

---

ComponentOne

# BarCode for WinForms

Copyright © 2012 ComponentOne LLC. All rights reserved.

*Corporate Headquarters*

**ComponentOne LLC**

201 South Highland Avenue  
3<sup>rd</sup> Floor  
Pittsburgh, PA 15206 • USA

**Internet:** [info@ComponentOne.com](mailto:info@ComponentOne.com)

**Web site:** <http://www.componentone.com>

**Sales**

E-mail: [sales@componentone.com](mailto:sales@componentone.com)

Telephone: 1.800.858.2739 or 1.412.681.4343 (Pittsburgh, PA USA Office)

**Trademarks**

The ComponentOne product name is a trademark and ComponentOne is a registered trademark of ComponentOne LLC. All other trademarks used herein are the properties of their respective owners.

**Warranty**

ComponentOne warrants that the original CD (or diskettes) are 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 CD (or disk) 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 a defective CD (or disk) by sending it and a check for \$25 (to cover postage and handling) to ComponentOne.

Except for the express warranty of the original CD (or disks) 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. We are 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**

While you are welcome to make backup copies of the software for your own use and protection, you are not permitted to make copies for the use of anyone else. We put a lot of time and effort into creating this product, and we appreciate your support in seeing that it is used by licensed users only.

This manual was produced using [ComponentOne Doc-To-Help™](#).

# Table of Contents

ComponentOne BarCode for WinForms Overview .....	1
Installing BarCode for WinForms .....	1
BarCode for WinForms Setup Files .....	1
System Requirements .....	2
Installing Demonstration Versions .....	2
Uninstalling BarCode for WinForms .....	2
Licensing FAQs .....	3
What is Licensing? .....	3
How does Licensing Work? .....	3
Common Scenarios .....	4
Troubleshooting .....	6
Technical Support .....	7
Redistributable Files .....	8
About this Documentation .....	8
Namespaces .....	9
Creating a .NET Project .....	10
Adding the C1BarCode Component to a Project .....	10
Migrating a ComponentOne BarCode for WinForms Project to Visual Studio 2005 .....	11
Key Features .....	12
BarCode for WinForms Quick Start .....	14
Step 1 of 3: Setting Up the Form .....	14
Step 2 of 3: Adding Code to the Project .....	15
Step 3 of 3: Running the Project .....	18
Using BarCode for WinForms .....	19
Supported Encodings .....	19
Customizing the C1BarCode Control .....	20
Using the C1BarCode Control in a Document .....	21
Using C1QRCode .....	22
How to Obtain an Image from C1QRCode .....	22
C1QRCode Encoding Limits .....	23

Improving the C1QRCode Image Resolution .....	24
BarCode for WinForms Samples .....	24

# ComponentOne BarCode for WinForms Overview

Add barcode images to grid cells, Web pages, or regular .NET PrintDocument objects with **ComponentOne BarCode for WinForms**.

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 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're done!

For a list of the latest features added to **ComponentOne Studio for WinForms**, visit [What's New in Studio for WinForms](#).



## Getting Started

To get started, review the following topics:

- [Key Features](#) (page 12)
- [Quick Start](#) (page 14)
- [Samples](#) (page 24)

## Installing BarCode for WinForms

The following sections provide helpful information on installing **ComponentOne BarCode for WinForms**.

### BarCode for WinForms Setup Files

The **ComponentOne Studio for WinForms** installation program will create the following directory: C:\Program Files\ComponentOne\Studio for WinForms. This directory contains the following subdirectories:

<b>bin</b>	Contains copies of all ComponentOne binaries (DLLs, EXEs).
<b>C1BarCode</b>	Contains files (at least a readme.txt) related to the <b>BarCode for WinForms</b> product.

The **ComponentOne Studio for WinForms Help Setup** program installs integrated Microsoft Help Viewer help to the C:\Program Files\ComponentOne\Studio for WinForms\HelpViewer folder.

### Samples

Samples for the product are installed in the **ComponentOne Samples** folder by default. The path of the **ComponentOne Samples** directory is slightly different on Windows XP and Windows 7/Vista machines:

**Windows XP path:** C:\Documents and Settings\\My Documents\ComponentOne Samples

**Windows 7/Vista path:** C:\Users\\Documents\ComponentOne Samples

The **ComponentOne Samples** folder contains the following subdirectories:

<b>Common</b>	Contains support and data files that are used by many of the demo programs.
<b>C1Barcode</b>	Contains samples and tutorials for <b>Barcode for WinForms</b> .

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on your desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for WinForms | Samples | Barcode Samples**.

## System Requirements

System requirements include the following:

<b>Operating Systems:</b>	Windows® 2000 Windows Server® 2003 Windows Server 2008 Windows XP SP2 Windows Vista™ Windows 7
<b>Environments:</b>	.NET Framework 2.0 or later C# .NET Visual Basic .NET
<b>Disc Drive:</b>	CD or DVD-ROM drive if installing from CD

## Installing Demonstration Versions

If you wish to try **ComponentOne Barcode for WinForms** and do not have a serial number, follow the steps through the installation wizard and use the default serial number.

The only difference between unregistered (demonstration) and registered (purchased) versions of our products is that registered versions will stamp every application you compile so a ComponentOne banner will not appear when your users run the applications.

## Uninstalling Barcode for WinForms

To uninstall **ComponentOne Barcode for WinForms**:

1. Open the **Control Panel** and select **Add or Remove Programs (Programs and Features)** in Windows 7/Vista).
2. Select **ComponentOne Studio for WinForms** and click the **Remove** button.
3. Click **Yes** to remove the program.

To uninstall **ComponentOne Barcode for WinForms** integrated help:

1. Open the Control Panel and select **Add or Remove Programs** (Programs and Features in Windows 7/Vista).
2. Select **ComponentOne Studio for WinForms Help** and click the **Remove** button.
3. Click **Yes** to remove the integrated help.

# Licensing FAQs

This section describes the main technical aspects of licensing. It may help the user to understand and resolve licensing problems he may experience when using ComponentOne .NET and ASP.NET products.

## What is Licensing?

Licensing is a mechanism used to protect intellectual property by ensuring that users are authorized to use software products.

Licensing is not only used to prevent illegal distribution of software products. Many software vendors, including ComponentOne, use licensing to allow potential users to test products before they decide to purchase them.

Without licensing, this type of distribution would not be practical for the vendor or convenient for the user. Vendors would either have to distribute evaluation software with limited functionality, or shift the burden of managing software licenses to customers, who could easily forget that the software being used is an evaluation version and has not been purchased.

## How does Licensing Work?

ComponentOne uses a licensing model based on the standard set by Microsoft, which works with all types of components.

**Note:** The **Compact Framework** components use a slightly different mechanism for run time licensing than the other ComponentOne components due to platform differences.

When a user decides to purchase a product, he receives an installation program and a Serial Number. During the installation process, the user is prompted for the serial number that is saved on the system. (Users can also enter the serial number by clicking the **License** button on the **About Box** of any ComponentOne product, if available, or by rerunning the installation and entering the serial number in the licensing dialog.)

