dLSoft

Active Barcode Components

By dLSoft

This manual was produced using *ComponentOne Doc-To-Help*.™

Contents

Active Barcode Components

Introduction

Active Barcode Components are Active-X/Automation Controls that allows barcode images to be created within the user's own software. A barcode image may be displayed on screen, printed on a printer, saved to a disk file, or passed to the Windows clipboard for incorporation into another Windows application.

Although primarily designed for Microsoft's Visual Basic, Active-Barcode Components may be employed with other Active-X-enabled applications, such as Delphi or the Visual Basic for Applications provided in Microsoft Office 97 and later. Examples provided in this manual are based on Visual Basic 6 and Office 97.

Users should note that the encoding calculations are performed when the component's image is created, by being used, accessed or copied. Information properties (such as PictureWidth) are not available until the image has been created.

The controls

There are several controls in the Active Barcode Components family, one for each of the most popular barcode types, and one which handle a wide range of different barcode types. The controls are:

Active Barcode Component Code 39
Abc39.ocx – for Code 39, Extended Code 39, Code 93 and Extended Code 93 barcodes

Active Barcode Component Code 128
Abc128.ocx – for Code 128, EAN 128, EAN-14 / UCC-14 (also know as DUN-14), and SSCC (Serial Shipping Container Code) barcodes

Active Barcode Component EAN/UPC
AbcEAN.ocx – for EAN-13, EAN-14, EAN-8, UPC-A, UPC-E, ISBN, ISSN, and ISMN barcodes

Active Barcode Component ITF
AbcITF.ocx – for 2 of 5, Interleaved 2 of 5, Matrix 2 of 5, ITF-14 and ITF-6 barcodes

Active Barcode Component Universal
Axbarcode.ocx – for most common barcode types (over 50 types supported; see the barcodes HELP file for a complete list).

With the exception of the CodeType property (which specifies the type of barcode to be generated) all controls share the same control properties, and in this manual all control are discussed using the name of the Active Barcode Control Universal. The values of the CodeType property for the individual controls is described under Properties.

Registering the control

Before an Active-X component may be used within a particular Windows installation it must be registered with the Windows System Registry. The Setup program that installs the product on your system provides automatic one-time registration. However, if you use the component on another system registration may NOT be automatic. Many container applications (Visual Basic, MS Office, etc.) offer facilities to register components by displaying a dialog and browsing for the component's file - axbarcode.ocx. As a last resort, the component may be registered manually be executing the following commands from within a command prompt:

To register the component

C:\path\regsvr32.exe C:\path\AXBARCODE.OCX.

Similarly the component may be un-registered using:

C:\path\regsvr32.exe C:\path\AXBARCODE.OCX -u.

If the component is moved or upgraded ir must be re-registered.

Once registered the component may be used within Visual Basic 6 (and other environments). The Active Barcode Component may be included on Visual Basic's Toolbar by checking the

Axbarcode OLE Custom Control

entry in the Custom Controls dialog accessible on the Tools menu. The OCX then appears as an icon on the toolbar.

Using Active Barcode Component in VB and .NET

Adding the control to a Visual Basic form

To add an Active Barcode Component control to a Visual Basic form, select Components from the VB Project menu. If axbarcode Active-X Control module is displayed in the list of available controls, then ensure that it is checked. If it does not appear in the list user the Browse button to locate it in its installation directory (or in Windows System or System32), then check its name in the list. Once checked, an icon representing the component will appear in the toolbox.

Clicking on the Active Barcode Component icon in the toolbox causes the cursor to change to a cross. Positioning the cross on a Visual Basic form, holding down the left mouse button and dragging the cursor down and to the left, then releasing the mouse button, causes a rectangle to be drawn on the form. An image of a (default) barcode will be shown within this rectangle. You may wish to use this image of the component. Alternatively you may hide the component (by setting its Visible property to false) and use the components methods and properties to create an image that is placed elsewhere on the form - as in the sample Visual Basic program that is supplied with the component and uses code of the form:

```
Axbarcode1.ImageHeight = i
Axbarcode1.ImageWidth = j
Axbarcode1.Xunit = 0
Axbarcode1.CodeType = type
Axbarcode1.Caption = Text
Image1.Picture = Axbarcode1.Picture
```

A single Visual Basic Form may contain any number of Active Barcode Components. The first to be added will be called Axbarcode1, the second Axbarcode2, and so on; the names may be changed by the user by modifying the Name property within the Visual Basic Properties box.

Alternatively an array of controls may be created using the Index property.

VB.NET and C#

The control may be added to the Visual Studio.NET Toolbox by right clicking on the Toolbox and selecting Customise Toolbox from the pop-up menu displayed, then checking the control in the list presented. The control will then appear as an icon on the Toolbox.

Setting and retrieving property values

The Active Barcode Component may be operated entirely by setting or retrieving Property values using the Basic language.

Clicking on the Active Barcode Component control on the form when VB'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

```
Axbarcode1.Source="12345"
```

Active Barcode Component properties that are set AFTER a Picture is created may be retrieved within user's programs by statements of the kind:

```
x=Axbarcode1.ErrorCode
```

Creating a barcode Picture within a program

Active Barcode Component produces its barcode image as a Windows metafile. This metafile is displayed directly in the area you create for the component - unless you set the Axbarcode.Visible property to FALSE.

