ComponentOne

BarCode for UWP

ComponentOne, a division of GrapeCity
201 South Highland Avenue, Third Floor
Pittsburgh, PA 15206 USA

Website: http://www.componentone.com
Sales: sales@componentone.com
Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

Trademarks

Warranty

ComponentOne warrants that the media on which the software is delivered is free from defects in material and workmanship, assuming normal use, for a period of 90 days from the date of purchase. If a defect occurs during this time, you may return the defective media to ComponentOne, along with a dated proof of purchase, and ComponentOne will replace it at no charge. After 90 days, you can obtain a replacement for the defective media by sending it and a check for $2 5 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original media on which the software is delivered is set forth here, ComponentOne makes no other warranties, express or implied. Every attempt has been made to ensure that the information contained in this manual is correct as of the time it was written. ComponentOne is not responsible for any errors or omissions. ComponentOne's liability is limited to the amount you paid for the product. ComponentOne is not liable for any special, consequential, or other damages for any reason.

Copying and Distribution

Table of Contents

BarCode for UWP Edition

Add barcode images to your applications with **BarCode for UWP Edition**.

Unlike barcode fonts, BarCode for UWP Edition automatically adds the necessary control symbols and checksums to the value being encoded, depending on the encoding being used, to eliminate reader errors.

BarCode for UWP Edition is easy to use – simply add the control to your form, set the encoding type, and you're done.

Getting Started with UWP Edition

Help with UWP Edition

Getting Started

For information on installing **ComponentOne Studio UWP Edition**, licensing, technical support, namespaces and creating a project with the control, please visit Getting Started with ComponentOne Studio UWP Edition.

Key Features

Some of the features of **Barcode for UWP** are as follows:

- **Supports 38 different encodings**

  The BarCode control supports nearly 38 encodings, including Codabar, Code128 Auto, Code39, Code93, DataMatrix, Ean13, Ean8, PostNet, QRCode, and RSS14.

- **Automatically adds checksums**

  The BarCode control automatically adds the necessary control symbols and checksums to the value being encoded, depending on the encoding being used, to guarantee a good read on your barcodes.

Quick Start

Step 1 of 3: Setting Up the Application

In this step, you will create a new Visual Studio application, add the appropriate references for the project, and add XAML markup to create the C1Barcode control.

1. Create a new Universal Windows application:
    1. Select **File | New | Project**. The **New Project** dialog box will open.
    2. Select Templates | Visual C# | Windows | Universal. From the templates list, select **Blank App (Universal Windows)**.
    3. Give your application a **Name** and select **OK**. Your new application will open.
2. Right-click the **References** folder in the Solution Explorer and select **Add | New Reference**. Expand Universal Windows and select Extensions; you should see the UWP assemblies in the center pane. Select the following assemblies and click OK:
    - C1.UWP.dll
    - C1.UWP.BarCode.dll
3. Create a folder named **Resources** in your application. Add an image file in this folder. In this example, we are adding c1logo.png.
4. Right-click the **Resources** folder and select **Add | Existing Item**. The **Add Existing Item** dialog box will open.
    1. Locate the image file you would like to add to your application.
    2. Select the file and click **OK**. The file will be added to the **Resources** folder.
    3. Rebuild the application so that your file is available to your application.
5. Open the **MainPage.xaml** file and locate the opening **<Page>** tag. This tag will include the necessary namespaces. Edit the tag so that it resembles the following markup:

```
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">
  <!-- BarCode.QS is the name of the application, this will be replaced by your app's name -->
```

    The following namespaces have been added to the tag:

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

6. Place your cursor between the **<Grid></Grid>** tags on the page. Add the following XAML markup between the **<Grid></Grid>** tags to set up the Grid's resources:

```
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>
```

7. The following markup adds three **TextBlock** controls, one **TextBox** and one **ComboBox** control. When you run your application, you will be able to change the type of **BarCode** control being displayed:

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"/>
```

8. Next, add the markup for the **C1BarCode** control, below the markups added in previous step. This markup will set the type of BarCode the application will display when you run it:

XAML

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

9. Now, use the below markup to add the image file, which you added in step 3:

XAML

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

You have completed Step 1 of this Quick Start. In this step, you created a new application, added references, and used XAML markup to set up your application. In the next step, you will add code to your application.

Step 2 of 3: Adding Code

In this step, you will add the code needed for your application.

1. Add the following namespace to the top of your page:

C#

