TAL Bar Code ActiveX Control Contents:

Introduction

Congratulations! You have purchased the most powerful and most versatile bar code programming tool available for Microsoft Windows. The TAL Bar Code ActiveX Control has all the features necessary to easily add professional quality bar codes to your own Windows applications including database, product packaging, document tracking, Postal bar coding and special purpose bar code labeling applications. Not only is the TAL Bar Code ActiveX Control powerful, it is also extremely easy to use and will work flawlessly with any Windows application that can host ActiveX controls including MS Word, Access, Visual Basic, Excel, PowerPoint, C++, etc.. The "Plus" version of the control can even be used on a web server to generate bar code graphics as GIF, JPEG or PNG files dynamically using CGI/Perl code or Active Server Pages. This makes it the perfect tool for producing browser as well as platform independent Internet based bar code applications.

The TAL Bar Code ActiveX Control supports all commonly used bar code symbologies and it allows complete control over all features of each individual symbology including: precise control over all bar code dimensions, full selection for the foreground, background and text colors, symbol rotation in 90 degree increments and complete font selection for the human readable text above or below a bar code symbol. The TAL Bar Code ActiveX Control is also a data bound control which allows you to easily incorporate it into database reporting and labeling applications. In addition to all of the traditional 1 dimensional bar code symbologies like Code 39, UPC, EAN, Code 128, PostNET, Codabar, Bookland, HIBC and Interleaved 2 of 5, the TAL Bar Code ActiveX Control also supports the newest two dimensional bar codes including PDF417, Aztec Code, Data Matrix and Maxicode, making it the most advanced bar code development tool available on the market today.

Unlike other products that create low quality bitmaps or use fonts to produce a bar code symbol, the TAL Bar Code Control draws itself using high resolution Windows MetaFile (vector) graphics that are completely device independent, fully scalable, and will print to the highest resolution of any printer supported by Windows. No knowledge of the printer resolution is required in advance, the graphics that are produced are extremely precise and require minimal system resources, and all bar codes will display and print lightening fast.

In addition to Windows Metafile vector style graphics, the "Plus" version of the control is also capable of saving bar code images to disk in all major graphic file formats including GIF, JPEG, PNG, TIF, EPS, BMP, WMF, TGA and PCX.

If you are a developer and plan to distribute the control with your application, you will be happy to know that the total uncompressed size of all the files that you need to distribute with your application is less than 500Kb. The compressed size is typically less than 200Kb. In fact, only four small files are required to provide complete support for all of the different types of bar codes. The control was authored using the Microsoft Active Template Library (ATL) therefore there are no runtime dependencies or additional files that need to be included when you distribute the control with your application.

You simply cannot obtain a higher quality bar code in the Windows environment than those produced by the TAL Bar Code ActiveX Control.

See Also

Introduction to Bar Code Technology (pg. 76)

License Agreement:

1. GRANT OF LICENSE
TAL Technologies, Inc. grants you the right to use one copy of the enclosed software program (the SOFTWARE) on a single terminal connected to a single computer (i.e., with a single CPU). You may not network the SOFTWARE or otherwise use it on more than one computer or computer terminal at the same time.

2. LICENSE TO DISTRIBUTE THE TAL BAR CODE ACTIVEX CONTROL.

You have a royalty free right to distribute up to ten thousand (10,000) copies of the unmodified ActiveX control (TalBarCd.OCX) and its supporting dynamic link libraries (.DLL files) with your own application programs provided you adhere to the following terms:

a. You must not distribute the license file TALBC.LIC or the help file TALBC.HLP (modified or unmodified). (The license file is used by the TAL Bar Code ActiveX Control when working in a program development environment and it is not required for use of the control in a compiled application.)

b. Any product that you create using the TAL bar code DLLs must not compete with the TAL Bar Code ActiveX Control in any way; e.g.; Your product must not be a software development tool (.EXE, .DLL, .VBX, .OCX, OLE Object, etc.) intended for distribution to other software developers or system integrators. You may not extend access to the OCX properties, methods or events either directly or indirectly.

c. If you distribute any application that uses any of the TAL Bar Code ActiveX Control or the DLL files, a valid copyright notice must be provided within the user documentation and/or start-up screen of your application that specifies TAL Technologies, Inc. as the legal copyright owner, e.g., "*All bar code technology provided in this product is copyrighted by TAL Technologies, Inc.*"

d. Distribution of the TAL Bar Code ActiveX Control in any quantity exceeding ten thousand units is subject to additional license fees which are due and payable to TAL Technologies, Inc.. A Product Distribution Disclosure form must be filed quarterly for any product developed and distributed in quantities over and above 10,000 units. A disclosure form is available from TAL Technologies, Inc. upon request.

e. You must register your ownership of this product with TAL Technologies, Inc. in order to activate your distribution rights.

3. COPYRIGHT
The SOFTWARE is owned by TAL Technologies, Inc. and is protected by United States copyright laws and treaties. Therefore, you must treat the SOFTWARE like any other copyrighted material (e.g., a book or musical recording) except that you may either (a) make one copy of the SOFTWARE solely for backup or archival purposes, or (b) transfer the SOFTWARE to a single hard disk provided you keep the original solely for backup or archival purposes. You may not copy the users manual accompanying the SOFTWARE.

4. OTHER RESTRICTIONS
You may not rent or lease the SOFTWARE, but you may transfer the SOFTWARE and accompanying written materials on a permanent basis provided you retain no copies and the recipient agrees to the terms of this Agreement. You may not reverse engineer, decompile, or disassemble the SOFTWARE. If SOFTWARE is an update, any transfer must include the update and all prior versions.

5. DISCLAIMER

No Warranty of any kind on the SOFTWARE is expressed or implied.

In no event shall TAL Technologies, Inc. or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profit, business interruption, loss of business information or other pecuniary loss) arising out of the use of or inability to use this product. Should you have any questions concerning this agreement, or wish to contact TAL Technologies, Inc. for any reason please write:

TAL Technologies, Inc.
2101 Brandywine Street
Philadelphia, PA 19130
United States of America

Tel: (215)-496-0222
Fax: (215)-496-0322
E-mail: sales@taltech.com
Website:http://www.taltech.com/

Distributing the TAL Bar Code Control with your application.

When you develop an application that incorporates the TAL Bar Code ActiveX control, you must adhere to the following installation guidelines in order for your application to function correctly.

Files that you can distribute with your compiled application:
The following files should be installed along with your application into the specified locations:

File	Installation Path	Notes:
TALBC.DLL	Windows\System	Required
TALBARCD.OCX	Windows\System or your application's path	Required and must be registered into the user's system registry by your installation program. See Notes below.
TALBCMX.DLL	Windows\System	Required for 2D Maxicode bar codes
TALBCDMX.DLL	Windows\System	Required for 2D Data Matrix bar codes
TALRSS14.DLL	Windows\System	Required for RSS14 bar codes

Note: If you purchased the "Linear bar code only" version of the TAL Bar Code ActiveX Control, the files "TALDMX.DLL", "TALMX.DLL" and "TALRSS14.DLL will not be included with the TAL Bar Code ActiveX Control package. These files are only includes with the "Plus" version of the control.

Most application installation packages including the Visual Basic Application Setup Wizard, InstallShield, etc. will automatically register OCX files that are included as part of an application installation package therefore in most cases you will not have to explicitly register the OCX from your installation program. The Wise Install Builder is one exception to this. With the Wise Install system you must specifically set parameters that tell the Wise installation system to register the OCX. This is done by displaying the Details for the OCX file after you add it to your installation and then choosing the "Self Register OCX…" option.
To register or unregister the OCX manually you can use the REGSVR32.EXE program that is provided with Windows using the following syntax:

REGSVR32 *pathname*\TALBARCD.OCX (to register the OCX)
REGSVR32 /u *pathname*\TALBARCD.OCX (to un-register the OCX)

where *pathname* is the directory path where you have installed the TALBARCD.OCX file.

Files that you may <u>not</u> distribute:
None of the other files provided with the TAL Bar Code ActiveX Control may be distributed with your application including the files: TALBC.HLP, TALBC.LIC or any of the sample applications that are provided with the package. Distribution of any of these files or disclosing their content is a violation of the license agreement for this product.

See Also

License Agreement: (pg. 4)

Distributing the TAL Bar Code Control in a Web Page

The TAL Bar Code ActiveX Control is shipped with a CAB file that you can use for installing the control over the Internet. If you are developing a web page that will contain the TAL Bar Code ActiveX Control, you may copy the included CAB file to your web server and reference it in your HTML code so that when a user visits a page that contains the ActiveX Control, their browser will automatically download and install the control on their system. The included CAB file contains only the files that you are legally allowed to distribute and it has also been digitally signed by TAL Technologies using Microsoft Authenticode technology. The Authenticode digital signature provides your end users with a guarantee that the control is safe to use on their system. The included CAB file will also install a special license file named TALBC.LPK that will allow the properties of the control to be modified using VB or Java Script code in your HTML document.

The following sample object tags show how to insert a TAL Bar Code Control into an HTML document. The first object points to the LPK (license pack) file that must be installed on the users system in their Windows folder. This tag should appear before the object tag for the control and only needs to be included in the HTML code once, even if you have multiple tags referencing the actual control. The TAL Bar Code ActiveX Control will not be scriptable in Internet Explorer without the LPK file installed in the users system.
The LPK file is installed automatically from the Internet CAB file. The second object tag below is for the actual control. The name of the control is set using the "ID=" setting and the "CODEBASE=" setting points to a web URL or a file path where the CAB file for the control can be found. If the control is not installed in the users system, then their browser will automatically download the control from the specified URL and install it. After the control has been installed, it will not be downloaded again if they visit any web page that references the control.

To place a TAL Bar Code ActiveX Control in an HTML document you add the following object tags to your document where you want a bar code to appear.

```
<object WIDTH="5" HEIGHT="5" align=" "
CLASSID="CLSID:5220cb21-c88d-11cf-b347-00aa00a28331">
<param name="LPKPath" value="Talbc.lpk">
</object>

<object ID="TALBarCd1" WIDTH="259" HEIGHT="109" align="center"
CLASSID="CLSID:C917E12F-9757-11D2-85DB-F01851C10000"
CODEBASE="http://www.YourWebSite.com/talbcocx.cab">
<param name="QuietZones" value="-1">
</object>
```

The "param name" tag in the above object tag demonstrates how to pre-set property values for the control when it first appears in the users browser. The code above sets the "QuietZones property of the bar code control to True (-1). To set properties of the TAL Bar Code ActiveX Control after it appears in the users browser, you could use VBScript or JavaScript code similar to the following line:
TALBarCd1.message="1234567890"

Refer to the Internet Explorer HTML Sample document that is provided with the TAL Bar Code ActiveX Control for a more detailed example of how to script the control in a web page.

TAL Bar Code ActiveX Control Properties

AutoSize Property

Returns or sets a value that determines whether a control is automatically resized to display its entire contents.

Syntax

object.AutoSize [= *boolean*]

The AutoSize property syntax has these parts:

Part	Description
object	An object expression that evaluates to TAL Bar Code object.
boolean	A Boolean expression that specifies whether the control is resized, as described in Settings.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Automatically resizes the control to display its entire contents.
False	Keeps the size of the control constant. Contents are clipped when they exceed the area of the control.

AztecPctECC Property

Returns or sets the percentage of total symbol area to use for error checking and correction in normal Aztec Symbols.

Syntax

object.AztecPctECC [= *value*]

The AztecPctECC Property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
value	A numeric expression that evaluates to a number between 0 and 99. (Default = 0)

Remarks

This property is ignored for anything other than Normal Aztec symbols.
See the AztecSymbolType Property for a description of the different Aztec symbol types.
If you specify a value of Zero for the AztecPctECC property, all Aztec bar code symbols will be generated using a default error correction of approximately 23% of the symbol area.

See Also

Aztec Code Symbology Description (pg. 99)

AztecSymbolType Property

Returns or sets the type of Aztec bar code symbol to generate.

Syntax

object.AztecPctECC [= *value*]

The AztecPctECC Property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
value	A numeric expression that evaluates to a number between 0 and 2 as described in the settings below.

Settings

The settings for value are:

Constant	Setting	Description
bcNormal	0	Normal Aztec symbols (Default)
bcFullRange	1	Full Range symbols
bcCompact	2	Compact symbols

Remarks

Normal Aztec symbols are the same as Full Range symbols except for the way that you specify the amount of error checking and correction to use. With Normal symbols, you specify a percentage of error correction overhead through the AztecPctECC property. With Full Range and Compact symbols you specify the total number of data layers to use in the symbol and any left over space, after the message is encoded, is used for error correction.
Full Range symbols may have from 1 to 32 data layers and Compact symbols may have from 1 to 4 data layers. The number of data layers for Full Range and Compact symbols is specified using the AztecTotalLayers property.

See Also

Aztec Code Symbology Description (pg. 99)

AztecTotalLayers Property

Returns or sets the total number of data layers in a Compact of Full Range Aztec bar code symbol.

Syntax

object.AztecTotalLayers [= *value*]

The AztecTotalLayers Property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
value	A numeric expression that evaluates to a number between 1 and 32. (Default = 32)

Remarks

The AztecTotalLayers property is only valid when the AztecSymbolType property is set to produce either Full Range or Compact Aztec symbols. Full Range symbols may have from 1 to 32 data layers and Compact symbols may have from 1 to 4 data layers. Setting the Total Number of data layers is equivalent to setting the exact size of the Aztec Symbol. If you choose a number of data layers that is more than is necessary to encode a particular message, the remaining data layers will be used for error correction data.

See Also

Aztec Code Symbology Description (pg. 99)

BackColor, ForeColor, TextColor Property

BackColor — returns or sets the background color of a bar code.
ForeColor — returns or sets the foreground color of all bars in a bar code object.
TextColor — returns or sets the color of all human readable text in a bar code.

Syntax

object.**BackColor** [= *color*]
object.**ForeColor** [= *color*]
object.**TextColor** [= *color*]

The **BackColor, ForeColor** and **TextColor** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
color	A value or constant that determines the background, foreground or text colors of a TAL Bar Code control object, as an RGB value.

Remarks

The **BackColor** property is ignored if the **BackStyle** property setting is 0 (Transparent).

The default **BackColor** is White (&HFFFFFF) and, the default **ForeColor** and **TextColor** is Black (0). The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower 3 bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Note: It is entirely possible to choose color combinations that render a bar code symbol unreadable. Although two colors may appear to the human eye to have a high contrast between them, a bar code reader may not be able to determine any difference at all between the two colors. Solid black bars on a solid white background always produces the best results. If you must use colors other than black on white, a good rule of thumb is to select solid foreground colors with a luminescence value no greater than 60 and select solid background colors with a luminescence value no less than 180. Because most laser bar code readers use a red laser beam, colors toward the red end of the spectrum should be avoided as the foreground color.

See Also

How To Produce Readable Bar Codes (pg. 83)

BackStyle Property

Returns or sets a value indicating whether the background of a TAL Bar Code control is transparent or opaque.

Syntax

object.**BackStyle** [= *number*]

The **BackStyle** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying transparency, as described in Settings.

Settings

The settings for number are:

Constant	Setting	Description
bcOpaque	0	(Default) Opaque — the control's BackColor property setting fills the control and obscures any color or graphics behind it.
bcTransparent	1	Transparent — background color and any graphics are visible behind the control.

Remarks

A control's **BackColor** property is ignored if **BackStyle** = 1 (Transparent).

Note: Some ActiveX container applications may not be capable of rendering a transparent background correctly therefore you may have to experiment with this property to see if it is supported by the container application that you will be using.

BarHeight Property

Returns or sets a value for the height of the bars in a TAL Bar Code control.

Syntax

object.**BarHeight** [= *number*]

The **BarHeight** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the height of the bars in units of mils (.001 inches)

Remarks

The **BarHeight** property specifies only the height of the bars in all linear bar code symbologies. The default value for the **BarHeight** property is 1000 (1 inch).
The overall height of the bar code will depend on whether the human text is included in the symbol a well as if a comment is included.

See Also

Bar Code Dimensions (pg. 81)

BarWidthReduction Property

Returns or sets a percentage value for the reduction or gain of the width of the bars in a TAL Bar Code control.

Syntax

object.**BarWidthReduction** [= *number*]

The **BarWidthReduction** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying either a positive or a negative percentage to reduce or increase the width of all bars in a bar code. The value for *number* may range between 99 (reduction) and -99 (gain). For example, specifying a value of 50 causes the width of all bars in a bar code to be reduced by 50 percent.

Remarks

The **BarWidthReduction** property allows you to set a Reduction or Gain factor ranging from 99 (% reduction) to -99 (% gain). The default value for the **BarWidthReduction** parameter is zero. Specifying a non-zero value for the **BarWidthReduction** parameter causes the TAL Bar Code control to reduce or enlarge the width of all solid bars in a bar code. Bar Width Reduction is often necessary to compensate for ink spread when generating bar codes that will be used in wet ink printing processes (i.e. printing presses). The percentage that you specify is based on the **NarrowBarWidth** that you choose for your bar codes. For example if you specify a **BarWidthReduction** value of 25 and your **NarrowBarWidth** is set at 10 mils, the width of all bars in your bar codes will be reduced by 2.5 mils (25% of 10 mils = 2.5 mils). Bar width gain is typically used when printing on glass or other surfaces that cause ink to bead up or shrink as it dries. To specify bar width gain instead of reduction, use a negative percentage value. An error event is generated if you specify a **BarWidthReduction** value greater than 99 or less than -99.

The amount of Bar Width Reduction to use depends on a number of factors including the type of paper that the bar codes will be printed on, the type of ink that is being used and the speed and plate pressure of the printing press. Typical values for this property range from 5 to 25 percent however it is recommended that you consult with your printer about the best value to choose.

BearerBars Property

Returns or sets a value that determines whether to include bearer bars around a bar code produced by the TAL Bar Code control.

Syntax

object.**BearerBars** [= *boolean*]

The **BearerBars** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to include bearer bars abound all bar code symbols.