To create a printable Picture object containing the barcode within a program the following steps are required:

1. Set those properties that Active Barcode Component requires to generate the barcode, the minimum of which are

Axbarcode1.CodeType - specifies the barcode type required

Axbarcode1.Caption - specifies the characters which make up the code

2. Allow the component's image to be displayed on the form (if required).

OR

Copy the components Picture property to an image on the form, eg.

```
If Axbarcode1.ErrorCode = 0 Then
   Image1.Picture = Axbarcode1.Picture
```

3. Printing the Barcode on the Printer

To print the barcode image on the printer use the PaintPicture method of the Printer object, eg.

```
Printer.ScaleMode = 6  ' sets printer scale to mm
Printer.PaintPicture Axbarcode1.Picture, 20, 20, Axbarcode1.PictureWidth,      Axbarcode1.PictureHeight
Printer.NewPage
Printer.EndDoc
```

4. Printing from VB.NET or C#

Printing from Managed code is accomplished using the PrintDocument control from the Visual Studio.NET ToolBox. Add the control to your application, then use it as illustrated below; in this example only the barcode image is printed.

```
  Private Sub print1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles print1.Click
    Try
      AddHandler PrintDocument1.PrintPage, AddressOf Me.PrintDocument1_PrintPage
      PrintDocument1.Print()
    Catch ex As Exception
      MessageBox.Show("An error occurred while printing", _
        ex.ToString())
    End Try
  End Sub

  Private Sub PrintDocument1_PrintPage(ByVal sender As System.Object, ByVal e As
System.Drawing.Printing.PrintPageEventArgs) Handles PrintDocument1.PrintPage
    Dim newImage As Image = Axbarcode1.Picture
    Dim x As Integer
    Dim y As Integer
    x = Axbarcode1.PictureWidth ' in mm
    y = Axbarcode1.PictureHeight ' in mm

    Dim ulCorner As New Point(50, 50) ‘ set position on page
    Dim urCorner As New Point(x + 50, 50)
    Dim llCorner As New Point(50, 50 + y)
    Dim destPara As Point() = {ulCorner, urCorner, llCorner}
    e.Graphics.PageUnit = GraphicsUnit.Millimeter
    e.Graphics.DrawImage(newImage, destPara)
    ' Indicate that this is the last page to print.
    e.HasMorePages = False
  End Sub
```

Using Active Barcode Component in Delphi

Importing the component into a Delphi Development System

To start using the Active Barcode Component in the Delphi IDE follow these steps:

From Delphi's Component menu choose Import ActiveX Control

Select Axbarcode ActiveX Control Module from the list of installed controls.

Push the Install button. You can then choose to import the control into an existing package, or create a new package.

When installation has finished, the Active Barcode Component icon will be available from the ActiveX toolbar.

To place the package on a Delphi form simply select the Active Barcode Component icon from the ActiveX toolbar and use the cursor to draw a position for it on the form. The Taxbarcode object is then available for use.

Using the component within Delphi

A single Delphi Form may contain any number of Active Barcode Components. The first to be added will be called Axbarcode1, the second Axbarcode2, and so on; the names may be changed by the user by modifying the Name property within the Delphi Object Inspector.

Each component added will be visible on the form. You may wish to use this image of the component. Alternatively you may hide the component (by setting its Visible property to false) and use the components methods and properties to create an image that is copied elsewhere on the form.

The principal steps involved in using a barcode image within Delphi are as follows:

1. Set those properties which Active Barcode Component requires to generate the barcode, the minimum of which are

Axbarcode1.CodeType - specifies the barcode type required
Axbarcode1.Caption - specifies the characters which make up the code

2. Allow the component's image to be displayed on the form (if required).

3. Printing the Barcode on the Printer

To print the barcode image on the printer StretchDraw the picture graphic to the Printer object, again using the PictureHeight and PictureWidth parameters to scale the picture to the Printer's canvas, eg.

```
{set the target rectangle for drawing on the printer}
{change from mm to pixels}
ii:=(Axbarcode1. PictureWidth*pScalex) div 100;
jj:=(Axbarcode1. PictureHeight*pScaley) div 100;
Rect.Left:=200;
Rect.Top:=200;
Rect.Right:=Rect.Left+ii;
Rect.Bottom:=Rect.Top+jj;
Printer.BeginDoc;
Printer.Canvas.StretchDraw(Rect, Axbarcode1.Picture.Graphic);
Printer.EndDoc;
```

where pScalex and pScaley are for converting the picture units to printer pixels.

Using Active Barcode Component in C++ Builder

Importing into a C++ Builder Development System

To start using the Active Barcode Component in the C++ Builder IDE follow these steps:

1. From Builder's Component menu choose Import ActiveX Control

2. Select Axbarcode ActiveX Control Module from the list of installed controls.

3. Push the Install button. You can then choose to import the control into an existing package, or create a new package.

4. When installation has finished, the Active Barcode Component icon will be available from the ActiveX toolbar.

To place the package on a Builder form simply select the Active Barcode Component icon from the ActiveX toolbar and use the cursor to draw a position for it on the form. The Taxbarcode object is then available for use.

Using the component within C++Builder

A single Builder Form may contain any number of Active Barcode Components. The first to be added will be called Axbarcode1, the second Axbarcode2, and so on; the user may change the names by modifying the Name property within the Builder Object Inspector.

Each component added will be visible on the form. You may wish to use this image of the component. Alternatively you may hide the component (by setting its Visible property to false) and use the components methods and properties to create an image, which is copied elsewhere on the form.

The principal steps involved in using a barcode image within Builder are as follows:

1. Set those properties which Active Barcode Component requires to generate the barcode, the minimum of which are

Axbarcode1.CodeType - specifies the barcode type required
Axbarcode1.Caption - specifies the characters which make up the code

2. Allow the component's image to be displayed on the form (if required).

3. Printing the Barcode on the Printer

To print the barcode image on the printer StretchDraw the picture graphic to the Printer object, again using the PictureHeight and PictureWidth parameters to scale the picture to the Printer's canvas, eg.

```
  //set the target rectangle for drawing on the printer
  //change from mm to pixels
Prntr = Printer();
i=GetDeviceCaps((Prntr->Handle),LOGPIXELSX);
pScalex=(1000*i) / 254; //100 * pixels per mm
i=GetDeviceCaps((Prntr->Handle),LOGPIXELSY);
pScaley=(1000*i) / 254; //100 * pixels per mm
ft=AxbarcodeProxy1->PictureWidth;
ii=int((ft*pScalex)/ 100.0);
ft=AxbarcodeProxy1->PictureHeight;
jj=int((ft*pScaley) /100.0);
Rect.Left=200;
Rect.Top=200;
Rect.Right=Rect.Left+ii;
Rect.Bottom=Rect.Top+jj;
Prntr->BeginDoc();
Prntr->Canvas->StretchDraw(Rect, AxbarcodeProxy1->Picture->Graphic);
Prntr->EndDoc();
```

where pScalex and pScaley are for converting the picture units to printer pixels.

