2D Barcode VCL Components

User Manual

Version: 13.3.0.2668

Copyright © 2001-2025 Han-soft Corporation. All rights reserved.

2025-04-06

Chapter 1. Introduction

1.1 Overview

2D Barcode VCL Components is the most flexible and powerful VCL components package which lets you to easily add advanced 2D barcode generation and printing features to your application.

2D Barcode VCL Components supports most popular Matrix and Stacked 2D Barcode Symbologies/Standards, including Aztec Code, Aztec Runes, Code 16K, Data Matrix (ECC 000-140, ECC 200), MaxiCode, PDF417, QR Code 2005, Micro QR Cod MicroPDF417.

All RSS barcode symbologies are supported, including RSS-14, RSS-14 Truncated, RSS-14 Stacked, RSS-14 Stacke Omnidirectional, RSS Limited, RSS Expanded, and RSS Expanded Stacked.

The EAN.UCC composite symbologies are supported.

All morden versions of Delphi and C++ Builder are supported, including the Delphi 4, 5, 6, 7, 2005, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens and C++ Builder 4, 5, 6, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens.

Both 32-bit and 64-bit components are included, and the RAD Studio 64-bits IDE is supported.

2D Barcode VCL Components are easy to use. Developers use them like any other VCL component.

1.2 Main features

- Allows to draw the barcode symbol to canvas (with scaling and rotating).
- Allows to print the barcode symbol to paper (with scaling and rotating).
- Ability to save barcode symbol in a variety of picture formats.
- Ability to encode the data block into a barcode symbol.
- Most popular matrix and stacked 2 dimensional barcode symbologies are supported.
- All RSS barcode symbologies are supported.
- Ability to create the EAN.UCC composite barcode symbols.
- All of the Quick Report, Fast Report, Report Builder, Rave Reports and ACE Reporter are supported.
- The database function is supported, including the data access components (such as the FireDAC, dbExpress, BDE) and the LiveBindings.
- The Delphi 4, 5, 6, 7, 2005, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens and C++ Builder 4, 5, 6, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens are

supported.

- Both 32-bit and 64-bit components are included, and the RAD Studio 64-bits IDE is supported.
- Structured append, ECI, etc. features are supported.
- Ability to add a logo picture to the barcode symbol.
- It is visible in design time.
- Ability to scale and rotate the barcode symbols.
- Foreground and background colors of barcode symbol can be changed freely.
- It's easy to use, and it has the excellent functionality.
- It's a very popular 2D barcode components package.

Chapter 2. Installation

2.1 Trial user

Installation step by step:

1. Before installing the components package, please close all Delphi and C++ Builder IDEs.

   Note, for each IDE, if it's a clean installation, please run it at least once before installing the components package, then closes it and continues installation.

2. Run the installation file **barcode2d_tri.exe**, and then click on the "Next" button in the installation dialog box.
3. Read the **End-User License Agreement** You must accept the terms of this agreement before continuing with the installation. And then click on the "Next" button.
4. Specify a target folder (it will be created if does not exist), the components package will be installed into it. And then click on the "Next" button.
5. All supported Delphi and C++ Builder IDEs will be listed automatically according on the existing IDEs in your computer. Please select the IDEs you want to install to them. And then click on the "Next" button.

   Note, The Delphi 4, 5, 6, 7, 2005, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10. Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens, and C++ Builder 4, 5, 6, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria 12.3 Athens are supported now. The Delphi 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 1(Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens and C++ Builder 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens are listed as RAD Studio (Delphi & C++ Builder) 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4 XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens.

6. Specify a shortcuts folder in "Start Menu" for the components package. And then click on the "Next" button. Later, you can open the manual or remove the components package in the shortcuts folder.
7. Click on the "Install" button to complete the components package installation.
8. Click on the "Finish" button to close the installation dialog box.
9. You can start your IDE to use the components package now.

Note:

- If multi-user accounts want to use the components package, please install it into **the same folder** in each user session.
- Both 32-bit and 64-bit components are included, and the RAD Studio 64-bits IDE is supported.

2.2 Registered user

Installation step by step:

1. Before installing the components package, please close all Delphi and C++ Builder IDEs.

   Note, for each IDE, if it's a clean installation, please run it at least once before installing the components package, then closes it and continues installation.

2. Please uninstall the trial release using the "Uninstall" shortcuts in the "Start Menu" if it is installed in your computer.
3. Please download the installation package using the download link that's sent from us after you purchase the components package. If your download link doesn't work, please visit the "Manage your licenses" page to request a new download link. Please open the page then enter your order ID, license user name or license e-mail address to locate your order, then click on the oder ID to display it, choose a license and click on the "**Request a new download link**", the new download link will be sent to this license e-mail address automatically.
4. Run the installation file **barcode2d_ful.exe**, and then click on the "Next" button in the installation dialog box.
5. Read the **End-User License Agreement** You must accept the terms of this agreement before continuing with the installation. And then click on the "Next" button.
6. Type your email address and the license key that they are sent from us after you purchase the components package, they are not case-sensitive. And then click on the "Next" button. If you forgot the license key, please visit the "Manage your licenses" page to retrieve it. Please open the page then enter your order ID, license user name or license e-mail address to locate your order, then click on the oder ID to display it, choose a license and click on the "**Retrieve the license key**", the license key will be sent to the license e-mail address automatically.
7. Specify a target folder (it will be created if does not exist), the components package will be installed into it. And then click on the "Next" button.
8. All supported Delphi and C++ Builder IDEs will be listed automatically according on the existing IDEs in your computer. Please select the IDEs you want to install to them. And then click on the "Next" button.

   Note, The Delphi 4, 5, 6, 7, 2005, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10. Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens, and C++ Builder 4, 5, 6, 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria 12.3 Athens are supported now. The Delphi 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 1(Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens, and C++ Builder 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4, XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens are listed as RAD Studio (Delphi & C++ Builder) 2006, 2007, 2009, 2010, XE, XE2, XE3, XE4 XE5, XE6, XE7, XE8, 10 Seattle, 10.1 Berlin, 10.2 Tokyo, 10.3 Rio, 10.4 Sydney, 11.3 Alexandria, 12.3 Athens.

9. Specify a shortcuts folder in "Start Menu" for the components package. And then click on the "Next" button. Later, you can open the manual or remove the components package in the shortcuts folder.
10. Click on the "Install" button to complete the components package installation.
11. Click on the "Finish" button to close the installation dialog box.
12. You can start your IDE to use the components package now.

Note:

- If multi-user accounts want to use the components package, please install it into **the same folder** in each user session.
- After installation, please delete all ".dcu" files in your projects that they are built using the trial release of the components package, then re-build these projects.
- Both 32-bit and 64-bit components are included, and the RAD Studio 64-bits IDE is supported.

Chapter 3. Quick start

3.1 How to use the barcode components

Usage:

1. Put a barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form.
2. Put a TImage control to your form.
3. Set the Image property of the barcode component to the TImage control.

   You can link single TImage control to multiple TBarcode2D components in order to display multiple barcode symbols in the TImage control (using the LeftMargin and TopMargin properties to specify the position for every barcode symbol).

Note:

If the barcode symbol isn't displayed, please check whether the length of barcode text exceeds the maximum length limit, or whether there is any invalid character in the barcode text.

You can create the OnInvalidLength and OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event handles to catch the invalid barcode length exception. And create the OnInvalidChar and OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event handles to catch the invalid character in the barcode text.

Also, please check whether the TImage control is large enough to accommodate entire barcode symbol.

3.2 How to use the barcode components with a database

Use the classic data access components such as BDE, dbExpress, FireDAC, AnyDAC, etc.

1. Put a TBarcode2D barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form.
2. Put a TDBBarcode2D component to your form.
3. Set the Barcode2D property of the TDBBarcode2D component to your TBarcode2D barcode component.
4. Set the DataSource property of the TDBBarcode2D component to your TDataSource component.
5. Set the DataField property of the TDBBarcode2D component to a field in your dataset.
6. If you use the Delphi/C++ Builder 2009 or later, Set theBindProperty property of the TDBBarcode2D component to indicate which property of the TBarcode2D component the data field value will be applied to.
7. If you want to represent the barcode symbol in form or QuickReport report, put a TImage control to your form, or put a TQRImage or TQRGzImage control to your report. Set the Image property of the TBarcode2D barcode component to the

TImage, TQRImage, or TQRGzImage control. The data in the data field will be represented as a barcode symbol in your TImage, TQRImage, or TQRGzImage control.

8. If you want to save the barcode symbol to a picture, put a TSave2D component, such as the TSave2D_Bmp, TSave2D_Png, and TSave2D_Svg to your form. Set the Barcode2D property of the TSave2D component to your TBarcode2D barcode component. Then you can use the Save method of the TSave2D component to save the barcode symbol to a picture file.

9. Also, you can use the Print method of the TBarcode2D component to print the barcode symbol to paper, or use the DrawTo method of the TBarcode2D component to draw the barcode symbol to any TCanvas.

Note, You can bind multiple TDBBarcode2D and TBarcode2D component pairs to a data field in order to represent the data field with multiple barcode symbols.

Use the LiveBindings (Delphi/C++ Builder XE2 or later).

1. Put a TBarcode2D barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form.

2. Open the "LiveBindings Designer" (right-click on the form then execute the "Bind visually..." menu item), click on the barcode component in the form to select it, change the "Visible Element" sub-item of the "LiveBindings Designer" item to true in the "Object Inspector".

3. Right-click on the barcode component in the "LiveBindings Designer", execute the "Bindable Members..." menu item, check the "Barcode" property or the "Data" property (only for the Delphi/C++ Builder 2009 or later) in the "Bindable Members" dialog, click on the "OK" button to close it.

4. Link the Barcode property or the Data property (only for the Delphi/C++ Builder 2009 or later) of the TBarcode2D barcode component to your data field in the TBindSourceDB component, or a string property of other control.

5. If you want to represent the barcode symbol in form or QuickReport report, put a TImage control to your form, or put a TQRImage or TQRGzImage control to your report. Set the Image property of the TBarcode2D barcode component to the TImage, TQRImage, or TQRGzImage control. The data in the data field will be represented as a barcode symbol in your TImage, TQRImage, or TQRGzImage control.

6. If you want to save the barcode symbol to a picture, put a TSave2D component, such as the TSave2D_Bmp, TSave2D_Png, and TSave2D_Svg to your form. Set the Barcode2D property of the TSave2D component to your barcode component. Then you can use the Save method of the TSave2D component to save the barcode symbol to a picture file.

7. Also, you can use the Print method of the TBarcode2D component to print the barcode symbol to paper, or use the DrawTo method of the TBarcode2D component to draw the barcode symbol to any TCanvas.

Note, You can link multiple TBarcode2D components to a data field in order to represent the data field with multiple barcode symbols.

3.3 How to use the barcode components with QuickReport

Usage:

1. Put a barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form.

   Also, put a TDBBarcode2D component to the form and link the TBarcode2D component to the TDBBarcode2D component if the database support is required.

2. Put a TQRImage or TQRGzImage control to your report.

3. Set the Image property of the barcode component to the TQRImage or TQRGzImage control.

   You can link single TQRImage or TQRGzImage control to multiple TBarcode2D components in order to display multiple barcode symbols in the TQRImage or TQRGzImage control (using the LeftMargin and TopMargin properties to specify the position for every barcode symbol).

Note:

If the barcode symbol cannot be read, please don't reduce/stretch the barcode symbol (set the Stretch property to false). You can change the barcode symbol size by changing its Module property value.

Also, please check whether the TQRImage or TQRGzImage control is large enough to accommodate entire barcode symbol.

3.4 How to use the barcode components with FastReport

Usage:

1. Edit your report, put a TfrxPictureView control to your report in order to present the barcode symbol.
2. Put a barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form that the TfrxReport component is in it.

   Also, put a TDBBarcode2D component to the form and link the TBarcode2D component to the TDBBarcode2D component if the database support is required.
3. Create the OnBeforePrint event function for the TfrxReport component.

   In the event function, change the properties of the TBarcode2D component such as Barcode, Module, and Orientation, and adjust the bitmap size of the TfrxPictureView control in order to accommodate entire barcode symbol, then use the DrawTo method of the TBarcode2D component to draw the barcode symbol to the TfrxPictureView control.

   For example:

```
var
  AWidth, AHeight, ASymbolWidth, ASymbolHeight: Integer;
begin
  Barcode2D_QRCode1.Barcode := '1235678';
  Barcode2D_QRCode1.Module := 1;
  ......
  Barcode2D_QRCode1.DrawToSize(AWidth, AHeight, ASymbolWidth, ASymbolHeight);
  with  TfrxPictureView(frxReport1.FindObject('Picture1')).Picture.Bitmap do
  begin
    Width := AWidth;
    Height := AHeight;
    Barcode2D_QRCode1.DrawTo(Canvas, 0, 0);
  end;
end;
```

   Note, if you use old FastReport 2.x, please use the OnBeginBand event.

3.5 How to use the barcode components with ReportBuilder

Usage:

1. Edit your report, put a TppImage control to your report in order to present the barcode symbol.
2. Put a barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form that the TppReport component is in it.

   Also, put a TDBBarcode2D component to the form and link the TBarcode2D component to the TDBBarcode2D component if the database support is required.
3. Create the OnBeforePrint event function for the TppReport component.

   In the event function, change the properties of the TBarcode2D component such as Barcode, Module, and Orientation, and adjust the bitmap size of the TppImage control in order to accommodate entire barcode symbol, then use the DrawTo method of the TBarcode2D component to draw the barcode symbol to the TppImage control.

   For example:

```
var
  AWidth, AHeight, ASymbolWidth, ASymbolHeight: Integer;
begin
  Barcode2D_QRCode1.Barcode := '1235678';
  Barcode2D_QRCode1.Module := 1;
  ......
  Barcode2D_QRCode1.DrawToSize(AWidth, AHeight, ASymbolWidth, ASymbolHeight);
  ppReport1Image1.Picture.Bitmap.Width := AWidth;
  ppReport1Image1.Picture.Bitmap.Height := AHeight;
  Barcode2D_QRCode1.DrawTo(ppReport1Image1.Picture.Bitmap.Canvas, 0, 0);
end;
```

3.6 How to use the barcode components with ACE Reporter

Usage:

1. Edit your report, put a TsctImageLabel control to your report in order to present the barcode symbol.
2. Put a barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form that the TsctReport component is in it.

   Also, put a TDBBarcode2D component to the form and link the TBarcode2D component to the TDBBarcode2D component if the database support is required.
3. Create the OnBeforePrint event function for the TsctImageLabel control.

   In the event function, change the properties of the TBarcode2D component such as Barcode, Module, and Orientation, and adjust the bitmap size of the TsctImageLabel control in order to accommodate entire barcode symbol, then use the DrawTo method of the TBarcode2D component to draw the barcode symbol to the TsctImageLabel control.

   For example:

```
var
  AWidth, AHeight, ASymbolWidth, ASymbolHeight: Integer;
```

```
begin
  Barcode2D_QRCode1.Barcode := '1235678';
  Barcode2D_QRCode1.Module := 1;
  ......
  Barcode2D_QRCode1.DrawToSize(AWidth, AHeight, ASymbolWidth, ASymbolHeight);
  with SctImageLabel1.Picture.Bitmap do
  begin
    Width := AWidth;
    Height := AHeight;
    Barcode2D_QRCode1.DrawTo(Canvas, 0, 0);
  end;
end;
```

3.7 How to use the barcode components with Rave Reports

Usage:

1. Use the Rave Reports Visual Designer to edit your report, put a bitmap component to your report in order to present the barcode symbol.
2. Put a TBarcode2D barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form that the TRvProject component is in it.

   Also, put a TDBBarcode2D component to the form and link the TBarcode2D component to the TDBBarcode2D component if the database support is required.

3. Insert code to call the Open method of the TRvProject component before print or preview the report, and call the Close method after print or preview the report.

   For example:

   ```
   RvProject1.Open;
   RvProject1.Execute;
   RvProject1.Close;
   ```

4. Add RvCsStd, RvProj, and RvClass units to the uses list.
5. Create the OnAfterOpen event function for the TRvProject component.

   In the event function, change the properties of the TBarcode2D component such as Barcode, Module, and Orientation. Then create a temporary TBitmap object, adjust its size in order to accommodate entire barcode symbol, and use the DrawTo method of the TBarcode2D component to draw the barcode symbol to the temporary TBitmap object. At last, adjust the size of the bitmap component in your report, and assign the temporary bitmap to it.

   For example:

   ```
   var
     RvReport: TRaveReport;
     RvPage : TRavePage;
     RvBitmap: TRaveBitmap;
     TmpBitmap: TBitmap;
     AWidth, AHeight, ASymbolWidth, ASymbolHeight: Integer;
     Scale: Integer;
   begin
     ......
   ```

```
  Barcode2D_QRCode1.Barcode := '1234567890';
  Barcode2D_QRCode1.Module := 2;
  ......
  with RvProject1.ProjMan do
  begin
    RvReport := FindRaveComponent('Report1', nil) as TRaveReport;
    RvPage := FindRaveComponent('Page1', RvReport) as TRavePage;
    RvBitmap := FindRaveComponent('Bitmap1', RvPage) as TRaveBitmap;
  end;
  TmpBitmap := TBitmap.Create;
  try
    Barcode2D_QRCode1.DrawToSize(AWidth, AHeight, ASymbolWidth, ASymbolHeight);
    TmpBitmap.Width := AWidth;
    TmpBitmap.Height := AHeight;
    Scale := 72;
    RvBitmap.Width := RvReport.XI2U(AWidth / Scale);
    RvBitmap.Height := RvReport.YI2U(AHeight / Scale);
    RvBitmap.MatchSide := msBoth;
    Barcode2D_QRCode1.DrawTo(TmpBitmap.Canvas, 0, 0);
    RvBitmap.Image.Assign(TmpBitmap);
  finally
    TmpBitmap.Free;
  end;
  ......
end;
```

3.8 How to generate the EAN.UCC composite barcode symbol

Usage:

1. Put a 2D barcode component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC to your form.

   Also, put a TDBBarcode2D component to the form and link the 2D barcode component to the TDBBarcode2D component if the database support is required.

2. Put a linear barcode component to your form. The TBarcode2D_RSS14, TBarcode2D_RSSLimited, and TBarcode2D_RSSExpanded can be used as the linear component. The TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, and TBarcode1D_EAN128 components can be used as linear barcode component too, they are in the **1D Barcode VCL Componnts**package. Note, only the TBarcode1D_EAN128 component can be used as the linear barcode component if you use the TBarcode2D_CCC as the 2D barcode component.

   Also, put a TDBBarcode2D component (the linear barcode component is the TBarcode2D_RSS14, TBarcode2D_RSSLimited, or TBarcode2D_RSSExpanded) or a TDBBarcode1D component (if the linear barcode component is the TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128) to the form and link the linear barcode component to it if the database support is required. The TDBBarcode1D component is in the **1D Barcode VCL Componnts** package.

3. Set the Linear property of the 2D barcode component to the linear barcode component.

4. Put a TImage control to your form, or put a TQRImage or TQRGzImage control to your report.

5. Set the Image property of the 2D barcode component to the TImage, TQRImage, or TQRGzImage control.

Note:

If you use the TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128 component as the linear barcode component, its Height property should be set to a value larger than zero.

The Image property of linear barcode component should not be normally set, the Image property's value of the 2D component will be used.

3.9 How to add a logo to barcode symbol

Usage:

1. Put a TBarcode2D barcode component, such as the TBarcode2D_QRCode, TBarcode2D_DataMatrixEcc200, and TBarcode2D_GridMatrix to your form.
2. Put a TLogo2D component to your form.
3. Set the Barcode2D property of the TLogo2D component to the TBarcode2D barcode component.
4. Add a logo picture to the Picture property of the TLogo2D component. The "bmp", "jpg"/"jpeg", "wmf", "emf", "ico", "png", "gif", and "tif"/"tiff" image formats are supported depend on the version of IDE.
5. Change the properties of the TLogo2D component to adjust the size and position of the logo picture.

If your TBarcode2D barcode component has linked to a TImage control, the barcode symbol with logo will be displayed in it.

If you use the DrawTo method to draw the barcode symbol to canvas, the logo will be drawn with the barcode symbol.

If you use the Print method to print the barcode symbol to paper, the logo will be printed with the barcode symbol.

If you use the TSave2D saving component to save the barcode symbol to picture file, including TSave2D_Bmp, TSave2D_Gif, TSave2D_Jpg, TSave2D_Png, and TSave2D_Svg saving components, the logo will be saved with the barcode symbol. Note, the TSave2D_Wmf, TSave2D_Emf, and TSave2D_Eps saving components don't save the logo picture.

3.10 How to print a barcode symbol to paper

Please use Print method of a barcode component to print the barcode symbol to paper. The TImage control isn't required.

For example:

```
Printer.BeginDoc;
... { Print other content }
Barcode2D_QRCode1.Print('1234567890', clBlack, clWhite, true, 20, 20, 0.3);
... { Print other content }
Printer.EndDoc;
```

or

```
Printer.BeginDoc;
```

```
... { Print other content }
with Barcode2D_QRCode1 do
begin
  ShowQuietZone := true;
  BarColor := clBlack;
  SpaceColor := clWhite;
  Barcode := '1234567890';
  Print(20, 20, 0.3);
end;
... { Print other content }
Printer.EndDoc;
```

3.11 How to save a barcode symbol to picture file

Usage:

1. Put a TBarcode2D barcode component, such as the TBarcode2D_QRCode, TBarcode2D_PDF417, and TBarcode2D_RSS14 to your form.
2. Put a TSave2D component, such as the TSave2D_Bmp, TSave2D_Png, and TSave2D_Svg to your form.
3. Set the Barcode2D property of the TSave2D component to the TBarcode2D barcode component.
4. Use the Save method of the TSave2D component to save the barcode symbol to picture file.

Note:

For the EAN.UCC compisite barcode component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, and for the TBarcode2D_MaxiCode component, please use the SaveToFile method of the TImage control that is linked to the barcode component to save the barcode symbol as a picture file.

For example:

```
Image1.Picture.Bitmap.SaveToFile('C:\2Dbarcode.bmp');
```

Also, You can bind multiple TSave2D components to a TBarcode2D barcode component in order to save the barcode symbol in multipe formats.

3.12 How to encode the UNICODE text in a 2D barcode symbol

By default, the text will be encoded in ANSI ecnoding sheme, you can use other encoding scheme, such as the UTF-8, UTF-16LE, UTF-16BE, etc.

- For Delphi/C++ Builder 2007 or early:

  Method 1, please convert the text to your encoding scheme, then assign it to the Barcode property of the barcode component, such as the TBarcode2D_QRCode, TBarcode2D_DataMatrixECC200, and the TBarcode2D_PDF417. The BOM may be placed depending on your application.

  For example:

```
var BarcodeText: string;
....
BarcodeText := '....';
// The text is encoded in UTF-8 format, and the BOM is placed.
Barcode2D_QRCode1.Barcode := #$EF#$BB#$BF + AnsiToUTF8(BarcodeText);
```

Method 2, Please create the OnEncode event function for the barcode component, such as the TBarcode2D_QRCode, TBarcode2D_DataMatrixECC200, and the TBarcode2D_PDF417. In the event function, you can encode the UNICODE text in your encoding scheme. The BOM may be placed depending on your application.

For example:

```
var BarcodeText: string;
....
BarcodeText := '....';
Barcode2D_QRCode1.Barcode := BarcodeText;
....
procedure TForm1.Barcode2D_QRCode1Encode(Sender: TObject; var Data: sAnsiString;
  Barcode: string);
begin
  // The text is encoded in UTF-8 format, and the BOM is placed.
  Data := #$EF#$BB#$BF + AnsiToUTF8(Barcode);
end;
```

- For Delphi/C++ Builder 2009 or later:

  Method 1, please convert the text to your encoding scheme, then assign it to the Data property of the barcode component, such as the TBarcode2D_QRCode, TBarcode2D_DataMatrixECC200, and the TBarcode2D_PDF417. The BOM may be placed depending on your application.

  For example:

```
// The text is encoded in UTF-8 format, and the BOM is placed.
var BarcodeText: string;
....
BarcodeText := '....';
Barcode2D_QRCode1.Data := #$EF#$BB#$BF + UTF8Encode(BarcodeText);
```

  Method 2, Please create the OnEncode event function for the barcode component, such as the TBarcode2D_QRCode, TBarcode2D_DataMatrixECC200, and the TBarcode2D_PDF417. In the event function, you can encode the UNICODE text in your encoding scheme. The BOM may be placed depending on your application.

  For example:

```
var BarcodeText: string;
....
BarcodeText := '....';
Barcode2D_QRCode1.Barcode := BarcodeText;
....
procedure TForm1.Barcode2D_QRCode1Encode(Sender: TObject; var Data: AnsiString;
  Barcode: string);
begin
```

```
  // The text is encoded in UTF-8 format, and the BOM is placed.
  Data := #$EF#$BB#$BF + UTF8Encode(Barcode);
end;
```

Chapter 4. Reference

4.1 TBarcode2D

It is the base class of all barcode components. And it cannot be instantiated. It's defined in the pBarcode2D unit.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.1 TBarcode2D_AztecCode

The component is used to create the Aztec Code 2D Barcodes symbols. It's defined in the pAztecCode unit.

Aztec Code is a two-dimensional matrix symbology whose symbols are nominally square, made up of square modules on a square grid, with a square bullseye pattern at their center. Aztec Code symbols can encode from small to large amounts of data with user-selected percentages of error correction. All 256 8-bit values can be encoded.

Aztec Code is invented by Andrew Longacre, Jr. and Robert Hussey in 1995. The code was published by AIM, Inc. in 1997. It is used for small item marking applications using a wide variety of printing and marking technologies.

Formats

There are two basic formats of Aztec Code symbols:

- **Compact**: There is a 2-ring bullseye pattern in the symbol center. It's useful for encoding shorter messages efficiently.

- **Full range**: There is a 3-ring bullseye pattern in the symbol center. It supports much larger symbols for longer data messages.

You can use the SymbolMode property to specify which formats and Aztec Code symbol sizes (see also the "Symbol sizes" section below) will be automatically selected between the minimum and maximum symbol sizes specified by corresponding MinSize and MaxSize properties, based on the length of barcode text. It can be one of these values (defined in the pAztecCode unit):

- **smNormal**: Only the symbol sizes that they are useful for data encoding operation, including all compact symbol sizes (azSize_15Compact, azSize_19Compact, azSize_23Compact, and azSize_27Compact), and full range symbol sizes from azSize_31 to azSize_151.
- **smCompact**: Only the symbol sizes in compact format, including the azSize_15Compact, azSize_19Compact, azSize_23Compact, and azSize_27Compact.
- **smFullRange**: Only the symbol sizes in full range format, including the azSize_19, azSize_23, azSize_27, and azSize_31 to azSize_151.
- **smProgram**: Only the symbol sizes that they are useful for reader initialization, including azSize_15Compact, azSize_19, azSize_23, azSize_27, and full range symbol sizes from azSize_31 to azSize_109. Note, in order to create a reader initialization symbol, a "\p" escape sequence should be placed into the barcode text, and the AllowEscape property should be set to true. See also the "Escape sequences" section below.
- **smAll**: All symbol sizes, including compact and full range formats.

Symbol sizes

The smallest Aztec Code symbol is 15 * 15 modules square, and the largest is 151 * 151. There are 36 square symbol sizes available in Aztec Code symbology. These are specified in following table (defined in the pAztecCode unit):

Sizes	Formats	Layers	Dimension (modules)	Maximum data capacities		
				Digits	Text	Bytes
azSize_15Compact	Compact	1	15 * 15	13	12	6
azSize_19	Full range	1	19 * 19	18	15	8
azSize_19Compact	Compact	2	19 * 19	40	33	19
azSize_23	Full range	2	23 * 23	49	40	24
azSize_23Compact	Compact	3	23 * 23	70	57	33
azSize_27	Full range	3	27 * 27	84	68	40
azSize_27Compact	Compact	4	27 * 27	110	89	53
azSize_31	Full range	4	31 * 31	128	104	62
azSize_37		5	37 * 37	178	144	87
azSize_41		6	41 * 41	232	187	114
azSize_45		7	45 * 45	294	236	145
azSize_49		8	49 * 49	362	291	179
azSize_53		9	53 * 53	433	348	214
azSize_57		10	57 * 57	516	414	256
azSize_61		11	61 * 61	601	482	298
azSize_67		12	67 * 67	691	554	343
azSize_71		13	71 * 71	793	636	394
azSize_75		14	75 * 75	896	718	446
azSize_79		15	79 * 79	1008	808	502
azSize_83		16	83 * 83	1123	900	559
azSize_87		17	87 * 87	1246	998	621
azSize_91		18	91 * 91	1378	1104	687
azSize_95		19	95 * 95	1511	1210	753
azSize_101		20	101 * 101	1653	1324	824
azSize_105		21	105 * 105	1801	1442	898
azSize_109		22	109 * 109	1956	1566	976
azSize_113		23	113 * 113	2116	1694	1056
azSize_117		24	117 * 117	2281	1826	1138
azSize_121		25	121 * 121	2452	1963	1224
azSize_125		26	125 * 125	2632	2107	1314
azSize_131		27	131 * 131	2818	2256	1407
azSize_135		28	135 * 135	3007	2407	1501
azSize_139		29	139 * 139	3205	2565	1600
azSize_143		30	143 * 143	3409	2728	1702
azSize_147		31	147 * 147	3616	2894	1806
azSize_151		32	151 * 151	3832	3067	1914

Note:

- Full range symbols with 1, 2, or 3 layers (azSize_19, azSize_23, and azSize_27) are useful only for reader initialization.
- The data capacities shown are based on the recommended error correction levels (23 % of symbol capacity plus 3 codewords).

You can use the MinSize and the MaxSize properties to specify the minimum and maximum sizes for an Aztec Code symbol. The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum and maximum symbol sizes.

If the barcode text does not fill the maximum data capacity of the Aztec Code symbol, remaining data capacity of the symbol will be used as excess error correction. If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxSize property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur. The

CurrentSize property can be used to get the factual symbol size.

Quiet zones

No quiet zone is required outside the bounds of the Aztec Code symbol. So the minimum values ofLeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 0.

Error checking and correcting (ECC)

The design of Aztec Code technically allows a symbol to include as little as none or as much as 99% in error correction codewords, though both limits are unsound. The recommended level of error correction for normal use is 23% of symbol capacity plus 3 codewords more. They may also choose to override the default error correction level, specifying either an alternate minimum error correction percentage or a fixed symbol format and ECC size.

Users, judging their applications to be especially benign or critical, may choose to specify an alternate minimum error correction percentage (specified by the ECCLevel property), ranging from 0% to 99% plus always, for data security, 3 or more additional error correction codewords (specified by the ECCCount property). This is called a "minimum" percentage because, depending on message length, the symbology will typically have to add extra error correction codewords above this minimum to fill out the symbol.

Some applications will be best served by specifying a fixed size (number of codewords) to be used for all Aztec Code symbols regardless of their data content. In this case, the ECCLevel property should be set to 0, and the ECCCount property should be set to a size which includes adequate error correction for the longest message anticipated; then, typically shorter messages will be encoded with excess error correction, creating more robust symbols for this application.

Character set

1. All 8-bit values can be encoded. The default interpretation shall be:
    - For values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646 (ASCII); Note this version consists of the GO set of ISO/IEC 646 and the CO set of ISO/IEC 6429 with values 28 to 31 modified to FS, GS, RS and U respectively).
    - For values 128 - 255, in accordance with ISO/IEC 8859-1 (Extended ASCII).

    This interpretation corresponds to ECI 000003.
2. Two non-data characters can be encoded, FNC1 for compatibility with some existing applications and ECI indicator blocks for the standardized encoding of message interpretation information.
    - **FNC1**: The FNC1 character following an application standard agreed with AIM International, identifies a symbol which conforms to a specific industry standard. FNC1 shall be used as defined in the EAN.UCC General Specifications either in the first position or implied by the mode character. The FNC1 character may also be used as a field separator, in which case it will be represented in the transmitted message as GS character (ASCII value 29). The escape sequence "\f" can be used to place the FNC1 character to barcode text.
    - **ECI**: The escape sequence "\e[m]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

    The AllowEscape property should be set to true in order to place the FNC1 characters or the ECI indicator blocks.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\f**: Insert a FNC1 character to barcode text. See also the "Character set" section above.
- **\e[<ECI_Number>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.
- **\s[<Index>,<Amount>,<Message_ID>]**: Insert a structured append block to barcode text in order to create the symbol in a structured append. See also the "Structured append" section below.

- **\p**: Indicates to create a reader initialization symbol. It can be placed anywhere in the barcode text. Only the azSize_15Compact, azSize_19, azSize_23, azSize_27, and full range symbol sizes from azSize_31 to azSize_109 are useful for reader initialization symbol, so the SymbolMode property should be set to "smProgram" in order to create the reader initialization symbol.

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set. Four broad types of interpretations are supported in Aztec Code:

- International character sets (or code pages).
- General purpose interpretations such as encryption and compaction.
- User defined interpretations for closed systems.
- Control information for structured append in unbuffered mode.

The ECI protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding. The ECI is identified by an integer (up to 6 digits) which is encoded in the Aztec Code symbol by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 999999 (including the boundaries), the leading zero is optional.

ECI indicator blocks may be placed anywhere in the barcode text in a single or structured append set of Aztec Code symbols. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Structured append

In order to fit a non-square area or to handle larger messages than are practical in a single symbol, a data message can be distributed across several Aztec Code symbols. Up to 26 Aztec Code symbols may be appended in a structured format to convey more data. If a symbol is part of a structured append this shall be indicated by a structured append block in barcode text. The escape sequence "\s[<Index>, <Amount>, <Message_ID>]" is used to place the structured append block to the barcode text:

- **Index**: The position index of the symbol within the set of Aztec Code symbols in the structured append format. It's an integer between 1 and 26 (including the boundaries) in string format.
- **Amount**: The total amount of the symbol within the set of Aztec Code symbols in the structured append format. It's an integer between 2 and 26 (including the boundaries) in string format.
- **Message_ID**: An optional message ID. The field is any number of data characters excluding the space character, and shall be the same for all symbols which comprise the same message. Though the field may be made up of any characters (except additional spaces), the most efficient encoding will result if it is a string of uppercase letters.

The AllowEscape property should be set to true in order to place the structured append block. The structured append block may only be placed only once in the barcode text. Also, it shall be placed at beginning of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the structured append block be placed more than once, or it isn't placed at beginning of the barcode text. The following is an example of structured append:

```
\s[2, 5, DESCRIPTION]ABCDEFGabcdefg1234567890
```

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Inversed
- Mirrored
- SymbolMode
- BytesAlwaysBackToUpper
- MinSize
- MaxSize
- ECCLevel
- ECCCount
- AllowEscape
- CurrentSize (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.2 TBarcode2D_AztecRunes

The component is used to create the Aztec Runes 2D Barcode symbols. It's defined in the pAztecRunes unit.

Aztec Runes are a series of small but distinct machine-readable marks designed to be graphically compatible with Aztec Code. They are in fact just the core symbol of a compact Aztec Code symbol with a numerically distinct mode message which in this case conveys 8 bits of actual data. Thus they comprise 256 11x11 module square marks which are conveniently found and read by an Aztec Code reader.

Symbol size

Each Aztec Runes symbol is of a fixed size, it is 11 * 11 modules square.

Quiet zones

No quiet zone is required outside the bounds of the Aztec Runes symbol. So the minimum values ofLeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 0.

Character set

All 8-bit values can be encoded, every Aztec Runes symbol can encode one 8-bit value, it's expressed in a decimal integer between 0 and 255 (including the boundaries), in string format, so only numeric characters can be used in the barcode text.

Data capacity

An integer between 0 and 255 (including the boundaries), in string format. The maximum length of barcode text is limited to 3 digits. The FixedLength property specifies whether adding leading zeros are required if the length is less than 3 digits.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Inversed
- Mirrored
- FixedLength

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.3 TBarcode2D_CCA

The component is used to create the EAN.UCC compisite barcode symbol that uses the CC-A as the 2D symbol. It's defined in the pCCA unit.

CC-A / RSS-14

CC-A / EAN-13

An EAN.UCC composite symbol consists of a linear symbol (encoding the item's primary identification) associated with an adjacent 2D symbol (encoding supplementary data, such as a batch number or expiration date). The CC-A is one of the three choices for the 2D symbol in a symbol encoded in the EAN.UCC composite symbology. It's a multi-row symbology component derived from MicroPDF417, for use only in EAN.UCC composite symbols. It's designed for efficient encoding of supplemental application identifier data.

Symbol sizes

The CC-A symbols have two, three, or four data columns. The number of columns will be automatically selected depending on the linear symbol that's associated with the CC-A symbol (specified by the Linear property).

Each CC-A symbol consists of a stack of vertically-aligned rows (with a minimum of three and a maximum of twelve rows,referred to as ccaRow_3 to ccaRow_10, and ccaRow_12, for example, the value ccaRow_5 denotations the CC-A symbol is 5 stacked rows, they are defined in the pCCA unit). The RowHeight property can be used to specify the height for each row, in modules. The allowable numbers of rows are specified separately for each of the two-, three-, and four-column symbol sizes. In other words, CC-A symbols shall conform with certain predefined combinations of numbers of rows, columns. These symbol sizes are defined in following table:

Number of data columns	Number of rows		Liner symbol	Max Capacity in Bits
2	5	ccaRow_5	RSS-14 Stacked, RSS-14 Stacked Omnidirectional, UPC-E, UPC-E0, UPC-E1	59
	6	ccaRow_6		78
	7	ccaRow_7		88
	8	ccaRow_8		108
	9	ccaRow_9		118
	10	ccaRow_10		138
	12	ccaRow_12		167
3	4	ccaRow_4	RSS Limited, EAN-8	78
	5	ccaRow_5		98
	6	ccaRow_6		118
	7	ccaRow_7		138
	8	ccaRow_8		167
4	3	ccaRow_3	RSS-14 Standard, RSS-14 Truncated, RSS Expanded, RSS Expanded Stacked, UPC-A, EAN-13, EAN-128	78
	4	ccaRow_4		108
	5	ccaRow_5		138
	6	ccaRow_6		167
	7	ccaRow_7		197

You can use the MinRows property to specify a minimum number of stacked rows, and use the MaxRows property to specifiy a maximum number of stacked rows. the smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum and the maximum number of stacked rows.

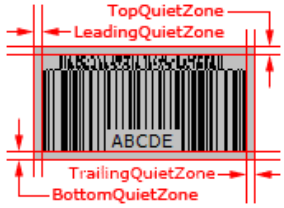
You can use the CurrentRows property to get the factual number of stacked rows.

If the barcode text is so long that it cannot be encoded using the maximum number of stacked rows specified by the MaxRows property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

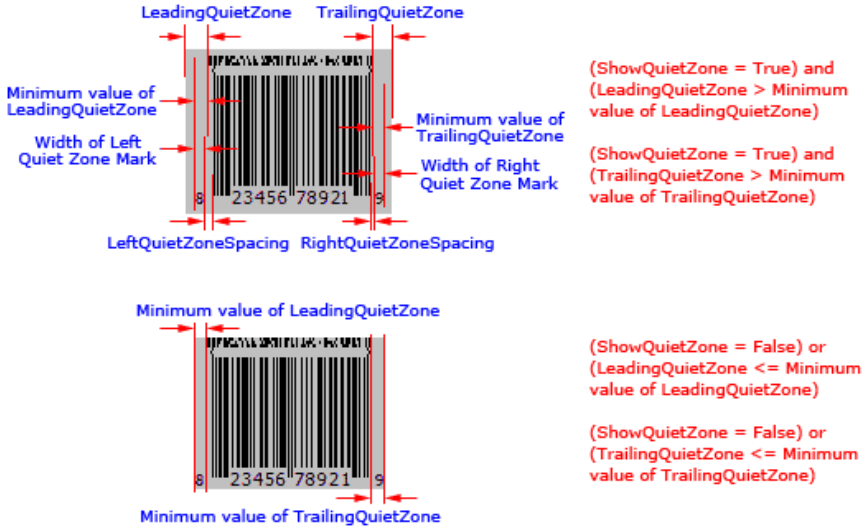
Quiet zones

For the CC-A symbol, The leading quiet zone and trailing quiet zone shall be a minimum of 1 modules. No top quiet zone and bottom quiet zone are required outside the bounds of the CC-A symbol.

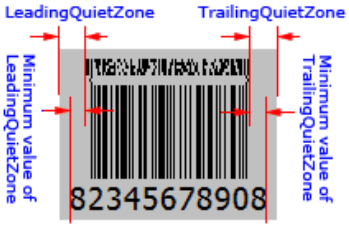
The LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the CC-A 2D symbol. So the minimum values of these properties are equal to 0. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



If the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone property is set to false, the minimum width of leading quiet zone that is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol will be used. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone property is set to false, the minimum width of trailing quiet zone that is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol will be used. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



If the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones is included if the ShowQuietZone property is set to true), the leading and trailing quiet zoens will be extanded automatically to cover the human readable text, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



(ShowQuietZone = True) and (LeadingQuietZone > Minimum value of LeadingQuietZone)

(ShowQuietZone = True) and (TrailingQuietZone > Minimum value of TrailingQuietZone)

(ShowQuietZone = False) or (LeadingQuietZone <= Minimum value of LeadingQuietZone)

(ShowQuietZone = False) or (TrailingQuietZone <= Minimum value of TrailingQuietZone)

Separator pattern

A separator pattern is required between the linear and 2D symbols of an EAN.UCC composite barcode symbol. If you use the TBarcode2D_CCA component together with a TBarcode2D_RSS14, TBarcode2D_RSSLimited, TBarcode2D_RSSExpanded, or TBarcode1D_EAN128 linear barcode component (specified by the Linear property), the separator pattern will be represented by the linear component. If you use the TBarcode2D_CCA component together with a TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, or TBarcode1D_EAN13 linear barcode component (specified by the Linear property), the separator pattern will be represented by the TBarcode2D_CCA component. See digiam (the color of separator patterns are changed to red in order to accentuate them):

CC-A / RSS-14 CC-A / EAN-13

Note, the TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, and TBarcode1D_EAN128 linear components are in the **Barcode VCL Components** package.

Error checking and correcting (ECC)

Each defined CC-A symbol size has a fixed number of error correction codewords. The error correction codewords provide capability for both error detection and correction.

Character set

A subset of ISO/IEC 646, consisting of the upper and lower case letters (A-Z, a-z), digits (0-9), space, and 19 selected punctuation characters (!:%&()*+,-./:;<=>?_) in addition to the special function characters, FNC1 and symbol separator.

If you want insert the FNC1, please insert the "^" character instead. And if you want insert the symbol separator, please insert the "#" character instead.

Note, in the barcode text, all Application Identifiers should be enclosed in parentheses "(" and ")", the parentheses are only for identifying Application Identifiers, and they are not encoded into the barcode symbol. If you want encode the parentheses "(" and ")" in an Application Identifier element string, please use the "{" and "}" instead.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- RowHeight
- Linear
- MinRows
- MaxRows
- CurrentRows (Raead only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.4 TBarcode2D_CCB

The component is used to create the EAN.UCC compisite barcode symbol that uses the CC-B as the 2D symbol. It's defined in the pCCB unit.

An EAN.UCC composite symbol consists of a linear symbol (encoding the item's primary identification) associated with an adjacent 2D symbol (encoding supplementary data, such as a batch number or expiration date). The CC-B is one of the three choices for the 2D symbol in a symbol encoded in the EAN.UCC composite symbology. A CC-B symbol is a MicroPDF417 symbol with a codeword of 920 in the first data position.

Symbol sizes

The CC-B symbols have two, three, or four data columns (The one-columnMicroPDF417 symbol isn't used). The number of columns will be automatically selected depending on the linear symbol that's associated with the CC-B symbol (specified by theLinear property).

Each CC-B symbol consists of a stack of vertically-aligned rows (with a minimum of four and a maximum of forty four rows, referred to as ccbRow_4 to ccbRow_44, for example, the value ccbRow_8 denotations the CC-B symbol is 8 stacked rows, they are defined in the pCCB unit). The RowHeight property can be used to specify the height for each row, in modules. The allowable numbers of rows are specified separately for each of the two-, three-, and four-column symbol sizes. In other words, CC-B symbols shall conform with certain predefined combinations of numbers of rows, columns. These symbol sizes are defined in following table:

Number of data columns	Number of rows		Liner symbol	Max Capacity in Bits
2	8	ccbRow_8	RSS-14 Stacked, RSS-14 Stacked Omnidirectional, UPC-E, UPC-E0, UPC-E1	59
	11	ccbRow_11		78
	14	ccbRow_14		88
	17	ccbRow_17		108
	20	ccbRow_20		118
	23	ccbRow_23		138
	26	ccbRow_26		167
3	6	ccbRow_6	RSS Limited, EAN-8	78
	8	ccbRow_8		98
	10	ccbRow_10		98
	12	ccbRow_12		118
	15	ccbRow_15		138
	20	ccbRow_20		167
	26	ccbRow_26		167
	32	ccbRow_32		167
	38	ccbRow_38		167
	44	ccbRow_44		167
4	4	ccbRow_4	RSS-14 Standard, RSS-14 Truncated, RSS Expanded, RSS Expanded Stacked, UPC-A, EAN-13, EAN-128	78
	6	ccbRow_6		98
	8	ccbRow_8		98
	10	ccbRow_10		98
	12	ccbRow_12		118
	15	ccbRow_15		138
	20	ccbRow_20		167
	26	ccbRow_26		167
	32	ccbRow_32		167
	38	ccbRow_38		167
	44	ccbRow_44		167

You can use the MinRows property to specify a minimum number of stacked rows, and use the MaxRows property to specifiy a maximum number of stacked rows. the smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum and the maximum number of stacked rows.

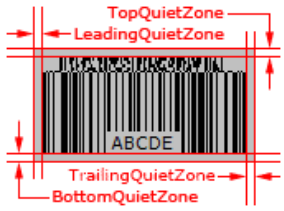
You can use the CurrentRows property to get the factual number of stacked rows.

If the barcode text is so long that it cannot be encoded using the maximum number of stacked rows specified by the MaxRows property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Quiet zones

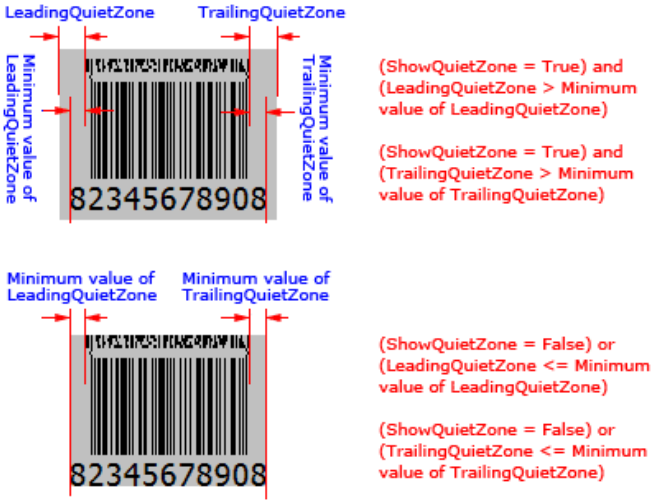
For the CC-B symbol, The leading quiet zone and trailing quiet zone shall be a minimum of 1 modules. No top quiet zone and bottom quiet zone are required outside the bounds of the CC-B symbol.

The LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the CC-B 2D symbol. So the minimum values of these properties are equal to 0. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



If the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone property is set to false, the minimum width of leading quiet zone that is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol will be used. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone property is set to false, the minimum width of trailing quiet zone that is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol will be used. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

If the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones is included if the ShowQuietZone property is set to true), the leading and trailing quiet zoens will be extanded automatically to cover the human readable text, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



Separator pattern

A separator pattern is required between the linear and 2D symbols of an EAN.UCC composite barcode symbol. If you use the TBarcode2D_CCB component together with a TBarcode2D_RSS14, TBarcode2D_RSSLimited, TBarcode2D_RSSExpanded, or TBarcode1D_EAN128 linear barcode component (specified by the Linear property), the separator pattern will be represented by the linear component. If you use the TBarcode2D_CCB component together with a TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, or TBarcode1D_EAN13 linear barcode component(specified by the Linear property), the separator pattern will be represented by the TBarcode2D_CCB component. See digiam (the color of separator patterns are changed to red in order to accentuate them):

Note, the TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, and TBarcode1D_EAN128 linear components are in the **Barcode VCL Components** package.

Error checking and correcting (ECC)

Each defined CC-B symbol size has a fixed number of error correction codewords. The error correction codewords provide capability for both error detection and correction.

Character set

A subset of ISO/IEC 646, consisting of the upper and lower case letters (A-Z, a-z), digits (0-9), space, and 19 selected punctuation characters (!:%&()*+,-./:;<=>?_) in addition to the special function characters, FNC1 and symbol separator.

If you want insert the FNC1, please insert the "^" character instead. And if you want insert the symbol separator, please insert the "#" character instead.

Note, in the barcode text, all Application Identifiers should be enclosed in parentheses "(" and ")", the parentheses are only for identifying Application Identifiers, and they are not encoded into the barcode symbol. If you want encode the parentheses "(" and ")" in an Application Identifier element string, please use the "{" and "}" instead.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- RowHeight
- Linear
- MinRows
- MaxRows
- CurrentRows (Raead only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.5 TBarcode2D_CCC

The component is used to create the EAN.UCC compisite barcode symbol that uses the CC-C as the 2D symbol. It is defined in the pCCC unit.

An EAN.UCC composite symbol consists of a linear symbol (encoding the item's primary identification) associated with an adjacent 2D symbol (encoding supplementary data, such as a batch number or expiration date). The CC-C is one of the three choices for the 2D symbol in a symbol encoded in the EAN.UCC composite symbology. A CC-C symbol is a PDF417 symbol with a codeword of 920 in the first data position.

Symbol sizes

Each CC-C symbol consists of a stack of vertically aligned rows with a minimum of 3 rows (maximum 90 rows). Each stacked row shall include a minimum of 1 symbol character column (maximum 30 symbol character columns), excluding start, stop and row indicator columns. They are defined in the pPDF417Custom unit. The RowHeight property can be used to specify the height for each

stacked row, in modules. See diagram:

You can use the MinRows and the MaxRows properties to specifiy the minimum and maximum number of stacked rows for a CC-C symbol. And use the MinColumns and the MaxColumns properties to specifiy the minimum and maximum number of symbol character columns for it. In other words, the MinRows and MinColumns properties specify a minimum symbol size, and the MaxRows and MaxColumns properties specify a maximum symbol size. According to the priority order specified by the StretchOrder property, the first symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

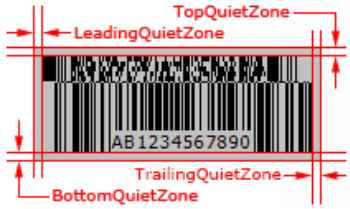
You can use the CurrentRows property to get the factual number of stacked rows. And use the CurrentColumns property to get the factual number of columns.

If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxRows and the MaxColumns properties, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Quiet zones

For the CC-C symbol, The leading quiet zone and trailing quiet zone shall be a minimum of 2 modules. No top quiet zone and bottom quiet zone are required outside the bounds of the CC-C symbol.

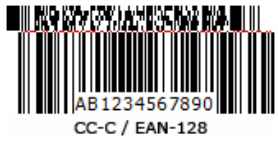
The LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the CC-C 2D symbol. So the minimum values of TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 0. For the LeadingQuietZone property, the minimum values of the property is equal to 2 because the first bar of the CC-C symbol is on the left of the EAN-128 symbol always. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



If the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones is included if the ShowQuietZone property is set to true), the leading and trailing quiet zoens will be extanded automatically to cover the human readable text, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

Separator pattern

A separator pattern is required between the EAN-128 linear and CC-C 2D symbols of an EAN.UCC composite barcode symbol. The separator pattern will be represented by the TBarcode1D_EAN128 linear component (specified by the Linear property). See digiam (the color of separator patterns are changed to red in order to accentuate them):



Note, the TBarcode1D_EAN128 linear component is in the **1D Barcode VCL Components** package.

Error checking and correcting (ECC)

In fact, a CC-C symbol is a specialPDF417 symbol, so CC-C symbols offer 9 levels of error correction too, referred to as ECC 0 to ECC 8 respectively in increasing order of recovery capacity. You can use the ECCLevel property to specify the error correction code level for a CC-C symbol. It can be one of values fromelEcc_0 to elEcc_8, corresponding to error correction code level from ECC 0 to ECC 8. They are defined in the pPDF417Custom unit.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined based on the length of barcode text, and the error correction code level specified by the ECCLevel property, in other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The CurrentECCLevel property can be used to get the factual error correction code level.

Character set

A subset of ISO/IEC 646, consisting of the upper and lower case letters (A-Z, a-z), digits (0-9), space, and 19 selected punctuation characters (!:%&()*+,-./:;<=>?_) in addition to the special function characters, FNC1 and symbol separator.

If you want insert the FNC1, please insert the "^" character instead. And if you want insert the symbol separator, please insert the "#" character instead.

Note, in the barcode text, all Application Identifiers should be enclosed in parentheses "(" and ")", the parentheses are only for identifying Application Identifiers, and they are not encoded into the barcode symbol. If you want encode the parentheses "(" and ")" in an Application Identifier element string, please use the "{" and "}" instead.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- RowHeight
- Linear
- MinRows
- MaxRows
- MinColumn
- MaxColumn
- StretchOrder
- ECCLevel
- ECCLevelUpgrade
- Compact
- CurrentRows (Raead only)
- CurrentColumns (Read only)
- CurrentECCLevel (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.6 TBarcode2D_Code16K

The component is used to create the Code 16K 2D Barcode symbols. It's defined in the pCode16K unit.

Code 16K was developed by Ted Williams in 1989 to provide a simple to print and decode multiple row symbology. It's a continuous, variable-length, stacked 2D barcode symbology that can encode the complete ASCII 128-character set. Extended ASCII characters (ASCII value 128 to 255) may also be encoded by using function character.

Each Code 16K symbol contains from 2 to 16 rows. Each row is divided by a separator bar. The top and bottom of the symbol also have separator bars that extend to the ends of the minimum quiet zones.

Symbol size

Each Code 16K symbol contains from 2 to 16 stacked rows, with 5 ASCII characters per stacked row. TheRowHeight property can be used to specify the height for each stacked row, in modules. The data capacities are listed in following table:

Rows	Maximum data capacities	
	Numeric characters	ASCII characters
2	14	7
3	24	12
4	34	17
5	44	22
6	54	27
7	64	32
8	74	37
9	84	42
10	94	47
11	104	52
12	114	57
13	124	62
14	134	67
15	144	72
16	154	77

You can use the MinRows and the MaxRows properties to specify the minimum and maximum number of stacked rows for a Code 16K symbol. They can be one of values from 2 to 16 (defined in the pCode16K unit). The smallest number of symbol stacked rows that accommodates the barcode text will be automatically selected between minimum and maximum number of stacked rows.

If the barcode text does not fill the maximum data capacity of the Code 16K symbol, remaining data capacity of the symbol will be filled by adding the PAD characters automatically. If the barcode text is so long that it cannot be encoded using the maximum number of stacked rows specified by the MaxRows property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur. You can use the CurrentRows property to get the factual number of stacked rows.

The Code 16K symbol width is 81 modules (inclusive of minimumleading quiet zone and trailing quiet zone). The minimum row height value is 8 times the module width. So the smallest Code 16K symbol is 81 * 19 modules square and the largest is 81 * 145 modules square (inclusive of minimum quiet zones, the row height is set to 8 modules, and the separator bar height is set to 1 module).

The minimum value of the module width is 7.5 mils (0.19 mm). So the minimum physical size is 15.4mm * 3.6mm. The maximum data density is 208 alphanumeric characters per square inch or 417 numeric digits per square inch when the symbol is printed at 7.5 mils.

Quiet zones

The leading quiet zone shall be a minimum of 10 modules, the trailing quiet zone shall be a minimum of 1 modules. No top quiet zone and bottom quiet zone are required outside the bounds of the symbol.

So the minimum value of LeadingQuietZone property is equal to 10, the minimum value of TrailingQuietZone property is equal to 1, and the minimum value of TopQuietZone, and BottomQuietZone properties are equal to 0.

Character set

- All 128 ASCII characters, i.e. ASCII characters 0 to 127 inclusive, in accordance with ISO/IEC 646:1991.

  Code 16K has three unique data character sets as code sets A, B and C, all 128 ASCII characters are encoded by internally switching between all 3 code sets:

  - **Code set A**: Includes characters with ASCII values from 00 to 95 (i.e. all of the standard upper case alphanumeric characters together with the control characters inclusive), and function characters.
  - **Code set B**: Includes characters with ASCII values from 32 to 127 (i.e. all of the standard upper case alphanumeric characters together with the lower case alphabetic characters inclusive), and function characters.
  - **Code set C**: includes the set of 100 digit pairs from 00 to 99 inclusive, as well as seven special characters. This allows numeric data to be encoded, two data digits per symbol character, at effectively twice the density of standard data.

  The code set will be switched automatically in a Code 16K symbol in order to minimize the symbol size. Also, you can manually switch the code set by using following escape sequences:

  - **\a**: Switch to code set A.
  - **\b**: Switch to code set B.
  - **\c**: Switch to code set C.

  The AllowEscape property should be set to true in order to place these escape sequences. If the symbol mode specified by the InitialMode property is set to emCodeC_Shift1B, they cannot be used as the first character in the symbol. If the InitialMode property is set to emCodeC_Shift2B, they cannot be used as the first two characters in the symbol. In addition they can be placed anywhere within the symbol.

- Characters with ASCII values 128 to 255 in accordance with ISO 8859-1:1998 may also be encoded. This is done by internally using the FNC4 character together with code sets A, B and C.

- 4 function characters, and PAD character:

  - **FNC1**: The FNC1 character following an application standard agreed with AIM International, identifies a symbol which conforms to a specific industry standard. FNC1 shall be used as defined in the EAN.UCC General Specifications either in the first position or implied by the mode character. The FNC1 character may also be used as a field separator, in which case it will be represented in the transmitted message as GS character (ASCII value 29). The escape sequence "\1" can be used to place the FNC1 character to barcode text.
  - **FNC2**: It's used internally to create a message append defined below. The escape sequence "\2" can be used to place the FNC2 character to barcode text. In general, you shouldn't place the FNC2 to the barcode text, it's used internally by the component when the symbol is in a message append. See also the "Message append" section below.
  - **FNC3**: Initialize. This instructs the reader to interpret the data contained in the symbol for reader initialization or programming. The FNC3 may be placed anywhere within the symbol. The escape sequence "\3" can be used to place the FNC3 character to barcode text.
  - **FNC4**: It's used internally together with code sets A, B and C to encode the extended ASCII characters (ASCII values from 128 to 255). The FNC4 character cannot be placed to barcode text directly, it's used internally by the component.
  - **PAD**: If the barcode text does not fill the maximum data capacity of a Code 16K symbol, PAD characters will be added automatically to fill the remaining data capacity of the symbol. Also, you can place a PAD character anywhere in the barcode symbol by using the escape sequence "\p", in order to create the special Code 16K symbol for closed system.

  The AllowEscape property should be set to true in order to place the functions characters, or the PAD character.

Initial modes

Code 16K has initial modes which are used to specify the initial code set and may also represent an implied leading FNC1 character or implied leading SHIFT B character as shown in following table. The code set will be automatically switched if a character is encountered that cannot be encoded by current code set. And implied characters function as if they were actual symbol characters but don not occupy any space.

There are seven kinds of initial mode, from 0 to 6, and a kind of extended data length mode. The initial mode values from emCodeA to emExtended, corresponding to these modes, are defined in the pCode16K unit. These modes and their values are listed in following table:

Initial Mode	Value	Initial code set	Implied character	Description
0	emCodeA	A	(None)	The code set will be automatically switched if another code set character is encountered.
1	emCodeB	B	(None)	
2	emCodeC	C	(None)	
3	emCodeB_FNC1	B	FNC1	
4	emCodeC_FNC1	C	FNC1	
5	emCodeC_Shift1B	C	SHIFT B	First character excepting the message append block (if exists), must be code set B character (ASCII 32 - ASCII 127), otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. The code set will be automatically switched if another code set character is encountered.
6	emCodeC_Shift2B	C	Double SHIFT B	First two characters excepting the message append block (if exists), must be code set B characters (ASCII 32 - ASCII 127), otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. The code set will be automatically switched if another code set character is encountered.
Extended data length mode	emMode_Extended	B	None	Indicates to create a Code 16K barcode symbol in extended data length mode. An extended data length mode block is required, and it should be placed at beginning of barcode text, otherwiset the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. See also the "Extended data length mode" section below.