Remarks

The purpose of bearer bars is to equalize the pressure exerted by a printing plate over the entire surface of the symbol. Bearer bars also enhance the reading reliability of a bar code by reduction of the probability of misreads or short scans which may occur when a skewed scanning beam enters or exits the symbol through the top or bottom edge of the bar code. When the scanner path leaves the symbol either through the top or bottom, it crosses the bearer bar, thereby resulting in an invalid start/stop code. Only the symbologies: Code 39, Code 93, Interleaved 2 of 5, CodaBar and Code 128 support bearer bars. This option is ignored by all other symbologies.

BitsPerPixel Property

(ActiveX Plus version only)

Returns or sets a value that determines the number of bits per pixel to use when saving a bar code to a disk file using any of the raster graphic file formats.

Syntax

object.BitsPerPixel [= *value*]

The BitsPerPixel property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
value	A numeric expression that evaluates to the value 1, 8 or 24 as described in Settings.

Settings

The settings for *value* are:

Setting	Description
1	(Default) All raster images are saved to disk using 1 bit per pixel using a color palette with two entries thereby allowing two color images only. If the bar code has three different colors set for the Foreground, Background and Text colors then the Text color will be set equal to the Foreground color. Setting the BitsPerPixel property to 1 will create the smallest image file size.
8	All raster images are saved using 8 bits per pixel using a color palette with 256 entries. If you need to bar codes with different colors for the Foreground, Background and Text colors then you should use 8 or 24 Bits Per Pixel.
24	All raster images are saved to disk using full 24 bit RGB colors with no color palette. This option will create extremely image large files.

Remarks

It is strongly recommended that you generate bar codes using black for the foreground color and white for the background color and also set the BitsPerPixel property to 1. This will result in the best quality image and the smallest image files.

See Also

The RasterImageResolution Property (pg. 61)
The SaveBarCode Method (pg. 71)

BMPCompression Property

(ActiveX Plus version only)

Returns or sets a value that determines whether BMP are files saved to disk using RLE compression.

Syntax

object.BMPCompression [= *boolean*]

The BMPCompression property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
boolean	A Boolean expression that specifies whether the control should use RLE compression for all BMP files.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Enables BMP compression.
False	BMP files saved to disk are not compressed.

Remarks

Enabling BMPCompression can drastically reduce the size of any bitmap files that you save however not all programs that can read BMP files are capable of uncompressing them. You may need to test any bitmap files that you create with whatever application will be using them to ensure that BMP compression is supported.

See Also

The SaveBarCode Method (pg. 71)

CodaBarOptionalCheckDigit Property

Returns or sets a value that determines whether to include an optional check digit with all CodaBar bar codes produced by the TAL Bar Code control.

Syntax

object.**CodaBarOptionalCheckDigit** [= *boolean*]

The **CodaBarOptionalCheckDigit** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to include an optional check digit with all CodaBar bar code symbols.

Remarks

The optional check digit for the CodaBar bar code symbology is rarely used. If the **CodaBarOptionalCheckDigit** property is set to *True*, a check digit will be calculated using the modulo 16 sum of the values of all characters in the CodaBar bar code message. The check digit is then appended to the original message before the bar code is produced.

See Also

CODABAR Symbology Description (pg. 92)

Code39OptionalCheckDigit Property

Returns or sets a value that determines whether to include an optional check digit with all CodaBar bar codes produced by the TAL Bar Code control.

Syntax

object.**Code39OptionalCheckDigit** [= *boolean*]

The **Code39OptionalCheckDigit** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to include an optional check digit with all Code 39 bar code symbols.

Remarks

If the **Code39OptionalCheckDigit** property is set to *True*, a check digit will be calculated using the modulo 43 sum of the values of all characters in the Code 39 bar code message. The check digit is then appended to the original message before the bar code is produced.

See Also

CODE 39 (Normal, Full ASCII and HIBC versions) Symbology Description (pg. 85)

Code39StartStopChars Property

Returns or sets a value that determines whether to display the start and stop characters in the human readable message in all Code 39 bar codes produced by the TAL Bar Code control.

Syntax

object.**Code39StartStopChars** [= *boolean*]

The **Code39StartStopChars** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to display the start and stop characters in the human readable message in all Code 39 bar code symbols.

Remarks

The start and stop characters in a Code 39 bar code symbol are represented by asterisks at the beginning and end of every Code 39 bar code. Start and stop characters are always included in the actual bar code symbol for all Code 39 bar codes however they are not normally displayed in the human readable message that is printed with the bar code. If the **Code39StartStopChars** property is set to *True*, the asterisks are displayed at either end of the human readable message displayed with the bar code symbol.

Note: You do not need to enter the start and stop characters (asterisks) in the **Message** property for Code 39 bar codes because they are automatically included.

See Also

CODE 39 (Normal, Full ASCII and HIBC versions)Symbology Description (pg. 85)

Comment Property

Returns or sets a string for the human readable comment printed with a bar code symbol produced by a TAL Bar Code control.

Syntax

object.**Comment** [= *string*]

The **Comment** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
string	A string expression specifying the comment text. The maximum length of the comment string is 100 characters.

See Also

The CommentAlignment Property (pg. 23)
The CommentOnTop Property (pg. 24)

CommentAlignment Property

Returns or sets a value indicating how to align the human readable comment printed with a bar code symbol produced by a TAL Bar Code control.

Syntax

*object.***CommentAlignment** [= *number*]

The **CommentAlignment** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying how the comment text is aligned, as described in Settings.

Settings

The settings for number are:

Constant	Setting	Description
bcLeftAlign	0	Left Alignment (Default) – Align the comment with the left edge of the bar code symbol.
bcCenterAlign	1	Center Alignment – Align the comment with the center of the bar code symbol.
bcRightAlign	2	Right Alignment – Align the comment with the right edge of the bar code symbol.

See Also

The Comment Property (pg. 22)
The CommentOnTop Property (pg. 24)

CommentOnTop Property

Returns or sets a value that determines whether to display the comment line above all bar codes produced by the TAL Bar Code control.

Syntax

*object.***CommentOnTop** [= *boolean*]

The **CommentOnTop** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to display the human readable comment above all bar code symbols.

Remarks

The default value for the **CommentOnTop** property is *True*. Setting this property to *False* causes the comment to be displayed below the bar code symbol.

See Also

The Comment Property (pg. 22)
The CommentAlignment Property (pg. 23)

DataBindings Property

Returns the DataBindings collection object containing the bindable properties available to the developer.

Syntax

object.DataBindings

The object placeholder represents an object expression that evaluates to a TAL Bar Code control.

Remarks

Data binding allows you to link data from a "data source" (i.e. database table or query, spreadsheet cell, etc.) to either the Message or the Comment properties of the TAL Bar Code ActiveX Control. When data in the data source changes, the bound properties automatically change to reflect the new data.
Not all programs that can host ActiveX controls support data binding therefore this property and the DataField and DataSource properties may not be listed in the Properties dialog box in the application that you are using. Microsoft Visual Basic, Excel and Access all support data binding however each program presents the data binding properties slightly differently and with different names for the DataField and DataSource properties.

See Also

The DataField Property (pg. 26)
The DataSource Property (pg. 31)

DataField Property

Returns or sets a value that binds a TAL Bar Code control to a field in the current record.

Syntax

object.DataField [= *value*]

The DataField property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
value	A string expression that evaluates to the name of one of the fields in the Recordset object specified by a Data control's RecordSource and DatabaseName properties.

Remarks

The DataBindings property will only be available in development environments that support data binding including Microsoft Visual Basic and Microsoft Access. In Access, the DataField property will appear in the properties dialog box as the "Control Source" property and in Excel, it will appear in the properties dialog box as the "Linked Cell" property.

Bound controls provide access to specific data in a database or "data source". Binding a TAL Bar Code control to a field in a database causes the control to automatically generate a new bar code as you change the current record in the database. The bound field supplies the message to be encoded by the TAL Bar Code control. The DataSource property of a bound control specifies a valid Data control name in Visual Basic (or a database table or query in Access), and the DataField property specifies a valid field name in the Recordset object created by the Data control. Together, these properties specify what data appears in the bound control.

Note In Visual Basic, make sure the DataField property setting is valid for each bound control. If you change the setting of a Data control's RecordSource property and then use Refresh, the Recordset identifies the new object. This may invalidate the DataField settings of bound controls and produce a trappable error.

Data Type

String

See Also

The DataSource Property (pg. 31)

DataMatrixECC Property

Returns or sets a value that indicates the Data Matrix Error Checking and Correction (ECC) algorithm to be used by a TAL Bar Code control when generating Data Matrix bar codes.

Syntax

object.**DataMatrixECC** [= *number*]

The **DataMatrixECC** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
value	An integer specifying the type of error correction to use for all Data Matrix bar codes as described in Settings.

Settings

The settings for value are:

Constant	Value	Description
bcECC000	0	ECC000
bcECC040	40	ECC040
bcECC080	80	ECC080
bcECC100	100	ECC100
bcECC140	140	ECC140
bcECC200	200	ECC200 (Default)

Remarks

ECC 200 is the newest and most highly recommended type of error correction. All other ECC options are provided for downward compatibility with older Data Matrix bar code applications.

ECC 000 - Provides no error correction
ECC 050 - Provides error correction for damage of up to 2.8% of the printed symbol.
ECC 080 - Provides error correction for damage of up to 5.5 % of the printed symbol.
ECC 100 - Provides error correction for damage of up to 12.6% of the printed symbol.
ECC 140 - Provides error correction for damage of up to 25% of the printed symbol.
ECC 200 - Uses a Reed Solomon error correction algorithm that will automatically provide a varying degree of error correction for damage ranging from a minimum of roughly 20% to greater than 60% damage depending on the amount of data encoded.
For the five ECC levels ECC 000 - ECC 140 there is also a selectable option for a "Data Format" which defines the type of data that may be encoded in a Data Matrix symbol. This parameter is passed to the TAL Bar Code control in the DataMatrixFormatID property.

See Also

Data Matrix Symbology Description (pg. 100)

DataMatrixFormatID Property

Returns or sets a value that indicates the Data Matrix Format ID to be used by a TAL Bar Code control when generating Data Matrix bar codes that do not use the Data Matrix ECC 200 error correction option.

Syntax

*object.***DataMatrixFormatID** [= *number*]

The **DataMatrixFormatID** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object .
number	An integer specifying the data format ID to use for all Data Matrix bar codes (ECC000 to ECC140) as described in Settings.

Settings

The settings for value are:

Constant	Value	Description
bcNumeric	1	Numeric digits (0-9) only
bcUpper_Alpha	2	Upper case alpha characters only
bcUpperAlphaNumeric	3	Upper case alpha and numeric characters only
bcUpperAlphaNumPunct	4	Upper case alpha, numeric and punctuation characters only
bcSeven_Bit_ASCII	5	All characters with ASCII codes 0 to 127
bcEight_Bit_ASCII	6	(Default) All ASCII characters
bc8x18	7	Create rectangular symbol 8 by 18 modules (ECC200)
bc8x32	8	Create rectangular symbol 8 by 32 modules (ECC200)
bc12x26	9	Create rectangular symbol 12 by 26 modules (ECC200)
bc12x36	10	Create rectangular symbol 12 by 36 modules (ECC200)
bc16x36	11	Create rectangular symbol 16 by 36 modules (ECC200)
bc16x48	12	Create rectangular symbol 16 by 48 modules (ECC200)

Remarks

The values 1 to 6 for the **DataMatrixFormatID** Property applies only to the five Data Matrix ECC levels ECC 000 - ECC 140 and it defines the type of data that may be encoded in a Data Matrix symbol. If you try to encode a character that is not supported by the chosen Format ID, the TAL Bar Code control raises an error event and the bar code symbol is not generated.
When ECC200 is selected, this property is used to set the shape of the resulting bar code by using the values 7 to 12. If the value is less than 7 then square Data Matrix symbols will be produced.
Rectangular symbols are greatly limited in the number of characters that they may contain.

See Also

Data Matrix Symbology Description (pg. 100)

DataMatrixTildeCodes Property

Returns or sets a value that determines whether to translate special “tilde codes” in the **Message** property for all Data Matrix bar codes produced by the TAL Bar Code control.

Syntax

*object.***DataMatrixTildeCodes** [= *boolean*]

The **DataMatrixTildeCodes** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to interpret special tilde codes embedded within a Data Matrix bar code message.

Remarks

Some applications for Data Matrix require bar codes that contain embedded control codes or other special command codes however some application programs have no way to embed these codes in a string. If you set the value of the **DataMatrixTildeCodes** property to True, the TAL Bar Code control will look for special “tilde codes” in the Message property for Data Matrix bar codes. Any tilde codes that are found will be converted to either a control code or to a special command code.

The tilde character (~) is used as a shift character for inserting control codes (characters with ASCII values 0 to 26) into a bar code message. For example, ~@ = NUL, ~A = ASCII 1,~G = BEL (ASCII 7), ~M = ASCII 13 (carriage return). If you need to insert ASCII control codes into a message, take the ASCII value for the control code (1-26) and find the corresponding letter in the alphabet and precede it with a tilde. i.e. The ASCII value for a carriage return character is ASCII 13 and the thirteenth letter of the alphabet is "M" therefore to insert a carriage return in a bar code message, you would use "~M". Note: You can also pass control codes directly to the TAL Bar Code control in the Message property without having to use the ~ before an alpha character. For example you could use either an ASCII 13 character or the sequence ~M to represent a carriage return.

To encode a tilde (~) use the string: ~~ (i.e. two tilde characters). If no tilde characters or Nulls (ASCII 0) are present in the input message, then enabling the "Standard_ASCII " option has no effect on the resulting bar code symbol.

Additional Tilde Command Codes

~1 is used to represent the FNC1 code and is followed by normal data.
To encode data to conform to specific industry standards as authorized by AIM International (Tel: 703-391-7621), a FNC1 character shall appear in the first or second symbol character position (or fifth or sixth data position of the first symbol of structured append). FNC1 encoded in any other position is used as a field separator and shall be transmitted as a GS control character (ASCII value 29). If the FNC1 code is used in the second character position, the input data before '~1' must be, between 'A' and 'Z', or between 'a' and 'z' or 2-digits between '01' and '99'.

~2 : is used to represent Structured Append and must be followed by a three 3-digit number between 1 and 255 representing the symbol sequence as well as a file identifier of six numeric digits. The file identifier is used to uniquely identify a sequence so that only logically linked sequences are processed as part of the same sequence. The symbol sequence identifier is a number between 1 and 255 that indicates the position of the symbol within a sequence of up to 16 symbols. The sequence identifier actually contains two four-bit values representing the sequence number and the total number of symbols in the sequence (i.e. m of n where m is the sequence number and n is the total number of symbols). The upper four bits of this value represent the position of the particular symbol as the binary value of (m-1) and the lower order four bits identify the total number of symbols to be concatenated as the binary value of (17-n). For example, symbol 3 in a sequence of 7 symbols with file ID: 001015 is represented by ~2042001015. The number 042 is derived as follows: 3-1=2 which equals 0010 when represented as a 4 bit binary number. 17-7=10, which equals 1010, when represented as a 4 bit binary number. After concatenating the two 4 bit binary values we end up with 00101010 which equals 42 in decimal.

~3 : Indicates that a message is to be used for reader programming and is followed by normal data. This feature requires knowledge of the programming commands for your bar code reader.

~5 and ~6 : indicates that the data will contain an abbreviated format header and trailer followed by normal data. The ~5 or ~6 must appear as the first two characters in a message and must not be used in conjunction with structured append. Data Matrix provides a means of abbreviating an industry specific header and trailer in one symbol character. This feature exists to reduce the number of characters needed to encode data using certain structured formats. If a ~5 is used as the first two characters of a message, the header []> + ASCII 30 + 05 + ASCII 29 will be transmitted by the reader before the data in the message and the trailer ASCII 30 + ASCII 4 will be transmitted following the data. Likewise, if a ~6 is used as the first two characters of a message, the header []>+ ASCII 30 + 06 + ASCII 29 will be transmitted by the reader before the data in the message and the trailer ASCII 30 + ASCII 4 will be transmitted following the data.

~7NNNNNN is used to indicate Extended Channel NNNNNN where NNNNNN is 6-digit EC value (000000 - 999999). e.g. Extended Channel 9 is represented by ~7000009

~dNNN creates ASCII decimal value NNN for a codeword (must be 3 digits). Please refer to the official Data Matrix symbology specification for details on the meanings of all codeword values for ECC 200. Contact AIM International at Tel: 703-391-7621 or email: adc@aimi.org

See Also

Data Matrix Symbology Description (pg. 100)

DataSource Property

Sets a value that specifies the database table, query or Data control through which the current control is bound to a database. Not available at run time.

Remarks

To bind a control to a field in a database at run time in Visual Basic, you must specify a Data control in the **DataSource** property at design time using the Properties window.
To complete the connection with a field in the Recordset managed by the Data control, you must also provide the name of a Field object in the **DataField** property. Unlike the **DataField** property, the **DataSource** property setting isn't available at run time.

In a Microsoft Access report, the DataSource property is implicitly set when you select a table or a query as the data source of the report and therefore the DataSource property will not appear in the properties dialog box for the control.

Data Type

String

See Also

The DataField Property (pg. 26)

Font Property

Returns a Font object.

Syntax

object.Font

The object placeholder represents an object expression that evaluates to aTAL Bar Code ActiveX control object.

Remarks

Use the Font property to identify a specific Font object whose properties you want to use. For example, the following code changes the Bold property setting of a Font object identified by the Font property of a TAL Bar Code ActiveX Control object:

TALBarCd1.Font.Bold = True

In Visual Basic you cannot create a Font object using code like Dim X As New Font. If you want to create a Font object, you must use the StdFont object as in the following code:

Dim X As New StdFont

If you put a TAL Bar Code control named TALBarCd1 on a form, you can dynamically change its Font object to another using the Set statement, as in the following example:

```
Dim X As New StdFont
X.Bold = True
X.Name = "Arial"
Set TALBarCd1.Font = X
```

FontName Property

Returns or sets the font used to display text in a control or in a run-time drawing or printing operation.