When a licensed component is added to a form or Web page, Visual Studio obtains version and licensing information from the newly created component. When queried by Visual Studio, the component looks for licensing information stored in the system and generates a run-time license and version information, which Visual Studio saves in the following two files:

- An assembly resource file which contains the actual run-time license
- A "licenses.licx" file that contains the licensed component strong name and version information

These files are automatically added to the project.

In WinForms and ASP.NET 1.x applications, the run-time license is stored as an embedded resource in the assembly hosting the component or control by Visual Studio. In ASP.NET 2.x applications, the run-time license may also be stored as an embedded resource in the App\_Licenses.dll assembly, which is used to store all run-time licenses for all components directly hosted by WebForms in the application. Thus, the App\_licenses.dll must always be deployed with the application.

The licenses.licx file is a simple text file that contains strong names and version information for each of the licensed components used in the application. Whenever Visual Studio is called upon to rebuild the application resources, this file is read and used as a list of components to query for run-time licenses to be embedded in the appropriate assembly resource. Note that editing or adding an appropriate line to this file can force Visual Studio to add run-time licenses of other controls as well.

Note that the licenses.licx file is usually not shown in the Solution Explorer; it appears if you press the **Show All Files** button in the Solution Explorer's Toolbox, or from Visual Studio's main menu, select **Show All Files** on the **Project** menu.

Later, when the component is created at run time, it obtains the run-time license from the appropriate assembly resource that was created at design time and can decide whether to simply accept the run-time license, to throw an

exception and fail altogether, or to display some information reminding the user that the software has not been licensed.

All ComponentOne products are designed to display licensing information if the product is not licensed. None will throw licensing exceptions and prevent applications from running.

## Common Scenarios

The following topics describe some of the licensing scenarios you may encounter.

### ***Creating components at design time***

This is the most common scenario and also the simplest: the user adds one or more controls to the form, the licensing information is stored in the licenses.licx file, and the component works.

Note that the mechanism is exactly the same for Windows Forms and Web Forms (ASP.NET) projects.

### ***Creating components at run time***

This is also a fairly common scenario. You do not need an instance of the component on the form, but would like to create one or more instances at run time.

In this case, the project will not contain a licenses.licx file (or the file will not contain an appropriate run-time license for the component) and therefore licensing will fail.

To fix this problem, add an instance of the component to a form in the project. This will create the licenses.licx file and things will then work as expected. (The component can be removed from the form after the licenses.licx file has been created).

Adding an instance of the component to a form, then removing that component, is just a simple way of adding a line with the component strong name to the licenses.licx file. If desired, you can do this manually using notepad or Visual Studio itself by opening the file and adding the text. When Visual Studio recreates the application resources, the component will be queried and its run-time license added to the appropriate assembly resource.

### ***Inheriting from licensed components***

If a component that inherits from a licensed component is created, the licensing information to be stored in the form is still needed. This can be done in two ways:

- Add a LicenseProvider attribute to the component.

This will mark the derived component class as licensed. When the component is added to a form, Visual Studio will create and manage the licenses.licx file, and the base class will handle the licensing process as usual. No additional work is needed. For example:

```
[LicenseProvider(typeof(LicenseProvider))]  
class MyGrid: C1.Win.C1FlexGrid.C1FlexGrid  
{  
    // ...  
}
```

- Add an instance of the base component to the form.

This will embed the licensing information into the licenses.licx file as in the previous scenario, and the base component will find it and use it. As before, the extra instance can be deleted after the licenses.licx file has been created.

Please note, that C1 licensing will not accept a run time license for a derived control if the run time license is embedded in the same assembly as the derived class definition, and the assembly is a DLL. This restriction is necessary to prevent a derived control class assembly from being used in other applications without a design time license. If you create such an assembly, you will need to take one of the actions previously described create a component at run time.

## Using licensed components in console applications

When building console applications, there are no forms to add components to, and therefore Visual Studio won't create a licenses.licx file.

In these cases, create a temporary Windows Forms application and add all the desired licensed components to a form. Then close the Windows Forms application and copy the licenses.licx file into the console application project.

Make sure the licenses.licx file is configured as an embedded resource. To do this, right-click the licenses.licx file in the Solution Explorer window and select **Properties**. In the property window, set the **Build Action** property to **Embedded Resource**.

## Using licensed components in Visual C++ applications

There is an issue in VC++ 2003 where the licenses.licx is ignored during the build process; therefore, the licensing information is not included in VC++ applications.

To fix this problem, extra steps must be taken to compile the licensing resources and link them to the project. Note the following:

1. Build the C++ project as usual. This should create an .exe file and also a licenses.licx file with licensing information in it.
2. Copy the licenses.licx file from the app directory to the target folder (Debug or Release).
3. Copy the C1Lc.exe utility and the licensed dlls to the target folder. (Don't use the standard lc.exe, it has bugs.)
4. Use C1Lc.exe to compile the licenses.licx file. The command line should look like this:  
`c1lc /target:MyApp.exe /complist:licenses.licx /i:C1.Win.C1FlexGrid.dll`
5. Link the licenses into the project. To do this, go back to Visual Studio, right-click the project, select Properties, and go to the **Linker/Command Line** option. Enter the following:  
`/ASSEMBLYRESOURCE:Debug\MyApp.exe.licenses`
6. Rebuild the executable to include the licensing information in the application.

## Using licensed components with automated testing products

Automated testing products that load assemblies dynamically may cause them to display license dialog boxes. This is the expected behavior since the test application typically does not contain the necessary licensing information, and there is no easy way to add it.

This can be avoided by adding the string "C1CheckForDesignLicenseAtRuntime" to the AssemblyConfiguration attribute of the assembly that contains or derives from ComponentOne controls. This attribute value directs the ComponentOne controls to use design time licenses at run time.

For example:

```
#if AUTOMATED TESTING
[AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime")]
#endif
public class MyDerivedControl : C1LicensedControl
{
    // ...
}
```

Note that the AssemblyConfiguration string may contain additional text before or after the given string, so the AssemblyConfiguration attribute can be used for other purposes as well. For example:

```
[AssemblyConfiguration("C1CheckForDesignLicenseAtRuntime,BetaVersion")]
```

THIS METHOD SHOULD ONLY BE USED UNDER THE SCENARIO DESCRIBED. It requires a design time license to be installed on the testing machine. Distributing or installing the license on other computers is a violation of the EULA.

## Troubleshooting

We try very hard to make the licensing mechanism as unobtrusive as possible, but problems may occur for a number of reasons.

Below is a description of the most common problems and their solutions.

### ***I have a licensed version of a ComponentOne product but I still get the splash screen when I run my project.***

If this happens, there may be a problem with the licenses.licx file in the project. It may not exist, it may contain incorrect information, or it may not be configured correctly.

First, try a full rebuild (**Rebuild All** from the Visual Studio **Build** menu). This will usually rebuild the correct licensing resources.