Using Active Barcode Component in a Container

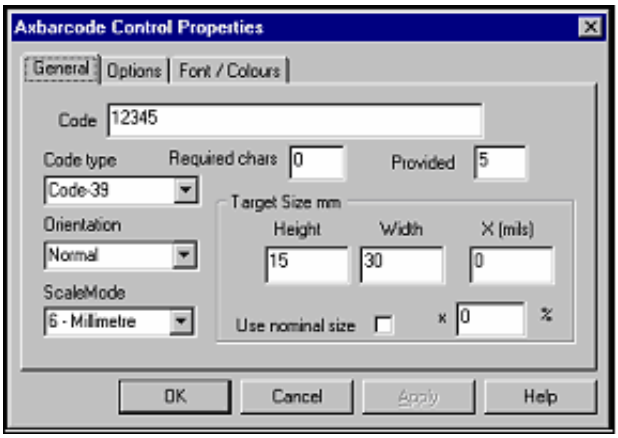
Active Barcode Component may be inserted into Container programs (such as WordPad) generally by selecting Insert Object from the EDIT or INSERT menus and choosing Axbarcode Control from the list of options presented.

The properties of the barcode image may be set by selecting the barcode and either double-clicking on it, or by summoning its Property Pages dialog from the Edit - Axbarcode Control Object menu item.

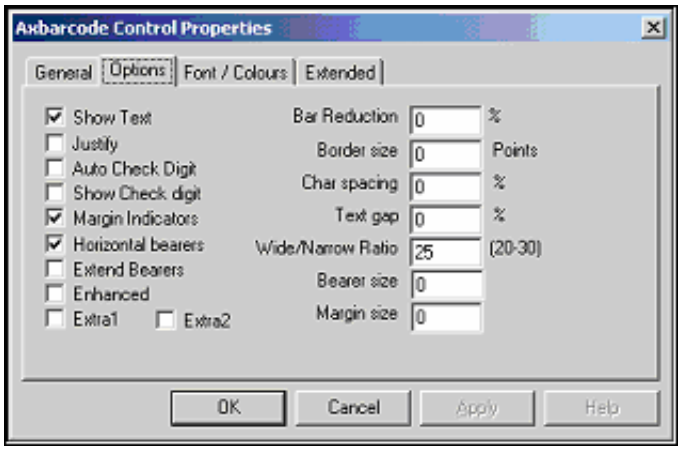
The Property Pages dialog provides a tabbed dialog box containing four pages.

The General Page provides the user with the opportunity to set the Codetype, the Code value (the Caption), and the target height and width. Checkboxes are provided to allow the Nominal size to be specified (valid only for barcodes which have a nominal size), and for the check digit to be calculated automatically (for those codes which have check digits. An X unit box allows the width of the barcode to be calculated by the control, using the X unit size The X unit is the width of the thinnest bar (in mils - 0.001 inches, not mm). If X is provided as 0 then the width of the barcode is determined by the content of the width box.

Details of the properties are given in the Reference section.

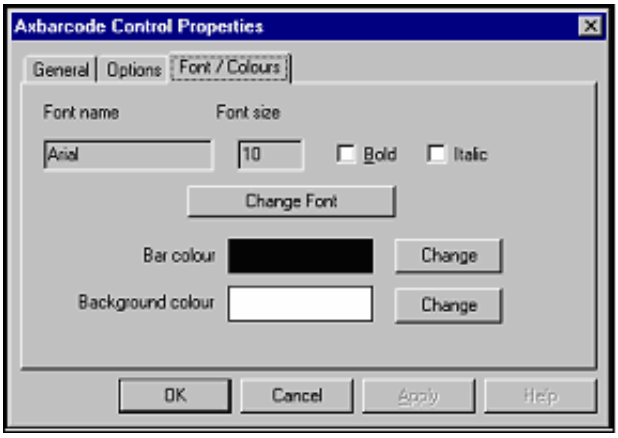


The Options page enables most of the remaining properties to be specified, either through checkboxes or by specifying numerical values in the edit boxes.

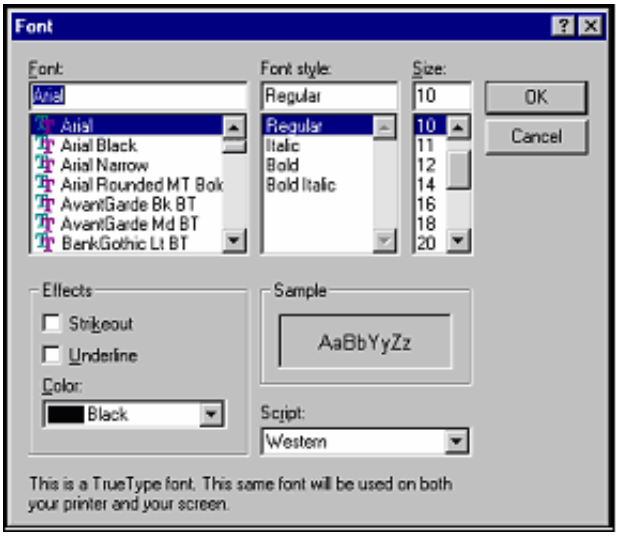


The Font/Colours page allows the foreground (bars and text) colour and the background colour to be selected from the system colours available, and the Font properties (for text under the barcode image) to be selected from the normal Windows Font dialog

.


Axbarcode Control Properties
General
Options
Font / Colours
Font name
Font size
Arial
10
Bold
Italic
Change Font
Bar colour
Change
Background colour
Change
OK
Cancel
Apply
Help


Color
Basic colors:
Custom colors:
Define Custom Colors >>
OK
Cancel



Note that the size of the font selected will be correctly reproduced only when the metafile is displayed at the size specified in the target height and width boxes on the Code page. The metafiles image created if fully resizable, so changing the size of the barcode image will also change the actual size of the font used for the text.

