ComponentOne

BarCode for WinForms

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

Copying and Distribution

Table of Contents

BarCode for WinForms Overview

The new barcode engine provides a stand-alone control, C1BarCode, that implements more standard barcode types than the old version. The new **C1.Win.Barcode** assembly replaces **C1BarCode** and **C1QRCode** controls with a single **C1BarCode** control that uses a different codebase.

In case if you are not ready to upgrade, the old C1.Win.C1Barcode assembly remains included into our toolset. You can check the documentation for the old barcode here.

Unlike barcode fonts, **BarCode for WinForms** automatically adds any necessary control symbols and checksums to the value being encoded, depending on the encoding being used, to eliminate reader errors. You can use several standard barcodes for adding barcode images to grid cells, to Web pages, or to regular .NET PrintDocument objects. You can also deploy **BarCode for WinForms** with your applications like any regular assembly. Because it is a royalty-free DLL, you do not have to worry about installing barcode fonts on the client-side and making sure they are royalty free. And **BarCode for WinForms** is so easy to use – just add the control to your form, set the encoding type, and you are done!

Help with WinForms Edition

Getting Started

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

Key Features

- **Supports 38 BarCode Symbologies**

  The C1BarCode control supports 38 standard barcodes for **Windows Forms** applications. It also offers several properties that can be set for rendering as well as customizing barcodes. The FNC1 characters are also supported in some barcodes. See Using BarCode for WinForms for more information.

- **Integrated QRCode Format**

  The QR code (Quick Response code) format is one of the most popular 2D barcode formats available today, with free readers available for virtually all smart phones. The **C1BarCode** control provides the functionality of QR code, so you don't need to add any additional control.

- **Automatically Adds Checksums**

  The **C1BarCode** control automatically adds necessary control symbols and checksums to the value being encoded, depending on the symbology being used, to guarantee a good read on your barcodes.

- **Royalty-free DLL for Easy Deployment**

  The **C1BarCode** is a royalty-free DLL that can be deployed with your applications like any regular assembly.

BarCode for WinForms Quick Start

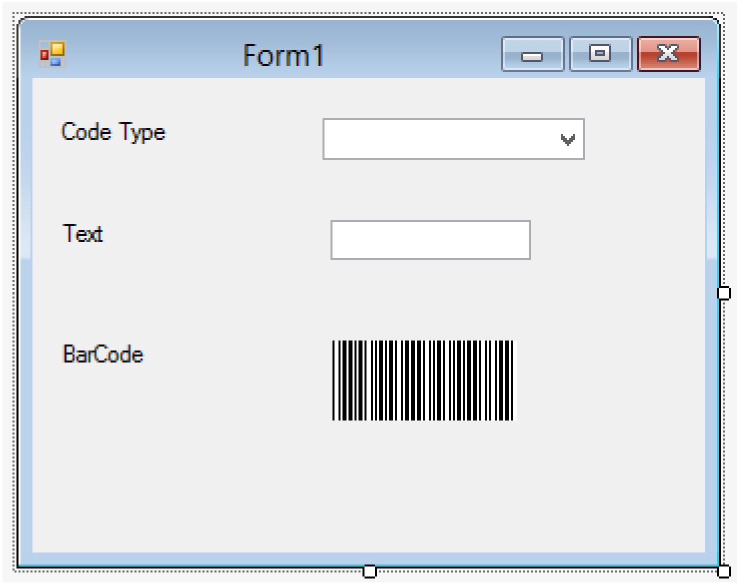
This quick start section walks through the steps of adding C1BarCode to your project and creating a simple barcode application. The quick start demonstrates how various symbologies available in **C1BarCode** are rendered on changing the text and the barcode type. For more information about available symbology, see BarCode Symbology.

Step 1: Setting Up the Project

To set up the project and add the C1BarCode control to the form, complete the following steps:

1. Create a new .NET project.
2. From the Toolbox, add the following controls:
    - C1BarCode
    - ComboBox (to contain the list of barcodes available in C1BarCode control)
    - TextBox (to enter the text for observing barcodes)
    - Label (add three labels for displaying name of the above controls)

   You will notice that on adding **C1BarCode** control, **C1.Win.BarCode** and **C1.Win** assemblies get added to the **References** folder in the project. If you do not see **C1BarCode** control in your Toolbox, do the following:
    1. In **Toolbox**, right-click a tab and select **Choose Items** option to open the **Choose Toolbox Items** wizard.
    2. Click **Browse**. Navigate through the default installation path: C:\Program Files (x86)\ComponentOne\WinForms Edition\bin\v4.0, select **C1.Win.BarCode.4 dll**, and then click **OK**.
3. Click the **ComboBox** and in the **Properties** window, rename it as cbCodeType.
4. Set the **Text** property of the three **Label** controls as Code Type, Text, and BarCode, respectively. Place the controls on the form as shown.



