BarCode for UWP

2018.03.06 更新

グレープシティ株式会社

BarCode for UWP

目次

BarCode for UWP

BarCode for UWP 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。

主な特長

Barcode for UWP のいくつかの機能は次のとおりです。

38 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。

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、新しい Visual Studio 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。

1. 新しい Windows ストアアプリケーションを作成します。
    1. **[ファイル]**、**[新規作成]**、**[プロジェクト]**を順に選択します。**[新しいプロジェクト]**ダイアログボックスが開きます。
    2. **[Visual C#]**の**[テンプレート]**リストで**[ストアアプリ]**を選択します。テンプレートリストで、**[新しいアプリケーション(ユニバーサルアプリケーション)]**を選択します。
    3. アプリケーションに**名前**を付けて、**[OK]**を選択します。新しいアプリケーションが開きます。
2. ソリューションエクスプローラーで**参照設定**フォルダを右クリックし、**[追加]**→**[新しい参照]**を選択します。次の参照を探して追加します。
    - C1.Xaml.dll
    - C1.Xaml.BarCode.dll
3. アプリケーションに「**Resources**」というフォルダを作成し、そこに1つの画像ファイルを追加します。この例では、「c1logo.png」という画像を追加しています。
4. **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。
    1. 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。
    2. 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。
    3. 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。
5. **MainPage.xaml** ファイルを開き、開始タグ **<Page>** を見つけます。このタグには、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

    ```
    <Page
        x:Class="BarCodeQS.MainPage"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:local="using:BarCodeQS"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:c1="using:C1.Xaml.BarCode"
        xmlns:Xaml="using:C1.Xaml"
        xmlns:BarCode="using:C1.BarCode"
        mc:Ignorable="d">
    ```

    次の名前空間がタグに追加されます。

    - xmlns:c1="using:C1.Xaml.BarCode"
    - xmlns:Xaml="using:C1.Xaml"
    - xmlns:BarCode="using:C1.BarCode"

5. ページの **<Grid></Grid>** タグの間にカーソルを置きます。次の XAML マークアップを **<Grid></Grid>** タグの間に追加して、グリッドのリソースを設定します。

    XAML

    ```
    <Grid Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"
    HorizontalAlignment="Left" Width="756">
        <Grid.Resources>
    ```

```
        <Style TargetType="TextBlock">
            <Setter Property="FontSize" Value="26.667"></Setter>
        </Style>
        <Style TargetType="TextBox">
            <Setter Property="FontSize" Value="26.667"></Setter>
        </Style>
        <Style TargetType="ComboBox">
            <Setter Property="FontSize" Value="26.667"></Setter>
        </Style>
    </Grid.Resources>
```

6. 次のマークアップは、4つの **TextBlock** コントロールと1つの **ComboBox** コントロールを追加します。アプリケーションを実行すると、表示する **BarCode** コントロールのタイプを変更することができます。

XAML

```
<TextBlock x:Name="textBlock" HorizontalAlignment="Left" Margin="10,68,0,617"
TextWrapping="Wrap" Text="CodeType:" VerticalAlignment="Center"/>
<TextBox x:Name="text" Text="{Binding Text, ElementName=barcode,
UpdateSourceTrigger=PropertyChanged, FallbackValue='', Mode=TwoWay}"
         HorizontalAlignment="Left" Margin="157,172,0,0" TextWrapping="Wrap"
VerticalAlignment="Top" TextChanged="text_TextChanged" Width="400" Height="57"
/>
<ComboBox x:Name="cbCodeType" HorizontalAlignment="Left" Margin="157,64,0,0"
         VerticalAlignment="Top" Width="400" RenderTransformOrigin="0.66,-4.493"
SelectionChanged="cbCodeType_SelectionChanged" Grid.ColumnSpan="3"/>
<TextBlock x:Name="textBlock1" HorizontalAlignment="Left" Margin="48,182,0,503"
         TextWrapping="Wrap" Text="Text:" VerticalAlignment="Center"/>
<TextBlock x:Name="textBlock2" HorizontalAlignment="Left" Margin="41,421,0,0"
           TextWrapping="Wrap" Text="Barcode:" VerticalAlignment="Top"/>
```