The InitialMode property can be used to specify the initial mode for a Code 16K symbol. It can be one of values shown in table above, corresponding to the initial modes 0 to 6, and the extended data length mode. Also, it can be set to emAuto (defined in the pCode16K unit), in this case, one of values from emCodeA to emModeC_Shift2B, corresponding to the initial modes 0 to 6 shown in table above will be selected automatically, depending on the barcode text, in order to minimize the symbol size.

You can always use the CurrentMode property to get the factual initial mode.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\1**: Insert a FNC1 character to barcode text. See also the "Character set" section above.
- **\2**: Insert a FNC2 character to barcode text. See also the "Character set" section above.
- **\3**: Insert a FNC3 character to barcode text. See also the "Character set" section above.
- **\a**: Manually switch code set to code set A. See also the "Character set" section above.
- **\b**: Manually switch code set to code set B. See also the "Character set" section above.
- **\c**: Manually switch code set to code set C. See also the "Character set" section above.
- **\p**: Insert a PAD character to barcode text. See also the "Character set" section above.
- **\m[<Index>, <Amount>]**: Insert a extended data length mode block to barcode text in order to create the symbol in extended data length mode. See also the "Extended data length mode" section below.
- **\s[<Index>, <Amount>]**: Insert a message append block to barcode text in order to create the symbol in a message append. See also the "Message append" section below.

Extended data length mode

The extended data length mode is used to encode data beyond the capacity of a single Code 16K symbol. In the extended data

length mode, up to 107 Code 16K 16-rows symbols may be arranged to convey more data (up to 8025 ASCII characters, or 16050 numeric digits). If a symbol is part of these extended data length mode symbols, this shall be indicated by extended data length mode block in barcode text. The escape sequence "\m[<Index>, <Amount>]" is used to place the extended data length mode block to the barcode text:

- **Index**: The position index of the symbol within the set of Code 16K symbols in the extended data length mode. It's an integer between 1 and 107 (including the boundaries) in string format.
- **Amount**: The total amount of the symbol within the set of Code 16K symbols in the extended data length mode. It's an integer between 2 and 107 (including the boundaries) in string format.

The AllowEscape property should be set to true in order to place the extended data length mode block. The extended data length mode block may be placed only once in the barcode text. Also, it shall be placed at beginning of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the extended data length mode block be placed more than once, or it isn't placed at beginning of the barcode text. The following is an example of extended data length mode symbol:

```
\m[2,12]ABCDEFG1234567
```

The InitialMode property should be set to emAuto or emMode_Extended if an extended data length mode block is placed in the barcode text, otherwise the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. Conversely, the extended data length mode block is required if the InitialMode property is set to emMode_Extended, otherwise the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur too.

The extended data length mode block shouldn't be placed together with message append block in the barcode text, otherwise the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. See also the "Message append (structured append)" section below.

If a symbol is encoded in extended data length mode, the number of stacked rows will be changed to 16 automatically.

Note, if only two symbols are to be logically linked in extended data length mode, they may be arranged horizontally or vertically adjacent to each other. If more symbols are to be logically linked, they shall be arranged in a single vertical stack. The maximum number of symbols should be specified for the application.

Message append (structured append)

The message append is a method similar to the extended data length mode to encode data beyond the capacity of a single Code 16K symbol. Up to 9 Code 16K symbols may be appended in a structured format to convey more data. It should only be used in closed systems. If a symbol is part of a message append this shall be indicated by a message append block in barcode text. The escape sequence "\s[<Index>, <Amount>]" is used to place the message append block to the barcode text:

- **Index**: The position index of the symbol within the set of Code 16K symbols in the message append format. It's an integer between 1 and 9 (including the boundaries) in string format.
- **Amount**: The total amount of the symbol within the set of Code 16K symbols in the message append format. It's an integer between 2 and 9 (including the boundaries) in string format.

The AllowEscape property should be set to true in order to place the message append block. The message append block may be placed only once in the barcode text. Also, it shall be placed at beginning of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the message append block be placed more than once or it isn't placed at beginning of the barcode text. The following is an example of message append symbol:

```
\s[2,5]ABCDEFGHIJKLMN012345
```

The message block shouldn't be placed together with extended data length mode block in the barcode text, otherwise the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. See also the "Extended data length mode" section above.

The message append block shouldn't be placed in the barcode text if the InitialMode property is set to emMode_Extended, otherwise the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

The symbols in a message append should not be arranged horizontally, but only in a single vertical stack.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- MinRows
- MaxRows
- RowHeight
- SeparatorBarHeight
- InitialMode
- AllowEscape
- CurrentRows (Read only)
- CurrentMode (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.7 TBarcode2D_CompactMatrix

The component is used to create the Compact Matrix 2D barcode symbols. It's defined in the pCompactMatrix unit.

Compact Matrix is a variable-sized two-dimensional matrix symbology. The code graph adopts sprocket hole positioning and graphic sectioning techniques to perform fast and accurate identification and handling of 2D barcode graph by analyzing the information of sprocket position and graphic section.

Compact Matrix

Compact Matrix can encode 7-bit ASCII, numeric, and binary data, in addition to any combination of data

types in the same symbol, particularly effective with Chinese characters.

Compact Matrix was invented by Syscan Technology Co., Ltd.

Symbol sizes

There are 32 vertical sizes of Compact Matrix symbol, referred to as version 1 to 32, in increasing order of symbol height and data capacity. In horizontal orientation, each Compact Matrix symbol consists of an array of segments with a minimum of 1 segment (maximum 32 segments).

You can use the MinVersion and the MaxVersion properties to specifiy the minimum and maximum version for a Compact Matrix symbol. And use the MinSegments and the MaxSegments properties to specifiy the minimum and maximum number of segments for it. In other words, the MinVersion and MinSegments properties specify a minimum symbol size, and the MaxVersion and MaxSegments properties specify a maximum symbol size. According to the priority order specified by the StretchOrder property, the first symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

You can use the CurrentVersion property to get the factual version. And use the CurrentSegments property to get the factual number of segments.

If the barcode text does not fill the maximum data capacity of the Compact Matrix symbol, remaining data capacity of the symbol will be filled by using pad bits (the ECCLevelUpgrade property is set to false), or will be used to upgrade the error correction code level (the ECCLevelUpgrade property is set to true). If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxVersion and the MaxSegments properties, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Error correction code (ECC)

There are eight user-selectable levels of error correction, from 1 to 8 respectively in increasing order of recovery capacity.

You can use the ECCLevel property to specify the error correction code level for a Compact Matrix symbol. It can be one of values from 1 to 8, corresponding to the ECC levels from 1 to 8.

These ECC levels are listed in following table:

Error correction code level	Percentage of total capacity for ECC data
1	8%
2	16%
3	24%
4	32%
5	40%
6	48%
7	56%
8	64%

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will be used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by the ECCLevel property (see also the "Symbol sizes" section above). In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The property CurrentECCLevel can be used to get the factual error correction code level.

Quiet zones

The minimum quiet zone is equal to 6 modules on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 6.

Character set

- All 7-bit ASCII values can be encoded.

- 8-bit binary data can be encoded.
- GB 18030 Chinese characters can be encoded.
- Five non-data characters can be encoded:
  - **AIM FNC1** It identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. It is immediately followed by an application indicator assigned to identify the specification concerned by AIM International. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode. The escape sequence "\0" can be used to placed the FNC1 character to barcode text.
  - **GS1 FNC1** It identifies symbols encoding data formatted according to the GS1 Application Identifiers standard. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode text. The escape sequence "\1" can be used to placed the GS1 FNC1 character to barcode text.
  - **FNC2**: The FNC2 is used to implement structured append function in order to handle larger messages than are practical in a single symbol. The escape sequence "\2[<File_Id>, <Amount>, <Index>]" is used to place the FNC2 character and other structured append information (collectively referred to as structured append block) to barcode text. See also the "Escape sequences" and "Structured append" sections below.
  - **FNC3**: The FNC3 is used to indicates that the symbol encodes a message used to program the reader system. The escape sequence "\3" can be used to placed the FNC3 character to barcode text. See also the "Escape sequences" section below.
  - **ECI**: ECI indicator blocks is for the standardized encoding of message interpretation information. The escape sequence "\e[<ECI_Number>]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

The AllowEscape property should be set to true in order to place these non-data characters.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\0**: Insert an AIM FNC1 character to barcode text in order to encode AIM structural data similar to GS1. It shall be placed at beginning of the barcode text and it shall not be used with GS1 FNC1 ("\1") or FNC3 ("\3"). If it is used together with the structured append block (FNC2, "\2"), it shall be placed on front of the structured append block ("\2"). See also the "Structured append" section below.
- **\1**: Insert a GS1 FNC1 character to barcode text in order to encode GS1 structural data. It shall be placed at beginning of the barcode text and it shall not be used with AIM FNC1 ("\0") or FNC3 ("\3"). If it is used together with the structured append block ("\2"), it shall be placed on front of the structured append block (FNC2, "\2"). See also the "Structured append" section below.
- **\2[<File_Id>, <Amount>, <Index>]** Insert a structured append block (FNC2 and other structured append information) to the barcode text in order to create the symbol in a structured append. See also the "Structured append" section below.
- **\3**: Insert a FNC3 character to barcode text, indicates that the symbol encodes a message used to program the reader system. It shall be placed at beginning of the barcode text and it shall not be used with AIM FNC1 ("\0") or GS1 FNC1 ("\1"). If it is used together with the structured append block (FNC2, "\2"), it shall be placed on front of the structured append block (FNC2, "\2"). See also the "Structured append" section below.
- **\e[<ECI_Number>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set.

The ECI protocol provides a consistent method to specify particular interpretations on byte values before printing and after

decoding. The ECI is identified by an integer (up to 6 digits) which is encoded in the Compact Matrix symbol by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 811799 (including the boundaries), the leading zero is optional.

In a single symbol or structured append set of Compact Matrix symbols, if the AIM FNC1 ("\0"), GS1 FNC1 ("\1"), FNC2 (structured append block, "\2"), and FNC3 ("\3") are used, ECI indicator blocks may be placed anywhere behind of them, otherwise, they may be placed anywhere in the barcode text. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Structured append

In order to handle larger messages than are practical in a single symbol, a data message can be distributed across several Compact Matrix symbols. Up to 16 Compact Matrix symbols may be appended in a structured format to convey more data. If a symbol is part of a structured append this shall be indicated by a structured append block in barcode text. The escape sequence "\2[<File_Id>, <Amount>, <Index>]" is used to place the structured append block to the barcode text:

- **File_Id**: The file identification. It's an integer between 0 and 255 (including the boundaries) in string format. The purpose of the file identification is to increase the probability that only logically linked symbols are processed as part of the same message.
- **Amount**: The total amount of the symbol within the set of Compact Matrix symbols in the structured append format. It's an integer between 1 and 16 (including the boundaries) in string format.
- **Index**: The position index of the symbol within the set of Compact Matrix symbols in the structured append format. It's an integer between 1 and 16 (including the boundaries) in string format.

The AllowEscape property should be set to true in order to place the structured append block. The structured append block may only be placed once in the barcode text. If the structured append block is used together with AIM FNC1 ("\0"), GS1 FNC1 ("\1") or FNC3 ("\3"), the AIM FNC1 ("\0"), GS1 FNC1 ("\1") or FNC3 ("\3") shall be placed at beginning of the barcode text, then the structured append block. Otherwise, the structured append block shall be placed at beginning of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the structured append block be placed more than once, or it isn't placed at beginning of the barcode text or behind of the AIM FNC1 ("\0"), GS1 FNC1 ("\1") or FNC3 ("\3"). The following is examples of structured append:

```
\2[79, 5, 2]ABCDEFGabcdefg1234567890...
\0\2[99, 6, 1]ASDFGHJKL098765...
```

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- AllowEscape
- MinVersion
- MaxVersion
- MinSegments
- MaxSegments
- StretchOrder
- ECCLevel
- ECCLevelUpgrade
- StartWidth
- StopWidth
- Placement
- CurrentVersion (Read only)
- CurrentSegments (Read only)
- CurrentECCLevel (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.8 TBarcode2D_DataMatrix

The component is used to create the Data Matrix (ECC 000 - 140) 2D barcode symbols. It's defined in the pDataMatrix unit.

Data Matrix code is a two-dimensional matrix barcode symbology consisting of black and white "cells" or modules arranged in either a square or rectangular pattern. The information to be encoded can be text or raw data. Usual data size is from a few bytes up to 2 kilobytes. The length of the encoded data depends on the symbol dimension used. Error correction codes are added to increase symbol strength: even if they are partially damaged, they can still be read.

Data Matrix was invented by International Data Matrix, Inc. (ID Matrix) which was merged into RVSI/Acuity CiMatrix, who was acquired by Siemens AG in October, 2005 and Microscan Systems in September 2008.

There are 2 types of Data Matrix symbology, namely ECC 000 - 140 and ECC 200. The component can be used to generate the ECC 000 - 140 symbols. It is the conventional coding for error correction that was used in the initial installations of Data Matrix systems. It offers five levels of error correction using convolutional code error correction. ECC 000 - 140 symbols have an odd number of rows and an odd number of columns. Symbols are square with sizes from 9 * 9 to 49 * 49 (modules) not including quiet zones. These symbols can be recognized by the upper right corner module being dark.

ECC 000 - 140 should only be used in closed applications where a single party controls both the production and reading of the symbols and is responsible for overall system performance.

If you want to generate the Data Matrix ECC 200 symbols, please use another componentTBarcode2D_DataMatrixECC200 in this components package.

Error correction code (ECC)

Data Matrix (ECC 000 - 140) symbols offer five levels of error correction, referred to as ECC 000, ECC 050, ECC 080, ECC 100 and ECC 140 respectively in increasing order of recovery capacity. They are listed in following table:

Error correction code level	Maximum % correctable damage	% increase in used bits from ECC 000
ECC 000	none	none
ECC 050	2.8	33
ECC 080	5.5	50
ECC 100	12.6	100
ECC 140	25	300

In an application, it is important to understand that these error correction levels result in the generation of a proportional increase in the number of bits in the message (and hence increase in the size of the symbol), and offer different levels of error recovery.

You can use the ECCLevel property to specify the error correction code level for a Data Matrix (ECC 000 - 140) symbol. It can be one of these values: dmECC000, dmECC050, dmECC080, dmECC100, and dmECC140 (they are defined in the pDataMatrix unit), corresponding to the ECC levels ECC 000, ECC 050, ECC 080, ECC 100, and ECC 140.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will be used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by the ECCLevel property (see also the "Symbol sizes" section below). In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The property CurrentECCLevel can be used to get the factual error correction code level.

Character set

All 8-bit values can be encoded. The default interpretation shall be:

- For values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646 (ASCII); Note this version consists of the GC set of ISO/IEC 646 and the CO set of ISO/IEC 6429 with values 28 to 31 modified to FS, GS, RS and US respectively).
- For values 128 - 255, in accordance with ISO/IEC 8859-1 (These are referred to as extended ASCII).

Note, the character set isn't different based on the encoding mode. See also the "Encoding modes" section below.

Encoding modes

There are six encoding modes of Data Matrix (ECC 000 - 140) symbols, they are shown in following list, in decreasing order of encoding density:

- **Numeric (Base 11)**: The encoding mode encodes 10 numeric characters 0 to 9, and the space character. The encoding density is 3.5 bits per data character. The encoding mode has highest encoding density.
- **Alpha (Base 27)**: The encoding mode encodes 26 upper case letters A to Z, and the space character. The encoding density is 4.8 bits per data character.
- **Alphanumeric (Base 37)**: The encoding mode encodes 26 upper case letters A to Z, 10 numeric characters 0 to 9, and the space character. The encoding density is 5.25 bits per data character.
- **Punctuation (Base 41)**: The encoding mode encodes 26 upper case letters A to Z, 10 numeric characters 0 to 9, and the space, point(.), hyphen(-), comma(,) and solidus(/) characters. The encoding density is 5.5 bits per data character.
- **ASCII**: The encoding mode encodes all 128 ASCII characters from ISO/IEC 646. The encoding density is 7 bits per data character.
- **Binary**: The encoding mode encodes all 256 8-bit bytes. It shall be used for closed applications, where the data interpretation shall be determined by the user. The encoding density is 8 bits per byte. The encoding mode has lowest encoding density.

You can use the EncodeMode property to specify the encoding mode for a Data Matrix (ECC 000 - 140) symbol. It can be one of these value: emNumeric, emAlpha, emPunctuation, emAlphanumeric, emASCII, and emBinary, corresponding to the encoding modes Numberic (Base 11), Alpha (Base 27), Alphanumeric (Base 37), Punctuation (Base 41), ASCII, and Binary. Also, it can be set to emAuto, in this case, the encoding mode will be selected automatically from the list shown above, depending on the barcode text, in other words, the barcode text to be encoded will be analysed, and an appropriates lowest level (highest encoding density) encoding mode will be selected, in order to minimize the symbol size. The property CurrentEncodeMode can be used to get the factual encoding mode. These property values are defined in the pDataMatrix unit.

Symbol sizes

ECC 000 - 140 symbols have an odd number of rows and an odd number of columns. Symbols are square with sizes from 9 * 9 to 49 * 49 (modules) square not including quiet zones. These symbol sizes (excluding quiet zones) and their maximum data capacities are listed in following table:

Symbol size values	Symbol sizes (modules)	Maximum data capacities (bits)				
		ECC 000	ECC 050	ECC 080	ECC 100	ECC 140
dmSize_09_09	9 * 9	12	-	-	-	-
dmSize_11_11	11 * 11	44	6	-	-	-
dmSize_13_13	13 * 13	84	36	18	7	-
dmSize_15_15	15 * 15	132	72	50	31	-
dmSize_17_17	17 * 17	188	114	86	59	9
dmSize_19_19	19 * 19	252	162	130	91	25
dmSize_21_21	21 * 21	324	216	178	127	43
dmSize_23_23	23 * 23	404	276	230	167	63
dmSize_25_25	25 * 25	492	342	290	211	85
dmSize_27_27	27 * 27	588	414	354	259	109
dmSize_29_29	29 * 29	692	492	422	311	135
dmSize_31_31	31 * 31	804	576	498	367	163
dmSize_33_33	33 * 33	924	666	578	427	193
dmSize_35_35	35 * 35	1052	762	662	491	225
dmSize_37_37	37 * 37	1188	864	754	559	259
dmSize_39_39	39 * 39	1332	972	850	631	295
dmSize_41_41	41 * 41	1484	1086	950	707	333
dmSize_43_43	43 * 43	1644	1206	1058	787	373
dmSize_45_45	45 * 45	1812	1332	1170	871	415
dmSize_47_47	47 * 47	1988	1464	1286	959	459
dmSize_49_49	49 * 49	2172	1602	1410	1051	505

The maximum number of characters in each encoding mode can be calculated by using the Maximum data capacities (bits). For example, the maximum data capacities are 114 bits if the symbol size is 17 * 17 and the error correction code level is ECC 050. The encoding density is 4.8 bits per data character in Alpha (Base 27) encoding mode, so the maximum number of characters is 23 (114 / 4.8 = 23.75) in the Alpha (Base 27) encoding mode. You can find the encoding density in "Encoding modes" section above.

You can use the MinSize and the MaxSize properties to specify the minimum and maximum sizes for a Data Matrix (ECC 000 - 140) symbol. They can be one of values from dmSize_09_09 to dmSize_49_49 (defined in the pDataMatrix unit), corresponding to every symbol size shown above. The smallest symbol size that accommodates the barcode text will be automatically selected between minimum and maximum symbol sizes.

If the barcode text does not fill the maximum data capacity of the Data Matrix (ECC 000 - 140) symbol, remaining data capacity of the symbol will be filled by using pad bits (the ECCLevelUpgrade property is set to false), or will be used to upgrade the error correction code level (the ECCLevelUpgrade property is set to true). If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxSize property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur. The CurrentSize property can be used to get the factual symbol size.

Quiet zones

The minimum quiet zone is equal to 1 module on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 1. For applications with moderate to excessive reflected noise in close proximity to the symbol, a quiet zone of 2 modules to 4 modules is recommended.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Inversed
- MinSize
- MaxSize
- EncodeMode
- ECCLevel
- ECCLevelUpgrade
- CurrentSize (Read only)
- CurrentECCLevel (Read only)
- CurrentEncodeMode (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.9 TBarcode2D_DataMatrixECC200

The component is used to create the Data Matrix (ECC 200) 2D barcode symbols. It's defined in the pDataMatrixEcc200 unit.

Data Matrix code is a two-dimensional matrix barcode symbology consisting of black and white "cells" or modules arranged in either a square or rectangular pattern. The information to be encoded can be text or raw data. Usual data size is from a few bytes up to 2 kilobytes. The length of the encoded data depends on the symbol dimension used. Error correction codes are added to increase symbol strength: even if they are partially damaged, they can still be read.

Data Matrix was invented by International Data Matrix, Inc. (ID Matrix) which was merged into RVSI/Acuity CiMatrix, who were

acquired by Siemens AG in October, 2005 and Microscan Systems in September 2008.

There are 2 types of Data Matrix symbology, namely ECC 000 - 140 and ECC 200. The component can be used to generate the ECC 000 - 140 symbols. It is the newest version of Data Matrix and supports advanced encoding error checking and correction algorithms. It allows the routine reconstruction of the entire encoded data string when the symbol has sustained 30% damage, assuming the matrix can still be accurately located. So it's recommended for new application or open systems.

ECC 200 symbols have an even number of rows and an even number of columns. Some symbols are square with sizes from 10 * 10 to 144 * 144 not including quiet zones. Some symbols are rectangular with sizes from 8 * 18 to 16 * 48 not including quiet zones. All ECC 200 symbols can be recognised by the upper right corner module being light.

If you want to generate the Data Matrix ECC (000 - 140) symbols, please use another componentTBarcode2D_DataMatrix in this components package.

Shapes

There are two shapes of Data Matrix (ECC 200) symbols, square and rectangle, as described in following list:

- **Square**: Indicates to generate the square symbols.

- **Rectangle**: Indicates to generate the rectangle symbols.

You can use the Shape property to specify which shape of symbol will be selected to generate the barcode symbol. It can be one of values dsSquare and dsRectangle, corresponding to the shapes square and rectangle. These values are defined in the pDataMatrixEcc200 unit.

Symbol sizes

Data Matrix (ECC 200) symbols have an even number of rows and an even number of columns. The sizes of the square symbols are from 10 * 10 to 144 * 144 (modules) not including quiet zones. The sizes of the rectangle symbols are from 8 * 18 to 16 * 48 (modules) not including quiet zone.

- The sizes of square symbols (excluding quiet zones) and their maximum data capacities are listed in following table:

Symbol size values	Symbol sizes (modules)	Maximum data capacities		
		Numeric	Alphanumeric	Byte
dmSize_10_10	10 * 10	6	3	1
dmSize_12_12	12 * 12	10	6	3
dmSize_14_14	14 * 14	16	10	6
dmSize_16_16	16 * 16	24	16	10
dmSize_18_18	18 * 18	36	25	16
dmSize_20_20	20 * 20	44	31	20
dmSize_22_22	22 * 22	60	43	28
dmSize_24_24	24 * 24	72	52	34
dmSize_26_26	26 * 26	88	64	42
dmSize_32_32	32 * 32	124	91	60
dmSize_36_36	36 * 36	172	127	84
dmSize_40_40	40 * 40	228	169	112
dmSize_44_44	44 * 44	288	214	142
dmSize_48_48	48 * 48	348	259	172
dmSize_52_52	52 * 52	408	304	202
dmSize_64_64	64 * 64	560	418	277
dmSize_72_72	72 * 72	736	550	365
dmSize_80_80	80 * 80	912	682	453
dmSize_88_88	88 * 88	1152	862	573
dmSize_96_96	96 * 96	1392	1042	693
dmSize_104_104	104 * 104	1632	1222	813
dmSize_120_120	120 * 120	2100	1573	1047
dmSize_132_132	132 * 132	2608	1954	1301
dmSize_144_144	144 * 144	3116	2335	1555

- The sizes of rectangle symbols (excluding quiet zones) and their maximum data capacities are listed in following table:

Symbol size values	Symbol sizes (modules)	Maximum data capacities		
		Numeric	Alphanumeric	Byte
dmSize_8_18	8 * 18	10	6	3
dmSize_8_32	8 * 32	20	13	8
dmSize_12_26	12 * 26	32	22	14
dmSize_12_36	12 * 36	44	31	20
dmSize_16_36	16 * 36	64	46	30
dmSize_16_48	16 * 48	98	72	47

You can use the MinSize and the MaxSize properties to specifiy the minimum and maximum sizes for a Data Matrix (ECC 200) symbol. They can be one of values from dmSize_10_10 to dmSize_16_48 (they are defined in the pDataMatrixEcc200 unit), corresponding to every symbol size shown above. The smallest symbol size that accommodates the barcode text will be automatically selected between minimum and maximum symbol sizes.

If the barcode text does not fill the maximum data capacity of the Data Matrix (ECC 200) symbol, remaining data capacity of the symbol will be filled by using PAD characters. If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxSize property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event

will occur. The CurrentSize property can be used to get the factual symbol size.

Quiet zones

The minimum quiet zone is equal to 1 module on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 1. For applications with moderate to excessive reflected noise in close proximity to the symbol, a quiet zone of 2 modules to 4 modules is recommended.

Error checking and correcting (ECC)

Data Matrix (ECC 200) symbols are fixed at a repair level of about 25% damage and overhead ranges from 60% for a small number of characters downward to 26% for a large number of characters encoded.

Character set

- All 8-bit values can be encoded. The default interpretation shall be:
  - For values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646 (ASCII); Note this version consists of the GO set of ISO/IEC 646 and the CO set of ISO/IEC 6429 with values 28 to 31 modified to FS, GS, RS and U respectively).
  - For values 128 - 255, in accordance with ISO/IEC 8859-1 (Extended ASCII).

  This interpretation corresponds to ECI 000003.
- Two non-data characters can be encoded, FNC1 for compatibility with some existing applications and ECI indicator blocks for the standardized encoding of message interpretation information.
  - **FNC1**: The FNC1 character following an application standard agreed with AIM International, identifies a symbol which conforms to a specific industry standard. FNC1 shall be used as defined in the EAN.UCC General Specifications in the first position. The FNC1 character may also be used as a field separator, in which case it will be represented in the transmitted message as GS character (ASCII value 29). The escape sequence "\f" can be used to placed the FNC1 character to barcode text.
  - **ECI**: The escape sequence "\e[<ECI_Number>]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

  The AllowEscape property should be set to true in order to place the FNC1 characters or the ECI indicator blocks.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\5**: Insert a 05 macro to barcode text. It's used to abbreviate the industry specific header "[)>{RS}05{GS}" and trailer "{RS}{EOT}", in order to reduce the size of symbol. It must be placed at beginning of the barcode text and it shall not be used in conjunction with structured append.
- **\6**: Insert a 06 macro to barcode text. It's used to abbreviate the industry specific header "[)>{RS}06{GS}" and trailer "{RS}{EOT}", in order to reduce the size of symbol. It must be placed at beginning of the barcode text and it shall not be used in conjunction with structured append.
- **\f**: Insert a FNC1 character to barcode text. See also the "Character set" section above.
- **\e[<ECI_Number>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.
- **\s[<Index>, <Amount>, <File_Id>]** Insert a structured append block to the barcode text in order to create the symbol in a structured append. See also the "Structured append" section below.
- **\r**: Indicates that the symbol encodes a message used to program the reader system. It shall be placed at beginning of the barcode text and it shall not be used with structured append.

Note, the "{RS}" is ASCII character RS (ASCII value 30), the{GS}" is ASCII character GS (ASCII value 29), and the{EOT}" is ASCII character EOT (ASCII value 4).

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set. Four broad types of interpretations are supported in Data Matrix (ECC 200):

- International character sets (or code pages).
- General purpose interpretations such as encryption and compaction.
- User defined interpretations for closed systems.
- Control information for structured append in unbuffered mode.

The ECI protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding. The ECI is identified by an integer (up to 6 digits) which is encoded in the Data Matrix (ECC 200) symbol by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 999999 (including the boundaries), the leading zero is optional.

ECI indicator blocks may be placed anywhere in the barcode text in a single or structured append set of Data Matrix (ECC 200) symbols. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Structured append

In order to handle larger messages than are practical in a single symbol, a data message can be distributed across several Data Matrix (ECC 200) symbols. Up to 16 Data Matrix (ECC 200) symbols may be appended in a structured format to convey more data. If a symbol is part of a structured append this shall be indicated by a structured append block in barcode text. The escape sequence "\s[<Index>, <Amount>, <File_Id>]" is used to place the structured append block to the barcode text:

- **Index**: The position index of the symbol within the set of Data Matrix (ECC 200) symbols in the structured append format. It's an integer between 1 and 16 (including the boundaries) in string format.
- **Amount**: The total amount of the symbol within the set of Data Matrix (ECC 200) symbols in the structured append format. It's an integer between 2 and 16 (including the boundaries) in string format.
- **File_Id**: The file identification. It's an integer between 1 and 64516 (including the boundaries) in string format. The purpose of the file identification is to increase the probability that only logically linked symbols are processed as part of the same message. Also the field can be expressed as two integers between 1 and 254 (including the boundaries), separated with a comma.

The AllowEscape property should be set to true in order to place the structured append block. The structured append block may only be placed once in the barcode text. Also, it shall be placed at beginning of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the structured append block be placed more than once, or it isn't placed at beginning of the barcode text. The following is the examples of structured append:

```
\s[2, 5, 25431]ABCDEFGabcdefg1234567890
\s[2, 5, 101, 31]ABCDEFGabcdefg1234567890
```

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Inversed
- MinSize
- MaxSize
- Shape
- CurrentSize (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.10 TBarcode2D_GridMatrix

The component is used to create the Grid Matrix 2D barcode symbols. It's defined in the pGridMatrix unit.

Grid Matrix is a square, variable-sized, two-dimensional matrix symbology with unique dark- and light-framed "macromodules" that create a grid design that provides a robust finder pattern. The unique finder pattern ensures that readers can locate and orient the symbol even with significant symbol damage.

Grid Matrix can encode 7-bit ASCII, numeric, and binary data, in addition to any combination of data types in the same symbol, particularly effective with Chinese characters.

Grid Matrix was invented by Syscan Technology Co., Ltd.

Symbol sizes

There are thirteen sizes of Grid Matrix symbol, referred to as version 1 to 13, in increasing order of size and data capacity. You can

use the MinVersion and the MaxVersion properties to specifiy the minimum and maximum sizes for a Grid Matrix symbol. They can be one of values from 1 to 13 (defined in the pGridMatrix unit), corresponding to the versions 1 to 13. The smallest symbol size that accommodates the barcode text will be automatically selected between minimum and maximum symbol sizes. The CurrentVersion property can be used to get the factual symbol size.

The symbol sizes and the maximum data capacity of each version are listed in following table:

Version	Symbol sizes (modules)	Maximum data capacities (bits)				
		ECC Level 1	ECC Level 2	ECC Level 3	ECC Level 4	ECC Level 5
1	18 * 18	-	105	91	77	63
2	30 * 30	315	280	245	210	175
3	42 * 42	623	553	483	413	343
4	54 * 54	1022	910	798	686	567
5	66 * 66	1526	1358	1190	1022	847
6	78 * 78	2135	1897	1659	1421	1183
7	90 * 90	2835	2520	2205	1890	1575
8	102 * 102	3647	3241	2835	2429	2023
9	114 * 114	4550	4046	3542	3038	2527
10	126 * 126	5558	4942	4326	3710	3087
11	138 * 138	6671	5929	5187	4445	3703
12	150 * 150	7875	7000	6125	5250	4375
13	162 * 162	9191	8169	7147	6125	5103

If the barcode text does not fill the maximum data capacity of the Grid Matrix symbol, remaining data capacity of the symbol will be filled by using pad bits (the ECCLevelUpgrade property is set to false), or will be used to upgrade the error correction code level (the ECCLevelUpgrade property is set to true). If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxVersion property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Error correction code (ECC)

There are five user-selectable levels of error correction, from 1 to 5 respectively in increasing order of recovery capacity.

You can use the ECCLevel property to specify the error correction code level for a Grid Matrix symbol. It can be one of these values: elLevel_1, elLevel_2, elLevel_3, elLevel_4, and elLevel_5 (they are defined in the pGridMatrix unit), corresponding to the ECC levels from 1 to 5.

These ECC levels are listed in following table:

Value of ECCLevel property	Error correction code level	Percentage of total capacity for ECC data
elLevel_1	1	10%
elLevel_2	2	20%
elLevel_3	3	30%
elLevel_4	4	40%
elLevel_5	5	50%

Note, the ECC level 1 is inapplicable to the version 1.

Also, you can set the ECCLevel property to elLevel_Recommend or elLevel_LowestRecommend. For each symbol version, the denotative ECC levels are listed in following table:

Version	Denotative ECC level	
	ECCLevel = elLevel_Recommend	ECCLevel = elLevel_LowestRecommend
1	5	4
2	4	2
3	4	1
4-13	3	1

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will be used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by the ECCLevel property (see also the "Symbol sizes" section above). In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The property CurrentECCLevel can be used to get the factual error correction code level.

Quiet zones

The minimum quiet zone is equal to 6 modules on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 6.

Character set

- All 7-bit ASCII values can be encoded.
- 8-bit binary data can be encoded.
- GB 18030 Chinese characters can be encoded.
- Five non-data characters can be encoded:
  - **AIM FNC1** It identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. It is immediately followed by an application indicator assigned to identify the specification concerned by AIM International. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode. The escape sequence "\0" can be used to placed the FNC1 character to barcode text.
  - **GS1 FNC1** It identifies symbols encoding data formatted according to the GS1 Application Identifiers standard. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode text. The escape sequence "\1" can be used to placed the GS1 FNC1 character to barcode text.
  - **FNC2**: The FNC2 is used to implement structured append function in order to handle larger messages than are practical in a single symbol. The escape sequence "\2[<File_Id>, <Amount>, <Index>]" is used to place the FNC2 character and other structured append information (collectively referred to as structured append block) to barcode text. See also the "Escape sequences" and "Structured append" sections below.
  - **FNC3**: The FNC3 is used to indicates that the symbol encodes a message used to program the reader system. The escape sequence "\3" can be used to placed the FNC3 character to barcode text. See also the "Escape sequences" section below.
  - **ECI**: ECI indicator blocks is for the standardized encoding of message interpretation information. The escape sequence "\e[<ECI_Number>]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

  The AllowEscape property should be set to true in order to place these non-data characters.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\0**: Insert an AIM FNC1 character to barcode text in order to encode AIM structural data similar to GS1. It shall be placed at beginning of the barcode text and it shall not be used with GS1 FNC1 ("\1") or FNC3 ("\3"). If it is used together with the

structured append block (FNC2, "\2"), it shall be placed on front of the structured append block ("\2"). See also the "Structured append" section below.

- **\1**: Insert a GS1 FNC1 character to barcode text in order to encode GS1 structural data. It shall be placed at beginning of the barcode text and it shall not be used with AIM FNC1 ("\0") or FNC3 ("\3"). If it is used together with the structured append block ("\2"), it shall be placed on front of the structured append block (FNC2, "\2"). See also the "Structured append" section below.
- **\2[<File_Id>, <Amount>, <Index>]** Insert a structured append block (FNC2 and other structured append information) to the barcode text in order to create the symbol in a structured append. See also the "Structured append" section below.
- **\3**: Insert a FNC3 character to barcode text, indicates that the symbol encodes a message used to program the reader system. It shall be placed at beginning of the barcode text and it shall not be used with AIM FNC1 ("\0") or GS1 FNC1 ("\1"). If it is used together with the structured append block (FNC2, "\2"), it shall be placed on front of the structured append block (FNC2, "\2"). See also the "Structured append" section below.
- **\e[<ECI_Number>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set.

The ECI protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding. The ECI is identified by an integer (up to 6 digits) which is encoded in the Grid Matrix symbol by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 811799 (including the boundaries), the leading zero is optional.

In a single symbol or structured append set of Grid Matrix symbols, if the AIM FNC1 ("\0"), GS1 FNC1 ("\1"), FNC2 (structured append block, "\2"), and FNC3 ("\3") are used, ECI indicator blocks may be placed anywhere behind of them, otherwise, they may be placed anywhere in the barcode text. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Structured append

In order to handle larger messages than are practical in a single symbol, a data message can be distributed across several Grid Matrix symbols. Up to 16 Grid Matrix symbols may be appended in a structured format to convey more data. If a symbol is part of a structured append this shall be indicated by a structured append block in barcode text. The escape sequence "\2[<File_Id>, <Amount>, <Index>]" is used to place the structured append block to the barcode text:

- **File_Id**: The file identification. It's an integer between 0 and 255 (including the boundaries) in string format. The purpose of the file identification is to increase the probability that only logically linked symbols are processed as part of the same message.
- **Amount**: The total amount of the symbol within the set of Grid Matrix symbols in the structured append format. It's an integer between 1 and 16 (including the boundaries) in string format.
- **Index**: The position index of the symbol within the set of Grid Matrix symbols in the structured append format. It's an integer between 1 and 16 (including the boundaries) in string format.

The AllowEscape property should be set to true in order to place the structured append block. The structured append block may only be placed once in the barcode text. If the structured append block is used together with AIM FNC1 ("\0"), GS1 FNC1 ("\1") or FNC3 ("\3"), the AIM FNC1 ("\0"), GS1 FNC1 ("\1") or FNC3 ("\3") shall be placed at beginning of the barcode text, then the structured append block. Otherwise, the structured append block shall be placed at beginning of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the structured append block be placed more than once, or it isn't placed at beginning of the barcode text or behind of the AIM FNC1 ("\0"), GS1 FNC1 ("\1") or FNC3 ("\3"). The

following is examples of structured append:

```
\2[79, 5, 2]ABCDEFGabcdefg1234567890...
\0\2[99, 6, 1]ASDFGHJKL098765...
```

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- AllowEscape
- MinVersion
- MaxVersion
- ECCLevel
- ECCLevelUpgrade
- CurrentVersion (Read only)
- CurrentECCLevel (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.11 TBarcode2D_HanXinCode

The component is used to create the Han Xin Code 2D barcode symbols. It's defined in the pHanXinCode unit.

Han Xin Code is also known as Chinese Sensible Code, It is a unique, variable size, matrix symbology specifically optimized for two and four byte alphabets such as Chinese and other ideographic/pictographic alphabets. It is equally suitable for single byte ISO Code Pages such as English.

Han Xin Code

Han Xin Code also includes an option for octet byte encoding for applications such as graphics and audio. Extended Channel Interpretation (ECIs) support is also included. It incorporates strong Reed-Solomon error correction to enable four levels of error correction abilities to recover information from damaged symbols.

Symbol sizes

There are eighty-four sizes of Han Xin Code symbols, referred to as version 1 to 84, in increasing order of size and data capacity. You can use the MinVersion and the MaxVersion properties to specifiy the minimum and maximum sizes for a Han Xin Code symbol. They can be one of values from 1 to 84 (defined in the pHanXin unit), corresponding to the versions 1 to 84. The smallest symbol size that accommodates the barcode text will be automatically selected between minimum and maximum symbol sizes. The CurrentVersion property can be used to get the factual symbol size.

If the barcode text does not fill the maximum data capacity of the Han Xin Code symbol, remaining data capacity of the symbol will be filled by using pad bits (the ECCLevelUpgrade property is set to false), or will be used to upgrade the error correction code level (the ECCLevelUpgrade property is set to true). If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxVersion property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Error correction code (ECC)

There are four user-selectable levels of error correction, from L1 to L4 respectively in increasing order of recovery capacity.

You can use the ECCLevel property to specify the error correction code level for a Han Xin Code symbol. It can be one of these values: eILevel_1, eILevel_2, eILevel_3, and eILevel_4 (they are defined in the pHanXinCode unit), corresponding to the ECC levels from L1 to L4.

These ECC levels are listed in following table:

Value of ECCLevel property	Error correction code level	Recevory capacities (%) (approx.)
eILevel_1	L1	8%
eILevel_2	L2	15%
eILevel_3	L3	23%
eILevel_4	L4	30%

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will be used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by the ECCLevel property (see also the "Symbol sizes" section above). In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The property CurrentECCLevel can be used to get the factual error correction code level.

Quiet zones

The minimum quiet zone is equal to 3 modules on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 3.

Character set

- All 7-bit ASCII values can be encoded.
- 8-bit binary data can be encoded.
- GB 18030 Chinese characters can be encoded.
- One non-data characters can be encoded:
    - **ECI**: ECI indicator blocks is for the standardized encoding of message interpretation information. The escape

sequence "\e[<ECI_Number>]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

The AllowEscape property should be set to true in order to place non-data characters.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\e[<ECI_Number>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set.

The ECI protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding. The ECI is identified by an integer (up to 6 digits) which is encoded in the Han Xin Code symbol by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 999999 (including the boundaries), the leading zero is optional.

In a Han Xin Code symbol, the ECI indicator blocks may be placed anywhere in the barcode text. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- AllowEscape
- MinVersion
- MaxVersion
- ECCLevel
- ECCLevelUpgrade
- ReviseVersion5
- CurrentVersion (Read only)
- CurrentECCLevel (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

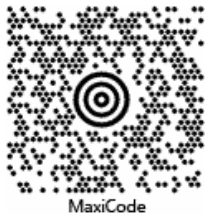
- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.12 TBarcode2D_MaxiCode

The component is used to create the MaxiCode 2D Barcode symbols. It's defined in the pMaxiCode unit.

MaxiCode is a public domain, machine-readable symbol system originally created and used by United Parcel Service in 1992. It was designed for tracking and managing the shipment of packages, and can be quickly automatically scanned on high-speed conveyor lines.



MaxiCode is a fixed-size symbology. A MaxiCode symbol appears as a 1.11 * 1.054 inchs square, with a bullseye in the middle, surrounded by a pattern of hexagonal dots (modules). Each hexagonal dot (module) measures 0.035 * 0.041 inches.

Modes

MaxiCode has modes which are used to define the format of the message and the level of error correction within a symbol:

- **Mode 2**: It encodes a structured carrier message with a numeric postal code, and an optional secondary message. A numeric postal code, a country code, and a class of service code assigned by the carrier are included in the structured carrier message.

  The symbol employs enhanced error correction for the structured carrier message and standard error correction for the secondary message, see also the "Error checking and correcting (ECC)" section below.

  It is designed for use in the transport industry. Primary use is US domestic destinations.

  The formats of mode 2 barcode text are specified in the following list:

  - **[)>{RS}01{GS}YYPPPPPPPPP{GS}CCC{GS}SSS{GS}MESSAGE{RS}{EOT}**
    - **YY**: Date (year), 2 digits.
    - **PPPPPPPPP**: Numeric postal code, up to 9 digits. Only first 9 digits will be encoded if its length is greater than 9 digits.
    - **CCC**: Country code, 3 digits.
    - **SSS**: Serivce class, 3 digits
    - **MESSAGE**: Optional secondary message. All 256 of the ASCII characters can be used. A maximum of about 71 alphanumeric characters or about 107 digits can be encoded in the message.
    - **{RS}**: ASCII character RS (ASCII value 30). The RS character after the secondary message is optional. If the AllowEscape property is set to true, you can use the "\r" instead of the RS character.
    - **{GS}**: ASCII character GS (ASCII value 29). It's used to delimit each field in barcode text. The GS character after the service class can be omitted if secondary message doesn't exist. If the AllowEscape property is set to true, you can use the "\g" instead of the GS character.
    - **{EOT}**: Optional ASCII character EOT (ASCII value 4). If the AllowEscape property is set to true, you can use the "\t" instead of the EOT character.

    For example (Date: 96; Postal code: 123456789; Country code: 840; Service class: 001; Secondary message: 'PA USA' + Chr(29) + 'UPS\123'):

    - ```
      '[)>' + Chr(30) + '01' + Chr(29) + '96123456789' + Chr(29) + '840' + Chr(29)
       + '001' + Chr(29) + 'PA USA' + Chr(29) + 'UPS\123' + Chr(30) + Chr(4)
      ```
    - ```
      '[)>\r01\g96123456789\g840\g001\gPA USA\gUPS\\123\r\t'
      ```

  - **PPPPPPPPP{GS}CCC{GS}SSS{GS}MESSAGE{EOT}**
    - **PPPPPPPPP**: Numeric postal code, up to 9 digits. Only first 9 digits will be encoded if its length is greater than 9 digits.
    - **CCC**: Country code, 3 digits.
    - **SSS**: Serivce class, 3 digits
    - **MESSAGE**: Optional secondary message. All 256 of the ASCII characters can be used. A maximum of about 82 alphanumeric characters or about 121 digits can be encoded in the message.

- **{GS}**: ASCII character GS (ASCII value 29). It's used to delimit each field in barcode text. The GS character after the service class can be omitted if secondary message doesn't exist. If the AllowEscape property is set to true, you can use the "\g" instead of the GS character.
- **{EOT}**: Optional ASCII character EOT (ASCII value 4). If the AllowEscape property is set to true, you can use the "\t" instead of the EOT character.

For example (Postal code: 123456789; Country code: 840; Service class: 001; Secondary message: 'PA USA' + Chr(29) + 'UPS\123'):

- `'123456789' + Chr(29) + '840' + Chr(29) + '001' + Chr(29) + 'PA USA' + Chr(29) + 'UPS\123' + Chr(4)`
- `'123456789\g840\g001\gPA USA\gUPS\\123\t'`

- **Mode 3**: It encodes a structured carrier message with an alphanumeric postal code, and an optional secondary message. An alphanumeric postal code, a country code, and a class of service code assigned by the carrier are included in the structured carrier message.

  The symbol employs enhanced error correction for the structured carrier message and standard error correction for the secondary message, see also the "Error checking and correcting (ECC)" section below.

  It is designed for use in the transport industry. Primary use is international destinations.

  The mode 3 is similar to mode 2, but the postal code field encodes 6 alphanumeric characters.

  The formats of mode 3 barcode text are specified in the following list:

  - **[)>{RS}01{GS}YYPPPPPP{GS}CCC{GS}SSS{GS}MESSAGE{RS}{EOT}**
    - **YY**: Date (year), 2 digits.
    - **PPPPPP**: Alphanumeric postal code, up to 6 alphanumeric characters. Only first 6 characters will be encoded if its length is greater than 6 characters. All upper case letters, all numeric characters, space character, and the punctuations and symbols corresponding to the ASCII values from 34 to 58 can be encoded.
    - **CCC**: Country code, 3 digits.
    - **SSS**: Serivce class, 3 digits
    - **MESSAGE**: Optional secondary message. All 256 of the ASCII characters can be used. A maximum of about 71 alphanumeric characters or about 107 digits can be encoded in the message.
    - **{RS}**: ASCII character RS (ASCII value 30). The RS character after the secondary message is optional. If the AllowEscape property is set to true, you can use the "\r" instead of the RS character.
    - **{GS}**: ASCII character GS (ASCII value 29). It's used to delimit each field in barcode text. The GS character after the service class can be omitted if secondary message doesn't exist. If the AllowEscape property is set to true, you can use the "\g" instead of the GS character.
    - **{EOT}**: Optional ASCII character EOT (ASCII value 4). If the AllowEscape property is set to true, you can use the "\t" instead of the EOT character.

    For example (Date: 96; Postal code: ABC12; Country code: 840; Service class: 001; Secondary message: 'PA USA' + Chr(29) + 'UPS\123'):

    - `'[)>' + Chr(30) + '01' + Chr(29) + '96ABC12' + Chr(29) + '840' + Chr(29) + '001' + Chr(29) + 'PA USA' + Chr(29) + 'UPS\123' + Chr(30) + Chr(4)`
    - `'[)>\r01\g96ABC12\g840\g001\gPA USA\gUPS\\123\r\t'`
  - **PPPPPP{GS}CCC{GS}SSS{GS}MESSAGE{EOT}**
    - **PPPPPP**: Alphanumeric postal code, up to 6 alphanumeric characters. Only first 6 characters will be encoded if its length is greater than 6 characters. All upper case letters, all numeric characters, space character, and the

punctuations and symbols corresponding to the ASCII values from 34 to 58 can be encoded.

- **CCC**: Country code, 3 digits.
- **SSS**: Serivce class, 3 digits
- **MESSAGE**: Optional secondary message. All 256 of the ASCII characters can be used. A maximum of about 82 alphanumeric characters or about 121 digits can be encoded in the message.
- **{GS}**: ASCII character GS (ASCII value 29). It's used to delimit each field in barcode text. The GS character after the service class can be omitted if secondary message doesn't exist. If the AllowEscape property is set to true, you can use the "\g" instead of the GS character.
- **{EOT}**: Optional ASCII character EOT (ASCII value 4). If the AllowEscape property is set to true, you can use the "\t" instead of the EOT character.

For example (Postal code: ABC12; Country code: 840; Service class: 001; Secondary message: 'PA USA' + Chr(29) + 'UPS\123'):

- `'ABC12' + Chr(29) + '840' + Chr(29) + '001' + Chr(29) + 'PA USA' + Chr(29) + 'UPS\123' + Chr(4)`
- `'ABC12\g840\g001\gPA USA\gUPS\\123\t'`

- **Mode 4**:

  Indicates that the symbol encodes an unformatted message. The barcode message is divided into a primary message and a secondary message internally, the symbol employs enhanced error correction for the primary message and standard error correction for the secondary message.

  All 256 of the ASCII characters can be used. A maximum of about 93 alphanumeric characters or about 138 digits can be encoded in the symbol.

- **Mode 5**:

  Indicates that the symbol encodes an unformatted message. The barcode message is divided into a primary message and a secondary message internally, the symbol employs enhanced error correction for both the primary and secondary messages.

  All 256 of the ASCII characters can be used. A maximum of about 77 alphanumeric characters or about 113 digits can be encoded in the symbol.

  The mode 5 is similar to mode 4, but it employs enhanced error correction for secondary messages.

- **Mode 6**:

  Indicates that the symbol encodes a message used to program the reader system. The barcode message is divided into a primary message and a secondary message internally, the symbol employs enhanced error correction for the primary message and standard error correction for the secondary message.

  All 256 of the ASCII characters can be used. A maximum of about 93 alphanumeric characters or about 138 digits can be encoded in the symbol.

You can use the Mode property to specify the factual mode for a MaxiCode symbol. It can be one of values from 2 to 6, corresponding to the modes from 2 to 6. They are defined in the pMaxiCode unit.

Also, you can set the AutoMode property to true, in order to automatically select the suitable mode depending on the barcode text and the value of Mode property. And use the CurrentMode property to get the factual mode:

- If the Mode property is set to mode 2 or 3, the factual mode will be selected between mode 2 and 3, depending on the postal code field in the barcode text.
- If the Mode property is set to mode 4, the factual mode will be selected between mode 4 and 5, depending on the length of barcode text. If the length of barcode text is so short that can be encoded by using mode 5, the mode 5 will be used in order to employ enhanced error correction, otherwise the mode 4 will be used in order to accommodate more barcode text.
- If the Mode property is set to mode 5, the mode 5 will be selected always, in order to insure high level of enhanced error

correction.

- If the Mode property is set to mode 6, the mode 6 will be selected always, in order to encodes a message used to program the reader system.

Symbol size

Each MaxiCode symbol is of a fixed size, having 884 hexagonal modules arranged in 33 rows around a central finder pattern. Each row consists of a maximum of 30 modules.

Also, each symbol, including the quiet zones, is of a fixed physical size, nominally 28.14mm wide, 26.91mm high. Each hexagonal dot (module) measures 0.889 * 1.041 millimeters.

Quiet zones

The MaxiCode symbology require the minimum 1 module quiet zones as measured from the outside edges. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 1.

Error checking and correcting (ECC)

MaxiCode symbology offer two levels of error checking and correction, Enhanced Error Correction (EEC) and Standard Error Correction (SEC), which are specified by the Mode, see also the "Modes" section above.

In mode 2 and 3, the symbol employs EEC for the structured carrier message and SEC for the secondary message. In mode 4, 5, and 6, MaxiCode symbols are internally divided into a primary message and a secondary message, the mode 4 and 6 symbol employ EEC for the primary message and SEC for the secondary message. the mode 5 symbol employs EEC for both the primary and secondary messages.

Character set

In mode 2, all numeric characters can be encoded in the country code, service class, and postal code fields. All 256 of the ASCII characters can be encoded in the secondary message.

In mode 3, all numeric characters can be encoded in the country code and service class fields, all upper case letters, all numeric characters, space character, and punctuations and symbols corresponding to the ASCII values from 34 to 58 can be encoded in the postal code field. All 256 of the ASCII characters can be encoded in the secondary message.

In mode 4, 5 and 6, All 256 of the ASCII characters can be encoded in the entire message.

The default interpretation of these characters shall be:

- Values 0-127, in accordance with ANSI X3.4, i.e. all 128 ASCII characters
- values 128-255 in accordance with ISO 8859-1: Latin Alphabet No. 1, i.e. extended ASCII characters.

This default interpretation corresponds to ECI 000003.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\f**: Insert a FS character (ASCII value 28) to barcode text. See also the "Modes" section above.
- **\g**: Insert a GS character (ASCII value 29) to barcode text. See also the "Modes" section above.
- **\r**: Insert a RS character (ASCII value 30) to barcode text. See also the "Modes" section above.
- **\t**: Insert a EOT character (ASCII value 4) to barcode text. See also the "Modes" section above.
- **\e[<ECI_Number>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.
- **\s[<Index>, <Amount>]**: Insert a structured append block to barcode text in order to create the symbol in a structured

append. See also the "Structured append" section below.

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set. Four broad types of interpretations are supported in MaxiCode:

- International character sets (or code pages).
- General purpose interpretations such as encryption and compaction.
- User defined interpretations for closed systems.
- Control information for structured append in unbuffered mode.

The ECI protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding. The ECI is identified by an integer (up to 6 digits) which is encoded in the MaxiCode symbol by the ECI indicator block. The escape sequence "\e[ECI_Number]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 999999 (including the boundaries), the leading zero is optional.

ECI indicator blocks may be placed anywhere in the barcode text in a single or structured append set of MaxiCode symbols, but cannot be within the primary message (structured carrier message) for modes 2 and 3:

- In mode 2 and 3, the ECI blocks may only be invoked within secondary message. For example:

```
[)>01\r01\g99123456789\g123\g456\g\e[002]ABCg\e[123]DEF\r\t
```

- In mode 4, 5 and 6, the ECI blocks may be invoked anywhere in the barcode text. For example:

```
ABC\e[23]DEFG\e[001]HIJKLMN
```

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Structured append

Up to eight MaxiCode symbols may be appended in a structured format to convey more data. If a symbol is part of a structured append this shall be indicated by a structured append block in barcode text. The escape sequence "\s[<Index>, <Amount>]" is used to place the structured append block to the barcode text:

- **Index**: The position index of the symbol within the set of MaxiCode symbols in the structured append format. It's an integer between 1 and 8 (including the boundaries) in string format.
- **Amount**: The total amount of the symbol within the set of MaxiCode symbols in the structured append format. It's an integer between 2 and 8 (including the boundaries) in string format.

The structured append block may only be placed once in the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the structured append block be placed more than once. TheAllowEscape property should be set to true in order to place the structured append block. The valid locations for structured append block in the barcode text are specified in the following:

- In the mode 2 and mode 3 of MaxiCode symbols, the structured append block shall be placed at beginning or end in the barcode text, or anywhere in the secondary message. For example:

```
\s[2,5][)>01\r01\g99123456789\g123\g456\gABCDEF\r\t
[)>01\r01\g99123456789\g123\g456\g\s[2,5]ABCDEF\r\t
[)>01\r01\g99123456789\g123\g456\gABC\s[2,5]DEF\r\t
[)>01\r01\g99123456789\g123\g456\gABCDEF\s[2,5]\r\t
[)>01\r01\g99123456789\g123\g456\gABCDEF\r\t\s[2,5]
```

- In the mode 4, 5 and 6 of MaxiCode symbols, the structured append block can be placed at anywhere in the barcode text. For

example:

```
\s[2,5]ABCDEFG
ABC\s[2,5]DEFG
ABCDEFG\s[2,5]
```

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Mode
- AutoMode
- AllowEscape
- CurrentMode (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.13 TBarcode2D_MicroPDF417

The component is used to create the MicroPDF417 Barcode symbols. It is defined in the pMicroPDF417 unit.

MicroPDF417 is a multi-row symbology, derived from and closely based onPDF417. MicroPDF417 is designed for applications with a need for improved area efficiency but without the requirement for PDF417's maximum data capacity. A limited set of symbol sizes is available, together with a fixed level of error correction for each symbol size. Module dimensions are user-specified to enable symbol production and reading by a wide variety of techniques.

Symbol sizes

MicroPDF417 symbols shall conform with certain predefined combinations of number of stacked rows, columns, and number of error-correction codewords, referred to as symbol sizes. The symbol size values from mpSize_1_11 to mpSize_4_44 corresponding to these symbol sizes, and denotation the number of columns and rows in every symbol size. For example, the symbol size value mpSize_2_23 denotations the MicroPDF417 symbol size is 23 stacked rows by 2 columns. These symbol size values are defined in the pMicroPDF417 unit.

The RowHeight property can be used to specify the height for each row, in modules.

You can use the MinSize property to specify a minimum symbol size, and use the MaxSize property to specifiy a maximum symbol size. According to the priority order specified by the StretchOrder property, the first symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

You can use the CurrentSize property to get the factual symbol size.

If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxSize property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Quiet zones

The symbol shall include a quiet zone on all four sides. The minimum quiet zone is equal to 1 module on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 1.

Error checking and correcting (ECC)

Each MicroPDF417 symbol contains at least seven error correction codewords. The error correction codewords provide capability for both error detection and correction. The number of error correction codewords for a MicroPDF417 symbol is fixed for each symbol size.

Character set

- All 8-bit values can be encoded. The default interpretation shall be:
  - For values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646 (ASCII).
  - For values 128 - 255, in accordance with ISO/IEC 8859-1 (Extended ASCII).

  This interpretation corresponds to ECI 000003.

  MicroPDF417 has three unique data compaction modes as Text compaction mode, Numeric compaction mode, and Byte compaction mode. All 8-bit values are encoded by switching internally between all 3 compaction modes:

  - **Text compaction mode**: Permits all printable ASCII characters, i.e. ASCII values from 32 to 126 inclusive accordance with ISO/IEC 646, as well as selected control characters (CR, HT, and LF).

    The mode has four sub-modes:

    - **Alpha**: Uppercase letters and space.
    - **Lower**: Lowercase letters and space.
    - **Mixed**: Numeric and some punctuations.
    - **Punctuation**: Some punctuations.

    The sub-mode will be switched automatically in order to minimize the symbol size.
  - **Numeric compaction mode**: Permits efficient encoding of numeric data string.
  - **Byte compaction mode**: Permits all 256 possible 8-bit byte values to be encoded. This includes all ASCII characters value from 0 to 127 inclusive and provides for international character set support.

  In the specification of MicroPDF417, the initial compaction mode in effect at the start of each symbol shall always be Byte compaction mode. You can change the initial compaction mode by using the DefaultEncodeMode and DefaultTextEncodeMode properties, in order to macth the reader. If the poMicroPDF417Explicit901 value is included in the

Options property, a mode latch to Byte compaction mode "\901" will be explicitly inserted into beginning of barcode text, in order to work with any reader.

The compaction mode will be switched automatically in a MicroPDF417 symbol in order to minimize the symbol size. Also, you can manually switch the code set by using escape sequences "\900", "\901", "\902", and "\924". See also the "Escape sequences" section below.

- The FNC1 characters can be encoded for compatibility with some existing applications. There are two mode FNC1 characters to identify symbols encoding messages formatted according to specific predefined industry or application specifications, also the FNC1 character can be used as the data field separator:
  - **First position mode**: FNC1 in this mode indicator identifies symbols encoding data formatted according to the GS1 Application Identifiers standard. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode text.

    If any one of escape sequences in "\903", "\904" and "\905" followed by an Application Identifier (2 or more digits) is placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode, and implies a mode latch to Text compaction mode or Numeric compaction mode. See also the "Escape sequences" section below.

    Also, you can place an escape sequence \f" followed by an Application Identifier (2 or more digits) at beginning of barcode text instead of the "\903", "\904" or "\905" escape sequence followed by the Application Identifier, to place the FNC1 character in "First position" mode. The component will automatically select one of escape sequences from "\903", "\904" and "\905" depending on on the barcode text, in order to minimize the symbol size.

    The escape sequences "\906", "\907", "\912", "\914" and "\915" can be placed the FNC1 character in "First position" mode too. See also the "Escape sequences" section below.
  - **Second position mode**: FNC1 in this mode indicator identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. It is immediately followed by an application indicator assigned to identify the specification concerned by AIM International. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode. An application indicator may take the form of any single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number.