You have just completed the first step toward creating a simple application on **BarCode for WinForms**.

Step 2: Adding Code to the Project

To add code to your project, complete the following steps:

1. Add namespace **Imports C1.BarCode** (Visual Basic projects) or **using C1.BarCode** (C# projects).
2. Double-click **Form1** to create **Form1_Load** event and switch to the code view. To set the default text of the **TextBox** and to populate the **ComboBox** control with the code types available in C1BarCode, add the

following code.

Visual Basic copyCode

```
Dim types As Array = [Enum].GetValues(GetType(CodeType))
Private Sub Form1_Load(sender As Object, e As EventArgs) Handles MyBase.Load
    tbText.Text = "HELLO WORLD!"
    cbCodeType.DataSource = types
    cbCodeType.SelectedIndex = Array.IndexOf(types, CodeType.Code39)
    cbCodeType.Select()
End Sub
```

C# copyCode

```
Array types = Enum.GetValues(typeof(CodeType));
 private void Form1_Load(object sender, EventArgs e)
{
    tbText.Text = "HELLO WORLD!";
    cbCodeType.DataSource = types;
    cbCodeType.SelectedIndex = Array.IndexOf(types, CodeType.Code39);
    cbCodeType.Select();
}
```

3. Click **ComboBox** on the form. From the events in the Property window, double-click **SelectedIndexChanged** to create **cbCodeType_SelectedIndexChanged** event and add following code.

Visual Basic copyCode

```
Private Sub cbCodeType_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles cbCodeType.SelectedIndexChanged
    C1BarCode1.CodeType = DirectCast(cbCodeType.SelectedValue, CodeType)
End Sub
```

C# copyCode

```
private void cbCodeType_SelectedIndexChanged(object sender, EventArgs e)
{
    c1BarCode1.CodeType = (CodeType)cbCodeType.SelectedValue;
}
```

4. Click the **TextBox** control on the form. From the events in the Property window, double-click **TextChanged** to create the **tbText_TextChanged** event and add following code to the event.

Visual Basic copyCode

```
Private Sub tbText_TextChanged(sender As Object, e As EventArgs) Handles
tbText.TextChanged
    C1BarCode1.Text = tbText.Text
End Sub
```

C# copyCode

```
private void tbText_TextChanged(object sender, EventArgs e)
{
    c1BarCode1.Text = tbText.Text;
}
```

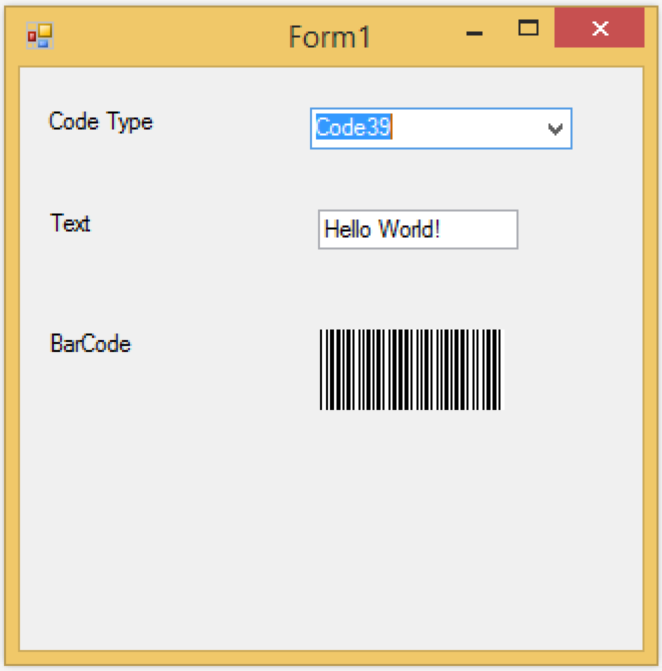
In this step, you added functionality to the controls. In the next step, you will run the project and observe runtime interactions.

