

# BarCode for WPF

2018.07.20 更新

グレースィティ株式会社

## 目次

<a href="#">BarCode for WPF</a>	2
<a href="#">はじめに</a>	2
<a href="#">ComponentOne for WPF のヘルプ</a>	2
<a href="#">主な特長</a>	2
<a href="#">クイックスタート</a>	2
<a href="#">手順 1:アプリケーションの設定</a>	2-3
<a href="#">手順 2:コードの追加</a>	3-5
<a href="#">手順 3:アプリケーションの実行</a>	5-7
<a href="#">BarCode for WPF の使い方</a>	7
<a href="#">サポートされるエンコーディング</a>	7-10
<a href="#">C1BarCode コントロールのカスタマイズ</a>	10-11
<a href="#">C1BarCode 画像の保存</a>	11-12
<a href="#">QR コード</a>	12

## BarCode for WPF

BarCode for WPF を使用して、バーコード画像をアプリケーションに追加します。

バーコードフォントとは異なり、BarCode は、使用しているエンコーディングに応じて、自動的に必要な制御シンボルとチェックサムを追加しながら値をエンコードすることで、読み取りエラーをなくします。

さらに、BarCode は、たいへん使いやすい製品です。フォームにコントロールを追加し、エンコーディングタイプを設定するだけで使用できます。

### はじめに

## ComponentOne for WPF のヘルプ

### はじめに

ComponentOne for WPF のインストール、ライセンス、テクニカルサポート、名前空間、およびこのコントロールを含む プロジェクトの作成方法については、「[ComponentOne for WPF ユーザーガイド](#)」を参照してください。

## 主な特長

- 36 種類のエンコーディングのサポート**  
**C1Barcode** コントロールは、Codabar、Code128 Auto、Code39、Code93、DataMatrix、Ean13、Ean8、PostNet、QRCode、RSS14 などの 36 種類のエンコーディングをサポートします。
- チェックサムを自動的に追加**  
**C1Barcode** コントロールは、バーコードが正しく読み取られるように、使用されているエンコーディングに応じて、必要な制御シンボルとチェックサムを自動的に追加しながら値をエンコードします。
- 配布が容易な無償の DLL**  
**C1Barcode** は、通常のアセンブリと同様に、無償の DLL としてアプリケーションと共に配布できます。

## クイックスタート

### 手順1:アプリケーションの設定

この手順では、Visual Studio で新しい WPF アプリケーションを作成し、適切な参照をプロジェクトに追加します。さらに、C1Barcode コントロールを作成するための XAML マークアップを追加します。

- Visual Studio で新しい WPF アプリケーションを作成します。
- ソリューションエクスプローラで **Resources** フォルダを右クリックし、**[追加]**、**[既存の項目]** を順に選択します。**[既存項目の追加]** ダイアログボックスが開きます。
- アプリケーションに追加する画像ファイルを見つけます。この例では、「c1logo.png」を使われます。
- ファイルを選択し、**[OK]** をクリックします。ファイルが **Resources** フォルダに追加されます。
- アプリケーションでファイルを使用できるように、アプリケーションをリビルドします。
- MainWindow.xaml** ファイルを開き、開始タグ `<Window>` `</Window>` を見つけます。このタグには、C1Barcode コントロールを使用できるように必要な名前空間が含まれています。次のマークアップになるように、タグを編集します。

XAML	copyCode
<pre>&lt;Window x:Class="BarCodeApp.MainWindow"         xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"         xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"         xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"         xmlns:Barcode="clr-namespace:C1.BarCode;assembly=C1.WPF.BarCode.4"         Title="BarCode Sample" Height="621.567" Width="849.09"&gt;</pre>	

# BarCode for WPF

7. ページの `<Grid></Grid>` タグの間にカーソルを置き、次の XAML マークアップを追加し、グリッドのリソースと行定義を設定します。その後、3つの `TextBlock` コントロールと1つの `TextBox` と1つの `ComboBox` コントロールを追加します。