```
using C1.BarCode;
```

2. Register a **MainPage_Loaded** event directly below the **InitializeComponent()** method. Your code should resemble the following:

C#

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

3. Next, add a **MainPage_Loaded** event:

C#

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

4. By adding a **SelectionChanged** event, when you change the type of barcode you are displaying, the code will check to see what type of barcode is being displayed. Based on the type of barcode, the application will either show or hide the image:

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("http://www.componentone.com"))
        {
            image.Opacity = 0;
        }
        else
        {
            image.Opacity = 1;
        }
    }
    catch (Exception ex)
    {
        await new Windows.UI.Popups.MessageDialog(ex.Message,
"Error").ShowAsync();
```

```
    }
}
```

5. The **TextChanged** event controls the text that is encoded in the **C1BarCode** control. In addition to this, if you choose a QRCode type barcode, then the image you chose will appear over the barcode:

C#

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

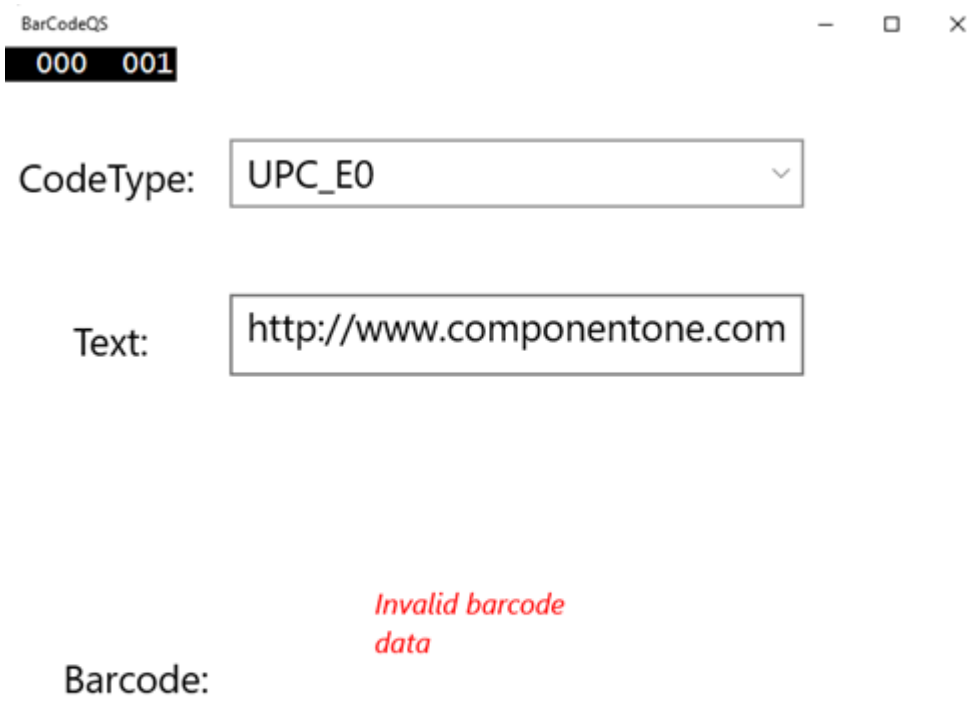
In this step, you added code to control the appearance of the **C1BarCode** control. In the next step, you will run the application.

Step 3 of 3: Running Your Application

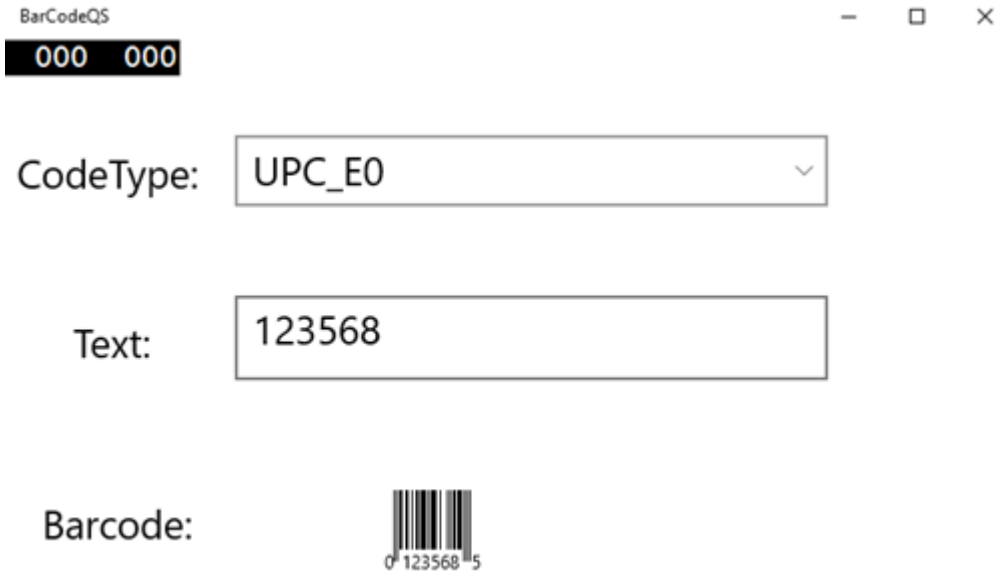
In the last step, you added code to control the appearance of your application. In this step, you will run your application.