Step 3: Running the Project

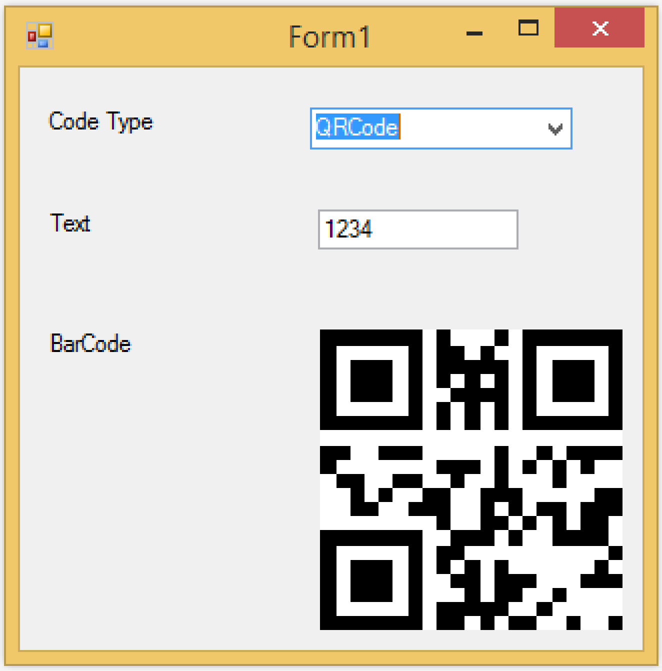
Now that you set up the project and added code, let's run the project to view code types supported by **BarCode for WinForms**.

Complete the following steps:

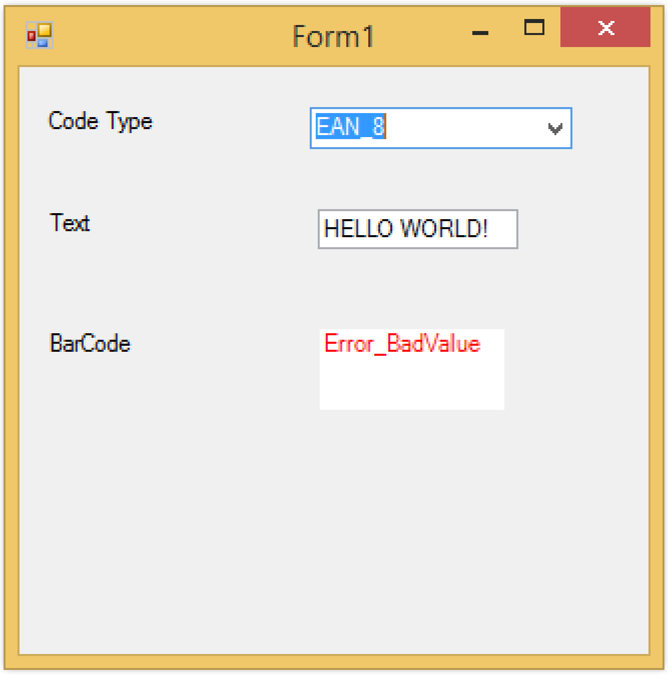
1. Run the project and select code type from the **ComboBox**.



2. Select different code types and notice the corresponding barcodes displayed.



3. Try entering different text strings of alpha numeric characters. If a text is not supported by the barcode, an error message is displayed as in image below.



Code Type EAN_8

Text HELLO WORLD!

BarCode Error_BadValue

Congratulation! You have completed the quick start in which you created a simple application for viewing different barcodes supported in **BarCode for WinForms**.

Note that some encodings have a minimum character requirement, while others work only with numeric values. For more information about supported barcode symbologies, see BarCode Symbology.

Using BarCode for WinForms

The new barcode engine lets you integrate various industry-standard barcodes into the **Windows Forms** applications. For understanding the working of barcodes, it is important to have knowledge of the barcode symbologies and the properties or options associated with them.

BarCode Symbology

The barcode symbology specifies the encoding scheme used to convert character data into the pattern of wide and narrow bars, and spaces in a barcode. The following table illustrates the barcode symbology provided by CodeType property in **BarCode for WinForms**.