#### **If that fails follow these steps:**

1. Open the affected project.
2. Select an instance of the updated component.
3. In the Visual Studio Properties window, change any property. It does not matter which property you change; you can change it back to the previous value.
4. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

#### **Alternative 1: Follow these steps:**

1. Open a new Visual Studio.NET project.
2. Add the updated component to the form.
3. Compile and run the new project.
4. Open the licenses.licx file in the new project.
5. Copy the line that starts with the namespace of the updated component (for example, C1.Win.C1Report) and ends with a public key token.
6. Open the existing, incorrectly licensed project.
7. Open the licenses.licx file in the new project.
8. Paste the line from step 5 into this file (replace the old licensing information with the new).
9. Rebuild the project using the **Rebuild All** option (not just **Rebuild**) and run the solution.

#### **Alternative 2: Follow these steps:**

1. Open the affected project.
2. Delete the licenses.licx file from the project.
3. Add a new instance of the updated component to the form.
4. Rebuild and run the solution. The nag screen should not appear.
5. Remove the newly added component from the form.

Try each of these options multiple times, if necessary. If that still does not help, are you creating any of the controls in code rather than design-time? If so, you must add an entry for the control in the licenses.licx file (see <http://helpcentral.componentone.com/PrintableView.aspx?ID=1886> for more information). Also if this is a

website, as opposed to an ASP.NET web application, please try right-clicking the licenses.licx file and selecting "Build Runtime Licenses" from the context menu.

### ***I have a licensed version of a ComponentOne product on my Web server but the components still behave as unlicensed.***

There is no need to install any licenses on machines used as servers and not used for development.

The components must be licensed on the development machine, therefore the licensing information will be saved into the executable (.exe or .dll) when the project is built. After that, the application can be deployed on any machine, including Web servers.

For ASP.NET 2.x applications, be sure that the App\_Licenses.dll assembly created during development of the application is deployed to the bin application bin directory on the Web server.

If your ASP.NET application uses winforms user controls with constituent licensed controls, the runtime license is embedded in the winforms user control assembly. In this case, you must be sure to rebuild and update the user control whenever the licensed embedded controls are updated.

### ***I downloaded a new build of a component that I have purchased, and now I'm getting the splash screen when I build my projects.***

Make sure that the serial number is still valid. If you licensed the component over a year ago, your subscription may have expired. In this case, you have two options:

#### **Option 1 – Renew your subscription to get a new serial number.**

If you choose this option, you will receive a new serial number that you can use to license the new components (from the installation utility or directly from the **About Box**).

The new subscription will entitle you to a full year of upgrades and to download the latest maintenance builds directly from <http://prerelease.componentone.com/>.

#### **Option 2 – Continue to use the components you have.**

Subscriptions expire, products do not. You can continue to use the components you received or downloaded while your subscription was valid.

## Technical Support

ComponentOne offers various support options. For a complete list and a description of each, visit the ComponentOne Web site at <http://www.componentone.com/SuperProducts/SupportServices/>.

Some methods for obtaining technical support include:

- **[Online Resources](#)**  
ComponentOne provides customers with a comprehensive set of technical resources in the form of FAQs, samples and videos, Version Release History, searchable Knowledge base, searchable Online Help and more. We recommend this as the first place to look for answers to your technical questions.
- **Online Support via our Incident Submission Form**  
This online support service provides you with direct access to our Technical Support staff via an [online incident submission form](#). When you submit an incident, you'll immediately receive a response via e-mail confirming that you've successfully created an incident. This email will provide you with an Issue Reference ID and will provide you with a set of possible answers to your question from our Knowledgebase. You will receive a response from one of the ComponentOne staff members via e-mail in 2 business days or less.
- **Product Forums**  
ComponentOne's [product forums](#) are available for users to share information, tips, and techniques

regarding ComponentOne products. ComponentOne developers will be available on the forums to share insider tips and technique and answer users' questions. Please note that a ComponentOne User Account is required to participate in the ComponentOne Product Forums.

- **Installation Issues**

Registered users can obtain help with problems installing ComponentOne products. Contact technical support by using the [online incident submission form](#) or by phone (412.681.4738). Please note that this does not include issues related to distributing a product to end-users in an application.

- **Documentation**

Microsoft integrated ComponentOne documentation can be installed with each of our products, and documentation is also available online. If you have suggestions on how we can improve our documentation, please email the [Documentation team](#). Please note that e-mail sent to the [Documentation team](#) is for documentation feedback only. [Technical Support](#) and [Sales](#) issues should be sent directly to their respective departments.

**Note:** You must create a ComponentOne Account and register your product with a valid serial number to obtain support using some of the above methods.

## Redistributable Files

**ComponentOne BarCode for WinForms** is developed and published by ComponentOne LLC. You may use it to develop applications in conjunction with Microsoft Visual Studio or any other programming environment that enables the user to use and integrate the control(s). You may also distribute, free of royalties, the following Redistributable Files with any such application you develop to the extent that they are used separately on a single CPU on the client/workstation side of the network:

- C1.Win.C1BarCode.2.dll
- C1.Win.C1BarCode.4.dll
- C1.Win.C1BarCode.4.Design.dll

Site licenses are available for groups of multiple developers. Please contact [Sales@ComponentOne.com](mailto:Sales@ComponentOne.com) for details.

## About this Documentation

### Acknowledgements

*Microsoft, Windows, Windows Vista, Windows Server, and Visual Studio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.*

### ComponentOne

If you have any suggestions or ideas for new features or controls, please call us or write:

*Corporate Headquarters*

#### **ComponentOne LLC**

201 South Highland Avenue  
3<sup>rd</sup> Floor  
Pittsburgh, PA 15206 • USA  
412.681.4343  
412.681.4384 (Fax)

<http://www.componentone.com/>

### ComponentOne Doc-To-Help

This documentation was produced using [ComponentOne Doc-To-Help® Enterprise](#).

# Namespaces

Namespaces organize the objects defined in an assembly. Assemblies can contain multiple namespaces, which can in turn contain other namespaces. Namespaces prevent ambiguity and simplify references when using large groups of objects such as class libraries.

The namespace for the C1Barcode component is **C1.Win.C1Barcode**. The following code fragment shows how to declare a C1Barcode component using the fully qualified name for this class:

- Visual Basic  

```
Dim barcode1 As C1.Win.C1Barcode.C1Barcode
```
- C#  

```
C1.Win.C1Barcode.C1Barcode barcode1;
```

Namespaces address a problem sometimes known as *namespace pollution*, in which the developer of a class library is hampered by the use of similar names in another library. These conflicts with existing components are sometimes called *name collisions*.

For example, if you create a new class named C1Barcode, you can use it inside your project without qualification. However, the C1Barcode assembly also implements a class called **C1Barcode**. So, if you want to use the C1Barcode class in the same project, you must use a fully qualified reference to make the reference unique. If the reference is not unique, Visual Studio .NET produces an error stating that the name is ambiguous. The following code snippet demonstrates how to declare these objects:

- Visual Basic  

```
' Define a new C1Barcode object  
Dim MyBarcode as C1Barcode  
' Define a new C1Barcode.C1Barcode object.  
Dim C1Barcode as C1.Win.C1Barcode.C1Barcode
```
- C#  

```
// Define a new C1Barcode object  
C1Barcode MyBarcode;  
// Define a new C1Barcode.C1Barcode object.  
C1.Win.C1Barcode.C1Barcode C1Barcode;
```

Fully qualified names are object references that are prefixed with the name of the namespace where the object is defined. You can use objects defined in other projects if you create a reference to the class (by choosing **Add Reference** from the **Project** menu) and then use the fully qualified name for the object in your code.