1. Select **Start Debugging** or press **F5** to start your application. It should resemble the following image:

2. Select a new **CodeType** from the **CodeType** drop down list. Since the text is set to a URL, for many of the CodeTypes, there will be an error message:



3. Finally, change the value in the **Text** TextBox:



What You've Accomplished

In this Quick Start, you created a new Visual Studio application, added XAML markup to create the framework for your application, and added code to control **TextChanged** and **SelectionChanged** events.

Using BarCode for UWP Edition

Supported Encodings

You can change the C1BarCode encoding type by setting the CodeType property. The **C1BarCode** control supports the following encodings:

Style Name	Example	Description
Ansi39		ANSI 3 of 9 (Code 39) uses upper case, numbers, - , * $ / + %. This is the default barcode style.
Ansi39x		ANSI Extended 3 of 9 (Extended Code 39) uses the complete ASCII character set.
Codabar		Codabar uses A B C D + - : . / $ and numbers.
Code_128_A		Code 128 A uses control characters, numbers, punctuation, and upper case.
Code_128_B		Code 128 B uses punctuation, numbers, upper case and lower case.
Code_128_C		Code 128 C uses only numbers.
Code_128auto		Code 128 Auto uses the complete ASCII character set. Automatically selects between Code 128 A, B and C to give the smallest barcode.
Code_2_of_5		Code 2 of 5 uses only numbers.
Code93		Code 93 uses uppercase, % $ * / , + -, and numbers.

Code25intlv		Interleaved 2 of 5 uses only numbers.
Code39		Code 39 uses numbers, % * $ /. , - +, and upper case.
Code39x		Extended Code 39 uses the complete ASCII character set.
Code49		Code 49 is a 2D high-density stacked barcode containing two to eight rows of eight characters each. Each row has a start code and a stop code. Encodes the complete ASCII character set.
Code93x		Extended Code 93 uses the complete ASCII character set.
DataMatrix		Data Matrix is a high density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern.
EAN_13		EAN-13 uses only numbers (12 numbers and a check digit). It takes only 12 numbers as a string to calculate a check digit (CheckSum) and add it to the thirteenth position. The check digit is an additional digit used to verify that a barcode has been scanned correctly. The check digit is added automatically when the CheckSum property is set to True.
EAN8		EAN-8 uses only numbers (7 numbers and a check digit).
EAN128FNC1		EAN-128 is an alphanumeric one-dimensional representation of Application Identifier (AI) data for marking containers in the shipping industry. This type of barcode contains the following sections:

		• Leading quiet zone (blank area) • Code 128 start character • FNC (function) 1 character which allows scanners to identify this as an EAN-128 barcode • Data (AI plus data field) • Symbol check character (Start code value plus product of each character position plus value of each character divided by 103. The checksum is the remainder value.) • Stop character • Trailing quiet zone (blank area) The AI in the Data section sets the type of the data to follow (i.e. ID, dates, quantity, measurements, etc.). There is a specific data structure for each type of data. This AI is what distinguishes the EAN-128 code from Code 128. Multiple AIs (along with their data) can be combined into a single barcode. EAN128FNC1 is a UCC/EAN-128 (EAN128) type barcode that allows you to insert FNC1 character at any place and adjust the bar size, etc., which is not available in UCC/EAN-128. To insert FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at runtime.
IntelligentMail		Intelligent Mail, formerly known as the 4-State Customer Barcode, is a 65-barcode used for domestic mail in the U.S.
JapanesePostal		This is the barcode used by the Japanese Postal system. Encodes alpha and numeric characters consisting of 18 digits including a 7-digit postal code number, optionally followed by block and house number information. The data to be encoded can include hyphens.
Matrix_2_of_5		Matrix 2 of 5 is a higher density barcode consisting of 3 black bars and 2 white bars.