Code Type	Example	Description
Ansi39	1234ABZ%	ANSI 3 of 9 (Code 39) uses upper case, numbers, - , * $ / + %. This is the default barcode style.
Ansi39x	11023OPA	ANSI Extended 3 of 9 (Extended Code 39) uses the complete ASCII character set.
Codabar	A4016B	Codabar uses A B C D + - : . / $ and numbers.
Code_128_A	MOU12DEF	Code 128 A uses control characters, numbers, punctuation, and upper case. Enabling Checksum is mandatory for this barcode type.
Code_128_B	MOU11DEX	Code 128 B uses punctuation, numbers, upper case and lower case. Enabling Checksum is mandatory for this barcode type.
Code_128_C	01143493	Code 128 C uses only numbers. Enabling Checksum is mandatory for this barcode type.
Code_128auto	1143493	Code 128 Auto uses the complete ASCII character set. Automatically selects between Code 128 A, B, and C to give the smallest barcode. Enabling Checksum is mandatory for this barcode type.

Code_2_of_5	3661239	Code 2 of 5 uses only numbers.
Code93	MSU 09382	Code 93 uses uppercase, % $ * / , + -, and numbers. Enabling Checksum is mandatory for this barcode type.
Code25intlv	1023392210	Interleaved 2 of 5 uses only numbers.
Code39	AUI%89032	Code 39 uses numbers, % * $ /. , - +, and upper case.
Code39x	BAR92112234	Extended Code 39 uses the complete ASCII character set.
Code49	1293829ABJISSH92K234	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	CODE349101%	Extended Code 93 uses the complete ASCII character set. Enabling Checksum is mandatory for this barcode type.
DataMatrix		Data Matrix is a high density, two-dimensional barcode with square modules arranged in a square or rectangular matrix pattern.
EAN_13	1 452736 201234	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 bar code has been scanned correctly. The check digit is added automatically when the CheckSum property is set to True.

EAN_8	2982 7367	EAN-8 uses only numbers (7 numbers and a check digit). Enabling Checksum is mandatory for this barcode type.
EAN128FNC1	(01)01228(15)0231	EAN-128 is an alphanumeric one-dimensional representation of Application Identifier (AI) data for marking containers in the shipping industry. This type of bar code 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 bar code. 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-bar code 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 20 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	790022312	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	80523	MSI Code uses only numbers. Enabling Checksum is mandatory for this barcode type.
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. Enabling Checksum is mandatory for this barcode type.

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. Enabling Checksum is mandatory for this barcode type.
RSS14	(01)13393821228905	RSS14 is a 14-digit Reduced Space Symbology that uses EAN.UCC item identification for point-of-sale omnidirectional scanning.
RSS14Stacked	(01)03939382122899	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	(01)01339382122891	RSS14StackedOmnidirectional uses the EAN.UCC information with omnidirectional scanning as in the RSS14, but stacked in two rows for a smaller width.
RSS14Truncated	(01)30944382332892	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	8110100706401002003100110120	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	81101007064010020031001101 20	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	(01)00006569232216	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	BARCODE2312	UCC/EAN –128 uses the complete ASCII character Set. This is a special version of Code 128 used in HIBC applications. Enabling Checksum is mandatory for this barcode type.
UPC_A	8 80087 25991 7	UPC-A uses only numbers (11 numbers and a check digit). Enabling Checksum is mandatory for this barcode type.
UPC_E0	0 534729 2	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	1 098672 7	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:

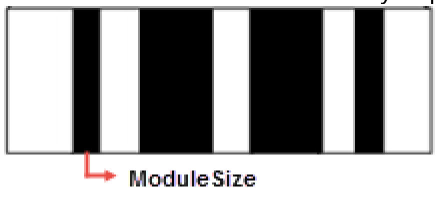
- EAN128FNC1
- MicroPDF417
- RSSExpanded
- RSSExpandedStacked

BarCode Options

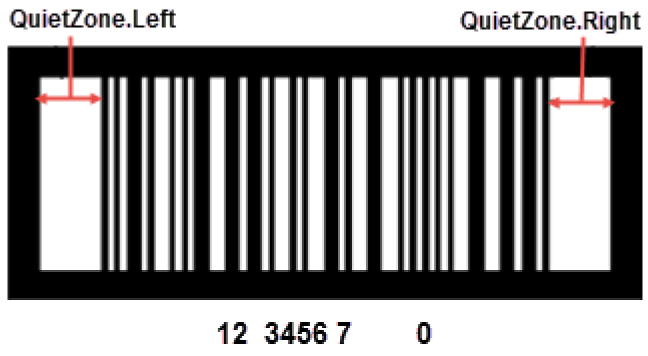
The C1BarCode provides several options that are common to all barcodes or specific to certain barcodes. These options are used to customize the appearance of **C1BarCode**.