Each Property page contains an OK button which, if pushed causes the current contents of all the pages to be used in re-creating the barcode image. The pages also contain a Cancel button that may be used to close the Property pages dialog and discard any changes made.

Using the Active Barcode Component in Access

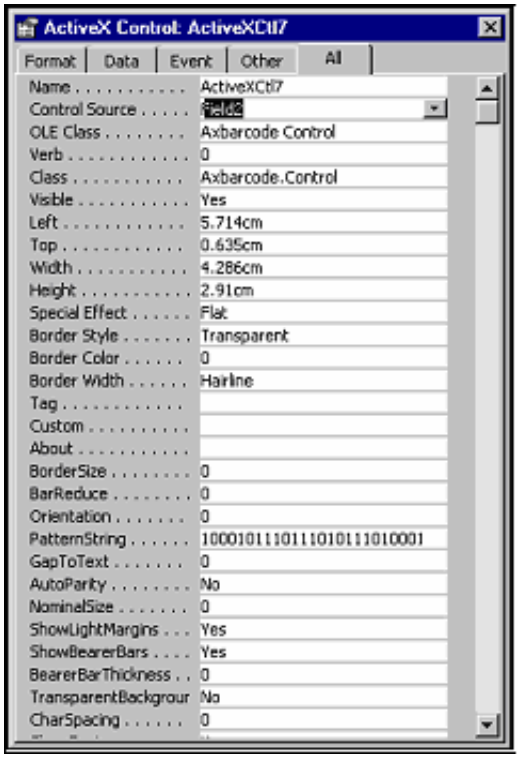
Microsoft Access is a database that holds data in tables. When an Active Barcode Component is added to an Access Form or Report as an ActiveX control, the component's Properties list allows the user to select one of the Table fields as a Source of the components data - ie the data to appear in the barcode.

The Record Source may be selected from a drop-down list in the form or report's properties list, and the barcode's Control Source may be selected from the drop down list in the component's properties list. All other properties (code type, size, font, etc.) can be selected from the other entries in the Access Properties list at design time.

This allows barcode creation to be fully automatic for every record in a database.

Of course, a barcode image that is the same on every record may also be created - by ensuring that the Control Source entry on the component's properties list is empty, and the required data is entered for the Caption property.

A sample Access database is included with the distribution



Active Barcode Component properties for a barcode image on a form or report may also be set at run-time using Visual Basic module code to modify the control's properties.

Using the Active Barcode Component in Word

There are two different ways of using the Active Barcode Component within Microsoft Word.

1. The component may be inserted as an Object - by choosing Object from the Insert menu, or

2. The component may be added as an ActiveX control from the Control Toolbox.

The component behaves in rather different ways in these two cases.

When inserted as an object the component behaves in just the same way as when inserted into any other container - such as WordPad. Double-clicking on the barcode image brings up the property pages that allow the barcode properties to be specified. (See Using Active Barcode Component in a Container for details.)

When added as an ActiveX Control the component is programmable from within Word's Visual Basic Editor. A number of simple macros are included in the ABC1.DOC sample supplied with Active Barcode Component. While the sample macros are run from the Tools - Macros menu, single keystroke shortcuts can be added by choosing Tools - Customise and then pushing the Keyboard button. Scroll the categories list down to Macros and then choose a key to be used to initiate each macro. That's all there is to it.

Of course the macros supplied are rather rudimentary. Almost anything can be done by setting properties and calling methods.

Setting the image size

Active Barcode Component creates a barcode image at a size specified by the user. However, the metafile image is fully resizeable by the user - so can be displayed or printed at a size other than that originally created.

There are three ways of specifying the size at which the image is created:

1. For a limited number of barcode types there is a Nominal size (ie. A size specified by the standards body responsible for that barcode type). EAN, UPC and ITF barcodes have nominal sizes. In general it is recommended that these barcode types be reproduced either at the nominal size or within a fairly narrow range of sizes around nominal (typically 80 - 120% of nominal). These barcode types may have their size specified by setting the NominalSize property. After the image has been created the PictureWidth and PictureHeight properties contain the actual width and height of the barcode image.

2. Any barcode type may have the image size specified by setting the xunit property to the width of the narrowest element in the barcode (narrowbar) when the NominalSize property is set to 0. Xunit values are specified in Mils. In this case the initial setting of the ImageWidth property is ignored. After the image has been created the PictureWidth and PictureHeight properties contain the actual width and height of the barcode picture.

3. Any barcode type may have the image size specified by setting the ImageWidth and ImageHeight properties when the NominalSize property is set to 0 and the xunit property is set to 0.

Distributing Programs using Active Barcode Component

Any program using one of the Active Barcode Components must have access to the component in OCX file (e.g. AXBARCODE.OCX). No other files are required by the control, although the target computer must has a suitable version of the operating system's OLE system files installed (OLEPRO32.DLL on Windows 95, and OLEAUT32.DLL on Windows NT). Any OLE enabled software, such as Microsoft Office, will have installed these files.

Single user versions of these products are licensed for use only on a single computer.

Developer versions of the OCX run-time file (e.g. AXBARCODE.OCX) may be distributed Royalty-free for up to 10,000 copies. The Design time licence may not be distributed.

Barcodes

Introduction

dLSoft products support a wide range of barcode types and we endeavour to keep up to date with barcode specifications. However, it is important to understand that the standards specified for barcodes have arisen from a wide range of sources, and some barcode specifications have been modified over a period of time. Furthermore some barcode types have been largely superceeded by more modern code types, usually because modern types have a higher reliability.

In these notes we aim to provide:

1) Details of the barcode types supported in the versions Active Barcode Component, the full list of barcode types support by Active Barcode Component Universal will be found in the barcode HELP file which accompanies the component.

2) Some general information about the codes you need to provide to produce satisfactory barcode images

3) The CodeType values required to access the barcode types if you are programming the component directly. Programmers who are using Active Barcode Component Universal and multiple code types are advised to print out the code type table from the barcode HELP file.