Note The FontName property is included for use with the CommonDialog control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new Font object properties (not available for the CommonDialog control).

Syntax

object.FontName [= *font*]

The FontName property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code object.
font	A string expression specifying the font name to use.

Remarks

The default for this property is determined by the system. Fonts available with Visual Basic vary depending on your system configuration, display devices, and printing devices. Font-related properties can be set only to values for which fonts exist.
In general, you should change FontName before setting size and style attributes with the FontSize, FontBold, FontItalic, FontStrikethru, and FontUnderline properties.

Different fonts behave differently thus some fonts may appear different on screen than when printed. True Type fonts are the most WYSIWYG and they also align better when rotated.

Note: Most bar code symbology specifications recommend the font OCR-B (Optical Character Recognition revision B). The choice of font is not critical however it is a good idea to choose fonts that are close to the recommended specification. The System font and the MS Sans Serif font are both very close to OCR-B as is the True Type Arial font.

FontSize Property

Returns or sets the size of the font to be used for text displayed in a control.

Note The FontSize property is included for use with the CommonDialog control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new Font object properties.

Syntax

object.FontSize [= *points*]

The FontSize property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
points	A numeric expression specifying the font size to use, in points, for all human readable text in a bar code symbol.

Remarks

Use this property to format text in the font size you want. The default is determined by the system. To change the default, specify the size of the font in points.
The maximum value for FontSize is 2160 points.

Note Fonts available with Visual Basic vary depending on your system configuration, display devices, and printing devices. Font-related properties can be set only to values for which fonts exist.
In general, you should change the FontName property before you set size and style attributes with the FontSize, FontBold, FontItalic, FontStrikethru, and FontUnderline properties. However, when you set TrueType fonts to smaller than 8 points, you should set the point size with the FontSize property, then set the FontName property, and then set the size again with the FontSize property. The Microsoft Windows operating environment uses a different font for TrueType fonts that are smaller than 8 points.

FontBold, FontItalic, FontStrikethru, FontUnderline Properties

Return or set font styles in the following formats: Bold, Italic, Strikethru, and Underline.
The font styles are applied to all human readable text in a bar code symbol including both the message and the comment text.

Note: The FontBold, FontItalic, FontStrikethru, and FontUnderline properties are included for use with the CommonDialog control and for compatibility with earlier versions of Visual Basic. For additional functionality, use the new Font object properties

Syntax

object.FontBold [= *boolean*]
object.FontItalic [= *boolean*]
object.FontStrikethru [= *boolean*]
object.FontUnderline [= *boolean*]

The FontBold, FontItalic, FontStrikethru, and FontUnderline property syntax's have these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code object.
boolean	A Boolean expression specifying the font style as described in Settings.

Settings

The settings for boolean are:

Setting	Description
True	Turns on the formatting in that style.
False	(Default for FontBold, FontItalic, FontStrikethru, and FontUnderline) Turns off the formatting in that style.

GIFCompression Property

(ActiveX Plus version only)

Returns or sets a value that determines whether GIF files are saved to using LZW compression. Note: Using GIF compression requires that you purchase a LZW license from Unisys Corporation. For details, see Remarks below.

Syntax

object.GIFCompression [= *boolean*]

The GIFCompression property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
boolean	A Boolean expression that specifies whether the control should use LZW compression for all GIF files saved to disk.

Settings

The settings for *boolean* are:

Setting	Description
True	Enables GIF compression if a valid LZW unlock key is set in the LZWUnlockKey property.
False	(Default) GIF files saved to disk are not compressed.

Remarks

The TAL Bar Code ActiveX control supports GIF compression however it requires that you purchase a license to use LZW compression from Unisys Corporation. The Unisys license fee ranges in price between $5000 and $7500 and must be obtained directly from Unisys. If you purchase a LZW license from Unisys, you can send a copy of your license to TAL Technologies, Inc. and we will provide you with a key that you can use in the LZWUnlockKey property to unlock the LZW compression capabilities of the control. For information concerning licensing the LZW compression and/or decompression capability, please contact: Unisys Corporation, Welch Licensing Department - C1SW19, Township Line & Union Meeting Roads, P O Box 500, Blue Bell PA 19424. Web Site: http://www.unisys.com/unisys/lzw/

Enabling GIFCompression can drastically reduce the size of any GIF files that you create. Most programs that can read GIF files, including all web browsers, are capable of uncompressing compressed GIF files. The most valuable use of GIF compression is for developing web server applications where download size of your bar code images is critical. A typical bar code saved as a uncomompressed GIF with one pit per pixel image at 300 pixels per inch will be roughly 150 KB in size. The same image saved as a compressed GIF will be 17KB in size, almost a 90 percent savings. Note: The same image saved as a PNG file will be only 916 bytes in size.

Note

Because of the high license fee imposed by Unisys on the LZW license, the Internet community has developed the PNG file format that supports compression that is as good as or better than the LZW compression used in GIF files. The PNG file format is in the public domain and therefore does not require any license fees. Most commonly used web browsers including Internet Explorer and Netscape support the new PNG file format therefore the PNG format is an excellent alternative to using compressed GIF files. The JPEG file format is also a public domain format that is supported by all web browsers. A major disadvantage to using JPEG files is that the compression techniques in them are optimized for complex graphic images and do not work well with simple graphics like bar codes. JPEG compression is also "lossy" meaning that you can lose image quality in compressed JPEG images. Uncompressed JPEG files are roughly the same size as uncompressed GIF files therefore if you want to reduce the size of a JPEG file by using compression, you will probably end up with distorted or even unreadable bar codes.

See Also

The LZWUnlockKey Property (pg. 40)
The JPEGQuality Property (pg. 39)
The SaveBarCode Method (pg. 71)

I2of5OptionalCheckDigit Property

Returns or sets a value that determines whether to include an optional check digit with all CodaBar bar codes produced by the TAL Bar Code control.

Syntax

*object.***I2of5OptionalCheckDigit** [= *boolean*]

The **I2of5OptionalCheckDigit** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to include an optional check digit with all Interleaved 2 of 5 (ITF) bar code symbols.

Remarks

If the **I2of5OptionalCheckDigit** property is set to *True*, a check digit will be calculated using the modulo 10 sum of the values of all characters in the Code 39 bar code message. The check digit is then appended to the original message before the bar code is produced.

See Also

INTERLEAVED 2 OF 5 (ITF) Symbology Description (pg. 92)

JPEGQuality Property

(ActiveX Plus version only)

Returns or sets a value that determines the quality and level of compression to use when saving JPEG images to a disk file.

Syntax

object.JPEGQuality [= *value*]

The JPEGQuality property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
value	An expression that evaluates to a number ranging between 1 and 100 (Default=100).

Remarks

JPEG images use a "lossy" compression scheme. As you apply higher levels of compression to an image, a higher amount of image resolution is lost. The JPEGQuality property actualy determines the amount of lossy compression that will be applied to an image. This number ranges from 1 to 100 where 100 means that no compression should be applied and therefore no image information is lost and decreasing values below 100 add increasing amounts of compression and subsequently additional loss of resolution to an image.
We strongly recommend that you use the default value of 100 for all bar code saved as JPEG files otherwise your bar codes could lose enough resolution information to render them unreadable by most bar code readers. If you require more highly compressed JPEG images then you should experiment with this setting and test all bar codes that you produce to make sure that they are readable by your bar code scanning equipment.

See Also

The SaveBarCode Method (pg. 71)

LZWUnlockKey Property

(ActiveX Plus version only)

Returns or sets a value that is used to unlock the LZW compression capabilities for GIF and TIFF files saved to a disk file.

Syntax

object.LZWUnlockKey [= *value*]

The LZWUnlockKey property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
value	An expression that evaluates to a long integer.

Remarks

The TAL Bar Code ActiveX control supports LZW compression when saving GIF and TIFF images to a disk file however it requires that you purchase a license to use LZW compression from Unisys Corporation. The Unisys license fee ranges in price between $5000 and $7500 and must be obtained directly from Unisys. If you purchase a LZW license from Unisys, you can send a copy of your license to TAL Technologies, Inc. and we will provide you with a key that you can use in the LZWUnlockKey property to unlock the LZW compression capabilities of the control. For information concerning licensing the LZW compression and/or decompression capability, please contact: Unisys Corporation, Welch Licensing Department - C1SW19, Township Line & Union Meeting Roads, P O Box 500, Blue Bell PA 19424. Web Site: http://www.unisys.com/unisys/lzw/

See Also

The GIFCompression Property (pg. 36)
The TiffCompression Property (pg. 67)
The SaveBarCode Method (pg. 71)

MatrixModuleSize Property

Returns or sets a value for the height and width of the modules in all matrix style (Aztec Code and Data Matrix) bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***MatrixModuleSize** [= *number*]

The **MatrixModuleSize** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the height and width in mils (.001 inches) of the modules in all matrix style (Aztec Code and Data Matrix) bar code symbols. The value for *number* may range between 1 and 100. (default = 20)

Remarks

Two dimensional matrix style symbologies are designed around an array of square dots or “modules”. The MatrixModuleSize property sets the size of these modules and therefore determines the overall height and with of a matrix style bar code symbol. Most matrix style bar code symbologies specify that you select a module size between 10 and 60 mils with the preferred size of 20 mils.

MaxiCodeClass Property

Returns or sets a value for the service class when generating structured carrier messages to be encoded in a MaxiCode bar code symbol produced by a TAL Bar Code control.

Syntax

*object.***MaxiCodeClass** [= *number*]

The **MaxiCodeClass** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the service class when generating structured carrier messages to be encoded in a MaxiCode bar code symbol

Remarks

The Class is used in Mode 2 and Mode 3 MaxiCode symbols to identify the service class for a structured carrier message. This parameter is encoded in the Primary message in the symbol.

See Also

MaxiCode Symbology Description (pg. 104)

MaxiCodeMode Property

Returns or sets a value for the Mode for all Maxicode bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***MaxiCodeMode** [= *number*]

The **MaxiCodeMode** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the Mode for all Maxicode bar code symbols, as described in Settings. (default = 2)

Settings

The settings for value are:

Constant	Value	Description
bcUS_Carrier	2	Structured carrier mode for shipments within the US (default)
bcIntl_Carrier	3	Structured carrier mode for shipments outside the US
bcStandard_Symbol	4	Standard MaxiCode mode for general purpose (i.e. not for shipping applications)
bcReader_Program	6	Reader programming mode

Remarks

The current MaxiCode symbology specification supports 5 modes numbered 2 through 6.
Modes 2 and 3 are reserved for structured carrier messages for use by carriers in the transportation industry. Mode 4 is designed for use as a "Standard Bar Code" where the ZipCode, Country Code and the Service Class parameters are not used and only the data in the Message property is encoded in the bar code symbol. Mode 5 is similar to Mode 4 except that Mode 5 uses a higher level of error correction. The TAL Bar Code control does not support Mode 5 in the current release of the control. Mode 6 is reserved for "Reader Programming" purposes and it is up to the bar code reader manufacturer to determine how to interpret Mode 6 messages.

See Also

MaxiCode Symbology Description (pg. 104)

MaxiCountryCode Property

Returns or sets a value representing the country used in Mode 2 or Mode 3 MaxiCode bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***MaxiCountryCode** [= *number*]

The **MaxiCountryCode** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the country code for a MaxiCode bar code symbol.

Remarks

The Country Code is used in Mode 2 and Mode 3 symbols to identify the destination country for a structured carrier message. This parameter is encoded in the Primary message in the symbol.

See Also

MaxiCode Symbology Description (pg. 104)

MaxiSymbolNum and MaxiNumSymbols Properties

Returns or sets a value for the “Symbol Number” and the “Total Number of Symbols” parameters in a MaxiCode bar code produced by a TAL Bar Code control.

Syntax

*object.***MaxiNumSymbols** [= *number*]
*object.***MaxiSymbolNum** [= *number*]

The **MaxiNumSymbols** and **MaxiSymbolNum** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression ranging from 1 to 8. (default = 1)

Remarks

The MaxiCode symbology supports a feature called "Structured Append" that allows a long message to be encoded in up to eight symbols. Each symbol in a Structured Append set contains an indicator that specifies the total number of symbols in the set as well as the particular position in the sequence for each given symbol. The **MaxiSymbolNum** property indicates the position of the current symbol in the sequence and the **MaxiNumSymbols** property specifies the total number of symbols in the sequence. The values for these parameters may range from 1 to 8. For a single symbol that is not part of a structured append sequence, the value 1 should be supplied for both the **MaxiSymbolNum** and **MaxiNumSymbols** properties.

See Also

MaxiCode Symbology Description (pg. 104)

MaxiZipCode Property

Returns or sets a value for the height and width of the modules in all matrix style (Aztec Code and Data Matrix) bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***MaxiZipCode** [= *string*]

The **MaxiZipCode** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
string	A string expression specifying the postal code to be encoded in a MaxiCode mode 2 or mode 3 bar code symbol.

Remarks

When generating Mode 2 or Mode 3 MaxiCode symbols, a postal code must be supplied that will be encoded in the primary message within the symbol. Mode 2 should be used when generating a structured carrier message for shipments within the USA therefore the **MaxiZipCode** property must consist of either a 5 or 9 digit postal ZIP Code (i.e. 5 or 9 numeric digits).
Mode 3 should be used when generating a structured carrier message for shipments outside the United States. In Mode 3, the **MaxiZipCode** property must consist of a valid postal code containing up to 6 AlphaNumeric characters. The **MaxiZipCode** property is encoded in the Primary message in the symbol.

See Also

MaxiCode Symbology Description (pg. 104)

Message Property

Returns or sets a string for the message to be encoded in a bar code symbol produced by a TAL Bar Code control.

Syntax

*object.***Message** [= *string*]

The **Message** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
string	A string expression specifying the message text.

Remarks

Different symbologies allow different sets of characters to be encoded. For example UPC, EAN, PostNET and Interleaved 2 of 5 can only encode numeric digits (0-9) and CodaBar can only encode numeric digits and the alpha characters A,B,C and D. If you pass a message that contains illegal characters for a particular symbology, the TAL Bar Code control will raise a "BarCodeError" event and no bar code will be generated.

Notes:

When specifying UPC A, UPC E, EAN 8, EAN 13 and BookLand bar code messages, to include a 2 or 5 digit supplemental message, append the 2 or 5 digit supplemental message to the main message with a comma between them.

When specifying RSS14 bar code messages, to include a composite message, append the composite message to the main message with a comma between them.

See Also

Bar Code Symbology Descriptions and Rules (pg. 84-104)

NarrowBarWidth Property

Returns or sets a value for the width of the bars in all linear bar code symbols produced by a TAL Bar Code control.

Syntax

object.**NarrowBarWidth** [= *number*]

The **NarrowBarWidth** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	(default = 13) A numeric expression specifying the width in mils (.001 inches) of the narrowest bar or space in a bar code. The value for *number* may range between 1 and 100.

Remarks

The **NarrowBarWidth** (expressed in integer units of .mils) specifies the width of the narrowest bar in the bar code. All other bar and space width dimensions are based on this width (referred to as the nominal X dimension). This parameter as well as the number of characters to encode, effectively determines the total width of a bar code symbol. The best choice for this dimension depends partly on the resolution of your bar code reader and also on the resolution of the printer being used to produce the bar code.
As a general rule the Narrow Bar Width should fall in a range between 10 to 30 mils and should never be less than 7.5 mils. 13 mils is the most commonly recommended value for most bar code readers. For UPC and EAN bar codes, the smallest allowable Narrow bar width is 10.4 mils.
The allowable range of values for **NarrowBarWidth** property in the TAL Bar Code control is 0 to 100. If you pass the value zero, the default value of 13 mils will be used.

See Also

The NarrowToWideRatio Property (pg. 49)
Bar Code Dimensions (pg. 81)

NarrowToWideRatio Property

Returns or sets a value for the width of the bars in all linear bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***NarrowToWideRatio** [= *number*]

The **NarrowToWideRatio** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	(default = 2.5) A numeric expression specifying the ratio between the width of the wide and narrow bars in a bar code. The value for *number* may range between 2 and 3.

Remarks
The symbologies Code 39, Interleaved 2 of 5 and CodaBar consist of bars and spaces with only two element widths, Narrow and Wide Elements where the width of the wide elements is a fixed multiple of the width of the narrow elements. The specifications for these symbologies allow you to choose a Narrow to Wide Element Ratio ranging from 2.0 to 3.0.

This property is valid only for Code 39, Interleaved 2 of 5 and CodaBar and all other symbologies will ignore it. The rules for these symbologies specify that when the Narrow Bar Width is less than 20 mils, the Narrow To Wide element ratio must be 2.2 or greater. The default **NarrowToWideRatio** of 2.5 should be acceptable for most applications.

Note: Higher quality readers may be able to read bar codes with a narrow to wide ratio less than 2.2 no matter what the narrow element width is. Lower quality readers often need a ratio of at least 2.5. Because of the variability between readers, you should always test different ratio values and select the value that produces bar codes with the best "first pass" read rate.

See Also

The NarrowBarWidth Property (pg. 48)
Bar Code Dimensions (pg. 81)

PDFAspectRatio Property

Returns or sets a value for the overall height to width ratio of all PDF417 bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***PDFAspectRatio** [= *number*]

The **PDFAspectRatio** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the ratio between the overall height to width of all PDF417 bar codes. The value for *number* may range between .01 and 1000. (default = .5)

Remarks

The PDFAspectRatio determines the overall shape of the PDF417 symbol and is defined as the overall height to width ratio. Higher values for the Aspect Ratio (greater than 1) produce tall, thin PDF417 bar codes and small values (greater than zero and less than 1) produce short, wide bar codes. A value of 1 should produce approximately square bar codes.