    If any one of escape sequences in "\908" and "\909" followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) is placed at beginning of barcode text, it represents a leading FNC1 character in "Second position" mode, and implies a mode latch to Text compaction mode or Numeric compaction mode. See also the "Escape sequences" section below.
  - **Data field separator**: The FNC1 character may also be used as a data field separator(i.e. at the end of a variable-length data field), in which case it will be represented in the transmitted message as GS character (ASCII value 29). The data field separator shall not be placed at beginning of barcode text.

    If any one of escape sequences in "\903", "\904" and "\905" is placed in the barcode text, but not at beginning of the barcode text, it represents a data field separator, and implies a mode latch to Text compaction mode or Numeric compaction mode. See also the "Escape sequences" section below.

    Also, you can use the escape sequence "\f" instead of the "\903", "\904" or "\905" to placed the FNC1 character as the data field separator. The component will automatically select one of escape sequences from "\903", "\904" and "\905" based on the barcode text, in order to minimize the symbol size.

    The appearance of an adjacent pair of repeated "\903", "\904" or "\905" escape sequences will cause the decoder to divide the transmission at that point, but without transmitting the two implied GS (ASCII value 29) characters themselves.
- The ECI indicator blocks can be encoded for the standardized encoding of message interpretation information. The escape sequence "\e[<ECI_Numnerm>]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\5**: Insert a 05 macro to barcode text. It's used to abbreviate the industry specific header "[)>{RS}05{GS}" and trailer "{RS}{EOT}", in order to reduce the size of symbol. It must only be placed once at beginning of the barcode text and it shall not be used in conjunction with a series of structured append symbols.
- **\6**: Insert a 06 macro to barcode text. It's used to abbreviate the industry specific header "[)>{RS}06{GS}" and trailer "{RS}{EOT}", in order to reduce the size of symbol. It shall only be placed once at beginning of the barcode text and it shall not be used in conjunction with a series of structured append symbols.
- **\r**: Instructs the reader to interpret the data contained within the MicroPDF417 symbol as programming for reader initialisation. It shall only be placed once at beginning of the barcode text. In the case of a structured append initialisation sequence, it shall appear in every symbol.
- **\e[<ECI_Numnerm>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.
- **\f**: Insert a FNC1 character either in "First position" mode or as a data field separator, to the barcode text. One of "\903", "\904", and "\905" will be automatically selected instead of the "\f", to insert to the symbol, depending on the barcode text. If it's placed at beginning of barcode text, it represents a FNC1 character in "First position" mode, otherwise, it represents a data field separator. See also the "Character set" section above.

  Note, when the "poFirstFNC1MatchAI01" is included in the value of Options property, if the "\f" followed by a SSC-14 number (including Application Identifier 01 and 13-digit data, the checkdigit isn't required) is placed at beginnig of barcode text, the "\905" will be selected to encode the FNC1 character in order to reduce the symbol size (the leading Application Identifier 01 is not encoded in the SSC-14 number).
- **\s[<Index>, <File_ID>, <File_Name>, <Amount>, <Time_Stamp>, <Sender>, <Address>, <File_Size>, <Checksum>]** Insert structured append control block to barcode text in order to create the structured append symbol. See also the "Structured append" section below.
- **\,**: Insert a comma to File name, Sender or Address field in the structured append control block. It shall only be placed in these fields of structured append control. See also the "Structured append" section below.
- **\t**: Structured append terminator. If the segment count is unused in the structured append control block of a structured append symbols set, this terminator is required in last symbol within the set of the structured append symbols. Otherwise, it's optional. See also the "Structured append" section below.
- **\<nnn>**: Insert a function codeword. This "**<nnn>**" can be one of these values:
  - **900**: Manually places a mode latch to Text compaction mode.
  - **901**: Manually places a mode latch to Byte compaction mode. It shall be used when the total number of subsequent bytes to be encoded is not a multiple of 6, otherwise, it will be changed to "\924" automatically by the component.
  - **902**: Manually places a mode latch to Numeric compaction mode.
  - **903**: If it's placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode and indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3). Also it implies a mode latch to Mixed sub-mode of Text compaction mode.

    If it's placed in the barcode text, but not at beginning of the barcode text, it represents an FNC1 character as field separator, and implies a mode latch to Alpha sub-mode of Text compaction mode.
  - **904**: If it's placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode and indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128

emulation, the symbology identifier prefix is set to]C1 or]L3). Also it implies a mode latch to Numeric compaction mode.

If it's placed in the barcode text, but not at beginning of the barcode text, it represents an FNC1 character as field separator, and implies a mode latch to Mixed sub-mode of Text compaction mode.

- **905**: If the "\905" followed by 13 required digits is placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode and encodes a SCC-14 number together with subsequent 13 digits (the Application Identifier 01 is implied and the last checkdigit isn't required). It indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3). Also it implies a mode latch to Numeric compaction mode.

  If the it is placed in the barcode text, but not at beginning of the barcode text, it represents an FNC1 character as field separator, and implies a mode latch to Numeric compaction mode.

- **906**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Mixed sub-mode of Text compaction mode. It not only indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that a linear symbol printed below the symbol is "linked" to the data of the symbol.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **907**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. it not only indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that a linear symbol printed below the symbol is "linked" to the data of the symbol.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **908**: The "\908" followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) represents a leading FNC1 character in "Second position" mode, and indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (used by AIM Global, the symbology identifier prefix is set to]C2 or]L4). Also it implies a mode latch to Mixed sub-mode of Text compaction mode.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **909**: The "\909" followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) represents a leading FNC1 character in "Second position" mode, and indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (used by AIM Global, the symbology identifier prefix is set to]C2 or]L4). Also it implies a mode latch to Numeric compaction mode.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **910**: It indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (standard data package, the symbology identifier prefix is set to]C0 or]L5), and implies a mode latch to Mixed sub-mode of Text compaction mode.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **911**: It indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (standard data package, the symbology identifier prefix is set to]C0 or]L5), and implies a mode latch to Numeric compaction mode.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **912[AAYYMMDDBB]**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. Not only does it indicate that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also it encodes that the data begins with a 6-digit date field (Application Identifier is 11, 13, 15, or 17), and this date field may be followed by an implied Application Identifier 10 or 21 as well:
  - **AA**: Application Identifier of the 6-digit data field. It can be one of 11, 13, 15, or 17.
  - **YY**: Year of the 6-digit data field.
  - **MM**: Month of the 6-digit data field.
  - **DD**: Day of the 6-digit data field.
  - **BB**: An optional Application Identifier that follows the 6-digit data field. It can be one of 10 or 21.

  For example:

  ```
  \912[1199010721]
  ```

  Also, You can encode the 6-digit date field and the subsequent Application Identifier by yourself:

  ```
  \912994071
  ```

  You can explicitly insert a mode latch to Numeric compaction mode after the "\912" by using the "-":

  ```
  \912[-1199010721]
  \912[-]994071
  ```

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **914**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. It not only indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that the data begins with an implied Application Identifier is 10.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **915**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. it not only indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that the data begins with an implied Application Identifier is 21.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **916**: Insert a 05 macro to barcode text. It shall only be placed once at beginning of the barcode text and it shall not be used in conjunction with a series of structured append symbols. It's completely equivalent to escape sequence "\5".

- **917**: Insert a 06 macro to barcode text. It shall only be placed once at beginning of the barcode text and it shall not be used in conjunction with a series of structured append symbols. It's completely equivalent to escape sequence "\6".

- **918**: It shall be used as a linkage flag to signal the presence of an associated linear component in a composite symbol (other than an ECC.UCC composite symbol). The "\918" may be placed anywhere in the barcode text.

- **919**: Reserved.

- **920**: It shall be used as a linkage flag to signal the presence of an associated linear component in an ECC.UCC composite symbol. The "\920" may be placed anywhere in the barcode text.

- **921**: Instructs the reader to interpret the data contained within the MicroPDF417 symbol as programming for reader

initialisation. It only shall be placed once at beginning of the barcode text. In the case of a structured append initialisation sequence, it shall appear in every symbol. It's completely equivalent to escape sequence "\r".

- **924**: Manually places a mode latch to Byte compaction mode. It shall be used when the total number of subsequent bytes to be encoded is an integer multiple of 6, otherwise, it will be changed to "\901" automatically by the component.

For some reader, the meanings of some function codewords don't conform with the MicroPDF417 specification. You can use the Options property to change the meanings of these fucntion codewords, in order to match the reader.

Note, the "{RS}" is ASCII character RS (ASCII value 30), the"{GS}" is ASCII character GS (ASCII value 29), and the"{EOT}" is ASCII character EOT (ASCII value 4).

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set. Five broad types of interpretations are supported in MicroPDF417:

- International character sets (or code pages).
- General purpose interpretations such as encryption and compaction.
- User defined interpretations for closed systems.
- Transmission of control information for structured append.
- Transmission of uninterpreted MicroPDF417 codewords.

The ECI is identified by an integer (up to 6 digits) which is encoded in the MicroPDF417 by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 999999 (including the boundaries), the leading zero is optional.

ECI indicator blocks may be placed anywhere in the barcode text in a single or structured append set of MicroPDF417 symbols. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

Normally, the sub-mode invoked immediately prior to the ECI escape sequence is preserved for the encodation immediately after it. Thus, sub-mode latches and shifts are preserved across an ECI escape sequence; and thus a sub-mode shift immediately before an ECI escape sequence is not ignored. If the value "poIgnoreShiftBeforeECI" is included in the Options property, the sub-mode shift immediately before an ECI escape sequence is ignored, but a sub-mode latch immediately before an ECI escape sequence is never ignored.

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Structured append

Structured append provides a mechanism for the data in a file too large to be split into blocks and be represented in more than one MicroPDF417 symbol. Up to 99999 individual MicroPDF417 symbols may be used to encode data in structured append.

Structured append symbols differ from ordinary MicroPDF417 symbols in that they contain additional control information in a structured append control block. The control block defines the file ID, the concatenation sequence and optionally other information about the file. structured append decoder uses the control block's information to reconstruct the file correctly independently of symbol scanning order. The escape sequence "\s[<Index>, <File_ID>, <File_Name>, <Amount>, <Time_Stamp>, <Sender>, <Address>, <File_Size>, <Checksum>]" is used to place the structured append control block to the barcode text:

- **Index**: The position index of the symbol within the set set of MicroPDF417 symbols in the structured append format. It's an integer between 1 and 999999 (including the boundaries) in decimal string format. Note, the field is required.
- **File_ID**: The file ID. It is a variable length numeric sequence which contains one or more triads of digits. Each triad of digits is an integer between 000 to 899 (the leading zero is required), in decimal string format. For each related structured append

symbol, the file ID should be specified using the same numeric sequence. This ensures that all reassembled symbol data belongs to the same distributed file representation. Note, the field is required.

- **File_Name**: The file name. It's a text string with an implied reset to ECI 000002. The ECI escape sequences may be used in order to set a different ECI within the field. Note, the field is optional.
- **Amount**: The segment count (identifying the total number of structured append symbols in the distributed file). It's an integer between 2 and 99999 (including the boundaries) in decimal string format. Note, the field is optional, if it's used, that field shall appear in every segment.
- **Time_Stamp**: The time stamp. It indicates the time stamp on the source file. It's a GMT time in "**yyyy-mm-dd hh:nn:ss**" format:
  - **yyyy**: 4-digit year.
  - **mm**: 2-digit month, the leading zero is optional.
  - **dd**: 2-digit day, the leading zero is optional.
  - **hh**: 2-digit hour in 24 hours format, the leading zero is optional.
  - **nn**: 2-digit minute, the leading zero is optional.
  - **ss**: 2-digit second, the leading zero is optional.

  Note, entire time portion or the minute and second portion are optional, i.e. the "**yyyy-mm-dd**", "**yyyy-mm-dd hh**", and "**yyyy-mm-dd hh:nn**" formats are valid. If the hour, minute or second portion is not included, they defaults to 00.
- **Sender**: The sender, It's a text string with an implied reset to ECI 000002. The ECI escape sequences may be used in order to set a different ECI within the field. Note, the field is optional.
- **Address**: The address, It's a text string with an implied reset to ECI 000002. The ECI escape sequences may be used in order to set a different ECI within the field. Note, the field is optional.
- **File_Size**: The file size. It's an integer in decimal string format, and indicates the size in bytes of the entire source file. Note, the field is optional.
- **Checksum**: The checksum, It's an integer value of the 16-bit (2 bytes) CRC checksum using the CCITT-16 polynomial computed over the entire source file, in decimal string format. You can use the GetCheckSum method to calculate the checksum for a source file. Note, the field is optional.

The ECI escape sequences, "\900", "\901", "\902", and "\924" can be used in the File_Name, Sender and Address fields. If you want place the comma in these fields, please use the escape sequence "\," instead.

An empty string indicates an unused optional field, the subsequent comma can be omitted if all fields of succeeding are unused. otherwise, the comma is required. All unused fields at the end can be omitted.

The control block may only be placed once in the barcode text. Also, it shall be placed at end of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the control block be placed more than once or it isn't placed at end of the barcode text.

The following is an example of structured append symbol:

- Position index: 2
- File ID: 001 287 023
- File name: Unused
- Segment count: 5
- Time stame: 05:30:00 GMT on 3 December, 2008
- Sender: Unused
- Address: USA
- File size: Unused

- Checksum: Unused

```
ABCDEFGHIJKLMN\s[2,001287023,,5,2008-12-03 05:30:00,,USA]
```

Note, if the segment count is unused, a structured append terminator, "\t" escape sequence is required in last symbol within the set of structured append symbols. Otherwise, it's optional. The "\t" escape sequence shall be placed at the end of the barcode text, after the structured append control block. For example:

```
ABCDEFGHIJKLMN\s[5,001287023]\t
```

The AllowEscape property should be set to true in order to place the structured append control block and the terminator.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- MinSize
- MaxSize
- StretchOrder
- RowHeight
- DefauldEncodeMode
- DefauldTextEncodeMode
- UseECIDescriptor
- Options
- AllowEscape
- CurrentSize (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize
- GetCheckSum

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.14 TBarcode2D_MicroQRCode

The component is used to create the Micro QR Code barcode symbols. It's defined in the pMicroQRCode unit.

Micro QR Code is a very smallQR Code that fits applications that require a smaller space and use smaller amounts of data, such as ID of printed circuit boards and electronics parts, etc. The efficiency of data encoding has been increased with the use of only one position detection pattern.

Micro QR Code is a two-dimensional matrix symbology. It is not capable of storing much data. However, because it can store data for each symbol size more efficiently than QR Code, the size of Micro QR Code symbols does not significantly increase, even though the amount of data is increased.

The Micro QR Code was created by Toyota subsidiary Denso-Wave in 1994.

Error checking and correcting (ECC)

There are three user-selectable levels of error correction, as shown in following table, offering the capability of recovery from the amounts of damage in the table:

Values of ECCLevel property	ECC levels	Recevory capacities (%) (approx.)
elLowest	L	7
elMedium	M	15
elQuality	Q	25

You can use the ECCLevel property to specify the error correction code level for a Micro QR Code symbol. It can be one of these values: elLowest, elMedium, and elQuality, corresponding to the error checking and correcting levels L, M, and Q. These values are defined in the pMicroQRCode unit.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by the ECCLevel property (see also the "Symbol sizes" section below). In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The property CurrentECCLevel can be used to get the factual error correction code level.

Note, for version 1 Micro QR Code symbols, the error checking and correcting capacity is limited to error detection only. See also the "Symbol sizes" section below.

Symbol sizes

There are four sizes of Micro QR Code symbol, referred to as version 1 to 4, in increasing order of data capacity. You can use the MinVersion and the MaxVersion properties to specifiy the minimum and maximum sizes for a Micro QR Code symbol. They can be one of values from 1 to 4 (defined in the pMicroQRCode unit), corresponding to the versions 1 to 4. The smallest symbol size that accommodates the barcode text will be automatically selected between minimum and maximum symbol sizes. The CurrentVersion property can be used to get the factual symbol size.

The symbol sizes and the maximum data capacity of each version are listed in following table:

Version	Symbol size (modules)	ECC level	Data capacities			
			Numeric mode	Alphanumeric mode	Byte mode	Kanji mode
1	11 * 11	L	5	-	-	-
2	13 * 13	L	10	6	-	-
		M	8	5	-	-
3	15 * 15	L	23	14	9	6
		M	18	11	7	4
4	17 * 17	L	35	21	15	9
		M	30	18	13	8
		Q	21	13	9	5

If the barcode text does not fill the maximum data capacity of the Micro QR Code symbol, remaining data capacity of the symbol will be filled by using PAD characters. If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxVersion property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Quiet zones

The minimum quiet zone is equal to 2 modules on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 2.

Character set

All 8-bit values can be encoded. The default interpretation shall be:

- For values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646 (ASCII).
- For values 128 - 255, in accordance with ISO/IEC 8859-1 (Extended ASCII).

Encoding modes

Micro QR Code has four encoding modes as Numeric mode, Alphanumeric mode, Kanji mode and Byte mode respectively in decreasing order of encoding density. All 256 8-bits value are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The character sets in each mode are listed in the following:

- **Numeric mode**: Encodes data from the decimal digit set (0 to 9, byte values 48 to 57).
- **Alphanumeric mode**: Encodes data from a set of 45 characters, i.e. 10 numeric digits (0 to 9) (byte values 48 to 57), 26 alphabetic characters (A - Z) (byte values 65 to 90) , and 9 symbols (SP, $, %, *, +, -, ., /, :) (byte values 32, 36, 37, 42, 43, 45 to 47, 58 respectively).

  Note, alphanumeric mode is not available in version 1 Micro QR Code symbols.
- **Byte mode**: All 8-bit values can be encoded.

  Note, Byte mode is not available in version 1 or 2 Micro QR Code symbols.
- **Kanji mode**: The Kanji mode efficiently encodes Kanji characters in accordance with the Shift JIS system based on JIS X 0208. The Shift JIS values are shifted from the JIS X 0208 values. JIS X 0208 gives details of the shift coded representation.

  It may be possible to achieve a smaller symbol size by using the Kanji mode compaction rules when an appropriate sequence of byte values occurs in the data. You can set the AllowKanjiMode property to false in order to disable the Kanji mode.

  Note, Kanji mode is not available in version 1 or 2 Micro QR Code symbols.

The EncodePolicy property indicates how to use these encoding mode by the component. This property can be one of these values (defined in the pMicroQRCode unit):

- **epMixingOptimal**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The optimal combination of encoding modes will be used in order to minimize the symbol size.

- **epMixingQuick**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The combination of encoding modes may not be the optimal one, in order to optimize encoding speed.
- **epSingleMode**: The encoding mode will be selected automatically and applied to entire symbol, based on the barcode text, in other words, the barcode text to be encoded will be analysed, and an appropriates lowest level (highest encoding density) encoding mode will be selected, in order to minimize the symbol size, and the encoding mode will not be switched in the sybmol. Note, the Kanji mode will not be used if the AllowKanjiMode property is set to false.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Inversed
- Mirrored
- EncodePolicy
- MaxSliceLen
- MinVersion
- MaxVersion
- ECCLevel
- ECCLevelUpgrade
- AllowKanjiMode
- CurrentVersion (Read only)
- CurrentECCLevel (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.15 TBarcode2D_PDF417

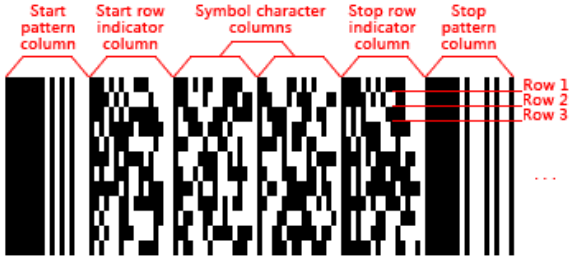
The component is used to create the PDF417 Barcode symbols. It is defined in the pPDF417 unit.

PDF417 is a continuous, stacked 2D linear barcode symbol format used in a variety of applications, primarily transport, identification cards, and inventory management. PDF stands for Portable Data File. It can encode all 8-bit values.

The PDF417 symbology was invented by Dr. Ynjiun P. Wang at Symbol Technologies in 1991.

Symbol size

Each PDF417 symbol consists of a stack of vertically aligned rows with a minimum of 3 rows (maximum 90 rows). Each stacked row shall include a minimum of 1 symbol character column (maximum 30 symbol character columns), excluding start, stop and row indicator columns. They are defined in the pPDF417Custom unit. The RowHeight property can be used to specify the height for each stacked row, in modules. See diagram:



You can use the MinRows and the MaxRows properties to specifiy the minimum and maximum number of stacked rows for a PDF417 symbol. And use the MinColumns and the MaxColumns properties to specifiy the minimum and maximum number of symbol character columns for it. In other words, the MinRows and MinColumns properties specify a minimum symbol size, and the MaxRows and MaxColumns properties specify a maximum symbol size. According to the priority order specified by the StretchOrder property, the first symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

You can use the CurrentRows property to get the factual number of stacked rows. And use the CurrentColumns property to get the factual number of columns.

If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxRows and the MaxColumns properties, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Quiet zones

The symbol shall include a quiet zone on all four sides. The minimum quiet zone is equal to 4 modules on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 4.

Error checking and correcting (ECC)

PDF417 symbols offer 9 levels of error correction, referred to as ECC 0 to ECC 8 respectively in increasing order of recovery capacity. You can use the ECCLevel property to specify the error correction code level for a PDF417 symbol. It can be one of values from eIEcc_0 to eIEcc_8, corresponding to error correction code level from ECC 0 to ECC 8. They are defined in the pPDF417Custom unit.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined based on the length of barcode text, and the error correction code level specified by the ECCLevel property, in other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The CurrentECCLevel property can be used to get the factual error correction code level.

Character set

- All 8-bit values can be encoded. The default interpretation shall be:
  - For values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646 (ASCII).
  - For values 128 - 255, in accordance with ISO/IEC 8859-1 (Extended ASCII).

  This interpretation corresponds to ECI 000003.

  PDF417 has three unique data compaction modes as Text compaction mode, Numeric compaction mode, and Byte compaction mode. All 8-bit values are encoded by switching internally between all 3 compaction modes:

  - **Text compaction mode**: Permits all printable ASCII characters, i.e. ASCII values from 32 to 126 inclusive accordance with ISO/IEC 646, as well as selected control characters (CR, HT, and LF).

    The mode has four sub-modes:

    - **Alpha**: Uppercase letters and space.
    - **Lower**: Lowercase letters and space.
    - **Mixed**: Numeric and some punctuations.
    - **Punctuation**: Some punctuations.

    The sub-mode will be switched automatically in order to minimize the symbol size.

  - **Numeric compaction mode**: Permits efficient encoding of numeric data string.
  - **Byte compaction mode**: Permits all 256 possible 8-bit byte values to be encoded. This includes all ASCII characters value from 0 to 127 inclusive and provides for international character set support.

  The compaction mode will be switched automatically in a PDF417 symbol in order to minimize the symbol size. Also, you can manually switch the code set by using escape sequences "\900", "\901", "\902", and "\924". See also the "Escape sequences" section below.

- The FNC1 characters can be encoded for compatibility with some existing applications. There are two mode FNC1 characters to identify symbols encoding messages formatted according to specific predefined industry or application specifications, also the FNC1 character can be used as the data field separator:
  - **First position mode**: FNC1 in this mode indicator identifies symbols encoding data formatted according to the GS1 Application Identifiers standard. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode text.

    If any one of escape sequences in "\903", "\904" and "\905" followed by an Application Identifier (2 or more digits) is placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode, and implies a mode latch to Text compaction mode or Numeric compaction mode. See also the "Escape sequences" section below.

    Also, you can place an escape sequence "\f" followed by an Application Identifier (2 or more digits) at beginning of barcode text instead of the "\903", "\904" or "\905" escape sequence followed by the Application Identifier, to place the FNC1 character in "First position" mode. The component will automatically select one of escape sequences from "\903", "\904" and "\905" depending on the barcode text, in order to minimize the symbol size.

    The escape sequences "\906", "\907", "\912", "\914" and "\915" can be placed the FNC1 character in "First position" mode too. See also the "Escape sequences" section below.

  - **Second position mode**: FNC1 in this mode indicator identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. It is immediately followed by an application indicator assigned to identify the specification concerned by AIM International. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode. An application indicator may take the form of any single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number.

    If any one of escape sequences in "\908" and "\909" followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) is placed at beginning of barcode text, it

represents a leading FNC1 character in "Second position" mode, and implies a mode latch to Text compaction mode or Numeric compaction mode. See also the "Escape sequences" section below.

- **Data field separator**: The FNC1 character may also be used as a data field separator(i.e. at the end of a variable-length data field), in which case it will be represented in the transmitted message as GS character (ASCII value 29). The data field separator shall not be placed at beginning of barcode text.

  If any one of escape sequences in "\903", "\904" and "\905" is placed in the barcode text, but not at beginning of the barcode text, it represents a data field separator, and implies a mode latch to Text compaction mode or Numeric compaction mode. See also the "Escape sequences" section below.

  Also, you can use the escape sequence "\f" instead of the "\903", "\904" or "\905" to placed the FNC1 character as the data field separator. The component will automatically select one of escape sequences from "\903", "\904" and "\905" depending on the barcode text, in order to minimize the symbol size.

  The appearance of an adjacent pair of repeated "\903", "\904" or "\905" escape sequences will cause the decoder to divide the transmission at that point (but without transmitting the two implied GS (ASCII value 29) characters themselves).

- The ECI indicator blocks can be encoded for the standardized encoding of message interpretation information. The escape sequence "\e[<ECI_Numnerm>]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\5**: Insert a 05 macro to barcode text. It's used to abbreviate the industry specific header "[)>{RS}05{GS}" and trailer "{RS}{EOT}", in order to reduce the size of symbol. It must only be placed once at beginning of the barcode text and it shall not be used in Macro PDF417 symbols.
- **\6**: Insert a 06 macro to barcode text. It's used to abbreviate the industry specific header "[)>{RS}06{GS}" and trailer "{RS}{EOT}", in order to reduce the size of symbol. It shall only be placed once at beginning of the barcode text and it shall not be used in Macro PDF417 symbols.
- **\r**: Instructs the reader to interpret the data contained within the PDF417 symbol as programming for reader initialisation. It shall only be placed once at beginning of the barcode text. In the case of a Macro PDF417 initialisation sequence, it shall appear in every symbol.
- **\e[<ECI_Numnerm>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.
- **\f**: Insert a FNC1 character either in "First position" mode or as a data field separator, to the barcode text. One of "\903", "\904", and "\905" will be automatically selected instead of the "\f", to insert to the symbol, depending on the barcode text. If it's placed at beginning of barcode text, it represents a FNC1 character in "First position" mode, otherwise, it represents a data field separator. See also the "Character set" section above.

  Note, When the "poFirstFNC1MatchAI01" is included in the value of Options property, if the "\f" followed by a SSC-14 number (including Application Identifier 01 and 13-digit data, the checkdigit isn't required) is placed at beginnig of barcode, the "\905" will be selected to encode the FNC1 character in order to reduce the symbol size (the leading Application Identifier 01 is not encoded in the SSC-14 number).

- **\s[<Index>, <File_ID>, <File_Name>, <Amount>, <Time_Stamp>, <Sender>, <Address>, <File_Size>, <Checksum>]** Insert additional control information to barcode text in order to create the Marco PDF417 symbol. See also the "Macro PDF417" section below.
- **\,**: Insert a comma to File name, Sender or Address field of the Marco PDF417 control information. It shall only be placed in

these fields of Marco PDF417 control information. See also the "Macro PDF417" section below.

- **\t**: Marco PDF417 terminator. If the segment count is unused in the control information of a Marco PDF417 symbols set, this terminator is required in last symbol within the set of the Marco PDF417 symbols. Otherwise, it's optional. See also the "Macro PDF417" section below.
- **\<nnn>**: Insert a function codeword. This "**<nnn>**" can be one of these values:
  - **900**: Manually places a mode latch to Text compaction mode.
  - **901**: Manually places a mode latch to Byte compaction mode. It shall be used when the total number of subsequent bytes to be encoded is not a multiple of 6, otherwise, it will be change to "\924" automatically by the component.
  - **902**: Manually places a mode latch to Numeric compaction mode.
  - **903**: If it's placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode and indicates that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3). Also it implies a mode latch to Mixed sub-mode of Text compaction mode.

    If it's placed in the barcode text, but not at beginning of the barcode text, it represents an FNC1 character as field separator, and implies a mode latch to Alpha sub-mode of Text compaction mode.
  - **904**: If it's placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode and indicates that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3). Also it implies a mode latch to Numeric compaction mode.

    If it's placed in the barcode text, but not at beginning of the barcode text, it represents an FNC1 character as field separator, and implies a mode latch to Mixed sub-mode of Text compaction mode.
  - **905**: If the "\905" followed by 13 required digits is placed at beginning of barcode text, it represents a leading FNC1 character in "First position" mode and encodes a SCC-14 number together with subsequent 13 digits (the Application Identifier 01 is implied and the last checkdigit isn't required). It indicates that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3). Also it implies a mode latch to Numeric compaction mode.

    If the it is placed in the barcode text, but not at beginning of the barcode text, it represents an FNC1 character as field separator, and implies a mode latch to Numeric compaction mode.
  - **906**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Mixed sub-mode of Text compaction mode. It not only indicates that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that a linear symbol printed below the symbol is "linked" to the data of the symbol.

    The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.
  - **907**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. It not only indicates that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that a linear symbol printed below the symbol is "linked" to the data of the symbol.

    The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.
  - **908**: The "\908" followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) represents a leading FNC1 character in "Second position" mode, and indicates that the PDF417 symbol's data output shall conform with the Code128 specification (used by AIM Global, the symbology identifier prefix is set to]C2 or]L4). Also it implies a mode latch to Mixed sub-mode of Text compaction mode.

The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **909**: The "\909" followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) represents a leading FNC1 character in "Second position" mode, and indicates that the PDF417 symbol's data output shall conform with the Code128 specification (used by AIM Global, the symbology identifier prefix is set to]C2 or]L4). Also it implies a mode latch to Numeric compaction mode.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **910**: It indicates that the PDF417 symbol's data output shall conform with the Code128 specification (standard data package, the symbology identifier prefix is set to]C0 or]L5), and implies a mode latch to Mixed sub-mode of Text compaction mode.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **911**: It indicates that the PDF417 symbol's data output shall conform with the Code128 specification (standard data package, the symbology identifier prefix is set to]C0 or]L5), and implies a mode latch to Numeric compaction mode.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **912[AAYYMMDDBB]**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. Not only does it indicate that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also it encodes that the data begins with a 6-digit date field (Application Identifier is 11, 13, 15, or 17), and this date field may be followed by an implied Application Identifier 10 or 21 as well:

  - **AA**: Application Identifier of the 6-digit data field. It can be one of 11, 13, 15, or 17.
  - **YY**: Year of the 6-digit data field.
  - **MM**: Month of the 6-digit data field.
  - **DD**: Day of the 6-digit data field.
  - **BB**: An optional Application Identifier that follows the 6-digit data field. It can be one of 10 or 21.

  For example:

  ```
  \912[1199010721]
  ```

  Also, You can encode the 6-digit date field and the subsequent Application Identifier by yourself:

  ```
  \912994071
  ```

  You can explicitly insert a mode latch to Numeric compaction mode after the "\912" by using the "-":

  ```
  \912[-1199010721]
  \912[-]994071
  ```

  The escape sequence shall only be placed once at beginning of barcode text, otherwise the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **914**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. It not only indicates that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that the data begins with an implied Application Identifier is 10.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **915**: It represents a leading FNC1 character in "First position" mode and implies a mode latch to Numeric compaction mode. it not only indicates that the PDF417 symbol's data output shall conform with the Code128 specification (UCC/EAN-128 emulation, the symbology identifier prefix is set to]C1 or]L3), but also indicates that the data begins with an implied Application Identifier is 21.

  The escape sequence shall only be placed once at beginning of barcode text, otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

- **916**: Insert a 05 macro to barcode text. It shall only be placed once at beginning of the barcode text and it shall not be used in Macro PDF417 symbols. It's completely equivalent to escape sequence "\5".

- **917**: Insert a 06 macro to barcode text. It shall only be placed once at beginning of the barcode text and it shall not be used in Macro PDF417 symbols. It's completely equivalent to escape sequence "\6".

- **918**: It shall be used as a linkage flag to signal the presence of an associated linear component in a composite symbol (other than an ECC.UCC composite symbol). The "\918" may be placed anywhere in the barcode text.

- **919**: Reserved.

- **920**: It shall be used as a linkage flag to signal the presence of an associated linear component in an ECC.UCC composite symbol. The "\920" may be placed anywhere in the barcode text.

- **921**: Instructs the reader to interpret the data contained within the PDF417 symbol as programming for reader initialisation. It only shall be placed once at beginning of the barcode text. In the case of a Macro PDF417 initialisation sequence, it shall appear in every symbol. It's completely equivalent to escape sequence "\r".

- **924**: Manually places a mode latch to Byte compaction mode. It shall be used when the total number of subsequent bytes to be encoded is an integer multiple of 6, otherwise, it will be changed to "\901" automatically by the component.

For some reader, the meanings of some function codewords don't conform with the PDF417 specification. You can use the Options property to change the meanings of these fucntion codewords, in order to match the reader.

Note, the "{RS}" is ASCII character RS (ASCII value 30), the "{GS}" is ASCII character GS (ASCII value 29), and the "{EOT}" is ASCII character EOT (ASCII value 4).

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set. Five broad types of interpretations are supported in PDF417:

- International character sets (or code pages).
- General purpose interpretations such as encryption and compaction.
- User defined interpretations for closed systems.
- Transmission of control information for Macro PDF417.
- Transmission of uninterpreted PDF417 codewords.

The ECI is identified by an integer (up to 6 digits) which is encoded in the PDF417 by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 999999 (including the boundaries), the leading zero is optional.

ECI indicator blocks may be placed anywhere in the barcode text in a single or Macro PDF417 set of symbols. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

Normally, the sub-mode invoked immediately prior to the ECI escape sequence is preserved for the encodation immediately after it. Thus, sub-mode latches and shifts are preserved across an ECI escape sequence; and thus a sub-mode shift immediately before an ECI escape sequence is not ignored. If the value "poIgnoreShiftBeforeECI" is included in the Options property, the sub-mode shift immediately before an ECI escape sequence is ignored, but a sub-mode latch immediately before an ECI escape sequence is never ignored.

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Compact PDF417

In an environment where space considerations are a primary concern and symbol damage is unlikely (e.g. an office), the right row indicators may be omitted and the stop pattern may be reduced to one module width bar. This reduction version is called Compact PDF417, which is fully decoder compatible with standard PDF417.

Note, the Compact PDF417 was referred as Truncated PDF417 in previous standard.

You can set the Compact property to true in order to generate a Compact PDF417 symbol.

Macro PDF417

Macro PDF417 provides a mechanism for the data in a file too large to be split into blocks and be represented in more than one PDF417 symbol. This mechanism is similar to the structured append feature in other symbologies. Up to 99999 individual PDF417 symbols may be used to encode data in Macro PDF417.

Each Macro PDF417 symbol shall contain additional control information to enable the original data file to be properly reconstructed, irrespective of the sequence in which the individual PDF417 symbols are scanned and decoded. The escape sequence "\s[<Index>, <File_ID>, <File_Name>, <Amount>, <Time_Stamp>, <Sender>, <Address>, <File_Size>, <Checksum>]"is used to place the control information to the barcode text:

- **Index**: The position index of the symbol within the set of Marco PDF417 symbols. It's an integer between 1 and 999999 (including the boundaries) in decimal string format. Note, the field is required.
- **File_ID**: The file ID. It is a variable length numeric sequence which contains one or more triads of digits. Each triad of digits is an integer between 000 to 899 (the leading zero is required), in decimal string format. For each related Macro PDF417 symbol, the file ID should be specified using the same numeric sequence. This ensures that all reassembled symbol data belongs to the same distributed file representation. Note, the field is required.
- **File_Name**: The file name. It's a text string with an implied reset to ECI 000002. The ECI escape sequences may be used in order to set a different ECI within the field. Note, the field is optional.
- **Amount**: The segment count (identifying the total number of Macro PDF417 symbols in the distributed file). It's an integer between 2 and 99999 (including the boundaries) in decimal string format. Note, the field is optional, if it's used, that field shall appear in every segment.
- **Time_Stamp**: The time stamp. It indicates the time stamp on the source file. It's a GMT time in "**yyyy-mm-dd hh:nn:ss**" format:
  - **yyyy**: 4-digit year.
  - **mm**: 2-digit month, the leading zero is optional.
  - **dd**: 2-digit day, the leading zero is optional.
  - **hh**: 2-digit hour in 24 hours format, the leading zero is optional.
  - **nn**: 2-digit minute, the leading zero is optional.
  - **ss**: 2-digit second, the leading zero is optional.

  Note, entire time portion or the minute and second portion are optional, i.e. the "**yyyy-mm-dd**", "**yyyy-mm-dd hh**", and "**yyyy-mm-dd hh:nn**" formats are valid. If the hour, minute or second portion is not included, they defaults to 00.
- **Sender**: The sender, It's a text string with an implied reset to ECI 000002. The ECI escape sequences may be used in order to set a different ECI within the field. Note, the field is optional.
- **Address**: The address, It's a text string with an implied reset to ECI 000002. The ECI escape sequences may be used in

order to set a different ECI within the field. Note, the field is optional.

- **File_Size**: The file size. It's an integer in decimal string format, and indicates the size in bytes of the entire source file. Note, the field is optional.
- **Checksum**: The checksum, It's an integer value of the 16-bit (2 bytes) CRC checksum using the CCITT-16 polynomial computed over the entire source file, in decimal string format. You can use the GetCheckSum method to calculate the checksum for a source file. Note, the field is optional.

The ECI escape sequences, "\900", "\901", "\902", and "\924" can be used in the File_Name, Sender and Address fields. If you want place the comma in these fields, please use the "\," instead.

An empty string indicates an unused optional field, the subsequent comma can be omitted if all fields of succeeding are unused. otherwise, the comma is required. All unused fields at the end can be omitted.

The control information may only be placed once in the barcode text. Also, it shall be placed at end of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the control information be placed more than once or it isn't placed at end of the barcode text.

The following is an example of Marco PDF417 symbol:

- Position index: 2
- File ID: 001 287 023
- File name: Unused
- Segment count: 5
- Time stame: 05:30:00 GMT on 3 December, 2008
- Sender: Unused
- Address: USA
- File size: Unused
- Checksum: Unused

```
ABCDEFGHIJKLMN\s[2,001287023,,5,2008-12-03 05:30:00,,USA]
```

Note, if the segment count is unused, a Marco PDF417 terminator, "\t" escape sequence is required in last symbol within the set of Marco PDF417 symbols. Otherwise, it's optional. The "\t" escape sequence shall be placed at the end of the barcode text, after the control information. For example:

```
ABCDEFGHIJKLMN\s[5,001287023]\t
```

The AllowEscape property should be set to true in order to place the control information and the terminator.

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- MinRows
- MaxRows
- MinColumn
- MaxColumn
- StretchOrder
- ECCLevel
- ECCLevelUpgrade
- RowHeight
- Compact
- Options
- AllowEscape
- CurrentRows (Read only)
- CurrentColumns (Read only)
- CurrentECCLevel (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize
- GetCheckSum

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.16 TBarcode2D_QRCode

The component is used to create the QR Code barcode symbols. It's defined in the pQRCode unit.

A QR Code is a two-dimensional matrix symbology, readable by QR scanners, mobile phones with a camera, and smartphones. The code consists of black modules arranged in a square pattern on white background. It has position detection patterns on three of its four corners. The information encoded can be text, URL or other data.

Common in Japan, where it was created by Toyota subsidiary Denso-Wave in 1994, the QR Code is one of the most popular types of two-dimensional barcodes. Although initially used for tracking parts in vehicle manufacturing, QR Codes are now used in a much broader context, including both commercial tracking applications and convenience-oriented applications aimed at mobile phone users.

QR is the abbreviation for Quick Response, as the creator intended the code to allow its contents to be decoded at high speed.

Error checking and correcting (ECC)

There are four user-selectable levels of error correction, as shown in following table, offering the capability of recovery from the amounts of damage in the table:

Values of ECCLevel property	ECC levels	Recevory capacities (%) (approx.)
elLowest	L	7
elMedium	M	15
elQuality	Q	25
elHighest	H	30

You can use the ECCLevel property to specify the error correction code level for a QR Code symbol. It can be one of these values: elLowest, elMedium, elQuality, and elHighest, corresponding to the error checking and correcting levels L, M, Q, and H. These values are defined in the pRCode unit.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by the ECCLevel property (see also the "Symbol sizes" section below). In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level. The property CurrentECCLevel can be used to get the factual error correction code level.

Symbol sizes

There are forty sizes of QR Code symbol. referred to as version 1 to 40, in increasing order of data capacity. You can use the MinVersion and the MaxVersion properties to specifiy the minimum and maximum sizes for a QR Code symbol.They can be one of values from 1 to 40 (defined in the pQRCode unit), corresponding to the versions 1 to 40. The smallest symbol size that accommodates the barcode text will be automatically selected between minimum and maximum symbol sizes. The CurrentVersion property can be used to get the factual symbol size.

These symbol sizes and the maximum data capacity of each version are listed in following table:

Version	Symbol size (modules)	ECC level	Data capacities			
			Numeric mode	Alphanumeric mode	Byte mode	Kanji mode
1	21 * 21	L	41	25	17	10
		M	34	20	14	8
		Q	27	16	11	7
		H	17	10	7	4
2	25 * 25	L	77	47	32	20
		M	63	38	26	16
		Q	48	29	20	12
		H	34	20	14	8
3	29 * 29	L	127	77	53	32
		M	101	61	42	26
		Q	77	47	32	20

		H	58	35	24	15
4	33 * 33	L	187	114	78	48
		M	149	90	62	38
		Q	111	67	46	28
		H	82	50	34	21
5	37 * 37	L	255	154	106	65
		M	202	122	84	52
		Q	144	87	60	37
		H	106	64	44	27
6	41 * 41	L	322	195	134	82
		M	255	154	106	65
		Q	178	108	74	45
		H	139	84	58	36
7	45 * 45	L	370	224	154	95
		M	293	178	122	75
		Q	207	125	86	53
		H	154	93	64	39
8	49 * 49	L	461	279	192	118
		M	365	221	152	93
		Q	259	157	108	66
		H	202	122	84	52
9	53 * 53	L	522	335	230	141
		M	432	262	180	111
		Q	312	189	130	80
		H	235	143	98	60
10	57 * 57	L	652	395	271	167
		M	513	311	213	131
		Q	364	221	151	93
		H	288	174	119	74
11	61 * 61	L	772	468	321	198
		M	604	366	251	155
		Q	427	259	177	109
		H	331	200	137	85
12	65 * 65	L	883	535	367	226
		M	691	419	287	177
		Q	489	296	203	125
		H	374	227	155	96
13	69 * 69	L	1022	619	425	262
		M	796	483	331	204
		Q	580	352	241	149
		H	427	259	177	109
14	73 * 73	L	1101	667	458	282
		M	871	528	362	223
		Q	621	376	258	159
		H	468	283	194	120
15	77 * 77	L	1250	758	520	320
		M	991	600	412	254
		Q	703	426	292	180
		H	530	321	220	136
		L	1408	854	586	361

16	81 * 81	M	1082	656	450	277
		Q	775	470	322	198
		H	602	365	250	154
17	85 * 85	L	1548	938	644	397
		M	1212	734	504	310
		Q	876	531	364	224
		H	674	408	280	173
18	89 * 89	L	1725	1046	718	442
		M	1346	816	560	345
		Q	948	574	394	243
		H	746	452	310	191
19	93 * 93	L	1903	1153	792	488
		M	1500	909	624	384
		Q	1063	644	442	272
		H	813	493	338	208
20	97 * 97	L	2061	1249	858	528
		M	1600	970	666	410
		Q	1159	702	482	297
		H	919	557	382	235
21	101 * 101	L	2232	1352	929	572
		M	1708	1035	711	438
		Q	1224	742	509	314
		H	969	587	403	248
22	105 * 105	L	2409	1460	1003	618
		M	1872	1134	779	480
		Q	1358	823	565	348
		H	1056	640	439	270
23	109 * 109	L	2620	1588	1091	672
		M	2059	1248	857	528
		Q	1468	890	611	376
		H	1108	672	461	284
24	113 * 113	L	2812	1704	1171	721
		M	2188	1326	911	561
		Q	1588	963	661	407
		H	1228	744	511	315
25	117 * 117	L	3057	1853	1273	784
		M	2395	1451	997	614
		Q	1718	1041	715	440
		H	1286	779	535	330
26	121 * 121	L	3283	1990	1367	842
		M	2544	1542	1059	652
		Q	1804	1094	751	462
		H	1425	864	593	365
27	125 * 125	L	3517	2132	1465	902
		M	2701	1637	1125	692
		Q	1933	1172	805	496
		H	1501	910	625	385
28	129 * 129	L	3669	2223	1528	940
		M	2857	1732	1190	732

		Q	2085	1263	868	534
		H	1581	958	658	405
29	133 * 133	L	3909	2369	1628	1002
		M	3035	1839	1264	778
		Q	2181	1322	908	559
		H	1677	1016	698	430
30	137 * 137	L	4158	2520	1732	1066
		M	3289	1994	1370	843
		Q	2358	1429	982	604
		H	1782	1080	742	457
31	141 * 141	L	4417	2677	1840	1132
		M	3486	2113	1452	894
		Q	2473	1499	1030	634
		H	1897	1150	790	486
32	145 * 145	L	4686	2840	1952	1201
		M	3693	2238	1538	947
		Q	2670	1618	1112	684
		H	2022	1226	842	518
33	149 * 149	L	4965	3009	2068	1273
		M	3909	2369	1628	1002
		Q	2805	1700	1168	719
		H	2157	1307	898	553
34	153 * 153	L	5253	3183	2188	1347
		M	4134	2506	1722	1060
		Q	2949	1787	1228	756
		H	2301	1394	958	590
35	157 * 157	L	5529	3351	2303	1417
		M	4343	2632	1809	1113
		Q	3081	1867	1283	790
		H	2361	1431	983	605
36	161 * 161	L	5836	3537	2431	1496
		M	4588	2780	1911	1176
		Q	3244	1966	1351	832
		H	2524	1530	1051	647
37	165 * 165	L	6153	3729	2563	1577
		M	4775	2894	1989	1224
		Q	3417	2071	1423	876
		H	2625	1591	1093	673
38	169 * 169	L	6479	3927	2699	1661
		M	5039	3054	2099	1292
		Q	3599	2181	1499	923
		H	2735	1658	1139	701
39	173 * 173	L	6743	4087	2809	1729
		M	5313	3220	2213	1362
		Q	3791	2298	1579	972
		H	2927	1774	1219	750
40	177 * 177	L	7089	4296	2953	1817
		M	5596	3391	2331	1435
		Q	3993	2420	1663	1024
		H	3057	1852	1273	784

If the barcode text does not fill the maximum data capacity of the QR Code symbol, remaining data capacity of the symbol will be filled by using PAD characters. If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by the MaxVersion property, an OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

Quiet zones

The minimum quiet zone is equal to 4 modules on all four sides. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 4.

Character set

- All 8-bit values can be encoded. The default interpretation shall be:
  - For values 0 to 127, in accordance with the U.S. national version of ISO/IEC 646 (ASCII).
  - For values 128 - 255, in accordance with ISO/IEC 8859-1 (Extended ASCII).

  This interpretation corresponds to ECI 000003.
- The FNC1 characters can be encoded for compatibility with some existing applications. In QR Code symbology, there are two mode FNC1 characters to identify symbols encoding messages formatted according to specific predefined industry or application specifications:
  - **First position**: FNC1 in this mode indicator identifies symbols encoding data formatted according to the GS1 Application Identifiers standard. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode text, and after any ECI indicator block or structured append block (if exists).

    The escape sequence "\t" or "\f" followed by an Application Identifier (2 or more digits), can be used to placed the FNC1 character in "First position" mode to barcode text. For example, the barcode symbol (01)00012345678905 can be represented using any one of following sequences:

    ```
    \t0100012345678905
    \f0100012345678905
    ```

    Where the GS1 specifications call for the FNC1 character to be used as a data field separator (i.e. at the end of a variable-length data field), QR Code symbols shall use the GS character (byte value 29) to perform this function. In this case, any one of "\g", "\d", "\t", and "\f" escape sequences can be used to place the GS character to the barcode text. For example:

    ```
    \t0100012345678905215 12\f2018
    ```

  - **Second position**: FNC1 in this mode indicator identifies symbols formatted in accordance with specific industry or application specifications previously agreed with AIM International. It is immediately followed by an application indicator assigned to identify the specification concerned by AIM International. For this purpose, it shall only be used once in a symbol and shall be placed at beginning of the barcode, and after any ECI indicator block or structured append block (if exists). An application indicator may take the form of any single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number.

    There are two ways to place a FNC1 character in "Second position" mode:

    - The escape sequence "\d" followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) can be used to placed the FNC1 character in "Second position" mode to barcode text. For example:

      ```
      \dA0001234567890
      \d0100001234567890
      ```

    - The escape sequence "\f[<AI>]" can be used to placed the FNC1 character in "Second position" mode to barcode text:
      - **AI**: The application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a

two-digit number).

For example:

```
\f[A]0001234567890
\f[01]0001234567890
```

Where the application specifications call for the FNC1 character to be used as a data field separator, QR Code symbols shall use the GS character (byte value 29) to perform this function. In this case, any one of "\g", "\d", "\t", and "\f" escape sequences can be used to place the GS character to the barcode text. For example:

```
\dA00012345678905215l2\g2018
```

Note, the "First position" and "Second position" are not used in a literal sense but is a historical reference to the position of the FNC1 symbol character in Code 128 symbols.

- The ECI indicator blocks can be encoded for the standardized encoding of message interpretation information. The escape sequence "\e[<ECI_Number>]" can be used to place the ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

The AllowEscape property should be set to true in order to place the FNC1 characters or the ECI indicator blocks.

Escape sequences

If the AllowEscape property is set to true, following escape sequences are supported by the component, you can insert them to the barcode text:

- ****: Insert a backslash to barcode text.
- **\f**: Insert a FNC1 character to barcode text. There are three usages:
  - Insert a FNC1 character in "First position" mode to the barcode text. In this case, it shall only be used once in a symbol and shall be placed at beginning of the barcode text, and after any ECI indicator block or structured append block (if exists), followed by an Application Identifier (2 or more digits). For example:

    ```
    \f0100012345678905
    ```

  - Insert a FNC1 character in "Second position" mode to the barcode text. In this case, it shall only be used once in a symbol and shall be placed at beginning of the barcode text, and after any ECI indicator block or structured append block (if exists), followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number), which is enclosed in square brackets. For example:

    ```
    \f[01]00012345678905
    ```

  - Insert a GS character (ASCII value 29) as the data field separator to the barcode text. In this case, it can be placed anywhere but at beginning of the barcode text. For example:

    ```
    \t0100012345678905215l2\f2018
    ```

  See also the "Character set" section above.
- **\t**: Insert a FNC1 character in "First position" mode to barcode text, or insert a GS character (ASCII value 29) as the data field separator to the barcode text. There are two usages:
  - Insert a FNC1 character in "First position" mode to the barcode text. It's completely equivalent to escape sequence "\f" that inserts the FNC1 character in "First position" mode. In this case, it shall only be used once in a symbol and shall be placed at beginning of the barcode text, and after any ECI indicator block or structured append block (if exists), followed by an Application Identifier (2 or more digits). For example:

    ```
    \t0100012345678905
    ```

  - Insert a GS character (ASCII value 29) as the data field separator to the barcode text. It's completely equivalent to the

escape sequence "\f" that inserts the FNC1 character as the data field separator. In this case, it can be placed anywhere but at beginning of the barcode text. For example:

```
\f0100012345678905215 12\t2018
```

See also the "Character set" section above.

- **\d**: Insert a FNC1 character in "Second position" mode to barcode text, or insert a GS character (ASCII value 29) as the data field separator to the barcode text. There are two usages:
  - Insert a FNC1 character in "Second position" mode to the barcode text. It's completely equivalent to escape sequence "\f" that inserts the FNC1 character in "Second position" mode. In this case, it shall only be used once in a symbol and shall be placed at beginning of the barcode text, and after any ECI indicator block or structured append block (if exists), followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number). For example:

    ```
    \d0100012345678905
    ```

  - Insert a GS character (ASCII value 29) as the data field separator to the barcode text. It's completely equivalent to the escape sequence "\f" that inserts the FNC1 character as the data field separator. In this case, it can be placed anywhere but at beginning of the barcode text. For example:

    ```
    \f0100012345678905215 12\d2018
    ```

  See also the "Character set" section above.

- **\g**: Insert a GS character (ASCII value 29) to the barcode text. When FNC1 is used as a data field separator, it can be used instead of the FNC1 character. For example:

  ```
  \f0100012345678905215 12\g2018
  ```

  See also the "Character set" section above.

- **\e[<ECI_Number>]**: Insert an ECI indicator block to barcode text. See also the "Extended Channel Interpretation (ECI)" section below.

- **\s[<Index>, <Amound>, <Parity>]** Insert a structured append block to barcode text in order to create the symbol in a structured append. See also the "Structured append" section below.

Encoding modes

QR Code has four encoding modes as Numeric mode, Alphanumeric mode, Kanji mode and Byte mode respectively in decreasing order of encoding density. All 256 8-bits value are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The character sets in each mode are listed in the following:

- **Numeric mode**: Encodes data from the decimal digit set (0 to 9) (byte values 48 to 57).

- **Alphanumeric mode**: Encodes data from a set of 45 characters, i.e. 10 numeric digits (0 to 9) (byte values 48 to 57), 26 alphabetic characters (A - Z) (byte values 65 to 90) , and 9 symbols (SP, $, %, *, +, -, ., /, :) (byte values 32, 36, 37, 42, 43, 45 to 47, 58 respectively).

- **Byte mode**: All 8-bit values can be encoded.

- **Kanji mode**: The Kanji mode efficiently encodes Kanji characters in accordance with the Shift JIS system based on JIS X 0208. The Shift JIS values are shifted from the JIS X 0208 values. JIS X 0208 gives details of the shift coded representation.

  It may be possible to achieve a smaller symbol size by using the Kanji mode compaction rules when an appropriate sequence of byte values occurs in the data. You can set the AllowKanjiMode property to false in order to disable the Kanji mode.

The EncodePolicy property indicates how to use these encoding mode by the component. This property can be one of these values (defined in the pQRCode unit):

- **epMixingOptimal**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The optimal combination of encoding modes will be used in order to minimize the symbol size.
- **epMixingQuick**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The combination of encoding modes may not be the optimal one, in order to optimize encoding speed.
- **epSingleMode**: The encoding mode will be selected automatically and applied to entire symbol, based on the barcode text, in other words, the barcode text to be encoded will be analysed, and an appropriates lowest level (highest encoding density) encoding mode will be selected, in order to minimize the symbol size, and the encoding mode will not be switched in the sybmol. Note, the Kanji mode will not be used if the AllowKanjiMode property is set to false.

Extended Channel Interpretation (ECI)

The Extended Channel Interpretation (ECI) protocol allows the output data stream to have interpretations different from that of the default character set. Four broad types of interpretations are supported in QR Code:

- International character sets (or code pages).
- General purpose interpretations such as encryption and compaction.
- User defined interpretations for closed systems.
- Control information for structured append in unbuffered mode.

The ECI protocol provides a consistent method to specify particular interpretations on byte values before printing and after decoding. The ECI is identified by an integer (up to 6 digits) which is encoded in the QR Code symbol by the ECI indicator block. The escape sequence "\e[<ECI_Number>]" is used to place the ECI indicator block to the barcode text:

- **ECI_Number**: The ECI number, it's an integer between 0 and 999999 (including the boundaries), the leading zero is optional.

ECI indicator blocks may be placed anywhere in the barcode text in a single or structured append set of QR Code symbols. For example:

```
ABC\e[123]DEFabc\e[000003]def
```

The default interpretation for QR Code is ECI 000003 representing the ISO/IEC 8859-1 character set.

The AllowEscape property should be set to true in order to place the ECI indicator blocks. Any ECI invoked shall apply until the end of the barcode text, or until another ECI indicator block is encountered. Thus the interpretation of the ECI may straddle two or more symbols.

Structured append

In order to handle larger messages than are practical in a single symbol, a data message can be distributed across several QR Code symbols. Up to 16 QR Code symbols may be appended in a structured format to convey more data. If a symbol is part of a structured append this shall be indicated by a structured append block in barcode text. The escape sequence "\s[<Index>, <Amount>, <Parity>]" is used to place the structured append block to the barcode text:

- **Index**: The position index of the symbol within the set of QR Code symbols in the structured append format. It's an integer between 1 and 16 (including the boundaries) in string format.
- **Amount**: The total amount of the symbol within the set of QR Code symbols in the structured append format. It's an integer between 2 and 16 (including the boundaries) in string format.
- **Parity**: The parity data. It shall be an 8-bit byte value obtained by using the GetParity method with the original input data as its parameter. It is identical in all symbols in the structured append, enabling it to be verified that all symbols read form part of the same structured append message.

The AllowEscape property should be set to true in order to place the structured append block. The structured append block may only be placed once in the barcode text. Also, it shall be placed at beginning of the barcode text. The OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur if the structured append block be placed more than

once, or it isn't placed at beginning of the barcode text. The following is the examples of structured append:

```
\s[2, 5, 125]ABCDEFGabcdefg1234567890
```

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Inversed
- Mirrored
- EncodePolicy
- MaxSliceLen
- MinVersion
- MaxVersion
- ECCLevel
- ECCLevelUpgrade
- AllowKanjiMode
- AllowEscape
- CurrentVersion (Read only)
- CurrentECCLevel (Read only)

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize
- GetParity

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.17 TBarcode2D_RSS14

The component is used to create the RSS-14 (Reduce Space Symbology) barcode symbols, including RSS-14 (Standard), RSS-14 Truncated, RSS-14 Stacked, and RSS-14 Stacked Omnidirectional. It's defined in the pRss14 unit.

RSS-14 is a continuous linear symbology capable encoding the full GS1 14-digit EAN.UCC item identification number (Global Trade Item Numbers, GTINs). The first digit represents the Indicator digit to indicate packaging level. The following twelve digits are the GS1 Company Prefix and the Item Reference. The last digit represents the Check Digit. A leading Application Identifier (01) is implied and is not encoded in the symbol.

Styles

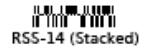
There are four styles (versions) of RSS-14 symbols, as described in following list:

- **Standard**: The standard style RSS-14 barcode symbol encodes the full 14-digit EAN.UCC item identification (Global Trade Item Numbers, GTINs) in a symbol that can be omnidirectionally scanned by suitably configured point-of-sale laser scanners.

- **Truncated**: RSS-14 Truncated is structured and encoded in the same way as the standard RSS-14 format, except its height is reduced to a 13 modules minimum. It may be used for small items, instead of RSS Limited It may also be used when the four-column 2D component is desired in order to minimize the height of an EAN.UCC Composite symbol.

  RSS-14 Truncated is designed to be read by scanners such as wands, handheld lasers, and linear and 2D imagers. It cannot be read efficiently by omnidirectional flat-bed and presentation scanners.

- **Stacked**: RSS-14 Stacked is an RSS-14 Truncated two-row format. It may be used for small items instead of RSS Limited when the available space is too narrow for RSS Limited Moreover, the narrower width of RSS-14 Stacked might allow for a larger module width and potentially higher print quality. However, RSS Limited or RSS-14 Truncated should be used in preference to the stacked format whenever space permits without reducing module width, as they are easier to scan with a wand or linear scanner.



  RSS-14 Stacked is designed to be read by scanners such as wands, handheld lasers, and linear and 2D imagers. It cannot be read efficiently by omnidirectional flat-bed and presentation scanners.

- **Stacked Omnidirectional** RSS-14 Stacked Omnidirectional is a full height RSS-14 two-row format. It can be used instead of RSS-14 for omnidirectional scanning applications where the different aspect ratio is needed.

You can use the Style property to specify which style will be used. It can be one of these values: rsStandard, rsTruncated, rsStacked, and rsStackedOmnidirectional, corresponding to the Standard, Truncated, Stacked, and Stacked Omnidirectional styls. They are defined in the pRss14 unit.