General points

Several fundamental characteristics of barcodes need to be understood by users of dLSoft barcode products:

1. The thickness of bars in barcodes is important. dLSoft barcode products will refuse to create a barcode image if the bar thickness within the metafile becomes too small. However, even when dLSoft barcode products creates an image you may resize it within another application so that when it is printed by the other application its lines may too small for the printer's resolution. Consequently it is essential that you check that a printed barcode is readable using an appropriate scanner or reader.

Barcodes printed by laser printer will, in general, be printed correctly, but codes printed by matrix printers must be reproduced at a large enough scale that the barcodes unit size is at least as large as the printer's pins.

Bar reduction: All dLSoft barcode products allow the thickness of bars to be reduced (for example to allow for ink spread during wet ink printing processes), but this adjustment should only be made when the knowledge of the extent of reduction required is available. Random guesses usually produce unreadable images!

2. Many barcode types may use codes only of a specific length. (e.g. EAN13 requires 13 digits in the code). Some barcode type use specific digits of the code as a checksum - so not every combination of digits can form a legal barcode.

dLSoft barcode products can optionally calculate checksum digits, requiring only the other digits to be entered by the user. Furthermore most coding schemes are limited to 32 characters or less.

3. The barcode types supported in this release are shown in the barcodes table below. If you plan to use a specific barcode type you should examine the notes on that type before printing any barcode images.

4. Users should be aware that it is possible to generate barcodes of a specific type and find that normal retail scanners are unable to decode the images. This does not necessarily mean that there is anything wrong with the barcode image. Most scanners aimed at the retail market are not programmed to interpret barcode codes reserved for other (eg. military) use.

5. The Extra options - Extra1 and Extra2. These options are used only for a limited number of barcodes that have "unusual" features. The effect of these options is described under the barcode types that use them. For all other barcode types these options may be ignored or set to 0.

Barcode types supported

The types of barcodes supported by this release of Active Barcode Component Universal are listed in the Barcodes Help file which can be found in the installation directory. The code# in the table of barcode types represents the CodeType value used in calls to the Active Barcode Component Universal. Barcode types supported by the other components are listed under the CodeType property below.

Telepen provides codes for upper and lower case letters and most printable characters. Only Extended Code 39, Extended Code 93 and the EAN 128 and Code 128 codes provide symbols for the full ASCII character set.

There are many named barcode types which are actually derivatives of major types. To avoid the table (and user-programming) becoming excessively complex, both the table and calls to the library report only the generic name.

For example: The JAN coding scheme is a variant of the EAN scheme, using a different country code (the first two digits).

There are several coding schemes (such as DEFCON) which are actually Code 39, and some countries use Code 128 under other names for mail tracking (as in the UK).

Location numbering

After March 1st 1995, EAN International agreed that all numbering organisations would standardise on the product numbering (ie standard EAN-13) check digit algorithm when calculating check digits for location numbers. The cut-off date for using older check digit algorithms was January 1st 1997.

When producing location numbering barcodes of the EAN-13 pattern users should note that the recommended check digit algorithm is that produced by the Active Barcode Component. Older algorithms are no longer supported.

Reference

Summary of Properties and Methods

These most important properties and methods are outlined below, and followed by sections in which the properties and methods are defined in detail.

Component properties.

The properties that can be set to generate an image

Axbarcode1.AutoParity specifies whether any check digits are calculated automatically

Axbarcode1.BackColor the colour behind the bars

Axbarcode1.BarRatio specifies the ratio of wide/narrow bar (times ten) for some barcode types

Axbarcode1.BarReduce specifies the percentage reduction in bar thickness (useful for allowing for ink spread in wet-ink printing processes.

Axbarcode1.BearerBarThickness specifies the thickness of bearer bars for those barcodes that may have bearer bars

Axbarcode1.BorderSize specifies the thickness of any border around the image (in points)

Axbarcode1.Caption specifies the characters that make up the code

Axbarcode1.CharSpacing the percentage of the barcode width over which the text under the barcode should be spread out.

Axbarcode1.CodeType specifies the barcode type required

Axbarcode1.ExtendBearers Allows bearer bars to extend into light margins.

Axbarcode1.ForeColor the colour of the bars and any text under the bars

Axbarcode1.Font The font used to render any text under the barcode

Axbarcode1.FontSize, FontName, FontBold, FontItalic are alternative ways of specifying font characteristics

Axbarcode1.ImageHeight required target height of barcode image (in units determined by the setting of ScaleMode)

Axbarcode1.ImageWidth required target width of barcode image (in units determined by the setting of ScaleMode)

Axbarcode1.JustifyText specifies the text justification for human readable text under the barcode

Axbarcode1.MarginSize specifies the size of the light margins

Axbarcode1.NominalSize specifies a percentage of the barcode's standard size at which the image should be created. If specified the calculated value overrides the ImageHeight and ImageWidth parameters.

Axbarcode1.Orientation specifies the orientation of the barcode image

Axbarcode1.PictureHeight the height of barcode picture generated (in units determined by the setting of ScaleMode)

Axbarcode1.PictureWidth the width of barcode picture generated (in units determined by the setting of ScaleMode)

Axbarcode1.ScaleMode specifies the units of ImageHeight and ImageWidth and PictureHeight and PictureWidth parameters (default is 6 = mm)

Axbarcode1.ShowBearerBars determines whether (or how many) bearer bars are displayed

Axbarcode1.ShowText specifies that the text content of the barcode should be displayed under the bars

Axbarcode.ShowCheckDigit specifies whether any automatic check digit is displayed (for those barcode type which permit this)

Axbarcode1.ShowLightMargin specifies whether light margin indicators should be displayed for those barcode types that support these

Axbarcode1.xunit specifies the thickness of each barcode element in mils

Many of these properties have default values (see the reference section), so do not require changing if you can make do with the default values. The properties that must be set for you to obtain a barcode are

Axbarcode1.CodeType specifies the barcode type required

Axbarcode1.Caption specifies the characters that make up the code

A typical example of setting these properties is:

Axbarcode1.CodeType=8 rem code type is 8 for Code-39 type barcode

Axbarcode1.Caption="123456" rem code is 123456

Methods of using the barcode image

To paint the control's Picture on the form use the PaintPicture method, or draw your own bars using the Printers Line method. Using the PaintPicture method the code will look like this:

```
If Axbarcode1.ErrorCode = 0 Then
   Form1.PaintPicture Axbarcode1.Picture, 20, 20, i, j
```

where the (20,20) specifies the coordinates on the form of the top left-hand corner of the barcode image, and i and j represent the required width and height of the image respectively.