See Also

PDF417 Symbology Description (pg. 95)

PDFMaxCols Property

Returns or sets a value for the maximum number of codeword columns to allow in all PDF417 bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***PDFMaxCols** [= *number*]

The **PDFMaxCols** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the maximum number of codeword columns to allow in all PDF417 bar code symbols. The value for number may range between 1 and 30. (default = 30)

Remarks

The **PDFMaxCols** and the **PDFMaxRows** properties allow you to set an upper limit on the width or the height of a PDF417 bar code symbol by limiting the maximum number of codeword rows or columns in the symbol.
The **PDFMaxCols** property specifies the maximum number of codeword columns in a PDF symbol. The allowable range is 1 to 30. If you specify a value outside the allowable range then the default value of 30 will be used.

See Also

The PDFMaxRows Property (pg. 52)
PDF417 Symbology Description (pg. 95)

PDFMaxRows Property

Returns or sets a value for the maximum number of codeword rows to allow in all PDF417 bar code symbols produced by a TAL Bar Code control.

Syntax

*object.***PDFMaxRows** [= *number*]

The **PDFMaxRows** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the maximum number of codeword rows to allow in all PDF417 bar code symbols. The value for number may range between 3 and 90. (default = 90)

Remarks

The **PDFMaxCols** and the **PDFMaxRows** properties allow you to set an upper limit on the width or the height of a PDF417 bar code symbol by limiting the maximum number of codeword rows or columns in the symbol.
The **PDFMaxRows** property specifies the maximum number of codeword rows in a PDF symbol. The allowable range is 3 to 90. If you specify a value outside the allowable range then the default value of 90 will be used.

See Also

The PDFMaxCols Property (pg. 51)
PDF417 Symbology Description (pg. 95)

PDFModuleHeight Property

Returns or sets a value for the height of the smallest modules in all PDF417 bar code symbols produced by a TAL Bar Code control.

Syntax

object.**PDFModuleHeight** [= *number*]

The **PDFModuleHeight** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the height in mils (.001 inches) of the smallest modules in a PDF417 bar code. The value for *number* may range between 1 and 100. (default = 30)

Remarks

The recommended value for the Module Height is approximately three times the value for the **PDFModuleWidth** property however the symbol specifications allow for module heights as small as 10 mils (.25mm). This translates to 3 printer dots on a 300 DPI laser printer.

See Also

The PDFModuleWidth Property (pg. 54)
The PDFTruncatedSymbol Property (pg. 57)
PDF417 Symbology Description (pg. 95)

PDFModuleWidth Property

Returns or sets a value for the width of the smallest modules in all PDF417 bar code symbols produced by a TAL Bar Code control.

Syntax

object.**PDFModuleWidth** [= *number*]

The **PDFModuleWidth** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the width in mils (.001 inches) of the smallest modules in a PDF417 bar code. The value for *number* may range between 1 and 100. (default = 10)

Remarks

The specification for PDF417 recommends that the **PDFModuleWidth** should fall in a range between 10 and 30 mils (.25mm to .76mm). The smallest allowable module width defined in the symbology specification is 6.56 mils (.17mm). This translates to 2 printer dots when printing to a 300 DPI laser printer. The best way to determine the ideal **PDFModuleWidth** for your application is to actually print out a sample bar code using several different values and try reading each one with your scanning equipment. You should choose the value that produces bar codes with the best read rate.

See Also

The PDFModuleHeight Property (pg. 53)
The PDFTruncatedSymbol Property (pg. 57)
PDF417 Symbology Description (pg. 95)

PDFPctOverhead Property

Returns or sets a value for the percentage of error correction overhead to use in all PDF417 bar code symbols produced by a TAL Bar Code control.

Syntax

object.**PDFPctOverhead** [= *number*]

The **PDFPctOverhead** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the percentage of total symbol area to use for error correction codewords. The value for *number* may range between 0 and 99. (default = 10)

Remarks

The **PDFPctOverhead** property is valid only when the **PDFSecurityLevel** property is set to 9 (automatic). If you enter zero for this property, the default value of 10% will be used.

See Also

The PDFSecurityLevel Property (pg. 56)
The PDFTruncatedSymbol Property (pg. 57)
PDF417 Symbology Description (pg. 95)

PDFSecurityLevel Property

Returns or sets a value for security level or Error Checking and Correction level (ECC) to use in all PDF417 bar code symbols produced by a TAL Bar Code control.

Syntax

object.**PDFSecurityLevel** [= *number*]

The **PDFSecurityLevel** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
number	A numeric expression specifying the error correction security level to use for PDF417 bar codes. The value for *number* may range between 0 and 9. The value 9 indicates to use automatic selection of the security level based on a percentage of the total symbol area as specified in the **PDFPctOverhead** property. (default = 9)

Remarks

The PDFSecurityLevel property allows you to select a PDF417 error correction level from 0 to 8 (or 9 for automatic). Each higher security level up to 8 adds additional overhead to a PDF417 symbol thereby consuming more symbol real estate. You can have the TAL Bar Code control automatically select an error correction level based on a percentage of total symbol area that you want to devote to error correction. If you select the value 9 for the **PDFSecurityLevel** property and also pass a percentage value (from 0 to 99%) in the **PDFPctOverhead** property, the control will automatically choose a value that will limit the amount of error correction overhead to the given percentage of symbol area. This mechanism is designed so that you do not waste space on redundant error correction.

See Also

The PDFPctOverhead Property (pg. 55)
The PDFTruncatedSymbol Property (pg. 57)
PDF417 Symbology Description (pg. 95)

PDFTruncatedSymbol Property

Returns or sets a value that determines whether to generate the truncated version of all PDF417 bar codes produced by the TAL Bar Code control.

Syntax

*object.***PDFTruncatedSymbol** [= *boolean*]

The **PDFTruncatedSymbol** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to generate the truncated version of all PDF417 bar codes instead of the standard PDF417 symbols.

See Also

The PDFSecurityLevel Property (pg. 56)
The PDFPctOverhead Property (pg. 55)
PDF417 Symbology Description (pg. 95)

PNGFilter Property

(ActiveX Plus version only)

Returns or sets the type of compression filter to use when saving PNG files to disk.

Syntax

object.PNGFilter [= *value*]

The PNGFilter Property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
value	A numeric expression that evaluates to one of the values as described in the settings below.

Settings

The settings for value are:

Constant	Setting	Description
bcPNGAllFilters	0	Use best filter for each row producing the best compression. (Default)
bcPNGNoFilter	2	No filter is used (fastest to execute)
bcPNGSubFilter	4	Difference filter with adjacent pixel
bcPNGUpFilter	6	Difference filter with pixel in previous row
bcPNGAvgFilter	8	Average filter
bcPNGPaethFilter	10	Paeth filter

Remarks

PNG files are highly compressed images that can use filters that determines how the compression algorithm works. The filters can affect how much compression is actually achieved. For bar codes, the type of filter will make little difference in the size of the file that is created. For this reason it is recommended that you use the default value of 0 for this property unless you will be using your bar code images in another application that may require a different filter type or does not support PNG image filters at all.

See Also

The PNGInterlace Property (pg. 59)
The SaveBarCode Method (pg. 71)

PNGInterlace Property

(ActiveX Plus version only)

Returns or sets a value that determines whether PNG files saved to disk are rendered using interlacing.

Syntax

object.PNGInterlace [= *boolean*]

The AutoSize property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
boolean	A Boolean expression that specifies whether the control should use interlacing for PNG files.

Settings

The settings for *boolean* are:

Setting	Description
True	Enables interlacing of PNG images.
False	(Default) PNG files saved to disk are not interlaced.

Remarks

Enabling PNG interlacing causes all PNG files saved to disk to be interlaced when displayed typically in a web browser. Interlacing allows the browser to start displaying the image almost immediately in an interlaced manner so that the end user sees an image gradually filling in and increasing in quality. For small PNG bar code images, most browsers will be able to render the image almost immediately with or without this option enabled. It is typically only useful if you generate large complex graphic images that may take a few seconds to render.

See Also

The SaveBarCode Method (pg. 71)

QuietZones Property

Returns or sets a value that determines whether to include quiet zones at either end of all linear bar code symbols produced by the TAL Bar Code control.

Syntax

object.**QuietZones** [= *boolean*]

The **QuietZones** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to include quiet zones at either end of all linear bar code symbols. (default = False)

Remarks

Quiet Zones are small areas of blank space at either end of a bar code image. This space helps to insure that a bar code reader will be able to correctly determine the true beginning and end of a bar code symbol. The width of the quiet zones will be 10 times the **NarrowBarWidth** value for all 1 dimensional symbols and 2 times the **PDFModuleWidth** value for PDF417 bar codes. Most bar code symbology specifications require quiet zones therefore it is highly recommended that you enable this option. Note: UPC, EAN and BookLand bar codes automatically include quiet zones in the symbol. Selecting this option causes the width of the quiet zones in these symbols to be twice the normal width.

RasterImageResolution Property

(ActiveX Plus version only)

Returns or sets a value that determines the dot resolution (in pixels per inch) to use when saving a bar code to a disk file using any of the raster graphic file formats (GIF, JPEG (JPG), PNG, TIF, TGA and BMP).

Syntax

object.RasterImageResolution [= *value*]

The RasterImageResolution property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
value	A numeric expression that evaluates to the value from 20 to 2540. (Default = 300 pixels per inch)

Remarks

The RasterImageResolution property dramatically affects the quality and size of the images that are produced when you save your bar codes to a disk file. Increasing the value higher than the resolution of the printer that you will be using will provide no additional increase in the quality of the printed output and would only serve to slow down the execution speed of the file saving operation as well as increase the size of your graphic files. On the other hand, if you set this value too low, then the resolution of the saved image will not be high enough to render readable bar codes. The ideal value for this property is the resolution of your printer or an integer divisor of your printer resolution. For example, if you have a 600 DPI printer then setting the RasterImageResolution property to either 300 or 600 would produce extremely high quality bar codes. If you set the RasterImageResolution property to a value that is too low then there will not be enough pixels in the image to render the bars and spaces in the bar code accurately and you will end up with unreadable bar codes. The default value of 300 is a good overall choice that will produce sufficiently high resolution images that save quickly and produce reasonable file sizes.

See Also

The BitsPerPixel Property (pg. 17)
The SaveBarCode Method (pg. 71)

Rotation Property

Returns or sets a value indicating how much to rotate a bar code symbol produced by the TAL Bar Code control.

Syntax

*object.***Rotation** [= *value*]

The **Rotation** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
value	An integer specifying the amount of rotation, as described in Settings.

Settings

The settings for value are:

Constant	Value	Description
bcZeroDegrees	0	(Default) Zero degrees – i.e. no rotation.
bcClockwise_90	1	Symbol is rotated 90 degrees clockwise
bcClockwise_180	2	Symbol is rotated 180 degrees clockwise
bcClockwise_270	3	Symbol is rotated 270 degrees clockwise

ShowCheckDigit Property

Returns or sets a value that determines whether to include optional check digits in the human readable portion of all bar code symbols produced by the TAL Bar Code ActiveX control.

Syntax

*object.***ShowCheckDigit** [= *boolean*]

The **ShowCheckDigit** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to include optional check digits in the human readable portion of all bar code symbols.

Remarks

Normally any optional check digits are not included in the human readable portion of a bar code symbol however there are some applications where you may want to display the check digit. This option is only valid for symbologies that support an optional check digit.
Note: The optional check digit for EAN/UCC 128 will be displayed with the human readable text regardless of whether this option is selected or not.

ShowHRText Property

Returns or sets a value that determines whether to print the human readable text version of a bar code message with all linear bar codes produced by the TAL Bar Code control.

Syntax

*object.***ShowHRText** [= *boolean*]

The **ShowHRText** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to print the human readable text version of a bar code message with all linear bar code symbols. (default = True)

See Also

The Comment Property (pg. 22)
The Message Property (pg. 47)

Symbology Property

Returns or sets a value indicating the type of bar code symbol (symbology) to be generated by a TAL Bar Code control object.

Syntax

*object.***Symbology** [= *value*]

The **Symbology** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object.
value	An integer specifying the type of bar code to generate, as described in Settings.

Settings

The settings for value are:

Constant	Value	Description
bcCode39	0	(Default) Normal Code 39
bcCode39_Full_ASCII	1	Full ASCII Code 39
bcCode39_HIBC	2	HIBC Code 39 (Health Industry Bar Code)
bcCodaBar	3	CodaBar
bcCode93	4	Code 93
bcCode128	5	Code 128
bcUCC_EAN_128	6	UCC/EAN 128
bcInterleaved_2of5	7	Interleaved 2 of 5 (ITF)
bcPostNET	8	PostNET
bcUPC_A	9	UPC-A
bcUPC_E	10	UPC-E
bcEAN_JAN_8	11	EAN/JAN-8
bcEAN_JAN_13	12	EAN/JAN-13
bcBookLand	13	BookLand
bcMSI_Plessey	14	MSI/Plessey
bcPDF417	15	PDF-417 (2-Dimensional Symbology)
bcAztec	16	Aztec Code (2-Dimensional Symbology)
bcDataMatrix	17	Data Matrix (2-Dimensional Symbology)
bcMaxiCode	18	MaxiCode (2-Dimensional Symbology)
bcRSS14	19	RSS14

See Also

Bar Code Symbology Descriptions and Rules (pg. 84-104)

TextOnTop Property

Returns or sets a value that determines whether to print the human readable text above all bar codes produced by the TAL Bar Code control.

Syntax

*object.***TextOnTop** [= *boolean*]

The **TextOnTop** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to print the human readable text above all bar code symbols. If the value is False (Default) then the text is printed below the bar code symbol.

Remarks

Normally the human readable message text is placed below the bar code symbol for all linear bar code symbologies. The two dimensional bar code symbologies (PDF417, Data Matrix, Aztec Code and MaxiCode) do not support the display of the human readable message with the bar code symbol.

See Also

The ShowHRText Property (pg. 64)
The CommentOnTop Property (pg. 24)

TiffCompression Property

(ActiveX Plus version only)

Returns or sets the type of compression to use when saving TIFF graphic files to disk.

Syntax

object.TIFFCompression [= *value*]

The TIFFCompression Property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
value	A numeric expression that evaluates to one of the values as described in the settings below.

Settings

The settings for value are:

Constant	Setting	Description
bcTIFFCompNone	0	Use no compression. (Default)
bcTFFCompLZW	1	Use LZW compression. Requires a LZW license from Unisys. See Remarks
bcTIFFCompPackBits	2	Use PackBits compression.
bcTIFFCompGroup3	3	Use Group 3 compression
bcTIFFCompGroup3	4	Use Group 4 compression

Remarks

The TAL Bar Code ActiveX control supports LZW compression however it requires that you purchase a LZW license from Unisys Corporation. If you purchase a LZW license, you can send a copy of your license to TAL Technologies, Inc. and we will provide you with a key that you can use in the LZWUnlockKey property to unlock the LZW compression capabilities of the control. For more information, please contact: Unisys Corporation, Welch Licensing Department - C1SW19, Township Line & Union Meeting Roads, P O Box 500, Blue Bell PA 19424. Web Site: http://www.unisys.com/unisys/lzw/

See Also

The LZWUnlockKey Property (pg. 40)
The GIFCompression Property (pg. 36)
The SaveBarCode Method (pg. 71)

TGACompression Property

(ActiveX Plus version only)

Returns or sets a value that determines whether TGA are files saved to disk using RLE compression.

Syntax

object.TGACompression [= *boolean*]

The BMPCompression property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL Bar Code ActiveX control object.
boolean	A Boolean expression that specifies whether the control should use RLE compression for all TGA files.

Settings

The settings for *boolean* are:

Setting	Description
True	(Default) Enables TGA compression.
False	TGA files saved to disk are not compressed.

Remarks

Enabling TGACompression can drastically reduce the size of any TGA files that you save however not all programs that can read TGA files are capable of uncompressing them. You may need to test any TGA files that you create with whatever application will be using them to ensure that TGA compression is supported.

See Also

The SaveBarCode Method (pg. 71)

UccEanOptionalCheckDigit Property

Returns or sets a value that determines whether to include an optional check digit with all UCC/EAN 128 bar codes produced by the TAL Bar Code control.

Syntax

*object.***UccEanOptionalCheckDigit** [= *boolean*]

The **UccEanOptionalCheckDigit** property syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
boolean	A boolean expression specifying whether to include an optional check digit with all UCC/EAN 128 bar code symbols.

Remarks

If the **UccEanOptionalCheckDigit** property is set to *True*, a check digit will be calculated using the modulo 10 sum of the values of all characters in the UCC/EAN 128 bar code message. The check digit is then appended to the original message before the bar code is produced.
Note: The bar code message must consist of all numeric characters if the **UccEanOptionalCheckDigit** property is set to *True*. an error event will be generated if the bar code message contains non-numeric characters.

See Also

EAN/UCC 128 Bar Code Symbology Description (pg. 94)

TAL Bar Code ActiveX Control Methods

Refresh Method

Forces the TALBarCode object to generate a bar code using the current property settings.

Syntax

object.Refresh

The Refresh Method syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object

Remarks
The TALBarCode object will automatically generate a bar code whenever any of its properties are set therefore the **Refresh** method is usually not necessary. This method is provided only as a means for forcing a bar code to be generated at a specific point in your code perhaps for the purpose of trapping errors.

See Also

The BarCodeError Event (pg. 75)

SaveBarCode Method

Saves the bar code image to a disk file.

Syntax

object.SaveBarCode *filename*

The SaveBarCode Method syntax has these parts:

Part	Description
object	An object expression that evaluates to a TAL bar code object
filename	The name of the disk file where you would like to store the image.

Remarks

The Linear Only version of the TAL Bar Code ActiveX control supports saving bar code images to a disk file in only the Windows Metafile graphic format.

The ActiveX Plus version of the TAL Bar Code control supports saving bar codes using the WMF, GIF, JPEG (or JPG), PNG, EPS, TIFF (or TIF), BMP, TGA, and PCX file formats. To save a bar code to a disk file in a particular graphic file format, use the file name extension for the desired format in the filename argument to the SaveBarCode method. If the file name extension that you use is not one that is recognized, then the file will be saved using the Windows Metafile (WMF format).