XAML	copyCode
<pre>&lt;Grid.Resources&gt;   &lt;Style TargetType="TextBlock"&gt;     &lt;Setter Property="FontSize" Value="18"&gt;&lt;/Setter&gt;   &lt;/Style&gt;   &lt;Style TargetType="TextBox"&gt;     &lt;Setter Property="FontSize" Value="18"&gt;&lt;/Setter&gt;   &lt;/Style&gt;   &lt;Style TargetType="ComboBox"&gt;     &lt;Setter Property="FontSize" Value="18"&gt;&lt;/Setter&gt;   &lt;/Style&gt; &lt;/Grid.Resources&gt; &lt;Border HorizontalAlignment="Center"&gt;   &lt;Grid&gt;     &lt;Grid.RowDefinitions&gt;       &lt;RowDefinition Height="20*" /&gt;       &lt;RowDefinition Height="20*" /&gt;       &lt;RowDefinition Height="60*" /&gt;     &lt;/Grid.RowDefinitions&gt;     &lt;Grid.ColumnDefinitions&gt;       &lt;ColumnDefinition Width="40*" /&gt;       &lt;ColumnDefinition Width="60*" /&gt;     &lt;/Grid.ColumnDefinitions&gt;      &lt;!-- 3つの TextBlock コントロールと1つのTextBox と1つの ComboBox コントロールを追加します --&gt;     &lt;TextBlock Text="CodeType:" VerticalAlignment="Center"&gt;&lt;/TextBlock&gt;     &lt;ComboBox x:Name="cbCodeType" HorizontalAlignment="Left" Grid.Column="1"       Width="414" Height="50"       SelectionChanged="cbCodeType_SelectionChanged" /&gt;     &lt;TextBlock Text="Text:" Grid.Row="1" VerticalAlignment="Center"&gt;&lt;/TextBlock&gt;     &lt;TextBox x:Name="text" Text="{Binding Text, ElementName=barcode,       UpdateSourceTrigger=PropertyChanged, FallbackValue='', Mode=TwoWay}"       HorizontalAlignment="Left" Grid.Column="1" Grid.Row="1" Height="50"       Width="414" TextChanged="text_TextChanged" /&gt;     &lt;TextBlock Text="BarCode:" Grid.Row="2" VerticalAlignment="Center" /&gt;     &lt;Grid Grid.Row="2" Grid.Column="1" Background="White" HorizontalAlignment="Left" Width="414"&gt;     &lt;/Grid&gt;   &lt;/Grid&gt; &lt;/Border&gt;</pre>	

8. 次のように、`<Grid></Grid>` タグ内に `C1Barcode` を配置します。

XAML	copyCode
<pre>&lt;c1:C1Barcode x:Name="barcode" AutoSize="False" BarHeight="0" CodeType="QRCode" CaptionPosition="Above"   Text="http://www.grapecity.co.jp" Height="200" Width="200" Margin="315,249,245,55"&gt;   &lt;c1:C1Barcode.QRCodeOptions&gt;     &lt;Barcode:QRCodeOptions ErrorLevel="High" /&gt;   &lt;/c1:C1Barcode.QRCodeOptions&gt; &lt;/c1:C1Barcode&gt;</pre>	

9. 次のマークアップは、手順4で `Resources` フォルダに追加した画像ファイルを追加します。

XAML	copyCode
<pre>&lt;Image Source="Resources/c1logo.png"   x:Name="image" Width="70" Height="70" Margin="378,310,312,124" /&gt;</pre>	

マークアップは `</Grid>` と `</Window>` で完了することに注意してください。

この手順では、このクイックスタートガイドで使用されるためのアプリケーションが作成しました。

## 手順2:コードの追加

この手順では、アプリケーションに必要なコードを追加します。

MainWindow.xaml.csファイルにコードを追加するには、以下の手順を実行します。

1. コードビューに切り替えて、次の名前空間を追加します。

```

    ◦ Visual Basic
Imports C1.BarCode
    ◦ C#
using C1.BarCode;

```

2. **MainWindow()**のコンストラクタ内に、次のイベントを一つずつ登録して、登録されたイベントに対してハンドラーを生成するために、TABキーを押します。

```

    ◦ Visual Basic
AddHandler Me.Loaded, AddressOf MainWindow_Loaded
AddHandler cbCodeType.SelectionChanged, AddressOf cbCodeType_SelectionChanged
AddHandler text.TextChanged, AddressOf text_TextChanged
    ◦ C#
this.Loaded += MainWindow_Loaded;
cbCodeType.SelectionChanged += cbCodeType_SelectionChanged;
text.TextChanged += text_TextChanged;

```

3. 次に、**MainWindow\_Loaded** イベントに次のコードを追加します。このコードは、**MainWindow**がロードされる次第BarCodeコントロールをロードします。

```

    ◦ Visual Basic
Private Sub MainWindow_Loaded(sender As Object, e As RoutedEventArgs)
    cbCodeType.ItemsSource = [Enum].GetValues(GetType(CodeType))
    cbCodeType.SelectedItem = barcode.CodeType
End Sub
    ◦ C#
void MainWindow_Loaded(object sender, RoutedEventArgs e)
{
    cbCodeType.ItemsSource = Enum.GetValues(typeof(CodeType));
    cbCodeType.SelectedItem = barcode.CodeType;
}

```

4. 手順2で作成された**cbCodeType\_SelectionChanged**イベントに次のコードを追加します。そうすると、ComboBoxコントロールでBarCodeのタイプが変更される際、MainWindowで表示されているBarCodeのタイプも変更されます。

```

    ◦ Visual Basic
Private Sub cbCodeType_SelectionChanged(sender As Object, e As SelectionChangedEventArgs)
    If barcode IsNot Nothing Then
        Try
            barcode.CodeType = DirectCast(cbCodeType.SelectedItem, CodeType)
            If barcode.CodeType <> CodeType.QRCode
                OrElse Not Text.Text.Equals("http://www.grapecity.co.jp") Then
                Image.Opacity = 1
            Else
                Image.Opacity = 0
            End If
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try
    End If
End Sub
    ◦ C#
void cbCodeType_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (barcode != null)
    {
        try
        {
            barcode.CodeType = (CodeType)cbCodeType.SelectedItem;
            if (barcode.CodeType != CodeType.QRCode
                || !text.Text.Equals("http://www.grapecity.co.jp"))
            {
                image.Opacity = 1;
            }
        }
    }
}