The control's Picture may be passed to the clipboard using

```
I = Axbarcode1.CopyImage()
```

The Visual Basic Picture object contains a metafile image of the generated barcode. In the event of an error in barcode image generation the image is empty, and the ErrorCode property is set to a non-zero value.

The control's Picture may be printed using the PaintPicture method of the Printer object, eg.

```
Printer.ScaleMode = 6  ' sets printer scale to mm
Printer.PaintPicture Axbarcode1.Picture, 20, 20,  Axbarcode1. PictureWidth,      Axbarcode1. PictureHeight
Printer.NewPage
Printer.EndDoc
```

Where the (20,20) refers to the top, left coordinates of the barcode image on the printer's page.

Note that the PrintForm method may also be used, although on some systems this may show images of unsuitable resolution.

For printing under VB.NET or from C# see the example in Creating a barcode Picture under Using Active Barcode Component in VB or .NET.

The image created by the Active Barcode Component may also be saved in a file in a graphics format using the SaveImage(filename) method. The image type must be specified by using the permitted filename extensions - .wmf, .bmp, .eps, .pcx, png .or .gif.

Active Barcode Component Properties

Active Barcode Component properties fall into two categories - those that are required to specify characteristics of the barcode image, and those that provide information about the barcode.

Required properties

The following properties are required - although all are provided with default values.

AllowRightProps

Type BOOL

Default: FALSE

Allowed values: TRUE (allows the property pages to be displayed on a right mouse button click within the controls visible area)

FALSE (property pages not displayed)

AutoParity

Type: BOOL

Default: FALSE

Allowed values: FALSE (checkdigit characters not calculated)

TRUE (checkdigit characters calculated and appended to code for appropriate code types)

BackColor

Type: LONG

Default: &H00FFFFFF&

Allowed values: 0 (black)

to &H00FFFFFF& (white)

Sets the colour of the image background. This value may be over-ridden by the Transparent property.

BarRatio

Type: SHORT

Default 25

Allowed values: 20 - 30

This setting allows some barcode types to have the Wide bar/Narrow bar ratio modified. The value is 10 times the ratio, so the default ratio is 2.5

Applies mainly to Code 39 and Interleaved 2 of 5 barcodes.

BarReduce

Type: SHORT

Default: 0 the thickness of each line drawn on the barcode image is reduced by this percentage amount. This property may be used to compensate for ink spreading during wet-ink printing.

Allowed values: 0 - 50 (%)

BearerBarThickness

Type: SHORT

Default: 0

Allowed values: 0 - 10000

Allows the thickness of bearer bars for those barcode which support bearers to be set in units of the current ScaleMode.

Note that for most barcode types the number of bearer bars displayed depends on the setting of the ShowBearerBars property. If ShowBearerBars is True the bars are displayed above and below the barcode. If ShowBearerBars is False then only a single bearer bar is displayed above the barcode.

BorderSize

Type: SHORT

Default: 0

Allowed values: 0 (no border)

to 255 in points

Caption

Type: BSTR

Default: "12345"

Allowed values: Any text string.

Note: only text strings recognised as valid barcodes will result in a barcode picture. Illegal text string will cause an ErrorCode value to be set.

The Caption should be the last property set - as setting this property causes the barcode image to be recreated.

CharSpacing

Type: SHORT

Default: 0

Allowed values 0 and 10 - 100

The percentage of the barcode width, which the text under the barcode will take up. A value of 0 causes the characters to be displayed with their "normal" spacing. It is the users responsibility to ensure that the value provided is sensible in comparison with the fontsize.

CodeType

Type: SHORT

For Active Barcode Control Universal

Default: 8

Allowed values: The range of values defined in the barcode types table in the Barcodes HELP file.

For all other controls

Default: 0

Allowed values:

Active Barcode Control Code 39
0 – Code 39
1 – Extended Code 39
2 – Code 93
1 – Extended Code 93

Active Barcode Control Code 128
0 – Code 128
1 – EAN 128
2 – EAN-14/UCC-14
3 – SSCC

Active Barcode Control EAN/UPC
0 – EAN-13 / JAN-13 etc.
1 – EAN-8
2 – EAN-13 + 2 digits
3 – EAN-13 + 5 digits
4 – UPC-A
5 – UPC-E
6 – UPC-A + 2 digits
7 – UPC-A + 5 digits
8 – EAN-8 + 2 digits
9 – EAN-8 + 5 digits
10 – UPC-E + 2 digits
11 – UPC-E + 5 digits
12 – ISBN
13 – ISSN
14 – ISMN
15 – EAN-14

Active Barcode Control ITF
0 – Interleaved 2 of 5
1 – ITF-14
2 – ITF-6
3 – 2 of 5
4 – Matrix 2 of 5

ExtendBearers

Type: BOOL

Default: FALSE

When TRUE allows bearer bars to extend into light margins. When FALSE bearer bars cover the bars only.

Extra1

Type: BOOL

Default FALSE

These additional properties are not normally used. However, they do provide additional functions for a limited number of specific barcode types. See Barcodes HELP for details.

Extra2

Type: BOOL

Default FALSE

These additional properties are not normally used. However, they do provide additional functions for a limited number of specific barcode types. See Barcodes HELP for details.

Font

Type: FONT

Default: Arial 10 point

Allowed values: Any accessible TrueType font.

Font details may be specified using the font parameters, eg.

Axbarcode1.FontName="Arial"
Axbarcode1.FontSize=8

ForeColor

Type: LONG

Default: 0

Allowed values: 0 (black)

to &H00FFFFFF& (white)

Set the colour of the image foreground, i.e. the bars and text colour.