Note: Saving LZW compressed GIF and TIF files requires purchasing a LZW compression license from Unisys.

The WMF and EPS file formats are vector based image formats meaning that they consist of a sequence of rendering instructions that are used by either the device driver for a printer or directly by a printer to render the image as closely as possible to the original image description. Vector based images produce the highest quality output; especially for bar code images where the widths of the bars and spaces must be rendered precisely. The EPS file format is designed specifically for printing to Postscript printers which makes it less portable than most other formats. The WMF file format is the most portable and most commonly supported vector image file format in most Windows applications. It also produces the highest quality as well as the smallest size bar code image files and will print perfectly on any printer that has a Windows printer driver. It is without question the best graphic file format to use for Windows based bar code applications.

All of the other file formats including GIF, JPEG, PNG, TIFF, BMP, TGA and PCX are raster style images meaning that they are made up of an array of pixel values. Raster images are not device independent meaning that you can get different results on different output devices. The reason for this is that raster images contain a specific number of pixels across and a specific number of rows of pixels. To obtain the highest quality output, raster images should be created at the same "dot resolution" as the printer that they will be printed on.

The dot resolution of most printer is almost always much higher that the dot resolution for most display monitors therefore when you display a raster image on a display monitor, it will be much larger than the same image outputted to a printer. For example if you have a raster image that consists of 300 pixels across and 300 rows of pixels and you print that image on a printer that has a dot resolution of 300 dots per inch, you will end up with a one inch square image. If you render the same image on a display monitor that has a dot resolution of 100 dots per inch, you end up with an image that is three inches square. If you want the image to print at a high enough resolution to produce a high quality bar code (typically 300 DPI or greater) however you want the image to appear on the display monitor at the same size that will actually be printed on the printer, you must scale the image by the ratio of the screen resolution to the printer resolution. Most programs that work with raster images provide features that allow you to scale images to any desired size on screen while retaining any higher resolution information when the image is sent to a printer. Scaling a raster image involves either adding or removing pixels to force the image to a particular size. This process usually causes the image quality to degrade significantly when it is outputted to the screen. Likewise, if you were to create the image at the screen resolution so that it appears at the correct size on screen and then scale the image up before sending it to a printer (so that it prints at the same size as the image on screen), you will end with a printed output that is too low resolution and too low quality to result in a readable bar code.

When creating bar codes, you are always better off creating the image at the resolution of the target printer and then scaling it down on screen to the desired size. This way, the image should at least print at a high resolution even though it may appear low in quality on screen. The important point is that you want your bar codes to be readable from a printed page.

When you do not know the resolution of the target printer (or the resolution of the display device), i.e., when generating bar code images that will be displayed in a web page, the best that you can do is make educated assumptions. You can assume that most of your target users will have a 300 DPI or greater printer and therefore create your bar code images using 300 for the RasterImageResolution property of the TAL Bar Code ActiveX Control.

Because you do not know with certainty the resolution of the display used to view your web pages, you do not know in advance how the image should be scaled in the browser window. You can set the Height and Width of an image in HTML code so that the browser will scale the image to a desired size in pixels, however without knowing the screen resolution you do not know how big a pixel is. The TAL Bar Code ActiveX control provides a Height and Width property (passed back in units of Twips where 1 Twip = 1/1440 inches) so that you can determine the true size of your bar codes however when working with raster images, the actual height and width that you end up with is dependent on the dot resolution of both the image and the output device. Fortunately the Windows operating system always assumes that the screen resolution is 96 pixels per inch no matter what the true value is so scaling images under Windows is trivially easy. It is probably safe to assume that Mac and UNIX systems also use the same value of 96 pixels per inch for all display devices however this has not been confirmed.

If you assume a display resolution of 96 pixels per inch then to convert the height and width values returned by the TAL Bar Code ActiveX control to Pixels using the formulas:

HeightPixels = (TalBarCd.Height * 96)/1440
WidthPixels = (TalBarCd.Width * 96)/1440

When attempting to produce bar codes as raster style images, we strongly recommend that you test the printed output thoroughly to make sure that your bar codes will be readable. Unfortunately we can make no guarantees on the readability of any bar codes rendered as raster images. You should be able to obtain extremely good results under most circumstances if you understand how raster images work and develop your application to work within their limitations.

Saving Uncompressed GIF Files

The most common reason for saving bar codes as GIF files is so that they can be used in web browsers. Unisys Corp. currently holds a patent on the LZW compression algorithm used in compressed GIF files and unfortunately, you are required to purchase a license from Unisys at a cost of between $5000 to $7500 if you want to generate true LZW compressed GIF files on a web server.

Because the TAL Bar Code ActiveX control is considered a "development tool", each user of the control is required to purchase their own license to use the LZW compression scheme. The TAL Bar Code ActiveX control fully supports LZW compression however this functionality must be enabled using a special "unlock key". If you purchase a LZW license from Unisys, you can send TAL Technologies a copy of your license and we will provide you with a special "unlock key" that you can use in the LZWUnlockKey property of the control to unlock this functionality. The LZW Patent is due to expire in the year 2003 after which the LZW compression scheme will enter the public domain and therefore not require you to purchase a license from Unisys.

The GIF specification technically does not support uncompressed GIFs however it is possible to create uncompressed GIFs that trick any program that can read GIF files into rendering them. If you do not purchase the LZW license from Unisys then you can only create uncompressed GIFs however the file size of your GIFs will be much larger than an equivalent compressed GIF bar code image. The trick that must be employed to create uncompressed GIFs actually causes them to be up to four times as large as an equivalent uncompressed bitmap. Uncompressed GIFs are typically eight times larger than the same image saved as a compressed GIF. For example, a typical bar code that is three inches wide and one inch tall saved with a resolution of 300 pixels per inch with one bit per pixel will be roughly 150K bytes in size when saved as an uncompressed GIF. The same bar code saved using GIF compression will be roughly 17K bytes in size.

One technique that you can use to keep the size of uncompressed GIFs to a minimum is to create the bar code with a very small height like 100 mils (one tenth of an inch) and then stretch the bar code height using a higher value in the Height setting in the image tag in your HTML code. If you create the same three inch wide bar code as above using a height of 100 mils (one tenth of an inch) instead of the full one inch, you will cut the size of the file down by a factor of ten. When you set the height setting in your HTML code to a value (in pixels) that is equivalent to one inch (96 in most browsers), the browser will stretch the height of the image to the desired size. Because all of the information encoded in a bar code is contained in the widths of the bars and spaces and not in the height, stretching the height will not degrade the readability of the bar code. Using this technique, you can drastically reduce the download time for both compressed and uncompressed GIF files. In the example above, our unreasonably large 150K byte uncompressed GIF image file drops in size to a perfectly reasonable 15K bytes.

Note: The PNG file format is an excellent alternative to the GIF file format. PNG files are highly compressed using a public domain algorithm therefore no license fee is required to use them. Most browsers including Netscape 4.0, Internet Explorer 4.0, and all newer releases of both, fully support PNG files. The same three inch by one inch bar code as in the above example squeezes down to an image file size of under 1000 bytes when saved as a PNG file.

See Also

The BitsPerPixel Property (pg. 17)
The RasterImageResolution Property (pg. 61)
The LZWUnlockKey Property (pg. 40)
Bar Code Dimensions (pg. 81)
The NarrowBarWidth Property (pg. 48)
The BarHeight Property (pg. 14)
Introduction to Bar Code Technology (pg. 76)
Printing Bar Codes From Microsoft Windows (pg. 78)
How To Produce Readable Bar Codes (pg. 83)

TAL Bar Code ActiveX Control Events:

The TAL Bar Code ActiveX Control provides a single native event, the "BarCodeError" event, that is fired when you try to generate a bar code using invalid input properties.

Different application programs that can host ActiveX controls will also provide support for additional events after the control has been placed in the container application. For example, after you insert the TAL Bar Code ActiveX Control on a form in a Visual Basic program, VB will automatically provide the control with a Click, DblClick, DragDrop, DragOver, MouseUp, MouseDown, MouseMove, KeyUp, KeyDown, KeyPress, GotFocus, LostFocus and Validate events. Other application programs may provide similar events depending on the application program. You would need to actually place the control in the other application first to determine what additional events it will support.

If you place the control in a Microsoft Access report, an "OnFormat" event will be fired for each section of the report immediately before the section is printed for each page. The OnFormat event therefore gives you the opportunity modify the properties of controls on a report immediately before each page is printed.

BarCodeError Event

Occurs when an attempt is made to generate a bar code using invalid property values.

Syntax
Sub *object*_BarCodeError([*index* As Integer,]*errornum* As Long)

The BarCodeError event syntax has these parts:

Part	Description
object	An object expression that evaluates to a TALBarCode object.
index	An integer that uniquely identifies a control if it's in a control array.
errornum	A long integer that identifies the error that occurred. See Error Codes below.

The value returned in errornum will be one of the following values:

Number	Description	Details
1	Invalid Message	Message is either empty or contains invalid characters for the chosen symbology.
2	Invalid message length	Message is too long
3	Invalid comment length	Comment length is too long
4	Invalid supplemental	UPC or EAN supplemental contains non numeric data
5	Invalid supplemental length	UPC or EAN supplemental has other than 2 or 5 digits
6	Invalid narrow bar width	Narrow bar width is either <= 0 or greater than 500
7	Invalid bar width reduction/gain	Bar width reduction is not between -99% to 99%
8	Invalid height	Height for standard bar codes must range from .1 to 200
9	Invalid foreground or background color	Color values must range from 0 to hex FFFFFF
10	Invalid orientation value	Orientation property is outside the allowable range 0 – 3
12	Invalid narrow to wide ratio	Must be a value between 2.0 and 3.0
13	Invalid font size	Font size is either zero or greater than 1000
14	Bar code too large	Overall dimensions of the bar code were too large.
17	Unable to create metafile file	Either a bad filename was specified or the DLL was unable to open file, etc...
101	Invalid Number of Bits Per Pixel	The BitsPerPixel property is set to a value that is not supported by the type of graphic that you are trying to create.
102	Could Not Save File	An error occurred while trying to save a bar code image to a disk file.
103	Could Not Allocate Memory	An error occurred while trying to allocate memory for an image to be saved to a disk file.
104	Invalid Dimensions	The dimensions for a bar code were either invalid or too large prior to saving a bar code to a disk file.

Introduction to Bar Code Technology

Bar code is an automatic identification technology that allows data to be collected rapidly and with extreme accuracy. Bar codes provide a simple and easy method of encoding text information in a printed symbol that is easily read by inexpensive electronic readers. A typical bar code consists of a series of parallel, adjacent bars and spaces. Pre-defined sets of bar and space patterns (symbologies) are used to represent individual characters in a printed symbol.

The data in a normal linear type bar code is encoded in the widths of the bars and spaces similar to the way Morse code encodes characters using audible dots and dashes.

Shown Below is the structure of a typical bar code symbol.

The basic structure of a bar code consists of a leading and trailing quiet zone, a start pattern, one or more data characters, optionally one or two check characters and a stop pattern.

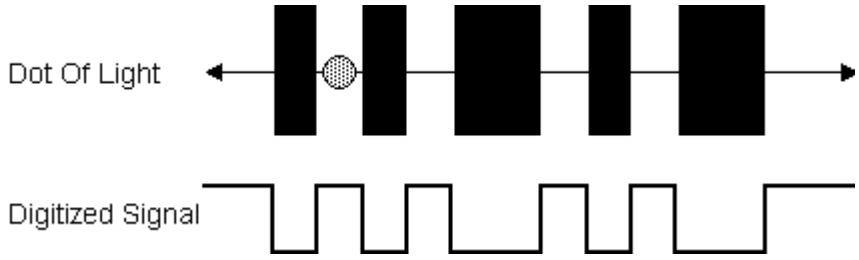
There are a variety of different bar code symbologies, each of which were developed to fulfill a specific need. Several of these symbologies have matured into de-facto standards that are used universally today throughout many industries. The bar code types supported by the TAL Bar Code Control are the most commonly used symbologies across all industries.

The different symbologies have different capabilities for encoding data. For example the UPC symbology used to identify retail products always contains 12 numeric digits whereas the general purpose Code 39 or Code 128 bar code symbologies can encode variable length alphanumeric data up to about 30 characters in length. These types of bar codes are called "linear symbologies" because they are made up of a series of lines of different widths. Most commercially available bar code scanners are able to read all of the different linear bar code symbologies therefore you do not need different readers for different types of bar codes.

New "2-Dimensional" bar code symbologies like PDF417, Aztec Code, and Data Matrix can encode several thousand bytes of data in a single bar code symbol including text or binary data. The newer 2D bar code symbologies typically require special bar code readers that are designed specifically for reading them.

How A Bar Code Reader Works

Bar codes can be thought of as a printed version of the Morse code with narrow bars representing dots, and wide bars representing dashes. A bar code reader decodes a bar code by scanning a light source across the bar code and measuring the intensity of light reflected back. The pattern of reflected light produces an electronic signal that exactly matches the printed bar code pattern. This signal is then decoded into the original data. Due to the design of most bar code symbologies it does not make any difference if you scan a bar code from right to left or from left to right. Special start and stop patterns are used to indicate which end of the bar code is the beginning and which is the end.



Bar code scanners can be purchased with different resolutions to enable them to read bar codes of different sizes (bar widths). The scanner resolution is measured by the size of the dot of light. The dot of light should be equal to or slightly smaller than the width of the narrowest bar ("X" dimension). If the dot is wider than the width of the narrowest bar or space, then the dot of light will overlap two or more bars at a time and there will not be sharp transitions in the digitized waveform produced by the circuit that measures the intensity of the light reflected back to the reader. If the dot is too small, then any spots or voids in the bars can be misinterpreted as light areas thus making a bar code unreadable.

The factors that make a bar code readable are: an adequate print contrast between the light and dark bars and having all bar and space dimensions within the tolerances for the symbology. It is also helpful to have sharp bar edges, few or no spots or voids, a smooth surface and clear margins or quiet zones at either end of the printed symbol.

See Also

Printing Bar Codes From Microsoft Windows (pg. 78)
Bar Code Symbology Descriptions and Rules (pg. 84-104)
Bar Code Dimensions (pg. 81)
How To Produce Readable Bar Codes (pg. 83)

Printing Bar Codes From Microsoft Windows

Because of the way a bar code reader works, it is extremely important that the widths of all bars and spaces are printed within precise tolerances. If the width of a single bar or space is just slightly off, it can cause a bar code symbol to be unreadable. How the bar code is produced including the type of graphic that is used is therefore extremely important. Microsoft Windows supports three native graphic types that are commonly used to create bar codes - bitmaps (raster graphics), fonts and metafiles. Of the three types of graphics, only the metafile format provides enough resolution and "device independence" to consistently produce bar codes within the necessary tolerance required by all standard bar code readers. This is the technology that is used by the TAL Bar Code ActiveX Control. Almost all other bar code programming tools on the market use either bitmaps or fonts as the basic graphic technology and as a result, none of them produce as high a quality bar code graphic as the TAL Bar Code ActiveX Control. The following paragraphs provide a detailed description of each of the different graphic formats.

BitMaps and other Raster Style Graphics

A bitmap is an array of dots or "pixels" where each pixel (picture element) has a value that represents the color of the pixel. The width of a bitmap is defined by the number of pixels across a row and the height is defined by the number of rows of pixels. Any graphic that is made up of rows of pixels is called a "Raster" graphic therefore a bitmap is a raster graphic. TIF, GIF, JPEG and most other standard graphic formats are also raster style images and for the purposes of this discussion are also categorized as "bitmaps". The overall printed dimensions of a raster image is dependent on the dot resolution of the device used to print it. For example if you create a bitmap that is 300 pixels wide and has 300 rows of pixels and then you print this bitmap on a printer that has a dot resolution of 300 dots per inch, you will end up with a printed image one inch square. If you display the same bitmap on a computer screen that has a dot resolution of 100 dots per inch, you end up with an image that is three inches square. This means that bitmaps are "device dependent" where the resolution of the rendering device (i.e. printer or screen) must be taken into consideration when you create the image. If you need to preserve the size of an image from one device to another, (i.e. screen to printer) you must "stretch" or "shrink" a bitmap to fit the desired size. The process of stretching and shrinking involves either adding or removing pixels to or from the original image. As you can imagine, resizing a bitmap to a desired size when moving from one output device to another generally causes a severe degradation of the original image quality. When creating precise graphics (like bar codes) it is extremely important that the image is created with the same dot resolution of the printer. If you do not know the dot resolution of the printer that will be used to print the bar code then you cannot fully guarantee that the image will be readable by all bar code readers.

A type of bitmap called a "Device Independent Bitmap" has been developed as an attempt to solve the device dependency problem however you still end up with distorted images when you render a device independent bitmap on a device that does not have the exact same dot resolution that the bitmap was created with.

Another problem with bitmaps is that they require large amounts of memory. A bitmap that is 300 pixels wide and 300 lines down (only 1 square inch on a 300 DPI laser printer) and has three bytes per pixel of color information (standard 24 bit RGB colors) will require 270,000 bytes of memory or disk space. For bar codes, this is a huge amount of memory for a very simple graphic made up of a relatively small number of rectangles.

Fonts

Although fonts are not normally thought of as graphics, they can be used to produce bar codes. A font is a collection of up to 256 different graphic elements that are assigned to each of the characters in the ASCII or ANSI character set. Because most bar code symbologies encode data by mapping characters to specific bar and space patterns, it is possible to use fonts to create bar codes. Unfortunately, fonts have many inherent problems when they are used to create bar codes.

The most important problem with fonts when they are used to create bar codes is that they are not "intelligent". Almost every bar code symbology has features like start and stop patterns, check digits, guard patterns, quiet zones and bearer bars. When you use a font to create a bar code you cannot simply select the text for the message that you want encoded and change the font to a bar code font. You first have to insert special characters for the start and stop patterns as well as manually calculate and insert a special character for the check digit. In almost all cases you have to use a special program provided with the font to calculate and add check digits and insert start and stop patterns.