The common options exposed by **C1.Win.BarCode** assembly for rendering barcodes in WinForms are as follows:

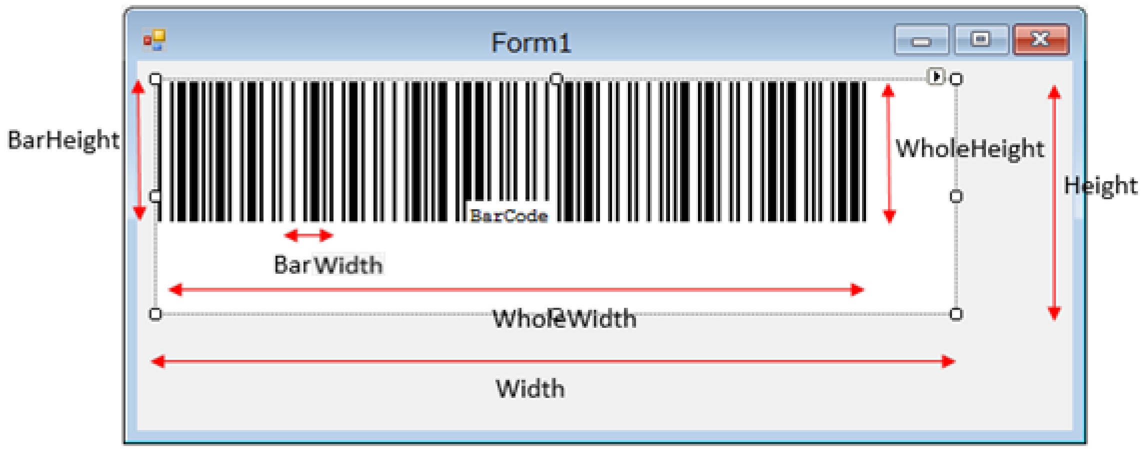
- **BarDirection:** Lets you select the barcode's direction. The available options are:
  - LeftToRight: The barcode symbol is printed left to right (default).
  - RightToLeft: The barcode symbol is printed right to left.
  - TopToBottom: The barcode symbol is printed top to bottom.
  - BottomToTop: The barcode symbol is printed bottom to top.
- **BarHeight:** Lets you specify the height of a barcode in screen pixels. If the bar height exceeds the height of the control, this property is ignored.
- **CodeType:** Lets you select encoding that should be applied to the value stored in the Text property to generate the barcode image.
- **CaptionAlignment:** Lets you select the display position of the value of barcode. The available options are Left, Right, and Center.
- **CaptionGrouping:** Lets you specify a value indicating whether to add spaces between groups of characters in the caption to make long numbers easier to read.
- **CaptionPosition**: Lets you select the caption's vertical position relative to the barcode symbol. The available options are None, Above, and Below.
- **Image:** Gets an image of the barcode that represents the value in the Text property, obtained using the encoding specified by the CodeType property.
- **ModuleSize:** Lets you specify the module (narrowest bar width) of a barcode in screen pixels. The width of wide bars is counted automatically depending on the barcode type.

- **QuietZone**: Lets you specify the quiet zone(s) in a barcode. A quiet zone is an area of blank space on either side of a barcode that tells the scanner where the symbology starts and stops. The options available are as follows:
  - Left: Enter the size of blank space to leave to the left of the barcode.
  - Right: Enter the size of blank space to leave to the right of the barcode.
  - Top: Enter the size of blank space to leave at the top of the barcode.
  - Bottom: Enter the size of blank space to leave at the bottom of the barcode.
    The following image shows Left and Right quiet zones:



- **Text**: Lets you specify the value that is encoded as a barcode image.
- **WholeSize:** Lets you specify the size of the overall barcode. WholeWidth represents the width and WholeHeight represents the height of the overall barcode.



- **FixLength**: Lets you specify the fixed number of digits of values of the barcode. It takes the integer value.
- **AutoSize:** Lets you specify whether the barcode should stretch to fit the control. It takes the value True or False.
  When AutoSize is set to True,
  - the barcode automatically stretches to fit the control.
  - the readable size is calculated by the barcode itself.
  - the size of Matrix barcodes is calculated by **OnCalculateSize** method.
  - the size of the non-matrix bardodes, it is calculated by **BarHeight** and **ModuleSize**.