Fully qualified names prevent naming conflicts because the compiler can always determine which object is being used. However, the names themselves can get long and cumbersome. To get around this, you can use the **Imports** statement (**using** in C#) to define an alias — an abbreviated name you can use in place of a fully qualified name. For example, the following code snippet creates aliases for two fully qualified names, and uses these aliases to define two objects:

- Visual Basic  

```
Imports C1Barcode = C1.Win.C1Barcode  
Imports MyBarcode = MyProject.C1Barcode  
  
Dim s1 As C1Barcode  
Dim s2 As MyBarcode
```
- C#  

```
using C1Barcode = C1.Win.C1Barcode;  
using MyBarcode = MyProject.C1Barcode;  
  
C1Barcode s1;  
MyBarcode s2;
```

If you use the **Imports** statement without an alias, you can use all the names in that namespace without qualification, provided they are unique to the project.

## Creating a .NET Project

To create a new .NET project, complete the following steps:

1. From the **File** menu in Microsoft Visual Studio .NET, select **New Project**. The **New Project** dialog box opens.
2. Under **Project Types**, choose either **Visual Basic** or **Visual C#**. Note that one of these options may be located under **Other Languages**.
3. Select **Windows Application** from the list of **Templates** in the right pane.
4. Enter or browse for a location for your application in the **Location** field and click **OK**. A new Microsoft Visual Studio .NET project is created in the specified location. In addition, a new Form1 is displayed in the Designer view.
5. Double-click the **C1Barcode** component from the Toolbox to add it to Form1. For information on adding a component to the Toolbox, see [Adding the C1Barcode Component to a Project](#) (page 10).

## Adding the C1Barcode Component to a Project

When you install **ComponentOne Studio for WinForms**, the **Create a ComponentOne Visual Studio Toolbox Tab** checkbox is checked, by default, in the installation wizard. When you open Visual Studio, you will notice a **ComponentOne Studio for WinForms** tab containing the ComponentOne controls has automatically been added to the Toolbox.

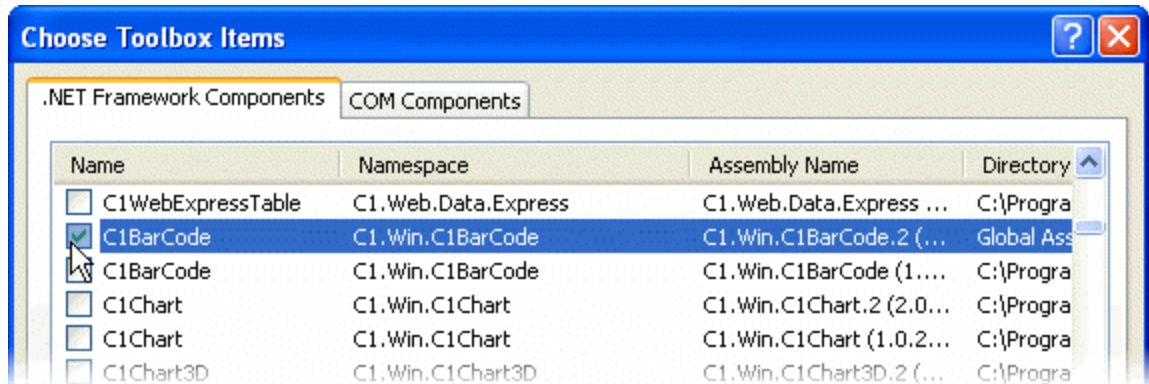
If you decide to uncheck the **Create a ComponentOne Visual Studio Toolbox Tab** checkbox during installation, you can manually add ComponentOne controls to the Toolbox at a later time.

**Barcode for WinForms** provides the C1Barcode control. To use C1Barcode, add it to the form or add a reference to the C1.Win.C1Barcode assembly to your project.

### Adding C1Barcode to the Toolbox

To add C1Barcode to the Visual Studio Toolbox:

1. Open the Visual Studio IDE (Microsoft Development Environment). Make sure the Toolbox is visible (select **Toolbox** in the **View** menu if necessary) and right-click it to open the context menu.
2. To make C1Barcode component appear on its own tab in the Toolbox, select **Add Tab** from the context menu and type in the tab name, **C1Barcode**, for example.
3. Right-click the tab where the component is to appear and select **Choose Items** from the context menu. The **Choose Toolbox Items** dialog box opens.
4. In the dialog box, select the **.NET Framework Components** tab. Sort the list by Namespace (click the Namespace column header) and check the check boxes for all components belonging to namespace C1.Win.C1Barcode. Note that there may be more than one component for each namespace.



### Adding C1Barcode to the Form

To add C1Barcode to a form:

1. Add C1Barcode to the Visual Studio Toolbox.
2. Double-click the control or drag it onto your form.

### Adding a Reference to the C1Barcode Assembly

To add a reference to the C1Barcode assembly:

1. Select the **Add Reference** option from the **Project** menu of your project.
2. Select the **ComponentOne C1Barcode** assembly from the list on the **.NET** tab or browse to find the C1.Win.C1Barcode.2.dll file and click **OK**.
3. Double-click the form caption area to open the code window. At the top of the file, add the following **Imports** statements (using in C#):

```
Imports C1.Win.C1Barcode
```

**Note:** This makes the objects defined in the C1Barcode assembly visible to the project. See [Namespaces](#) (page 9) for more information.

## Migrating a ComponentOne Barcode for WinForms Project to Visual Studio 2005

To migrate a project using ComponentOne components to Visual Studio 2005, there are two main steps that must be performed. First, you must convert your project to Visual Studio 2005, which includes removing any references to a previous assembly and adding a reference to the new assembly. Secondly, the .licx file, or licensing file, must be updated in order for the project to run correctly.

### To convert the project:

1. Open Visual Studio 2005 and select **File, Open Project**.
2. Locate the **.sln** file for the project that you wish to convert to Visual Studio 2005. Select it and click **Open**. The **Visual Studio Conversion Wizard** appears.
3. Click **Next**.
4. Select **Yes, create a backup before converting** to create a backup of your current project and click **Next**.
5. Click **Finish** to convert your project to Visual Studio 2005. The **Conversion Complete** window appears.
6. Click **Show the conversion log when the wizard is closed** if you want to view the conversion log.

- Click **Close**. The project opens. Now you must remove references to any of the previous ComponentOne .dlls and add references to the new ones.
- Go to the Solution Explorer (**View | Solution Explorer**), select the project, and click the **Show All Files** button.

**Note:** The **Show All Files** button does not appear in the Solution Explorer toolbar if the Solution project node is selected.

- Expand the **References** node, right-click C1.Common and select **Remove**. Also remove C1.Win.C1Barcode.2.dll the same way.
- Right-click the **References** node and select **Add Reference**.
- Browse and select **C1.Win.C1Barcode.2.dll**. Click **OK** to add it to the project.

