてご利用頂く場合､利用する人数が 2 人であれば、必要なライセンス数は 2 つです。
    - 例 2)　1 人の人が自宅と職場の両方で BarCode.NET をご利用頂く場合、明らかにその使用目的が異なる場合であれば、それぞれに 1 ライセンスが必要です。(職場では法人扱い、自宅では個人扱いという場合。)
- 1 ライセンス当たりの価格
  - 18,000 円
  - ソースコード付：54,000 円
    - 本価格には消費税を含みません。
    - バグフィックス等のバージョンアップは原則として無償とさせて頂きます。
    - 大幅な機能追加等によるバージョンアップの場合には別ライセンスとさせていただく場合がございます。
    - 本価格は BarCode.NET の使用権に対するものです｡カスタマイズや保守等の費用は一切含まれておりません。
- お支払方法

(18,000 円 or 54,000 円×ライセンス数)+5%(消費税)を下記口座へ銀行振込､または、郵便振替による送金をして下さい。

銀行名	本支店名	口座番号	名義
三井住友銀行	日暮里支店	普通 6629094	ユ）パオアツトオフイス

郵便口座番号	名義
００１５０－０－　５７６８４５	有限会社　パオ・アット・オフィス

  - 振込手数料は利用者様負担でお願い致します。
- お支払いの通知と製品の送付
  - 振り込みが完了した時点で、必ず弊社 WebSite の「BarCode.NET 入金連絡フォーム」から入金のご連絡をお願いいたします。
    http://www.pao.ac/products/barcode.net/buy.html#form
  - 弊社では上記連絡を受けて入金確認を行い､BarCode.NET の製品自体を利用者様へ電子メールにてお送りさせて頂きます。
    - 利用者様へは電子メール以外での製品の提供は原則として行いません。
    - 製品の再送付は原則として行いません。製品のファイルは消去してしまわないように大切にお取り扱いください。
  - お振り込み頂いても入金の連絡がない場合、こちらか振り込み人様の情報が分からないため、製品の送付が行えません。必ず入金連絡を行って頂くようお願いいたします。

- 見積書/請求書/領収書の発行について
  見積書/請求書/領収書は原則として発行致しません。法人様での利用等でどうしても必要となる場合等は、お手数ですが弊社までメールにてご連絡下さい。