Symbol size

- **RSS-14 (Standard)**: Normally, the overall size of this format is 96 modules wide by a minimum of 33 modules high. You can use the TotalHeight property to specify the height for an RSS-14 (Standard) symbol, in modules.
- **RSS-14 Truncated**: Normally, the overall size of this format is 96 modules wide by a minimum of 13 modules high. You can use the TotalHeight property to specify the height for an RSS-14 Truncated symbol, in modules.
- **RSS-14 Stacked**: Normally, the top row is 5 modules high and the bottom row is 7 modules high with a 1 module (minimum) high separator pattern between the two rows. So the overall size of this format is 50 modules wide by 13 modules high. You can use the SeparatorRowHeight property to specify the height of separator pattern between the two rows, in modules. And use the TotalHeight property to specify the total height for an RSS-14 Stacked symbol, in modules. the height of separator pattern is included in the total height.
- **RSS-14 Stacked Omnidirectional** Normally, Each row is 33 modules high minimum with a 3 modules high separator pattern between the two rows. So the overall size of this format is 50 modules wide by 69 modules high minimum. You can use the SeparatorRowHeight property to specify the height of separator pattern between the two rows, in modules. And use the TotalHeight property to specify the total height for an RSS-14 Stacked Omnidirectional symbol, in modules. the height of separator pattern is included in the total height.

If both the Link2D and Show2DSeparator properties are set to Ture, a contiguous separator pattern is represented and its minimum height is 1 module (the pattern height can be specified by the SeparatorRowHeight property). In this case, the height of the RSS-14 symbol will be reached to increase the contiguous separator pattern height.

Quiet zones

No quiet zones is required outside the bounds of the RSS-14 symbol. The first and last elements may appear wider than one module without affecting the symbol if the adjacent background area is the same color. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 0.

Character set

- All 10 numeric characters, i.e. 0 through 9.

Check digit

All styles of RSS-14 symbols have a check digit, it is a form of redundancy check used for error detection. It consists of a single numeric character computed from the other character in the EAN.UCC item identification number.

If the AutoCheckDigit property is set to false, the check digit shall be included in the barcode text. Otherwise, It wii be appended automatically by the component, so the check digit isn't required in the barcode text.

Data capactiy

All styles of RSS-14 symbologies encode the full GS1 14-digit EAN.UCC item identification number (Global Trade Item Numbers GTINs) in a symbol, including a leading indicator digit, 12 data characters (GS1 Company Prefix and the Item Reference), and a check digit. Note, A leading Application Identifier (01) is implied and is not encoded in the symbol, so it isn't required in the barcode text.

If the AutoCheckDigit property is set to false, the check digit shall be included in the barcode text, so the length of barcode text shall is 14 numeric characters, and the last numeric character in the barcode text is the check digit.

If the AutoCheckDigit property is set to true, the check digit will be appended automatically by the component, so the length of barcode text shall is 13 numeric characters, and the check digit isn't required in the barcode text.

Composite symbols

Any styles of an RSS-14 barcode symbol can be used together with a CC-A or CC-B barcode symbol to create the EAN.UCC composite symbol. Only a 4-column CC-A or a 4-column CC-B symbol can be used together with an RSS-14 (Standard) or an RSS-14 Truncated symbol. Only a 3-column CC-A or a 3-column CC-B symbol can be used together with an RSS-14 Stacked or an RSS-14 Stacked Omnidirectional symbol.

If you use it together with a CC-A or a CC-B component that's from another components package, the Link2D property should be set to true in order to denote the RSS-14 symbol is used as the linear component in an EAN.UCC composite symbol. In this case, the Show2DSeparator property specifies whether to represent the contiguous separator pattern between the RSS-14 linear and the 2D components in the EAN.UCC composite symbol.

If you use it together with the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component that's in this **2D Barcode VCL Components** package, it can be used as the Linear property's value of the 2D barcode component to create the EAN.UCC composite symbols. In such case, only the Barcode, Data (only for Delphi/C++ Builder 2009 or later),AutoCheckDigit, Style, TotalHeight, and SeparatorRowHeight properties are useful, the value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D and Show2DSeparator properties will be set to true automatically). The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed automatically depending on the value of the Style property. In addition, the Image property should not be normally set, the Image property's value of the 2D component will be used.

See also the "How to generate the EAN.UCC composite barcode symbol".

See diagram:

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- TotalHeight
- Link2D
- Show2DSeparator
- Style
- SeparatorRowHeight
- AutoCheckDigit

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.18 TBarcode2D_RSSExpanded

The component is used to create the RSS Expanded barcode symbols, including RSS Expanded (Standard) and RSS Expanded Stacked. It's defined in the pRssExpanded unit.

RSS Expanded is a variable length linear symbology capable of encoding up to 74 numeric or 41 alphabetic characters of AI element string data internally represented as a binary number. RSS Expanded can be used to encode primary and supplementary data intended for use in retail point-of-

sale and other applications where the scanners and application software have been appropriately programmed.

RSS Expanded can be scanned and decoded in up to 22 segments and then reconstructed. This facilitates omnidirectional scanning.

Styles

There are two styles (versions) of RSS Expanded symbols, as described in following list:

- **Standard**: RSS Expanded (Standard) is a single row, variable length linear symbology. It can identify small items and carry more information than the current EAN/UPC barcode.

- **Stacked**: RSS Expanded Stacked is an RSS Expanded multi-row format. It may be stacked in two to eleven rows. RSS Expanded Stacked is used when the symbol area or print mechanism is not wide enough to accommodate the full single row symbol.

You can use the Style property to specify which style will be used. It can be one of values rsStandard and rsStacked, corresponding to the Standard and Stacked styls. They are defined in the pRss14Expanded unit.

Symbol size

In both RSS Expanded (Standard) and RSS Expandedn Stacked symbols, the barcode text will be encoded as a symbol segments sequence, there are between 4 and 22 symbol segments (including the boundaries) in the sequence. The symbol sizes are based on the number of symbol segments, and are described in following list:

- **Standard**: It's a variable length symbology, the symbol width is based on the number of symbol segments. The symbol height are specified by the RowHeight property, in modules.
- **Stacked**: The symbol width is based on the number of symbol segments in each stacked row. The symbol height is based on the number of symbol stacked rows and the height of each stacked row.

  You can use the RowSymbols property to specify the number of symbol segments in each row for an RSS Expanded Stacked symbol, between 2 and 20 (including the boundaries, only even values are valid), the barcode symbol will be stacked in two to eleven rows based on the total numbers of symbol segments and the number of symbol segments in each row. There are three separator patterns between the stacked rows.

  Each row is minimum 34 modules high, and each separator pattern is minimum 1 module high. You can use the RowHeight property to specify the height for each row, in modules. And use the SeparatorRowHeight property to specify the height of each separator pattern, in modules.

If both the Link2D and Show2DSeparator properties are set to Ture, a contiguous separator pattern is represented and its minimum height is 1 module, the height can be specified by the SeparatorRowHeight property too.

Quiet zones

No quiet zones is required outside the bounds of the RSS Expanded symbol. The first and last elements may appear wider than one module without affecting the symbol if the adjacent background area is the same color. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 0.

Character set

A subset of ISO/IEC 646, consisting of the upper and lower case letters (A-Z, a-z), digits (0-9), space, and 19 selected punctuation characters (!:%&()*+,-./:;<=>?_) in addition to the special function character, FNC1.

If you want insert the FNC1, please insert the "^" character instead.

Note, in the barcode text, all Application Identifiers should be enclosed in parentheses "(" and ")", the parentheses are only for identifying Application Identifiers, and they are not encoded into the barcode symbol. If you want encode the parentheses "(" and ")" in an Application Identifier element string, please use the "{" and "}" instead.

Data capactiy

74 numeric or 41 alphabetic characters.

Note, the RSS Expanded data capacity depends on the encodation method. The maximum is 74 digits for (01) + other AIs, the maximum is 70 digits for any AIs, and the maximum is 77 digits for (01) + (392x) + any AIs.

Composite symbols

Any style of an RSS Expanded barcode symbol can be used together with a 4-column CC-A or a 4-column CC-B barcode symbol to create the EAN.UCC composite symbol. In this case, the RSS Expanded Stacked symbol shall contain at least four symbol characters within its top row, in other words, the value of RowSymbols property should be greater than or equal to 4.

If you use it together with a CC-A or a CC-B component that's from another components package, the Link2D property should be set to true in order to denote the RSS Expanded symbol is used as the linear component in an EAN.UCC composite symbol. In this case, the Show2DSeparator property specifies whether to represent the contiguous separator pattern between the RSS Expanded linear and the 2D components in the EAN.UCC composite symbol.

If you use it together with the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component that's in this **2D Barcode VCL Components** package, it can be used as the Linear property's value of the 2D barcode component to create the EAN.UCC composite symbols. In such case, only the Barcode, Data (only for Delphi/C++ Builder 2009 or later), Style, RowSymbols, RowHeight, and SeparatorRowHeight properties are useful, and the value of RowSymbols property should be greater than or equal to 4. The value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D and Show2DSeparator properties will be set to true automatically). The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed to 4 automatically. In addition, the Image property should not be normally set, the Image property's value of the 2D component will be used.

See also the "How to generate the EAN.UCC composite barcode symbol".

See diagram:

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- Style
- RowHeight
- RowSymbols
- SeparatorRowHeight
- Link2D
- Show2DSeparator

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.1.19 TBarcode2D_RSSLimited

The component is used to create the RSS Limited barcode symbols. It's defined in the pRssLimited unit.

RSS Limited is a continuous linear symbology capable encoding the entire GS1 14-digit EAN.UCC item identification number (Global Trade Item Numbers, GTINs) set with Indicator digits 1 and 0. The first digit represents the Indicator digit to indicate packaging level (only 0 and 1). The following twelve digits are the GS1 Company Prefix and the Item Reference. The last digit represents the Check Digit. A leading Application Identifier (01) is implied and is not encoded in the symbol.

Symbol size

Normally, the overall size of the RSS Limited symbol is 74 modules wide by a minimum of 10 modules high.

If both the Link2D and Show2DSeparator properties are set to Ture, a contiguous separator pattern is represented and its minimum

height is 1 module (the pattern height can be specified by the SeparatorRowHeight property). In this case, the height of the RSS Limited symbol will be reached to increase the contiguous separator pattern height. Namely, If height of the contiguous separator pattern is 1 module, then the minimum height of the RSS Limited symbol is 11 modules.

The TotalHeight property can be used to specify the height for an RSS Limited symbol, in modules. If the contiguous separator pattern is represented, its height is included in the property value.

Quiet zones

No quiet zones is required outside the bounds of the RSS Limited symbol. The first and last elements may appear wider than one module without affecting the symbol if the adjacent background area is the same color. So the minimum values of LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties are equal to 0.

Character set

- All 10 numeric characters, i.e. 0 through 9.

Check digit

The RSS Limited symbols have a check digit, it is a form of redundancy check used for error detection. It consists of a single numeric character computed from the other character in the EAN.UCC item identification number.

If the AutoCheckDigit property is set to false, the check digit shall be included in the barcode text. Otherwise, It wii be appended automatically by the component, so the check digit isn't required in the barcode text.

Data capactiy

The RSS Limited symbology encodes the entire GS1 14-digit EAN.UCC item identification number (Global Trade Item Numbers GTINs) set with Indicator digits 1 and 0 in a symbol, including a leading indicator digit (only 0 and 1), 12 data characters (GS1 Company Prefix and the Item Reference), and a Check digit. Note, A leading Application Identifier (01) is implied and is not encoded in the symbol, so it isn't required in the barcode text.

Note, First numeric character is the Indicator digit in the barcode text, only 1 and 0 can be encoded.

If the AutoCheckDigit property is set to false, the check digit shall be included in the barcode text, so the length of barcode text shall is 14 numeric characters, and the last numeric character in the barcode text is the check digit.

If the AutoCheckDigit property is set to true, the check digit will be appended automatically by the component, so the length of barcode text shall is 13 numeric characters, and the check digit isn't required in the barcode text.

Composite symbols

The RSS Limited barcode symbol can be used together with a 3-column CC-A or a 3-column CC-B barcode symbol to create the EAN.UCC composite symbol.

If you use it together with a CC-A or a CC-B component that's from another components package, the Link2D property should be set to true in order to denote the RSS Limited symbol is used as the linear component in an EAN.UCC composite symbol. In this case, the Show2DSeparator property specifies whether to represent the contiguous separator pattern between the RSS Limited linear and the 2D components in the EAN.UCC composite symbol.

If you use it together with the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component that's in this **2D Barcode VCL Components** package, it can be used as the Linear property's value of the 2D barcode component to create the EAN.UCC composite symbols. In such case, only the Barcode, Data (only for Delphi/C++ Builder 2009 or later),AutoCheckDigit, TotalHeight, and SeparatorRowHeight properties are useful, the value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D and Show2DSeparator properties will be set to true automatically). The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed to 3 automatically. In addition, the Image property should not be normally set, the Image property's value of the 2D component will be used.

See also the "How to generate the EAN.UCC composite barcode symbol".

See diagram:

Properties:

- Image
- Barcode
- Data (*)
- Module
- BarColor
- SpaceColor
- Orientation
- Stretch
- LeftMargin
- TopMargin
- BarcodeWidth
- BarcodeHeight
- ShowQuietZone
- LeadingQuietZone
- TopQuietZone
- TrailingQuietZone
- BottomQuietZone
- Locked
- TotalHeight
- Link2D
- SeparatorRowHeight
- Show2DSeparator
- AutoCheckDigit

Methods:

- Create
- Destroy
- Assign
- Clear
- Draw
- Size
- CopyToClipboard
- DrawTo
- DrawToSize
- Print
- PrintSize

Events:

- OnChange
- OnEncode
- OnInvalidChar
- OnInvalidLength
- OnInvalidDataChar (*)
- OnInvalidDataLength (*)
- OnDrawBarcode

(*): The Data property, OnInvalidDataLength and OnInvalidDataChar events are available only for the Delphi/C++ Builder 2009 or later.

4.2 TDBBarcode2D

TDBBarcode2D provides an interface between a TDataSource component and a barcode component. It connects the barcode component to a dataset. It's defined in the pDBBarcode2D unit.

Use TDBBarcode2D to provide a conduit between a dataset and a barcode component that enable use the data underlying the dataset to represent the barcode symbol.

See also the "How to use the barcode components with a database".

Properties:

- DataField
- DataSource
- Barcode2D
- BindProperty (*)
- ReadOnly
- Field

(*): The BindProperty property is available only for the Delphi/C++ Builder 2009 or later.

4.3 TLogo2D

The component is used to add a logo picture to the barcode symbol. It is defined in the pLogo2D unit.

Properties:

- Barcode2D
- Picture
- Width
- Height
- Padding
- HorizontalAlignment
- HorizontalMargin
- VerticalAlignment
- VerticalMargin
- AlignToModule

Methods:

- Create
- Destroy
- Assign

4.4 TSave2D

It is the base class of all image saving components, these components can be used to save barcode symbol to picture file in multiple formats. And it cannot be instantiated. It's defined in the pSave2D unit.

Properties:

- Barcode2D

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.1 TSave2D_Bmp

The component is used to save the barcode symbol to image file in Bitmap image (BMP) format. It is defined in the pSave2D_Bmp unit. If a logo picture is added using the TLogo2D component, it will be saved in the image file too.

Properties:

- Barcode2D
- ColorDepth
- XPPM
- YPPM

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.2 TSave2D_Emf

The component is used to save the barcode symbol to image file in Enhaced Metafile Format (EMF). It is defined in the pSave2D_Emf unit.

Note, if a logo picture is added using the TLogo2D component, it will not be saved in the image file.

Properties:

- Barcode2D
- DeviceWidthInPixel
- DeviceHeightInPixel
- DeviceWidthInMM
- DeviceHeightInMM
- DeviceWidthInUM
- DeviceHeightInUM
- EmbedBarcodeText

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.3 TSave2D_Eps

The component is used to save the barcode symbol to image file in Encapsulated PostScript (EPS) format. It is defined in the

pSave2D_Eps unit.

Note, if a logo picture is added using the TLogo2D component, it will not be saved in the image file.

Properties:

- Barcode2D
- EmbedBarcodeText

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.4 TSave2D_Gif

The component is used to save the barcode symbol to image file in Graphics Interchange Format (GIF). It is defined in the pSave2D_Gif unit. If a logo picture is added using the TLogo2D component, it will be saved in the image file too.

Properties:

- Barcode2D
- Version
- ColorDepth
- Interlaced
- EmbedBarcodeText

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.5 TSave2D_Jpg

The component is used to save the barcode symbol to image file in JPEG image (JPG, JPEG) format. It is defined in the pSave2D_Jpg unit. If a logo picture is added using the TLogo2D component, it will be saved in the image file too.

Properties:

- Barcode2D
- Progressive
- Quality

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.6 TSave2D_Png

The component is used to save the barcode symbol to image file in Portable Network Graphics (PNG) format. It is defined in the pSave2D_Png unit. If a logo picture is added using the TLogo2D component, it will be saved in the image file too.

Properties:

- Barcode2D
- BarOpacity
- SpaceOpacity
- ColorDepth
- Interlaced
- CompressionLevel
- EmbedBarcodeText

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.7 TSave2D_Svg

The component is used to save the barcode symbol to image file in Scalable Vector Graphics (SVG) format. It is defined in the pSave2D_Svg unit. If a logo picture is added using the TLogo2D component, it will be saved in the image file too.

Properties:

- Barcode2D
- BarOpacity
- OriginalLogo
- SpaceOpacity
- EmbedBarcodeText

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.4.8 TSave2D_Wmf

The component is used to save the barcode symbol to image file in Windows Metafile Format (WMF). It is defined in the pSave2D_Wmf unit.

Note, if a logo picture is added using the TLogo2D component, it will not be saved in the image file.

Properties:

- Barcode2D
- Placeable
- LogicUnitsPerInch

Methods:

- Create
- Destroy
- Assign
- Save

Events:

- OnSave
- OnSaved

4.5 TUserWorkArea

It contains fields to specify the parameters (e.g. position, size, etc.) for the barcode symbol. It is defined in the pBarcode2D unit.

Syntax:

```
type
  { Defined in the pBarcode2D unit }
  TUserWorkArea = record
    WCanvas: TCanvas;
    WBarcode: string;
    WBarColor: TColor;
    WSpaceColor: TColor;
    WModule: Integer;
    WLeftInPixels: Integer;
    WTopInPixels: Integer;
    WAngle: Integer;
    WShowQuietZone: Boolean;
    WMirror: Boolean;
    WAnsiBarcode: AnsiString;
    WBarcodeLength: Integer;
    WQuietZoneWidthInModules_Left: Integer;
    WQuietZoneWidthInModules_Top: Integer;
    WQuietZoneWidthInModules_Right: Integer;
    WQuietZoneWidthInModules_Bottom: Integer;
    WSymbolZoneWidthInModules: Integer;
    WSymbolZoneHeightInModules: Integer;
    WTotalWidthInPixels: Integer;
    WTotalHeightInPixels: Integer;
    WSymbolZoneOffsetInPixels_Left: Integer;
    WSymbolZoneOffsetInPixels_Top: Integer;
    WAlpha: Double;
    WOrigin: TPoint;
    WDensityRate: Double;
  end;
```

Fields:

- **WCanvas**: TCanvas; The target canvas that the barcode symbol will be represented on it.
- **WBarcode**: string; The barcode text which will be encoded to the barcode symbol.

  If you use the Barcode property, or use Draw, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method, It's equal to the value of Barcode property.

  If you use the DrawTo (Syntax 2), DrawToSize (Syntax 2), Print (Syntax 2), or PrintSize (Syntax 2) method, it is equal to their Barcode parameter value.

  For Delphi/C++ Builder 2009 or later, if you use the Data property, or use DrawTo (Syntax 3), DrawToSize (Syntax 3), Print (Syntax 3), and PrintSize (Syntax 3) methods, it's an empty string.
- **WBarColor**: TColor; It's the color used by all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.
- **WSpaceColor**: TColor; It's the color used by all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also the leading, trailing, top, and bottom quiet zones are represented using the color too.
- **WModule**: Integer; It's the module size in pixels (Draw, DrawTo) or dots (Print). See also the "Module" property.
- **WLeftInPixels**: Integer; It's the margin between the barcode symbol and the left side of the canvas in pixels (Draw, DrawTo) or dots (Print). If the quiet zones are represented, they are included in the barcode symbol. See also the "LeftMargin" property.
- **WTopInPixels**: Integer; It's the margin between the barcode symbol and the top side of the canvas in pixels (Draw, DrawTo) or dots (Print). If the quiet zones are represented, they are included in the barcode symbol. See also the "TopMargin" property.
- **WAngle**: Integer; It's the angle to rotate the barcode symbol anticlockwise, in degrees.
- **WShowQuietZone**: Boolean; Denotes whether the leading, trailing, top and bottom quiet zones are represented. See also the "ShowQuietZone" property.
- **WMirror**: Boolean; Denotes whether the barcode symbol is reversed right to left (horizontal mirror reversal). See also the "Mirrored" property.
- **WAnsiBarcode**: AnsiString; The barcode text which will be encoded to the barcode symbol, in AnsiString format.

  If you use the Barcode property, or use Draw, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method, it is an AnsiString converted from the Barcode property (if you used the OnEncode event function, it's equal to its Data parameter value).

  If you use the DrawTo (Syntax 2), DrawToSize (Syntax 2), Print (Syntax 2), or PrintSize (Syntax 2) method, it is an AnsiString converted from their Barcode parameter value (if you used the OnEncode event function, it's equal to its Data parameter value).

  For Delphi/C++ Builder 2009 or later, if you use the Data property, it's equal to the value of Data property. If you use DrawTo (Syntax 3), DrawToSize (Syntax 3), Print (Syntax 3), or PrintSize (Syntax 3) method, it is equal to their Data parameter value.
- **WBarcodeLength**: Integer; It's the length of WAnsiBarcode feild's value by character count.
- **WQuietZoneWidthInModules_Left**: Integer; It's the width of leading quiet zone if it is represented, in modules. See also the "LeadingQuietZone" property.
- **WQuietZoneWidthInModules_Top**: Integer; It's the height of top quiet zone if it is represented, in modules. See also the "TopQuietZone" property.
- **WQuietZoneWidthInModules_Right**: Integer; It's the width of trailing quiet zone if it is represented, in modules. See also the "TrailingQuietZone" property.
- **WQuietZoneWidthInModules_Bottom**: Integer; It's the height of bottom quiet zone if it is represented, in modules. See also the "BottomQuietZone" property.

- **WSymbolZoneWidthInModules**: Integer; It's the height of entire barcode symbol before the barcode symbol is rotated, in modules. The quiet zones aren't included even if they are represented.
- **WSymbolZoneHeightInModules**: Integer; It's the width of entire barcode symbol before the barcode symbol is rotated, in modules. The quiet zones aren't included even if they are represented.
- **WTotalWidthInPixels**: Integer; It's the height of entire barcode symbol before the barcode symbol is rotated, in pixels (Draw, DrawTo) or dots (Print). The quiet zones are included if they are represented.
- **WTotalHeightInPixels**: Integer; It's the width of entire barcode symbol before the barcode symbol is rotated, in pixels (Draw, DrawTo) or dots (Print). The quiet zones are included if they are represented.
- **WSymbolZoneOffsetInPixels_Left**: Integer; It's the horizontal offset from the upper-left corner of entire barcode symbol (the quiet zones are included if they are represented) to the upper-left of the barcode symbol (the quiet zones aren't included even if they are represented) before the barcode symbol is rotated, in in pixels (Draw, DrawTo) or dots (Print).
- **WSymbolZoneOffsetInPixels_Top**: Integer; It's the vertical offset from the upper-left corner of entire barcode symbol (the quiet zones are included if they are represented) to the upper-left of the barcode symbol (the quiet zones aren't included even if they are represented) before the barcode symbol is rotated, in in pixels (Draw, DrawTo) or dots (Print).
- **WAlpha**: Double; It's the angle that the barcode symbol will be rotated, in radian.
- **WOrigin**: TPoint; The coordinate of the upper-left corner of the barcode symbol after the barcode symbol is rotated, in pixels (Draw, DrawTo) or dots (Print).
- **WDensityRate:** Double; It is the ratio of the horizontal logical DPI and veritical logical DPI of the canvas.

Chapter 5. FAQs

5.1 How to download the full version

Please download the installation package using the download link that's sent from us after you purchase the components package, within the validity period.

If the download link expired, please visit the "Manage your licenses" page to request a new download link.

Please open the page then enter your order ID, license user name or license e-mail address to locate your order, then click on the oder ID to display it, choose a license and click on the "**Request a new download link**", the new download link will be sent to this license e-mail address automatically. Please use the new download link whthin the validity period.

5.2 I forgot my license key

If you forgot the license key, please visit the "Manage your licenses" page to retrieve it. Please open the page then enter your order ID, license user name or license e-mail address to locate your order, then click on the oder ID to display it, choose a license and click on the "**Retrieve the license key**", the license key will be sent to the license e-mail address automatically.

Chapter 6. Information

6.1 Support

This help is designed to be used on-screen. Many troubleshooting questions can be answered by this help. If your question is not covered in the help document, please read the next section.

Technical Support

If you have a specific technical issue, if you would like to send general comments about the product, please reads the following:

- Please read the "Frequently Asked Question" in our web site to see whether your question is already answered.
- You may email the technical support issues to support@han-soft.com. The registered users will get priority, so include your order number in the email to ensure a prompt response.

In order to provide more accurate service, please provides the following information:

- Whether the problem can be reproduced? How is it reproduced?
- What development system do you use?
- Which version of 2D Barcode VCL components do you use?

Feedback

If you have any comments or concerns about this product please send them to feedback@han-soft.com. Your feedback is very important for us because it helps to us to make our software better and more efficient. Many of its features have been heavily influenced by comments from users. So if you have a grand idea for a new feature, or a better way of doing something, please drop us a note.

6.2 Purchase

The 2D Barcode VCL Components is a Shareware product. If you find it useful, please purchase the full version of this product. After your purchase you will receive an e-mail with a download link of full version for downloading.

Why should I purchase

After your purchase you can continue to use the 2D Barcode VCL Components and entitles you to the following benefits:

- The trial version is fully functional, though limited, and adds a watermark that appears in the final 2D Barcode symbol. So purchase will allow you to use 2D Barcode VCL Components without limitations.
- You are entitled to free upgrades during 1 year since the date of your purchase. This includes both major and minor upgrades in appropriate time period. It means that during one year you can download and install the latest versions of the software from our site.
- You will be entitled to a 50% discount for all future major upgrades if you purchased the software more than one year ago, and allow you to have free upgrades for another year.

- You will be always entitled to free minor upgrades for the version that is purchased.
- You will also have priority in technical support.
- Purchasing the product may also entitle you to discounts on new software releases from Han-soft.
- Finally, by purchasing the software, you provide us with the resources and incentive to support the software with updates and to develop additional quality products in the future.

How to purchase

You can purchase the 2D Barcode VCL Components by using following steps:

1. Please open the "Purchase" web page of our web site http://www.han-soft.com.
2. In the "Purchase" web page, choose the license type and open the order form web page. Fill in the information on the web page, and choose from the available ordering options that best suits your needs. We accept credit card orders, PayPal orders, orders by phone and fax, checks, purchase orders, and wire transfers.
3. Once the purchase is completed, the download link of full version and the license key will be sent instantaneously to your email address. If you do not receive your download link and license key within a few minutes, please check your SPAM filter inbox.

   Important:

   When completing the order form, please double-check that your email address is correct. If it is not, you will be unable to receive the download link and the license key.

6.3 Contact us

Contact us if you have any questions, comments or suggestions:

Web site

- Main site: http://www.han-soft.com
- Mirror: http://www.han-soft.biz

E-mail

- General information: information@han-soft.com
- Technical support: support@han-soft.com
- Sales: sales@han-soft.com
- Service: service@han-soft.com
- Feedback: feedback@han-soft.com
- Web information: webmaster@han-soft.com

6.4 End-User License Agreement

This End-User License Agreement ("EULA") is a legal agreement between you (either an individual or a single entity) and the Han-soft, which includes computer software and may include associated media, printed materials, and "online" or electronic documentation ("SOFTWARE PRODUCT").

By installing, copying, or otherwise using the SOFTWARE PRODUCT, you agree to be bound by the terms of this EULA. If you do not agree to the terms of this EULA, do not install or use the SOFTWARE PRODUCT; you may, however, return it to your place of purchase for a full refund.

SOFTWARE PRODUCT LICENSE

The SOFTWARE PRODUCT is protected by copyright laws and international copyright treaties, as well as other intellectual property laws and treaties. The SOFTWARE PRODUCT is licensed, not sold.

1. **GRANT OF LICENSE.**

   This EULA grants you the following rights:

   - **Installation and Use.**

     You may install and use an unlimited number of copies of the TRIAL SOFTWARE PRODUCT.

   - **Reproduction and Distribution.**

     You may reproduce and distribute an unlimited number of copies of the TRIAL SOFTWARE PRODUCT; provided that each copy shall be a true and complete copy, including all copyright and trademark notices, and shall be accompanied by a copy of this EULA. Copies of the TRIAL SOFTWARE PRODUCT may be distributed as standalone product or included with your own product.

   - **Purchase.**

     You may use the registered SOFTWARE PRODUCT on that number of computers for which you have purchased a separate license as indicated on the invoice or sales receipt. If the SOFTWARE PRODUCT is installed on a network server or other storage device, you must purchase a license for each separate computer on which the SOFTWARE PRODUCT is used. A license for the SOFTWARE PRODUCT may not be shared by alternating use of the SOFTWARE PRODUCT between different computers. The primary user of a computer for which a license has been purchased may make and use one copy of the SOFTWARE PRODUCT on his or her portable computer. You may also make one copy of the SOFTWARE PRODUCT for back-up or archival purposes. Otherwise, you may not copy the SOFTWARE PRODUCT in whole or in part. You may not transfer your rights under this license.

2. **DESCRIPTION OF OTHER RIGHTS AND LIMITATIONS.**

   - **Limitations on Components and Source Code.**

     You may distribute the runtime packages with your end-user applications. You must not distribute the library in such a way that it allows direct programmatic access to the SOFTWARE PRODUCT, or such that it competes with the SOFTWARE PRODUCT in any way - this library is intended for inclusion in end user products. You may not publish or otherwise expose the licensed source code outside of your organization whatsoever. You may modify the source code as required for your own software products but the code may only be used to produce compiled software. Software built incorporating our licensed source code must not allow programmatic access to the software components or in any way compete with our software products.

   - **Limitations on Reverse Engineering, Decompilation, and Disassembly.**

     You may not reverse engineer, decompile, or disassemble the SOFTWARE PRODUCT, except and only to the extent that such activity is expressly permitted by applicable law notwithstanding this limitation.

   - **Separation of Components.**

     The SOFTWARE PRODUCT is licensed as a single product. Its components parts may not be separated for use on more than one computer.

   - **Software Transfer.**

     You may permanently transfer all of your rights under this EULA, provided the recipient agrees to the terms of this EULA.

- **Termination.**

  Without prejudice to any other rights, the Author of this Software may terminate this EULA if you fail to comply with the terms and conditions of this EULA. In such event, you must destroy all copies of the SOFTWARE PRODUCT and all of its components parts.

- **Distribution.**

  The SOFTWARE PRODUCT may not be sold or be included in a product or package which intends to receive benefits through the inclusion of the SOFTWARE PRODUCT. The registered SOFTWARE PRODUCT may not b included in any free or non-profit packages or products.

3. **COPYRIGHT.**

   All title and copyrights in and to the SOFTWARE PRODUCT (including but not limited to any images, photographs, animations, video, audio, music, text, and "applets" incorporated into the SOFTWARE PRODUCT), the accompanying printed materials, and any copies of the SOFTWARE PRODUCT are owned by the Author of this Software. The SOFTWARE PRODUCT is protected by copyright laws and international treaty provisions. Therefore, you must treat the SOFTWARE PRODUCT like any other copyrighted material except that you may install the SOFTWARE PRODUCT on single computer provided you keep the original solely for backup or archival purposes.

MISCELLANEOUS

Should you have any questions concerning this EULA, or if you desire to contact the author of this Software for any reason, please contact him at the email address mentioned at the bottom of this EULA.

LIMITED WARRANTY

1. **NO WARRANTIES.**

   The Author of this Software expressly disclaims any warranty for the SOFTWARE PRODUCT. The SOFTWARE PRODUC and any related documentation is provided "as is" without warranty of any kind, either express or implied, including, without limitation, the implied warranties or merchantability, fitness for a particular purpose, or non-infringement. The entire risk arising out of use or performance of the SOFTWARE PRODUCT remains with you.

2. **NO LIABILITY FOR DAMAGES.**

   In no event shall the author of this Software be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of or inability to use this product, even if the Author of this Software has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

CONTACT US

Please sent email to the sales@han-soft.com email address.

Annex A. Properties

A.1 TBarcode2D

A.1.1 AllowEscape

Specifies whether to allow users to insert the escape sequences to barcode text, in order to place the function characters and additional control information.

Syntax:

```
property AllowEscape: Boolean;
```

Description:

The property specifies whether to allow users to insert the escape sequences to barcode text, in order to place the function characters and additional control information. An escape sequence is a series of characters used to place the function character and special character, or the control information.

Escape sequences use an escape character "\" to change the meaning of the characters which follow it. For example:

```
ABCDEFG\pHIJKLMN
```

Some escape sequences may be followed by a parameters list, all parameters are comma delimited, and must be enclosed by square brackets. For example:

```
\s[1, 2]ABCDEFGHIJKLMN
```

The escape sequences definition is different for every barcode symbology component, please read the "Escape sequences" section in the article of every barcode symbology component.

A.1.2 AllowKanjiMode

(TBarcode2D_MicroQRCode)

Specifies whether to allow the Kanji mode to be used when encoding the barcode text into a Micro QR Code symbol.

Syntax:

```
property AllowKanjiMode: Boolean;
```

Description:

Micro QR Code symbology has four encoding modes as Numeric mode, Alphanumeric mode, Kanji mode and Byte mode respectively in decreasing order of encoding density. The Kanji mode can encode the Kanji characters more effectively. The property specifies whether to allow the Kanji mode to be used when encoding the barcode text into a Micro QR Code symbol.

If the property is set to true, the component will automatically select the encoding mode between Kanji mode and Byte mode when the Kanji characters are encountered, in order to minimize the symbol size. Otherwise, all Kanji characters are always encoded in Byte mode.

See also the "Encoding modes" section in the "TBarcode2D_MicroQRCode" article.

A.1.3 AllowKanjiMode

(TBarcode2D_QRCode)

Specifies whether to allow the Kanji mode to be used when encoding the barcode text into a QR Code symbol.

Syntax:

```
property AllowKanjiMode: Boolean;
```

Description:

QR Code symbology has four encoding modes as Numeric mode, Alphanumeric mode, Kanji mode and Byte mode respectively in decreasing order of encoding density. The Kanji mode can encode the Kanji characters more effectively. The property specifies whether to allow the Kanji mode to be used when encoding the barcode text into a QR Code symbol.

If the property is set to true, the component will automatically select the encoding mode between Kanji mode and Byte mode when the Kanji characters are encountered, in order to minimize the symbol size. Otherwise, all Kanji characters are always encoded in Byte mode.

See also the "Encoding modes" section in the "TBarcode2D_QRCode" article.

A.1.4 AutoCheckDigit

(TBarcode2D_RSS14, TBarcode2D_RSSLimited)

Specifies whether the check digit should be automatically appended to the barcode symbol.

Syntax:

```
property AutoCheckDigit: Boolean;
```

Description:

The RSS-14 and the RSS Limited barcode symbologies encode the GS1 14-digit EAN.UCC item identification number (Global Trade Item Numbers, GTINs) in a symbol, its last digit represents the check digit. It is a form of redundancy check used for error detection. It consists of a single numeric character computed from the other character in the EAN.UCC item identification number.

The property specifies whether the check digit should be automatically appended to the barcode symbol.

If the AutoCheckDigit property is set to false, the check digit shall be included in the barcode text, so the length of barcode text shall is 14 numeric characters, and the last numeric character in the barcode text is the check digit.

If the AutoCheckDigit property is set to true, the check digit will be appended automatically by the component, so the length of barcode text shall is 13 numeric characters, and the check digit isn't required in the barcode text.

Note, the leading Application Identifier (01) is implied and is not encoded in the symbol. so it isn't required in the barcode text.

See also the "Check digit" section in the "TBarcode2D_RSS14" and "TBarcode2D_RSSLimited" articles.

A.1.5 AutoMode

(TBarcode2D_MaxiCode)

Specifies whether to automatically select the mode for a MaxiCode barcode symbol.

Syntax:

```
property AutoMode: Boolean;
```

Description:

The property specifies whether to automatically select the mode for a MaxiCode barcode symbol, depending on the barcode text and the Mode property value.

If the property is set to true, the TBarcode2D_MaxiCode component will automatically select the suitable mode depending on the barcode text and the value of Mode property:

- If the Mode property is set to mode 2 or 3, the factual mode will be selected between mode 2 and 3, depending on the postal code field in the barcode text.
- If the Mode property is set to mode 4, the factual mode will be selected between mode 4 and 5, depending on the length of barcode text. If the length of barcode text is so short that can be encoded by using mode 5, the mode 5 will be used in order to employ enhanced error correction, otherwise the mode 4 will be used in order to accommodate more barcode text.
- If the Mode property is set to mode 5, the mode 5 will be selected always, in order to insure high level of enhanced error correction.
- If the Mode property is set to mode 6, the mode 6 will be selected always, in order to encodes a message used to program the reader system.

In this case, you can use the CurrentMode property to get the factual mode.

If the property is set to false, the mode will be specified by the Mode property. The value of CurrentMode property is always equal to the Mode property value.

See also the "Modes" sections in the "TBarcode2D_MaxiCode" article.

A.1.6 Barcode

Specifies the barcode text.

Syntax:

```
property Barcode: string;
```

Description:

Specifies the barcode text to encode it into the barcode symbol. The OnChange event will occur when the property value is changed. The OnInvalidChar event will occur if there is any invalid character in the Barcode property value, and the OnInvalidLength event will occur if the length of the Barcode property value is invalid.

The property is of type string. For Delphi/C++ Builder 2007 or early, it is of type AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the

converted string in the property. For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Data property.

For Delphi/C++ Builder 2007 or early, you can use the property to encode block of binary (bytes) data; For Delphi/C++ Builder 2009 or later, in order to encode block of binary (bytes) data, please use the Data property, it is of type AnsiString.

For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be entered in the here, otherwise the check digit can be specified by you in here.

Note, if a 2D component (including the TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC) together with a linear component (including the TBarcode2D_RSS14, TBarcode2D_RSSLimited, TBarcode2D_RSSExpanded, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128) is assembled to generate an EAN.UCC composite barcode symbol, the Barcode property of the 2D component only specifies the barcode text for the 2D component, and the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component only specifies the barcode text for the linear component.

A.1.7 BarcodeHeight

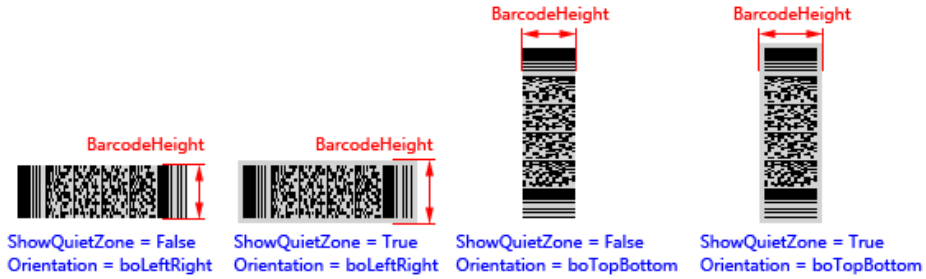
Specifies the distance between the top and bottom of a barcode symbol in pixels.

Syntax:

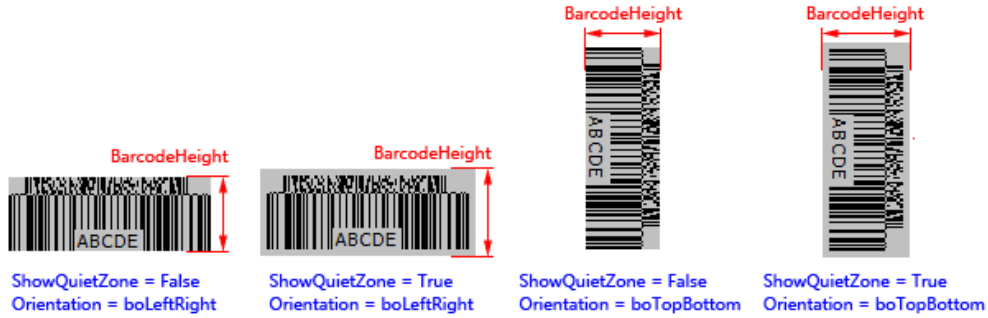
```
property BarcodeHeight: Integer;
```

Description:

Specifies the distance between the top and bottom of a barcode symbol in pixels. If the quiet zones are represented (the ShowQuietZone property is set to true), the top quiet zone and the bottom quiet zone are included. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear component is included in the composite barcode symbol. And the property specifies the height for entire EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



The property is set using the following formula:

- When the property Stretch is set to false:

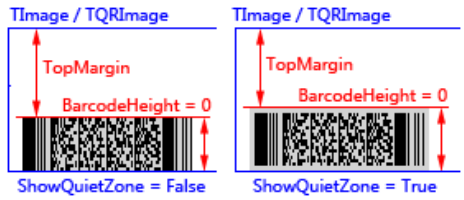
  The BarcodeHeight property will be ignored, and the height of barcode symbol will be calculated based on the Module property value.

  You can get the height value by using the Size method.

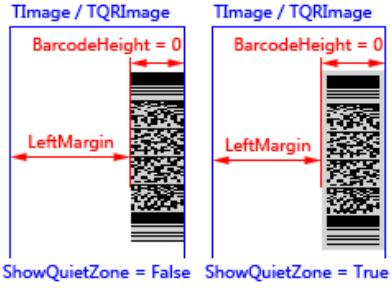
- When the property Stretch is set to true:

  - If the BarcodeHeight property value is equal to zero:

    When the Orientation property is set to "boLeftRight" or "boRightLeft", the TopMargin property value will be subtracted from the height of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode height, the barcode symbol will be reduced/stretched to fit this final height value. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

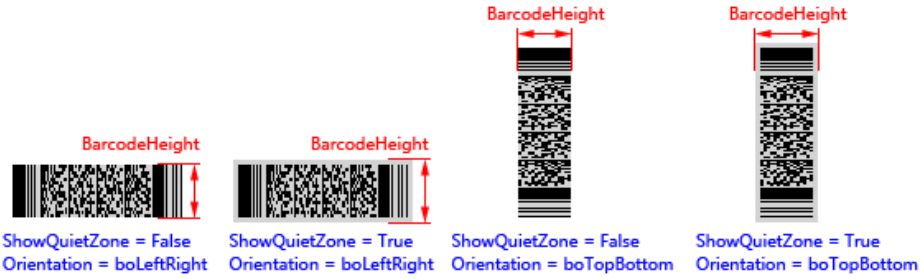
    

    When the Orientation property is set to "boTopBottom" or "boBottomTop", the LeftMargin property value will be subtracted from the width of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode height, the barcode symbol will be reduced/stretched to fit this final height value. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

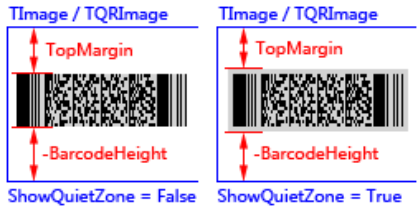
  - If the BarcodeHeight property value is greater than zero:

    The barcode symbol will be reduced/stretched to fit this height value. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

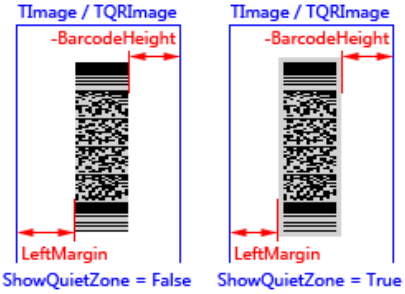


- If the BarcodeHeight property value is less than zero:

  When the Orientation property is set to "boLeftRight" or "boRightLeft", the TopMargin property value and the absolute value of the negative height will be subtracted from the height of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode height, the barcode symbol will be reduced/stretched to fit this final height value (it specifies the bottom margin of the barcode symbol, -1 denotes the bottom margin is 1, -2 denotes the bottom margin is 2, ...). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



  When the Orientation property is set to "boTopBottom" or "boBottomTop", the LeftMargin property value and the absolute value of the negative height will be subtracted from the width of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode height, the barcode symbol will be reduced/stretched to fit this final height value (it specifies the right margin of the barcode symbol, -1 denotes the right margin is 1, -2 denotes the right margin is 2, ...). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



Note:

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, when the linear component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented, the human readable text and the vertical spacing between the text and linear symbol are included in the value of BarcodeHeight property. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

A.1.8 BarcodeWidth

Specifies the distance between the beginning and end of a barcode symbol in pixels.

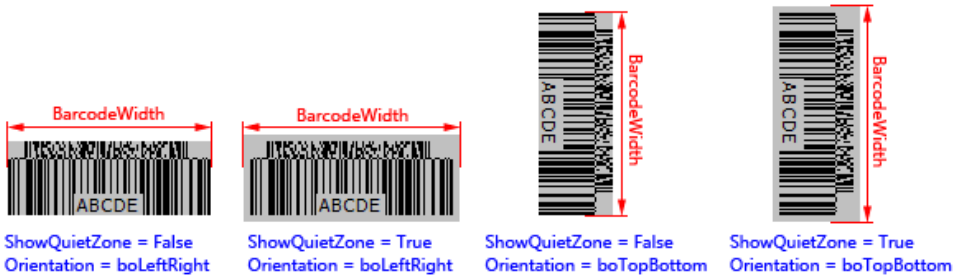
Syntax:

```
property BarcodeWidth: Integer;
```

Description:

Specifies the distance between the beginning and end of a barcode symbol in pixels. If the quiet zones are represented (the ShowQuietZone property is set to true), the leading quiet zone and the trailing quiet zone are included. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear component is included in the composite barcode symbol. And the property specifies the width for entire EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



The property is set using the following formula:

- When the property Stretch is set to false:

  The BarcodeWidth property will be ignored, and the width of barcode symbol will be calculated based on the Module property value.

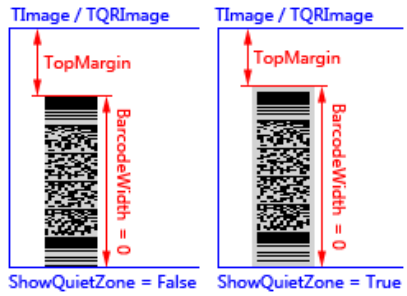
  You can get the width value by using the Size method.

- When the property Stretch is set to true:

- If the property value is equal to zero:

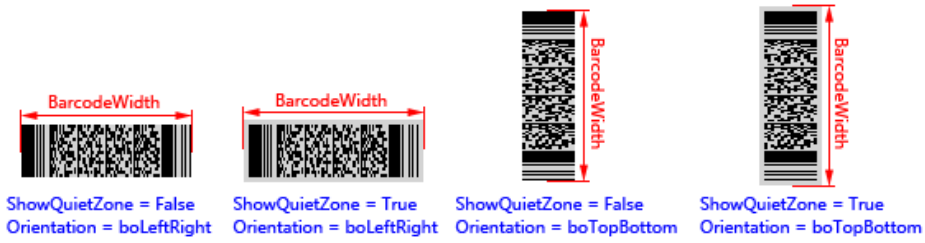
  When the Orientation property is set to "boLeftRight" or "boRightLeft", the LeftMargin property value will be subtracted from the width of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode width, the barcode symbol will be reduced/stretched to fit this final width value. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

  When the Orientation property is set to "boTopBottom" or "boBottomTop", the TopMargin property value will be subtracted from the height of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode width, the barcode symbol will be reduced/stretched to fit this final width value. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- If the property value is greater than zero:

  The barcode symbol will be reduced/stretched to fit this width value. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

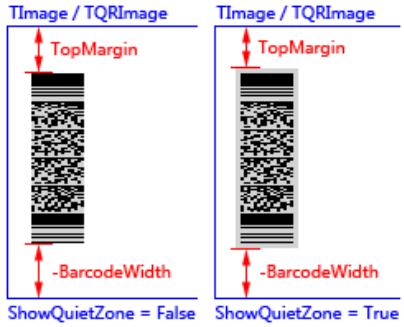


- If the property value is less than zero:

  When the Orientation property is set to "boLeftRight" or "boRightLeft", the LeftMargin property value and the absolute value of the negative width will be subtracted from the width of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode width, the barcode symbol will be reduced/stretched to fit this final width value (it specifies the right margin of the barcode symbol, -1 denotes the right margin is 1, -2 denotes the right margin is 2, ...). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

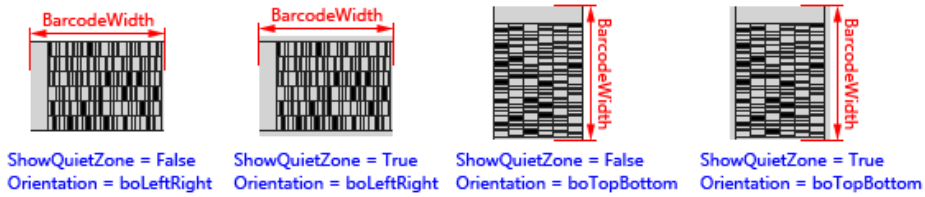
  When the Orientation property is set to "boTopBottom" or "boBottomTop", the TopMargin property value and the

absolute value of the negative width will be subtracted from the height of the TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be used as the final barcode width, the barcode symbol will be reduced/stretched to fit this final width value (it specifies the bottom margin of the barcode symbol, -1 denotes the bottom margin is 1, -2 denotes the bottom margin is 2, ...). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

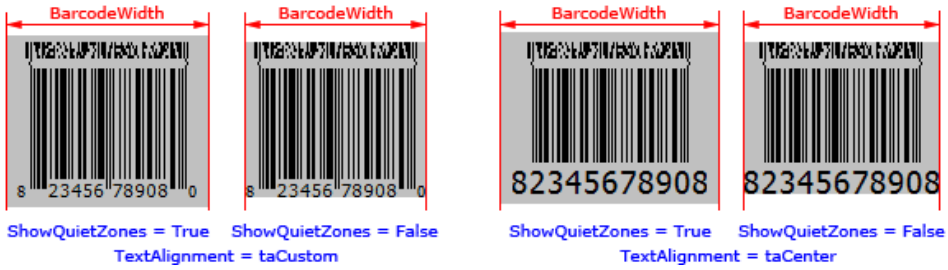


Note:

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, when the linear component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the value of BarcodeWidth property. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



A.1.9 BarColor

Specifies the color for all bars or dark modules in the barcode symbol, By default, it's clBlack.

Syntax:

```
property BarColor: TColor;
```

Description:

For the **Matrix** 2D barcode symbologies, including TBarcode2D_AztecCode, TBarcode2D_AztecRunes, TBarcode2D_DataMatrix, TBarcode2D_DataMatrixECC200, TBarcode2D_HanXinCode, TBarcode2D_GridMatrix, TBarcode2D_CompactMatrix, TBarcode2D_QRCode, TBarcode2D_MaxiCode, and TBarcode2D_MicroQRCode, the property specifies the color of every dark module in matrix symbol if the Inversed property is set to false, otherwise, it specifies the color of every light module. The module is single cell in the matrix symbol used to encode one bit data, nominally a suqare shape, in MaxiCode symbology, it's a regular hexagonal shape.

For **Stacked** 2D barcode symbologies and **Linear** 1D barcode symbologies, including TBarcode2D_Code16K, TBarcode2D_PDF417, TBarcode2D_MicroPDF417, TBarcode2D_CCA, TBarcode2D_CCB, TBarcode2D_CCC, TBarcode2D_RSS14, TBarcode2D_RSSLimited, and TBarcode2D_RSSExpanded, the property specifies the color for all bars in the barcode symbol if the Inversed property is set to false, otherwise, it specifies the color for all spaces.

For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the property is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

Also, when the leading quiet zone, trailing quiet zone, top quiet zone, and bottom quiet zone are represented (the ShowQuietZone property value is set to true), if the Inversed property is set to true, the color specified by this BarColor property will be used to represent them, otherwise, the color specified by the SpaceColor property will be used.

A.1.10 BottomQuietZone

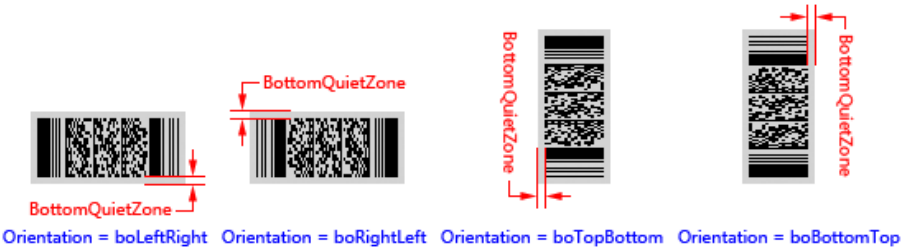
Specifies the vertical height of the bottom quiet zone in modules.

Syntax:

```
property BottomQuietZone: Integer;
```

Description:

Specifies the vertical height of the bottom quiet zone in modules. The quiet zone is represented using the color specified by the SpaceColor property if the Inversed property is set to false. Otherwise, it's represented using the color specified by the BarColor property. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



This property is useful only when the ShowQuietZone property value is set to true.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the bottom quiet zone is close to the bottom boundaries of entire EAN.UCC composite barcode symbol, namely, the bottom boundaries of the linear symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

A.1.11 BytesAlwaysBackToUpper

(TBarcode2D_AztecCode)

The property is an advanced feature which allows low level control over data encoding for an Aztec Code symbol.

Syntax:

```
property BytesAlwaysBackToUpper: Boolean;
```

Description:

In the Aztec Code symbology specification, at the end of bytes mode, the encoding mode returns to the mode from which bytes mode was shifted into. For some readers, the encoding mode always returns to the upper mode, change the property to true in order to ensure barcode symbols can be read by these readers.

Please use your reader to read the following symbol, if the output text is "~~~a9a9a9TTT", please set the property to true, if the output text is "~~~a9a9a9\\\", please set the property to false.

A.1.12 Compact

(TBarcode2D_PDF417, TBarcode2D_CCC)

Specifies whether the PDF417 barcode symbol is generated in Compact PDF417 format, or the CC-C 2D component of an EAN.UCC composite barcode symbol is generated in Compact format.

Syntax:

```
property Compact: Boolean;
```

Description:

In an environment where space considerations are a primary concern and symbol damage is unlikely (e.g. an office), the right row indicators of a PDF417 symbol may be omitted and the stop pattern may be reduced to one module width bar. This reduction version is called Compact PDF417, which is fully decoder compatible with standard PDF417. The property specifiies whether the PDF417 barcode symbol is generated in Compact PDF417 format. See diagram:

For the CC-C 2D component of an EAN.UCC composite barcode symbol, it's a variant of PDF417 symbol and similar with the PDF417 barcode symbol, the property specifies whether its right row indicators should be omitted and its stop pattern should be reduced to one module width bar. See diagram:

Note, the Compact PDF417 was referred as Truncated PDF417 in previous standard.

See also the "Compact PDF417" section in the "TBarcode2D_PDF417" article and "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.13 CurrentColumns

(TBarcode2D_CCC)

Contains the factual number of columns of the CC-C 2D component in an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Columns = 1..30;
property CurrentColumns: TPDF417_Columns;
```

Description:

For the CC-C 2D component in an ENA.UCC composite barcode symbol, the minimum number of columns (specified by the MinColumns property) and the minimum number of stacked rows (specified by the MinRows property) indicate the minimum symbol size for the CC-C 2D component of the EAN.UCC composite barcode symbol. The maximum number of columns (specified by the MaxColumns property) and the maximum number of stacked rows (specified by the MaxRows property) indicate the maximum symbol size for it. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size. Read the property to retrieve the factual number of columns of the CC-C 2D component.

The property can be one of values from 1 to 30, denotations the factual number of columns of a CC-C 2D barcode symbol. They are defined in the pPDF417Custom unit.

The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology. It's a variant of PDF417 symbol which begins with PDF417 codeword 920.

The property is read only.

See also the "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.14 CurrentColumns

(TBarcode2D_PDF417)

Contains the factual number of columns of a PDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Columns = 1..30;
property CurrentColumns: TPDF417_Columns;
```

Description:

The minimum number of columns (specified by the MinColumns property) and the minimum number of rows (specified by the MinRows property) indicate the minimum symbol size for a PDF417 barcode symbol. The maximum number of columns (specified by the MaxColumns property) and the maximum number of rows (specified by the MaxRows property) indicate the maximum symbol size for the PDF417 barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size. Read the property to retrieve the factual number of columns of the PDF417 barcode symbol.

The property can be one of values from 1 to 30, denotations the factual number of columns of a PDF417 2D barcode symbol. They are defined in the pPDF417Custom unit.

The property is read only.

See also the "Symbol size" section in the "TBarcode2D_PDF417" article.

A.1.15 CurrentECCLevel

(TBarcode2D_CompactMatrix)

Contains the factual error correction code level of a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_EccLevel = 1..8;
property  CurrentECCLevel: TCompactMatrix_EccLevel;
```

Description:

The property always contains the factual error correction code level of a Compact Matrix barcode symbol. It can be one of values from 1 to 8, corresponding to the ECC levels from 1 to 8.

If the ECCLevelUpgrade property is set to true, the remaining data capacity of a Compact Matrix barcode symbol will be used to automatically upgrade the error correction code level, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Read the property to retrieve the factual error correction code level of the Compact Matrix barcode symbol.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of this property is always equal to the value of the ECCLevel property.

The property is read only.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_CompactMatrix" article.

A.1.16 CurrentECCLevel

(TBarcode2D_DataMatrix)

Contains the factual error correction code level of a Data Matrix (ECC 000 - 140) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrix unit }
  TDataMatrix_ECCLevel = (dmECC000, dmECC050, dmECC080, dmECC100, dmECC140);
property CurrentECCLevel: TDataMatrix_ECCLevel;
```

Description:

The property always contains the factual error correction code level of a Data Matrix (ECC 000 - 140) barcode symbol. It can be one of five values dmECC000, dmECC050, dmECC080, dmECC100, and dmECC140, corresponding to error correction code level ECC 000, ECC 050, ECC 080, ECC 100, and ECC 140. They are defined in the pDataMatrix unit.

If the ECCLevelUpgrade property is set to true, the remaining data capacity of a Data Matrix (ECC 000 - 140)barcode symbol will be used to automatically upgrade the error correction code level, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Read the property to retrieve the factual error correction code level of the Data Matrix (ECC 000 - 140) barcode symbol.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of this property is always equal to the value of the ECCLevel property.

The property is read only.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_DataMatrix" article.

A.1.17 CurrentECCLevel

(TBarcode2D_GridMatrix)

Contains the factual error correction code level of a Grid Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pGridMatrix unit }
  TGridMatrix_EccLevel = (elLevel_Recommend, elLevel_1, elLevel_2, elLevel_3, elLevel_4,
    elLevel_5, elLevel_LowestRecommend);
property CurrentECCLevel: TGridMatrix_EccLevel;
```

Description:

The Grid Matrix symbology offers five levels of error correction, from 1 to 5 respectively in increasing order of recovery capacity.

The property always contains the factual error correction code level of a Grid Matrix barcode symbol. It can be one of these values (defined in the pGridMatrix unit):

- **elLevel_1**: The ECC level of a Grid Matrix barcode symbol is 1. The percentage of total capacity for ECC data is 10%.
- **elLevel_2**: The ECC level of a Grid Matrix barcode symbol is 2. The percentage of total capacity for ECC data is 20%.
- **elLevel_3**: The ECC level of a Grid Matrix barcode symbol is 3. The percentage of total capacity for ECC data is 30%.

- **elLevel_4**: The ECC level of a Grid Matrix barcode symbol is 4. The percentage of total capacity for ECC data is 40%.
- **elLevel_5**: The ECC level of a Grid Matrix barcode symbol is 5. The percentage of total capacity for ECC data is 50%.

If the ECCLevelUpgrade property is set to true, the remaining data capacity of a Grid Matrix barcode symbol will be used to automatically upgrade the error correction code level, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Read the property to retrieve the factual error correction code level of the Grid Matrix barcode symbol.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of this property is always equal to the value of the ECCLevel property.

The property is read only.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_GridMatrix" article.

A.1.18 CurrentECCLevel

(TBarcode2D_HanXinCode)

Contains the factual error correction code level of a Han Xin Code barcode symbol.

Syntax:

```
type
  { Defined in the pHanXinCode unit }
  THanXinCode_EccLevel = (elLevel_1, elLevel_2, elLevel_3, elLevel_4);
property CurrentECCLevel: THanXinCode_EccLevel;
```

Description:

The Han Xin Code symbology offers four levels of error correction, from L1 to L4 respectively in increasing order of recovery capacity. The property always contains the factual error correction code level of a Han Xin Code barcode symbol. It can be one of these values (defined in the pHanXinCode unit):

- **elLevel_1**: The ECC level of a Han Xin Code barcode symbol is L1. The recevory capacities (approx.) is 8%.
- **elLevel_2**: The ECC level of a Han Xin Code barcode symbol is L2. The recevory capacities (approx.) is 15%.
- **elLevel_3**: The ECC level of a Han Xin Code barcode symbol is L3. The recevory capacities (approx.) is 23%.
- **elLevel_4**: The ECC level of a Han Xin Code barcode symbol is L4. The recevory capacities (approx.) is 30%.

If the ECCLevelUpgrade property is set to true, the remaining data capacity of a Han Xin Code barcode symbol will be used to automatically upgrade the error correction code level, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Read the property to retrieve the factual error correction code level of the Han Xin Code barcode symbol.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of this property is always equal to the value of the ECCLevel property.

The property is read only.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_HanXinCode" article.

A.1.19 CurrentECCLevel

(TBarcode2D_MicroQRCode)

Contains the factual error correction code level of a Micro QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pMicroQRCode unit }
  TMicroQRCode_ECCLevel = (elLowest, elMedium, elQuality);
property CurrentECCLevel: TMicroQRCode_ECCLevel;
```

Description:

The property always contains the factual error correction code level of a Micro QR Code barcode symbol. It can be one of these values (defined in the pMicroQRCode unit):

- **elLowest**: The ECC level of a Micro QR Code barcode symbol is L. The capability of recovery from the amounts of damage is 7%.
- **elMedium**: The ECC level of a Micro QR Code barcode symbol is M. The capability of recovery from the amounts of damage is 15%.
- **elQuality**: The ECC level of a Micro QR Code barcode symbol is Q. The capability of recovery from the amounts of damage is 25%.