Another problem with fonts is that they cannot be scaled in a single direction. If you increase the size of a font, both the height and the width change. When creating bar codes, it is extremely important that the width of the bars and spaces remain constant. Typically the height of a bar code font is not adequate and it must be scaled up. When you do this by increasing the font size, the width of the bars and spaces as well as the overall width of the bar code increases proportionally which causes the bar code symbol to be rendered out of spec. In general fonts are the worst choice for creating bar codes. They offer the least control and produce the lowest quality output.

The Uniform Code Council, Inc., Guidelines for Providers of EAN/UPC Symbol Design Software states that:
"Bar code fonts have been known to create EAN/UPC symbols with serious quality defects. The problems may be caused by the inherent design of the font, operator input, or a combination of both..."
"For these reasons, extreme caution should be used when producing EAN/UPC symbols with bar code fonts. They should only be used by highly experienced bar code design professionals utilizing appropriate [process] controls."

Metafiles (i.e. vector graphics)

A metafile is a very precisely detailed description of an image. Instead of containing an actual raster style image like a bitmap, a vector graphic contains a sequence of drawing instructions that describe how to render the image. For example it might contain an instruction that tells the output device to move to a point exactly two inches down and to the right from the upper left corner of the screen or page and draw solid black rectangle that is exactly thirteen mils wide and one inch tall. Because metafiles contain drawing instructions instead of an actual raster style image, they are completely device independent meaning that you should obtain consistently high quality output no matter what printer you print to. When you print a metafile to a printer or a screen device, the device driver for the output device is responsible for taking the instructions in the metafile and rendering the image using the highest dot resolution that the device supports.

The Windows metafile (WMF) and the Encapsulated Postscript (EPS) graphics formats are the two most widely used "vector" graphic formats. Unfortunately the EPS format is only supported by Postscript printers however the WMF format is fully supported by all printers that have a Windows printer driver (including Postscript printers). The prominent features of the WMF format are that it is completely device independent, it supports extremely precise dimensions for all graphic elements (down to 1/100th of a millimeter) and the amount of memory required to store a metafile is extremely small. Most importantly, every printer that has a Windows printer driver must support printing metafiles therefore there is never an issue with being able to print metafiles on a particular printer. The characteristics of metafiles are ideal for creating bar codes in the Windows environment.

The TAL Bar Code ActiveX Control uses the Windows Metafile graphics format for all bar code symbols. Because of this, the TAL Bar Code ActiveX Control produces the highest quality bar codes possible in the Windows environment.

See Also

Introduction to Bar Code Technology (pg. 76)
Bar Code Symbology Descriptions and Rules (pg. 84-104)
Bar Code Dimensions (pg. 81)
How To Produce Readable Bar Codes (pg. 83)

Bar Code Dimensions

Linear Bar Code Symbol Dimensions

Because of differences in the design of each bar code symbology, there are differences in the way that the dimensions for each symbology are expressed. For most linear bar code symbologies, the two main dimensions used to define the size of most bar codes are the Narrow Bar Width and the overall Bar Height. The Height is generally less important than the Narrow Bar Width and you can scale the height to any size that you like. For the sake of readability the height should not be less than a quarter of an inch or 15% of the total width of the symbol, whichever is greater.

UPC and EAN bar codes have similar size requirements except that the terminology is slightly different. The specifications for UPC and EAN bar codes require a narrow element width of exactly 13 mils and a bar height of one inch; however, the specifications for these symbologies allow for a magnification factor of between 80% and 200%. This translates to an allowable range of narrow element widths of between 10.4 and 26 mils and bar heights between .8 and 2 inches. The TAL Bar Code Control does not have a parameter for a magnification factor, however you can scale the narrow bar width and the bar height accordingly by multiplying the **NarrowBarWidth** and the **BarHeight** properties by the desired magnification factor (.8 to 2).

The Narrow Bar Width effectively determines the total width of a bar code symbol. All other bar and space width dimensions are based on this width (referred to as the nominal X dimension). The best choice for this dimension depends partly on the resolution of your bar code reading equipment and also on the resolution of the printer being used to produce the bar code.

As a general rule for all linear bar code symbologies, the Narrow Bar Width should fall in a range between 10 to 30 mils (.25 to .76mm) and should never be less than 7.5 mils (13 mils (.33mm) is the most commonly recommended value for most bar code readers) . For UPC and EAN bar codes, the smallest allowable Narrow bar width is 10.4 mils (.26mm). One way to determine a good Narrow Bar Width is to actually print out a typical bar code using several different values and try reading each one with your scanning equipment. You should choose the value that produces bar codes with the highest "first pass" read rate.

X=13 Sample

X=20 Sample

2-Dimensional Bar Code Symbol Dimensions

The two dimensional bar code symbologies like PDF417, Data Matrix, Aztec Code and MaxiCode encode data in an entirely different manner than traditional linear bar codes and therefore use different properties for their dimensions. The Data Matrix and Aztec Code symbologies are designed around a 2-dimensional matrix of square dots or “modules” and the **MatrixModuleSize** is the only dimension property that you can specify. With Data Matrix and Aztec Code, you specify the module size and the overall size of the bar code is determined by the amount of data that will be encoded as well as the type of error correction to be used when the data is encoded. With Data Matrix and Aztec Code you always end up with a square bar code symbol.

PDF417 works like a stacked set of linear bar codes and it provides a great deal of flexibility in how you specify the overall size and shape of the symbol. Similar to the matrix style symbologies, the smallest element in a PDF417 symbol is also called a module however you can specify both a width and a height for the modules. In addition to the module height and width, you can also specify an “Aspect Ratio” that determines the overall height to width ratio of the symbol. Other parameters can be used to constrain the overall size of the symbol or specify how much error correction overhead to use.

MaxiCode bar codes are fixed in size and therefore provide no way to change the size of a symbol.

See Also

The NarrowBarWidth Property (pg. 48)
The BarHeight Property (pg. 14)
The NarrowToWideRatio Property (pg. 49)
The MatrixModuleSize Property (pg. 41)
PDF417 Bar Code Dimensions (pg. 96)
Printing Bar Codes From Microsoft Windows (pg. 78)

How To Produce Readable Bar Codes

Although there are many different types of bar codes, they all share the same requirements in order to be readable by most commercially available scanning devices. Because a beam of light is used to read a bar code, it should be clean and free of defects or smudges and there should be a high contrast between the color of the bars and the color of the spaces. Black bars on a white background yield the best results. If you intend to use colored bar codes or colored paper, you should always test the readability of your bar codes before committing to a color scheme.

Another important consideration is that there should always be a small amount of space or Quiet Zones preceding and following the bar code so that the reading device is able to properly determine the true start and end of the bar code symbol. A good rule of thumb is to reserve at least a quarter of an inch or 10 times the width of a single narrow bar (whichever is greater) for blank space at either end of a bar code. The TAL Bar Code Control will automatically include Quiet Zones of ten times the narrow bar width if the **QuietZones** property is set to True

When printing bar codes, laser, ink jet and thermal transfer printers produce the best results. Dot matrix printers produce the poorest quality, but not necessarily unacceptable bar codes. Dot matrix printers are especially poor when the bar code dimensions are set to a small size. You should try to avoid very small or very large bar codes, both narrow bar widths and overall bar code dimensions. As a precaution you should always test your printed output with whatever bar code reading equipment you eventually intend to use.

See Also

Printing Bar Codes From Microsoft Windows (pg. 78)
Bar Code Symbology Descriptions and Rules (pg. 84-104)

Bar Code Symbology Descriptions and Rules

The TAL Bar Code ActiveX Control supports the following linear bar code symbologies:

Symbology	**Also Known As:**
Code 39	C-39, Code 3 of 9, AIAG
Code 39 Full ASCII	
HIBC Code 39	HIBC
UPC-A	
UPC-E	Zero Suppressed UPC
EAN - 8	JAN - 8, EAN/JAN - 8
EAN - 13	JAN - 13, EAN/JAN - 13
Bookland	(same as EAN - 13)
Code 93	
Codabar	
Interleaved 2 of 5	ITF, ITF-14, I 2 of 5
Code 128	C - 128
EAN/UCC 128	UCC/EAN 128, UCC 128, EAN 128
MSI-Plessey	Plessey Code
PostNET	ZIP + 4, ZIP + 6

If you purchased the TAL Bar Code ActiveX Control with support for 2 dimensional bar codes, then the following symbologies will also be supported:

Symbology	**Original Publisher of Specification**
PDF417	Symbol Technologies, Inc (www.symbol.com)
Aztec Code	Welch Allyn (www.hhp.com)
Data Matrix	CI Matrix
Maxicode	United Parcel Service (www.maxicode.com)
RSS14	Uniform Code Council (www.uc-council.org)

The following pages contain a brief overview of each of the different bar code symbologies including a general description and all rules that describe the type and amount of data that may be encoded in each one. For more detailed descriptions of each symology please contact the Automatic Identification Manufacturers Association (AIM USA) at:

AIM, Inc.
634 Alpha Drive, Pittsburgh, PA 15238-2802
Phone: +1 412 963 8588
Fax: +1 412 963 8753
Email: info@aimglobal.org
www.aimglobal.org

CODE 39 (Normal, Full ASCII and HIBC versions)

Normal Code 39

The Normal CODE 39 is a variable length symbology that can encode the following 44 characters: 1234567890ABCDEFGHIJKLMNOPQRSTUVWXYZ-. *$/+%. Code 39 is the most popular symbology in the non-retail world and is used extensively in manufacturing, military, and health care applications. Each Code 39 bar code is framed by a start/stop character represented by an asterisk (*). The Asterisk is reserved for this purpose and may not be used in the body of a message. The TAL Bar Code Control automatically adds the start and stop character to each bar code therefore you should not include them as part of your bar code message. If you select the Normal version of Code 39 and your bar code text contains lower case characters, the control will convert them to upper case. If your bar code text contains invalid characters, the control will raise an error event with the error code for an "Invalid Message" error and a bar code will not be generated.

Code 39 allows for an optional (modulo 43) check. The health care industry has adopted the use of this check character for health care applications (HIBC bar codes). To enable the Code 39 check, set the **Code39OptionalCheckDigit** property to True. When this option is enabled, the control will automatically calculate and append the proper check character to all Code 39 symbols. If you set the **Symbology** property to bcCode39_HIBC then the control will automatically add the check digit no matter what value the **Code39OptionalCheckDigit** property is set to.

Full ASCII Code 39

The FULL ASCII version of Code 39 is a modification of the NORMAL (standard) version that can encode the complete 128 ASCII character set (including asterisks). The Full ASCII version is implemented by using the four characters: $/+%. as shift characters to change the meanings of the rest of the characters in the Normal Code 39 character set. Because the Full ASCII version uses shift characters in combination with other standard characters to represent data not in the Normal Code 39 character set, each non-standard character requires twice the width of a standard character in a printed symbol. The table below shows the character combinations used to produce the Full ASCII version of Code 39.

ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39	ASCII	CODE 39
NUL	%U	SP	SPACE	@	%V	`	%W
SOH	$A	!	/A	A	A	a	+A
STX	$B	"	/B	B	B	b	+B
ETX	$C	#	/C	C	C	c	+C
EOT	$D	$	/D	D	D	d	+D
ENQ	$E	%	/E	E	E	e	+E
ACK	$F	&	/F	F	F	f	+F
BEL	$G	'	/G	G	G	g	+G
BS	$H	(	/H	H	H	h	+H
HT	$I	)	/I	I	I	i	+I
LF	$J	*	/J	J	J	j	+J
VT	$K	+	/K	K	K	k	+K
FF	$L	,	/L	L	L	l	+L
CR	$M	-	-	M	M	m	+M
SO	$N	.	.	N	N	n	+N
SI	$O	/	/O	O	O	o	+O
DLE	$P	0	0	P	P	p	+P
DC1	$Q	1	1	Q	Q	q	+Q
DC2	$R	2	2	R	R	r	+R
DC3	$S	3	3	S	S	s	+S
DC4	$T	4	4	T	T	t	+T
NAK	$U	5	5	U	U	u	+U
SYN	$V	6	6	V	V	v	+V
ETB	$W	7	7	W	W	w	+W
CAN	$X	8	8	X	X	x	+X
EM	$Y	9	9	Y	Y	y	+Y
SUB	$Z	:	/Z	Z	Z	z	+Z
ESC	%A	;	%F	[	%K	{	%P
FS	%B	<	%G	/	%L	:	%Q
GS	%C	=	%H	]	%M	}	%R
RS	%D	>	%I	^	%N	~	%S
US	%E	?	%J	_	%O	DEL	%T,%X,%Y,%Z

Note: Because all of the characters used to implement Full ASCII Code 39 are part of the Normal Code 39 character set, readers that do not support Full ASCII Code 39 will still read Full ASCII Code 39 symbols. The reader will output shifted characters as if they were normal Code 39 characters. For example the following string: PROGRAMMING=FUN will be read as:
PROGRAMMING%HFUN by a reader that only supports Normal Code 39. Instead of converting the Full ASCII encoded characters %H to an equal sign, the reader blindly outputs %H.

HIBC Code 39

HIBC or "Health Industry Bar Code" is a specification for a standard method of encoding data using Code 39 in healthcare applications. The standard requires that the first character in a HIBC bar code be a plus character (+) and that the symbol use a MOD 43 check digit. To generate HIBC bar codes, all you need to do is to set the **Symbology** property to *bcCode39_HIBC*. and the TAL Bar Code control will automatically add the starting plus sign to your bar code message as well as calculate and append the check digit for you.

Code 39 Symbol Concatenation

Some bar code readers support a feature of Code 39 that allows for concatenation of two or more bar codes so that you can break long messages into multiple, shorter, symbols. If the first data character of a Code 39 symbol is a space, some readers will store the remainder of the symbol in a buffer and not transmit the data. This operation continues for all successive Code 39 symbols with a leading space, with each message appended to the previous one. When a message without a leading space is read, it is appended to the previously scanned data and the entire buffer is transmitted as one long message.

UPC-A, UPC-E, and UPC Supplementals

UPC-A is a 12 digit, numeric symbology used in retail applications. UPC-A symbols consist of 11 data digits and one check digit. The first digit is a number system digit that usually represents the type of product being identified. The following 5 digits are a manufacturers code and the next 5 digits are used to identify a specific product. UPC numbers are assigned to manufacturers by the Uniform Code Council (UCC). For more information, contact the UCC at 8163 Old Yankee Road, Suite J, Dayton, OH 45458 Tel: 937-435-3870, web site: http://www.uc-council.org

UPC-E is a smaller, six-digit UPC symbology for number system 0. It is often used for small retail items like candy and cigarettes. UPC-E is also called "zero suppressed" because UPC-E compresses a normal 12-digit UPC-A code into a six digit code by "suppressing" the number system digit, trailing zeros in the manufacturers code and leading zeros in the product identification part of the bar code. A seventh check digit is encoded into a parity pattern for the six main digits. Many bar code readers can be configured to automatically uncompress UPC-E symbols into their 12-digit equivalent before transferring the data to a computer.

When specifying UPC-A or UPC-E messages, you may pass a message with either 6,7,11 or 12 digits. If you pass a message with 6 or 7 digits the TAL Bar Code control will generate a UPC-E symbol. If you pass 7 digits the 7th digit will be removed. (The control will assume that you are entering a message complete with check digit.) If you pass a message with 11 or 12 digits the TAL bar code control will generate a UPC-A symbol. If you pass 12 digits the 12th digit will be removed. (Again the control will assume that you are entering a message with a check digit.) The TAL Bar Code control automatically calculates the check digit for you and appends it to your bar code message text. This insures that you always have the correct check digit.

Both UPC-A and UPC-E allow for a supplemental two or five digit number to be appended to the main bar code symbol. The supplemental is a small additional bar code that is added onto the right side of a standard UPC symbol. This supplemental message was designed for use on publications and periodicals. To include a supplemental message, append it to the main message with a comma separating it from the main message. If you enter a supplemental message, it must consist of either two or five numeric digits.

The following table contains sample messages for the different variations of UPC symbols.

Message	**Symbol Generated**
123456	UPC-E
123456,12345	UPC-E with a five digit supplemental.
123456789012	UPC-A
12345678901,12	UPC-A with a two digit supplemental.

EAN-8 / EAN-13, and EAN Supplementals

EAN or European Article Numbering system (also called JAN in Japan) is a European version of UPC. It uses the same size requirements and a similar encoding scheme as UPC codes.
EAN-8 encodes 8 numeric digits consisting of two country code digits, five data digits and one check digit.
EAN-13 is the European version of UPC-A. The difference between EAN-13 and UPC-A is that EAN-13 encodes a 13th digit into the parity pattern of the left six digits of a UPC-A symbol. This 13th digit, combined with the 12th digit, usually represent a country code.

EAN bar code numbers are assigned to specific products and manufacturers by an organization called ICOF located in Brussels, Belgium. Tel: 011-32-2218-7674, web site: http://www.ean.be/

When specifying EAN-8 or EAN-13 messages, you may pass a message with either 7,8, 12 or 13 digits. If you pass a message with 7 or 8 digits the TAL Bar Code control will generate a EAN-8 symbol. If you pass 8 digits, the 8th digit will be removed. (The control will assume that you are entering a message complete with check digit.) If you pass a message with 12 or 13 digits the TAL Bar Code control will generate a EAN-13 symbol. If you pass 13 digits, the 13th digit will be removed. (Again the control will assume that you are entering a message with a check digit.) The TAL Bar Code control automatically calculates the check digit for you and appends it to your bar code message text. This insures that you always have the correct check digit.

Like UPC, EAN-8 and EAN-13 allow for a supplemental two or five digit number to be appended to the main bar code symbol. The supplemental is simply a small additional bar code that is added onto the right side of a standard EAN symbol. To include a supplemental message, append it to the main message with a comma separating it from the main message. If you enter a supplemental message, it must consist of either two or five numeric digits.

The following table contains sample messages for the different variations of EAN symbols.