7. **C1BarCode** コントロールのマークアップを最後の **Grid** タグの間に追加します。このマークアップは、アプリケーションの実行時に表示される BarCode のタイプを設定します。

XAML

```
<c1:C1BarCode x:Name="barcode" HorizontalAlignment="Left" Margin="258,373,0,0"
VerticalAlignment="Top" CodeType="QRCode" CaptionPosition="Below" Text="
http://c1.grapecity.com">
    <c1:C1BarCode.QRCodeOptions>
        <BarCode:QRCodeOptions ErrorLevel="High"/>
    </c1:C1BarCode.QRCodeOptions>
</c1:C1BarCode>
```

8. 次に、以下のマークアップを使用して、手順3で追加した画像ファイルを追加します。

XAML

```
<Image Source="ms-appx:/BarCodeQS/Resources/c1logo1.png" x:Name="image"
Width="70" Height="70" Grid.Column="1" Margin="82,316,0,334" />
```

これで 、このクイックスタートの手順1は完了です。この手順で、新しいアプリケーションを作成し、参照を追加し、XAMLマークアップを使用してアプリケーションを設定しました。次の手順では、コードをアプリケーションに追加します。

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。

1. 次の名前空間をページの先頭に追加します。

C#

```
using C1.BarCode;
```

2. **MainPage_Loaded** イベントを **InitializeComponent()** メソッドの直下に登録します。コードは次の図のようになります。

C#

```
this.InitializeComponent();
this.Loaded += MainPage_Loaded;
```

3. 次に、**MainPage_Loaded** イベントを追加します。

C#

```
void MainPage_Loaded(object sender, RoutedEventArgs e)
        {
            cbCodeType.ItemsSource = Enum.GetValues(typeof(CodeType));
            cbCodeType.SelectedItem = barcode.CodeType;
        }
```

4. **SelectionChanged** イベントを追加すると、表示中のバーコードのタイプを変更する際に、表示中のバーコードのタイプがこのコードによって確認されます。バーコードのタイプに応じて、画像が表示または非表示になります。

C#

```
private async void cbCodeType_SelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    try
    {
        if (barcode != null &&
            cbCodeType != null &&
            cbCodeType.SelectedItem != null)
        {
            barcode.CodeType = (CodeType)cbCodeType.SelectedItem;
        }
        if (barcode.CodeType != CodeType.QRCode
                    || !text.Text.Equals(https://www.grapecity.co.jp))
        {
            image.Opacity = 0;
        }
        else
        {
            image.Opacity = 1;
        }
    }
    catch (Exception ex)
```

```
    {
        await new Windows.UI.Popups.MessageDialog(ex.Message,
"Error").ShowAsync();
    }
}
```

5. この **TextChanged** イベントは、**C1BarCode** コントロールでエンコードされるテキストを制御します。さらに、QRCode タイプのバーコードが選択された場合は、選択した画像がバーコードの上に表示されます。

C#

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

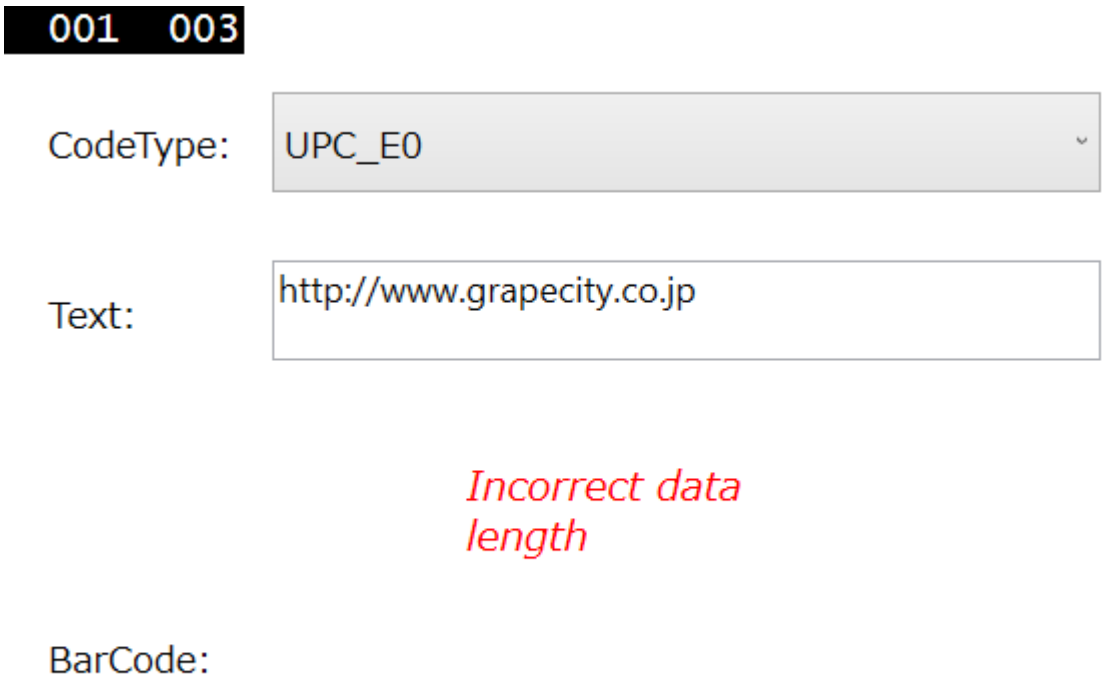
この手順では、**C1BarCode** コントロールの外観を制御するコードを追加しました。次の手順では、このアプリケーションを実行します。