If the ECCLevelUpgrade property is set to true, the remaining data capacity of a Micro QR Code barcode symbol will be used to automatically upgrade the error correction code level, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Read the property to retrieve the factual error correction code level of the Micro QR Code barcode symbol.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of this property is always equal to the value of the ECCLevel property.

The property is read only.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_MicroQRCode" article.

A.1.20 CurrentECCLevel

(TBarcode2D_PDF417, TBarcode2D_CCC)

Contains the factual error correction code level of a PDF417 barcode symbol, or of the CC-C 2D component in an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_EccLevel = (elEcc_0, elEcc_1, elEcc_2, elEcc_3, elEcc_4, elEcc_5, elEcc_6,
    elEcc_7, elEcc_8, elEcc_Auto);
property CurrentECCLevel: TPDF417_EccLevel;
```

Description:

The property always contains the factual error correction code level of a PDF417 barcode symbol, or of the CC-C 2D component in

an EAN.UCC composite barcode symbol. It can be one of values from elEcc_0 to elEcc_8, corresponding to error correction code level from ECC 0 to ECC 8. They are defined in the pPDF417Custom unit. The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

If the ECCLevelUpgrade property is set to true, the remaining data capacity of a PDF417 barcode symbol, or of the CC-C 2D component in an EAN.UCC composite barcode symbol, will be used to automatically upgrade the error correction code level. The highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Read the property to retrieve the factual error correction code level.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. If the ECCLevel property is set to one of values from elECC_0 to elECC_8, the value of this property is always equal to the value of the ECCLevel property. If the ECCLevel property is set to elEcc_Auto, the value of this property is the recommended minimum ECC level, depending on the length of barcode text.

The property is read only.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_PDF417" and "TBarcode2D_CCC" articles.

A.1.21 CurrentECCLevel

(TBarcode2D_QRCode)

Contains the factual error correction code level of a QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pQRCode unit }
  TQRCode_ECCLevel = (elLowest, elMedium, elQuality, elHighest);
property CurrentECCLevel: TQRCode_ECCLevel;
```

Description:

The property always contains the factual error correction code level of a QR Code barcode symbol. It can be one of these values (defined in the pQRCode unit):

- **elLowest**: The ECC level of a QR Code barcode symbol is L. The capability of recovery from the amounts of damage is 7%.
- **elMedium**: The ECC level of a QR Code barcode symbol is M. The capability of recovery from the amounts of damage is 15%.
- **elQuality**: The ECC level of a QR Code barcode symbol is Q. The capability of recovery from the amounts of damage is 25%.
- **elHighest**: The ECC level of a QR Code barcode symbol is H. The capability of recovery from the amounts of damage is 30%.

If the ECCLevelUpgrade property is set to true, the remaining data capacity of a QR Code barcode symbol will be used to automatically upgrade the error correction code level, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Read the property to retrieve the factual error correction code level of the QR Code barcode symbol.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of this property is always equal to the value of the ECCLevel property.

The property is read only.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_QRCode" article.

A.1.22 CurrentEncodeMode

(TBarcode2D_DataMatrix)

Contains the factual data encoding mode of a Data Matrix (ECC 000 - 140) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrix unit }
  TDataMatrix_EncodeMode = (emAuto, emNumeric, emAlpha, emPunctuation, emAlphanumeric,
    emASCII, emBinary);
property CurrentEncodeMode: TDataMatrix_EncodeMode;
```

Description:

Read the property to retrieve the factual data encoding mode of a Data Matrix (ECC 000 - 140)barcode symbol. It can be one of these values: emNumeric, emAlpha, emPunctuation, emAlphanumeric, emASCII, and emBinary, corresponding to the Numeric (Base 11), Alpha (Base 27), Punctuation (Base 37), Alphanumeric (Base 41), ASCII, and Binary encoding modes. They are defined in the pDataMatrix unit.

If the EncodeMode property is set to emAuto, the data encoding mode will be selected automatically, depending on the barcode text in order to minimize the symbol size. It can be one of these value: emNumeric, emAlpha, emPunctuation, emAlphanumeric, emASCII, and emBinary. You can read this property to get the factual data encoding mode.

If the EncodeMode property isn't set to emAuto, the value of this property is equal to the value of EncodeMode property.

The property is read only.

See also the "Encoding modes" section in the "TBarcode2D_DataMatrix" article.

A.1.23 CurrentMode

(TBarcode2D_Code16K)

Contains the factual initial mode of a Code 16K barcode symbol.

Syntax:

```
type
  { Defined in the pCode16K unit }
  TCode16K_EncodeMode = (emAuto, emModeA, emModeB, emModeC, emModeB_FNC1, emModeC_FNC1,
    emModeC_Shift1B, emModeC_Shift2B, emMode_Extended);
property CurrentMode: TCode16K_EncodeMode;
```

Description:

Read the property to retrieve the factual initial mode of a Code 16K barcode symbol. It can be one of these values: emModeA, emModeB, emModeC, emModeB_FNC1, emModeC_FNC1, emModeC_Shift1B, emModeC_Shift2B, and emMode_Extended, corresponding to the initial modes 0 to 6, and extended data length mode. They are defined in the pCode16K unit.

If the InitialMode property is set to emAuto, the initial mode will be selected automatically, depending on the barcode text in order to

minimize the symbol size. It can be one of these value: emModeA, emModeB, emModeC, emModeB_FNC1, emModeC_FNC1, emModeC_Shift1B, or emModeC_Shift2B. You can read this property to get the factual initial mode.

If the InitialMode property isn't set to emAuto, the value of this property is equal to the value of InitialMode property.

The property is read only.

See also the "Initial modes" section in the "TBarcode2D_Code16K" article.

A.1.24 CurrentMode

(TBarcode2D_MaxiCode)

Contains the factual mode of a MaxiCode barcode symbol.

Syntax:

```
type
  { Defined in the pMaxiCode unit }
  TMaxiCode_Mode = 2..6;
property CurrentMode: TMaxiCode_Mode;
```

Description:

Read the property to retrieve the factual mode of a MaxiCode barcode symbol. It can be one of values from 2 to 6, denotations the factual mode of a MaxiCode 2D barcode symbol. They are defined in the pMaxiCode unit.

If the AutoMode property is set to Ture, the mode of MaxiCode symbol will be selected automatically, depending on the barcode text and the value of Mode property. You can read this property to get the factual mode.

If the AutoMode property is set to false, the value of this property is equal to the value of Mode property.

The property is read only.

See also the "Modes" section in the "TBarcode2D_MaxiCode" article.

A.1.25 CurrentRows

(TBarcode2D_CCA)

Contains the factual number of stacked rows of the CC-A 2D component in an ENA.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pCCA unit }
  TCCA_Rows = (ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9,
    ccaRow_10, ccaRow_12);
property CurrentRows: TCCA_Rows;
```

Description:

For the CC-A 2D component in an ENA.UCC composite barcode symbol, the smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by the MinRows property)

and the maximum number of stacked rows (specified by the MaxRows property). Read the property to retrieve the factual number of stacked rows of the CC-A 2D component in the ENA.UCC composite barcode symbol. For example, the value ccaRow_6 denotations the factual number of rows is 6 stacked rows.

The property can be one of these values: ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9, ccaRow_10, and ccaRow_12, corresponding to the number of stacked rows 3, 4, 5, 6, 7, 8, 9, 10, and 12. They are defined in the pCCA unit.

The CC-A is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_CCA" article.

A.1.26 CurrentRows

(TBarcode2D_CCB)

Contains the factual number of stacked rows of the CC-B 2D component in an ENA.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pCCB unit }
  TCCB_Rows = (ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_11, ccbRow_12, ccbRow_14,
    ccbRow_15, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26, ccbRow_32, ccbRow_38,
    ccbRow_44);
property CurrentRows: TCCA_Rows;
```

Description:

For the CC-B 2D component in an ENA.UCC composite barcode symbol, the smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by the MinRows property) and the maximum number of stacked rows (specified by the MaxRows property). Read the property to retrieve the factual number of stacked rows of the CC-B 2D component in the ENA.UCC composite barcode symbol. For example, the value ccbRow_6 denotations the factual number of rows is 6 stacked rows.

The property can be one of these values: ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_11, ccbRow_12, ccbRow_14, ccbRow_15, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26, ccbRow_32, ccbRow_38, and ccbRow_44, corresponding to the number of stacked rows 4, 6, 8, 10, 11, 12, 14, 15, 17, 20, 23, 26, 32, 38 and 44. They are defined in the pCCB unit.

The CC-B is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_CCB" article.

A.1.27 CurrentRows

(TBarcode2D_CCC)

Contains the factual number of stacked rows of the CC-C 2D component in an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Rows = 3..90;
property CurrentRows: TPDF417_Rows;
```

Description:

For the CC-C 2D component in an ENA.UCC composite barcode symbol, the minimum number of stacked rows (specified by the MinRows property) and the minimum number of columns (specified by the MinColumns property) indicate the minimum symbol size for the CC-C 2D component of an EAN.UCC composite barcode symbol. The maximum number of stacked rows (specified by the MaxRows property) and the maximum number of columns (specified by the MaxColumns property) indicate the maximum symbol size for it. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size. Read the property to retrieve the factual number of stacked rows of the CC-C 2D component.

The property can be one of values from 3 to 90, denotations the factual number of rows. They are defined in the pPDF417Custom unit.

The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology. It's a variant of PDF417 symbol which begins with PDF417 codeword 920.

The property is read only.

See also the "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.28 CurrentRows

(TBarcode2D_Code16K)

Contains the factual number of stacked rows of a Code 16K barcode symbol.

Syntax:

```
type
  { Defined in the pCode16K unit }
  TCode16K_Rows = 2..16;
property CurrentRows: TCode16K_Rows;
```

Description:

The smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by the MinRows property) and the maximum number of stacked rows (specified by the MaxRows property). Read the property to retrieve the factual number of stacked rows of a Code 16K barcode symbol.

The property can be one of values from 2 to 16, denotations the factual number of rows. They are defined in the pCode16K unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_Code16K" article.

A.1.29 CurrentRows

(TBarcode2D_PDF417)

Contains the factual number of stacked rows of a PDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Rows = 3..90;
property CurrentRows: TPDF417_Rows;
```

Description:

The minimum number of stacked rows (specified by the MinRows property) and the minimum number of columns (specified by the MinColumns property) indicate the minimum symbol size for a PDF417 barcode symbol. The maximum number of stacked rows (specified by the MaxRows property) and the maximum number of columns (specified by the MaxColumns property) indicate the maximum symbol size for the PDF417 barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size. Read the property to retrieve the factual number of stacked rows of the PDF417 barcode symbol.

The property can be one of values from 3 to 90, denotations the factual number of rows. They are defined in the pPDF417Custom unit.

The property is read only.

See also the "Symbol size" section in the "TBarcode2D_PDF417" article.

A.1.30 CurrentSegments

(TBarcode2D_CompactMatrix)

Contains the factual number of segments of a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_Segments = 1..32;
property CurrentSegments: TCompactMatrix_Segments;
```

Description:

In horizontal orientation, each Compact Matrix symbol consists of an array of segments with a minimum of 1 segment (maximum 32 segments).

The minimum version (specified by the MinVersion property) and the minimum number of segments (specified by the MinSegments property) indicate the minimum symbol size for a Compact Matrix barcode symbol. The maximum version (specified by the MaxVersion property) and the maximum number of segments (specified by the MaxSegments property) indicate the maximum symbol size for the Compact Matrix barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size. Read the property to retrieve the factual number of segments of the Compact Matrix barcode symbol.

The property can be one of values from 1 to 32, denotations the factual number of segments of a Compact Matrix 2D barcode symbol. They are defined in the pCompactMatrix unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_CompactMatrix" article.

A.1.31 CurrentSize

(TBarcode2D_AztecCode)

Contains the factual symbol size of an Aztec Code barcode symbol.

Syntax:

```
type
  { Defined in the pAztecCode unit }
  TAztecCode_Size = (azSize_15Compact, azSize_19, azSize_19Compact, azSize_23,
    azSize_23Compact, azSize_27, azSize_27Compact, azSize_31, azSize_37, azSize_41,
    azSize_45, azSize_49, azSize_53, azSize_57, azSize_61, azSize_67, azSize_71,
    azSize_75, azSize_79, azSize_83, azSize_87, azSize_91, azSize_95, azSize_101,
    azSize_105, azSize_109, azSize_113, azSize_117, azSize_121, azSize_125, azSize_131,
    azSize_135, azSize_139, azSize_143, azSize_147, azSize_151);
property CurrentSize: TAztecCode_Size;
```

Description:

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by the MaxSize property. Read the property to retrieve the factual symbol size of an Aztec Code barcode symbol.

The property can be one of values from azSize_15Compact to azSize_151, denotations the factual format, number of rows and columns in current symbol. For example, the symbol size value azSize_19Compact denotations the Aztec Code symbol size is 19 rows by 19 columns, in compact format. These symbol size values are defined in the pAztecCode unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_AztecCode" article.

A.1.32 CurrentSize

(TBarcode2D_DataMatrix)

Contains the factual symbol size of a Data Matrix (ECC 000 - 140) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrix unit }
  TDataMatrix_Size = (dmSize_09_09, dmSize_11_11, dmSize_13_13, dmSize_15_15,
    dmSize_17_17, dmSize_19_19, dmSize_21_21, dmSize_23_23, dmSize_25_25, dmSize_27_27,
    dmSize_29_29, dmSize_31_31, dmSize_33_33, dmSize_35_35, dmSize_37_37, dmSize_39_39,
    dmSize_41_41, dmSize_43_43, dmSize_45_45, dmSize_47_47, dmSize_49_49);
property CurrentSize: TDataMatrix_Size;
```

Description:

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by the MaxSize property. Read the property to retrieve the factual symbol size of a Data Matrix (ECC 000 - 140) barcode symbol.

The property can be one of values from dmSize_09_09 to dmSize_49_49, denotations the factual number of rows and columns in current symbol. For example, the symbol size value dmSize_29_29 denotations the Data Matrix (ECC 000 - 140)symbol size is 29 rows by 29 columns. These symbol size values are defined in the pDataMatrix unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_DataMatrix" article.

A.1.33 CurrentSize

(TBarcode2D_DataMatrixECC200)

Contains the factual symbol size of a Data Matrix (ECC 200) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrixEcc200 unit }
  TDataMatrixEcc200_Size = (dmSize_10_10, dmSize_12_12, dmSize_14_14, dmSize_16_16,
    dmSize_18_18, dmSize_20_20, dmSize_22_22, dmSize_24_24, dmSize_26_26, dmSize_32_32,
    dmSize_36_36, dmSize_40_40, dmSize_44_44, dmSize_48_48, dmSize_52_52, dmSize_64_64,
    dmSize_72_72, dmSize_80_80, dmSize_88_88, dmSize_96_96, dmSize_104_104,
    dmSize_120_120, dmSize_132_132, dmSize_144_144, dmSize_8_18, dmSize_8_32,
    dmSize_12_26, dmSize_12_36, dmSize_16_36, dmSize_16_48);
property CurrentSize: TDataMatrixEcc200_Size;
```

Description:

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by the MaxSize property. Read the property to retrieve the factual symbol size of a Data Matrix (ECC 200) barcode symbol.

The property can be one of values from dmSize_10_10 to dmSize_16_48, denotations the factual number of rows and columns in current symbol. For example, the symbol size value dmSize_8_18 denotations the Data Matrix (ECC 200)symbol size is 8 rows by 18 columns. These symbol size values are defined in the pDataMatrixEcc200 unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_DataMatrixECC200" article.

A.1.34 CurrentSize

(TBarcode2D_MicroPDF417)

Contains the factual symbol size of an MicroPDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_Size = (mpSize_1_11, mpSize_1_14, mpSize_1_17, mpSize_1_20, mpSize_1_24,
    mpSize_1_28, mpSize_2_8, mpSize_2_11, mpSize_2_14, mpSize_2_17, mpSize_2_20,
    mpSize_2_23, mpSize_2_26, mpSize_3_6, mpSize_3_8, mpSize_3_10, mpSize_3_12,
    mpSize_3_15, mpSize_3_20, mpSize_3_26, mpSize_3_32, mpSize_3_38, mpSize_3_44,
    mpSize_4_4, mpSize_4_6, mpSize_4_8, mpSize_4_10, mpSize_4_12, mpSize_4_15,
    mpSize_4_20, mpSize_4_26, mpSize_4_32, mpSize_4_38, mpSize_4_44);
property CurrentSize: TMicroPDF417_Size;
```

Description:

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by the MaxSize property. Read the property to retrieve the factual symbol size of an MicroPDF417 barcode symbol.

The property can be one of values from mpSize_1_11 to mpSize_4_44, denotations the factual number of columns and rows in current symbol. For example, the symbol size value mpSize_3_12 denotations the MicroPDF417 symbol size is 12 stacked rows by 3 columns. These symbol size values are defined in the pMicroPDF417 unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_MicroPDF417" article.

A.1.35 CurrentVersion

(TBarcode2D_CompactMatrix)

Contains the factual version value of a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_Version = 1..32;
property CurrentVersion: TCompactMatrix_Version;
```

Description:

There are 32 vertical sizes of Compact Matrix symbol, referred to as version 1 to 32, in increasing order of symbol height and data capacity.

The minimum version (specified by the MinVersion property) and the minimum number of segments (specified by the MinSegments property) indicate the minimum symbol size for a Compact Matrix barcode symbol. The maximum version (specified by the MaxVersion property) and the maximum number of segments (specified by the MaxSegments property) indicate the maximum symbol size for the Compact Matrix barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size. Read the property to retrieve the factual version value of the Compact Matrix barcode symbol.

The property can be one of values from 1 to 32, denotations the factual version value. They are defined in the pCompactMatrix unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_CompactMatrix" article.

A.1.36 CurrentVersion

(TBarcode2D_GridMatrix)

Contains the factual symbol version (symbol size) of a Grid Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pGridMatrix unit }
  TGridMatrix_Version = 1..13;
property CurrentVersion: TGridMatrix_Version;
```

Description:

There are thirteen sizes of Grid Matrix symbol, referred to as version 1 to 13, in increasing order of size and data capacity.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by the MaxVersion property. Read the property to retrieve the factual symbol version (symbol size) of a Grid Matrix barcode symbol.

The property can be one of values from 1 to 13, denotations the factual version of a Grid Matrix 2D barcode symbol. They are defined in the pGridMatrix unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_GridMatrix" article.

A.1.37 CurrentVersion

(TBarcode2D_HanXinCode)

Contains the factual symbol version (symbol size) of a Han Xin Code barcode symbol.

Syntax:

```
type
  { Defined in the pHanXinCode unit }
  THanXinCode_Version = 1..84;
property CurrentVersion: THanXinCode_Version;
```

Description:

There are eighty-four sizes of Han Xin Code symbols, referred to as version 1 to 84, in increasing order of size and data capacity. The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by the MaxVersion property. Read the property to retrieve the factual symbol version (symbol size) of a Han Xin Code barcode symbol.

The property can be one of values from 1 to 84, denotations the factual version of a Han Xin Code 2D barcode symbol. They are defined in the pHanXinCode unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_HanXinCode" article.

A.1.38 CurrentVersion

(TBarcode2D_MicroQRCode)

Contains the factual symbol version (symbol size) of a Micro QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pMicroQRCode unit }
  TMicroQRCode_Version = 1..4;
property CurrentVersion: TMicroQRCode_Version;
```

Description:

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by the MaxVersion property. Read the property to retrieve the factual symbol version (symbol size) of a Micro QR Code barcode symbol.

The property can be one of values from 1 to 4, denotations the factual version of a Micro QR Code2D barcode symbol. They are defined in the pMicroQRCode unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_MicroQRCode" article.

A.1.39 CurrentVersion

(TBarcode2D_QRCode)

Contains the factual symbol version (symbol size) of a QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pQRCode unit }
  TQRCode_Version = 1..40;
property CurrentVersion: TQRCode_Version;
```

Description:

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by the MaxVersion property. Read the property to retrieve the factual symbol version (symbol size) of a QR Code barcode symbol.

The property can be one of values from 1 to 40, denotations the factual version of a QR Code 2D barcode symbol. They are defined in the pQRCode unit.

The property is read only.

See also the "Symbol sizes" section in the "TBarcode2D_QRCode" article.

A.1.40 Data

Specifies a block of binary (bytes) data to encode it into the barcode symbol. Or specifies the barcode text in the AnsiString format.

Syntax:

```
property Data: AnsiString;
```

Description:

If you use the Delphi/C++ Builder 2009 or later, you can use the property to specify a block of binary (bytes) data, and encode it into a barcode symbol. Or specifies the barcode text in AnsiString format. The OnChange event will occur when the property value is changed. The OnInvalidDataChar event will occur if there is any invalid character in the Data property value, and the OnInvalidDataLength event will occur if the length of the Data property value is invalid.

If you want to encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please specify it in the Barcode property.

The Barcode property of TBarcode2D component is of type string, for Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), you can specify the converted string in the Data property. For the Delphi/C++ Builder 2007 or early, the Barcode property is of type AnsiString in ANSI encoding, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please specify the converted string in the Barcode property.

For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be entered in the here, otherwise the check digit can be specified by you in here.

Note, if a 2D component (including the TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC) together with a linear component (including the TBarcode2D_RSS14, TBarcode2D_RSSLimited, TBarcode2D_RSSExpanded, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128) is assembled to generate an EAN.UCC composite barcode symbol, the Data property of the 2D component only specifies the barcode text for the 2D component, and the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component only specifies the barcode text for the linear component.

Note:

The property is available only for the Delphi/C++ Builder 2009 or later.

A.1.41 DefaultEncodeMode

(TBarcode2D_MicroPDF417)

Specifies the initial compaction mode for a MicroPDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_EncodeMode = (emText, emByte, emNumeric, emNull);
property DefaultEncodeMode: TMicroPDF417_EncodeMode;
```

Description:

The property specifies the initial compaction mode for a MicroPDF417 barcode symbol. In the specification of MicroPDF417, the initial compaction mode in effect at the start of each symbol shall always be Byte compaction mode, but some readers use the Text compaction mode (Alpha sub-mode) as the initial compaction mode. The property should be set to match your reader. If you don't known the initial compaction mode used by your reader, please set the property value to emNull, or you can change the Options property to include the "poMicroPDF417Explicit901" value, in order to explicitly insert a mode latch to Byte compaction mode into beginning of symbol data.

The property can be one of these values (defined in the pMicroPDF417 unit).

- **emText**: Indicates the initial compaction mode to Text compaction mode. You can use the DefaultTextEncodeMode property to specify the Text compaction sub-mode.
- **emNumeric**: Indicates the initial compaction mode to Numeric compaction mode.
- **emByte**: Indicates the initial compaction mode to Byte compaction mode.
- **emNull**: The appropriate initial compaction mode will be introduced by an explicit mode latch depending on the barcode text.

See also the "Character set" section in the "TBarcode2D_MicroPDF417" article.

A.1.42 DefaultTextEncodeMode

(TBarcode2D_MicroPDF417)

Specifies the initial Text compaction sub-mode for a MicroPDF417 barcode symbol if its initial compaction mode is set to Text compaction mode.

Syntax:

```
type
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_TextEncodeMode = (tmAlpha, tmLower, tmMixed, tmPunct);
property DefaultTextEncodeMode: TMicroPDF417_TextEncodeMode;
```

Description:

In the specification of MicroPDF417, the initial compaction mode in effect at the start of each symbol shall always be Byte compaction mode, but some readers use the Text compaction mode (Alpha sub-mode) as the initial compaction mode. The property specifies the initial Text compaction sub-mode for a MicroPDF417 barcode symbol if its initial compaction mode is set to Text compaction mode (the DefaultEncodeMode property is set to emText). It should be set to match your reader.

The property can be one of these values (defined in the pMicroPDF417 unit).

- **tmAlpha**: Indicates the initial Text compaction sub-mode to Alpha sub-mode.
- **tmLower**: Indicates the initial Text compaction sub-mode to Lower sub-mode.
- **tmMixed**: Indicates the initial Text compaction sub-mode to Mixed sub-mode.
- **tmPunct**: Indicates the initial Text compaction sub-mode to Punctuation sub-mode.

Note, the property is available only when the DefaultEncodeMode property is set to emText.

See also the "Character set" section in the "TBarcode2D_MicroPDF417" and the "DefaultEncodeMode" articles.

A.1.43 ECCCount

(TBarcode2D_AztecCode)

Specifies the additional error correction size (number of codewords) for an Aztec Code symbol.

Syntax:

```
property ECCCount: Word;
```

Description:

The property specifies the additional error correction size (number of codewords) for an Aztec Code symbol.

If the ECCLevel property is set to 0, the property specify the minimum error correction size (number of codewords) for an Aztec Code symbol. Depending on message length, the symbology will typically have to add extra error correction codewords above this minimum to fill out the symbol.

If the ECCLevel property isn't set to 0, the property determines the error correction level together with the ECCLevel property. For example, if the ECCLevel property is set to 23 and the ECCCount property is set to 3, a symbol (azSize_37) which holds 120 codewords should include at least 31 (120 * 23% + 3) error correction codewords, leaving up 89 codewords for encoding the data message. Depending on message length, the symbology will typically have to add extra error correction codewords above this minimum to fill out the symbol.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_AztecCode" article.

A.1.44 ECCLevel

(TBarcode2D_AztecCode)

Specifies the alternate minimum error correction percentage for an Aztec Code symbol.

Syntax:

```
type
  { Defined in the pAztecCode unit }
  TAztecCode_EccLevel = 0..99;
property ECCLevel: TAztecCode_EccLevel;
```

Description:

The property specifies the alternate minimum error correction percentage for an Aztec Code symbol (defined in the pAztecCode unit), ranging from 0% to 99% plus always, for data security, 3 or more additional error correction codewords (specified by the ECCCount property). This is called a "minimum" percentage because, depending on message length, the symbology will typically have to add extra error correction codewords above this minimum to fill out the symbol.

It determines the error correction level together with the ECCCount property. For example, if the ECCLevel property is set to 23 and the ECCCount property is set to 3, a symbol (azSize_37) which holds 120 codewords should include at least 31 (120 * 23% + 3) error correction codewords, leaving up 89 codewords for encoding the data message. Depending on message length, the symbology will typically have to add extra error correction codewords above this minimum to fill out the symbol.

The property can be one of values from 0 to 99, corresponding to the ECC level from 0% to 99%. They are defined in the pAztecCode unit.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_AztecCode" article.

A.1.45 ECCLevel

(TBarcode2D_CompactMatrix)

Specifies which ECC level (error correction code level) will be used to increase strength of a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_EccLevel = 1..8;
property ECCLevel: TCompactMatrix_EccLevel;
```

Description:

The Compact Matrix symbology offers eight levels of error correction, from 1 to 8 respectively in increasing order of recovery capacity.

The property specifies which ECC level (error correction code level) will be used to increase strength of a Compact Matrix barcode symbol. It can be one of values from 1 to 8, corresponding to the ECC levels from 1 to 8.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by this property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by this property. In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level.

The CurrentECCLevel property can always be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_CompactMatrix" article.

A.1.46 ECCLevel

(TBarcode2D_DataMatrix)

Specifies which ECC level (error correction code level) will be used to increase strength of a Data Matrix (ECC 000 - 140) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrix unit }
  TDataMatrix_ECCLevel = (dmECC000, dmECC050, dmECC080, dmECC100, dmECC140);
property ECCLevel: TDataMatrix_ECCLevel;
```

Description:

The Data Matrix (ECC 000 - 140) symbology offers five levels of error correction, referred to as ECC 000, ECC 050, ECC 080, ECC 100 and ECC 140 respectively in increasing order of recovery capacity. In an application, it is important to understand that these error correction levels result in the generation of a proportional increase in the number of bits in the message (and hence increase in the size of the symbol), and offer different levels of error recovery.

The property specifies which ECC level (error correction code level) will be used to increase strength of a Data Matrix (ECC 000 - 140) barcode symbol. It can be one of these values (defined in the pDataMatrix unit):

- **dmECC000**: Indicates to use the ECC 000, namely, no error correction.
- **dmECC050**: Indicates to use the ECC 050, 25% data capacity is used for error correction, up to 2.8% damage can be corrected.
- **dmECC080**: Indicates to use the ECC 080, 33.3% data capacity is used for error correction, up to 5.5% damage can be corrected.
- **dmECC100**: Indicates to use the ECC 100, 50% data capacity is used for error correction, up to 12.6% damage can be corrected.
- **dmECC140**: Indicates to use the ECC 140, 75% data capacity is used for error correction, up to 25% damage can be corrected.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by this property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by this property. In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level.

The CurrentECCLevel property can always be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_DataMatrix" article.

A.1.47 ECCLevel

(TBarcode2D_GridMatrix)

Specifies which ECC level (error correction code level) will be used to increase strength of a Grid Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pGridMatrix unit }
  TGridMatrix_EccLevel = (elLevel_Recommend, elLevel_1, elLevel_2, elLevel_3, elLevel_4,
    elLevel_5, elLevel_LowestRecommend);
property ECCLevel: TGridMatrix_EccLevel;
```

Description:

The Grid Matrix symbology offers five levels of error correction, from 1 to 5 respectively in increasing order of recovery capacity.

The property specifies which ECC level (error correction code level) will be used to increase strength of aGrid Matrix barcode symbol. It can be one of these values (defined in the pGridMatrix unit):

- **elLevel_1**: Indicates to use the ECC level 1. The percentage of total capacity for ECC data is 10%. Note, it is inapplicable to the version 1.
- **elLevel_2**: Indicates to use the ECC level 2. The percentage of total capacity for ECC data is 20%.
- **elLevel_3**: Indicates to use the ECC level 3. The percentage of total capacity for ECC data is 30%.
- **elLevel_4**: Indicates to use the ECC level 4. The percentage of total capacity for ECC data is 40%.
- **elLevel_5**: Indicates to use the ECC level 5. The percentage of total capacity for ECC data is 50%.
- **elLevel_Recommend**: For each symbol version, the denotative ECC levels are listed in the following table:

Version	Denotative ECC level
1	ECC level 5
2, 3	ECC level 4
4 - 13	ECC level 3

- **elLevel_LowestRecommend**: For each symbol version, the denotative ECC levels are listed in the following table:

Version	Denotative ECC level
1	ECC level 4
2	ECC level 2
3 - 13	ECC level 1

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by this property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by this property. In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level.

The CurrentECCLevel property can always be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_GridMatrix" article.

A.1.48 ECCLevel

(TBarcode2D_HanXinCode)

Specifies which ECC level (error correction code level) will be used to increase strength of a Han Xin Code barcode symbol.

Syntax:

```
type
  { Defined in the pHanXinCode unit }
  THanXinCode_EccLevel = (elLevel_1, elLevel_2, elLevel_3, elLevel_4);
property ECCLevel: THanXinCode_EccLevel;
```

Description:

The Han Xin Code symbology offers four levels of error correction, from L1 to L4 respectively in increasing order of recovery capacity.

The property specifies which ECC level (error correction code level) will be used to increase strength of aHan Xin Code barcode symbol. It can be one of these values (defined in the pHanXinCode unit):

- **elLevel_1**: Indicates to use the ECC level L1. The recevory capacities (approx.) is 8%.
- **elLevel_2**: Indicates to use the ECC level L2. The recevory capacities (approx.) is 15%.
- **elLevel_3**: Indicates to use the ECC level L3. The recevory capacities (approx.) is 23%.
- **elLevel_4**: Indicates to use the ECC level L4. The recevory capacities (approx.) is 30%.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by this property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction

code level specified by this property. In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level.

The CurrentECCLevel property can always be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_HanXinCode" article.

A.1.49 ECCLevel

(TBarcode2D_MicroQRCode)

Specifies which ECC level (error correction code level) will be used to increase strength of a Micro QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pMicroQRCode unit }
  TMicroQRCode_EccLevel = (elLowest, elMedium, elQuality);
property ECCLevel: TMicroQRCode_EccLevel;
```

Description:

The Micro QR Codesymbology offers three levels of error correction, referred to as L, M, and Q respectively in increasing order of recovery capacity.

The property specifies which ECC level (error correction code level) will be used to increase strength of aMicro QR Codebarcode symbol. It can be one of these values (defined in the pMicroQRCode unit):

- **elLowest**: Indicates to use the ECC level L, up to 7% (approx.) damage can be corrected.
- **elMedium**: Indicates to use the ECC level M, up to 15% (approx.) damage can be corrected.
- **elQuality**: Indicates to use the ECC level Q, up to 25% (approx.) damage can be corrected.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by this property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by this property. In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level.

The CurrentECCLevel property can always be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_MicroQRCode" article.

A.1.50 ECCLevel

(TBarcode2D_PDF417, TBarcode2D_CCC)

Specifies which ECC level (error correction code level) will be used to increase strength of aPDF417 barcode symbol, or of the CC-C 2D component in an EAN.UCC composite barcode symbol.

Syntax:

```
type
```

```
  { Defined in the pPDF417Custom unit }
  TPDF417_EccLevel = (elEcc_0, elEcc_1, elEcc_2, elEcc_3, elEcc_4, elEcc_5, elEcc_6,
    elEcc_7, elEcc_8, elEcc_Auto);
property ECCLevel: TPDF417_EccLevel;
```

Description:

The PDF417 symbology and the CC-C symbology (the CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology, and it's a variant of PDF417) offer nine levels of error correction, referred to as ECC 0 to ECC 8 respectively in increasing order of recovery capacity.

The property specifies which ECC level (error correction code level) will be used to increase strength of a PDF417 barcode symbol, or of the CC-C 2D component in an EAN.UCC composite barcode symbol. It can be one of values from elECC_0 to elECC_8, corresponding to the ECC level from 0 to 8. Also, it can be set to elEcc_Auto, indicates the recommended minimum ECC level will be used, depending on the length of barcode text. These ECC levels are defined in the pPDF417Custom unit, they are listed in following table:

Property value	ECC Level	Number of Error correction codewrods
elECC_0	ECC 0	2
elECC_1	ECC 1	4
elECC_2	ECC 2	8
elECC_3	ECC 3	16
elECC_4	ECC 4	32
elECC_5	ECC 5	64
elECC_6	ECC 6	128
elECC_7	ECC 7	256
elECC_8	ECC 8	512

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by this property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by this property. In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level.

The CurrentECCLevel property can always be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_PDF417" and the "TBarcode2D_CCC" articles.

A.1.51 ECCLevel

(TBarcode2D_QRCode)

Specifies which ECC level (error correction code level) will be used to increase strength of a QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pQRCode unit }
  TQRCode_ECCLevel = (elLowest, elMedium, elQuality, elHighest);
property ECCLevel: TQRCode_ECCLevel;
```

Description:

The QR Code symbology offers four levels of error correction, referred to as L, M, Q and H respectively in increasing order of recovery capacity.

The property specifies which ECC level (error correction code level) will be used to increase strength of aQR Code barcode symbol. It can be one of these values (defined in the pQRCode unit):

- **elLowest**: Indicates to use the ECC level L, up to 7% (approx.) damage can be corrected.
- **elMedium**: Indicates to use the ECC level M, up to 15% (approx.) damage can be corrected.
- **elQuality**: Indicates to use the ECC level Q, up to 25% (approx.) damage can be corrected.
- **elHighest**: Indicates to use the ECC level H, up to 30% (approx.) damage can be corrected.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by this property, and the symbol size will not be increased, it may be determined depending on the length of barcode text, and the error correction code level specified by this property. In other words, only the remaining capacity in current symbol size will be used to upgrade the error correction code level.

The CurrentECCLevel property can always be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_QRCode" article.

A.1.52 ECCLevelUpgrade

(TBarcode2D_CompactMatrix)

For the Compact Matrix barcode symbols, the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level in order to increase symbols strength.

Syntax:

```
property ECCLevelUpgrade: Boolean;
```

Description:

For a Compact Matrix barcode symbol, the symbol size may be determined depending on the barcode text length and the error correction code level specified by the ECCLevel property. And the barcode text and the error correction code don't usually fill the maximum data capacity of the symbol. the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level, in order to increase symbols strength.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased. The CurrentECCLevel property can be used to get the factual error correction code level.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by theECCLevel property will be used always. The value of CurrentECCLevel property is always equal to the value of ECCLevel property.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_CompactMatrix" article.

A.1.53 ECCLevelUpgrade

(TBarcode2D_DataMatrix)

For the Data Matrix (ECC 000 - 140) barcode symbols, the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level in order to increase symbols strength.

Syntax:

```
property ECCLevelUpgrade: Boolean;
```

Description:

For a Data Matrix (ECC 000 - 140) barcode symbol, the symbol size may be determined depending on the barcode text length and the error correction code level specified by the ECCLevel property. And the barcode text and the error correction code don't usually fill the maximum data capacity of the symbol. the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level, in order to increase symbols strength.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased. The CurrentECCLevel property can be used to get the factual error correction code level.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of CurrentECCLevel property is always equal to the value of ECCLevel property.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_DataMatrix" article.

A.1.54 ECCLevelUpgrade

(TBarcode2D_GridMatrix)

For the Grid Matrix barcode symbols, the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level in order to increase symbols strength.

Syntax:

```
property ECCLevelUpgrade: Boolean;
```

Description:

For a Grid Matrix barcode symbol, the symbol size may be determined depending on the barcode text length and the error correction code level specified by the ECCLevel property. And the barcode text and the error correction code don't usually fill the maximum data capacity of the symbol. the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level, in order to increase symbols strength.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased. The CurrentECCLevel property can be used to get the factual error correction code level.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by the ECCLevel property will be used always. The value of CurrentECCLevel property is always equal to the value of ECCLevel property.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_GridMatrix" article.

A.1.55 ECCLevelUpgrade

(TBarcode2D_HanXinCode)

For the Han Xin Code barcode symbols, the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level in order to increase symbols strength.

Syntax:

```
property ECCLevelUpgrade: Boolean;
```

Description:

For a Han Xin Code barcode symbol, the symbol size may be determined depending on the barcode text length and the error correction code level specified by the ECCLevel property. And the barcode text and the error correction code don't usually fill the maximum data capacity of the symbol. the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level, in order to increase symbols strength.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased. The CurrentECCLevel property can be used to get the factual error correction code level.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by theECCLevel property will be used always. The value of CurrentECCLevel property is always equal to the value of ECCLevel property.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_HanXinCode" article.

A.1.56 ECCLevelUpgrade

(TBarcode2D_MicroQRCode)

For the Micro QR Code barcode symbols, the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level in order to increase symbols strength.

Syntax:

```
property ECCLevelUpgrade: Boolean;
```

Description:

For a Micro QR Code barcode symbol, the symbol size may be determined depending on the barcode text length and the error correction code level specified by the ECCLevel property. And the barcode text and the error correction code don't usually fill the maximum data capacity of the symbol. the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level, in order to increase symbols strength.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased. The CurrentECCLevel property can be used to get the factual error correction code level.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by theECCLevel property will be used always. The value of CurrentECCLevel property is always equal to the value of ECCLevel property.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_MicroQRCode" article.

A.1.57 ECCLevelUpgrade

(TBarcode2D_PDF417, TBarcode2D_CCC)

For a PDF417 barcode symbol, or the CC-C 2D component of an EAN.UCC composite barcode symbol, the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level in order to increase symbols strength.

Syntax:

```
property ECCLevelUpgrade: Boolean;
```

Description:

For a PDF417 barcode symbol, or the CC-C 2D component of an EAN.UCC composite barcode symbol, the symbol size may be determined depending on the barcode text length and the error correction code level specified by the ECCLevel property. And the barcode text and the error correction code don't usually fill the maximum data capacity of the symbol. the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level, in order to increase symbols strength.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by theECCLevel property will be used always. If the ECCLevel property is set to one of values from eIECC_0 to eIECC_8, the factual error correction code level is always equal to the value of the ECCLevel property. If the ECCLevel property is set to eIEcc_Auto, the factual error correction code level is the recommended minimum ECC level, depending on the length of barcode text.

The CurrentECCLevel property can be used to get the factual error correction code level.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_PDF417" and "TBarcode2D_CCC" articles.

A.1.58 ECCLevelUpgrade

(TBarcode2D_QRCode)

For the QR Code barcode symbols, the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level in order to increase symbols strength.

Syntax:

```
property ECCLevelUpgrade: Boolean;
```

Description:

For a QR Code barcode symbol, the symbol size may be determined depending on the barcode text length and the error correction code level specified by the ECCLevel property. And the barcode text and the error correction code don't usually fill the maximum data capacity of the symbol. the property specifies whether to use the remaining capacity to automatically upgrade the error correction code level, in order to increase symbols strength.

If the ECCLevelUpgrade property is set to true, the highest error correction code level that can be accommodated by current symbol size will by used for creating more robust symbols. Note, the new level is always no lower than the level specified by the ECCLevel property, and the symbol size will not be increased. The CurrentECCLevel property can be used to get the factual error correction code level.

If the ECCLevelUpgrade property is set to false, the error correction code level specified by theECCLevel property will be used always. The value of CurrentECCLevel property is always equal to the value of ECCLevel property.

See also the "Error checking and correcting (ECC)" section in the "TBarcode2D_QRCode" article.

A.1.59 EncodeMode

(TBarcode2D_DataMatrix)

Specifies the data encoding mode for a Data Matrix (ECC 000 - 140) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrix unit }
  TDataMatrix_EncodeMode = (emAuto, emNumeric, emAlpha, emPunctuation, emAlphanumeric,
    emASCII, emBinary);
property EncodeMode: TDataMatrix_EncodeMode;
```

Description:

The Data Matrix (ECC 000 - 140) symbology offers six kinds of encoding mode, referred to as Numeric (Base 11), Alpha (Base 27), Alphanumeric (Base 37), Punctuation (Base 41), ASCII, and Binary respectively in decreasing order of encoding density. The property specifies which data encoding mode will be used.

The property can be one of these values (defined in the pDataMatrix unit):

- **emAuto**: The encoding mode will be selected automatically, depending on the barcode text, in other words, the barcode text to be encoded will be analysed, and an appropriates lowest level (highest encoding density) encoding mode will be selected, in order to minimize the symbol size. The property CurrentEncodeMode can be used to get the factual encoding mode.
- **emNumeric**: Indicates to use the Numeric (Base 11) encoding mode. It encodes 10 numeric characters 0 to 9, and the space character. The encoding density is 3.5 bits per data character. The encoding mode has highest encoding density.
- **emAlpha**: Indicates to use the Alpha (Base 27) encoding mode. It encodes 26 upper case letters A to Z, and the space character. The encoding density is 4.8 bits per data character.
- **emAlphanumeric**: Indicates to use the Alphanumeric (Base 37) encoding mode. It encodes 26 upper case letters A to Z, 10 numeric characters 0 to 9, and the space character. The encoding density is 5.25 bits per data character.
- **emPunctuation**: Indicates to use the Punctuation (Base 41) encoding mode. It encodes 26 upper case letters A to Z, 10 numeric characters 0 to 9, and the space, point(.), hyphen(-), comma(,) and solidus(/) characters. The encoding density is 5.5 bits per data character.
- **emASCII**: Indicates to use the ASCII encoding mode. It encodes all 128 ASCII characters from ISO/IEC 646. The encoding density is 7 bits per data character.
- **emBinary**: Indicates to use the Binary encoding mode. It encodes all 256 8-bit bytes. It shall be used for closed applications, where the data interpretation shall be determined by the user. The encoding density is 8 bits per byte. The encoding mode has lowest encoding density.

The CurrentEncodeMode property can always be used to get the factual encoding mode.

See also the "Encoding modes" section in the "TBarcode2D_DataMatrix" article.

A.1.60 EncodePolicy

(TBarcode2D_MicroQRCode)

Indicates how to use the encoding mode in a Micro QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pQRCodeCom unit }
  TQRCode_EncodePolicy = (epMixingOptimal, epMixingQuick, epSingleMode);
property EncodePolicy: TQRCode_EncodePolicy;
```

Description:

The Micro QR Code symbology offers four kinds of encoding mode, referred to as Numeric mode, Alphanumeric mode, Kanji mode and Byte mode respectively in decreasing order of encoding density.

This property indicates how to use these encoding mode by the component. It can be one of these values (defined in the pQRCodeCom unit):

- **epMixingOptimal**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The optimal combination of encoding modes will be used in order to minimize the symbol size.
- **epMixingQuick**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The combination of encoding modes may not be the optimal one, in order to optimize encoding speed.
- **epSingleMode**: The encoding mode will be selected automatically and applied to entire symbol, based on the barcode text, in other words, the barcode text to be encoded will be analysed, and an appropriates lowest level (highest encoding density) encoding mode will be selected, in order to minimize the symbol size, and the encoding mode will not be switched in the sybmol. Note, the Kanji mode will not be used if the AllowKanjiMode property is set to false.

See also the "Encoding modes" section in the "TBarcode2D_MicroQRCode" article.

A.1.61 EncodePolicy

(TBarcode2D_QRCode)

Indicates how to use the encoding mode in a QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pQRCodeCom unit }
  TQRCode_EncodePolicy = (epMixingOptimal, epMixingQuick, epSingleMode);
property EncodePolicy: TQRCode_EncodePolicy;
```

Description:

The QR Code symbology offers four kinds of encoding mode, referred to as Numeric mode, Alphanumeric mode, Kanji mode and Byte mode respectively in decreasing order of encoding density.

This property indicates how to use these encoding mode by the component. It can be one of these values (defined in the pQRCodeCom unit):

- **epMixingOptimal**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The optimal combination of encoding modes will be used in order to minimize the symbol size.
- **epMixingQuick**: All 256 8-bits values are encoded by switching automatically between all 4 (the AllowKanjiMode property is

set to true) or 3 (the Kanji mode will not be used if the AllowKanjiMode property is set to false) encoding modes. The combination of encoding modes may not be the optimal one, in order to optimize encoding speed.

- **epSingleMode**: The encoding mode will be selected automatically and applied to entire symbol, based on the barcode text, in other words, the barcode text to be encoded will be analysed, and an appropriates lowest level (highest encoding density) encoding mode will be selected, in order to minimize the symbol size, and the encoding mode will not be switched in the sybmol. Note, the Kanji mode will not be used if the AllowKanjiMode property is set to false.

See also the "Encoding modes" section in the "TBarcode2D_QRCode" article.

A.1.62 FixedLength

(TBarcode2D_AztecRunnes)

Specifies whether the leading zeros are required in the barcode text of an Aztec Runes barcode symbol.

Syntax:

```
property FixedLength: Boolean;
```

Description:

For the Aztec Runes barcode symbology, if the barcode text is specified by the Barcode property, it must be set to a number string. If it's secified by the Data property (only for Delphi/C++ Builder 2009 or later), it must be set to a number inAnsiString format. The number is an integer between 0 and 255 (including the boundaries). The property specifies whether the leading zeros are required in the barcode text.

If the property is set to true, the length of the number string must be exactly equal to 3 digits, adding leading zeros are required if the length is less than 3 digits, such as '0', '2', and '25'.

If the property is set to false, the length of the number string should be less than, or equal to 3 digits, the leading zeros are optional, such as '0', '12', '012', and '255'. In this case, the barcode symbol isn't changed when you remove leading zeros from the barcode text or add leading zeros to the barcode text. For example, the barcode symbol '12' is same as barcode symbol '012'.

Note, for the Aztec Runes barcode symbology, the Barcode property cannot be set to an empty string, otherwise, an OnInvalidLength event will occur. Also, the Data (only for Delphi/C++ Builder 2009 or later) property cannot be set to an emptyAnsiString, otherwise, an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

See also the "Character set" and the "Data capacity" sections in the "TBarcode2D_AztecRunes" article.

A.1.63 Image

Specifies a TImage, TQRImage, or TQRGzImage control to represent the barcode symbol.

Syntax:

```
property Image: TControl;
```

Description:

Specifies a TImage control to represent the barcode symbol. The barcode picture will not be saved into the DFM file in design time. If you use the component with QuikeReport, you may specifiey a TQRImage or TQRGzImage control for the property.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D

symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be represent on the TImage, TQRImage, or TQRGzImage control.

You can link single TImage, TQRImage, or TQRGzImage control to multiple TBarcode2D components in order to display multiple barcode symbols in the TImage, TQRImage, or TQRGzImage control (using the LeftMargin and TopMargin properties to specify the position for every barcode symbol).

A.1.64 InitialMode

(TBarcode2D_Code16K)

Specifies the initial mode for Code 16K barcode symbology.

Syntax:

```
type
  { Defined in the pCode16K unit }
  TCode16K_EncodeMode = (emAuto, emModeA, emModeB, emModeC, emModeB_FNC1, emModeC_FNC1,
    emModeC_Shift1B, emModeC_Shift2B, emMode_Extended);
property InitialMode: TCode16K_EncodeMode;
```

Description:

The property specifies the initial mode for Code 16K barcode symbology. The initial mode indicates the initial code set and may also represent an implied leading FNC1 character or implied leading SHIFT B character as shown in following table. The code set will be automatically switched if a character is encountered that cannot be encoded by current code set. And Implied characters function as if they were actual symbol characters but don not occupy any space.

There are seven kinds of initial mode, from 0 to 6, and a kind of extended data length mode. The initial mode values from emCodeA to emExtended, corresponding to these modes, are defined in the pCode16K unit. These modes and their values are listed in following table:

Initial Mode	Value	Initial code set	Implied character	Description
0	emCodeA	A	(None)	The code set will be automatically switched if another code set character is encountered.
1	emCodeB	B	(None)	
2	emCodeC	C	(None)	
3	emCodeB_FNC1	B	FNC1	
4	emCodeC_FNC1	C	FNC1	
5	emCodeC_Shift1B	C	SHIFT B	First character excepting the message append block (if exists), must be code set B character (ASCII 32 - ASCII 127), otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. The code set will be automatically switched if another code set character is encountered.
6	emCodeC_Shift2B	C	Double SHIFT B	First two characters excepting the message append block (if exists), must be code set B characters (ASCII 32 - ASCII 127), otherwise an OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. The code set will be automatically switched if another code set character is encountered.
Extended data length mode	emMode_Extended	B	None	Indicates to create a Code 16K barcode symbol in extended data length mode. An extended data length mode block is required, and it should be placed at beginning of barcode text, otherwiset the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur. See also the "Extended data length mode" section in the "TBarcode2D_Code16K" article.

The property can be one of these values: emModeA, emModeB, emModeC, emModeB_FNC1, emModeC_FNC1, emModeC_Shift1B, emModeC_Shift2B, and emMode_Extended, corresponding to the initial modes 0 to 6, and the extended data length mode. In this case, it specifies the factual initial mode shown in table above.

Also, the InitialMode property can be set to emAuto (defined in the pCode16K unit), in this case, one of values from emCodeA to emModeC_Shift2B, corresponding to the initial modes 0 to 6 shown in table above will be selected automatically, depending on the barcode text, in order to minimize the symbol size.

You can always use the CurrentMode property to get the factual initial mode.

See also the "Initial modes" section in the "TBarcode2D_Code16K" article.

A.1.65 Inversed

(TBarcode2D_AztecCode, TBarcode2D_DataMatrix, TBarcode2D_QRCode, etc.)

Specifies whether to represent a barcode symbol in inversed color (reflectance reversal).

Syntax:

```
property Inversed: Boolean;
```

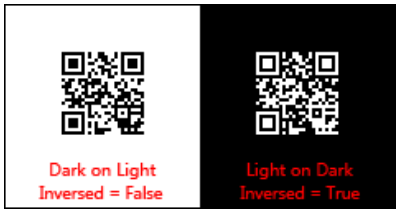
Description:

Specifies whether to represent a barcode symbol in inversed color (reflectance reversal).

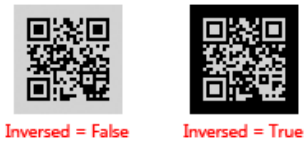
In general, the barcode symbol is dark foreground on light background. In this case, the Inversed property should be set to false, and all light modules in the barcode symbol are represented using the color specified by the SpaceColor property, all dark modules are represented using the color specified by the BarColor property.

If the barcode symbol is light foreground on dark background, the Inversed property should be set to true, and all light modules in the barcode symbol are represented using the color specified by the BarColor property, all dark modules are represented using the color specified by the SpaceColor property.

See diagram:



When the leading quiet zone, trailing quiet zone, top quiet zone, and bottom quiet zone are represented (the ShowQuietZone property value is set to true), if the Inversed property is set to false, the color specified by the SpaceColor property will be used to represent the quiet zones, otherwise, the color specified by the BarColor property will be used. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



A.1.66 LeadingQuietZone

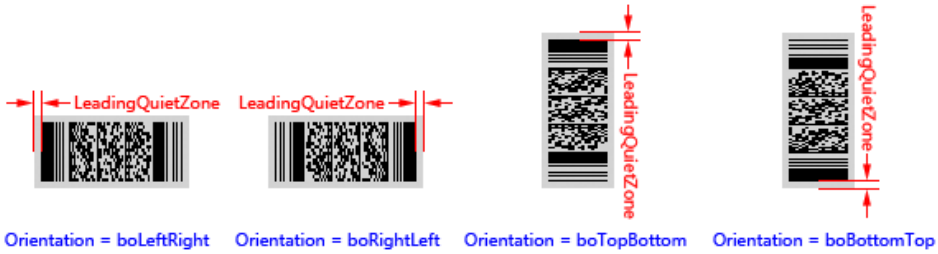
Specifies the horizontal width of the leading quiet zone in modules.

Syntax:

```
property LeadingQuietZone: Integer;
```

Description:

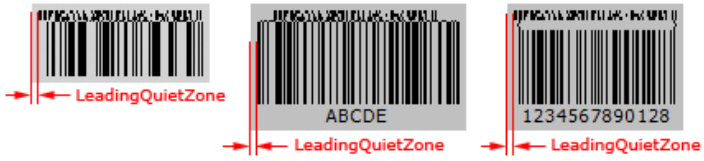
Specifies the horizontal width of the leading quiet zone in modules. The quiet zone is represented using the color specified by the SpaceColor property if the Inversed property is set to false. Otherwise, it's represented using the color specified by the BarColor property. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



This property is useful only when the ShowQuietZone property value is set to true or the barcode symbology is a Code 16K symbol.

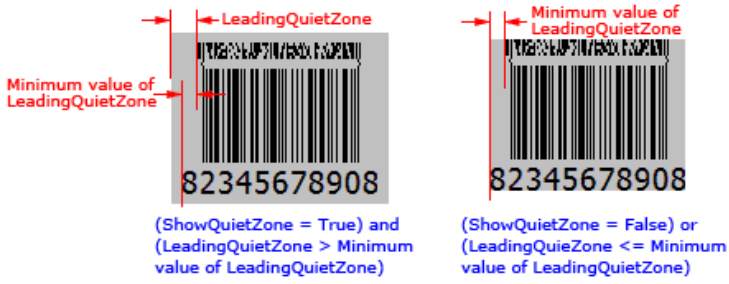
For the TBarcode2D_Code16K barcode component, the leading quiet zone is represented always, even if the ShowQuietZone property value is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the leadinging quiet zone reachs to the first bar or space of entire EAN.UCC composite barcode symbol that includes the linear symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



Note, For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is one of the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, and TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, its ShowQuietZoneMark property is set to true too), the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol is included in the width of leading quiet zone, and it's the minimum width of the leading quiet zone. In other words, the leading quiet zone always reachs to the first bar of the 2D symbol. The minimum width will be used if the value of the LeadingQuietZone property is less than it or the value of the ShowQuietZone property is set to false, the leading quiet zoens will be automatically extanded to cover the left quiet zone mark, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning of entire barcode symbol (the leading quiet zones is included if the ShowQuietZone property is set to true), the leading quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



A.1.67 LeftMargin

Specifies the left margin of the barcode symbol in pixels.

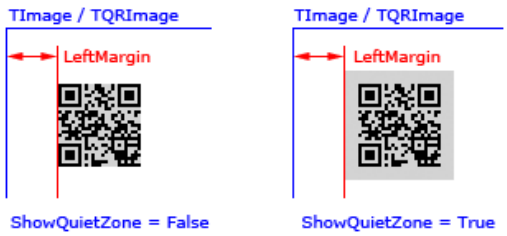
Syntax:

```
property LeftMargin: Integer;
```

Description:

Specifies the margin between the leftmost side of the barcode symbol and the left side of the TImage, TQRImage, or TQRGzImage control that is specified by the Image property, in pixels.

If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

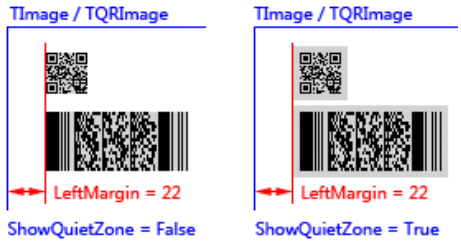


It is set using the following formula:

- The Orientation property is set to "boLeftRight":

  It is the margin between the beginning of the barcode symbol and the left side of the TImage, TQRImage, or TQRGzImage control.

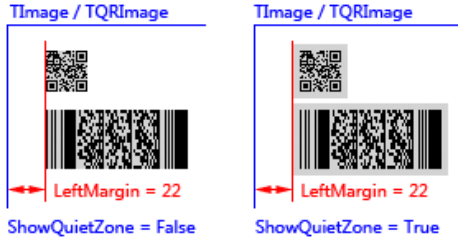
  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- The Orientation property is set to "boRightLeft":

  It is the margin between the end of the barcode symbol and the left side of the TImage, TQRImage, or TQRGzImage control.

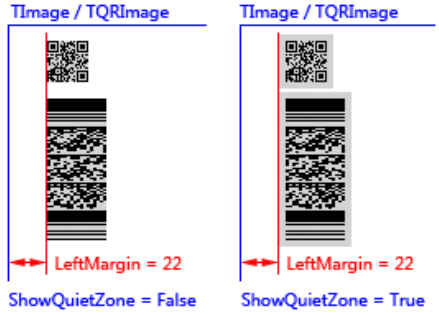
  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- The Orientation property is set to "boTopBottom":

  It is the margin between the bottom of the barcode symbol and the left side of the TImage, TQRImage, or TQRGzImage control.

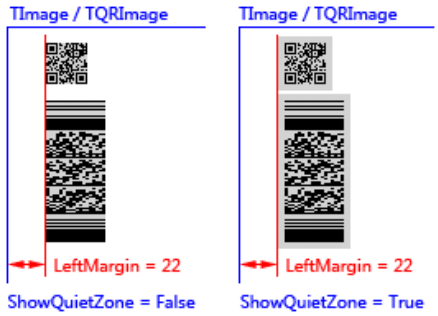
  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- The Orientation property is set to "boBottomTop":

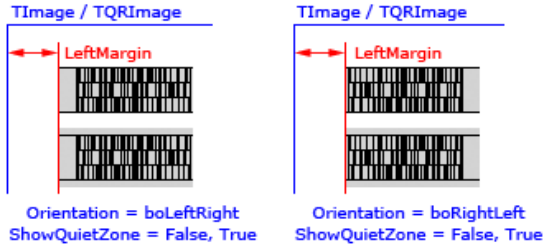
  It is the margin between the top of the barcode symbol and the left side of the TImage, TQRImage, or TQRGzImage control.