#### To update the .licx file:

- In the Solution Explorer, right-click the **licenses.licx** file and select **Delete**.
- Click **OK** to permanently delete **licenses.licx**. The project must be rebuilt to create a new, updated version of the .licx file.
- Click the **Start Debugging** button to compile and run the project. The new .licx file may not be visible in the Solution Explorer.
- Select **File, Close** to close the form and then double-click the **Form.vb** or **Form.cs** file in the Solution Explorer to reopen it. The new **licenses.licx** file appears in the list of files.


The migration process is complete.

## Key Features

The C1Barcode code was originally developed for use in the **C1Report** component and has been a part of **C1Report** since build 151, released in May 2004.

We feel that **ComponentOne Barcode for WinForms** is a valuable addition to the ComponentOne Studios. We decided to package it as a stand-alone component because it can be useful in many applications besides reports. For example, you can use C1Barcode to add barcode images to grid cells, to Web pages, or to regular .NET PrintDocument objects. The image below shows a **C1Report** listing products along with the C1Barcode for each:

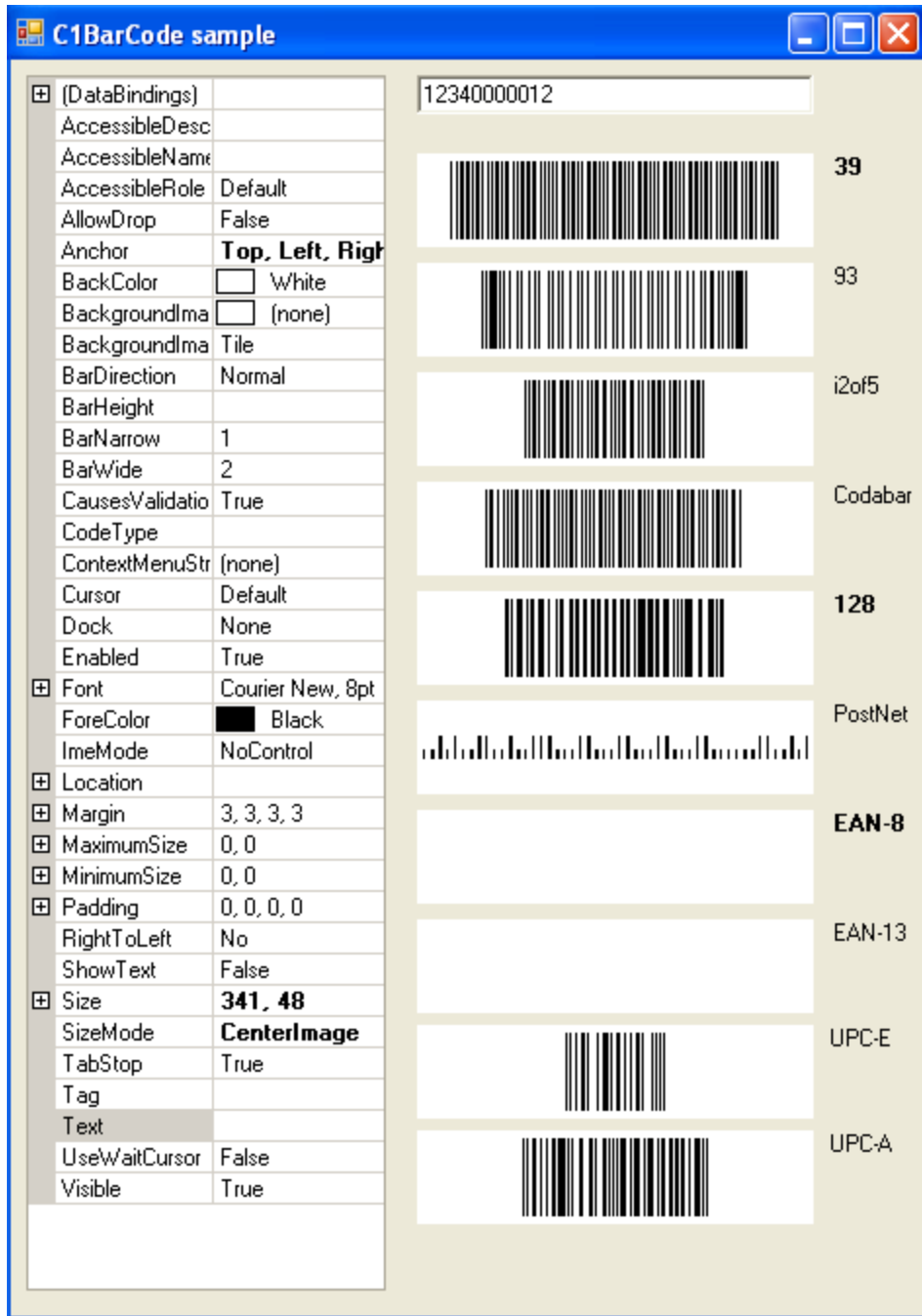
### *Products Report*

<i>ProductID</i>	<i>ProductName</i>	<i>UnitPrice</i>	<i>UnitsInSt</i>	<i>C1Barcode</i>
1	Chai	18	39	
2	Chang	19	17	
3	Aniseed Syrup	10	13	
4	Chef Anton's Cajun Seasoning	22	53	
5	Chef Anton's Gumbo Mix	21.35	0	
6	Grandin's Boysenberry Spread	25	120	
7	Uncle Bob's Organic Dried Pears	30	15	

The main features of **BarCode for WinForms** include:

- Supports 10 different encodings

The C1Barcode control supports 10 encodings, including: Codabar, Cod128, Code39, Code93, Code120%, Ean13, Ean8, PostNet, UpcA, and UpcE.



- Provides the **C1QRCode** 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. See [Using C1QRCode](#) (page 22) for more information.

- Automatically adds checksums

The C1Barcode control automatically adds necessary control symbols and checksums to the value being encoded, depending on the encoding being used, to guarantee a good read on your barcodes.

- Royalty-free DLL for easy deployment

C1Barcode is a royalty-free DLL that can be deployed with your applications like any regular assembly.

# Barcode for WinForms Quick Start

This section details some of the features of **ComponentOne Barcode for WinForms**. This quick start will walk through the steps of adding C1Barcode to your project and setting C1Barcode appearance and behavior settings. The project that is created in this quick start will demonstrate the various encodings available in **ComponentOne Barcode for WinForms**. For more information about available encodings, see [Supported Encodings](#) (page 19).