```

```
    }  
    else  
    {  
        image.Opacity = 0;  
    }  
}  
catch (Exception ex)  
{  
    MessageBox.Show(ex.Message);  
}  
}  
}
```

5. テキストの制御を有効にするには、手順2で作成された`text_TextChanged`イベントに次のコードを追加します。

○ **Visual Basic**

```
Private Sub text_TextChanged(sender As Object, e As TextChangedEventArgs)  
    If Not String.IsNullOrEmpty(Text.Text)  
        AndAlso Text.Text.Equals("http://www.grapecity.co.jp")  
        AndAlso barcode.CodeType = CodeType.QRCode Then  
        Image.Opacity = 1  
    Else  
        Image.Opacity = 0  
    End If  
End Sub
```

○ **C#**

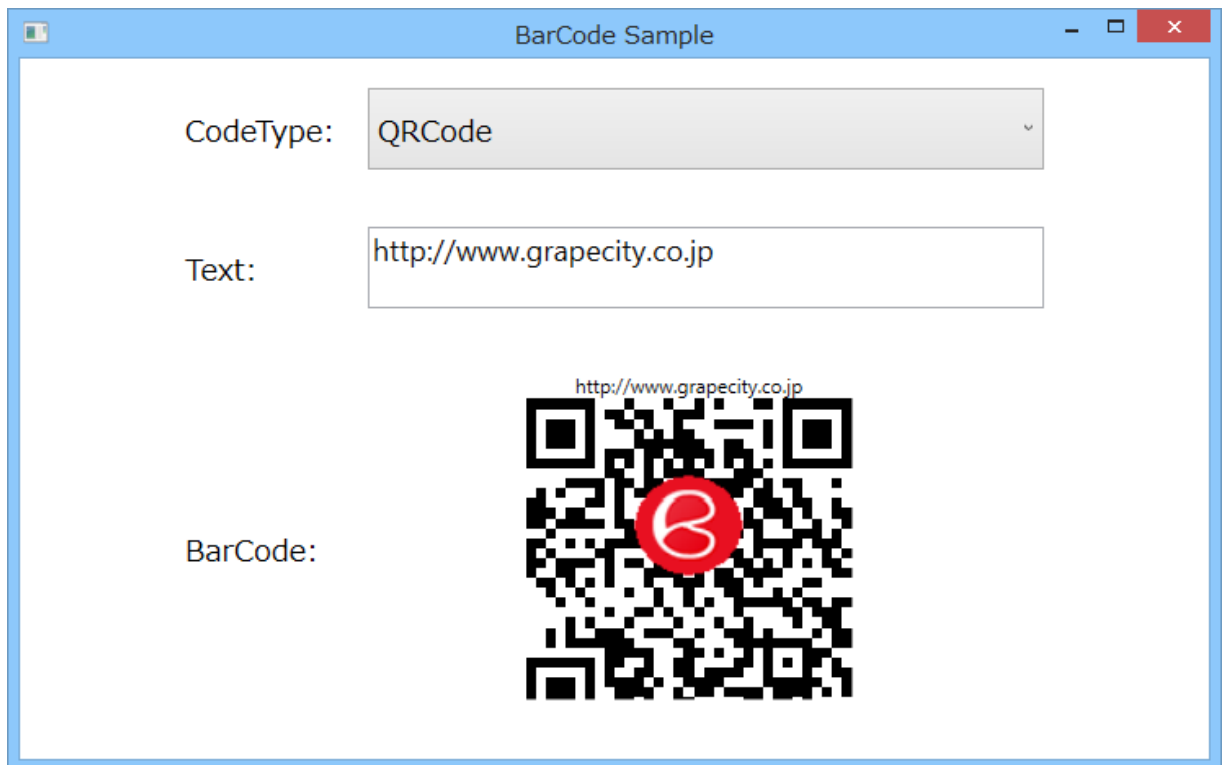
```
void text_TextChanged(object sender, TextChangedEventArgs e)  
{  
    if (!string.IsNullOrEmpty(text.Text) &&  
        text.Text.Equals("http://www.grapecity.co.jp") &&  
        barcode.CodeType == CodeType.QRCode)  
        image.Opacity = 1;  
    else  
        image.Opacity = 0;  
}
```