  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



Note:

For the TBarcode2D_Code16K barcode components, the leading quiet zone and trailing quiet zone are included always, even if the ShowQuietZone is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the property specifies the left margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

When the linear component of the TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The property always specifies the left margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

A.1.68 Linear

(TBarcode2D_CCA, TBarcode2D_CCB, TBarcode2D_CCC)

Specifies a linear component for current 2D barcode component in order to generate the EAN.UCC composite barcode symbol.

Syntax:

```
property Linear: TComponent;
```

Description:

Specifies a linear component for current 2D barcode component (the TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC) so that the 2D barcode component together with the linear component generates the EAN.UCC composite barcode symbol.

For the TBarcode2D_CCA and the TBarcode2D_CCB 2D barcode components, the TBarcode2D_RSS14, TBarcode2D_RSSLimted, and TBarcode2D_RSSExpanded can be used as their linear component. The TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, and TBarcode1D_EAN128 that they are from the **1D Barcode VCL Components** package can be used as their linear component too.

- If the TBarcode2D_RSS14 is used as the linear component, only its Barcode, Data (only for Delphi/C++ Builder 2009 or later), AutoCheckDigit, Style, TotalHeight, and SeparatorRowHeight properties are useful, the value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D and Show2DSeparator properties will be set to true automatically). The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed automatically depending on the value of the Style property. In addition, the Image property should not be normally set, the Image property's value of the 2D component will be used.

- If the TBarcode2D_RSSLimited is used as the linear component, only its Barcode, Data (only for Delphi/C++ Builder 2009 or later), AutoCheckDigit, TotalHeight, and SeparatorRowHeight properties are useful, the value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D and Show2DSeparator properties will be set to true automatically). The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed to 3 automatically. In addition, the Image property should not be normally set, the Image property's value of the 2D component will be used.

- If the TBarcode2D_RSSExpanded is used as the linear component, only its Barcode, Data (only for Delphi/C++ Builder 2009 or later), Style, RowSymbols, RowHeight, and SeparatorRowHeight properties are useful, and the value of RowSymbols property should be greater than or equal to 4. The value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D and Show2DSeparator properties will be set to true automatically). The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed to 4 automatically. In addition, the Image property should not be normally set, the Image property's value of the 2D component will be used.

- If the TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, or TBarcode1D_EAN13 is used as the linear component, only its Barcode, Data (only for Delphi/C++ Builder 2009 or later), AutoCheckDigit, Height, DisplayText, TextPosition, TextAlignment, TextFont, TextVSpacing, TextHSpacing, ExtraFontSize (only for TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, and TBarcode1D_UPCE1), LeftQuietZoneSpacing, and the RightQuietZoneSpacing properties are useful, the value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component. The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed automatically. In addition, the BarcodeHeight property should be set to a value larger than zero. And the Image property should not be normally set, the Image property's value of the 2D component will be used.

- If the TBarcode1D_EAN128 barcode component is used as the linear component, only its Barcode, Data (only for Delphi/C++ Builder 2009 or later), AutoCheckDigit, CheckStart, CheckLength, InitialSubSet, Height, DisplayText, TextPosition, TextAlignment, TextFont, TextVSpacing, and the TextHSpacing properties are useful, the value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D property will be set automatically depending on which 2D component is used). The number of columns of the TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component will be changed to 4 automatically. In addition, the BarcodeHeight property should be set to a value larger than zero. And the Image property should not be normally set, the Image property's value of the 2D component will be used.

For the TBarcode2D_CCC 2D barcode component, only the TBarcode1D_EAN128 that's from the **1D Barcode VCL Components** package can be used as its linear component. And only its Barcode, Data (only for Delphi/C++ Builder 2009 or later), AutoCheckDigit, CheckStart, CheckLength, InitialSubSet, Height, DisplayText, TextPosition, TextAlignment, TextFont, TextVSpacing, and the TextHSpacing properties are useful, the value of other properties will be ignored and they will be set automatically depending on the settings of the 2D barcode component (the Link2D property will be set to ldCCC automatically). The number of columns of the TBarcode2D_CCC barcode component will be changed automatically. In addition, the BarcodeHeight property should be set to a value larger than zero. And the Image property should not be normally set, the Image property's value of the 2D component will be used.

A.1.69 Link2D

(TBarcode2D_RSS14, TBarcode2D_RSSLimited, TBarcode2D_RSSExpanded)

Specifies whether an RSS symbol (RSS-14, RSS Limited, or RSS Expanded) can be used as the linear component of an EAN.UCC Composite symbol.

Syntax:

```
property Link2D: Boolean;
```

Description:

An RSS barcode symbol (including RSS-14, RSS Limited, and RSS Expanded) can be used together with a CC-A or a CC-B 2D component to create the EAN.UCC composite symbol. The property specifies whether the RSS symbol can be used as the linear component of an EAN.UCC Composite symbol.

If the property is set to true, the RSS symbol and its contiguous separator pattern shall be aligned with a 2D component. Normally the

RSS symbol, its contiguous separator pattern, and the 2D component are represented at the same time, comprising a single EAN.UCC composite symbol.

If the property is set to false, the RSS symbol will be used as a stand-alone barcode symbol.

Note, a contiguous separator pattern is required between the RSS and the 2D component in an EAN.UCC Composite symbol. In order to represent the separator pattern on top of the RSS symbol, both the Show2DSeparator and the Link2D properties must be set to true.

If you use the RSS symbol as the Linear property's value of a TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component to create the EAN.UCC composite symbol, the Show2DSeparator and the Link2D properties will be set to true automatically.

See also the "Composite symbol" section in the "TBarcode2D_RSS14", "TBarcode2D_RSSLimited", and "TBarcode2D_RSSExpanded" articles.

The following is two examples of the EAN.UCC composite symbol:

A.1.70 Locked

Set the property to true to prevents updating of the barcode component until the property is reset to false.

Syntax:

```
property Locked: Boolean;
```

Description:

Set the property to true before making multiple changes to the barcode component. When all changes are complete, change the property to false so that the changes can be reflected on screen. It prevents excessive redraws and speed processing time when change the barcode component.

Set the property to true is similar to BeginUpdate method of an other Delphi/C++ Builder control such as a ListBox, Memo, TreeList, ListView, and set the property to false is similar to its EndUpdate method.

A.1.71 MaxColumns

(TBarcode2D_CCC)

Specifies the maximum number of rows for the CC-C 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Columns = 1 .. 30;
property MaxColumns: TPDF417_Columns;
```

Description:

The property specifies the maximum number of columns for a CC-C 2D component of an EAN.UCC composite barcode symbol. The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology. It's a variant of PDF417 symbol which begins with PDF417 codeword 920. The property can be one of values from 1 to 30, they are defined in the pPDF417Custom unit.

The maximum number of columns together with the maximum number of stacked rows (specified by the MaxRows property) indicates the maximum symbol size for a CC-C 2D component of the EAN.UCC composite barcode symbol. And the minimum number of columns (specified by the MinColumns property) together with the minimum number of rows (specified by the MinRows property) indicates the minimum symbol size for it. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

If the barcode text is so long that it cannot be encoded using the maximum symbol size, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value less than the MinColumns property's value, the MinColumns property will be automatically changed to equal to this property's value. In other words, the MaxColumns property's value is always greater than or equal to the MinColumns property's value.

See also the "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.72 MaxColumns

(TBarcode2D_PDF417)

Specifies the maximum number of columns for a PDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Columns = 1 .. 30;
property MaxColumns: TPDF417_Columns;
```

Description:

The property specifies the maximum number of columns for a PDF417 barcode symbol. It can be one of values from 1 to 30, they are defined in the pPDF417Custom unit.

The maximum number of columns together with the maximum number of stacked rows (specified by the MaxRows property) indicates the maximum symbol size for a PDF417 barcode symbol. And the minimum number of columns (specified by the MinColumns property) together with the minimum number of stacked rows (specified by the MinRows property) indicates the minimum symbol size for the PDF417 barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

If the barcode text is so long that it cannot be encoded using the maximum symbol size, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentRows property can be used to get the factual number of rows.

Note, if the property is set to a value less than the MinColumns property's value, the MinColumns property will be automatically changed to equal to this property's value. In other words, the MaxColumns property's value is always greater than or equal to the MinColumns property's value.

See also the "Symbol size" section in the "TBarcode2D_PDF417" article.

A.1.73 MaxRows

(TBarcode2D_CCA)

Specifies the maximum number of stacked rows for the CC-A 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pCCA unit }
  TCCA_Rows = (ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9,
    ccaRow_10, ccaRow_12);
property MaxRows: TCCA_Rows;
```

Description:

The property specifies the maximum number of stacked rows for a CC-A 2D component of an EAN.UCC composite barcode symbol. It can be one of these values: ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9, ccaRow_10, and ccaRow_12, corresponding to the number of stacked rows 3, 4, 5, 6, 7, 8, 9, 10, and 12. For example, the value ccaRow_12 denotations the maximum number of stacked rows is 12 rows. These values are defined in the pCCA unit. The CC-A is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

The smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by the MinRows property) and the maximum number of stacked rows (specified by this property).

Note, only available number of stacked rows can be automatically selected between the range specified by the MinRows property and this property depending on the linear component of the EAN.UCC composite symbol (specified by the Linear property). They are listed in the following table:

Linear component	Availiable number of stacked rows
UPC-E, UPC-E0, UPC-E1, RSS-14 Stacked, RSS-14 Stacked Omnidirectional	ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9, ccaRow_10, ccaRow_12
EAN-8, RSS Limited	ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8
UPC-A, EAN-13, EAN-128, RSS-14 Standard, RSS-14 Truncated, RSS Expanded, RSS Expanded Stacked	ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7

If the barcode text is so long that it cannot be encoded using the maximum number of stacked rows specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value less than the MinRows property's value, the MinRows property will be automatically changed to equal to this property's value. In other words, the MaxRows property's value is always greater than or equal to the MinRows property's value.

See also the "Symbol size" section in the "TBarcode2D_CCA" article.

A.1.74 MaxRows

(TBarcode2D_CCB)

Specifies the maximum number of stacked rows for the CC-B 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pCCB unit }
  TCCB_Rows = (ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_11, ccbRow_12, ccbRow_14,
    ccbRow_15, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26, ccbRow_32, ccbRow_38,
    ccbRow_44);
property MaxRows: TCCB_Rows;
```

Description:

The property specifies the maximum number of stacked rows for a CC-B 2D component of an EAN.UCC composite barcode symbol. It can be one of these values: ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_11, ccbRow_12, ccbRow_14, ccbRow_15, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26, ccbRow_32, ccbRow_38, and ccbRow_44, corresponding to the number of stacked rows 4, 6, 8, 10, 11, 12, 14, 15, 17, 20, 23, 26, 32, 38 and 44. For example, the value ccbRow_44 denotations the maximum number of stacked rows is 44 rows. These values are defined in the pCCB unit. The CC-B is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

The smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by the MinRows property) and the maximum number of stacked rows (specified by this property).

Note, only available number of stacked rows can be automatically selected between the range specified by the MinRows property and this property depending on the linear component of the EAN.UCC composite symbol (specified by the Linear property). They are listed in the following table:

Linear component	Availiable number of stacked rows
UPC-E, UPC-E0, UPC-E1, RSS-14 Stacked, RSS-14 Stacked Omnidirectional	ccbRow_8, ccbRow_11, ccbRow_14, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26
EAN-8, RSS Limited	ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_12, ccbRow_15, ccbRow_20, ccbRow_26, ccbRow_32, ccbRow_38, ccbRow_44
UPC-A, EAN-13, EAN-128, RSS-14 Standard, RSS-14 Truncated, RSS Expanded, RSS Expanded Stacked	ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_12, ccbRow_15, ccbRow_20, ccbRow_26, ccbRow_32, ccbRow_38, ccbRow_44

If the barcode text is so long that it cannot be encoded using the maximum number of stacked rows specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value less than the MinRows property's value, the MinRows property will be automatically changed to equal to this property's value. In other words, the MaxRows property's value is always greater than or equal to the MinRows property's value.

See also the "Symbol size" section in the "TBarcode2D_CCB" article.

A.1.75 MaxRows

(TBarcode2D_CCC)

Specifies the maximum number of stacked rows for the CC-C 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Rows = 3 .. 90;
property MaxRows: TPDF417_Rows;
```

Description:

The property specifies the maximum number of stacked rows for a CC-C 2D component of an EAN.UCC composite barcode symbol. It can be one of values from 3 to 90. They are defined in the pPDF417Custom unit. The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology. It's a variant of PDF417 symbol which begins with PDF417 codeword 920.

The maximum number of stacked rows together with the maximum number of columns (specified by the MaxColumns property) indicates the maximum symbol size for a CC-C 2D component of the EAN.UCC composite barcode symbol. And the minimum number of stacked rows (specified by the MinRows property) together with the minimum number of columns (specified by the MinColumns property) indicates the minimum symbol size for it. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

If the barcode text is so long that it cannot be encoded using the maximum symbol size, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value less than the MinRows property's value, the MinRows property will be automatically changed to equal to this property's value. In other words, the MaxRows property's value is always greater than or equal to the MinRows property's value.

See also the "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.76 MaxRows

(TBarcode2D_Code16K)

Specifies the maximum number of stacked rows for a Code 16K barcode symbol.

Syntax:

```
type
  { Defined in the pCode16K unit }
  TCode16K_Rows = 2..16;
property MaxRows: TCode16K_Rows;
```

Description:

The property specifies the maximum number of stacked rows for a Code 16K barcode symbol. It can be one of values from 2 to 16. They are defined in the pCode16K unit.

The smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by the MinRows property) and the maximum number of stacked rows (specified by this property).

If the barcode text is so long that it cannot be encoded using the maximum number of stacked rows specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value less than the MinRows property's value, the MinRows property will be automatically changed to equal to this property's value. In other words, the MaxRows property's value is always greater than or equal to the MinRows property's value.

See also the "Symbol sizes" section in the "TBarcode2D_Code16K" article.

A.1.77 MaxRows

(TBarcode2D_PDF417)

Specifies the maximum number of stacked rows for a PDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Rows = 3 .. 90;
property MaxRows: TPDF417_Rows;
```

Description:

The property specifies the maximum number of stacked rows for a PDF417 barcode symbol. It can be one of values from 3 to 90. They are defined in the pPDF417Custom unit.

The maximum number of stacked rows together with the maximum number of columns (specified by the MaxColumns property) indicates the maximum symbol size for a PDF417 barcode symbol. And the minimum number of stacked rows (specified by the MinRows property) together with the minimum number of columns (specified by the MinColumns property) indicates the minimum symbol size for the PDF417 barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

If the barcode text is so long that it cannot be encoded using the maximum symbol size, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value less than the MinRows property's value, the MinRows property will be automatically changed to equal to this property's value. In other words, the MaxRows property's value is always greater than or equal to the MinRows property's value.

See also the "Symbol size" section in the "TBarcode2D_PDF417" article.

A.1.78 MaxSegments

(TBarcode2D_CompactMatrix)

Specifies the maximum number of segments for a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_Segments = 1..32;
property MaxSegments: TCompactMatrix_Segments;
```

Description:

In horizontal orientation, each Compact Matrix symbol consists of an array of segments with a minimum of 1 segment (maximum 32 segments). The property specifies the maximum number of segments for a Compact Matrix barcode symbol. It can be one of values from 1 to 32, they are defined in the pCompactMatrix unit.

The maximum number of segments together with the maximum version (specified by the MaxVersion property) indicates the maximum symbol size for a Compact Matrix barcode symbol. And the minimum number of segments (specified by the MinSegments property) together with the minimum version (specified by the MinVersion property) indicates the minimum symbol size for the Compact Matrix barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

If the barcode text is so long that it cannot be encoded using the maximum symbol size, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentSegments property can be used to get the factual number of segments.

Note, if the property is set to a value less than the MinSegments property's value, the MinSegments property will be automatically changed to equal to this property's value. In other words, the MaxSegments property's value is always greater than or equal to the MinSegments property's value.

See also the "Symbol sizes" section in the "TBarcode2D_CompactMatrix" article.

A.1.79 MaxSize

(TBarcode2D_AztecCode)

Specifies the maximum symbol size for an Aztec Code barcode symbol.

Syntax:

```
type
  { Defined in the pAztecCode unit }
  TAztecCode_Size = (azSize_15Compact, azSize_19, azSize_19Compact, azSize_23,
    azSize_23Compact, azSize_27, azSize_27Compact, azSize_31, azSize_37, azSize_41,
    azSize_45, azSize_49, azSize_53, azSize_57, azSize_61, azSize_67, azSize_71,
    azSize_75, azSize_79, azSize_83, azSize_87, azSize_91, azSize_95, azSize_101,
    azSize_105, azSize_109, azSize_113, azSize_117, azSize_121, azSize_125, azSize_131,
    azSize_135, azSize_139, azSize_143, azSize_147, azSize_151);
```

```
property MaxSize: TAztecCode_Size;
```

Description:

The property specifies the maximum symbol size for an Aztec Code barcode symbol. It can be one of values from azSize_15Compact to azSize_151. These values denotation the symbol format, number of rows and columns in every symbol size. For example, the symbol size value azSize_91 denotations the Aztec Code symbol size is 91 rows by 91 columns, not in compact format. These symbol size values are defined in the pAztecCode unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value less than the MinSize property's value, the MinSize property will be automatically changed to equal to this property's value. In other words, the MaxSize property's value is always greater than or equal to the MinSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_AztecCode" article.

A.1.80 MaxSize

(TBarcode2D_DataMatrix)

Specifies the maximum symbol size for a Data Matrix (ECC 000 - 140) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrix unit }
  TDataMatrix_Size = (dmSize_09_09, dmSize_11_11, dmSize_13_13, dmSize_15_15,
    dmSize_17_17, dmSize_19_19, dmSize_21_21, dmSize_23_23, dmSize_25_25, dmSize_27_27,
    dmSize_29_29, dmSize_31_31, dmSize_33_33, dmSize_35_35, dmSize_37_37, dmSize_39_39,
    dmSize_41_41, dmSize_43_43, dmSize_45_45, dmSize_47_47, dmSize_49_49);
property MaxSize: TDataMatrix_Size;
```

Description:

The property specifies the maximum symbol size for a Data Matrix (ECC 000 - 140)barcode symbol. It can be one of values from dmSize_09_09 to dmSize_49_49. These values denotation the number of rows and columns in every symbol size. For example, the symbol size value dmSize_45_45 denotations the Data Matrix (ECC 000 - 140)symbol size is 45 rows by 45 columns. These symbol size values are defined in the pDataMatrix unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by this property, an OnInvalidLength the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value less than the MinSize property's value, the MinSize property will be automatically changed to

equal to this property's value. In other words, the MaxSize property's value is always greater than or equal to the MinSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_DataMatrix" article.

A.1.81 MaxSize

(TBarcode2D_DataMatrixECC200)

Specifies the maximum symbol size for a Data Matrix (ECC 200) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrixEcc200 unit }
  TDataMatrixEcc200_Size = (dmSize_10_10, dmSize_12_12, dmSize_14_14, dmSize_16_16,
    dmSize_18_18, dmSize_20_20, dmSize_22_22, dmSize_24_24, dmSize_26_26, dmSize_32_32,
    dmSize_36_36, dmSize_40_40, dmSize_44_44, dmSize_48_48, dmSize_52_52, dmSize_64_64,
    dmSize_72_72, dmSize_80_80, dmSize_88_88, dmSize_96_96, dmSize_104_104,
    dmSize_120_120, dmSize_132_132, dmSize_144_144, dmSize_8_18, dmSize_8_32,
    dmSize_12_26, dmSize_12_36, dmSize_16_36, dmSize_16_48);
property MaxSize: TDataMatrixEcc200_Size;
```

Description:

The property specifies the maximum symbol size for a Data Matrix (ECC 200) barcode symbol. It can be one of values from dmSize_10_10 to dmSize_16_48. These values denotation the number of rows and columns in every symbol size. For example, the symbol size value dmSize_16_36 denotations the Data Matrix (ECC 200) symbol size is 16 rows by 36 columns. These symbol size values are defined in the pDataMatrixEcc200 unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value less than the MinSize property's value, the MinSize property will be automatically changed to equal to this property's value. In other words, the MaxSize property's value is always greater than or equal to the MinSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_DataMatrixECC200" article.

A.1.82 MaxSize

(TBarcode2D_MicroPDF417)

Specifies the maximum symbol size for a MicroPDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_Size = (mpSize_1_11, mpSize_1_14, mpSize_1_17, mpSize_1_20, mpSize_1_24,
    mpSize_1_28, mpSize_2_8, mpSize_2_11, mpSize_2_14, mpSize_2_17, mpSize_2_20,
    mpSize_2_23, mpSize_2_26, mpSize_3_6, mpSize_3_8, mpSize_3_10, mpSize_3_12,
    mpSize_3_15, mpSize_3_20, mpSize_3_26, mpSize_3_32, mpSize_3_38, mpSize_3_44,
    mpSize_4_4, mpSize_4_6, mpSize_4_8, mpSize_4_10, mpSize_4_12, mpSize_4_15,
    mpSize_4_20, mpSize_4_26, mpSize_4_32, mpSize_4_38, mpSize_4_44);
property MaxSize: TMicroPDF417_Size;
```

Description:

The property specifies the maximum symbol size for a MicroPDF417 barcode symbol. It can be one of values from mpSize_1_11 to mpSize_4_44. These values denotation the number of columns and rows in every symbol size. For example, the symbol size value mpSize_4_10 denotations the MicroPDF417 symbol size is 10 stacked rows by 4 columns. These symbol size values are defined in the pMicroPDF417 unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by the MinSize property and the maximum symbol size specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol size specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, the barcode text is specified in the Data property) event will occur.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value less than the MinSize property's value, the MinSize property will be automatically changed to equal to this property's value. In other words, the MaxSize property's value is always greater than or equal to the MinSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_MicroPDF417" article.

A.1.83 MaxSliceLen

(TBarcode2D_MicroQRCode)

Specifies the maximum number of bytes to slice the barcode data, in order to limit the depth of recursion.

Syntax:

```
property MaxSliceLen: Integer;
```

Description:

If the property EncodePolicy is set to epMixingOptimal, this property specifies the maximum number of bytes to slice the barcode data or text, each piece will be encoded in optimal combination of encoding modes using the recursion algorithm. It's useful to limit the depth of recursion, in order to prevent the computer has no response.

If the property is set to 0, the barcode data or text will not be sliced, and it will be encoded in optimal combination of encoding modes at a time, using the recursion algorithm.

If the property EncodePolicy isn't set to epMixingOptimal, the property value will be ignored.

A.1.84 MaxSliceLen

(TBarcode2D_QRCode)

Specifies the maximum number of bytes to slice the barcode data, in order to limit the depth of recursion.

Syntax:

```
property MaxSliceLen: Integer;
```

Description:

If the property EncodePolicy is set to epMixingOptimal, this property specifies the maximum number of bytes to slice the barcode data or text, each piece will be encoded in optimal combination of encoding modes using the recursion algorithm. It's useful to limit the depth of recursion, in order to prevent the computer has no response.

If the property is set to 0, the barcode data or text will not be sliced, and it will be encoded in optimal combination of encoding modes at a time, using the recursion algorithm.

If the property EncodePolicy isn't set to epMixingOptimal, the property value will be ignored.

A.1.85 MaxVersion

(TBarcode2D_CompactMatrix)

Specifies the maximum version for a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_Version = 1..32;
property MaxVersion: TCompactMatrix_Version;
```

Description:

There are 32 vertical sizes of Compact Matrix symbol, referred to as version 1 to 32, in increasing order of symbol height and data capacity. The property specifies the maximum version for a Compact Matrix barcode symbol. It can be one of values from 1 to 32. They are defined in the pCompactMatrix unit.

The maximum version together with the maximum number of segments (specified by the MaxSegments property) indicates the maximum symbol size for a Compact Matrix barcode symbol. And the minimum version (specified by the MinVersion property) together with the minimum number of segments (specified by the MinSegments property) indicates the minimum symbol size for the Compact Matrix barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

If the barcode text is so long that it cannot be encoded using the maximum symbol size, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentVersion property can be used to get the factual version.

Note, if the property is set to a value less than the MinVersion property's value, the MinVersion property will be automatically changed to equal to this property's value. In other words, the MaxVersion property's value is always greater than or equal to the MinVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_CompactMatrix" article.

A.1.86 MaxVersion

(TBarcode2D_GridMatrix)

Specifies the maximum symbol version (symbol size) for a Grid Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pGridMatrix unit }
  TGridMatrix_Version = 1..13;
property MaxVersion: TGridMatrix_Version;
```

Description:

There are thirteen sizes of Grid Matrix symbol, referred to as version 1 to 13, in increasing order of size and data capacity. The property specifies the maximum symbol version (symbol size) for a Grid Matrix barcode symbol. It can be one of values from 1 to 13. They are defined in the pGridMatrix unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol version (symbol size) specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value less than the MinVersion property's value, the MinVersion property will be automatically changed to equal to this property's value. In other words, the MaxVersion property's value is always greater than or equal to the MinVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_GridMatrix" article.

A.1.87 MaxVersion

(TBarcode2D_HanXinCode)

Specifies the maximum symbol version (symbol size) for a Han Xin Code barcode symbol.

Syntax:

```
type
  { Defined in the pHanXinCode unit }
  THanXinCode_Version = 1..84;
property MaxVersion: THanXinCode_Version;
```

Description:

There are eighty-four sizes of Han Xin Code symbols, referred to as version 1 to 84, in increasing order of size and data capacity. The property specifies the maximum symbol version (symbol size) for a Han Xin Code barcode symbol. It can be one of values from

1 to 84. They are defined in the pHanXinCode unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol version (symbol size) specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value less than the MinVersion property's value, the MinVersion property will be automatically changed to equal to this property's value. In other words, the MaxVersion property's value is always greater than or equal to the MinVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_HanXinCode" article.

A.1.88 MaxVersion

(TBarcode2D_MicroQRCode)

Specifies the maximum symbol version (symbol size) for a Micro QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pMicroQRCode unit }
  TMicroQRCode_Version = 1..4;
property MaxVersion: TMicroQRCode_Version;
```

Description:

The property specifies the maximum symbol version (symbol size) for a Micro QR Code barcode symbol. It can be one of values from 1 to 4, they are defined in the pMicroQRCode unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol version (symbol size) specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value less than the MinVersion property's value, the MinVersion property will be automatically changed to equal to this property's value. In other words, the MaxVersion property's value is always greater than or equal to the MinVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_MicroQRCode" article.

A.1.89 MaxVersion

(TBarcode2D_QRCode)

Specifies the maximum symbol version (symbol size) for a QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pQRCode unit }
  TQRCode_Version = 1..40;
property MaxVersion: TQRCode_Version;
```

Description:

The property specifies the maximum symbol version (symbol size) for a QR Code barcode symbol. It can be one of values from 1 to 40. They are defined in the pQRCode unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by the MinVersion property and the maximum symbol version (symbol size) specified by this property.

If the barcode text is so long that it cannot be encoded using the maximum symbol version (symbol size) specified by this property, an OnInvalidLength (the barcode text is specified in the Barcode property) or an OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value less than the MinVersion property's value, the MinVersion property will be automatically changed to equal to this property's value. In other words, the MaxVersion property's value is always greater than or equal to the MinVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_QRCode" article.

A.1.90 MinColumns

(TBarcode2D_CCC)

Specifies the minimum number of columns for the CC-C 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Columns = 1 .. 30;
property MinColumns: TPDF417_Columns;
```

Description:

The property specifies the minimum number of columns for the CC-C 2D component of an EAN.UCC composite barcode symbol. The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology. It's a variant of PDF417 symbol which begins with PDF417 codeword 920. The property can be one of values from 1 to 30, they are defined in the pPDF417Custom unit.

The minimum number of columns together with the minimum number of stacked rows (specified by the MinRows property) indicates the minimum symbol size for a CC-C 2D component of the EAN.UCC composite barcode symbol. And the maximum number of columns (specified by the MaxColumns property) together with the maximum number of stacked rows (specified by the MaxRows property) indicates the maximum symbol size for it. Based on the stretch order (specified by the StretchOrder property), the smallest

symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The CurrentColumns property can be used to get the factual number of columns.

Note, if the property is set to a value greater than the MaxColumns property's value, the MaxColumns property will be automatically changed to equal to this property's value. In other words, the MinColumns property's value is always less than or equal to the MaxColumns property's value.

See also the "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.91 MinColumns

(TBarcode2D_PDF417)

Specifies the minimum number of columns for a PDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Columns = 1 .. 30;
property MinColumns: TPDF417_Columns;
```

Description:

The property specifies the minimum number of columns for a PDF417 barcode symbol. It can be one of values from 1 to 30, they are defined in the pPDF417Custom unit.

The minimum number of columns together with the minimum number of stacked rows (specified by the MinRows property) indicates the minimum symbol size for a PDF417 barcode symbol. And the maximum number of columns (specified by the MaxColumns property) together with the maximum number of stacked rows (specified by the MaxRows property) indicates the maximum symbol size for the PDF417 barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The CurrentColumns property can be used to get the factual number of columns.

Note, if the property is set to a value greater than the MaxColumns property's value, the MaxColumns property will be automatically changed to equal to this property's value. In other words, the MinColumns property's value is always less than or equal to the MaxColumns property's value.

See also the "Symbol size" section in the "TBarcode2D_PDF417" article.

A.1.92 MinRows

(TBarcode2D_CCA)

Specifies the minimum number of stacked rows for the CC-A 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
```

```
  { Defined in the pCCA unit }
  TCCA_Rows = (ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9,
    ccaRow_10, ccaRow_12);
property MinRows: TCCA_Rows;
```

Description:

The property specifies the minimum number of stacked rows for the CC-A 2D component of an EAN.UCC composite barcode symbol. It can be one of these values: ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9, ccaRow_10, and ccaRow_12, corresponding to the number of stacked rows 3, 4, 5, 6, 7, 8, 9, 10, and 12. For example, the value ccaRow_3 denotations the minimum number of stacked rows is 3 rows. These values are defined in the pCCA unit. The CC-A is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

The smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by this property) and the maximum number of stacked rows (that's specified by the MaxRows property).

Note, only available number of stacked rows can be automatically selected between the range specified by this property and the MaxRows property depending on the linear component of the EAN.UCC composite symbol (specified by the Linear property). They are listed in the following table:

Linear component	Availiable number of stacked rows
UPC-E, UPC-E0, UPC-E1, RSS-14 Stacked, RSS-14 Stacked Omnidirectional	ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8, ccaRow_9, ccaRow_10, ccaRow_12
EAN-8, RSS Limited	ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7, ccaRow_8
UPC-A, EAN-13, EAN-128, RSS-14 Standard, RSS-14 Truncated, RSS Expanded, RSS Expanded Stacked	ccaRow_3, ccaRow_4, ccaRow_5, ccaRow_6, ccaRow_7

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value greater than the MaxRows property's value, the MaxRows property will be automatically changed to equal to this property's value. In other words, the MinRows property's value is always less than or equal to the MaxRows property's value.

See also the "Symbol size" section in the "TBarcode2D_CCA" article.

A.1.93 MinRows

(TBarcode2D_CCB)

Specifies the minimum number of stacked rows for the CC-B 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pCCB unit }
  TCCB_Rows = (ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_11, ccbRow_12, ccbRow_14,
    ccbRow_15, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26, ccbRow_32, ccbRow_38,
    ccbRow_44);
property MinRows: TCCB_Rows;
```

Description:

The property specifies the minimum number of stacked rows for the CC-B 2D component of an EAN.UCC composite barcode

symbol. It can be one of these values: ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_11, ccbRow_12, ccbRow_14, ccbRow_15, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26, ccbRow_32, ccbRow_38, and ccbRow_44, corresponding to the number of stacked rows 4, 6, 8, 10, 11, 12, 14, 15, 17, 20, 23, 26, 32, 38 and 44. For example, the value ccbRow_4 denotations the minimum number of stacked rows is 4 rows. These values are defined in the pCCB unit. The CC-B is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

The smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by this property) and the maximum number of stacked rows (that's specified by the MaxRows property).

Note, only available number of stacked rows can be automatically selected between the range specified by this property and the MaxRows property depending on the linear component of the EAN.UCC composite symbol (specified by the Linear property). They are listed in the following table:

Linear component	Availiable number of stacked rows
UPC-E, UPC-E0, UPC-E1, RSS-14 Stacked, RSS-14 Stacked Omnidirectional	ccbRow_8, ccbRow_11, ccbRow_14, ccbRow_17, ccbRow_20, ccbRow_23, ccbRow_26
EAN-8, RSS Limited	ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_12, ccbRow_15, ccbRow_20, ccbRow_26, ccbRow_32, ccbRow_38, ccbRow_44
UPC-A, EAN-13, EAN-128, RSS-14 Standard, RSS-14 Truncated, RSS Expanded, RSS Expanded Stacked	ccbRow_4, ccbRow_6, ccbRow_8, ccbRow_10, ccbRow_12, ccbRow_15, ccbRow_20, ccbRow_26, ccbRow_32, ccbRow_38, ccbRow_44

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value greater than the MaxRows property's value, the MaxRows property will be automatically changed to equal to this property's value. In other words, the MinRows property's value is always less than or equal to the MaxRows property's value.

See also the "Symbol size" section in the "TBarcode2D_CCB" article.

A.1.94 MinRows

(TBarcode2D_CCC)

Specifies the minimum number of stacked rows for the CC-C 2D component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
  TPDF417_Rows = 3 .. 90;
property MinRows: TPDF417_Rows;
```

Description:

The property specifies the minimum number of stacked rows for the CC-C 2D component of an EAN.UCC composite barcode symbol. It can be one of values from 3 to 90. They are defined in the pPDF417Custom unit. The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology. It's a variant of PDF417 symbol which begins with PDF417 codeword 920.

The minimum number of stacked rows together with the minimum number of columns (specified by the MinColumns property) indicates the minimum symbol size for the CC-C 2D component of the EAN.UCC composite barcode symbol. And the maximum

number of stacked rows (specified by the MaxRows property) together with the maximum number of columns (specified by the MaxColumns property) indicates the maximum symbol size for it. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value greater than the MaxRows property's value, the MaxRows property will be automatically changed to equal to this property's value. In other words, the MinRows property's value is always less than or equal to the MaxRows property's value.

See also the "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.95 MinRows

(TBarcode2D_Code16K)

Specifies the minimum number of stacked rows for a Code 16K barcode symbol.

Syntax:

```
type
  { Defined in the pCode16K unit }
  TCode16K_Rows = 2..16;
property MinRows: TCode16K_Rows;
```

Description:

The property specifies the minimum number of stacked rows for a Code 16K barcode symbol. It can be one of values from 2 to 16. They are defined in the pCode16K unit.

The smallest number of stacked rows that accommodates the barcode text will be automatically selected between the minimum number of stacked rows (specified by this property) and the maximum number of stacked rows (that's specified by the MaxRows property).

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value greater than the MaxRows property's value, the MaxRows property will be automatically changed to equal to this property's value. In other words, the MinRows property's value is always less than or equal to the MaxRows property's value.

See also the "Symbol sizes" section in the "TBarcode2D_Code16K" article.

A.1.96 MinRows

(TBarcode2D_PDF417)

Specifies the minimum number of stacked rows for a PDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pPDF417Custom unit }
```

```
  TPDF417_Rows = 3 .. 90;
property MinRows: TPDF417_Rows;
```

Description:

The property specifies the minimum number of stacked rows for a PDF417 barcode symbol. It can be one of values from 3 to 90. They are defined in the pPDF417Custom unit.

The minimum number of stacked rows together with the minimum number of columns (specified by the MinColumns property) indicates a minimum symbol size for a PDF417 barcode symbol. And the maximum number of stacked rows (specified by the MaxRows property) together with the maximum number of columns (specified by the MaxColumns property) indicates the maximum symbol size for the PDF417 barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The CurrentRows property can be used to get the factual number of stacked rows.

Note, if the property is set to a value greater than the MaxRows property's value, the MaxRows property will be automatically changed to equal to this property's value. In other words, the MinRows property's value is always less than or equal to the MaxRows property's value.

See also the "Symbol size" section in the "TBarcode2D_PDF417" article.

A.1.97 MinSegments

(TBarcode2D_CompactMatrix)

Specifies the minimum number of segments for a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_Segments = 1..32;
property MinSegments: TCompactMatrix_Segments;
```

Description:

In horizontal orientation, each Compact Matrix symbol consists of an array of segments with a minimum of 1 segment (maximum 32 segments). The property specifies the minimum number of segments for a Compact Matrix barcode symbol. It can be one of values from 1 to 32, they are defined in the pCompactMatrix unit.

The minimum number of segments together with the minimum version (specified by the MinVersion property) indicates the minimum symbol size for a Compact Matrix barcode symbol. And the maximum number of segments (specified by the MaxSegments property) together with the maximum version (specified by the MaxVersion property) indicates the maximum symbol size for the Compact Matrix barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The CurrentSegments property can be used to get the factual number of segments.

Note, if the property is set to a value greater than the MaxSegments property's value, the MaxSegments property will be automatically changed to equal to this property's value. In other words, the MinSegments property's value is always less than or equal to the MaxSegments property's value.

See also the "Symbol sizes" section in the " TBarcode2D_CompactMatrix" article.

A.1.98 MinSize

(TBarcode2D_AztecCode)

Specifies the minimum symbol size for an Aztec Code barcode symbol.

Syntax:

```
type
  { Defined in the pAztecCode unit }
  TAztecCode_Size = (azSize_15Compact, azSize_19, azSize_19Compact, azSize_23,
    azSize_23Compact, azSize_27, azSize_27Compact, azSize_31, azSize_37, azSize_41,
    azSize_45, azSize_49, azSize_53, azSize_57, azSize_61, azSize_67, azSize_71,
    azSize_75, azSize_79, azSize_83, azSize_87, azSize_91, azSize_95, azSize_101,
    azSize_105, azSize_109, azSize_113, azSize_117, azSize_121, azSize_125, azSize_131,
    azSize_135, azSize_139, azSize_143, azSize_147, azSize_151);
property MinSize: TAztecCode_Size;
```

Description:

The property specifies the minimum symbol size for an Aztec Code barcode symbol. It can be one of values from azSize_15Compact to azSize_151. These values denotation the symbol format, number of rows and columns in every symbol size. For example, the symbol size value azSize_19Compact denotations the Aztec Code symbol size is 19 rows by 19 columns, in compact format. These symbol size values are defined in the pAztecCode unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by this property and the maximum symbol size that's specified by the MaxSize property.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value greater than the MaxSize property's value, the MaxSize property will be automatically changed to equal to this property's value. In other words, the MinSize property's value is always less than or equal to the MaxSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_AztecCode" article.

A.1.99 MinSize

(TBarcode2D_DataMatrix)

Specifies the minimum symbol size for a Data Matrix (ECC 000 - 140) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrix unit }
  TDataMatrix_Size = (dmSize_09_09, dmSize_11_11, dmSize_13_13, dmSize_15_15,
    dmSize_17_17, dmSize_19_19, dmSize_21_21, dmSize_23_23, dmSize_25_25, dmSize_27_27,
    dmSize_29_29, dmSize_31_31, dmSize_33_33, dmSize_35_35, dmSize_37_37, dmSize_39_39,
    dmSize_41_41, dmSize_43_43, dmSize_45_45, dmSize_47_47, dmSize_49_49);
property MinSize: TDataMatrix_Size;
```

Description:

The property specifies the minimum symbol size for a Data Matrix (ECC 000 - 140)barcode symbol. It can be one of values from dmSize_09_09 to dmSize_49_49. These values denotation the number of rows and columns in every symbol size. For example, the symbol size value dmSize_29_29 denotations the Data Matrix (ECC 000 - 140)symbol size is 29 rows by 29 columns. These symbol size values are defined in the pDataMatrix unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by this property and the maximum symbol size that's specified by the MaxSize property.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value greater than the MaxSize property's value, the MaxSize property will be automatically changed to equal to this property's value. In other words, the MinSize property's value is always less than or equal to the MaxSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_DataMatrix" article.

A.1.100 MinSize

(TBarcode2D_DataMatrixECC200)

Specifies the minimum symbol size for a Data Matrix (ECC 200) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrixEcc200 unit }
  TDataMatrixEcc200_Size = (dmSize_10_10, dmSize_12_12, dmSize_14_14, dmSize_16_16,
    dmSize_18_18, dmSize_20_20, dmSize_22_22, dmSize_24_24, dmSize_26_26, dmSize_32_32,
    dmSize_36_36, dmSize_40_40, dmSize_44_44, dmSize_48_48, dmSize_52_52, dmSize_64_64,
    dmSize_72_72, dmSize_80_80, dmSize_88_88, dmSize_96_96, dmSize_104_104,
    dmSize_120_120, dmSize_132_132, dmSize_144_144, dmSize_8_18, dmSize_8_32,
    dmSize_12_26, dmSize_12_36, dmSize_16_36, dmSize_16_48);
property MinSize: TDataMatrixEcc200_Size;
```

Description:

The property specifies the minimum symbol size for a Data Matrix (ECC 200)barcode symbol. It can be one of values from dmSize_10_10 to dmSize_16_48. These values denotation the number of rows and columns in every symbol size. For example, the symbol size value dmSize_8_18 denotations the Data Matrix (ECC 200)symbol size is 8 rows by 18 columns. These symbol size values are defined in the pDataMatrixEcc200 unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by this property and the maximum symbol size that's specified by the MaxSize property.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value greater than the MaxSize property's value, the MaxSize property will be automatically changed to equal to this property's value. In other words, the MinSize property's value is always less than or equal to the MaxSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_DataMatrixECC200" article.

A.1.101 MinSize

(TBarcode2D_MicroPDF417)

Specifies the minimum symbol size for an MicroPDF417 barcode symbol.

Syntax:

```
type
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_Size = (mpSize_1_11, mpSize_1_14, mpSize_1_17, mpSize_1_20, mpSize_1_24,
    mpSize_1_28, mpSize_2_8, mpSize_2_11, mpSize_2_14, mpSize_2_17, mpSize_2_20,
    mpSize_2_23, mpSize_2_26, mpSize_3_6, mpSize_3_8, mpSize_3_10, mpSize_3_12,
    mpSize_3_15, mpSize_3_20, mpSize_3_26, mpSize_3_32, mpSize_3_38, mpSize_3_44,
    mpSize_4_4, mpSize_4_6, mpSize_4_8, mpSize_4_10, mpSize_4_12, mpSize_4_15,
    mpSize_4_20, mpSize_4_26, mpSize_4_32, mpSize_4_38, mpSize_4_44);
property MinSize: TMicroPDF417_Size;
```

Description:

The property specifies the minimum symbol size for a MicroPDF417 barcode symbol. It can be one of values from mpSize_1_11 to mpSize_4_44. These values denotation the number of columns and rows in every symbol size. For example, the symbol size value mpSize_3_12 denotations the MicroPDF417 symbol size is 12 stacked rows by 3 columns. These symbol size values are defined in the pMicroPDF417 unit.

The smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size specified by this property and the maximum symbol size that's specified by the MaxSize property.

The CurrentSize property can be used to get the factual symbol size.

Note, if the property is set to a value greater than the MaxSize property's value, the MaxSize property will be automatically changed to equal to this property's value. In other words, the MinSize property's value is always less than or equal to the MaxSize property's value.

See also the "Symbol sizes" section in the "TBarcode2D_MicroPDF417" article.

A.1.102 MinVersion

(TBarcode2D_CompactMatrix)

Specifies the minimum version for a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_Version = 1..32;
property MinVersion: TCompactMatrix_Version;
```

Description:

There are 32 vertical sizes of Compact Matrix symbol, referred to as version 1 to 32, in increasing order of symbol height and data capacity. The property specifies the minimum version for a Compact Matrix barcode symbol. It can be one of values from 1 to 32. They are defined in the pCompactMatrix unit.

The minimum version together with the minimum number of segments (specified by the MinSegments property) indicates the

minimum symbol size for a Compact Matrix barcode symbol. And the maximum version (specified by the MaxVersion property) together with the maximum number of segments (specified by the MaxSegments property) indicates the maximum symbol size for the Compact Matrix barcode symbol. Based on the stretch order (specified by the StretchOrder property), the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The CurrentVersion property can be used to get the factual version.

Note, if the property is set to a value greater than the MaxVersion property's value, the MaxVersion property will be automatically changed to equal to this property's value. In other words, the MinVersion property's value is always less than or equal to the MaxVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_CompactMatrix" article.

A.1.103 MinVersion

(TBarcode2D_GridMatrix)

Specifies the minimum symbol version (symbol size) for a Grid Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pGridMatrix unit }
  TGridMatrix_Version = 1..13;
property MinVersion: TGridMatrix_Version;
```

Description:

There are thirteen sizes of Grid Matrix symbol, referred to as version 1 to 13, in increasing order of size and data capacity. The property specifies the minimum symbol version (symbol size) for a Grid Matrix barcode symbol. It can be one of values from 1 to 13. They are defined in the pGridMatrix unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by this property and the maximum symbol version (symbol size) that's specified by the MaxVersion property.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value greater than the MaxVersion property's value, the MaxVersion property will be automatically changed to equal to this property's value. In other words, the MinVersion property's value is always less than or equal to the MaxVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_GridMatrix" article.

A.1.104 MinVersion

(TBarcode2D_HanXinCode)

Specifies the minimum symbol version (symbol size) for a Han Xin Code barcode symbol.

Syntax:

```
type
  { Defined in the pHanXinCode unit }
  THanXinCode_Version = 1..84;
property MinVersion: THanXinCode_Version;
```

Description:

There are eighty-four sizes of Han Xin Code symbols, referred to as version 1 to 84, in increasing order of size and data capacity. The property specifies the minimum symbol version (symbol size) for a Han Xin Code barcode symbol. It can be one of values from 1 to 84. They are defined in the pHanXinCode unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by this property and the maximum symbol version (symbol size) that's specified by the MaxVersion property.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value greater than the MaxVersion property's value, the MaxVersion property will be automatically changed to equal to this property's value. In other words, the MinVersion property's value is always less than or equal to the MaxVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_HanXinCode" article.

A.1.105 MinVersion

(TBarcode2D_MicroQRCode)

Specifies the minimum symbol version (symbol size) for a Micro QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pMicroQRCode unit }
  TMicroQRCode_Version = 1..4;
property MinVersion: TMicroQRCode_Version;
```

Description:

The property specifies the minimum symbol version (symbol size) for a Micro QR Code barcode symbol. It can be one of values from 1 to 4, they are defined in the pMicroQRCode unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by this property and the maximum symbol version (symbol size) that's specified by the MaxVersion property.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value greater than the MaxVersion property's value, the MaxVersion property will be automatically changed to equal to this property's value. In other words, the MinVersion property's value is always less than or equal to the MaxVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_MicroQRCode" article.

A.1.106 MinVersion

(TBarcode2D_QRCode)

Specifies the minimum symbol version (symbol size) for a QR Code barcode symbol.

Syntax:

```
type
  { Defined in the pQRCode unit }
  TQRCode_Version = 1..40;
property MinVersion: TQRCode_Version;
```

Description:

The property specifies the minimum symbol version (symbol size) for a QR Code barcode symbol. It can be one of values from 1 to 40. They are defined in the pQRCode unit.

The smallest symbol version (symbol size) that accommodates the barcode text will be automatically selected between the minimum symbol version (symbol size) specified by this property and the maximum symbol version (symbol size) that's specified by the MaxVersion property.

The CurrentVersion property can be used to get the factual symbol version (symbol size).

Note, if the property is set to a value greater than the MaxVersion property's value, the MaxVersion property will be automatically changed to equal to this property's value. In other words, the MinVersion property's value is always less than or equal to the MaxVersion property's value.

See also the "Symbol sizes" section in the "TBarcode2D_QRCode" article.

A.1.107 Mirrored

(TBarcode2D_AztecCode, TBarcode2D_DataMatrix, TBarcode2D_QRCode, etc.)

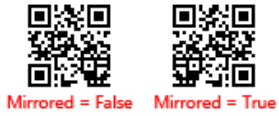
Specifies whether the barcode symbol is reversed right to left (horizontal mirror reversal).

Syntax:

```
property Mirrored: Boolean;
```

Description:

The property specifies whether the barcode symbol is reversed right to left (horizontal mirror reversal). If barcode symbols are obtained using a reflected optical path, a reversed scan direction or from behind through a clear substrate, you can set the Mirror property to true, so that the barcode symbols are easily autodiscriminated and decode with a standard reader. See diagram:



If the leading, trailing, top, and bottom quiet zones are represented (the ShowQuietZone property is set to true), they will not be included in the barcode symbol, and will not be reversed. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

If the barcode symbol is represented using non-left-to-right orientation (the Orienation property isn't equal to boLeftRight, or the angle parameter isn't equal to zero degree in the DrawTo or the Print methods), the left to right reversal operation will be performed firstly, then the rotation operation. See diagram:

A.1.108 Mode

(TBarcode2D_MaxiCode)

Specifies which mode of MaxiCode symbol will be used to generate the MaxiCode barcode symbol.

Syntax:

```
type
  { Defined in the pMaxiCode unit }
  TMaxiCode_Mode = 2..6;
property Mode: TMaxiCode_Mode;
```

Description:

The property specifies which mode of MaxiCode symbol will be used to generate the MaxiCode barcode symbol. It can be one of these values (defined in the pMaxiCode unit):

- **2**: Indicates to use the mode 2. It encodes a structured carrier message with a numeric postal code, and an optional secondary message. A numeric postal code, a country code, and a class of service code assigned by the carrier are included in the structured carrier message. The symbol employs enhanced error correction for the structured carrier message and standard error correction for the secondary message.
- **3**: Indicates to use the mode 3. It encodes a structured carrier message with an alphanumeric postal code, and an optional secondary message. An alphanumeric postal code, a country code, and a class of service code assigned by the carrier are included in the structured carrier message. The symbol employs enhanced error correction for the structured carrier message and standard error correction for the secondary message.

  The mode 3 is similar to mode 2, but the postal code field encodes 6 alphanumeric characters.
- **4**: Indicates to use the mode 4. It encodes an unformatted message. The barcode message is divided into a primary message and a secondary message internally, the symbol employs enhanced error correction for the primary message and standard error correction for the secondary message.
- **5**: Indicates to use the mode 5. It encodes an unformatted message. The barcode message is divided into a primary message and a secondary message internally, the symbol employs enhanced error correction for both the primary and secondary messages.

The mode 5 is similar to mode 4, but it employs enhanced error correction for secondary messages.

- **6**: Indicates to use the mode 6. It encodes a message used to program the reader system. The barcode message is divided into a primary message and a secondary message internally, the symbol employs enhanced error correction for the primary message and standard error correction for the secondary message.

You can set the AutoMode property to true, in order to automatically select the suitable mode depending on the barcode text and the value of this property. And use the CurrentMode property to get the factual mode. See also the "AutoMode" property.

See also the "Modes" sections in the "TBarcode2D_MaxiCode" article.

A.1.109 Module

Specifies the module size in pixels.

Syntax:

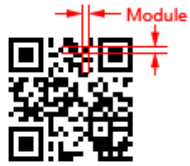
```
property Module: Integer;
```

Description:

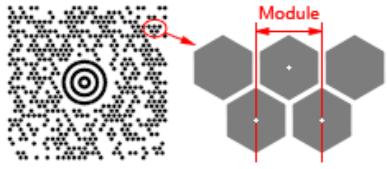
The property specifies the module size in pixels in the horizontal direction.

For the **Matrix** 2D barcode symbologies, the module is single cell in the matrix symbol used to encode one bit data.

In the TBarcode2D_AztecCode, TBarcode2D_AztecRunes, TBarcode2D_CompactMatrix, TBarcode2D_DataMatrix, TBarcode2D_DataMatrixECC200, TBarcode2D_GridMatrix, TBarcode2D_HanXinCode, TBarcode2D_QRCode, and TBarcode2D_MicroQRCode barcode symbologies, the module is a suqare shape, the property specifies the module width and height in pixels. See diagram:



In the TBarcode2D_MaxiCode symbology, the module is a regular hexagonal shape. The property specifies the horizonal width of a module, in pixels, including spacing between modules. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



For **Stacked** 2D barcode symbology and **Linear** 1D barcode symbologies, including TBarcode2D_Code16K, TBarcode2D_PDF417, TBarcode2D_MicroPDF417, TBarcode2D_CCA, TBarcode2D_CCB, TBarcode2D_CCC, TBarcode2D_RSS14, TBarcode2D_RSSLimited, and TBarcode2D_RSSExpanded symbologies, the module is the thinest bar (or space) in the barcode symbol. The property specifies the module width in pixels. See diagram:

Note, for the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the property specifies the module width for entire EAN.UCC composite symbol (including both 2D and linear symbols), in pixels. See diagram:

A.1.110 Options

(TBarcode2D_MicroPDF417)

The Options property is an advanced feature which allows low level control over data encoding for a MicroPDF417 symbol.

Syntax:

```
type
  { Defined in the pPDF417Com unit }
  TPDF417_Option = (poIgnoreShiftBeforeECI, poFirst903TextAlphaLatch,
    poFirst904TextMixedLatch, po906TextAlphaLatch, po907TextAlphaLatch,
    po908TextAlphaLatch, po910TextAlphaLatch, po912TextAlphaLatch, po914TextAlphaLatch,
    po915TextAlphaLatch, poFirstFNC1MatchAI01, poMicroPDF417Explicit901);
  { Defined in the pPDF417Com unit }
  TPDF417_Options = set of TPDF417_Option;
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_Options = TPDF417_Options;
property Options: TMicroPDF417_Options;
```

Description:

For some reader, the encoding algorithm and meanings of some function codewords don't conform with the MicroPDF417 specification. The property can be used to change the encoding algorithm and meanings of these fucntion codewords, in order to match the reader.

The following list describes the values that can be included in the Options property (defined in the pPDF417Com unit):

- **poIgnoreShiftBeforeECI**: An ECI escape sequence may be placed anywhere within Text Compaction mode. Normally, the sub-mode invoked immediately prior to the ECI escape sequence is preserved for the encodation immediately after it. Thus, sub-mode latches and shifts are preserved across an ECI escape sequence; and thus a sub-mode shift immediately before an ECI escape sequence is not ignored. If the value is included in the Options property, the sub-mode shift immediately before an ECI escape sequence is ignored, but a sub-mode latch immediately before an ECI escape sequence is never ignored.
- **poFirst903TextAlphaLatch**: If the function codeword 903 is placed at beginning of barcode text, it represents a leading FNC1 character in First position mode. Also, it normally implies a mode latch to Mixed sub-mode of Text Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text Compaction mode.
- **poFirst904TextMixedLatch**: If the function codeword 904 is placed at beginning of barcode text, it represents a leading FNC1 character in First position mode. Also, it normally implies a mode latch to Numeric Compaction mode, the value indicates that it implies a mode latch to Mixed sub-mode of Text Compaction mode.
- **po906TextAlphaLatch**: The function codeword 906 represents a leading FNC1 character in First position mode. Also, it

normally implies a mode latch to Mixed sub-mode of Text Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text Compaction mode.

- **po907TextAlphaLatch**: The function codeword 907 represents a leading FNC1 character in First position mode. Also, it normally implies a mode latch to Numeric Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text Compaction mode.
- **po908TextAlphaLatch**: The function codeword 908 followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) represents a leading FNC1 character in "Second position" mode. Also, it normally implies a mode latch to Mixed sub-mode of Text Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.
- **po910TextAlphaLatch**: The function codeword 910 indicates that the MicroPDF417 symbol's data output shall conform with the Code128 specification. Also, it normally implies a mode latch to Mixed sub-mode of Text compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.
- **po912TextAlphaLatch**: Normally, the function codeword 912 represents a leading FNC1 character in First position mode and implies a mode latch to Numeric Compaction mode. Also, it encodes that the data begins with a 6-digit date field (Application Identifier is 11, 13, 15, or 17), and this date field may be followed by an implied Application Identifier 10 or 21 as well. The value indicates that it only implies a mode latch to Alpha sub-mode of Text compaction mode.
- **po914TextAlphaLatch**: The function codeword 914 represents a leading FNC1 character in First position mode and indicates that the data begins with an implied Application Identifier is 10. Also, it normally implies a mode latch to Numeric compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.

  If the value isn't included in Options property, the function codeword 914 shall be followed by at least 1 numeric character.
- **po915TextAlphaLatch**: The function codeword 915 represents a leading FNC1 character in First position mode and indicates that the data begins with an implied Application Identifier is 21. Also, it normally implies a mode latch to Numeric compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.

  If the value isn't included in Options property, the function codeword 915 shall be followed by at least 1 numeric character.
- **poFirstFNC1MatchAI01**: When the poFirstFNC1MatchAI01 is included in the value of Options property, if the "\f" followed by a SSC-14 number (including Application Identifier 01 and 13-digit data, the checkdigit isn't required) is placed at beginnig of barcode, the function codeword 905 will be selected to encode the FNC1 character in order to reduce the symbol size (the leading Application Identifier 01 is not encoded in the SSC-14 number). Otherwise, the function codeword 905 will be selected to encode the FNC1 character.
- **poMicroPDF417Explicit901**: In the specification of MicroPDF417, the initial compaction mode in effect at the start of each symbol shall always be Byte compaction mode, but some readers use the other compaction mode (e.g. Alpha sub-mode of Text compaction) as the initial compaction mode. The value indicates to explicitly insert a mode latch to Byte compaction mode into beginning of symbol data, in order to work with these reader.

See also the "Character set", "Escape sequences", and "Extended Channel Interpretation (ECI)" sections in the "TBarcode2D_MicroPDF417" article.

A.1.111 Options

(TBarcode2D_PDF417)

The Options property is an advanced feature which allows low level control over data encoding for a PDF417 symbol.

Syntax:

```
type
```

```
  { Defined in the pPDF417Com unit }
  TPDF417_Option = (poIgnoreShiftBeforeECI, poFirst903TextAlphaLatch,
    poFirst904TextMixedLatch, po906TextAlphaLatch, po907TextAlphaLatch,
    po908TextAlphaLatch, po910TextAlphaLatch, po912TextAlphaLatch, po914TextAlphaLatch,
    po915TextAlphaLatch, poFirstFNC1MatchAI01, poMicroPDF417Explicit901);
  { Defined in the pPDF417Com unit }
  TPDF417_Options = set of TPDF417_Option;

property Options: TPDF417_Options;
```

Description:

For some reader, the encoding algorithm and meanings of some function codewords don't conform with the PDF417 specification. The property can be used to change the encoding algorithm and meanings of these fucntion codewords, in order to match the reader.

The following list describes the values that can be included in the Options property (defined in the pPDF417Com unit):

- **poIgnoreShiftBeforeECI**: An ECI escape sequence may be placed anywhere within Text Compaction mode. Normally, the sub-mode invoked immediately prior to the ECI escape sequence is preserved for the encodation immediately after it. Thus, sub-mode latches and shifts are preserved across an ECI escape sequence; and thus a sub-mode shift immediately before an ECI escape sequence is not ignored. If the value is included in the Options property, the sub-mode shift immediately before an ECI escape sequence is ignored, but a sub-mode latch immediately before an ECI escape sequence is never ignored.
- **poFirst903TextAlphaLatch**: If the function codeword 903 is placed at beginning of barcode text, it represents a leading FNC1 character in First position mode. Also, it normally implies a mode latch to Mixed sub-mode of Text Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text Compaction mode.
- **poFirst904TextMixedLatch**: If the function codeword 904 is placed at beginning of barcode text, it represents a leading FNC1 character in First position mode. Also, it normally implies a mode latch to Numeric Compaction mode, the value indicates that it implies a mode latch to Mixed sub-mode of Text Compaction mode.
- **po906TextAlphaLatch**: The function codeword 906 represents a leading FNC1 character in First position mode. Also, it normally implies a mode latch to Mixed sub-mode of Text Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text Compaction mode.
- **po907TextAlphaLatch**: The function codeword 907 represents a leading FNC1 character in First position mode. Also, it normally implies a mode latch to Numeric Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text Compaction mode.
- **po908TextAlphaLatch**: The function codeword 908 followed by an application indicator (single Latin alphabetic character from the set "a" to "z" and "A" to "Z", or a two-digit number) represents a leading FNC1 character in "Second position" mode. Also, it normally implies a mode latch to Mixed sub-mode of Text Compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.
- **po910TextAlphaLatch**: The function codeword 910 indicates that the PDF417 symbol's data output shall conform with the Code128 specification. Also, it normally implies a mode latch to Mixed sub-mode of Text compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.
- **po912TextAlphaLatch**: Normally, the function codeword 912 represents a leading FNC1 character in First position mode and implies a mode latch to Numeric Compaction mode. Also, it encodes that the data begins with a 6-digit date field (Application Identifier is 11, 13, 15, or 17), and this date field may be followed by an implied Application Identifier 10 or 21 as well. The value indicates that it only implies a mode latch to Alpha sub-mode of Text compaction mode.
- **po914TextAlphaLatch**: The function codeword 914 represents a leading FNC1 character in First position mode and indicates that the data begins with an implied Application Identifier is 10. Also, it normally implies a mode latch to Numeric compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.