## Step 1 of 3: Setting Up the Form

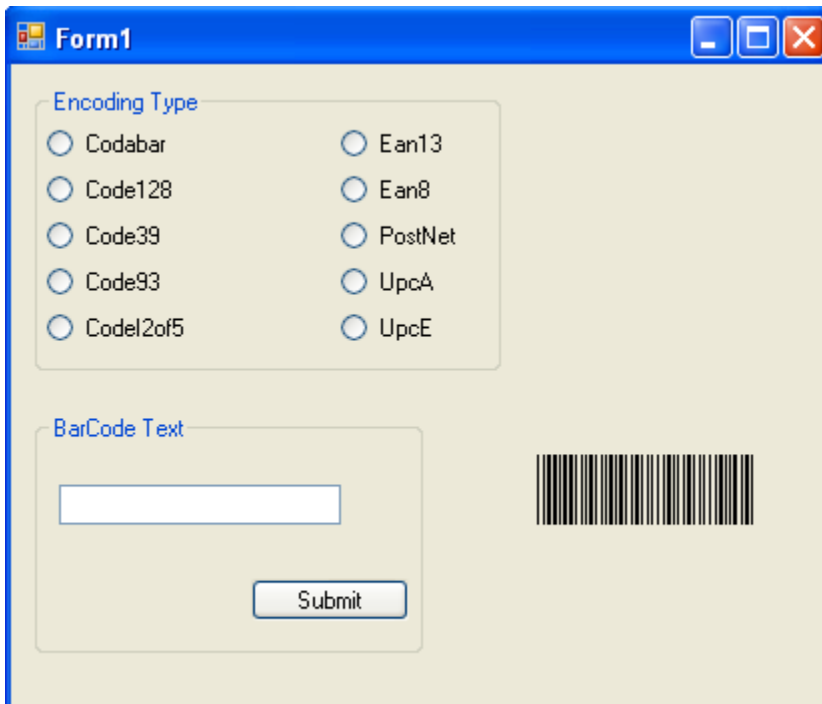
To begin, we will add C1Barcode to the form and set up the project.

Complete the following steps:

1. Create a new .NET project.
2. From the Toolbox, add the following controls to the form:
  - Two GroupBox controls
  - Ten RadioButton controls
  - One TextBox control
  - One command Button
  - One C1Barcode control
3. Set the following Text properties for the controls, and arrange the controls as in the image below:

Name	Text Property
GroupBox1	Encoding Type
GroupBox2	Bar Code Text
Button1	Submit
RadioButton1	Codabar
RadioButton2	Code128
RadioButton3	Code39
RadioButton4	Code93
RadioButton5	CodeI2of5

RadioButton6	Ean13
RadioButton7	Ean8
RadioButton8	PostNet
RadioButton9	UpcA
RadioButton10	UpcE



4. In the Properties window, set C1Barcode1's BackColor property to **Transparent**.
5. Resize the C1Barcode1 control.

You've just completed the first step in the **Barcode for WinForms** quick start. Continue to the next step to add code to the project.

## Step 2 of 3: Adding Code to the Project

In the last step you set up the form and added a C1Barcode control to the project, you will continue by adding code to the project.

Complete the following steps:

1. Double-click **Button1** to switch to code view and add the **Button1\_Click** event. Add the following code inside the **Button1\_Click** event to set C1Barcode1's Text property to the text entered in TextBox1:
  - Visual Basic

```
C1Barcode1.Text = TextBox1.Text
```
  - C#

```
c1Barcode1.Text = TextBox1.Text
```

2. Add the following code to handle Radio Button selection:

**Note:** You must add the **Imports C1.Win.C1Barcode** (Visual Basic projects) or **using C1.Win.C1Barcode;** (C# projects) to the top of your form in order for the following code to work correctly.

- Visual Basic

```
Private Sub RadioButton1_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton1.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.Codabar
End Sub

Private Sub RadioButton2_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton2.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.Code128
End Sub

Private Sub RadioButton3_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton3.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.Code39
End Sub

Private Sub RadioButton4_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton4.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.Code93
End Sub

Private Sub RadioButton5_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton5.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.CodeI2of5
End Sub

Private Sub RadioButton6_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton6.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.Ean13
End Sub

Private Sub RadioButton7_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton7.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.Ean8
End Sub

Private Sub RadioButton8_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton8.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.PostNet
End Sub

Private Sub RadioButton9_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton9.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.UpcA
End Sub

Private Sub RadioButton10_CheckedChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles RadioButton10.CheckedChanged
    C1Barcode1.CodeType = CodeTypeEnum.UpcE
End Sub
```

- C#

```
private void radioButton1_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.Codabar;
}

private void radioButton2_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.Code128;
}

private void radioButton3_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.Code39;
}

private void radioButton4_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.Code93;
}

private void radioButton5_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.CodeI2of5;
}

private void radioButton6_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.Ean13;
}

private void radioButton7_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.Ean8;
}

private void radioButton8_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.PostNet;
}

private void radioButton9_CheckedChanged(object sender,
System.EventArgs e)
{
    c1Barcode1.CodeType = CodeTypeEnum.UpcA;
}

private void radioButton10_CheckedChanged(object sender,
System.EventArgs e)
{

```

```
c1Barcode1.CodeType = CodeTypeEnum.UpcE;  
}
```

You've just completed the second step in the **ComponentOne BarCode for WinForms** quick start. Continue to the next step to run and view the project.

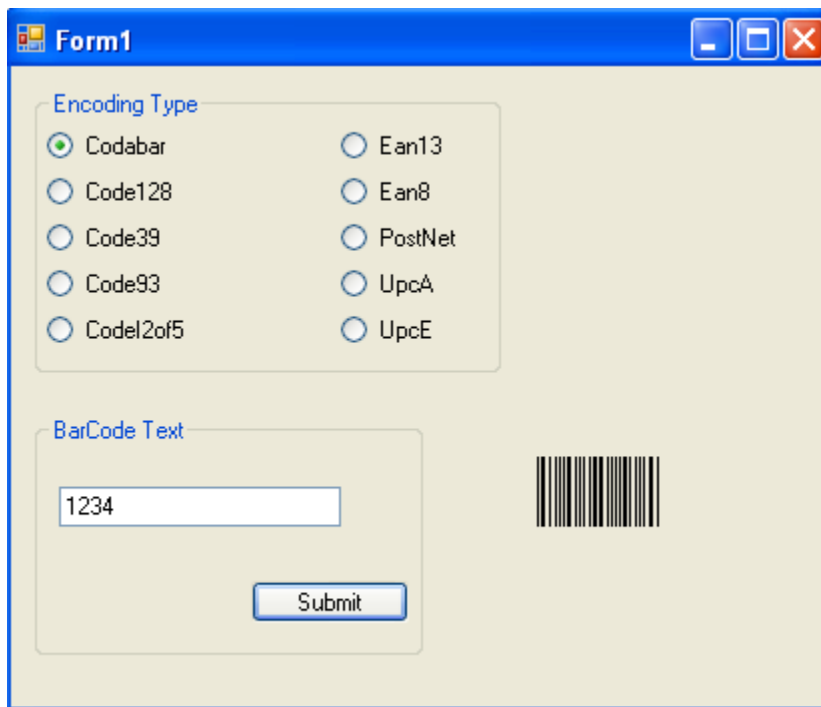
## Step 3 of 3: Running the Project

Now that we've set up the project and added code, we'll run the project to view the encodings supported by **ComponentOne BarCode for WinForms**.