この手順では、XAMLを使用してコードの追加が完了しました。

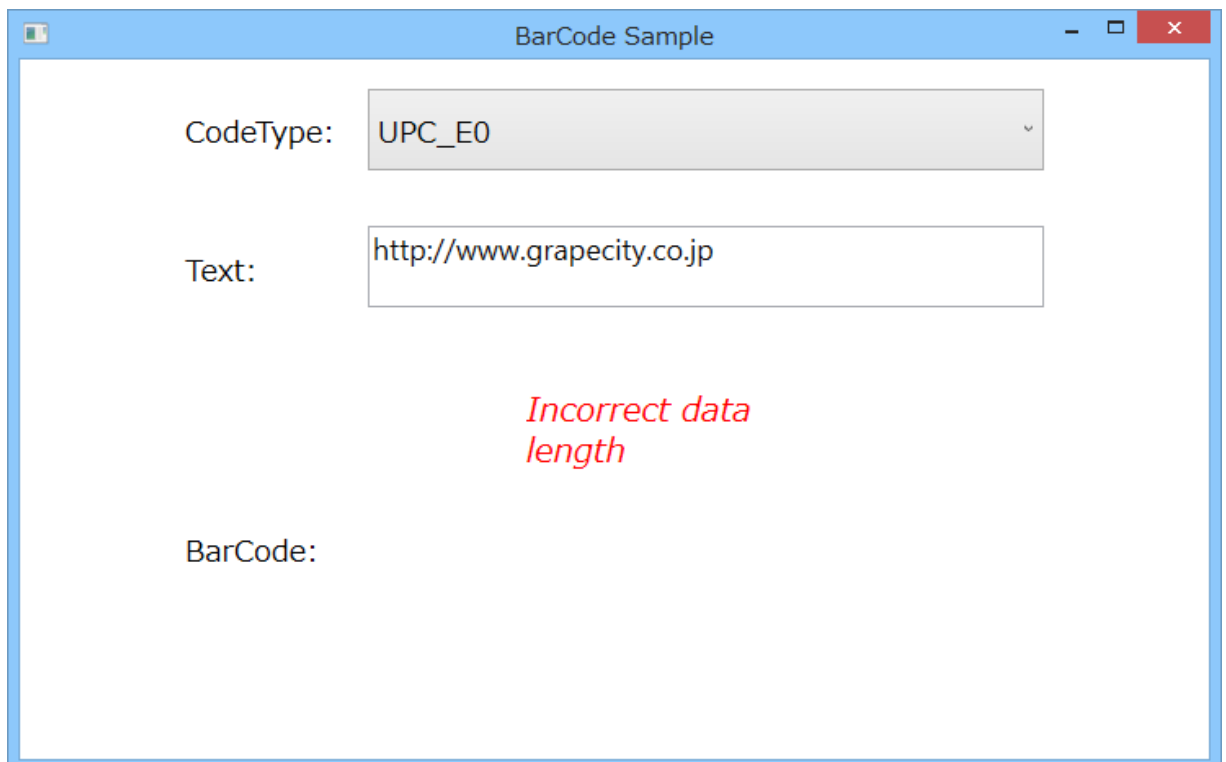
## 手順3:アプリケーションの実行

最後の手順では、アプリケーションの外観を制御するコードを追加しました。この手順では、作成したアプリケーションを実行します。

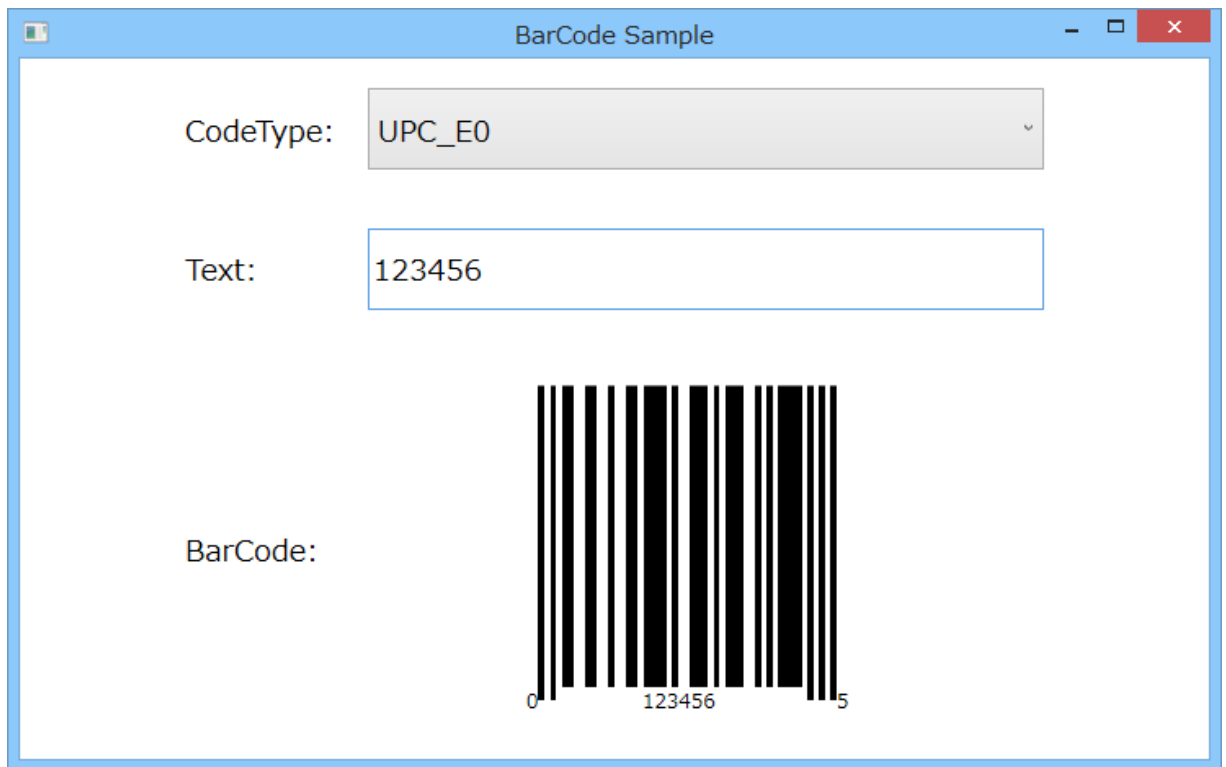
1. **[デバッグ開始]**を選択するか**[F5]**を押して、アプリケーションを開始します。次の図のようになります。