  If the value isn't included in Options property, the function codeword 914 shall be followed by at least 1 numeric character.

- **po915TextAlphaLatch**: The function codeword 915 represents a leading FNC1 character in First position mode and indicates that the data begins with an implied Application Identifier is 21. Also, it normally implies a mode latch to Numeric compaction mode, the value indicates that it implies a mode latch to Alpha sub-mode of Text compaction mode.

  If the value isn't included in Options property, the function codeword 915 shall be followed by at least 1 numeric character.

- **poFirstFNC1MatchAI01**: When the poFirstFNC1MatchAI01 is included in the value of Options property, if the "\f" followed by a SSC-14 number (including Application Identifier 01 and 13-digit data, the checkdigit isn't required) is placed at beginnig of barcode, the function codeword 905 will be selected to encode the FNC1 character in order to reduce the symbol size (the leading Application Identifier 01 is not encoded in the SSC-14 number). Otherwise, the function codeword 905 will be selected to encode the FNC1 character.

- **poMicroPDF417Explicit901**: The value isn't used in the TBarcode2D_PDF417 component.

See also the "Character set", "Escape sequences", and "Extended Channel Interpretation (ECI)" sections in the "TBarcode2D_PDF417" article.

A.1.112 Orientation

Controls the orientation of the barcode symbol.

Syntax:

```
type
  { Defined in the pBarcode2D unit }
  TBarcodeOrientation = (boLeftRight, boRightLeft, boTopBottom, boBottomTop);
property Orientation: TBarcodeOrientation;
```

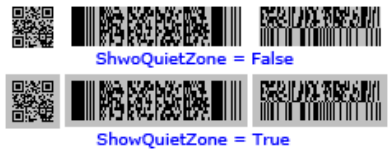
Description:

Specifies the direction of the barcode symbol. The barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone property about whether or not the quiet zones will be represented) will be rotated based on the property value. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules.

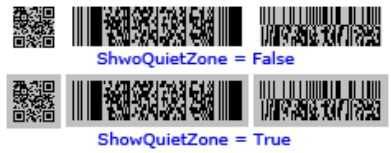
For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety.

This property can be one of these values (defined in the pBarcode2D unit):

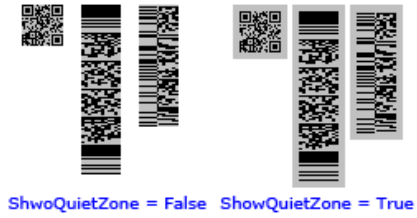
- **boLeftRight**: Left to right horizontal direction (rotates the barcode symbol 0 degrees counter-clockwise). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



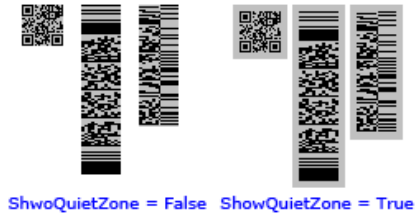
- **boRightLeft**: Right to left horizontal direction (rotates the barcode symbol 180 degrees counter-clockwise). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- **boTopBottom**: Top to bottom vertical direction (rotates the barcode symbol 270 degrees counter-clockwise). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- **boBottomTop**: Bottom to top vertical direction (rotates the barcode symbol 90 degrees counter-clockwise). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



A.1.113 Placement

(TBarcode2D_CompactMatrix)

Specifies how to place symbol characters when encoding the barcode text into a Compact Matrix symbol.

Syntax:

```
type
  TCompactMatrix_Placement = (pmSequence, pmInterlacement);
property Placement: TCompactMatrix_Placement;
```

Description:

Compact Matrix symbology has two symbol charcacters placement methods as sequence and interlacement. The property specifies which placement method will be used to generate the Compact Matrix symbol. It can be one of values pmSequence and pmInterlacement, corresponding to the symbol charcacters placement methods sequence and interlacement.

A.1.114 ReviseVersion5

(TBarcode2D_HanXinCode)

Specifies whether to revise the position of alignment pattern for a Han Xin Code version 5 symbol.

Syntax:

```
property ReviseVersion5: Boolean;
```

Description:

The property specifies whether to revise the position of alignment pattern for a Han Xin Code version 5 symbol. By default, it's set to false, if the symbol can be read by your reader, please don't change it.

See also the "Symbol sizes" section in the "TBarcode2D_HanXinCode" article.

A.1.115 RowHeight

(TBarcode2D_CCA, TBarcode2D_CCB)

Specifies the height of each stacked row for the CC-A or CC-B 2D component of an EAN.UCC composite symbol, in modules.

Syntax:

```
type
  { Defined in the pCCCustom unit }
  TCCX_RowHeight = 1 .. 99;
property RowHeight: TCCX_RowHeight;
```

Description:

The property specifies the height of each stacked row for the CC-A or CC-B 2D component of an EAN.UCC composite symbol, in modules. It should be set to an integer between 1 and 99 (including the boundaries, defined in the pCCCustom unit). In general, it shall be a minimum of 2 modules. The CC-A and CC-B is two of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

See diagram:

A.1.116 RowHeight

(TBarcode2D_Code16K)

Specifies the height of each stacked row for a Code 16K stacked barcode symbol, in modules.

Syntax:

```
property RowHeight: Integer;
```

Description:

The property specifies the height of each stacked row for a Code 16K barcode symbol, in modules. It should be set to an integer greater than or equal to 8, and the height of separator bar isn't included. If the property is set to a value less than 8, the operation will be ignored.

See diagram:

A.1.117 RowHeight

(TBarcode2D_MicroPDF417)

Specifies the height of each stacked row for a MicroPDF417 stacked barcode symbol, in modules.

Syntax:

```
type
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_RowHeight = 1 .. 99;
property RowHeight: TMicroPDF417_RowHeight;
```

Description:

The property specifies the height of each stacked row for a MicroPDF417 barcode symbol, in modules. It should be set to an integer between 1 and 99 (including the boundaries, defined in the pMicroPDF417 unit). In general, it shall be a minimum of 2 modules.

See diagram:

A.1.118 RowHeight

(TBarcode2D_PDF417, TBarcode2D_CCC)

Specifies the height of each stacked row for a PDF417 stacked barcode symbol, or for the CC-C 2D component of an EAN.UCC composite barcode symbol, in modules.

Syntax:

```
type
```

```
  { Defined in the pPDF417Custom unit }
  TPDF417_RowHeight = 1 .. 99;
property RowHeight: TPDF417_RowHeight;
```

Description:

The property specifies the height of each stacked row for a PDF417 barcode symbol, or for the CC-C 2D component of an EAN.UCC composite barcode symbol, in modules. The CC-C is one of the three choices for the 2D component in a barcode symbol encoded in EAN.UCC composite barcode symbology.

The property should be set to an integer between 1 and 99 (including the boundaries, defined in the pPDF417Custom unit). In general, it shall be a minimum of 3 modules.

See diagram:

A.1.119 RowHeight

(TBarcode2D_RSSExpanded)

Specifies the height of each stacked row for an RSS Expanded barcode symbol, in modules.

Syntax:

```
property RowHeight: Integer;
```

Description:

For the RSS Expanded (Standard) barcode symbol (the Style property is set to rsStandard), the property specifies the height of the RSS Expanded (Standard) barcode symbol, in modules. See diagram:

For the RSS Expanded Stacked barcode symbol (the Style property is set to rsStacked), the property specifies the height of each stacked row for the RSS Expanded Stacked barcode symbol, in modules. See diagram:

The property should be set to an integer greater than or equal to 34.

A.1.120 RowSymbols

(TBarcode2D_RSSExpanded)

Specifies the number of symbol segments in each row for an RSS Expanded Stacked symbol.

Syntax:

```
property RowSymbols: Integer;
```

Description:

The property specifies the number of symbol segments in each row for an RSS Expanded Stacked symbol (the Style property is set to rsStacked). It's an integer between 2 and 20 (including the boundaries, only even values are valid), the barcode symbol will be stacked in two to eleven rows based on the total numbers of symbol segments and the number of symbol segments in each row.

If the RSS Expanded Stacked symbol is used together with a 4-column CC-A or a 4-column CC-B barcode symbol to create the EAN.UCC composite symbol, the RSS Expanded Stacked symbol shall contain at least four symbol characters within its top row. In other words, the value of RowSymbols property should be greater than or equal to 4.

See also the "Symbol size" section in the "TBarcode2D_RSSExpanded" article.

A.1.121 SeparatorBarHeight

(TBarcode2D_Code16K)

Specifies the height of separator bar for Code 16K barcode symbology, in modules.

Syntax:

```
property SeparatorRowHeight: Integer;
```

Description:

The property specifies the height of separator bar for Code 16K barcode symbology, in modules. It's an integer between 1 and 4 (including the boundaries).

The separator bar is horizontal bar separating two rows of a Code 16K barcode symbol, or abutting the top or bottom of first or last row respectively.

See diagram:

A.1.122 SeparatorRowHeight

(TBarcode2D_RSS14, TBarcode2D_RSSExpanded)

Specifies the height of separator pattern row between symbol rows for RSS-14 (Stacked and Stacked Omnidirectional) and RSS Expanded (Stacked) symbols, in modules.

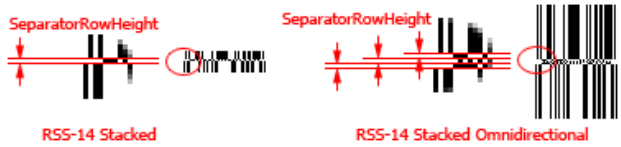
Also, it sepecifies the height of contiguous separator pattern row between the linear and 2D components of an EAN.UCC Composite symbol which use the RSS symbol (including any style of RSS-14, RSS Limited, or any style of RSS Expanded symbol) as its linear component, in modules.

Syntax:

```
property SeparatorRowHeight: Integer;
```

Description:

For the RSS-14 barcode symbols, if the Style property is set to the rsStacked (Stacked style) or rsStackedOmnidirectional (Stacked Omnidirectional style), the property specifies the height of separator pattern row between top row and bottom row, in modules. See diagram:



For the RSS Expanded barcode symbols, if the Style property is set to rsStacked (Stacked style), the property specifies the height of separator pattern row between symbol rows, in modules. See diagram:

If an RSS barcode symbol (including any style of RSS-14, RSS Limited, or any style of RSS Expanded symbol) is used together with an adjacent 2D component to create the EAN.UCC Composite symbol (its Link2D property is set to true), and the contiguous separator pattern is respresented (its Show2DSeparator property is set to true), the property specifies the height of contiguous separator pattern too, in modules.

See diagram:

See also the "Composite symbol" section in the "TBarcode2D_RSS14", "TBarcode2D_RSSLimited", and "TBarcode2D_RSSExpanded" articles.

A.1.123 SeparatorRowHeight

(TBarcode2D_RSSLimited)

Specify the height of contiguous separator pattern row between the linear and 2D components of an EAN.UCC Composite symbol which use the RSS Limited symbol as its linear component, in modules.

Syntax:

```
property SeparatorRowHeight: Integer;
```

Description:

If an RSS Limited barcode symbol is used together with an adjacent 2D component to create the EAN.UCC Composite symbol (its Link2D property is set to true), and the contiguous separator pattern is respresented (its Show2DSeparator property is set to true), the property specifies the height of contiguous separator pattern, in modules.

See diagram:

See also the "Composite symbol" section in the "TBarcode2D_RSSLimited" article.

A.1.124 Shape

(TBarcode2D_DataMatrixEcc200)

Specifies which shape of Data Matrix (ECC 200) symbol will be selected to generate the Data Matrix (ECC 200) barcode symbol.

Syntax:

```
type
  { Defined in the pDataMatrixEcc200 unit }
  TDataMatrixEcc200_Shape = (dsSquare, dsRectangle);
property Shape: TDataMatrixEcc200_Shape;
```

Description:

There are two shapes of Data Matrix (ECC 200)symbols, square and rectangle. The property specifies which shape of Data Matrix (ECC 200) symbol will be selected to generate the barcode symbol.

The property can be one of these values (defined in the pDataMatrixEcc200 unit):

- **dsSquare**: Indicates to generate the square symbols. In this case, if the MinSize property value isn't between dmSize_10_10 and dmSize_144_144 then it's default to dmSize_10_10, and if the MaxSize property value isn't between dmSize_10_10 and dmSize_144_144 then it's default to dmSize_144_144.

- **dsRectangle**: Indicates to generate the rectangle symbols. In this case, if the MinSize property value isn't between dmSize_8_18 and dmSize_16_48 then it's default to dmSize_8_18, and if the MaxSize property value isn't between dmSize_8_18 and dmSize_16_48 then it's default to dmSize_16_48.

See also the "Shapes" section in the "TBarcode2D_DataMatrixECC2OO" article.

A.1.125 Show2DSeparator

(TBarcode2D_RSS14, TBarcode2D_RSSLimited, TBarcode2D_Expanded)

Specifies whether to represent a 2D contiguous separator pattern on top of the RSS-14, RSS Limited, or RSS Expanded symbols.

Syntax:

```
property Show2DSeparator: Boolean;
```

Description:

An RSS barcode symbol (including RSS-14, RSS Limited, and RSS Expanded) can be used together with a CC-A or a CC-B 2D component to create the EAN.UCC composite symbol. A contiguous separator pattern is required between the RSS and the 2D components in an EAN.UCC composite symbol. The property specifies whether to represent the contiguous separator pattern on top of the RSS symbol, in order to create EAN.UCC composite symbols using the RSS symbol. See digiam (the color of separator patterns are changed to red in order to accentuate them):

The height of contiguous separator pattern is specified by the SeparatorRowHeight (for RSS-14 and RSS Expanded components) and SeparatorRowHeight (for the RSS Limited component) property, in modules.

Note, in order to represent the contiguous separator pattern, both the Show2DSeparator and the Link2D properties must be set to true.

If you use the RSS symbol as the Linear property's value of a TBarcode2D_CCA or TBarcode2D_CCB 2D barcode component to create the EAN.UCC composite symbol, the Show2DSeparator and the Link2D properties will be set to true automatically.

See also the "Composite symbol" section in the "TBarcode2D_RSS14", "TBarcode2D_RSSLimited", and "TBarcode2D_RSSExpanded" articles.

A.1.126 ShowQuietZone

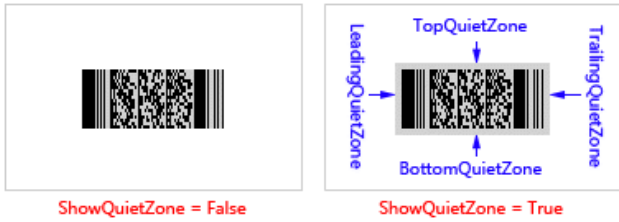
Specifies whether to represent the leading quiet zone, trailing quiet zone, top quiet zone, and bottom quiet zone.

Syntax:

```
property ShowQuietZone: Boolean;
```

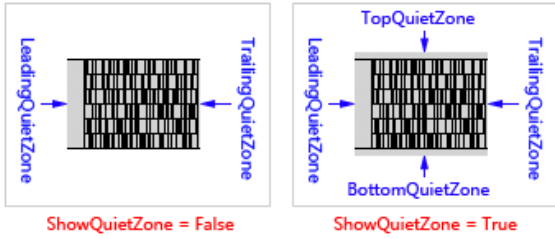
Description:

The property specifies whether to represent the leading quiet zone, trailing quiet zone, top quiet zone, and bottom quiet zone. If the property value is set to true, these quiet zones are represented. Otherwise, they don't be represented. And they are represented using the color specified by the SpaceColor property if the Inversed property is set to false. Otherwise, they are represented using the color specified by the BarColor property. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

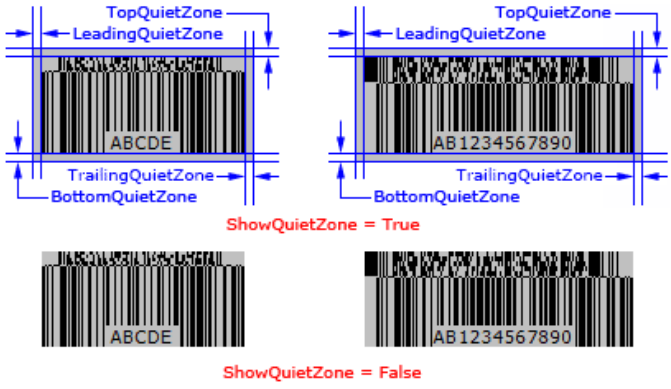


You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules.

For the TBarcode2D_Code16K barcode components, the leading and trailing quiet zones (their size is specified by LeadingQuietZone and TrailingQuietZone properties) will be represented always, even if the ShowQuietZone property value is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

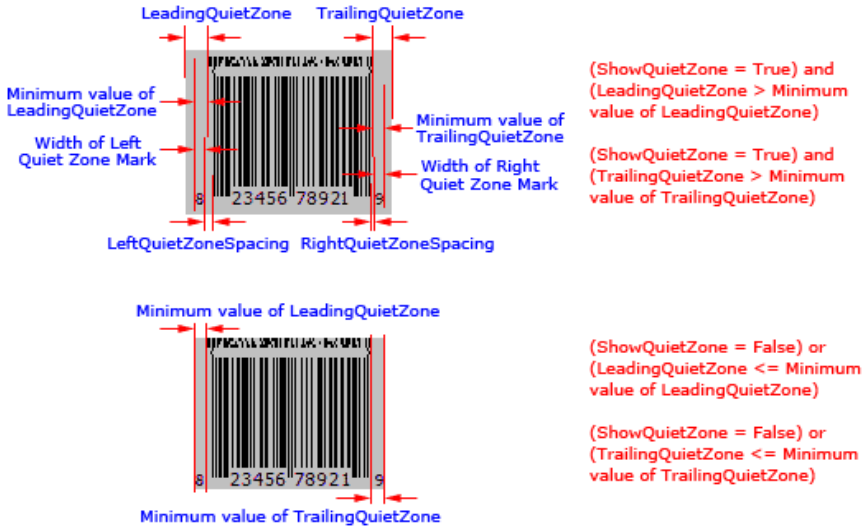


For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And theLeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the CC-A 2D symbol. See diagram (theSpaceColor property value is set to clSilver in order to accentuate the quiet zones):

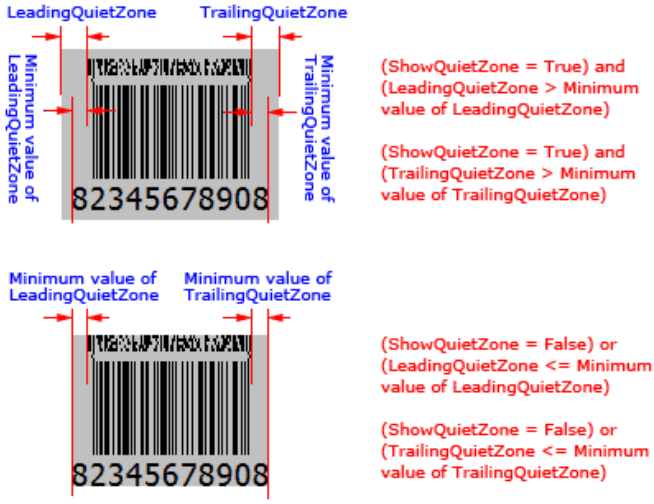


For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone property is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone

property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone property is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone property is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



A.1.127 SpaceColor

Specifies the color for all spaces or light modules in the barcode symbol, By default, it's clWhite.

Syntax:

```
property SpaceColor: TColor;
```

Description:

For the **Matrix** 2D barcode symbologies, including TBarcode2D_AztecCode, TBarcode2D_AztecRunes, TBarcode2D_DataMatrix, TBarcode2D_DataMatrixECC200, TBarcode2D_HanXinCode, TBarcode2D_GridMatrix, TBarcode2D_CompactMatrix, TBarcode2D_QRCode, TBarcode2D_MaxiCode, and TBarcode2D_MicroQRCode, the property specifies the color of every light module in matrix symbol if the Inversed property is set to false, otherwise, it specifies the color of every dark module. The module is single cell in the matrix symbol used to encode one bit data, nominally a suqare shape, in MaxiCode symbology, it's a regular hexagonal shape.

For **Stacked** 2D barcode symbologies and **Linear** 1D barcode symbologies, including TBarcode2D_Code16K, TBarcode2D_PDF417, TBarcode2D_MicroPDF417, TBarcode2D_CCA, TBarcode2D_CCB, TBarcode2D_CCC, TBarcode2D_RSS14, TBarcode2D_RSSLimited, and TBarcode2D_RSSExpanded, the property specifies the color for all spaces in the barcode symbol if the Inversed property is set to false, otherwise, it specifies the color for all bars.

For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the property is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

Also, when the leading quiet zone, trailing quiet zone, top quiet zone, and bottom quiet zone are represented (the ShowQuietZone property value is set to true), the color specified by this SpaceColor property will be used to represend them if the Inversed property is set to false, otherwise, the color specified by the BarColor property will be used.

A.1.128 StartWidth

(TBarcode2D_CompactMatrix)

Specifies the width of start pattern for a Compact Matrix barcode symbol, in modules.

Syntax:

```
property StartWidth: Integer;
```

Description:

The property specifies the width of start pattern for a Compact Matrix barcode symbol, in modules. It should be set to an integer greater than or equal to 2. If the property is set to a value less than 2, an exception will be raised.

See diagram:

A.1.129 StopWidth

(TBarcode2D_CompactMatrix)

Specifies the width of stop pattern for a Compact Matrix barcode symbol, in modules.

Syntax:

```
property StopWidth: Integer;
```

Description:

The property specifies the width of stop pattern for a Compact Matrix barcode symbol, in modules. It should be set to an integer greater than or equal to 2. If the property is set to a value less than 2, an exception will be raised.

See diagram:

A.1.130 Stretch

Specifies whether to reduce/stretch the barcode symbol to specified size.

Syntax:

```
property Stretch: Boolean;
```

Description:

The property specifies whether to reduce/stretch the barcode symbol to fit the size specified by BarcodeWidth and BarcodeHeight properties. The barcode symbol together with its quiet zones (if they are represented) will be reduced/stretched if the property is set to true. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone property to specify whether to represent these quiet zones.

For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be reduced/stretched as an entirety.

- If the property is set to false, the barcode symbol will not be reduced/stretched. The values of BarcodeWidth and BarcodeHeight properties will be ignored.

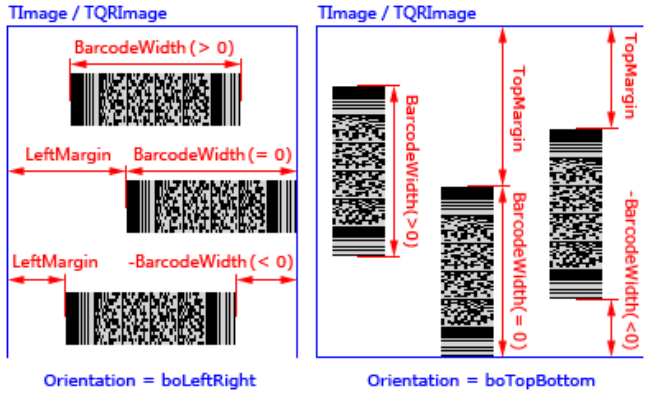
  The barcode symbol width and height will be calculated based on the Module property value.

  You can get the width and height by using the Size method.

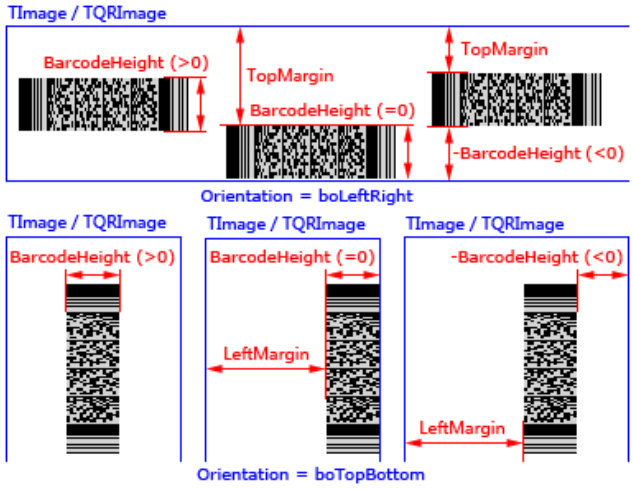
- If the property is set to true, the barcode symbol will be reduced/stretched to fit the size specified by BarcodeWidth and BarcodeHeight properties.

  The BarcodeWidth property specifies the width of barcode symbol. If the property value is less than or equal to zero, it specifies the right margin of the barcode symbol (the Orientation property is set to "boLeftRight" or "boRightLeft"), or the

bottom margin of the barcode symbol (the Orientation property is set to "boTopBottom" or "boBottomTop"). See also the "BarcodeWidth" property. See diagram:



The BarcodeHeight property specifies the height of barcode symbol. If the property value is less than or equal to zero, it specifies the bottom margin of the barcode symbol (the Orientation property is set to "boLeftRight" or "boRightLeft"), or the right margin of the barcode symbol (the Orientation property is set to "boTopBottom" or "boBottomTop"). See also the "BarcodeHeight" property. See diagram:



A.1.131 StretchOrder

(TBarcode2D_CCC)

Specifies the priority order of selecing appropriate symbol size for the CC-C component of an EAN.UCC composite barcode symbol.

Syntax:

```
type
  { Defined in the pCCC unit }
  TCCC_StretchOrder = (soMaxColumnsInLinear, soMaxColumnsOutLinear, soRowColumn,
    soColumnRow);
property StretchOrder: TCCC_StretchOrder;
```

Description:

The minimum number of stacked rows (specified by the MinRows property) and the minimum number of columns (specified by the MinColumns property) indicates the minimum symbol size for the CC-C 2D component of an EAN.UCC composite barcode symbol. And the maximum number of stacked rows (specified by the MaxRows property) together with the maximum number of columns (specified by the MaxColumns property) indicates the maximum symbol size for it. The property specifies a priority order in order to automatically select the symbol size. According to the priority order, the first symbol size (number of rows and columns) that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The property can be one of these values (defined in the pCCC unit):

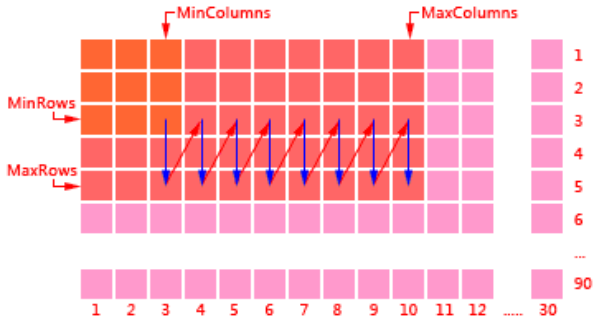
- **soMaxColumnsInLinear**: The maximum number of columns that enables the CC-C 2D component (including those for the start and stop patterns and row indicators) to fit within the lateral boundaries of the linear component (including its quiet zones) will be chosen. If the maximum number of columns is greater than the value of MaxColumns property, the MaxColumns property's value will be used as the final chosen number of columns.

  According to the final chosen number of columns and the error checking and correcting level specified by the ECCLevel property, the minimum number of stacked rows that accommodates the barcode text will be automatically chosen. If the minimum number of stacked rows is less than the value of MinRows property, the MinRows property's value will be used as the final chosen number of stacked rows. If the barcode text cannot be accommodated by the maximum symbol size (specified by the MaxColumns and MaxRows properties), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

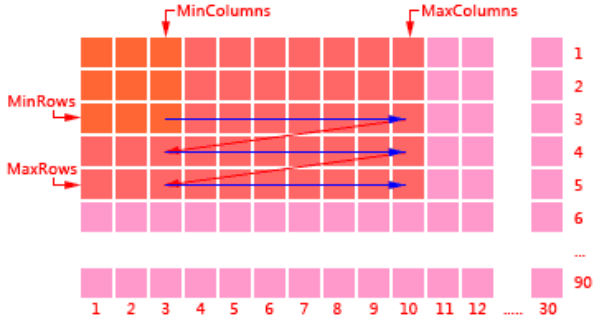
- **soMaxColumnsOutLinear**: Create a wider CC-C symbol that exceedsthe right bound of the trailing quiet zone of the linear component in order to minimize the height of the CC-C 2D component.

  Between the minimum symbol size (specified by the MinColumns and MinRows properties) and the maximum symbol size (specified by the MaxColumns and MaxRows properties), only symbol sizes that their total number of CC-C columns is less than four times the number of stacked rows will be used in order to avoid very large width to height ratios and maintain angular tolerance for single-line scanners. According to the data capacity, in increasing order of recovery capacity, the smallest symbol size that accommodates the barcode text and the error checking and correcting level (specified by the ECCLevel property) will be automatically selected. If the barcode text cannot be accommodated by the maximum symbol size (specified by the MaxColumns and MaxRows properties), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soRowColumn**: If the CC-C symbol that use minimum symbol size (specified by the MinColumns and MinRows properties) cannot accommodate the barcode text, the number of stacked rows will be increased firstly until it reachs the maximum number of stacked rows (specified by the MaxRows property) or the barcode text is accommodated by the symbol. If the number of stacked rows has reached the maximum number of stacked rows (specified by the MaxRows property) but the barcode text cannot yet be accommodated, the number of columns will be increased and the number of stacked rows will be reset to minimum number of stacked rows (specified by the MinRows property). Repeat these until the barcode text is accommodated by the symbol, or not only does the number of stacked rows reach maximum number of stacked rows (specified by the MaxRows property), but also the number of columns reachs maximum number of columns (specified by the MaxColumns property), in this case (the barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur. See diagram:



- **soColumnRow**: If the CC-C symbol that use minimum symbol size (specified by the MinColumns and MinRows properties) cannot accommodate the barcode text, the number of columns will be increased firstly until it reachs the maximum number of columns (specified by the MaxColumns property) or the barcode text is accommodated by the symbol. If the number of columns has reached the maximum number of columns (specified by the MaxColumns property) but the barcode text cannot yet be accommodated, the number of stacked rows will be increased and the number of columns will be reset to minimum number of columns (specified by the MinColumns property). Repeat these until the barcode text is accommodated by the symbol, or not only does the number of columns reach maximum number of columns (specified by the MaxColumns property), but also the number of stacked rows reachs maximum number of stacked rows (specified by the MaxRows property), in this case (the barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur. See diagram:



See also the "Symbol size" section in the "TBarcode2D_CCC" article.

A.1.132 StretchOrder

(TBarcode2D_CompactMatrix)

Specifies the priority order of selecing appropriate symbol size for a Compact Matrix barcode symbol.

Syntax:

```
type
  { Defined in the pCompactMatrix unit }
  TCompactMatrix_StretchOrder = (soVersionSegment, soSegmentVersion, soFixAspect,
    soFixAspectWithQuietZones);
property StretchOrder: TCompactMatrix_StretchOrder;
```

Description:

There are 32 vertical sizes of Compact Matrix symbol, referred to as version 1 to 32, in increasing order of symbol height and data

capacity. In horizontal orientation, each Compact Matrix symbol consists of an array of segments with a minimum of 1 segment (maximum 32 segments).

The minimum version (specified by the MinVersion property) and the minimum number of segments (specified by the MinSegments property) indicates the minimum symbol size for a Compact Matrix barcode symbol. The maximum version (specified by the MaxVersion property) and the maximum number of segments (specified by the MaxSegments property) indicates the maximum symbol size for the Compact Matrix barcode symbol. The property specifies a priority order in order to automatically select the symbol size. According to the priority order, the first symbol size (version and number of segments) that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The property can be one of these values (defined in the pCompactMatrix unit):

- **soVersionSegment**: If the Compact Matrix symbol that use minimum symbol size (specified by the MinVersion and MinSegments properties) cannot accommodate the barcode text, the symbol version will be increased firstly until it reachs the maximum value (specified by the MaxVersion property) or the barcode text is accommodated by the symbol. If the version has reached the maximum value (specified by the MaxVersion property) but the barcode text cannot yet be accommodated, the number of segments will be increased and the version will be reset to minimum value (specified by the MinVersion property). Repeat these until the barcode text is accommodated by the symbol, or not only does the version reach maximum value (specified by the MaxVersion property), but also the number of segments reachs maximum number of segments (specified by the MaxSegments property), in this case (the barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soSegmentVersion**: If the Compact Matrix symbol that use minimum symbol size (specified by the MinVersion and MinSegments properties) cannot accommodate the barcode text, the number of segments will be increased firstly until it reachs the maximum number of segments (specified by the MaxSegments property) or the barcode text is accommodated by the symbol. If the number of segments has reached the maximum number of segments (specified by the MaxSegments property) but the barcode text cannot yet be accommodated, the version will be increased and the number of segments will be reset to minimum number of segments (specified by the MinSegments property). Repeat these until the barcode text is accommodated by the symbol, or not only does the number of segments reach maximum number of segments (specified by the MaxSegments property), but also the version reachs maximum value (specified by the MaxVersion property), in this case (the barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soFixAspect**: An overall aspect ratio is specified by the minimum version (specified by the MinVersion property) and minumum number of segments (specified by the MinSegments property), the ratio is height to width of the symbol excluding quiet zones. Between the minimum symbol size (specified by the MinVersion and MinSegments properties) and the maximum symbol size (specified by the MaxVersion and MaxSegments properties), only symbol sizes that they approximately match the aspect ratio will be used. According to the data capacity, in increasing order of recovery capacity, the smallest symbol size that accommodates the barcode text will be automatically selected. If the barcode text cannot be accommodated by the maximum symbol size (specified by the MaxVersion and MaxSegments properties), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soFixAspectWithQuietZones**: An overall aspect ratio is specified by the minimum version (specified by the MinVersion property) and minumum number of segments (specified by the MinSegments property), the ratio is height to width of the symbol including quiet zones. Between the minimum symbol size (specified by the MinVersion and MinSegments properties) and the maximum symbol size (specified by the MaxVersion and MaxSegments properties), only symbol sizes that they approximately match the aspect ratio will be used. According to the data capacity, in increasing order of recovery capacity, the smallest symbol size that accommodates the barcode text will be automatically selected. If the barcode text cannot be accommodated by the maximum symbol size (specified by the MaxVersion and MaxSegments properties), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

See also the "Symbol sizes" section in the "TBarcode2D_CompactMatrix" article.

A.1.133 StretchOrder

(TBarcode2D_MicroPDF417)

Specifies the priority order of selecing appropriate symbol size for a MicroPDF417 barcode symbol.

Syntax:

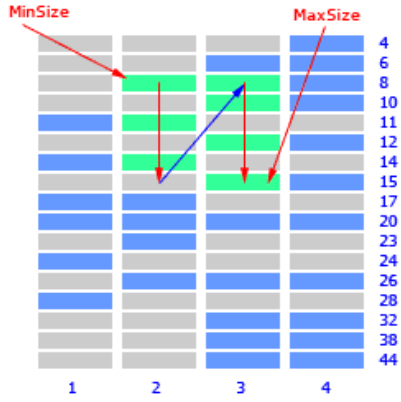
```
type
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_StretchOrder = (soRowColumn, soColumnRow, soRowPrior, soColumnPrior,
    soGlobal_1, soGlobal_2, soGlobal_3, soGlobal_4);
property StretchOrder: TMicroPDF417_StretchOrder;
```

Description:

The property specifies a priority order in order to automatically select the symbol size for a MicroPDF417 barcode symbol. According to the priority order, the first symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size (specified by the MinSize property) and the maximum symbol size (specified by the MaxSize property).

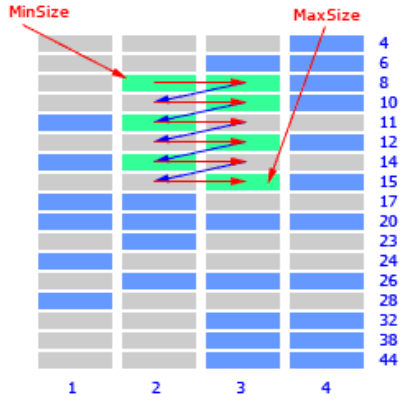
The property can be one of these values (defined in the pMicroPDF417 unit):

- **soRowColumn**: If the MicroPDF417 symbol that use minimum symbol size cannot accommodate the barcode text, the number of rows will be increased firstly until it reachs the maximum number of rows (specified by the MaxSize property) or the barcode text is accommodated by the symbol. If the number of rows has reached the maximum number of rows (specified by the MaxSize property) but the barcode text cannot yet be accommodated, the number of columns will be increased and the number of rows will be reset to minimum number of rows (specified by the MinSize property). Repeat these until the barcode text is accommodated by the symbol, or the symbol size reachs maximum symbol size (specified by the MaxSize property), in this case (tha barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur. See diagram:

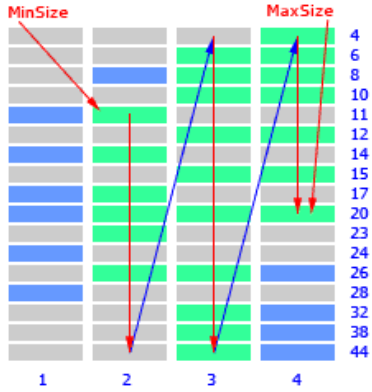


- **soColumnRow**: If the MicroPDF417 symbol that use minimum symbol size cannot accommodate the barcode text, the number of columns will be increased firstly until it reachs the maximum number of columns (specified by the MaxSize property) or the barcode text is accommodated by the symbol. If the number of columns has reached the maximum number of columns (specified by the MaxSize property) but the barcode text cannot yet be accommodated, the number of rows will be increased and the number of columns will be reset to minimum number of columns (specified by the MinSize property). Repeat these until the barcode text is accommodated by the symbol, or the symbol size reachs maximum symbol size (specified by the MaxSize property), in this case (tha barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is

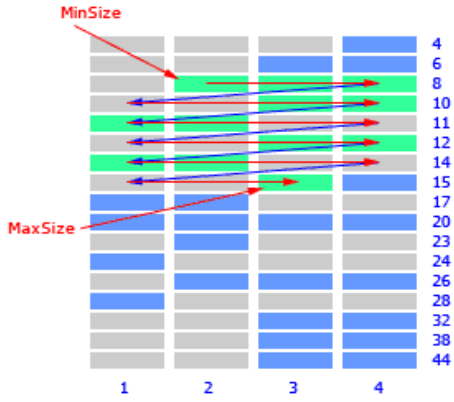
specified in the Data property) event will occur. See diagram:



- **soRowPrior**: If the MicroPDF417 symbol that use minimum symbol size cannot accommodate the barcode text, the number of rows will be increased firstly until it reachs the maximum number of rows defined by the specification or the barcode text is accommodated by the symbol. If the number of rows has reached the maximum number of rows defined by the specification but the barcode text cannot yet be accommodated, the number of columns will be increased and the number of rows will be reset to minimum number of rows defined by the specification. Repeat these until the barcode text is accommodated by the symbol, or the symbol size reachs maximum symbol size (specified by the MaxSize property), in this case (tha barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur. See diagram:



- **soColumnPrior**: If the MicroPDF417 symbol that use minimum symbol size cannot accommodate the barcode text, the number of columns will be increased firstly until it reachs the maximum number of columns defined by the specification or the barcode text is accommodated by the symbol. If the number of columns has reached the maximum number of columns defined by the specification but the barcode text cannot yet be accommodated, the number of rows will be increased and the number of columns will be reset to minimum number of columns defined by the specification. Repeat these until the barcode text is accommodated by the symbol, or the symbol size reachs maximum symbol size (specified by the MaxSize property), in this case (tha barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur. See diagram:



- **soGlobal_1**: All available symbol sizes will be sorted in ascending order of data capacity. If the data capacitoies of any two or three symbol sizes are equivalent, only smaller one in number of rows will be sorted. According to the order, the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size (specified by the MinSize property) and the maximum symbol size (specified by the MaxSize property). The order is listed below:

  mpSize_1_11, mpSize_3_6, mpSize_1_14, **mpSize_2_8**, **mpSize_1_17**, **mpSize_1_20**, mpSize_2_11, mpSize_3_10, mpSize_1_24, **mpSize_3_12**, mpSize_2_14, mpSize_1_28, mpSize_2_17, mpSize_2_20, mpSize_4_12, mpSize_2_23, mpSize_3_20, mpSize_2_26, mpSize_4_15, mpSize_3_26, mpSize_4_20, mpSize_3_32, mpSize_3_38, mpSize_4_26, mpSize_3_44, mpSize_4_32, mpSize_4_38, mpSize_4_44

  If tha barcode text cannot be accommodated by the maximum symbol size (specified by the MaxSize property), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soGlobal_2**: All available symbol sizes will be sorted in ascending order of data capacity. If the data capacitoies of any two or three symbol sizes are equivalent, only smaller one in number of columns will be sorted. According to the order, the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size (specified by the MinSize property) and the maximum symbol size (specified by the MaxSize property). The order is listed below:

  mpSize_1_11, mpSize_3_6, mpSize_1_14, **mpSize_4_4**, **mpSize_3_8**, **mpSize_4_6**, mpSize_2_11, mpSize_3_10, mpSize_1_24, **mpSize_4_8**, mpSize_2_14, mpSize_1_28, mpSize_4_10, mpSize_2_20, mpSize_4_12, mpSize_2_23, mpSize_3_20, mpSize_2_26, mpSize_4_15, mpSize_3_26, mpSize_4_20, mpSize_3_32, mpSize_3_38, mpSize_4_26, mpSize_3_44, mpSize_4_32, mpSize_4_38, mpSize_4_44

  If tha barcode text cannot be accommodated by the maximum symbol size (specified by the MaxSize property), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soGlobal_3**: All available symbol sizes will be sorted in ascending order of data capacity. If the data capacitoies of any two symbol sizes are equivalent, only smaller one in number of rows will be sorted. If the data capacitoies of any three symbol sizes are equivalent, only middle one in number of rows and columns will be sorted. According to the order, the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size (specified by the MinSize property) and the maximum symbol size (specified by the MaxSize property). The order is listed below:

  mpSize_1_11, mpSize_3_6, mpSize_1_14, **mpSize_2_8**, **mpSize_1_17**, **mpSize_1_20**, mpSize_2_11, mpSize_3_10, mpSize_1_24, **mpSize_3_12**, mpSize_2_14, mpSize_1_28, mpSize_3_15, mpSize_2_20, mpSize_4_12, mpSize_2_23, mpSize_3_20, mpSize_2_26, mpSize_4_15, mpSize_3_26, mpSize_4_20, mpSize_3_32, mpSize_3_38, mpSize_4_26, mpSize_3_44, mpSize_4_32, mpSize_4_38, mpSize_4_44

  If tha barcode text cannot be accommodated by the maximum symbol size (specified by the MaxSize property), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soGlobal_4**: All available symbol sizes will be sorted in ascending order of data capacity. If the data capacitoies of any two

symbol sizes are equivalent, only smaller one in number of columns will be sorted. If the data capacitoies of any three symbol sizes are equivalent, only middle one in number of rows and columns will be sorted. According to the order, the smallest symbol size that accommodates the barcode text will be automatically selected between the minimum symbol size (specified by the MinSize property) and the maximum symbol size (specified by the MaxSize property). The order is listed below:

mpSize_1_11, mpSize_3_6, mpSize_1_14, **mpSize_4_4**, **mpSize_3_8**, **mpSize_4_6**, mpSize_2_11, mpSize_3_10, mpSize_1_24, **mpSize_4_8**, mpSize_2_14, mpSize_1_28, mpSize_3_15, mpSize_2_20, mpSize_4_12, mpSize_2_23, mpSize_3_20, mpSize_2_26, mpSize_4_15, mpSize_3_26, mpSize_4_20, mpSize_3_32, mpSize_3_38, mpSize_4_26, mpSize_3_44, mpSize_4_32, mpSize_4_38, mpSize_4_44

If tha barcode text cannot be accommodated by the maximum symbol size (specified by the MaxSize property), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

See also the "Symbol sizes" section in the "TBarcode2D_MicroPDF417" article.

A.1.134 StretchOrder

(TBarcode2D_PDF417)

Specifies the priority order of selecing appropriate symbol size for a PDF417 barcode symbol.

Syntax:

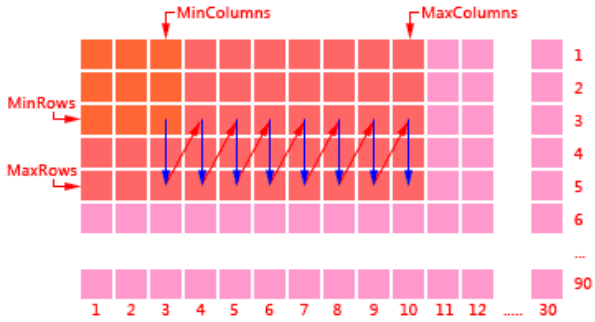
```
type
  { Defined in the pPDF417 unit }
  TPDF417_StretchOrder = (soRowColumn, soColumnRow, soFixAspect,
    soFixAspectWithQuietZones);
property StretchOrder: TPDF417_StretchOrder;
```

Description:

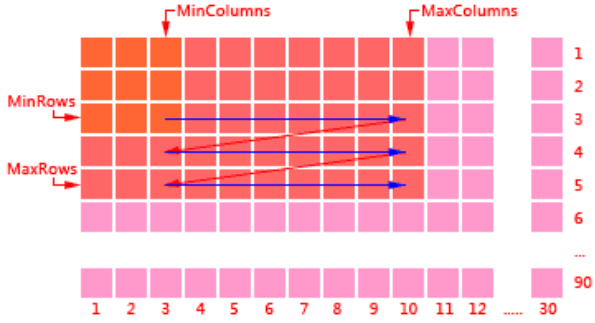
The minimum number of rows (specified by the MinRows property) and the minimum number of columns (specified by the MinColumns property) indicates the minimum symbol size for a PDF417 barcode symbol. The maximum number of rows (specified by the MaxRows property) and the maximum number of columns (specified by the MaxColumns property) indicates the maximum symbol size for the PDF417 barcode symbol. The property specifies a priority order in order to automatically select the symbol size. According to the priority order, the first symbol size (number of rows and columns) that accommodates the barcode text will be automatically selected between the minimum symbol size and the maximum symbol size.

The property can be one of these values (defined in the pPDF417 unit):

- **soRowColumn**: If the PDF417 symbol that use minimum symbol size (specified by the MinColumns and MinRows properties) cannot accommodate the barcode text, the number of stacked rows will be increased firstly until it reachs the maximum number of stacked rows (specified by the MaxRows property) or the barcode text is accommodated by the symbol. If the number of stacked rows has reached the maximum number of stacked rows (specified by the MaxRows property) but the barcode text cannot yet be accommodated, the number of columns will be increased and the number of stacked rows will be reset to minimum number of stacked rows (specified by the MinRows property). Repeat these until the barcode text is accommodated by the symbol, or not only does the number of rows reach maximum number of stacked rows (specified by the MaxRows property), but also the number of columns reachs maximum number of columns (specified by the MaxColumns property), in this case (the barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur. See diagram:



- **soColumnRow**: If the PDF417 symbol that use minimum symbol size (specified by the MinColumns and MinRows properties) cannot accommodate the barcode text, the number of columns will be increased firstly until it reachs the maximum number of columns (specified by the MaxColumns property) or the barcode text is accommodated by the symbol. If the number of columns has reached the maximum number of columns (specified by the MaxColumns property) but the barcode text cannot yet be accommodated, the number of stacked rows will be increased and the number of columns will be reset to minimum number of columns (specified by the MinColumns property). Repeat these until the barcode text is accommodated by the symbol, or not only does the number of columns reach maximum number of columns (specified by the MaxColumns property), but also the number of stacked rows reachs maximum number of stacked rows (specified by the MaxRows property), in this case (the barcode text cannot always be accommodated), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur. See diagram:



- **soFixAspect**: An overall aspect ratio is specified by the minimum number of stacked rows (specified by the MinRows property) and minumum number of columns (specified by the MinColumns property), the ratio is height to width of the symbol excluding quiet zones. Between the minimum symbol size (specified by the MinColumns and MinRows properties) and the maximum symbol size (specified by the MaxColumns and MaxRows properties), only symbol sizes that they approximately match the aspect ratio will be used. According to the data capacity, in increasing order of recovery capacity, the smallest symbol size that accommodates the barcode text will be automatically selected. If the barcode text cannot be accommodated by the maximum symbol size (specified by the MaxColumns and MaxRows properties), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

- **soFixAspectWithQuietZones**: An overall aspect ratio is specified by the minimum number of stacked rows (specified by the MinRows property) and minumum number of columns (specified by the MinColumns property), the ratio is height to width of the symbol including quiet zones. Between the minimum symbol size (specified by the MinColumns and MinRows properties) and the maximum symbol size (specified by the MaxColumns and MaxRows properties), only symbol sizes that they approximately match the aspect ratio will be used. According to the data capacity, in increasing order of recovery capacity, the smallest symbol size that accommodates the barcode text will be automatically selected. If the barcode text cannot be accommodated by the maximum symbol size (specified by the MaxColumns and MaxRows properties), an OnInvalidLength (the barcode text is specified in the Barcode property) or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later, and the barcode text is specified in the Data property) event will occur.

See also the "Symbol size" section in the "TBarcode2D_PDF417" article.

A.1.135 Style

(TBarcode2D_RSS14)

Specifies which style (version) of RSS-14 symbols will be used.

Syntax:

```
type
  { Defined in the pRss14 unit }
  TRSS14_Style = (rsStandard, rsTruncated, rsStacked, rsStackedOmnidirectional);
property Style: TRSS14_Style;
```

Description:

There are four styles (versions) of RSS-14 symbols, referred to as RSS-14 (Standard), RSS-14 Truncated, RSS-14 Stacked, and RSS-14 Stacked Omnidirectional respectively. The property specifies which style (version) will be used.

The property can be one of these values (defined in the pRss14 unit):

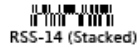
- **rsStandard**: Indicates to generate the standard style RSS-14 symbol. It encodes the full 14-digit EAN.UCC item identification (Global Trade Item Numbers, GTINs) in a symbol that can be omnidirectionally scanned by suitably configured point-of-sale laser scanners.

- **rsTruncated**: Indicates to generate the RSS-14 Truncated barcode symbol. It is structured and encoded in the same way as the standard RSS-14 format, except its height is reduced to a 13 modules minimum. It may be used for small items, instead of RSS Limited It may also be used when the four-column 2D component is desired in order to minimize the height of an EAN.UCC Composite symbol.

  RSS-14 Truncated is designed to be read by scanners such as wands, handheld lasers, and linear and 2D imagers. It cannot be read efficiently by omnidirectional flat-bed and presentation scanners.

- **rsStacked**: Indicates to generate the RSS-14 Stacked barcode symbol. It is an RSS-14 Truncated two-row format. It may be used for small items instead of RSS Limited when the available space is too narrow for RSS Limited. Moreover, the narrower width of RSS-14 Stacked might allow for a larger module width and potentially higher print quality. However, RSS Limited or RSS-14 Truncated should be used in preference to the stacked format whenever space permits without reducing module width, as they are easier to scan with a wand or linear scanner.

  RSS-14 Stacked is designed to be read by scanners such as wands, handheld lasers, and linear and 2D imagers. It cannot be read efficiently by omnidirectional flat-bed and presentation scanners.



- **rsStackedOmnidirectional**: Indicates to generate the RSS-14 Stacked Omnidirectional barcode symbol. It is a full height RSS-14 two-row format. It can be used instead of RSS-14 for omnidirectional scanning applications where the different aspect ratio is needed.

See also the "Styles" section in the "TBarcode2D_RSS14" article.

A.1.136 Style

(TBarcode2D_RSSExpanded)

Specifies which style (version) of RSS Expanded symbols will be used.

Syntax:

```
type
  { Defined in the pRssExpanded unit }
  TRSSExpanded_Style = (rsStandard, rsStacked);
property Style: TRSSExpanded_Style;
```

Description:

There are two styles (versions) of RSS Expanded symbols, referred to as RSS Expanded (Standard) and RSS Expanded Stacked respectively. The property specifies which style (version) of RSS Expanded symbol will be used.

The property can be one of these values (defined in the pRssExpanded unit):

- **rsStandard**: Indicates to generate the standard style RSS Expanded symbol. It is a single row, variable length linear symbology. It can identify small items and carry more information than the current EAN/UPC barcode.

- **rsStacked**: Indicates to generate the RSS Expanded Stacked symbol. It is an RSS Expanded multi-row format. It may be used when the symbol area or print mechanism is not wide enough to accommodate the full single row symbol.

See also the "Styles" section in the "TBarcode2D_RSSExpanded" article.

A.1.137 SymbolMode

(TBarcode2D_AztecCode)

Specifies which symbol format and size will be automatically selected to generate an Azec Code symbol.

Syntax:

```
type
  { Defined in the pAztecCode unit }
  TAztecCode_SymbolMode = (smNormal, smCompact, smFullRange, smProgram, smAll);
property SymbolMode: TAztecCode_SymbolMode;
```

Description:

The property specifies which Aztec Code symbol formats and sizes will be automatically selected between minimum and maximum symbol sizes specified by corresponding MinSize and MaxSize properties, depending on the length of barcode text.

The property can be one of these values (defined in the pAztecCode unit):

- **smNormal**: Only the symbol sizes that they are useful for data encoding operation, including all compact symbol sizes (azSize_15Compact, azSize_19Compact, azSize_23Compact, and azSize_27Compact), and full range symbol sizes from azSize_31 to azSize_151.
- **smCompact**: Only the symbol sizes in compact format, including the azSize_15Compact, azSize_19Compact, azSize_23Compact, and azSize_27Compact.
- **smFullRange**: Only the symbol sizes in full range format, including the azSize_19, azSize_23, azSize_27, and azSize_31 to azSize_151.
- **smProgram**: Only the symbol sizes that they are useful for reader initialization, including azSize_15Compact, azSize_19, azSize_23, azSize_27, and full range symbol sizes from azSize_31 to azSize_109. Note, in order to create a reader initialization symbol, a "\p" escape sequence shoule be placed into the barcode text, and the AllowEscape property should be set to true.
- **smAll**: All symbol sizes, including compact and full range formats.

See also the "Formats" and "Symbol sizes" sections in the "TBarcode2D_AztecCode" article.

A.1.138 TopMargin

Specifies the top margin of the barcode symbol in pixels.

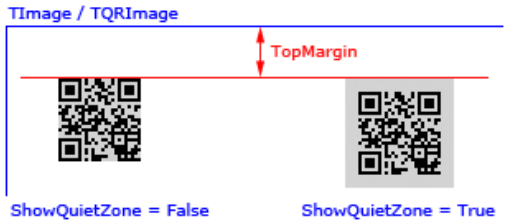
Syntax:

```
property TopMargin: Integer;
```

Description:

Specifies the margin between the topmost of the barcode symbol and the top side of the TImage, TQRImage, or TQRGzImage control that is specified by the Image property, in pixels.

If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



It is set using the following formula:

- The Orientation property is set to "boLeftRight":

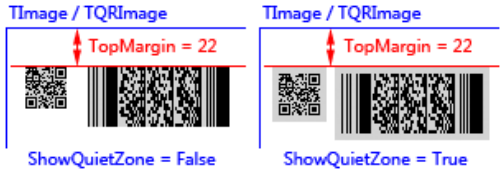
  It is the margin between the top of the barcode symbol and the top side of the TImage, TQRImage, or TQRGzImage control.

  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

- The Orientation property is set to "boRightLeft":

  It is the margin between the bottom of the barcode symbol and the top side of the TImage, TQRImage, or TQRGzImage control.

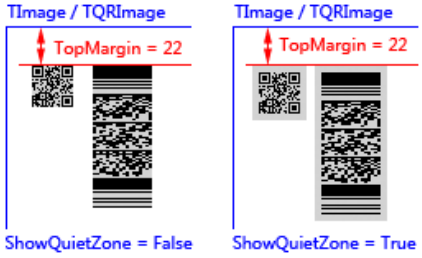
  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- The Orientation property is set to "boTopBottom":

  It is the margin between the beginning of the barcode symbol and the top side of the TImage, TQRImage, or TQRGzImage control.

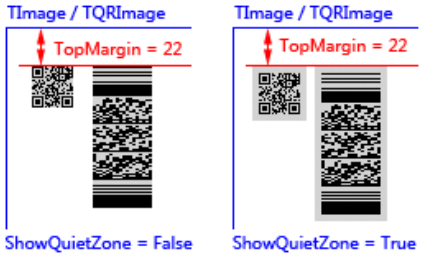
  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



- The Orientation property is set to "boBottomTop":

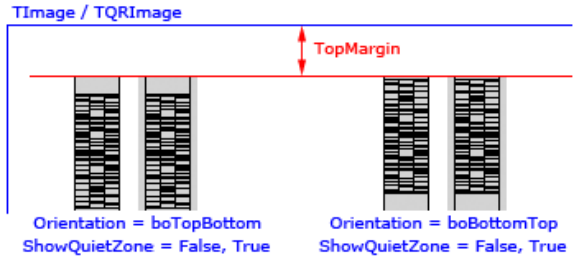
  It is the margin between the end of the barcode symbol and the top side of the TImage, TQRImage, or TQRGzImage control.

  See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



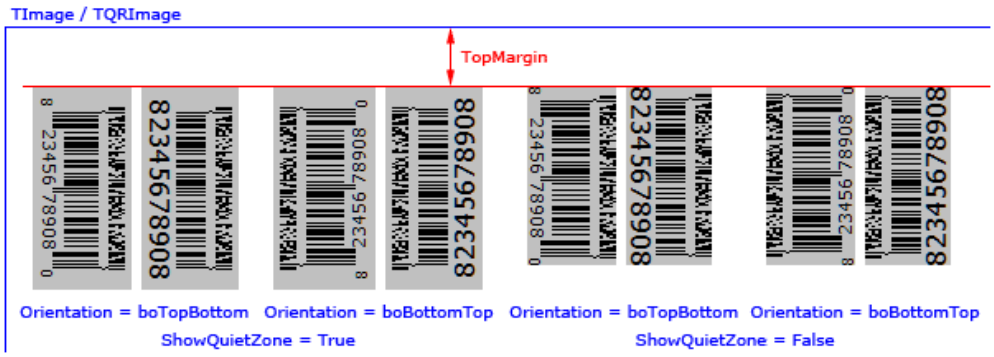
Note:

For the TBarcode2D_Code16K barcode components, the leading quiet zone and trailing quiet zone are included always, even if the ShowQuietZone is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear component is included in the composite barcode symbol, and the parameter specifies the top margin for entire EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

When the linear component of the TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The property always specifies the top margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



A.1.139 TopQuietZone

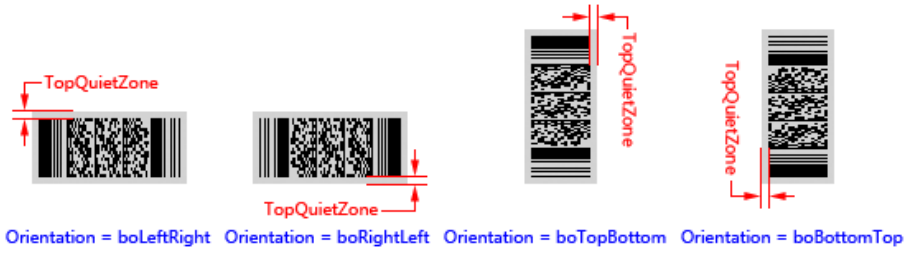
Specifies the vertical height of the top quiet zone in modules.

Syntax:

```
property TopQuietZone: Integer;
```

Description:

Specifies the vertical height of the top quiet zone in modules. The quiet zone is represented using the color specified by the SpaceColor property if the Inversed property is set to false. Otherwise, it's represented using the color specified by the BarColor property. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



This property is useful only when the ShowQuietZone property value is set to true.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the top quiet zone is close to the top boundaries of entire EAN.UCC composite barcode symbol, namely, the top boundaries of the 2D symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

A.1.140 TotalHeight

(TBarcode2D_RSS14, TBarcode2D_RSSLimited)

Specifies the total height for an RSS-14 or an RSS Limited symbol, in modules.

Syntax:

```
property TotalHeight: Integer;
```

Description:

The property specifies the total height for an RSS-14 or an RSS Limited symbol, in modules.

For all styles of RSS-14 symbol, and RSS Limited symbol, if both the Link2D and Show2DSeparator properties are set to Ture, a contiguous separator pattern is represented and its minimum height is 1 module (the pattern height can be specified by the SeparatorRowHeight property). The height of contiguous separator pattern is included in the value of TotalHeight property.