MicroPDF417		MicroPDF417 is two-dimensional (2D), multi-row symbology, derived from PDF417. Micro-PDF417 is designed for applications that need to encode data in a two-dimensional (2D) symbol (up to 150 bytes, 250 alphanumeric characters, or 366 numeric digits) with the minimal symbol size. MicroPDF417 allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs). To insert FNC1 character, set "\n" for C#, or "vbLf" for VB to Text property at runtime.
MSI		MSI Code uses only numbers.
Pdf417		Pdf417 is a popular high-density 2-dimensional symbology that encodes up to 1108 bytes of information. This barcode consists of a stacked set of smaller barcodes. Encodes the full ASCII character set. It has ten error correction levels and three data compaction modes: Text, Byte, and Numeric. This symbology can encode up to 1,850 alphanumeric characters or 2,710 numeric characters.
PostNet		PostNet uses only numbers with a check digit.
QRCode		QRCode is a 2D symbology that is capable of handling numeric, alphanumeric and byte data as well as Japanese kanji and kana characters. This symbology can encode up to 7,366 characters.
RM4SCC		Royal Mail RM4SCC uses only letters and numbers (with a check digit). This is the barcode used by the Royal Mail in the United Kingdom.
RSS14		RSS14 is a 14-digit Reduced Space Symbology that uses EAN.UCC item identification for point-of-sale omnidirectional scanning.

RSS14Stacked		RSS14Stacked uses the EAN.UCC information with Indicator digits as in the RSS14Truncated, but stacked in two rows for a smaller width. RSS14Stacked allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.
RSS14StackedOmnidirectional		RSS14StackedOmnidirectional uses the EAN.UCC information with omnidirectional scanning as in the RSS14, but stacked in two rows for a smaller width.
RSS14Truncated		RSS14Truncated uses the EAN.UCC information as in the RSS14, but also includes Indicator digits of zero or one for use on small items not scanned at the point of sale.
RSSExpanded		RSSExpanded uses the EAN.UCC information as in the RSS14, but also adds AI elements such as weight and best-before dates. RSSExpanded allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs). To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime.
RSSExpandedStacked		RSSExpandedStacked uses the EAN.UCC information with AI elements as in the RSSExpanded, but stacked in two rows for a smaller width. RSSExpandedStacked allows you to insert an FNC1 character as a field separator for variable length Application Identifiers (AIs). To insert FNC1 character, set “\n” for C#, or “vbLf” for VB to Text property at runtime.

RSSLimited		RSS Limited uses the EAN.UCC information as in the RSS14, but also includes Indicator digits of zero or one for use on small items not scanned at the point of sale. RSSLimited allows you to set Composite Options, where you can select the type of the barcode in the **Type** drop-down list and the value of the composite barcode in the **Value** field.
UCCEAN128		UCC/EAN –128 uses the complete ASCII character Set. This is a special version of Code 128 used in HIBC applications.
UPC_A		UPC-A uses only numbers (11 numbers and a check digit).
UPC_E0		UPC-E0 uses only numbers. Used for zero-compression UPC symbols. For the Caption property, you may enter either a six-digit UPC-E code or a complete 11-digit (includes code type, which must bezero) UPC-A code. If an 11-digit code is entered, the Barcode control will convert it to a six-digit UPC-E code, if possible. If it is not possible to convert from the 11-digit code to the six-digit code, nothing is displayed.
UPC_E1		UPC-E1 uses only numbers. Used typically for shelf labeling in the retail environment. The length of the input string for U.P.C. E1 is six numeric characters.

Note that the following barcodes support FNC1 characters:

- EAN128FNC1
- MicroPDF417
- RSSExpanded
- RSSExpandedStacked

Customizing the C1BarCode Control

To use the C1BarCode control, set the CodeType property to the type of encoding you want to use, then set the Text property to the value you want to encode.

Note: Some encodings have a minimum character requirement, while others will only work with numeric values.

The following image shows the C1BarCode control set to the **QRCode** CodeType, and the Text set to an URL. Depending on the type of barcode used in an application, there will be more options available for customization.