2. **[CodeType]**ドロップダウンリストから新しい**コードタイプ**を選択します。テキストが URL に設定されているため、多くのコードタイプでは、エラーメッセージが表示されます。



3. 最後に、**[Text]**テキストボックスの値を変更します。



## ここまでの成果

このクイックスタートでは、新しい Visual Studio アプリケーションを作成し、アプリケーションのフレームワークを作成する XAML マークアップを追加しました。さらに、**TextChanged** イベントと **SelectionChanged** イベントを制御するコードを追加しました。

## BarCode for WPF の使い方

### サポートされるエンコーディング

**C1Barcode** のエンコーディングタイプは、**CodeType** プロパティを設定して変更できます。**C1Barcode** コントロールは次のエンコーディングをサポートします。

エンコーディング	説明
BC412	The BC412 barcode was invented by IBM to meet the needs of the semiconductor wafer identification application.
Code11	Code11, also known as USD-8, is a high-density barcode symbology developed by Intermecc in 1977. It is primarily used to label telecommunication equipments. This symbology is discrete and is able to encode numeric digits through 0-9, dash (-), and start/stop characters.
Code 39	Code 39 は英数字エンコーディングです。「3 of 9」、「LOGMARS」とも呼ばれます。これは最初に開発された英数字バーコードで、最も広く使用されているエンコーディングの1つです。
Code 39x	Code 39 Extended では2文字エンコーディングが使用されているため、128 個の ASCII 文字すべてがサポートされます。
Codabar	Codabar は 16 種類の文字 (0~9と -\$.:/.+ )と、4種類のスタート/ストップ文字 (A ~ D) をエンコードできます。Codabar は、米国血液バンク、現像所、FedEx の航空貨物受取証などで使用されています。
Code 128A	Code 128 は最も記録密度が高い英数字バーコードです。Code 128A では、ASCII 文字 00 ~ 95



	(0～9、A～Z、制御コード)、特殊文字、および FNC1～4が使用されます
Code 128B	Code 128 は最も記録密度が高い英数字バーコードです。Code 128B では、ASCII 文字 32～127(0～9、A～Z、a～z)、特殊文字、および FNC1～4が使用されます
Code 128C	Code 128 は最も記録密度が高い英数字バーコードです。Code 128C では、00～99(1つのコードで2桁をエンコード)および FNC1 が使用されます。
Code 128 Auto	Code 128 Auto では、最も少ないバー数でデータがエンコードされます。
Code 2 of 5	Code 2 of 5 は、数値のみのバーコードです。Code 2 of 5 では、すべての情報がバーでエンコードされます。バーの間隔は固定されます。
Code 93	Code 93 は Code 39 より少し記録密度が高い英数字エンコーディングです。
Code25intlv	Interleaved Code 2 of 5 では、2桁ずつがエンコードされます。最初の桁は最初の5本のバーに、2つ目の桁は5つのスペースにエンコードされます。
Code39	Code 39 は英数字エンコーディングです。「3 of 9」、「LOGMARS」とも呼ばれます。これは最初に開発された英数字バーコードで、最も広く使用されているエンコーディングの1つです。
Code49	Code 49 はスタック型バーコードで、ASCII 文字セット全体(128 文字)をエンコードすることができます。
Code93x	Code 93 Extended は Code 93 に基づいており、ASCII 文字セット全体(128 文字)をエンコードすることができます。
DataMatrix	Data Matrix は、正方形または長方形のマトリックスパターンに並べられた正方形モジュールから成る2次元高密度バーコードです。
EAN 13	EAN-13 は、欧州の International Article Numbering Association(EAN)によって導入されました。EAN-13 は、2桁のシステムコード、5桁のメーカーコード、5桁の製品コードの順で構成される12 桁のコードをエンコードします。この12 桁のコードの後にチェックサム桁が付加されます(コントロールによって自動的に追加される)。
EAN 8	EAN-8 は小包向けの短いバーコードです。これは、2～3桁のシステムコード、4～5桁の製品コードの順で構成される7桁のコードをエンコードします。この7桁のコードの後にチェックサム桁が付加されます(コントロールによって自動的に追加される)。
EAN128FNC1	EAN128FNC1 は UCC/EAN-128(EAN128)バーコードタイプの1つで、任意の場所に FNC1 文字を挿入したり、バーサイズを調整することができます。FNC1 文字を挿入するには、実行時に Text プロパティに "\n"(C# の場合)または "\vLf"(VB の場合)を設定します。  これは、UCC/EAN-128 では使用できません。
HIBCode39	HIBCCode39 is a Health Industry Bar Code 39 implementation.
HIBCode128	HIBCCode128 is a Health Industry Bar Code 128 implementation.
Iata25	Represents an IATA 2 of 5 barcode.
IntelligentMail	Intelligent Mail(以前の 4-State Customer Barcode)は、米国内の郵便に使用される 65 本のバーコードです。
IntelligentMailPackage	Intelligent Mail Package Barcode.
ISBN	The International Standard Book Number (ISBN) is special commercial book identifier which encodes 9 numeric digits apart from the start number "978", "979".
ISMN	The International Standard Music Number or ISMN (ISO 10957) is a thirteen-character alphanumeric identifier for printed music developed by ISO.
ISSN	The International Standard Serial Number (ISSN) is an eight-digit number used for printed or electronic periodical publications like magazines, etc. This ISSN system was drafted as an