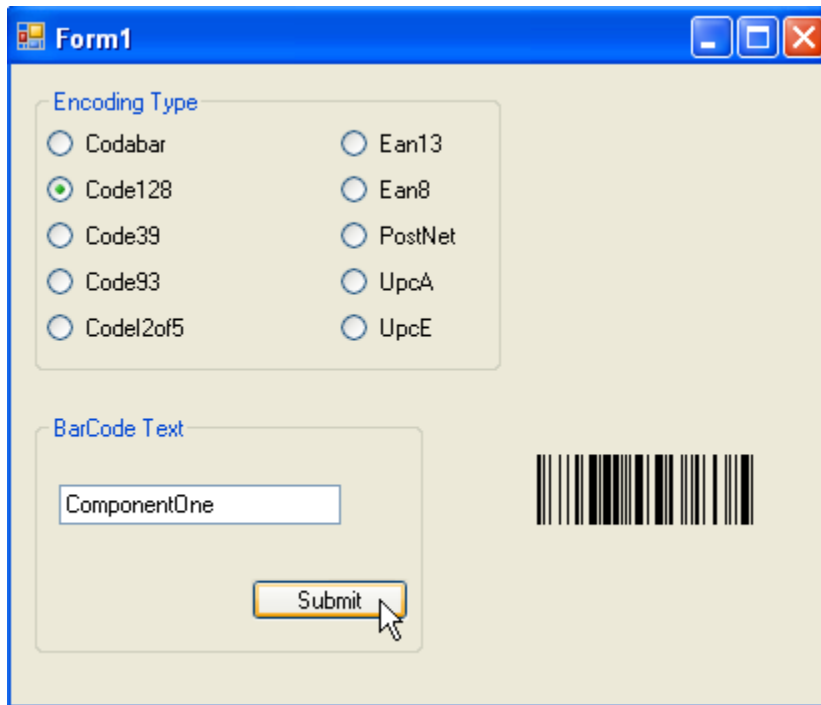
Complete the following steps:

1. Run the project and select the **Codabar** radio button.
2. Enter a numeric string, for example 1234, in the text box and click the **Submit** button.

Notice that the barcode now appears similar to the following:



3. Select different radio buttons to change the encoding type, and notice that the C1Barcode changes in appearance.
4. Try entering your name or different strings of alpha numeric characters to see what characters each encoding type will accept:



Note that some encodings have a minimum character requirement, while others will only work with numeric values. For more information about encodings, see [Supported Encodings](#) (page 19).

This concludes the **ComponentOne BarCode for WinForms** quick start.

## Using BarCode for WinForms

The following section details information about using **ComponentOne BarCode for WinForms**, including information about supported encoding formats.

### Supported Encodings

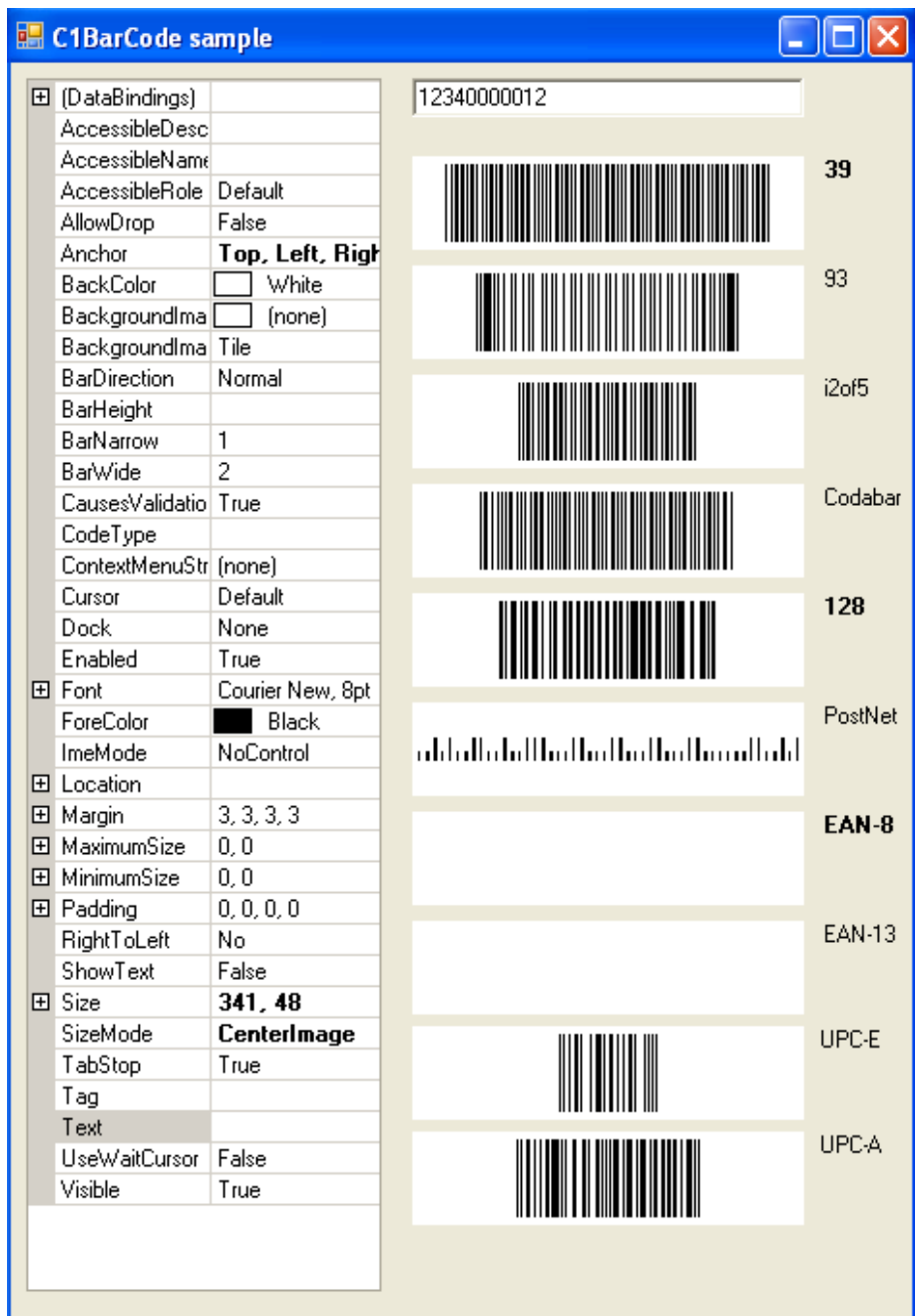
You can change the C1BarCode encoding type by setting the CodeType property. The C1BarCode control supports the following encodings:

Encoding	Description
<b>Codabar</b>	Codabar may encode 16 different characters (0 through 9 plus -\$/.+), plus an additional 4 start/stop characters (A through D). Codabar is used by some US blood banks, photo labs, and on FedEx airbills.
<b>Code128</b>	Code 128 is a very high density alpha-numeric barcode. It will use the least amount of space of any current 1-D symbology for symbols with 6 characters or more.
<b>Code39</b>	Code 39 is an alpha-numeric encoding also known as 3 of 9 and LOGMARS. This was the first alphanumeric symbology developed, and is one of the most widely used encodings.
<b>Code93</b>	Code 93 is an alpha-numeric encoding that is slightly denser than code 39.
<b>CodeI2of5</b>	Code I2of5 is a numeric encoding. The symbol can be as long as necessary to store the encoded data.

