# Códigos de Barras en proyectos .NET

dBarcode .NET components provide the ideal way of adding barcodes to any .NET project. The discussion here focuses on Windows Forms projects in Visual Studio .NET, but the components may be used in any .NET or ASP.NET project.

Screenshot - *click to enlarge*

## Adding a dBarcode.NET Component to the Visual Studio ToolBox

To add a dBarcode Component to the Visual Studio ToolBox, display the ToolBox and select the Components tab. Right click on the Components pane and select Add/Remover Items.. from the pop-up menu displayed. A dialog box is displayed listing the currently installed components. Ensure that the .NET Framework Components page is displayed.

Push the Browse button and navigate to the location where you have installed or copied your dBarcode.NET component and select the DLL (e.g. AbcnetLib.dll).

Then push the Open button.

The list of installed components is now displayed, including your dBarcode.NET component. Ensure that the checkbox alongside the component name is checked. Now push the OK button.

Adding a dBarcode.NET component to a project.

With a project's form open in design mode drag the dBarcode.NET component icon from the toolbox onto the form.

The component icon appears on the panel below the form – it does NOT appear on the form itself. The instance of the component will be given a default name (eg Abcnet1) which appears in the properties panel when the component is selected. A single Form may contain any number of dBarcode.NET Components. The first to be added will be called Abcnet1, the second Abcnet2, and so on; the names may be changed by the user by modifying the Name property within the Properties box.

The properties panel also displays all other settable properties for the component, and these values will be used as defaults unless properties are changed programmatically within your project.

## Setting and retrieving property values programmatically

The dBarcode.NET Components may be operated entirely by setting or retrieving Property values programmatically.

Clicking on the dBarcode Component in the panel under the form when Visual Studio's Properties box is displayed will show the current settings for component's available properties. Most of these may be edited using the Properties box, or may have their values set from within the user's program by statements of the kind

```
Abcnet1.Caption="12345"          Visual Basic

Abcnet1.Caption="1234";          C#

Abcnet1.set_Caption("1234");     J#
```

dBarcode Component properties that are set AFTER a barcode has been created may be retrieved within user's programs by statements of the kind:

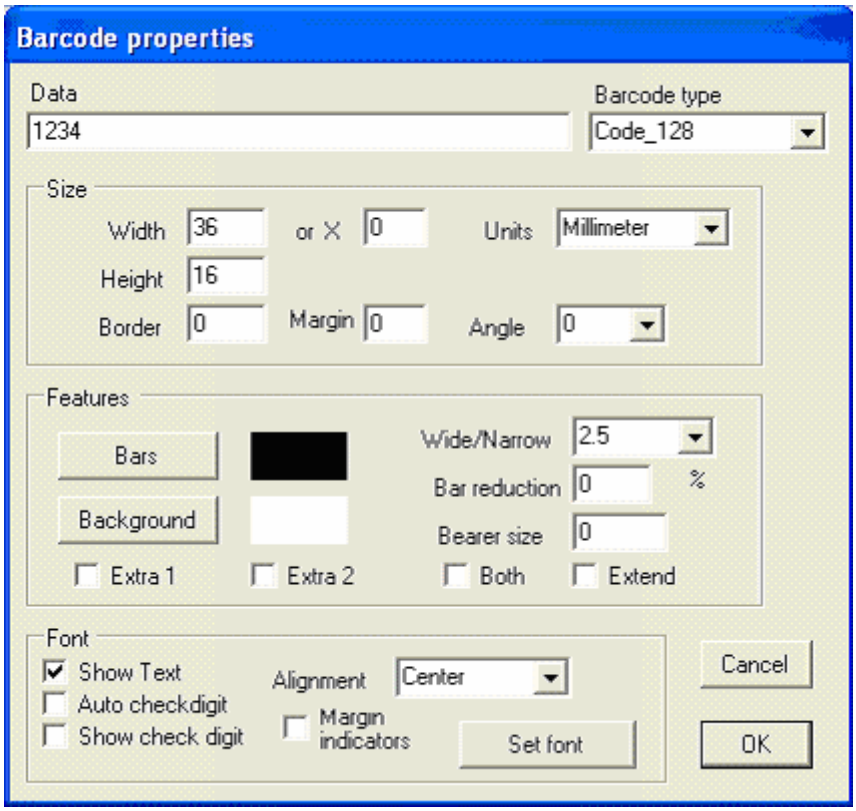
```
x=Abcnet1.Error        Visual Basic

x=Abcnet1.Error;       C#

x=Abcnet1.get_Error()  J#
```

## Setting properties through the Barcode properties dialog box

Using the Method AbcProperties() causes the Barcode properties dialog to be displayed. This displays all settable properties in a convenient form and enables changes to be made by selecting from drop-down lists or entering values into edit boxes, or summoning standard Windows dialogs for the selection of colors or fonts.




## Displaying a barcode on a form

To display a barcode on a form a PictureBox is used to hold the image.

Place a PictureBox Windows Forms control on the form and add some code to your program to take the barcode image from the dBarcode.NET component and use it as the Image for the PictureBox

For example:

```
private void DoBarcode()                    Visual Basic
{
      Dim g As Graphics = PictureBox1.CreateGraphics()
      Abcnet1.Caption="1234"
      PictureBox1.Image=Abcnet1.Barcode(g)
      g.Dispose()
}


private void DoBarcode()                    C#
{
      Graphics g=pictureBox1.CreateGraphics();
      abcnet391.Caption="1234";
      pictureBox1.Image=abcnet391.Barcode(g);
      g.Dispose();
}


private void DoBarcode()
{
      Graphics g=pictureBox1.CreateGraphics();
      abcnet391.set_Caption("1234");
      pictureBox1.set_Image(abcnet391.Barcode(g));
      g.Dispose();
}
```

## Printing a barcode image

Printing the image returned by the Barcode() call may be accomplished by any of the printing techniques available for Visual Studio.NET project. However, probably the most useful approach is to use the DrawImage() method in a PrintPage handler as illustrated below, and in the examples provided with the components:

### Visual Basic

```
Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object,
ByVal e As System.Drawing.Printing.PrintPageEventArgs) Handles
PrintDocument1.PrintPage
   e.Graphics.PageUnit = GraphicsUnit.Document '
   e.Graphics.DrawImage(Abcnet1.Barcode(e.Graphics), 300.0F,
300.0F)
   e.HasMorePages = False
End Sub
```

### C#

```
private void printDocument1_PrintPage(object sender,
System.Drawing.Printing.PrintPageEventArgs e)
{
      e.Graphics.PageUnit = GraphicsUnit.Document;
      e.Graphics.DrawImage(abcnet1.Barcode(e.Graphics), 500F,500F);
      e.HasMorePages = false;
}
```

### J#

```
private void pd_PrintPage(Object sender,
System.Drawing.Printing.PrintPageEventArgs e)
```

```
{
      e.get_Graphics().set_PageUnit(GraphicsUnit.Document);

e.get_Graphics().DrawImage(abcnet1.Barcode(e.get_Graphics()),500F,
500F);
      e.set_HasMorePages(false);
}
```

While the PageUnit  setting can be any of the allowed values, we have found that the most accurately sized barcodes are produced when the highest resolution setting (Document, equivalent to 300 units per inch) is used.

---

Para más información contacte con su commercial habitual en Danysoft en el 902 123146 o con info@danysoft.com