## BarCode for WPF

	International Standard in 1971 and published as ISO 3297 in 1975.
ITF14	ITF14 barcode is the GS1 implementation of an Interleaved 2 of 5 bar code to encode a Global Trade Item Number. It is continuous, self-checking, bidirectionally decodable and it will always encode 14 digits. ITF14 is used on packaging levels of a product in general.
JapanesePostal	これは、日本の郵便制度で使用されるバーコードです。18桁の英数字をエンコードします。これには、7桁の郵便番号の後に、必要に応じて番地や部屋番号などの情報が含まれます。エンコードするデータにはハイフンを含めることができます。
Matrix 2 of 5	Matrix 2 of 5 は、3本の黒バーと2本の白バーから成る比較的高密度のバーコードです。Matrix 2 of 5 は数字だけを使用します。
MicroPDF417	MicroPDF417 は、PDF417 から派生された2次元多段バーコードです。Micro-PDF417 は、データをできる限り小さなサイズの2次元シンボル(最大 150 バイト、英数字 250 文字、数字 366 桁)にエンコードする必要があるアプリケーション用に設計されています。
MicroQRCode	MicroQRCode is a variant of QR Code 2005. Compared with other regular QR Codes, it has only one position detection pattern which reduces the barcode size so that it can be used to applications where the space for barcode image is severely restricted.
MSI	MSI Code は数字だけを使用します。
Pdf417	Pdf417 バーコードは、3～90行で構成されているスタック型の1次元バーコードです。各行は、小さな1次元バーコード状になっています。
Pharmacode	Pharmacode, also known as Pharmaceutical Binary Code, is a barcode standard, 1D barcode that is used in the pharmaceutical manufacturing industry as a packing control system.
Plessey	MSI barcode, also known as Modified Plessey, is a numeric symbology developed by the MSI Data Corporation, which is used primarily for marking retail shelves for inventory control. Though continuous and self-checking, MSI Plessey provides several module checksum situations.
PostNet	PostNet は米国郵便サービスで使用されている数値エンコーディングです。これは、バーの幅ではなく、バーの高さに基づいている点で多くのバーコードとは異なっています。
PZN	PZN or Pharma-Zentral-Nummer is a barcode standard used in the German pharmaceutical industry for identification of medicines and health-care products.
QRCode	QRコードは、機械で読み取ることができるマトリックスバーコードです。QRコードでは、英数字データ、数値情報、バイトデータ、漢字などのさまざまな情報をエンコードすることができます。このバーコードは最大 7,366 文字をエンコードできます。
RM4SCC	RM4SCC は、ロイヤルメールの Cleanmail サービスで使用される情報をエンコードする際に使用されます。RM4SCC では、英数字情報が最大 36 個のシンボル(26 個の文字と 10 個の数値)を使用してエンコードされます。
RSS14	RSS14 は、コンポジットコンポーネント(CC)で拡張された EAN および UPC 情報を小さなスペースにエンコードできる省スペースシンボル(Reduced Space Symbology: RSS)の1つです。このバージョンは、全方向型 POS スキャナと共に使用される 14 桁の EAN.UCC 品目識別用です。
RSS14 Stacked	RSS14Stacked は、コンポジットコンポーネント(CC)で拡張された EAN および UPC 情報を小さなスペースにエンコードできる省スペースシンボル(Reduced Space Symbology: RSS)の1つです。このバージョンは、RSS14Truncated では広すぎる場合に2段に積み重ねられること以外は、RSS14Truncated と同じです。
RSS14 Stacked Omnidirectional	RSS14StackedOmnidirectional は、コンポジットコンポーネント(CC)で拡張された EAN および UPC 情報を小さなスペースにエンコードできる省スペースシンボル(Reduced Space Symbology: RSS)の1つです。このバージョンは、RSS14 では広すぎる場合に2段に積み重ねられること以外