GapToText

Type: SHORT

Default: 0 A value given as a percentage of the text font point size by which text is raised within its bounding rectangle.

Allowed values: 0-100 (%)

Note: For most applications this value should be set to 0.

ImageHeight

Type: FLOAT

Default: 20 mm

Allowed values: 5 mm (minimum)

to form height in units determined by the setting of ScaleMode

The target height of the barcode image. Note that this is the target size of the resizable metafile image. The font height will be as requested if the metafile is reproduced at this size. If the metafile is resized to say double this value then the font will also be increase in absolute size. Also this value will be placed in the metafile header for those applications that can reproduce metafiles at their specified size.

ImageWidth

Type: FLOAT

Default: 25 mm

Allowed values: 0.1 - 30,000

to form width in units determined by the setting of ScaleMode

The target width of the barcode image. Note that this is the target size of the resizable metafile. The font height will be as requested if the metafile is reproduced at this size. If the metafile is resized to say double this value then the font will also be increase in absolute size. Also this value will be placed in the metafile header for those applications that can reproduce metafiles at their specified size.

JustifyText

Type: SHORT

Default: 0

Allowed values 0,1 and 2

JustifyText sets the justification of any text displayed under the barcode. A value of 0 centres the text, 1 gives left justification and 2 gives right justification. Justification is to the outside of the light margins, so no justification can be used when light margin indicators are displayed.

Left

Type: SHORT

Default: value determined when user adds control to a form.

Note this is the Left position of the Control's rectangle.

MarginSize

Type: FLOAT

Default: 0

Allowed values: 0 - 20000

The MarginSize property sets the Light Margin space on either side of a barcode image. The units are the units specified by the current ScaleMode.

NominalSize

Type: SHORT

Default: 0 When this property is set to a value between 50 and 200 the width of the image created will be scaled to generate that percentage of the barcodes Standard. When this property is 0 no scaling is applied.

Allowed values: 0 or 50-200

Note that this feature is relevant only for some EAN, UPC and ITF code types. It is ignored for barcode types that do not have a standard size.

Orientation

Type: SHORT

Default: 0 The value of this parameter determines the orientation of the barcode image created.

Allowed values: 0 = normal orientation

1 = image rotated left by 90 degrees (clockwise)

2 = image rotated right by 90 degrees (anti-clockwise)

3 = image inverted.

Note that the rotation of text is only supported by TrueType and other rotatable fonts. Note also that some applications do not correctly handle metafiles that contain rotated text.

SaveImageFile

Type: LPCTSTR

No default value.

When set the current barcode image is saved to the specified filename, exactly as when using the SaveImage method.

Eg. Axbarcode1.SaveFileImage="c:\barimage.wmf"

A fully qualified filename is required, and the extension is taken to determine the type of graphic file saved. See the SaveImage method for details.

ScaleMode

Type SHORT

Default: 6

The scaling mode applied to values of ImageWidth and ImageHeight values.

Allowed values:

0 - Windows HIMETRIC (0.01 mm)
1 - Twips
2 - Points

3 - Windows TEXT
4 - Windows HIENGLISH (0.001 inches)
5 - Inches
6 - millimetres
7 – centimetres

ShowBearerBars

Type: BOOL

Default: FALSE

Allowed values: TRUE or FALSE

When TRUE causes bearer bars to be draw above and below the barcode. When FALSE no bearer bars are drawn if the BearerBarThickness property is zero. Otherwise a FALSE setting causing only a single bearer bar to be displayed above the barcode.

ShowCheckDigit

Type: BOOL

Default: 0 When set to 1 this property causes the any automatically calculated check digit to be included in the any text displayed along with the barcode. When set to 0 the check digit is not displayed.

Note that this property has no effect on those codetypes for which the checkdigit display is mandatory (including EAN and UPC codes).

ShowLightMargins

Type: BOOL

Default: FALSE A value of TRUE causes the light margin indicators to be displayed. For some EAN barcodes there are recommended ways for the margin indicators to be shown on the image, and the Prefix code and checkdigit for UPC-A and UPC-E codetype to be displayed in the light margins. A value of FALSE prevents the display of the light margin indicators.

Note that for ITF files specifying LightMargins=1 causes the border bars between the extremes of the bearer bars to be drawn. The required light margin distances are always included in ITF codes.

ShowText

Type: BOOL

Default: TRUE

Allowed values FALSE (text version of the code is NOT included in the barcode image)

TRUE (text version of the code IS included in the barcode image)

Top

Type: SHORT

Default: value determined when user adds control to a form.

Note this is the Top position available for the Control's icon. It is not related to the barcode picture. In most case the Control will be set to invisible, so this parameter may be ignored.

TransparentBackground

Type: BOOL

Default: FALSE

Allowed values TRUE or FALSE. When set to TRUE this property causes the barcode image to be created on a transparent background (ie. and background colour is ignored). The property should be set to 1 only by users who are confident that the final barcode image will be produced on a background that does not interfere with barcode scanning.

Units

Type: BSTR

Returns the current units in words, as set using the ScaleMode parameter.

UseEnhanced

Type: BOOL

Default: TRUE

Allowed values: TRUE and FLASE

When TRUE, an Enhanced metafile image is used to draw the barcode image on the visible control. When FALSE a standard Windows Metafile is used.

Width

Type: SHORT

Default: value determined when user adds control to a form.

Note this is the Width available for the Control's icon. It is not related to the barcode picture. In most case the Control will be set to invisible, so this parameter may be ignored.

xunit

Type: SHORT

Default: 0

Allowed range: 10 - 255

The xunit property may be used to specify the width (in Mils) of the smallest element in the barcode.

Note that setting this property to a value other than 0 causes the control to resize itself to a width calculated from the number of X units in the barcodes. Using values smaller than 8 will produce a barcode image, but that image will not meet standard specifications and may not scan.

Information properties

CodeName

Returns: the barcode name of the current barcode type as an LPSTR string.

CodeNameB

Returns: the barcode name of the current barcode type as a BSTR string.

ErrorCode

Type: SHORT

Returns a value representing the error code if a valid barcode image cannot be created. Otherwise returns 0.