  When AutoSize is set to False,

  - the size size of the barcode is determined by Width or Height properties.
  - the control gets clipped if the BarHeight is larger than control's height
  - some empty space between the barcode and the control is left if the BarHeight is smaller than height,

The options that are specific to the type of barcodes are as follows:

CheckSumEnabled: Lets you specify whether the check digits are automatically added or not. When data to be bound already includes check digits, programmers sometimes want to prevent controls from automatically including them. This property is supported for Code49, Code128, PostNet5/9/11, and JapanesePostal barcodes.

Ean128Fnc1Options:

- **Dpi:** Lets you specify the resolution of the printer. It takes the integer value.
- **BarAdjust:** Lets you specify the adjustment size by dot.
- **ModuleSize:** Lets you specify the horizontal size of the barcode module. It takes the integer value.

Code25intlvOptions:

- **BearBar:** Lets you select whether or not to display bearer bar to ITF (Interleaved Two of Five) barcode. It takes the value True or False.
- **LineStroke:** Lets you select the color of the bearer bar.
- **LineStrokeThickness:** Lets you select the line width of the bearer bar. It takes the integer values.

Code49Options:

- **Grouping:** Lets you use grouping in the barcode. Its value is either True or False.
- **Group:** Obtains or sets group numbers for barcode grouping. Its value is between 0 and 8. If the value of **Grouping** is True, the range of value of Group is from 0 to 8. If the value of Grouping is False, value of Group is 0. If the value of **Grouping** is True, and the **Group** value is smaller than 0 or larger than 8, the BarCodeException.EnumErrorCode.Code49GroupNo will be thrown.

DataMatrixOptions:

- **EccMode:** Lets you select the ECC mode. The possible values are ECC000, ECC050, ECC080, ECC100, ECC140, or ECC200.
- **Ecc200SymbolSize:** Lets you select the size of ECC200 symbol. The default value is SquareAuto.
- **Ecc200EncodingMode:** Lets you select the ECC200 encoding mode. The possible values are Auto, ASCII, C40, Text, X12, EDIFACT, or Base256.
- **Ecc000_140SymbolSize:** Lets you select the size of the ECC000_140 symbol.
- **StructuredAppend:** Lets you select whether the current barcode symbol is part of structured append symbols.
- **StructureNumber:** Lets you specify the structure number of current symbol within the structured append symbols. The range of this value is from 0 to 15.
- **FileIdentifier:** Lets you specify the file identifier of a related group of structured append symbols. The valid file indentifier value should be within [1,254]. Setting file identifier to 0 lets the file identifier to be calculated automatically.

GS1CompositeOptions:

- **Type:** Lets you select the composite symbol type. Its value can be None or CCA. CCA (Composite Component - Version A) is the smallest variant of the 2-dimensional composite component.
- **Value:** Lets you specify the CCA character data.

MicroPDF417Options:

- **CompactionMode:** Lets you select the type of CompactionMode. The possible values are Auto, TextCompactionMode, NumericCompactionMode, and ByteCompactionMode.
- **FileID:** Lets you specify the file id of structured append symbol. It takes the value from 0 to 899. If this value is smaller than 0 or larger than 899, the BarCodeException.EnumErrorCode.MicroPDF417FileID is thrown.
- **SegmentCount:** Lets you specify the segment count of structured append symbol. It takes the value from 0 to 99999. If this value is smaller than 0 or larger than 99999, the BarCodeException.EnumErrorCode.MicroPDF417SegmentCount is thrown.
- **SegmentIndex:** Lets you specify the segment index of structured append symbol. It takes the value from 0 to 99998 and less than the value of segment count. If this value is smaller than 0 or larger than 99998, the BarCodeException.EnumErrorCode.MicroPDF417SegmentIndex is thrown.
- **Version:** Lets you select the symbol size. The default value is ColumnPriorAuto.