	は、RSS14と同じです。
RSS14 Truncated	RSS14Truncated は、コンポジットコンポーネント(CC)で拡張された EAN および UPC 情報を小さなスペースにエンコードできる省スペースシンボル (Reduced Space Symbology: RSS) の1つです。このバージョンは、小さな品目に使用される、インジケータデジット付きの 14 桁の EAN.UCC 品目識別用です。POS スキャナ用ではありません。
RSS Expanded	RSSExpanded は、コンポジットコンポーネント(CC)で拡張された EAN および UPC 情報を小さなスペースにエンコードできる省スペースシンボル (Reduced Space Symbology: RSS) の1つです。このバージョンは、全方向型 POS スキャナと共に使用される、AI 要素文字列 (有効期限、重量など) 付きの 14 桁の EAN.UCC 品目識別用です。
RSS Expanded Stacked	RSSExpandedStacked は、コンポジットコンポーネント(CC)で拡張された EAN および UPC 情報を小さなスペースにエンコードできる省スペースシンボル (Reduced Space Symbology: RSS) の1つです。このバージョンは、RSSExpanded では広すぎる場合に2段に積み重ねられること以外は、RSSExpanded と同じです。
RSS Limited	RSS Limited は、コンポジットコンポーネント(CC)で拡張された EAN および UPC 情報を小さなスペースにエンコードできる省スペースシンボル (Reduced Space Symbology: RSS) の1つです。このバージョンは、0または1のインジケータデジット付きの 14 桁の EAN.UCC 品目識別用です。POS スキャナでスキャンされない小さなシンボルに使用されます。
SSCC 18	Serial Shipping Container Code-18 (SSCC-18) Barcode is a type of barcode that can print in the lower 2-inch (or local equivalent) extended area of the Thermal 4" x 8" or 4" x 8¼" (or local equivalent) label.
Telepen	Telepen is a name of a barcode symbology designed in the UK, in 1972, to directly represent the full ASCII character set without using shift characters for code switching, and use only two different widths for bars and spaces.
UCC/EAN-128	UCC/EAN-128 は、ASCII 文字セット全体を使用します。HIBC アプリケーションで使用される特殊な Code 128 です。
UPC A	UPC-A は、本、雑誌、新聞のほか、巷のスーパーマーケットの棚にあるほとんどすべての商品に見られる一般的なエンコードです。EAN-13 に似ていますが、11 桁の数値データと末尾にチェックデジットがエンコードされています。
UPC E0	UPC-E0 は数字だけを使用します。Zero Suppression の UPC シンボルに使用されます。Caption プロパティには、6桁の UPC-E コードまたは完全な 11 桁 (必須のコードタイプ0を含む) の UPC-A コードを入力する必要があります。11 桁のコードが入力された場合、バーコードコントロールは、可能であればそれを6桁の UPC-E コードに変換します。11 桁のコードを6桁のコードに変換できない場合は、何も表示されません。
UPC E1	UPC-E1 は数字だけを使用します。主に小売業界で在庫ラベルに使用されます。UPC-E1 の入力文字列の長さは数字6個です。