Read only. Do not set this property.

HasParity

Type BOOL

Default: returns TRUE for code types that allow the inclusion of a checksum, otherwise returns FALSE.

Note that this property returns FALSE for EAN 128 and Code 128 in spite of the fact that these codes define checksum digits. This is because these codes cannot be used without the check digits and these digits are always calculated by the control and are not under user control. Check digits for these codes are not displayed.

Read only. Do not set this property. The value of this parameter can be determined following the setting of code type.

Nrequired

Type: SHORT

Returns: 0 or the number of characters required for a valid barcode of the current type. 0 indicates any number of characters permitted.

Read only. Do not set this property. The value of this parameter can be determined following the setting of the code type

The value returned depends on the current setting for AutoParity - ie. the value specifies the number of characters that need to be provided by the user.

Pattern

Type: BSTR

Returns: A Visual Basic string containing a pattern of 0s and 1s, where 0 represents a space of minimum width and a 1 represents a bar of minimum width. Within Visual Basic the 1s may be used to indicate that a bar (filled rectangle) must be drawn.

Read only. Do not set this property.

Picture

Type: PICTURE

Returns: The Picture of metafile type returned by the Active Barcode Component. This contains the image of the barcode and (if enabled) its text.

Read only. Do not set this property.

PictureHeight

Type: FLOAT

Returns: the height of the picture generated (in units determined by the setting of ScaleMode)

Read only. Do not set this property.

Note that PictureHeight differs from ImageHeight if the barcode image is rotated left or right.

PictureWidth

Type: FLOAT

Returns: the width of the picture generated (in units determined by the setting of ScaleMode)

Read only. Do not set this property.

Note that PictureWidth differs from ImageWidth if the barcode image is rotated left or right.

String2

Type: BSTR

Returns the Caption property plus any additional characters generated as check digits.

Read only. Do not set this property.

Methods

CopyBitmap()

This method causes a barcode image to be created and copied to the Windows clipboard. CopyImage copies the barcode image as a Windows metafile, and CopyBitmap copies the image as a bitmap. Both functions return 0 if an error occurs and one if the copy is successful.

CopyImage()

This method causes a barcode image to be created and copied to the Windows clipboard. CopyImage copies the barcode image as a Windows metafile, and CopyBitmap copies the image as a bitmap. Both functions return 0 if an error occurs and one if the copy is successful.

GetError()

Type: BSTR

Syntax: GetError(LPINT Errorcode)

The GetError(Errorcode) method returns the text (a BSTR string) of the error message corresponding to the Errorcode value. So a string variable may be filled with the error message using

X$=Axbarcode1.GetError(Axbarcode1.ErrorCode)

GetTypeName()

Type: SHORT

Syntax: GetTypeName(int, BSTR)

Returns: the barcode name of the current barcode type as a BSTR string in the calling parameter.

May be determined after setting the code type (eg. For filling a list box). Return a NULL string (zero length) for invalid code type.

GetTypeNameB()

Type: BSTR

Syntax: GetTypeName(int)

Returns: the barcode name of the current barcode type as a BSTR string.

May be determined after setting the code type (eg. For filling a list box). Return a NULL string (zero length) for invalid code type.

Refresh()

This method causes the barcode calculation to be repeated and the Image (if being shown) to be redrawn.

SaveImage(filename)

When called with a valid filename (including extension), this method copies the barcode image in a graphics format to a named disk file. The filename parameter must contain the fully qualified filename of the file that will receive the image. The specified directory path must exist. If the file exists it will be overwritten; if it does not exist it will be created. A return value of 0 indicates an error.

The extension of the filename specifies the type of file to be created. .WMF causes a Windows metafile image to be saved. .BMP specifies a Bitmap file. [Additional file types: .EPS specifies an encapsulated PostScript file (as a bitmap), .PCX a Paintbrush format bitmap, PNG a graphic format for web use, and GIF Compuserve graphic file - the latter requiring a user licence from Unisys.]

Note that the Property SaveImageFile may also be set to a filename to initiate the saving of a graphic file.

ShowProps()

Displays the property pages for the control.

Events

The following events are exported by the control and may be used by the container:

DblClick() – mouse double click

KeyDown (short KeyCode, short Shift) – a key pressed down; KeyCode identifies the keycode and Shift represents the state of the shift key

KeyUp (short KeyCode, short Shift) – a key released

MouseDown (short Button, short Shift, OLE_XPOS_PIXELS x, OLE_XPOS_PIXELS y) – a mouse button is pressed, Button identifies the mouse button, Shift represents the state of the shift key, a and y identify the location of mouse cursor.

MouseUp (short Button, short Shift, OLE_XPOS_PIXELS x, OLE_XPOS_PIXELS y) – a mouse button is released.

Error codes

The Property Axbarcode1.ErrorCode AFTER Axbarcode1.Caption has been set may contain an error code if the component was unable to generate a barcode Picture. The property value will be one of the following:

0 No error
1 Wrong code length
2 Unrecognised code type
3 Wrong add-on code length
4 Illegal character in code
5 Error in embedded code
6 Generated line width less than 1 unit
7 Invalid text font
8 Invalid device context
9 Invalid Caption property

A Note on Metafile pictures

The picture images placed on the clipboard by dLSoft's Active Barcode Component's CopyImage method are ANISOTROPIC metafiles. This means that they can be resized within applications (usually by dragging a corner).

While the barcode bars can be resized over very wide ranges, any text included within the image may not resize as expected. In general changing the height of the image by resizing within another application will change the fontsize used to render the text. Changing the width of the image within another application may cause the position of any text under the barcode to change.

The use of TrueType fonts is recommended to prevent unusual effects caused by resizing of text. Furthermore, the use of Enhanced Metafiles has the advantage of producing images in which the fonts scale more smoothly than they do in standard metafiles.

The CopyImage method produces both standard metafile and enhanced metafile pictures on the clipboard. In most case an application that is to receive the picture by pasting will choose the most appropriate picture.

The SaveImage method (when used with a wmf file extension) produces a standard metafile image.

Index