PDF417Options:

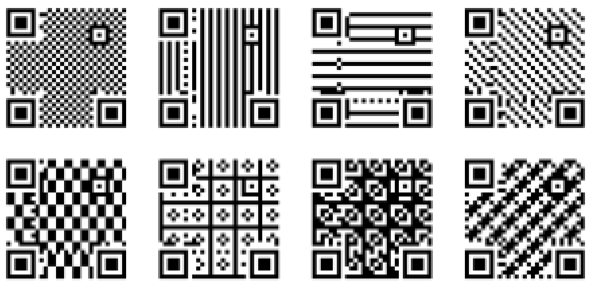
- **Column:** Lets you specify the column numbers for the barcode. It takes the integer value; the default value is -1 and the range of this value is 1 to 30. If this value is not equal to -1 or smaller than 1 or larger than 30, the BarCodeException.EnumErrorCode.PDF417Col is thrown.
- **Row:** Lets you specify the row numbers for the barcode. It takes the integer value; the default value is -1 and the range of this value is from 3 to 90. If this value is not equal to -1 or smaller than 3 or larger than 90, the BarCodeException.EnumErrorCode.PDF417Row is thrown.
- **ErrorLevel:** Lets you specify the error correction level for the barcode. It takes the integer value; the value is -1

or the range of this value is from 0 to 8. If this value is not equal to -1 or is smaller than 0 or larger than 8, the BarCodeException.EnumErrorCode.PDF417ErrorLevel is thrown.

- And Level 0 is low strength and the default value is -1.
- **Type:** Lets you select the type of PDF417 barcode. The available types are Normal and Simple.

QRCodeOptions:

- **Model:** Lets you select the model of QRCode. The available models are Model1 and Model2.
- **ErrorLevel:** Lets you select the error correction level for the barcode. The available options are Low, Medium, Quality, and High.
- **Version:** Lets you specify the version of the barcode.
- **Mask:** Lets you select the pattern used for masking barcode. In order to make sure QRCode being successfully read, mask process is required to balance brightness. The options available are Auto, Mask000, Mask001, Mask010, Mask011, Mask100, Mask101, Mask110, and Mask111. The following image shows masking in QRCode:



- **Connection:** Lets you select whether connection is used for the barcode. It takes the value True of False.
- **ConnectionNumber:** Lets you specify the connection number for the barcode. It takes the integer value ranging from 0 to 15. If this value is smaller than 0 or larger than 15, EnumErrorCode.QRCodeConnectionNo exception will be thrown.
- **Encoding:** Lets you select the encoding for the barcode. It takes the integer value. The value is -1 or the range is from 1 to 14 when the Model property is set to Model1. The value is -1 or the range is from 1 to 40 when the Model property is set to Model2.

RssExpandedStackedOptions:

- **RowCount:** Lets you specify the number of stacked rows. It takes the integer value; the range is from 1 to 11. If this value is smaller than 1 or larger than 11, the BarCodeException.EnumErrorCode.RSSExpandedStackedCount is thrown.