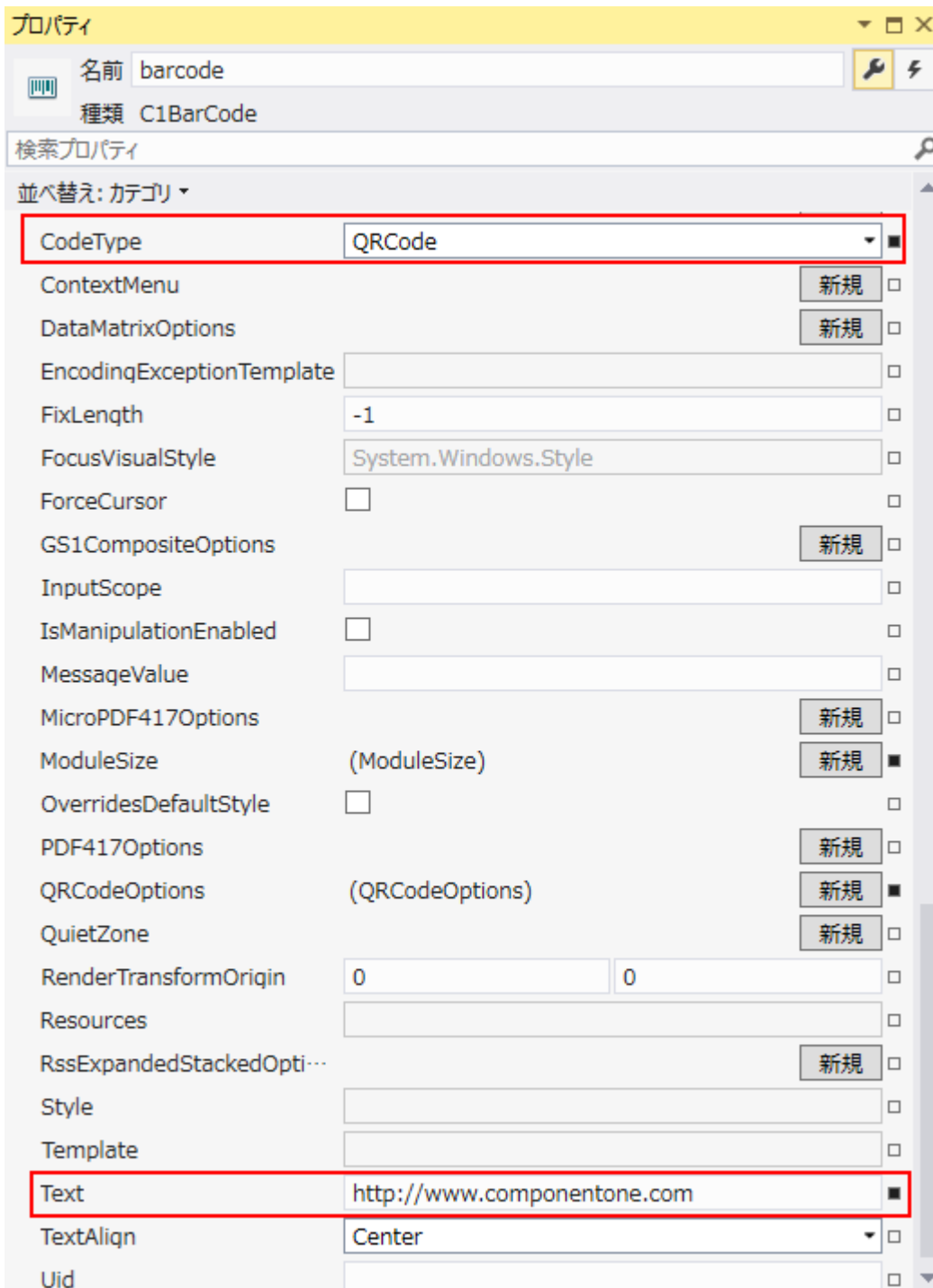
## C1Barcode コントロールのカスタマイズ

**C1Barcode** コントロールを使用するには、使用するエンコーディングのタイプを **CodeType** プロパティで設定し、エンコードする値を **Text** プロパティで設定します。

 **メモ:** 文字要件が極めて少ないエンコーディングもあれば、数値しか使用できないエンコーディングもあります。

次の図は、**CodeType** が **QRCode** に、また **Text** が URL に設定された **C1Barcode** コントロールを示しています。アプリケーションで使用されるバーコードのタイプによっては、さらに多くのオプションをカスタマイズに使用できます。

# BarCode for WPF



## C1Barcode 画像の保存

C1Barcodeコントロールでは、バーコードの画像を保存する機能が用意されています。C1BarcodeクラスのSave()メソッドを使用して画像を.bmp、.png、.jpegのいずれかの形式で保存できます。

C1Barcodeを.pngファイルとして保存するには次の手順を実行します。

この例では、「クイックスタート」トピックで作成されたサンプルを使用します。

1. [ツールボックス]に移動して、デザイナーにボタンコントロールを追加します。これで、XAMLビューに<Button>タグが追加されます。
2. XAMLで追加された<Button>タグのプロパティを以下のように設定して編集します。

```
XAML copyCode  
<Button x:Name="button" Content="Save BarCode Image" HorizontalAlignment="Left" VerticalAlignment="Top"  
        Width="138" Margin="502,510,0,0" RenderTransformOrigin="0.266,0.099" Height="51"/>
```

3. MainWindow()コンストラクタ内にbutton\_Clickイベントを購読します。

- **Visual Basic**  
`AddHandler` button.Click, `AddressOf` button\_Click
- **C#**  
`button.Click += button_Click;`

4. button\_Clickイベントのハンドラーを生成するには、「Tab」キーを2回押します。

5. 以下のコードをbutton\_Clickイベントのハンドラー内に追加します。

- **Visual Basic**  

```
Private Sub button_Click(sender As Object, e As RoutedEventArgs)
    Using stm = System.IO.File.Create("../Pictures/barcode.jpg")
        barcode.Save(stm, ImageFormat.Jpeg)
    End Using
End Sub
```
- **C#**  

```
void button_Click(object sender, RoutedEventArgs e)
{
    using (var stm = System.IO.File.Create("../Pictures/barcode.jpg"))
    {
        barcode.Save(stm, ImageFormat.Jpeg);
    }
}
```

上記のコードでは、ボタンクリックで画像を保存するパスをCreate()メソッドに指定する必要があります。

## QR コード

QRコード(Quick Response コード)形式は、今日最もよく使用されている2次元バーコード形式の1つで、ほとんどすべてのスマートフォンの無料リーダーが対応しています。デンソーウェーブが開発したQRコード形式は、効率がよくコンパクトで、読み取りに特別なスキャナを必要としません。これは、無償で利用できるオープンな規格(ISO/IEC18004など)です。

このコードは、一定のパターンで配列された白黒のピクセルだけで構成されます。C1QRCodeコントロールを使用すると、エンコードする値に基づいてQRパターンが構成されます。この値は、Textプロパティで指定できます。