手順 3:アプリケーションの実行

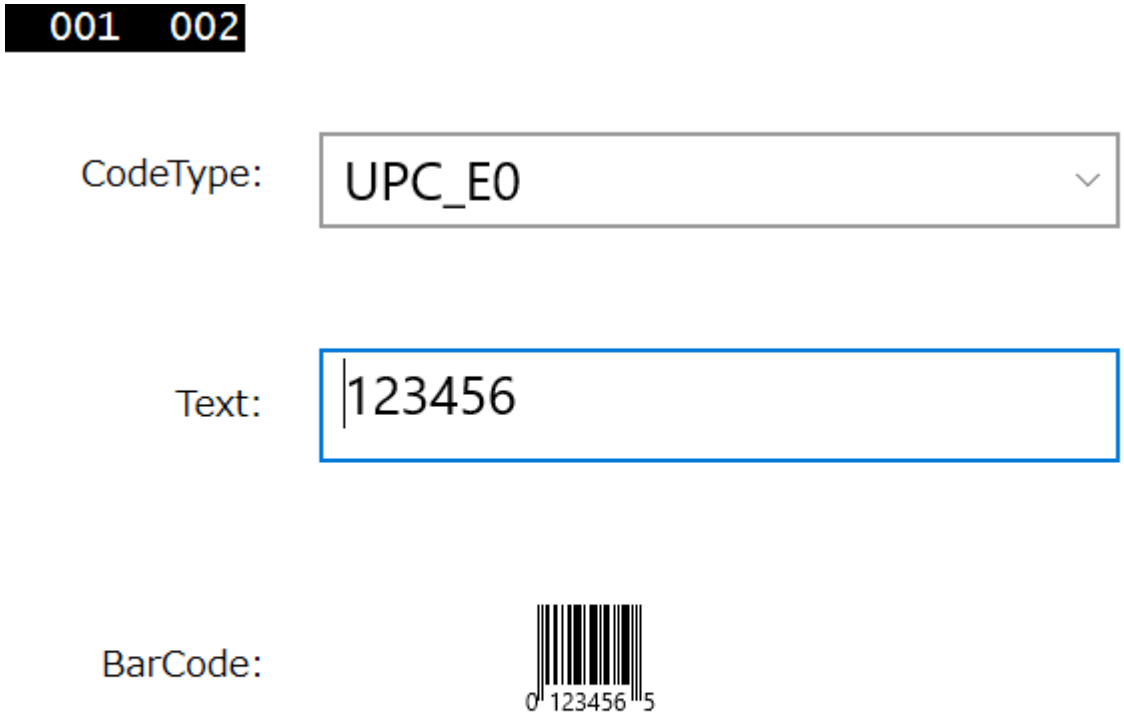
最後の手順では、アプリケーションの外観を制御するコードを追加しました。この手順では、作成したアプリケーションを実行します。

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。

C1BarCode 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
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 "vbLf"（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。
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。
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。
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。
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。
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。
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。

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。