<b>Ean13</b>	EAN-13 was implemented by the International Article Numbering Association (EAN) in Europe. EAN-13 encodes a 12-digit code that consists of a 2 digit system code followed by a 5 digit manufacturer code and a 5-digit product code. The 12-digit code is followed by a checksum digit (automatically added by the control).
<b>Ean8</b>	EAN-8 provides a short barcode for small packages. It encodes a 7-digit code that consists of a 2 or 3 digit system code followed by a 4 or 5 digit product code. The 7-digit code is followed by a checksum digit (automatically added by the control).
<b>PostNet</b>	PostNet is a numeric encoding used by the US postal service. It differs from most others in that it is based on the height of the bars rather than on their width.
<b>UpcA</b>	UPC-A is the the common encoding you will find on virtually every consumer good on the shelves of your local supermarket, as well as books, magazines, and newspapers. It is similar to EAN-13, and encodes 11 digits of numeric data along with a trailing check digit.
<b>UpcE</b>	<p>UPC-E is a variation of UPC-A which allows for a more compact barcode by eliminating "extra" zeros. Since the resulting UPC-E barcode is about half the size of a UPC-A barcode, it is generally used on products with very small packaging.</p> <p>When using the UPC-E encoding, set the Text property to an 11-digit string as if you were using the UPC-A encoding.</p> <p>Note that not all UpcA codes can be encoded in UpcE. If the manufacturer code ends with "000", "100", or "200", the product number must be <math>\leq 900</math>. If the manufacturer code ends with "00" but not with "100", "200", or "300", then the product number must be <math>\leq 90</math>. If the manufacturer code ends with "0" but not with "00", then the product number must be <math>\leq 9</math>. If the manufacturer code does not end with "0", then the product number must be between 5 and 9.</p>

## 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.



The control will show the barcode image. Note that some encodings have a minimum character requirement, while others will only work with numeric values.

## Using the C1Barcode Control in a Document

If you want to include the barcode in a document, use the Image property to retrieve a resolution-independent image of the barcode.

For example:

- Visual Basic
 

```
C1Barcode1.CodeType = C1.Win.C1Barcode.CodeTypeEnum.Code39
```

```
C1Barcode1.Text = "123456"  
PictureBox1.Image = C1Barcode1.Image
```

- C#

```
c1Barcode1.CodeType = CodeTypeEnum.Code39;  
c1Barcode1.Text = "123456";  
pictureBox1.Image = c1Barcode1.Image;
```

## Using C1QRCode

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. Developed by the DENSO-WAVE company, the **QR code** format is efficient and compact, it doesn't require a special scanner to read it, and it is an open and freely available standard (ISO/IEC18004 and others).

The code is simply made up of black and white pixels arranged in patterns. With the C1QRCode control, the **QR** patterns are based on the value you want to encode, which you can specify in the Text property.



## How to Obtain an Image from C1QRCode

The **C1QRCode** control is used to show the QR image. If you want to include the QR image in a document, you can use the **C1QRCode.Image** property to retrieve an image of the barcode.

To use the **C1QRCode** control, follow these steps:

1. Drag a **C1QRCode** control from the Visual Studio Toolbox to your form.
2. In the Properties window, set **C1QRCode's C1QRCode.Text** property to the value you want encoded. The patterns will change based on the value entered in the **C1QRCode.Text** property.

Take a look at the **QRCodeConstructor** sample provided with the product. Samples are installed by default to the Studio for WinForms\C1Barcode\VB and CS folders within *C:\Users\<username>\Documents\ComponentOne Samples* (Windows 7/Vista) or *C:\Documents and Settings\<username>\My Documents\ComponentOne Samples* (Windows XP).

This sample demonstrates how formatted information can be entered and converted into a QR code that can be printed or saved.

For example, the following code trims the text for a URL that is entered, returning a string with all leading and trailing white-space characters removed; checks the text to make sure is it less than 271 characters; and then creates the QR image based on the text entered.

```
Private Sub CreateURLCode()  
    _QRText = Trim(txtFormWebsiteAddress.Text)  
    If QrCodeSizeCheck() Then  
        _CodeBuilt = True  
        EnableButtons()  
    Else  
        DisplayLengthWarning()  
    End If  
End Sub
```

```
Return
End If
End Sub
```

When a URL is entered and "Create QR Code" is clicked, this is the resulting QR image:



## C1QRCode Encoding Limits

Although the QR specification includes 40 levels, or encodings, the C1QRCode control only implements levels 1 through 10. The table below summarizes the image sizes and content length supported by each level.

Correction Level: L (higher correction levels reduce encoding capacity)

Level	Size	Numeric	Alpha	Bin
1	21x21	41	25	17
2	25x25	77	47	32
3	29x29	127	77	53
4	33x33	187	114	78
5	37x37	255	154	78
6	41x41	322	195	134
7	45x45	370	224	154
8	49x49	461	279	192
9	53x53	552	335	230
10	57x57	652	395	271

**NOTES:**

The **Level** is the encoding and can be set to *Automatic*.

The **Size** is the image size in pixels.

Alphanumeric characters that are supported include [0-9][A-Z][\$%\*+-./:].

For details on the QR format, please see: <http://www.denso-wave.com/qrcode/qrstandard-e.html> and [http://en.wikipedia.org/wiki/QR\\_code](http://en.wikipedia.org/wiki/QR_code).

## Improving the C1QRCode Image Resolution

You can change the size of the symbols used to build the **QR** image by setting the **SymbolSize** property.

Simply select the **C1QRCode** control on the form and set the **SymbolSize** property in the Visual Studio Properties window to a value between **2** and **10**. Larger values will result in larger images which consume more space but may be easier to for some scanners to read.

The default value for the **SymbolSize** property is **3**, which looks like this:



You can set the **SymbolSize** property to any value between **2** and **10**. To give you an example of the difference in sizes, look at the following images:

**SymbolSize = 2**



**SymbolSize = 10**



## BarCode for WinForms Samples

Please be advised that this ComponentOne software tool is accompanied by various sample projects and/or demos which may make use of other development tools included with the ComponentOne Studios.

Samples can be accessed from the **ComponentOne Sample Explorer**. To view samples, on the desktop, click the **Start** button and then click **All Programs | ComponentOne | Studio for WinForms | Samples | BarCode Samples**. The following tables provide a short description for each sample.

### Visual Basic Samples

Sample	Description
PrintBarCodes	Demonstrates how to print barcodes created with the C1Barcode control. This sample uses the C1Barcode control.
QRCodeConstructor	Demonstrates how formatted information can be entered and converted into a <b>QR</b> code that can be saved or printed. This sample uses the C1QRCode control.

## C# Samples

Sample	Description
BarCodeSample	Demonstrates the different types of barcode encoding. This sample uses the C1Barcode control.
PrintBarCodes	Demonstrates how to print barcodes created with the C1Barcode control. This sample uses the C1Barcode control.