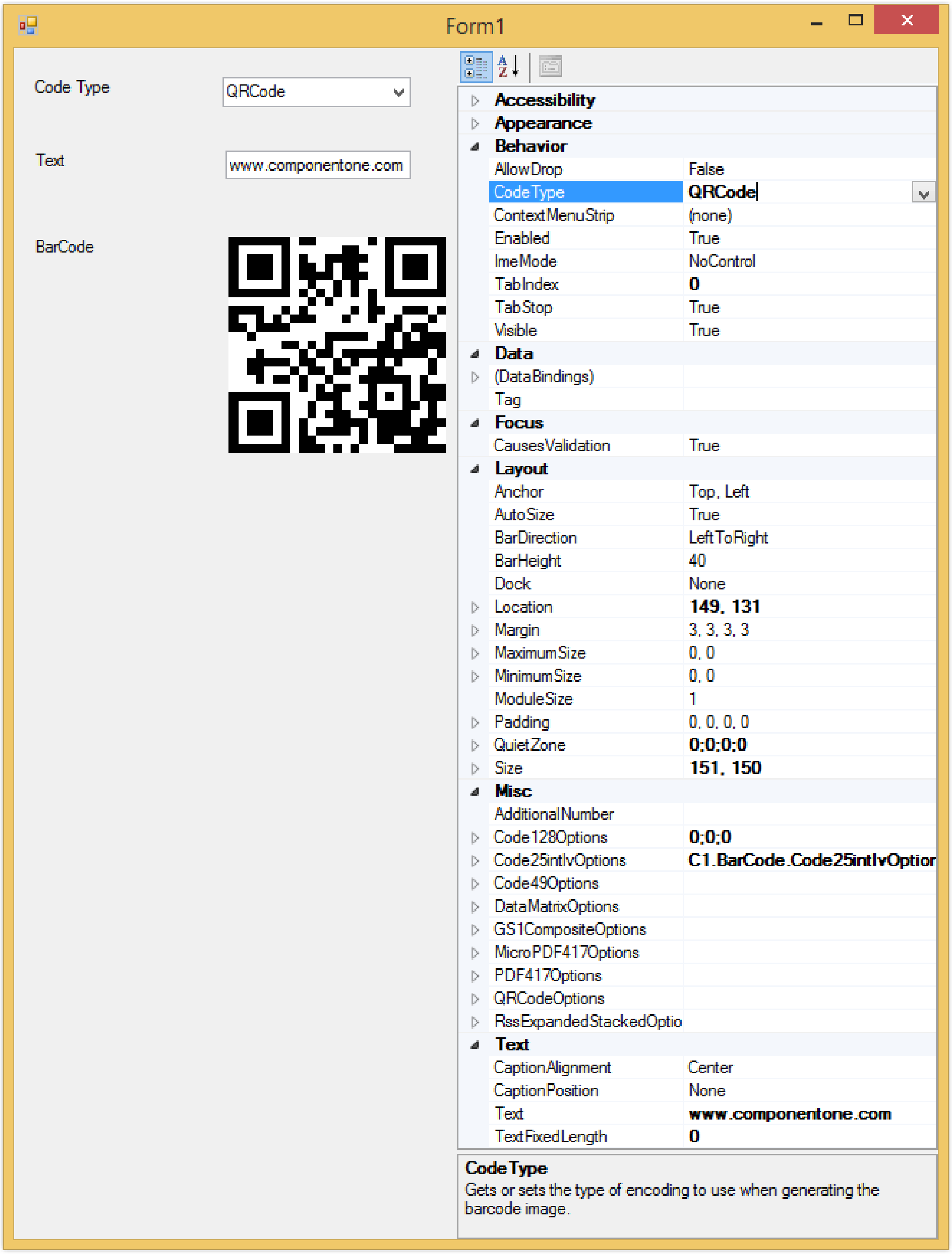
Customizing the C1BarCode Control

The Property window provides many design time options to customize the overall appearance of the C1BarCode control. These options depend on the type of barcode used in an application. You can also perform run time customization on C1BarCode control using the PropertyGrid control. See the following steps to customize the **C1BarCode** control during run time. This code uses the sample created in the BarCode for WinForms Quick Start section.

1. Add **PropertyGrid** control to the form.
2. Add following code to the **Form_Load** event.

   Visual Basic
   ```
   PropertyGrid1.SelectedObject = C1BarCode1
   ```
   C#
   ```
   propertyGrid1.SelectedObject = c1BarCode1;
   ```

   The above code sets the **PropertyGrid** to display the options available in the **C1BarCode**.
3. Run the project. The following images shows the **CodeType** set to the **QRCode** and the **Text** set to a URL.



Code Type	QRCode
Text	www.componentone.com
BarCode	

Accessibility	
Appearance	
Behavior	
AllowDrop	False
CodeType	**QRCode**
ContextMenuStrip	(none)
Enabled	True
ImeMode	NoControl
TabIndex	**0**
TabStop	True
Visible	True
Data	
(DataBindings)	
Tag	
Focus	
CausesValidation	True
Layout	
Anchor	Top, Left
AutoSize	True
BarDirection	LeftToRight
BarHeight	40
Dock	None
Location	**149, 131**
Margin	3, 3, 3, 3
MaximumSize	0, 0
MinimumSize	0, 0
ModuleSize	1
Padding	0, 0, 0, 0
QuietZone	**0;0;0;0**
Size	**151, 150**
Misc	
AdditionalNumber	
Code128Options	**0;0;0**
Code25intlvOptions	**C1.BarCode.Code25intlvOptior**
Code49Options	
DataMatrixOptions	
GS1CompositeOptions	
MicroPDF417Options	
PDF417Options	
QRCodeOptions	
RssExpandedStackedOptio	
Text	
CaptionAlignment	Center
CaptionPosition	None
Text	**www.componentone.com**
TextFixedLength	**0**

CodeType
Gets or sets the type of encoding to use when generating the barcode image.