Message	**Symbol Generated**
1234567	EAN-8 no supplemental
1234567,12345	EAN-8 with a five digit supplemental.
123456789012	EAN-13 no supplemental
1234567890128	EAN-13 no supplemental
123456789012,12	EAN-13 with a two digit supplemental.

BookLand

EAN-13 has been adopted as the standard in the publishing industry for encoding ISBN numbers on books. An ISBN or BookLand bar code is simply an EAN-13 symbol consisting of the first 9 digits of an ISBN number preceded by the digits 978 and terminated with a standard EAN check digit. The supplemental in an ISBN bar code is usually the retail price of the book preceded by the digit 5 or it can be the number 90000 when no price is specified.

For example, if you want to encode the ISBN number 1-56276-008-4 and the price of the book is $29.95 then you could use 978156276008 as the main bar code message and 52995 for the supplemental with the **Symbology** property set to bcEAN_JAN_13 and the TAL Bar Code control would generate the correct BookLand bar code.

If you set the **Symbology** property to bcBookLand, the TAL Bar Code control only requires you to use the 10 digit ISBN number (with or without dashes) and optionally a price for the supplemental separated from the ISBN number with a comma. This feature saves you the trouble of having to remove the 10th digit from the ISBN number, insert the preceding 978 and further process the price supplemental.

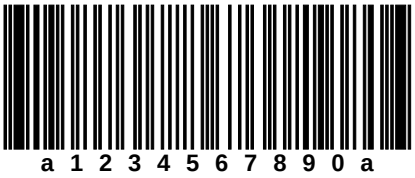
The following table contains sample messages for the different variations of BookLand symbols. The following samples apply only when the **Symbology** property is set to bcBookLand.

BookLand Message	Message in Generated BookLand Bar Code
1-56276-008-4	97815602763 no supplemental
1562760084,29.95	97815602763 with supplemental 52995
1-56276-008-4,2995	97815602763 with supplemental 52995
1-56276-008-4,395	97815602763 with supplemental 50395
1234567890,90000	9781234567897 with supplemental 90000
1234567890,12	9781234567897 with 2 digit supplemental 12

CODE 93

CODE 93 is a variable length symbology that can encode the complete 128 ASCII character set. Code 93 was developed as an enhancement to the CODE 39 symbology by providing a slightly higher character density than CODE 39. CODE 93 also incorporates two check digits as an added measure of security. Although CODE 93 is considered more robust than CODE 39, it has never achieved the same popularity as Code 39. CODE 93 bar codes are framed by a special start/stop character. The TAL Bar Code control will automatically add the start and stop characters as well as the check digits to each Code 93 bar code therefore you should not attempt to include them as part of your bar code message.

CODABAR



CodaBar is a variable length symbology that allows encoding of the following 20 characters: 0123456789-$:/.+ABCD. CodaBar is commonly used in libraries, blood banks, and the air parcel business. CodaBar uses the characters A B C and D only as start and stop characters. Thus, the first and last digits of a CodaBar message must be A B C or D and the body of the message should not contain these characters. The TAL Bar Code control will allow any length of CodaBar message as long as it contains valid characters and starts and ends with a valid start/stop character. If you use lower case letters for A B C or D, the control will convert them to upper case.

INTERLEAVED 2 OF 5 (ITF)

Interleaved 2 of 5 is a high-density variable length numeric only symbology that encodes digit pairs in an interleaved manner. The odd position digits are encoded in the bars and the even position digits are encoded in the spaces. Because of this, Interleaved 2 of 5 bar codes must consist of an even number of digits. Also, because partial scans of I 2 of 5 bar codes have a slight chance of being decoded as a valid (but shorter) bar code, readers are usually configured to read a fixed (even) number of digits when reading Interleaved 2 of 5 symbols. The number of digits are usually pre-defined for a particular application and all readers used in the application are programmed to only accept ITF bar codes of the chosen length. Shorter data can be left padded with zeros to fit the proper length. The TAL Bar Code Control will only accept numeric digits for Interleaved 2 of 5 bar codes. If an odd number of digits is entered, the control will Left-Pad one zero to the number entered.

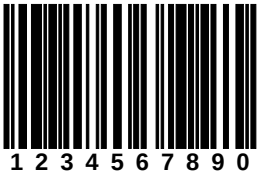
Interleaved 2 of 5 optionally allows for a weighted modulo 10 check character for special situations where data security is important. To enable the Interleaved 2 of 5 check character, set the **I2of5OptionalCheckDigit** property to True and the control will automatically calculate and append the proper check character to all Interleaved 2 of 5 symbols.

CODE 128

Code 128 is a variable length, high density, alphanumeric symbology. Code 128 has 106 different bar and space patterns and each pattern can have one of three different meanings depending on which of three different character sets is employed. Special start characters tell the reader which of the character sets is initially being used and three special shift codes permit changing character sets inside a symbol. One character set encodes all upper case characters and ASCII control characters, another encodes all upper and lower case characters and the third set encodes numeric digit pairs 00 through 99. This third character set effectively doubles the code density when printing numeric data. Code 128 also employs a check digit for data security. In addition to ASCII characters, Code 128 also allows encoding of four special function codes (FNC1 - FNC4). The meaning of function code FNC1 and FNC4 were originally left open for application specific purposes. Recently an agreement was made by the Automatic Identification Manufacturers Assoc. (AIM) and the European Article Numbering Assoc. (EAN) to reserve FNC1 for use in EAN applications. FNC4 remains available for use in closed system applications. FNC2 is used to instruct a bar code reader to concatenate the message in a bar code symbol with the message in the next symbol. FNC3 is used to instruct a bar code reader to perform a reset. When FNC3 is encoded anywhere in a symbol, any data also contained in the symbol is discarded. The four function codes can be included in a message by using the ASCII characters ASCII 128 for FNC1, ASCII 129 for FNC2, ASCII 130 for FNC3 and ASCII 131 for FNC4.

Note: The TAL Bar Code control will automatically select the proper character sets and insert the necessary start character and shift codes so that the resulting bar code will be as short as possible. The check digit will also be calculated automatically by the control.

EAN/UCC 128



The EAN/UCC 128 symbology is a variation of the original Code 128 symbology designed primarily for use in product identification applications. The EAN/UCC 128 specification uses the same code set as Code 128 except that it does not allow function codes FNC2-FNC4 to be used in a symbol and FNC1 is used as part of the start code in the symbol. Some applications for EAN/UCC 128 require an additional Mod 10 check digit. You can enable this check digit by setting the **UccEanOptionalCheckDigit** property to True. If the optional check digit is enabled, the data that you supply in the **Message** property must consist of numeric characters only otherwise an error event is raised and no bar code symbol is generated.

MSI-PLESSEY

MSI-PLESSEY is a variable length, numeric only, symbology. The symbology is one of the earliest bar code symbologies ever developed and is based on a four bit binary number scheme. Each symbol is framed by a start and a stop pattern and contains a check character that is calculated from the values of each of the encoded data digits. MSI-Plessey is rarely used in anything other than grocery store shelf marking applications. Because the symbology is so rarely used, most modern bar code readers do not provide support for MSI-Plessey symbols.

POSTNET

POSTNET (Postal Numeric Encoding Technique) is a 5, 9 or 11 digit numeric only bar code symbology used by the US Postal Service to encode ZIP Code information for automatic mail sorting. The bar code may represent a five digit ZIP Code (32 bars), a nine digit ZIP + 4 code (52 bars) or an eleven digit Delivery Point code (62 bars).

POSTNET is unlike other bar codes because data is encoded in the height of the bars instead of in the widths of the bars and spaces. Most commercially available bar code readers cannot decode POSTNET. This symbology was chosen by the Postal Service mainly because it is extremely easy to print on almost any type of printer. POSTNET is a fixed dimension symbology meaning that the height, width and spacing of all bars must fit within tight tolerances. The TAL Br Code control will only create POSTNET bar codes that follow the guidelines published by the Postal Service. The control does not allow you to set the size of POSTNET bar codes. In other words the **NarrowBarWidth**, **BarWidthReduction** and the **BarHeight** properties are ignored. POSTNET also does not support human readable text, quietzones, bearer bars or optional check digits.

The TAL Bar Code control will ignore non-numeric data in any POSTNET bar code message. For example, if you supply "Chicago, IL 60601-3222" for the **Message** property of a POSTNET bar code, the control will still create a correct bar code. This feature allows you to pass an address line from another Windows program to the control without having to parse out the Zip code.

PDF417

PDF417 is a high density 2 dimensional bar code symbology that resembles a stacked set of smaller bar codes. The symbology is capable of encoding the entire (256 character) ASCII or ANSI character set. PDF stands for "Portable Data File" because it can encode as many as 2710 data characters in a single bar code. The complete specification for PDF417 provides many encoding options including data compaction options, error checking and correction options, and variable size and aspect ratio symbols. The low level structure of a PDF417 symbol consists of an array of code words (small bar and space patterns) that are grouped together and stacked on top of each other to produce the complete printed symbol. An individual code word consists of a bar and space pattern 17 modules wide. The user may specify the module width, the module height, and the overall aspect ratio (overall height to width ratio) for the complete symbol. A complete PDF417 symbol consists of at least 3 rows of up to 30 code words and may contain up to 90 code word rows per symbol with a maximum of 928 code words per symbol.

The code words in a PDF417 symbol are generated using one of three data compaction modes currently defined in the symbology specifications. This allows more than one character to be encoded into a single data code word. Because different data compaction algorithms may be used, it is possible for different printed symbols to be created from the same input data. The TAL Bar Code control will automatically switch between the different data compaction modes to produce the smallest possible bar code symbol for a given bar code message.

The PDF417 symbology also allows for varying degrees of data security or error correction and detection. Nine different error correction levels are available with each higher level adding additional overhead to the printed symbol. The TAL Bar Code control also provides an “automatic” error correction option that can help make the most efficient use of the error correction capabilities in PDF417.

The TAL Bar Code control allows complete control over all features of PDF417.

See Also

PDF417 Symbology Description (pg. 95)
PDF417 Error Detection and Correction (pg. 98)

PDF417 Bar Code Dimensions

A PDF417 bar code symbol consists of multiple rows of data encoded in units called code words. Each symbol can contain from 3 to 90 rows and each row consists of a Start/Stop pattern and from 3 to 32 code words (1 to 30 code words for data and 2 for Right and Left Row Indicators). The smallest element in a PDF417 symbol is called a module. Each code word consists of a unique pattern of 4 bars and 4 spaces each with a width of up to 6 modules within a total width of 17 modules per code word. This is where the 417 comes from in the name PDF417 - 4 bars within 17 modules.

Because the number of code words in a row and the total number of rows in a symbol are variable quantities, the three primary dimensions used to define the size of a PDF417 bar code are the **PDFModuleWidth**, **PDFModuleHeight**, and the **PDFAspectRatio** properties. The **PDFModuleHeight** and **PDFModuleWidth** values define the height and width of the smallest rectangular element in the PDF417 symbol and the **PDFAspectRatio** value specifies the overall height to width ratio of the symbol.

The best choice for the **PDFModuleWidth** property depends partly on the resolution of your bar code reading equipment and also on the resolution of the printer being used to produce the bar code. The specification for PDF417 recommends that the module width should fall in a range between 10 and 30 mils. The smallest allowable module width defined in the symbology specification is 6.56 mils. (This translates to 2 printer dots when printing to a 300 DPI laser printer.) The best way to determine the ideal module width for your application is to actually print out a sample bar code using several different values and try reading each one with your scanning equipment. Again, you should choose the value that produces bar codes with the best read rate.

The recommended value for the **PDFModuleHeight** is three times the value for the **PDFModuleWidth** however the symbol specifications allow for module heights as small as 10 mils (3 printer dots on a 300 DPI laser printer).

The value for the **PDFAspectRatio** property determines the overall shape of the PDF417 symbol and is defined as the overall height to width ratio. Higher values for the **PDFAspectRatio** (greater than 1) produce tall, thin PDF417 bar codes and small values (greater than zero and less than 1) produce short, wide bar codes. A value of 1 should produce approximately square bar codes. Because each row of data in a PDF417 symbol contains 2 additional code words of overhead (right and left row indicators) as well as start and stop patterns, tall and thin bar codes with lots of rows (high aspect ratios) will be larger in total area and contain more overhead than short wide bar codes (low aspect ratios). On the other hand, short wide bar codes (low aspect ratios) may be difficult to read depending on the type of scanner that you use.

NOTE: For any given PDF417 bar code symbol there are only a small number of possible aspect ratios that are physically possible. Because of this, you will probably not be able to produce a bar code with the exact **PDFAspectRatio** that you specify. The TAL Bar Code control will automatically generate a bar code with the closest match to the **PDFAspectRatio** value that you specify, within the limits of the symbol specification.

The TAL Bar Code ActiveX control also provides more detailed control over the size and shape of a PDF417 symbol by allowing you to specify the maximum number of rows and a maximum number of data code word columns in a symbol using the **PDFMaxRows** and **PDFMaxCols** properties. The default maximum number of data rows and columns are the maximum values allowable in the PDF417 symbol specification (i.e. 30 rows and 90 Columns) . If you specify smaller values for these parameters, you are essentially defining an upper limit to the overall height or width of all generated PDF symbols. For example, suppose you need to generate a PDF symbol that absolutely must be no wider than two inches and you chose 10 mils for the **PDFModuleWidth** property. Because there are 17 modules in a code word and there are 69 modules of overhead per row, (start/stop patterns and right/left row indicators) the maximum number of code word columns allowable in a 2 inch wide symbol can be calculated using the formula:

10mils x [(17 x PDFMaxCols) + 69]<=2000 mils (2 inches)

If we solve the equation above we obtain the value of 7 for the Maximum number of code word columns (**PDFMaxCols**). Thus, if we set the Maximum Number of Data Columns parameter to 7, the TAL Bar Code control will only produce PDF417 symbols that are less than 2 inches wide.

You can also cause the TAL Bar Code control to produce a version of PDF417 called Truncated PDF417 that removes the Right Row Indicator code words and the stop pattern on the right hand side of the symbol thus reducing the total size of a PDF417 symbol. When creating Truncated PDF417 symbols, there are only 35 overhead modules per row instead of the normal 69 overhead modules. This option can be selected by setting the **PDFTruncatedSymbol** property to True.

See Also

PDF417 Symbology Description (pg. 95)
PDF417 Error Detection and Correction (pg. 98)

PDF417 Error Detection and Correction

One of PDF417's primary features is error detection and correction or data security. Each PDF417 symbol has 2 code words for error detection. (Similar to a check digit in standard bar code symbols.) The error correction capacity may be selected by the user, based on the needs of a specific application. Error correction compensates for label defects and misdecodes. There are essentially 2 types of errors that can occur in a bar code symbol, Erasures and Misdecodes. Erasures are missing or undecodable code words and misdecodes are errors that cause the reader to interpret a particular code word incorrectly. Nine error correction levels numbered 0 through 8 are available in the symbol specification. Each higher security level allows for a higher number of erasures and misdecodes to be recovered from. (Since it requires 2 code words to recover from a misdecode, one to detect the error and one to correct for it, a given security level can support half the number of misdecodes that it can undecoded or missing code words.)

Since error correction is encoded into additional code words, each higher security level adds additional overhead to a printed symbol. Higher security levels reduce the maximum number of data characters that can be encoded and also increase the size of a printed symbol. (A PDF417 symbol can contain a maximum of 928 total code words for data and error correction combined.)

The relationship between security level, error correction capacity and the number of additional code words or overhead required for a given security level is as follows:

Error Correction Level	Maximum Limit of Allowable Erasures + (2 x Misdecodes)	Symbol Overhead (Number of Additional Code words)
0	0	2
1	2	4
2	6	8
3	14	16
4	30	32
5	62	64
6	126	128
7	254	256
8	510	512

The **PDFSecurityLevel** property allows you to select a specific PDF417 error correction level from 0 to 8. You can also have the TAL Bar Code control automatically select a security level based on a percentage of total symbol area to be used for error correction. If you set the **PDFSecurityLevel** property to 9, you can then pass a percentage value in the **PDFPctOverhead** property (from 0% to 99%). This "automatic" **PDFSecurityLevel** mode is the best way to specify a security level because it guarantees that you will not waste symbol real estate with more error correction overhead than is necessary for small messages. It also insures that enough error correction will be generated for larger messages - A small message only needs a small amount of error correction capacity while a larger message needs more. If you set the value for the **PDFPctOverhead** property to zero, the default percentage of 10% overhead will be used.

See Also

PDF417 Symbology Description (pg. 95)
PDF417 Bar Code Dimensions (pg. 96)

Aztec Code

Aztec Code is a high density 2 dimensional matrix style symbology that can encode up to 3750 characters from the entire 256 byte ASCII character set. The symbol is built on a square grid with a bulls-eye pattern at its center. Data is encoded in a series of "layers" that circle around the bulls-eye pattern. Each additional layer completely surrounds the previous layer thus causing the symbol to grow in size as more data is encoded. Aztec's primary features include: a wide range of sizes allowing both small and large messages to be encoded, orientation independent scanning and a user selectable error checking and correction mechanism.

The smallest element in an Aztec symbol is called a "module" (i.e. a square dot). The module size and the amount of error correction are the only "dimensions" that can be specified for an Aztec symbol and both are user selectable. It is recommended that the module size should range between 15 to 30 mils in order to be readable by most of the scanners that are currently available. The module size is specified using the **MatrixModuleSize** property.

The overall size of an Aztec symbol is dependent on the **MatrixModuleSize** setting, the total amount of encoded data and also on the level of error correction capacity chosen by the user. The smallest Aztec symbol is 15 modules square and can encode up to 14 digits with 40% error correction. The largest symbol is 151 modules square and can encode 3000 characters or 3750 numeric digits with 25% error correction.

There are four types of Aztec symbols, “Normal”, "Compact", "Full Range" and "Menu" symbols. You specify the type of Aztec symbol to generate using the **AztecSymbolType** property. Normal Aztec symbols can grow or shrink in size depending on the amount of data that you are encoding. With Normal Aztec symbols you also specify the amount of error correction overhead to include by specifying the percentage of the total symbol area to use for error correction. The **AztecPctECC** property is used for this purpose. Compact symbols have a smaller bulls-eye pattern and are limited in their overall size to having up to four "layers" of data surrounding the bulls-eye pattern. Full Range symbols have a larger bulls-eye pattern and can have up to 32 layers of data surrounding the bulls-eye. Do not confuse "Layers" with modules. A layer actually is made up of two stacked modules resembling a domino with each domino laid out around the bulls-eye pattern so that the long edge of the domino points away from the center of the symbol. If you set the TAL Bar Code control to generate Compact or Full Range Aztec symbols, you must also specify the total number of data layers in the symbol using the **AztecTotalLayers** property. If the total number of data layers is greater than the amount of layers required to encode a particular message, the remaining layers are filled with error correction data. If the amount of data that you need to encode will not fit in the number of data layers that you specify, an error event will be raised and no bar code symbol will be generated. Menu symbols are a special type of Aztec bar code that are typically used by scanner manufacturers to create bar codes that contain commands for enabling and disabling features in a bar code reader. Menu symbols can be either Compact or Full Range symbols and are only useful if you know the commands that a particular reader will recognize. The TAL Bar Code control does not provide support for Aztec Menu symbols.

Data Matrix

Data Matrix is a high density 2 dimensional matrix style bar code symbology that can encode up to 3116 characters from the entire 256 byte ASCII character set. The symbol is built on a square grid arranged with a finder pattern around the perimeter of the bar code symbol.

There are two types of Data Matrix symbols each using a different error checking and correction scheme (ECC). The different types of Data Matrix symbols are identified using the terminology "ECC" followed by a number representing the type of error correction that is used by the encoding software. ECC 000 to ECC 140 are the original type of Data Matrix symbols and are now considered obsolete. The newest version of Data Matrix is called ECC 200 and is recommended for all new Data Matrix applications. The ECC 200 version of Data Matrix uses a much more efficient algorithm for encoding data in a symbol as well as an advanced error checking and correction scheme. The TAL Bar Code control fully supports all variations of the Data Matrix symbology however the author of the original symbology specification (CI Matrix Co.) highly recommends that ECC 000 - ECC 140 be used only where absolutely necessary to remain compatible with older systems.

The TAL Bar Code control allows you to select the ECC value using the **DataMatrixECC** property. Each of the different ECC options are outlined below:

ECC 000 - ECC 140

Data Matrix symbols designated by the terms ECC 000 to ECC 140 are the original type of Data Matrix symbol that uses a convolutional error correction scheme. There are actually five levels of error correction available for this type of Data Matrix symbol with each higher level of error correction designated as follows.

ECC 000	Provides no error correction
ECC 050	Provides error correction for damage of up to 2.8% of the printed symbol.
ECC 080	Provides error correction for damage of up to 5.5 % of the printed symbol.
ECC 100	Provides error correction for damage of up to 12.6% of the printed symbol.
ECC 140	Provides error correction for damage of up to 25% of the printed symbol.

For all five ECC levels ECC 000 - ECC 140 there is also a user selectable option for a "Data Format" which defines the type of data that may be encoded in a Data Matrix symbol. The data format is selected using the **DataMatrixFormatID** property. The available formats are listed below:

Format ID	**Allowable Data In The Bar Code Message**
1	Numeric digits 0 to 9 and the space character
2	Upper case alpha A-Z and the space character
3	Upper case alphanumeric A-Z, 0-9 and the space character
4	A-Z, 0-9, space, minus, period, comma & forward slash (/)
5	7 bit ASCII - all ASCII characters between ASCII 0 to ASCII 127
6	8 bit ASCII - all ASCII characters between ASCII 0 to ASCII 255

ECC200
ECC 200 is the latest and most advanced version of Data Matrix and is therefore strongly recommended for use in all new Data Matrix applications. ECC 200 uses a Reed-Solomon error correction algorithm that will automatically provide a varying degree of error correction for damage ranging from a minimum of roughly 20% to greater than 60% damage depending on the amount of data encoded ECC200 also uses a code set switching mechanism that is much more efficient at packing data into a symbol than any of the earlier encoding schemes (ECC 000 - ECC 140). ECC 200 is capable of encoding the entire 8 bit ASCII character set in a highly efficient manner and therefore does not provide for or require any Data Format options. In addition to encoding 8 bit ASCII data, ECC 200 also allows for a special function code called "FNC1" as well as special indicators for features like "Structured Append" and "Extended Channel Interpretation".

Structured append is a means for generating multiple bar codes containing a much larger data message than can be encoded into a single symbol. With structured append, each bar code contains a portion of a larger data message along with a number that identifies the portion of the message contained in the symbol. When a bar code reader scans a symbol that is part of a sequence, it will not transmit the data until all symbols in a sequence have been read. It does not matter what order the bar codes are read in - the reader will correctly build the complete message.

Extended Channel Interpretation is a mechanism for creating user definable data encoding schemes. For example, suppose you wanted to replace the standard ASCII character set with a different character set (a foreign language character set for example), you could use the Extended Channel Interpretation feature in ECC 200 to indicate that the message encoded in a symbol conforms to the new interpretation. Notes: To encode data to conform to specific industry standard it needs to be authorized by AIM International.

The TAL Bar Code control supports all features of the ECC 200 version of Data Matrix. Because ECC 200 supports the entire 8 bit ASCII character set and therefore cannot use ASCII character values to represent special features (like FNC1), the control provides two methods for interpreting the input data message. The control provides a **DataMatrixTildeCodes** property that determines how the input data message is interpreted. If the **DataMatrixTildeCodes** property is set to False then the **Message** property is assumed to not contain any special function codes like FNC1 or Extended Channel Interpretation codes. For most normal applications this option would normally be used.

If the **DataMatrixTildeCodes** property is set to True then the tilde character (~) can be used in the input message as an indicator that the character(s) following the tilde are to be interpreted with a special meaning as outlined below. To encode a tilde (~) use the string: ~~ (i.e. two tilde characters). If no tilde characters or Nulls (ASCII 0) are present in the input message, then setting the **DataMatrixTildeCodes** property to True has no effect on the resulting bar code symbol.

Tilde Control Codes
~X (a tilde character followed by any upper case alpha character) is used as a shift character for inserting control codes (characters with ASCII values 0 to 26) into a bar code message. For example, ~@ = NUL, ~A = ASCII 1,~G = BEL (ASCII 7), ~M = ASCII 13 (carriage return). If you need to insert ASCII control codes into a message, take the ASCII value for the control code (1-26) and find the corresponding letter in the alphabet and precede it with a tilde. i.e. The ASCII value for a carriage return character is ASCII 13 and the thirteenth letter of the alphabet is "M" therefore to insert a carriage return in a bar code message, you would use "~M". Note: You can also pass control codes directly to the DLL without having to use the ~ before an alpha character. For example you could use either an ASCII 13 character or the sequence ~M to represent a carriage return.

~1 is used to represent the FNC1 code and is followed by normal data.
To encode data to conform to specific industry standards as authorized by AIM International, a FNC1 character shall appear in the first or second symbol character position (or fifth or sixth data position of the first symbol of structured append). FNC1 encoded in any other position is used as a field separator and shall be transmitted as a GS control character (ASCII value 29).
Notes: To encode data to conform to specific industry standard, it needs to be authorized by AIM International. Contact AIM International at Tel: 703-391-7621 or email: adc@aimi.org
If the FNC1 code is used in the second character position, the input data before '~1' must be, between 'A' and 'Z', or between 'a' and 'z' or 2-digits between '01' and '99'.

~2 : is used to represent Structured Append and must be followed by a three 3-digit number between 1 and 255 representing the symbol sequence as well as a file identifier of six numeric digits. The file identifier is used to uniquely identify a sequence so that only logically linked sequences are processed as part of the same sequence. The symbol sequence identifier is a number between 1 and 255 that indicates the position of the symbol within a sequence of up to 16 symbols. The sequence identifier actually contains two four bit values representing the sequence number and the total number of symbols in the sequence (i.e. m of n where m is the sequence number and n is the total number of symbols). The upper four bits of this value represent the position of the particular symbol as the binary value of (m-1) and the lower order four bits identify the total number of symbols to be concatenated as the binary value of (17-n). For example, symbol 3 in a sequence of 7 symbols with file ID: 001015 is represented by ~2042001015. The number 042 is derived as follows: 3-1=2 which equals 0010 when represented as a 4 bit binary number. 17-7=10, which equals 1010, when represented as a 4 bit binary number. After concatenating the two 4 bit binary values we end up with 00101010 which equals 42 in decimal.

~3 : Indicates that a message is to be used for reader programming purposes and is followed by normal data. This feature is only useful if you know the specific programming commands for your bar code reader.

~5 and **~6** : indicates that the data will contain an abbreviated format header and trailer followed by normal data. The ~5 or ~6 must appear as the first two characters in a message and must not be used in conjunction with structured append. Data Matrix provides a means of abbreviating an industry specific header and trailer in one symbol character. This feature exists to reduce the number of characters needed to encode data using certain structured formats. If a ~5 is used as the first two characters of a message, the header []> + ASCII 30 + 05 + ASCII 29 will be transmitted by the reader before the data in the message and the trailer ASCII 30 + ASCII 4 will be transmitted following the data. Likewise, if a ~6 is used as the first two characters of a message, the header []>+ ASCII 30 + 06 + ASCII 29 will be transmitted by the reader before the data in the message and the trailer ASCII 30 + ASCII 4 will be transmitted following the data.

~7NNNNNN is used to indicate Extended Channel NNNNNN where NNNNNN is 6-digit EC value (000000 - 999999). e.g. Extended Channel 9 is represented by ~7000009

~dNNN creates ASCII decimal value NNN for a codeword (must be 3 digits).
Please refer to the official Data Matrix symbology specification for details on the meanings of all codeword values for ECC 200. Contact AIM International at Tel: 703-391-7621 or email: adc@aimi.org

Rectangular Data Matrix Symbols

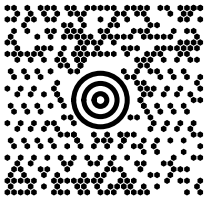
The ECC 200 version of Data Matrix supports the ability to produce rectangular bar codes.
The TAL Bar Code ActiveX Control uses the **DataMatrixFormatID** property to indicate the overall shape of all ECC200 bar codes. Six options are available that force the bar code symbol to be both fixed in size and rectangular in shape. The overall size of the rectangular symbols is dependent on the size of the module that you have selected for the **MatrixModuleSize** property. The table below shows the 6 different rectangular symbol options available along with size in modules and the amount of data that may be encoded for each option.

Format ID	Shape of bar code and amount of data that may be encoded
1 - 6	Square - symbols grow in size depending on amount of data being encoded.
7	Rectangular - 8 modules tall by 18 modules wide - may contain up to 10 bytes
8	Rectangular - 8 modules tall by 32 modules wide - may contain up to 20 bytes
9	Rectangular -12 modules tall by 26 modules wide - may contain up to 32 bytes
10	Rectangular -12 modules tall by 36 modules wide - may contain up to 44 bytes
11	Rectangular -16 modules tall by 36 modules wide - may contain up to 64 bytes
12	Rectangular -16 modules tall by 48 modules wide - may contain up to 98 bytes

See Also

The DataMatrixTildeCodes Property (pg. 29)

MaxiCode



MaxiCode is a fixed size matrix style symbology made up of rows of hexagonal modules arranged around a bulls-eye finder pattern. A MaxiCode symbol has 884 hexagonal modules arranged in 33 rows with each row containing up to 30 modules. The maximum data capacity for a MaxiCode symbol is 93 Alphanumeric or 138 Numeric characters. The symbology was designed by United Parcel Service for package tracking applications. The design of the MaxiCode symbology was chosen because it is well suited to high speed, orientation independent scanning. Although the capacity of a MaxiCode symbol is not as high as other matrix style symbologies, it was primarily designed to encode address data, which rarely requires more than 80 characters. MaxiCode symbols actually encode two messages; a Primary and a Secondary message. The Primary message normally encodes a postal code, a 3 digit country code and a 3 digit class of service number. The Secondary message normally encodes an address and any other required data.

MaxiCode Modes:

MaxiCode has several "Modes" that determine how data is encoded in the symbol. The original MaxiCode specification supported modes 0 and 1, which are now obsolete and are therefore not supported by the TAL Bar Code control. The mode is selected using the **MaxiCodeMode** property. The current specification supports the following Modes:

Mode 2	Structured Carrier Message – Numeric Postal Codes (up to 9 digits)
Mode 3	Structured Carrier Message – AlphaNumeric Postal Codes (up to 6 characters)
Mode 4	Standard Symbol – Standard Error Correction
Mode 5	Standard Symbol – Enhanced Error Correction (not supported)
Mode 6	Reader Programming Mode

Modes 2 and 3 are reserved for use as a destination sortation symbol for use by carriers in the transportation industry. In Modes 2 and 3 a postal code, a country code and a service class number must be supplied along with a secondary message usually consisting of an address.

Mode 4 is designed to encode data as a "standard bar code symbol" where the data encoded in the symbol is not restricted to a specific application. In other words, Mode 4 should be used in all general purpose bar code applications other than transportation industry applications. Mode 5 is similar to mode 4 except that a higher level of error correction is employed. Because more symbol real estate us used for error correction in Mode 5, the amount of actual data that can be encoded in a Mode 5 symbol is reduced. The TAL Bar Code control does not support Mode 5.

Mode 6 is reserved for bar code reader programming purposes and it has been left up to the bar code reader manufacturers to determine how to interpret data encoded using Mode 6.

Structured Carrier Messages in Mode 2 and 3 MaxiCode Symbols:

The TAL Bar Code control provides input properties for each of the three required Primary message fields (Postal Code, Country Code and Service Class) as well as a property for the secondary message (the Message property). United Parcel Service has defined a special standard for formatting data in a MaxiCode symbol for use in UPS package tracking applications. The standard defines several things including the content and format of the data in the primary and secondary messages, a special sequence of "header" characters, and the specific characters to be used as delimiters or format codes and application identifiers. For complete details on how to encode Structured Carrier Messages conforming to the UPS standards, please contact your local United Parcel Service representative or visit the UPS web site at http://www.maxicode.com.
The TAL Bar Code control does not attempt to format the data in any way except in a special case as outlined below. In other words, when you use the TAL Bar Code control to produce a Mode 2 or Mode 3 bar code, you must provide all required data for the primary message and the data in the secondary message must contain all required header, delimiter and special formatting characters.

As a convenience to the programmer, the TAL Bar Code control will automatically parse out a Postal Code, the Country Code and the Service Class from any message that is passed in as a structured carrier message starting with the character sequence: []>$^{R}_{S}$01 $^{G}_{S}$ yy. In other words, if the input message (passed in the Message property) to the control starts with the string: []>$^{R}_{S}$ 01 $^{G}_{S}$ yyaaaaaaaaa $^{G}_{S}$ bbb $^{G}_{S}$ ccc $^{G}_{S}$ where yy is a two digit year, aaaaaaaaa is a valid postal code, bbb is a three digit country code and ccc is a class value, the TAL Bar Code control will automatically set the Postal Code, Country Code and Service Class parameters for the primary message to the values embedded in the input message. These values will also be removed from the secondary message.
(Note: The $^{R}_{S}$ and $^{G}_{S}$ symbols used here represent ASCII characters 30 and 29 respectively.)

For example, if the following message is supplied in the **Message** property, the Postal Code, Country Code and Service Class will be extracted from the Message and placed in the **MaxiZipCode**, **MaxiCountryCode** and **MaxiCodeClass** properties automatically so that you do not need to fill in these properties in advance.

[]>$^{R}_{S}$01$^{G}_{S}$98152382802 $^{G}_{S}$ 840GS001$^{G}_{S}$1Z00004951$^{G}_{S}$UPSN$^{G}_{S}$06X610$^{G}_{S}$159$^{G}_{S}$1234567$^{G}_{S}$ 1/1$^{G}_{S}$$^{G}_{S}Y^{G}_{S}$634 ALPHA DRIVEGSPITTSBURGHGSPA$^{R}_{S}$$^{E}_{OT}$

If the above message is supplied in the **Message** property with the **MaxiCodeMode** property set to 2 or 3, the Postal Code value "152382802" and the Country Code value "840" and the Service Class value "001" will be removed from the secondary message and automatically encoded in the primary message. The actual message that will be encoded in the secondary message will be:

[]>$^{R}_{S}$01 $^{G}_{S}$981Z00004951 $^{G}_{S}$ UPSN $^{G}_{S}$ 06X610 $^{G}_{S}$ 159 $^{G}_{S}$ 1234567 $^{G}_{S}$ 1/1 $^{G}_{S}$ $^{G}_{S}$ Y $^{G}_{S}$ 634 ALPHA DRIVEGSPITTSBURGHGSPA $^{R}_{S}$ $^{E}_{OT}$

Note: In Mode 2, the Postal Code parameter must be either a 5 or 9 digit number (all numeric characters) and in Mode 3, the Postal Code may contain up to 6 Alphanumeric characters.

RSS14

RSS14 (**R**educed **S**pace **S**ymbology) is a relatively new symbology that encodes up to 14 numeric digits (13 digits plus a check digit) along with an optional two-dimensional “Composite” bar code component that can encode up to 338 additional bytes of alpha/numeric data. The principle use of the RSS symbology is to identify items that cannot be marked with current linear symbols because of size restrictions. The Composite Symbols provide additional supply chain data while allowing for the co-existence of symbologies already being used.

To include a composite message in a RSS14 bar code, append the composite message to the main bar code message in the Message property of the TAL Bar Code ActiveX control with a comma or a pipe character (|) between the two messages. For example, if you specify “1234567890123,Testing 123” for the Message property of the control, the string “1234567890123” will be encoded in the main RSS14 symbol and the message “Testing 123” will be encoded in the composite portion of the bar code symbol.

The recommended Bar Height for standard RSS bar code symbols is .43 inches and the recommended Narrow Bar Width is 13 mils.

For more information on the RSS14 symbology, visit: http://www.uc-council.org