For the RSS-14 Stacked and RSS-14 Stacked Omnidirectional symbols, the height of separator pattern between symbol rows (specified by the SeparatorRowHeight property) is included.

The property is set using the following formula:

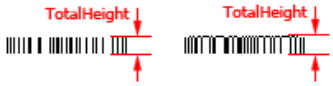
- For the RSS-14 symbols:
  - **RSS-14 (Standard)** Appropriate TotalHeight value should be set in order to ensure that the height of symbol row is greater than or equal to 33 modules. In other words, if the contiguous separator pattern isn't represented, minimum value of the property is 33 modules. Otherwise, it's 33 modules plus height of the contiguous separator pattern (specified by the SeparatorRowHeight property). See diagram:

  - **RSS-14 Truncated** Appropriate TotalHeight value should be set in order to ensure that the height of symbol row is greater than or equal to 13 modules. In other words, if the contiguous separator pattern isn't represented, minimum value of the property is 13 modules. Otherwise, it's 13 modules plus height of the contiguous separator pattern (specified by the SeparatorRowHeight property). See diagram:

  - **RSS-14 Stacked** Appropriate TotalHeight value should be set in order to ensure that the sum of each symbol row's height is greater than or equal to 12 modules. In other words, if the contiguous separator pattern isn't represented, minimum value of the property is 12 modules plus the height of separator pattern between two rows (specified by the SeparatorRowHeight property). Otherwise, it's 12 modules plus the height of separator pattern between two rows (specified by the SeparatorRowHeight property), and plus height of the contiguous separator pattern (specified by the SeparatorRowHeight property too). See diagram:

  - **RSS-14 Stacked Omnidirectional** Appropriate TotalHeight value should be set in order to ensure that the sum of each symbol row's is greater than or equal to 66 modules. In other words, if the contiguous separator pattern isn't represented, minimum value of the property is 66 modules plus 3 times height of separator pattern between two rows (specified by the SeparatorRowHeight property). Otherwise, it's 66 modules plus the 3 times height of separator pattern between two rows (specified by the SeparatorRowHeight property), and plus height of the contiguous separator pattern (specified by the SeparatorRowHeight property too). See diagram:

- For the RSS Limited symbols, appropriate TotalHeight value should be set in order to ensure that the height of symbol row is greater than or equal to 10 modules. In other words, if the contiguous separator pattern isn't represented, minimum value of the property is 10 modules. Otherwise, it's 10 modules plus height of the contiguous separator pattern (specified by the SeparatorRowHeight property). See diagram:



See also the "Symbol size" and "Composite symbol" sections in the "TBarcode2D_RSS14" and "TBarcode2D_RSSLimited" articles.

A.1.141 TrailingQuietZone

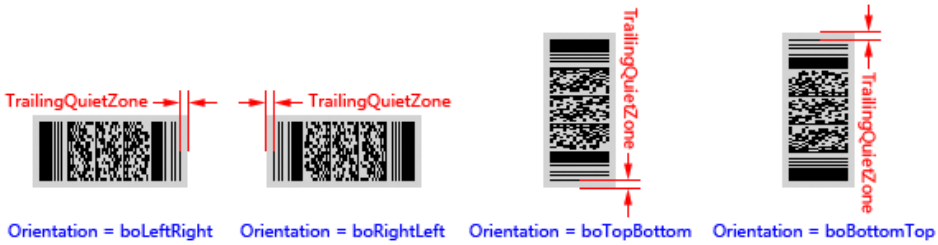
Specifies the horizontal width of the trailing quiet zone in modules.

Syntax:

```
property TrailingQuietZone: Integer;
```

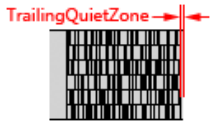
Description:

Specifies the horizontal width of the trailing quiet zone in modules. The quiet zone is represented using the color specified by the SpaceColor property if the Inversed property is set to false. Otherwise, it's represented using the color specified by the BarColor property. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

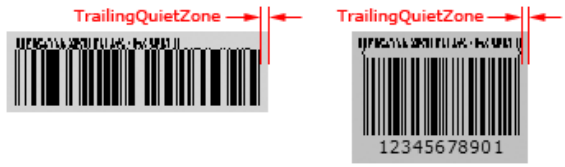


This property is useful only when the ShowQuietZone property value is set to true or the barcode symbology is a TBarcode2D_Code16K symbol.

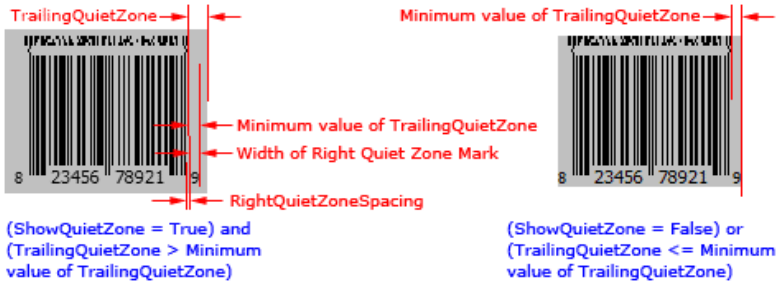
For the TBarcode2D_Code16K barcode component, the trailing quiet zone is represented always, even if the ShowQuietZone property value is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



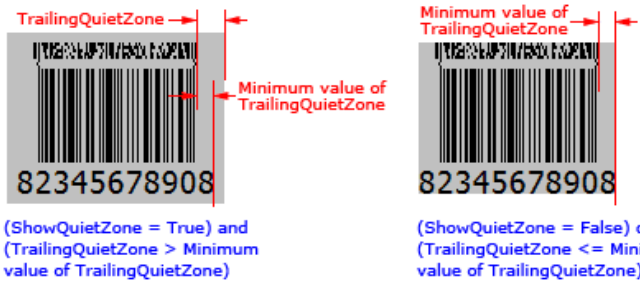
For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the trailing quiet zone begins from the last bar or space of entire EAN.UCC composite barcode symbol that includes the linear symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



Note, For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is one of the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, and TBarcode1D_EAN13, and its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol is included in the width of trailing quiet zone, and it's the minimum width of the trailing quiet zone. In other words, the trailing quiet zone always begins from the last bar of the 2D symbol. The minimum width will be used if the value of the TrailingQuietZone property is less than it or the value of the ShowQuietZone property is set to false, the trailing quiet zoens will be automatically extanded to cover the right quiet zone mark, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the end of entire barcode symbol (the trailing quiet zones is included if the ShowQuietZone property is set to true), the trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone property is set to true or false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



A.1.142 UseECIDescriptor

(TBarcode2D_MicorPDF417)

Specifies whether to use the optional ECI Descriptor in aMicroPDF417 barcode symbol, in order to efficiently encode a single Transformation ECI sequence at the start of barcode text.

Syntax:

```
property UseECIDescriptor: Boolean;
```

Description:

Because MicroPDF417 is intended for applications where symbol size is critical, it is expected that MicroPDF417 will frequently be used in conjunction with a Transformation ECI appropriate for the data requirements of the application.MicroPDF417 supports efficient encoding of a single Transformation ECI sequence at the start of each data stream by using an optional ECI Descriptor that can be encoded at the start of the first symbol of each MicroPDF417-encoded data stream. The property specifies whether to use the ECI Descriptor in a MicroPDF417 symbol.

This Transformation ECI co-exists with the Interpretative ECI (whether default or explicitly encoded) currently in effect, since Interpretative ECIs and Transformation ECIs have independent scopes.

Please use the escape sequence \e[<ECI_Number>] to place a Transformation ECI at the beginning of the barcode text (at the start of the symbol or at the start of the first symbol of a structured append series). If the property is set to true, the Transformation ECI sequence will be ecncoded as an ECI Descriptor. Otherwist, it will be encoded using the standard ECI encoding method.

See also the "Extended Channel Interpretation (ECI)" section in the "TBarcode2D_MicroPDF417" article.

A.2 TDBBarcode2D

A.2.1 Barcode2D

Links a TBarcode2D component to the TDBBarcode2D component in order to represent the barcode symbol from a data field of the current record of a dataset.

Syntax:

```
property Barcode2D: TBarcode2D;
```

Description:

Use the Barcode2D property to specify a TBarcode2D component such as the TBarcode2D_QRCode, the TBarcode2D_PDF417, and the TBarcode2D_RSS14. It will represent the barcode symbol from a data field (specified by the DataField property of the TDBBarcode2D component) of the current record of a dataset (specified by the DataSource property of the TDBBarcode2D component) to a TImage, TQRImage, or TQRGzImage control that's specified by the Image property of the TBarcode2D component.

When a TBarcode2D barcode component is linked to the TDBBarcode2D component, for Delphi/C++ Builder 2007 or early, the data field value will be applied to its Barcode property. For the Delphi/C++ Builder 2009 or later, the data field value will be applied to its Barcode or Data property, please use the BindProperty property to indicate which property the data field value will be applied to.

A.2.2 BindProperty

Indicates which property of the TBarcode2D component the data field value will be applied to, in order to represent the barcode symbol from a data field of the current record of a dataset.

Syntax:

```
type
```

```
  { Defined in the pDBBarcode2D unit }
  TBindProperty = (bpBarcode, bpData);
property BindProperty: TBindProperty;
```

Description:

The Barcode property of TBarcode2D component is of type string, for Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. In order to encode block of binary (bytes) data, we added a Data property, it is of type AnsiString.

When a TBarcode2D barcode component is linked to the TDBBarcode2D component, it will represent the barcode symbol from a data field (specified by the DataField property of the TDBBarcode2D component) of the current record of a dataset (specified by the DataSource property of the TDBBarcode2D component). The BindProperty property indicates which property of the TBarcode2D component the data field value will be applied to.

This property can be one of these values (defined in the pDBBarcode2D unit):

- **bpBarcode**: The data field value will be applied to the Barcode property of the TBarcode2D barcode component.
- **bpData**: The data field value will be applied to the Data property of the TBarcode2D barcode component.

Note:

The property is available only for the Delphi/C++ Builder 2009 or later.

A.2.3 DataField

Specifies the field from which the TDBBarcode2D component represents barcode symbol.

Syntax:

```
property DataField: string;
```

Description:

Use DataField to bind the TDBBarcode2D component to a field in the dataset. To fully specify a data field, both the dataset and the field within that dataset must be defined. The DataSource property of the TDBBarcode2D component specifies the dataset which contains the data field. The data field should be specifed as a string, bytes, blob, memo, etc field.

When a TBarcode2D barcode component is linked to the TDBBarcode2D component, for Delphi/C++ Builder 2007 or early, the data field value will be applied to its Barcode property. For the Delphi/C++ Builder 2009 or later, the data field value will be applied to its Barcode or Data property, please use the BindProperty property to indicate which property the data field value will be applied to.

You can bind multiple TDBBarcode2D components to one data field in order to represent the data field with multiple barcode symbols.

A.2.4 DataSource

Links the TDBBarcode2D component to the dataset that contains the field it represents.

Syntax:

```
property DataSource: TDataSource;
```

Description:

Use DataSource to specify the data source component through which the data from a dataset component is provided to the TDBBarcode2D component. To fully specify a data field for the TDBBarcode2D component, both the dataset and a field within that dataset must be defined. Use the DataField property to specify the particular field within the dataset.

You can bind multiple TDBBarcode2D components to one data field in order to represent the data field with multiple barcode symbols.

A.2.5 Field

Indicates the TField object whose current value the barcode component represents.

Syntax:

```
property Field: TField;
```

Description:

Use the TField reference provided by the Field property when you want to read or to write the value of the data in the field programmatically.

A.2.6 ReadOnly

Determines whether the user can change the value of the field.

Syntax:

```
property ReadOnly: Boolean;
```

Description:

Use the ReadOnly property to specify whether the TBarcode2D barcode component specified in the Barcode2D property of a TDBBarcode2D component allows the user to change the field value by changing the value of its Barcode or Data (only for Delphi/C++ Builder 2009 or later) property.

When the ReadOnly property is set to true, the TBarcode2D barcode component can only be used to represent the value of the field on the current record.

When the ReadOnly property is set to false, for Delphi/C++ Builder 2007 or early, the user can change the field value by changing the Barcode property value of the TBarcode2D barcode component specified in the Barcode2D property of the TDBBarcode2D component. For the Delphi/C++ Builder 2009 or later, if the BindProperty is set to bpBarcode, the user can change the field value by changing the Barcode property value of the TBarcode2D barcode component specified in the Barcode2D property of the TDBBarcode2D component. If the BindProperty is set to bpData, the user can change the field value by changing the Data property value of the TBarcode2D barcode component specified in the Barcode2D property of the TDBBarcode2D component.

A.3 TLogo2D

A.3.1 AlignToModule

Specifies how to place the logo picture in a barcode symbol in horizontal and vertical directions.

Syntax:

```
property AlignToModule: Boolean;
```

Description:

The property specifies how to place the logo picture in a barcode symbol, in horizontal and vertical directions:

- In horizontal dircection, if the HorizontalAlignment property is set to haMiddle, the logo picture will be centred in horizontal direction, the number of modules on left of the logo picture is equal to the number of modules on right of the logo picture. If the total number of modules on left and right of logo picture is odd, set the property to true to align the logo picture to module edge, and the number of modules on left of the logo picture is 1 less than the number of modules on right of the logo picture. If the property is set the false, a half of module will be included in the modules on left and right of logo picture, the logo picture will be centred in horizontal direction absolutely. See diagram:

AlignToModule = False

AlignToModule = True

If the HorizontalAlignment property is set to haLeft or haRight, the AlignToModule property value will be ignored in horizontal direction.

- In vertical dircection, if the VerticalAlignment property is set to vaCenter, the logo picture will be centred in vertical direction, the number of modules on top of the logo picture is equal to the number of modules on bottom of the logo picture. If the total number of modules on top and bottom of logo picture is odd, set the property to true to align the logo picture to module edge, and the number of modules on top of the logo picture is 1 less than the number of modules on bottom of the logo picture. If the property is set the false, a half of module will be included in the modules on top and bottom of logo picture, the logo picture will be centred in vertical direction absolutely. See diagram:

AlignToModule = False

AlignToModule = True

If the VerticalAlignment property is set to vaTop or vaBottom, the AlignToModule property value will be ignored in vertical direction.

A.3.2 Barcode2D

Links a TBarcode2D component to the TLogo2D component in order to add a logo picture to barcode symbol thats specified in the TBarcode2D component.

Syntax:

```
property Barcode2D: TBarcode2D;
```

Description:

Use the Barcode2D property to specify a TBarcode2D component such as the TBarcode2D_QRCode, the TBarcode2D_PDF417, and the TBarcode2D_RSS14.

A.3.3 Height

Specifies the distance between the top and bottom of the logo picture, in modules.

Syntax:

```
property Height: Integer;
```

Description:

Specifies the distance between the top and bottom of the logo picture in modules. The paddings isn't included if the Padding property is set to 1 or more. See diagram:

Orientation = boLeftRight

Note:

If the Orientation property of TBarcode2D is set to "boRightLeft", "boTopBottom" or "boBottomTop", the logo picture will be rotated too, and the property specifies the height of logo picture before rotating. See diagram:

Orientation = boTopBottom

A.3.4 HorizontalAlignment

Specifies the horizontal alignment of the logo picture in a barcode symbol.

Syntax:

```
type
  { Defined in the pLogo2D unit }
  THorizontalAlignment = (haLeft, haMiddle, haRight);
property HorizontalAlignment: THorizontalAlignment default haMiddle;
```

Description:

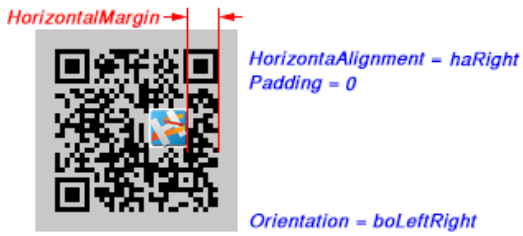
Specifies the horizontal alignment of the logo picture in a barcode symbol. It can be one of the following values (defined in the pLogo2D unit):

- **haLeft**: The logo picture with its left margin is aligned to the beginning edge of barcode symbol (the leading queit zone isn't included if it is represented). The left margin of the logo picture is the distance between the beginning edges of barcode symbol and the logo picture, in modules. It's specified by the HorizontalMargin property. See diagram:

- **haMiddle**: The logo picture will be centred in horizontal direction, the number of modules on left of the logo picture is equal to the number of modules on right of the logo picture.

  If the total number of modules on left and right of logo picture is odd, the property AlignToModule will be used to place the logo picture in a barcode symbol in horizontal direction. Set the property to true to align the logo picture to module edge, and the number of modules on left of the logo picture is 1 less than the number of modules on right of the logo picture. If the property is set the false, a half of module will be included in the modules on left and right of logo picture, the logo picture will be centred in horizontal direction absolutely. See diagram:

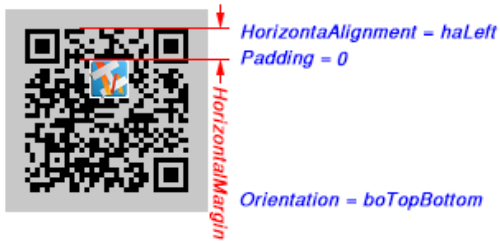
- **haRight**: The logo picture with its right margin is aligned to the end edge of barcode symbol (the trailing queit zone isn't included if it is represented). The right margin of the logo picture is the distance between the end edges of barcode symbol and the logo picture, in modules. It's specified by the HorizontalMargin property. See diagram:



Note:

The paddings is included in the logo picture if the Padding property is set to 1 or more. See diagram:

If the Orientation property of TBarcode2D is set to "boRightLeft", "boTopBottom" or "boBottomTop", the logo picture will be rotated too, and the property specifies the horizontal alignment of the logo picture before rotating. See diagram:



A.3.5 HorizontalMargin

Specifies the distance between the edges of barcode symbol and logo picture in horizontal direction.

Syntax:

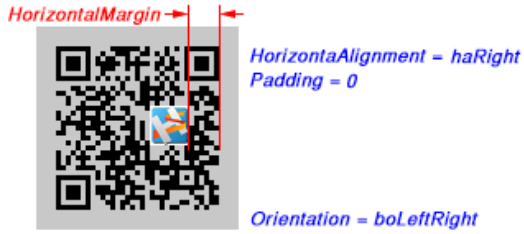
```
property HorizontalMargin: Integer;
```

Description:

Specifies the distance between the edges of barcode symbol and logo picture in horizontal direction, in modules:

- If the HorizontalAlignment property is set to haLeft, the property specifies the distance between the beginning edges of barcode symbol and the logo picture, in modules. The leading queit zone isn't included if it is represented. See diagram:

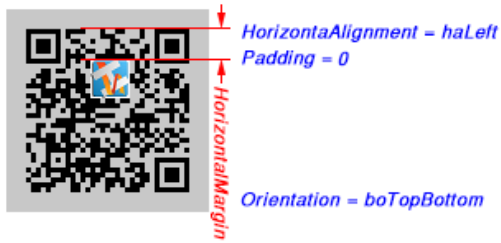
- If the HorizontalAlignment property is set to haMiddle, the property is ignored.
- If the HorizontalAlignment property is set to haRight, the property specifies the distance between the end edges of barcode symbol and logo picture, in modules. The trailing queit zone isn't included if it is represented. See diagram:



Note:

The paddings is included in the logo picture if the Padding property is set to 1 or more. See diagram:

If the Orientation property of TBarcode2D is set to "boRightLeft", "boTopBottom" or "boBottomTop", the logo picture will be rotated too, and the property specifies the distance before rotating. See diagram:



A.3.6 Padding

Specifies the top, right, bottom, and left padding around the logo picture, in modules.

Syntax:

```
property Padding: Integer;
```

Description:

The property specifies top, right, bottom, and left padding around the logo picture in modules, by default it is 1 module. See diagram:

A.3.7 Picture

Specifies the logo picture for barcode symbol.

Syntax:

```
property Padding: Integer;
```

Description:

The property specifies a logo picture for barcode symbol. The "bmp", "jpg"/"jpeg", "wmf", "emf", "ico", "png", "gif", and "tif"/"tiff" image formats are supported depend on the version of IDE:

	BMP, JPG/JPEG, ICO, WMF, EMF	GIF	PNG	TIF/TIFF
Delphi/C++ Builder 4 - 2006	V	-	-	-
Delphi/C++ Builder 2007	V	V	-	-
Delphi/C++ Builder 2009	V	V	V	-
Delphi/C++ Builder 2010-XE7	V	V	V	V

A.3.8 VerticalAlignment

Specifies the vertical alignment of the logo picture in a barcode symbol.

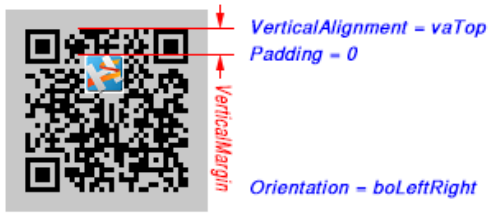
Syntax:

```
type
  { Defined in the pLogo2D unit }
  TVerticalAlignment = (vaTop, vaCenter, vaBottom);
property VerticalAlignment: TVerticalAlignment default vaCenter;
```

Description:

Specifies the vertical alignment of the logo picture in a barcode symbol. It can be one of the following values (defined in the pLogo2D unit):

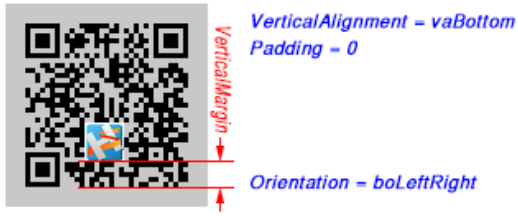
- **vaTop**: The logo picture with its top margin is aligned to the top edge of barcode symbol (the top queit zone isn't included if it is represented). The top margin of the logo picture is the distance between the top edges of barcode symbol and the logo picture, in modules. It's specified by the VerticalMargin property. See diagram:



- **vaCenter**: The logo picture will be centred in vertical direction, the number of modules on top of the logo picture is equal to the number of modules on bottom of the logo picture.

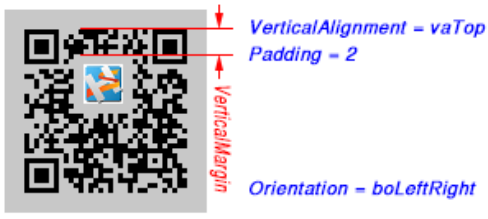
  If the total number of modules on top and bottom of logo picture is odd, the property AlignToModule will be used to place the logo picture in a barcode symbol in vertical direction. Set the property to true to align the logo picture to module edge, and the number of modules on top of the logo picture is 1 less than the number of modules on bottom of the logo picture. If the property is set the false, a half of module will be included in the modules on top and bottom of logo picture, the logo picture will be centred in vertical direction absolutely. See diagram:

- **vaBottom**: The logo picture with its bottom margin is aligned to the bottom edge of barcode symbol (the bottom queit zone isn't included if it is represented). The bottom margin of the logo picture is the distance between the bottom edges of barcode symbol and the logo picture, in modules. It's specified by the VerticalMargin property. See diagram:

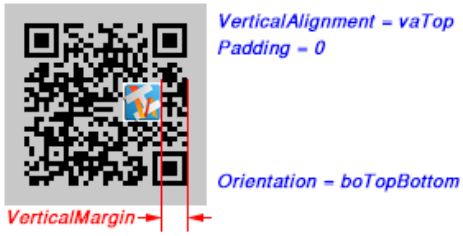


Note:

The paddings is included in the logo picture if the Padding property is set to 1 or more. See diagram:



If the Orientation property of TBarcode2D is set to "boRightLeft", "boTopBottom" or "boBottomTop", the logo picture will be rotated too, and the property specifies the vertical alignment of the logo picture before rotating. See diagram:



A.3.9 VerticalMargin

Specifies the distance between the edges of barcode symbol and logo picture in vertical direction.

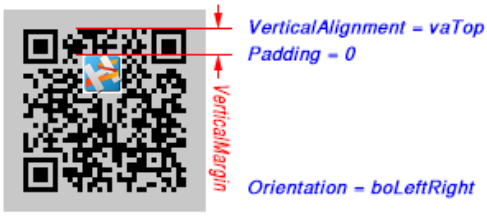
Syntax:

```
property VerticalMargin: Integer;
```

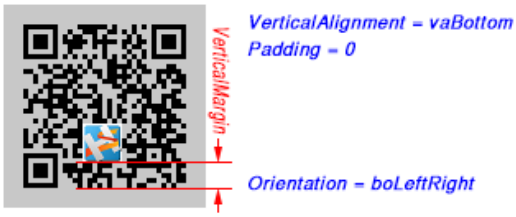
Description:

Specifies the distance between the edges of barcode symbol and logo picture in vertical direction, in modules:

- If the VerticalAlignment property is set to vaTop, the property specifies the distance between the top edges of barcode symbol and the logo picture, in modules. The top queit zone isn't included if it is represented. See diagram:

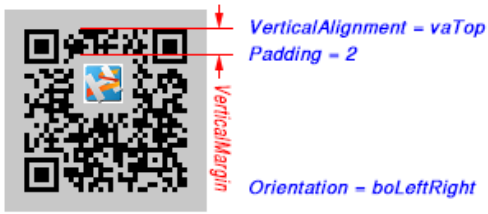


- If the VerticalAlignment property is set to vaCenter, the property is ignored.
- If the VerticalAlignment property is set to vaBottom, the property specifies the distance between the bottom edges of barcode symbol and logo picture, in modules. The trailing queit zone isn't included if it is represented. See diagram:

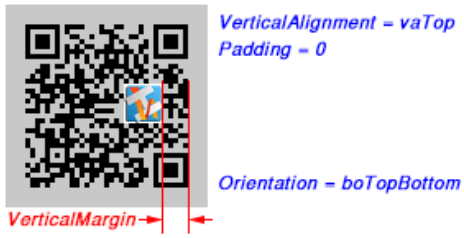


Note:

The paddings is included in the logo picture if the Padding property is set to 1 or more. See diagram:



If the Orientation property of TBarcode2D is set to "boRightLeft", "boTopBottom" or "boBottomTop", the logo picture will be rotated too, and the property specifies the distance before rotating. See diagram:



A.3.10 Width

Specifies the distance between the beginning and end of the logo picture, in modules.

Syntax:

```
property Width: Integer;
```

Description:

Specifies the distance between the beginning and end of the logo picture in modules. The paddings isn't included if the Padding property is set to 1 or more. See diagram:

Note:

If the Orientation property of TBarcode2D is set to "boRightLeft", "boTopBottom" or "boBottomTop", the logo picture will be rotated too, and the property specifies the width of logo picture before rotating. See diagram:

A.4 TSave2D

A.4.1 Barcode2D

Links a TBarcode2D component to the TSave2D component in order to save the barcode symbol to a picture file.

Syntax:

```
property Barcode2D: TBarcode2D;
```

Description:

Use the Barcode2D property to specify a TBarcode2D component such as the TBarcode2D_QRCode, the TBarcode2D_PDF417, and the TBarcode2D_RSS14. Then execute the Save method will save the barcode symbol specified by the 2D barcode component to a picture file.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Only the 2D portion will be saved in the picture file now.

The TBarcode2D_MaxiCode component is not supported now.

A.4.2 BarOpacity

(TSave2D_Png, TSave2D_Svg)

Specifies the transparency-level of entire barcode symbol.

Syntax:

```
property BarOpacity: Single;
```

Description:

For the TSave2D_Png and TSave2D_Svg components, when save a barcode symbol to a picture file, the property specifies the transparency-level of foreground color (bars or dark modules) if the Inversed property is set to false, otherwise, it specifies the transparency-level of bakcground color (spaces or light modules, and quiet zones). Where 1 is not transparant at all, 0.5 is 50% see-through, and 0 is completely transparent.

The property defaults to 1, meaning not transparant at all.

A.4.3 ColorDepth

(TSave2D_Bmp)

Specifies the color depth of a Bitmap image (BMP) file for TSave2D_Bmp component.

Syntax:

```
type
  { Defined in the pSave2D_Bmp unit }
  TBmpColorDepth = (bcdColors_01b, bcdColors_04b, bcdColors_08b, bcdColors_12b,
    bcdColors_15b, bcdColors_16b, bcdColors_24b, bcdColors_32b);
property Format: TBmpColorDepth;
```

Description:

Color depth, also known as bit depth, pixel format, is either the number of bits used to indicate the color of a single pixel. In a Bitmap image (BMP) file, the pixels can be defined by a varying number of bits. The property specifies which pixel format will be used to save a barcode symbol to a Bitmap image (BMP) file. It can be one of these value (defined in the pSave2D_Bmp unit):

- **bcdColors_01b**: Uses the 1-bit per pixel (1bpp) format. It supports 2 distinct colors (for example: black and white) and stores 8 pixels per 1 byte. Each pixel is an 1-bit index into a table of up to 2 colors.
- **bcdColors_02b**: Uses the 2-bit per pixel (2bpp) format. It supports 4 distinct colors and stores 4 pixels per 1 byte. Each pixel value is a 2-bit index into a table of up to 4 colors.
- **bcdColors_04b**: Uses the 4-bit per pixel (4bpp) format. It supports 16 distinct colors and stores 2 pixels per 1 byte. Each pixel value is a 4-bit index into a table of up to 16 colors.
- **bcdColors_08b**: Uses the 8-bit per pixel (8bpp) format. It supports 256 distinct colors and stores 1 pixel per 1 byte. Each byte is an index into a table of up to 256 colors.
- **bcdColors_12b**: Uses the 12-bit per pixel (12bpp) format. It supports 4096 distinct colors and stores 1 pixel per 2 byte. Each pixel value define the red, green and blue samples of the pixel, uses 4 bits for each color primary with 4 bits left over.
- **bcdColors_15b**: Uses the 15-bit per pixel (15bpp) format. It supports 32768 distinct colors and stores 1 pixel per 2 byte. Each pixel value define the red, green and blue samples of the pixel, uses 5 bits for each color primary with one bit left over.
- **bcdColors_16b**: Uses the 16-bit per pixel (16bpp) format. It supports 65536 distinct colors and stores 1 pixel per 2 byte. Each pixel value define the red, green and blue samples of the pixel, uses 5 bits for red and blue colors, 6 bits for green color.
- **bcdColors_24b**: Uses the 24-bit per pixel (24bpp) format. It supports 16777216 distinct colors and stores 1 pixel value per 3 bytes. Each pixel value defines the red, green and blue samples of the pixel.
- **bcdColors_32b**: Uses the 32-bit per pixel (32bpp) format. It supports 4294967296 distinct colors and stores 1 pixel per 4 byte. Each pixel value define the Alpha, Red, Green and Blue samples of the pixel.

A.4.4 ColorDepth

(TSave2D_Gif)

Specifies the color depth of a Graphics Interchange Format (GIF) file for the TSave2D_Gif component.

Syntax:

```
type
```

```
  { Defined in the pSave2D_Gif unit }
  TGifColorDepth = (gcdColors_1b, gcdColors_2b, gcdColors_3b, gcdColors_4b, gcdColors_5b,
    gcdColors_6b, gcdColors_7b, gcdColors_8b);
property ColorDepth: TGifColorDepth;
```

Description:

Color depth, also known as bit depth, pixel format, is either the number of bits used to indicate the color of a single pixel. In a Graphics Interchange Format (GIF) file, the pixels can be defined by a varying number of bits. The property specifies which color depth will be used to save a barcode symbol to a Graphics Interchange Format (GIF) file. It can be one of these value (defined in the pSave2D_Gif unit):

- **gcdColors_1b**: Uses the 1-bit per pixel (1bpp) format. It supports 2 distinct colors (for example: black and white). Each pixel is an 1-bit index into a table of up to 2 colors.
- **gcdColors_2b**: Uses the 2-bit per pixel (2bpp) format. It supports 4 distinct colors. Each pixel value is a 2-bit index into a table of up to 4 colors.
- **gcdColors_3b**: Uses the 3-bit per pixel (3bpp) format. It supports 8 distinct colors. Each pixel value is a 3-bit index into a table of up to 8 colors.
- **gcdColors_4b**: Uses the 4-bit per pixel (4bpp) format. It supports 16 distinct colors. Each pixel value is a 4-bit index into a table of up to 16 colors.
- **gcdColors_5b**: Uses the 5-bit per pixel (5bpp) format. It supports 32 distinct colors. Each pixel value is a 5-bit index into a table of up to 32 colors.
- **gcdColors_6b**: Uses the 6-bit per pixel (6bpp) format. It supports 64 distinct colors. Each pixel value is a 6-bit index into a table of up to 64 colors.
- **gcdColors_7b**: Uses the 7-bit per pixel (7bpp) format. It supports 128 distinct colors. Each pixel value is a 7-bit index into a table of up to 128 colors.
- **gcdColors_8b**: Uses the 8-bit per pixel (8bpp) format. It supports 256 distinct colors. Each pixel value is a 8-bit index into a table of up to 256 colors.

A.4.5 ColorDepth

(TSave2D_Png)

Specifies the color depth of a Portable Network Graphics (PNG) format file for TSave2D_Png component.

Syntax:

```
type
  { Defined in the pSave2D_Png unit }
  TPngColorDepth = (pcdIndex_1, pcdIndex_2, pcdIndex_4, pcdIndex_8, pcdRGB_8, pcdRGBA_8);
property ColorType: TPngColorType;
```

Description:

Color depth, also known as bit depth, pixel format, is either the number of bits used to indicate the color of a single pixel. In a Portable Network Graphics (PNG) file, the pixels can be defined by a varying number of bits. The property specifies which color depth will be used to save a barcode symbol to a Portable Network Graphics (PNG) file. It can be one of these value (defined in the pSave2D_Png unit):

- **pcdIndex_1**: Uses the 1-bit per pixel (1bpp) format. It supports 2 distinct colors (for example: black and white) and stores 8 pixels per 1 byte. Each pixel is an 1-bit index into a table of up to 2 colors.

- **pcdIndex_2**: Uses the 2-bit per pixel (2bpp) format. It supports 4 distinct colors and stores 4 pixels per 1 byte. Each pixel value is a 2-bit index into a table of up to 4 colors.
- **pcdIndex_4**: Uses the 4-bit per pixel (4bpp) format. It supports 16 distinct colors and stores 2 pixels per 1 byte. Each pixel value is a 4-bit index into a table of up to 16 colors.
- **pcdIndex_8**: Uses the 8-bit per pixel (8bpp) format. It supports 256 distinct colors and stores 1 pixel per 1 byte. Each byte is an index into a table of up to 256 colors.
- **pcdRGB_8**: Uses the 24-bit per pixel (24bpp) format. It supports 16777216 distinct colors and stores 1 pixel value per 3 bytes. Each pixel value defines the red, green and blue samples of the pixel.
- **pcdRGBA_8**: Uses the 32-bit per pixel (32bpp) format. It supports 4294967296 distinct colors and stores 1 pixel per 4 byte. Each pixel value define the Alpha, Red, Green and Blue samples of the pixel.

A.4.6 CompressionLevel

(TSave2D_Png)

Specifies the compression level of a Portable Network Graphics (PNG) file for the TSave2D_Png component.

Syntax:

```
type
  { Defined in the pSave2D_Png unit }
  TPngCompressionLevel = (pclFastest, pclDefault, pclMaximum);
property CompressionLevel: TPngCompressionLevel;
```

Description:

The Portable Network Graphics (PNG) uses a non-patented lossless data compression method known as DEFLATE. The property specifies what level of compression should be used to save a barcode symbol to a Portable Network Graphics (PNG) file. It can be one of these value (defined in the pSave2D_Png unit):

- **pclFastest**: Uses the fastest compression. The compression operation should complete as quickly as possible, even if the resulting file is not optimally compressed.
- **pclDefault**: Uses the default compression level. It represents a balance between the compression ratio and compression speed.
- **pclMaximum**: Uses the maximum compression. The compression operation should be optimally compressed, even if the operation takes a longer time to complete.

A.4.7 DeviceHeightInMM

(TSave2D_Emf)

Specifies the vertical height of the reference device for an Enhanced Metafile Format (EMF) file was generated by the TSave2D_Emf component, in millimetres.

Syntax:

```
property DeviceHeightInMM: DWord;
```

Description:

The property specifies the vertical height of the reference device, in millimetres, when use the TSave2D_Emf component to save a barcode symbol to an Enhanced Metafile Format (EMF) file.

Note, the property cannot be set to zero, if you set it to zero, an ESave2DError exception occurs.

A.4.8 DeviceHeightInPixel

(TSave2D_Emf)

Specifies the vertical height of the reference device for an Enhanced Metafile Format (EMF) file was generated by the TSave2D_Emf component, in pixels.

Syntax:

```
property DeviceHeightInPixel: DWord;
```

Description:

The property specifies the vertical height of the reference device, in pixels, when use the TSave2D_Emf component to save a barcode symbol to an Enhanced Metafile Format (EMF) file.

Note, the property cannot be set to zero, if you set it to zero, an ESave2DError exception occurs.

A.4.9 DeviceHeightInUM

(TSave2D_Emf)

Specifies the vertical height of the reference device for an Enhanced Metafile Format (EMF) file was generated by the TSave2D_Emf component, in micrometres.

Syntax:

```
property DeviceHeightInUM: DWord;
```

Description:

The property specifies the vertical height of the reference device, in micrometres, when use the TSave2D_Emf component to save a barcode symbol to an Enhanced Metafile Format (EMF) file.

The DeviceHeightInUM and the DeviceWidthInUm properties add the ability to measure device surfaces in micrometers, which enhances the resolution and scalability of Enhanced Metafile Format (EMF) files. If one of them is set to zero, they will be ignored.

A.4.10 DeviceWidthInMM

(TSave2D_Emf)

Specifies the horizontal width of the reference device for an Enhanced Metafile Format (EMF) file was generated by the TSave2D_Emf component, in millimetres.

Syntax:

```
property DeviceWidthInMM: DWord;
```

Description:

The property specifies the horizontal width of the reference device, in millimetres, when use the TSave2D_Emf component to save a barcode symbol to an Enhanced Metafile Format (EMF) file.

Note, the property cannot be set to zero, if you set it to zero, an ESave2DError exception occurs.

A.4.11 DeviceWidthInPixel

(TSave2D_Emf)

Specifies the horizontal width of the reference device for an Enhanced Metafile Format (EMF) file was generated by the TSave2D_Emf component, in pixels.

Syntax:

```
property DeviceWidthInPixel: DWord;
```

Description:

The property specifies the horizontal width of the reference device, in pixels, when use the TSave2D_Emf component to save a barcode symbol to an Enhanced Metafile Format (EMF) file.

Note, the property cannot be set to zero, if you set it to zero, an ESave2DError exception occurs.

A.4.12 DeviceWidthInUM

(TSave2D_Emf)

Specifies the horizontal width of the reference device for an Enhanced Metafile Format (EMF) file was generated by the TSave2D_Emf component, in micrometres.

Syntax:

```
property DeviceWidthInUM: DWord;
```

Description:

The property specifies the horizontal width of the reference device, in micrometres, when use the TSave2D_Emf component to save a barcode symbol to an Enhanced Metafile Format (EMF) file.

The DeviceWidthInUm and the DeviceHeightInUm properties add the ability to measure device surfaces in micrometers, which enhances the resolution and scalability of Enhanced Metafile Format (EMF) files. If one of them is set to zero, they will be ignored.

A.4.13 EmbedBarcodeText

(TSave2D_Emf, TSave2D_Eps, TSave2D_Gif, TSave2D_Svg)

Specifies whether to embed the barcode text to a picture file.

Syntax:

```
property EmbedBarcodeText: Boolean;
```

Description:

The property specifies whether to embed the barcode text to a picture file. Only the Portable Network Graphics (PNG), Graphics Interchange Format (GIF) version 89a, Scalable Vector Graphics (SVG), Encapsulated PostScript (EPS), and Enhanced Metafile (EMF) formats support to embed the barcode text.

Note, the Graphics Interchange Format (GIF) version 87a doesn't support to embed the other data into the picture file, so for the TSave2D_Gif component, if the Version property is set to gv87a, the value of this property will be ignored.

Note, only the encoded barcode text will be embed to the picture file.

A.4.14 EmbedBarcodeText

(TSave2D_Png)

Specifies whether to embed the barcode text to a Portable Network Graphics (PNG) picture file for TSave2D_Png component.

Syntax:

```
type
  { Defined in the pSave2D_Png unit }
  TPngEmbedBarcodeText = (pebNone, pebText, pebCompressedText);
property EmbedBarcodeText: TPngEmbedBarcodeText;
```

Description:

The property specifies whether to embed the barcode text to a Portable Network Graphics (PNG) picture file.

This property can be one of these values (defined in the pSave2D_Png unit):

- **pebNone**: Don't embed the barcode text to the picture file.
- **pebText**: Embed the barcode text to the picture file directly.
- **pebCompressedText**: Compress the barcode text then embed it to the picture file.

Note, only the encoded barcode text will be embed to the picture file.

A.4.15 Interlaced

(TSave2D_Png, TSave2D_Gif)

Specifies whether to generate the interlaced picture file when use the TSave2D_Gif or TSave2D_Png component to save a barcode symbol.

Syntax:

```
property Interlaced: Boolean;
```

Description:

The Graphics Interchange Format (GIF) and Portable Network Graphics (PNG) allow the images is interlaced; i.e., that the order of the raster lines in its data block is not sequential. This allows a partial display of the image that can be recognized before the full image is painted. The property specifies whether to generate the interlaced images file.

A.4.16 LogicUnitsPerInch

(TSave2D_Wmf)

Specifies the number of logical units per inch when use the TSave2D_Wmf component to save a barcode symbol in placeable Windows Metafile Format (WMF).

Syntax:

```
property LogicUnitsPerInch: Word;
```

Description:

Specifies the number of logical units per inch for a placeable Windows Metafile Format (WMF) file, in order to represent the image. This value can be used to scale a placeable image.

By convention, an image is considered to be recorded at 1440 logical units (twips) per inch. Thus, a value of 720 specifies that the image SHOULD be rendered at twice its normal size, and a value of 2880 specifies that the image SHOULD be rendered at half its normal size.

Note, this property must be set if this Placeable property is set to true. If this Placeable property is set to false, the value of this property will be ignored.

A.4.17 OriginalLogo

(TSave2D_Svg)

Determines how to save the logo picture in the target SVG file.

Syntax:

```
property OriginalLogo: Boolean default True;
```

Description:

If a TLogo2D component is linked to the TBarcode2D component which the TSave2D_Svg component is linked to, the logo picture can be saved in the target SVG file. The property determines how to save the logo picture in the target SVG file.

If the property is set true, the logo picture will be saved in the target SVG file using original format and size, otherwise, the logo picture will be saved using the BMP format and using the size which speified by the Width and Height properties of the TLogo2D component.

A.4.18 Placeable

(TSave2D_Wmf)

Specifies whether to generate a placeable Windows Metafile Format (WMF) file when use theTSave2D_Wmf component to save a barcode symbol.

Syntax:

```
property Placeable: Boolean;
```

Description:

Original Windows Metafile Format (WMF) were device-specific; that is, the graphical images they contained would only be rendered correctly if played back on the output device for which they were recorded. To overcome this limitation, "placeable" Windows Metafile Format (WMF) were developed. The property specifies whether to generate a placeable Windows Metafile Format (WMF) file.

Note, the propertyLogicUnitsPerInch must be set if this property is set to true. If this property is set to false, the value of LogicUnitsPerInch property will be ignored.

A.4.19 Progressive

(TSave2D_Jpg)

Determines whether a JPEG image can be progressively displayed when it is decompressed, when you use theTSave2D_Jpg component to save a barcode symbol in JPEG format.

Syntax:

```
property Progressive: Boolean;
```

Description:

Use Progressive to compress an image when saving barcode symbol in JPEG format, so that it can be progressively displayed when it is decompressed.

A.4.20 Quality

(TSave2D_Jpg)

Indicates the trade-off ratio between the image quality and the file size when use the TSave2D_Jpg component to save a barcode symbol.

Syntax:

```
type
  { Defined in the pSave2DJpg unit }
  TJPEGQuality = 1..100;
property Quality: TJPEGQuality default 100;
```

Description:

Use the Quality property to set the compression quality of the JPEG image when saving barcode symbol to a jpeg image. Higher compression results in a poorer picture quality, but a smaller file size. The higher the property value (up to a maximum of 100), the better the image quality, but the larger the file size. The lower the property value (to a minimum of 1), the smaller the resulting file size, but at the expense of picture quality.

A.4.21 SpaceOpacity

(TSave2D_Png, TSave2D_Svg)

Specifies the transparency-level of entire barcode symbol.

Syntax:

```
property SpaceOpacity: Single;
```

Description:

For the TSave2D_Png and TSave2D_Svg components, when save a barcode symbol to a picture file, the property specifies the transparency-level of bakcground color (spaces or light modules, and quiet zones) if the Inversed property is set to false, otherwise, it specifies the transparency-level of foreground color (bars or dark modules). Where 1 is not transparant at all, 0.5 is 50% see-through, and 0 is completely transparent.

The property defaults to 1, meaning not transparant at all.

A.4.22 Version

(TSave2D_Gif)

Specifies the version of a Graphics Interchange Format (GIF) file for the TSave2D_Gif component.

Syntax:

```
type
  { Defined in the pSave2D_Gif unit }
  TGifVersion = (gv87a, gv89a);
property Version: TGifVersion;
```

Description:

There are two versions of the Graphics Interchange Format (GIF): versions 87a and 89a. The property specifies which version will be used to save a barcode symbol to a Graphics Interchange Format (GIF) file. It can be one of these values (defined in the pSave2D_Gif unit):

- **gv87a**: Uses the version 87a, the original version of the Graphics Interchange Format (GIF). It doesn't support to embed the other data into the picture file, so the value of EmbedBarcodeText property will be ignored if you use the version.
- **gv89a**: Uses the version 89a, the enhanced version of the Graphics Interchange Format (GIF). If you use the version, you can set the EmbedBarcodeText property to true to embed the encoded barcode text into the Graphics Interchange Format (GIF) picture file.

A.4.23 XPPM

(TSave2D_Bmp)

Specifies the horizontal resolution of a Bitmap image (BMP) file for TSave2D_Bmp component, in pixels per meter.

Syntax:

```
property XPPM: Integer;
```

Description:

The property specifies the horizontal resolution of the bitmap in pixels per meter. The value is used to help a bitmap image (BMP) file reader choose a proper horizontal resolution when printing or displaying the Bitmap image (BMP) file.

A.4.24 YPPM

(TSave2D_Bmp)

Specifies the vertical resolution of a Bitmap image (BMP) file for TSave2D_Bmp component, in pixels per meter.

Syntax:

```
property YPPM: Integer;
```

Description:

The property specifies the vertical resolution of the bitmap in pixels per meter. The value is used to help a bitmap image (BMP) file reader choose a proper vertical resolution when printing or displaying the Bitmap image (BMP) file.

Annex B. Methods

B.1 TBarcode2D

B.1.1 Assign

Copies a barcode component from another barcode component.

Syntax:

```
procedure Assign(Source: TPresistent); override;
```

Description:

If the Source parameter is an object created from a subclass of TBarcode2D component class, and the class is same to current barcode component class, Assign copies all property values and event handles except the Locked and the Image properties from the source barcode component to current one. If Source is any other type of object, an EBarcode2DError exception occurs.

Parameters:

- **Source**: TPersistent; Specifies the source object.

B.1.2 Clear

Erases current barcode symbol in the TImage, TQRImage, or TQRGzImage control that's specified by the Image property.

Syntax:

```
function Clear(UseSpaceColor: Boolean = false): Boolean; virtual;
```

Description:

The method erases the barcode symbol without erasing the background around it from the TImage, TQRImage, or TQRGzImage control that's specified by the Image property.

If the quiet zones are displayed (the ShowQuietZone property is set to true), they are included in the barcode symbol and will be erased too. For the TBarcode2D_Code16K barcode components, the leading quiet zone and trailing quiet zone are included always, even if the ShowQuietZone property is set to false.

Parameters:

- **UseSpaceColor**: Boolean; Specifies whether the space color that's specified by SpaceColor property is used to erase the barcode symbol. If it's set to false, the current brush color of the TImage, TQRImage, or TQRGzImage control will be used. If the parameter isn't provided, it's default to false.

Return:

- If the method succeeds, the return value is true.

- For the Delphi/C++ Builder 2007 or early, if the length of Barcode property value is invalid, or there is any invalid character in the Barcode property value, the return value is false. Corresponding to the OnInvalidLength event or the OnInvalidChar event will occur.

  For Delphi/C++ Builder 2009 or later, if you use the Barcode property to specify the barcode text, when the length of Barcode property value is invalid, or there is any invalid character in the Barcode property value, the return value is false. Corresponding to the OnInvalidLength event or the OnInvalidChar event will occur. If you use the Data property to specify the barcode text, when the length of Data property value is invalid, or there is any invalid character in the Data property value, the return value is false. Corresponding to the OnInvalidDataLength event or the OnInvalidDataChar event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, and the length of its Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value is invalid, the return value is false too, corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur. If there is any invalid character in the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value, the return value is false too, corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

B.1.3 CopyToClipboard

Copies a barcode symbol to the system clipboard. There are several different overloading methods, Syntax 1, Syntax 2, and Syntax 3 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Copies the barcode symbol that is specified by the properties of this barcode component to the system clipboard.
- **Syntax 2**: Copies the barcode symbol that is specified by the parameters of this method to the system clipboard. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the CopyToClipboard (Syntax 3) overloading method and specify the converted string in its Data parameter. If you want to encode a block of binary (bytes) data, please use the CopyToClipboard (Syntax 3) overloading method.

- **Syntax 3**: Copies the barcode symbol that is specified by the parameters of this method to the system clipboard. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you want to encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the CopyToClipboard (Syntax 2) overloading method.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be copied to the system clipboard.

B.1.3.1 CopyToClipboard - Syntax 1

Copies a barcode symbol to the system clipboard. The barcode symbol is specified by the properties of this barcode component.

Syntax:

```
function CopyToClipboard(Module: Integer = 0; Angle: Integer = -1): Integer; overload;
  virtual;
```

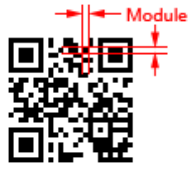
Description:

Copies current barcode symbol that is specified by the properties of this barcode component to the system clipboard.

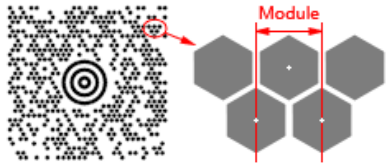
If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol.

Parameters:

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction. It defaults to 0 if the Module parameter is not provided, and the value of Module property will be used.
    - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



    - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:

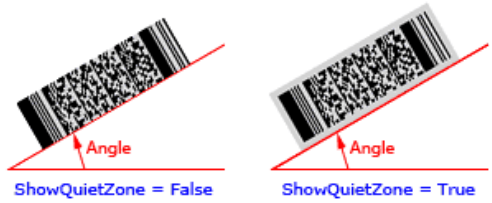


    - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

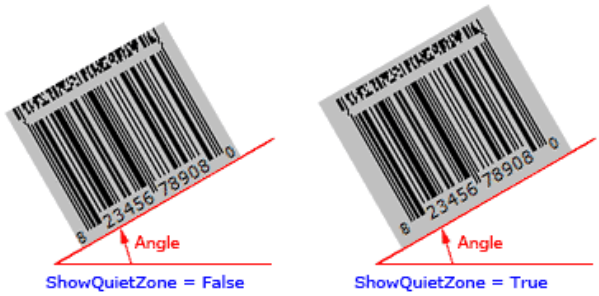
    - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone property about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone property to specify whether to represent these quiet zones. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



The parameter defaults to -1 if the Angle is not provided, and the barcode symbol will be rotated based on the value of the Orientation property:

  - boLeftRight: 0 degrees
  - boRightLeft: 180 degrees
  - boTopBottom: 270 degrees
  - boBottomTop: 90 degrees

  If you want to use the -1 degrees, the 359 degrees can be used instead.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property is invalid, the return valus is -1 or -65537. Corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the

return valus is -65537.

For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character. Corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.3.2 CopyToClipboard - Syntax 2

Copies a barcode symbol to the system clipboard. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function CopyToClipboard(Barcode: string; BarColor, SpaceColor: TColor; ShowQuietZone:
  Boolean; Module: Integer; Angle: Integer = 0): Integer; overload; virtual;
```

Description:

Copies a barcode symbol that is specified by the parameters of this method to the system clipboard.

If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol.

Parameters:

- **Barcode**: String; Specifies the barcode text. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the CopyToClipboard (Syntax 3) overloading method and specify the converted string in its Data parameter. If you want to encode a block of binary (bytes) data, please use the CopyToClipboard (Syntax 3) overloading method.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **BarColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

  If the Inversed property is set to true, it specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "BarColor" property.

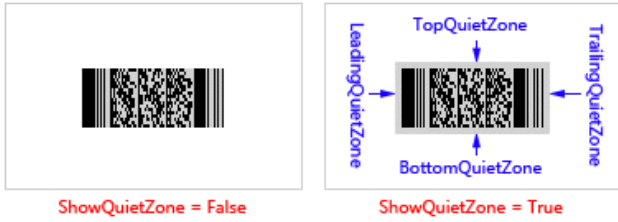
- **SpaceColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  If the Inversed property is set to true, it specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

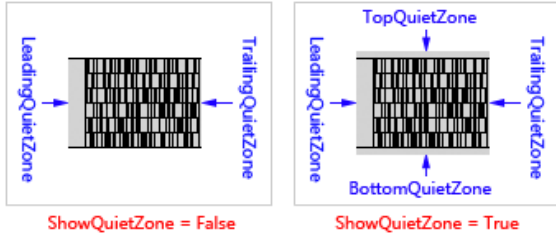
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "SpaceColor" property.

- **ShowQuietZone**: Boolean; Specifies whether to represent the leading, trailing, top and bottom quiet zones. If the parameter value is set to true, these quiet zones are represented. Otherwise, they don't be represented. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

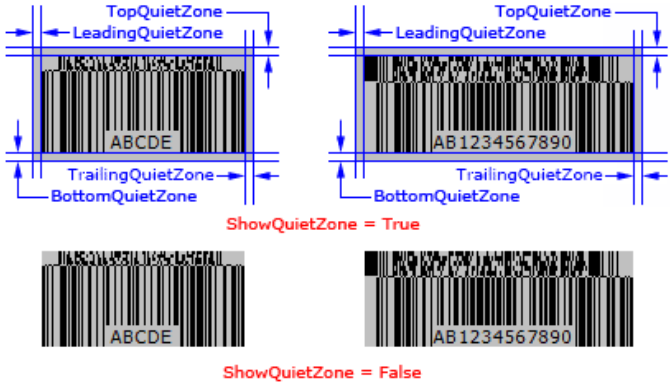


  For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be represented always, even if the ShowQuietZone parameter value is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

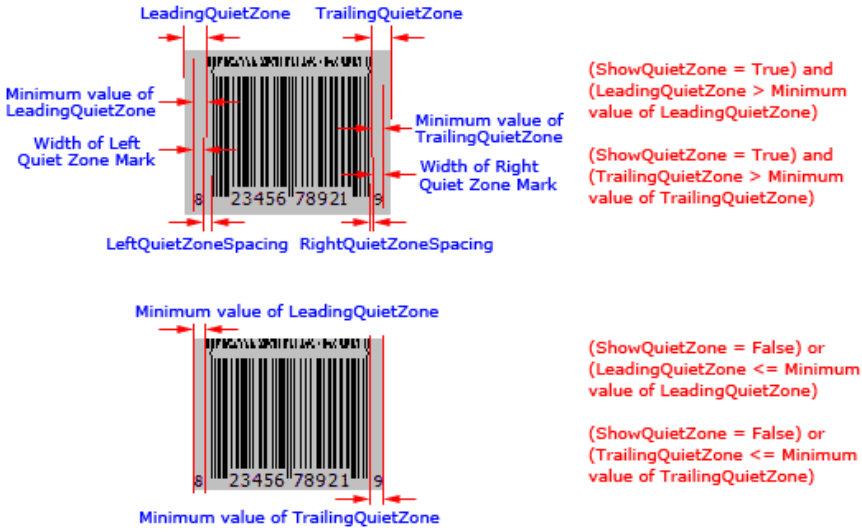


  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for

entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

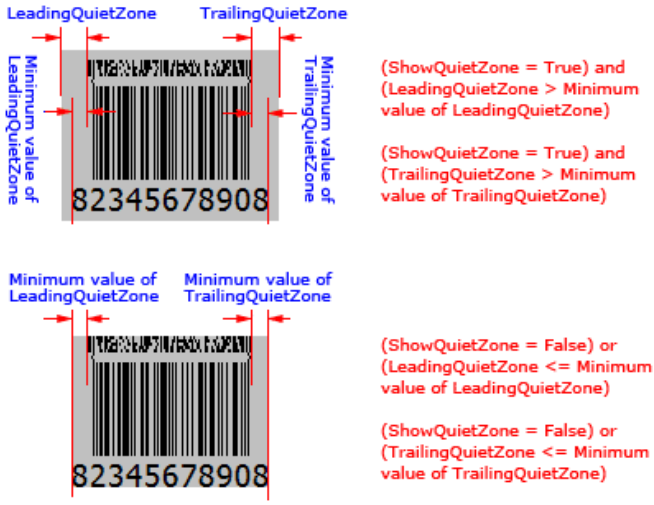


For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver

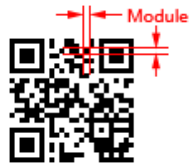
in order to accentuate the quiet zones):



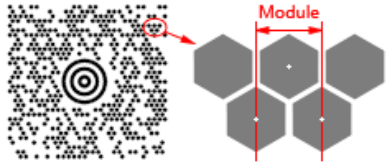
In general, the Inversed property is set to false. In this case, these quiet zones are represented using the color specified by SpaceColor parameter. If the Inversed property is set to true, these quiet zones are represented using the color specified by BarColor parameter.

See also the "ShowQuietZone" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



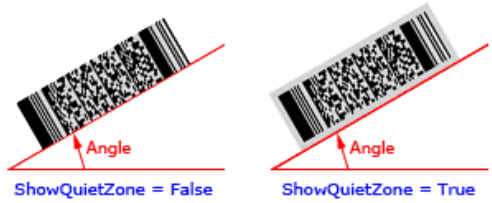
  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter

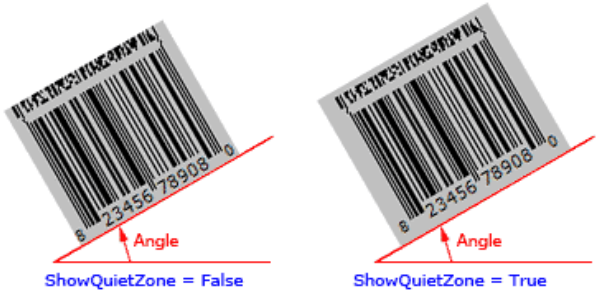
specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set,

the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.3.3 CopyToClipboard - Syntax 3

Copies a barcode symbol to the system clipboard. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function CopyToClipboard(Data: AnsiString; BarColor, SpaceColor: TColor; ShowQuietZone:
  Boolean; Module: Integer; Angle: Integer = 0): Integer; overload; virtual;
```

Description:

Copies a barcode symbol that is specified by the parameters of this method to the system clipboard.

If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol.

Parameters:

- **Data**: AnsiString; Specifies the barcode text. It is of type AnsiString.

  For the Delphi/C++ Builder 2009 or later, you can use the method to encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you want to encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the CopyToClipboard (Syntax 2) overloading method.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **BarColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

  If the Inversed property is set to true, it specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

See also the "BarColor" property.

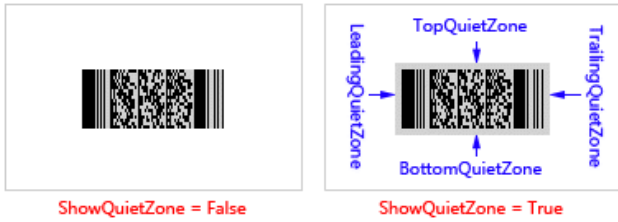
- **SpaceColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  If the Inversed property is set to true, it specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

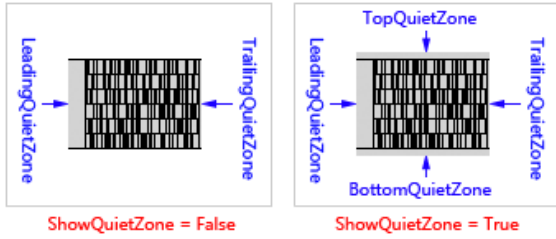
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "SpaceColor" property.

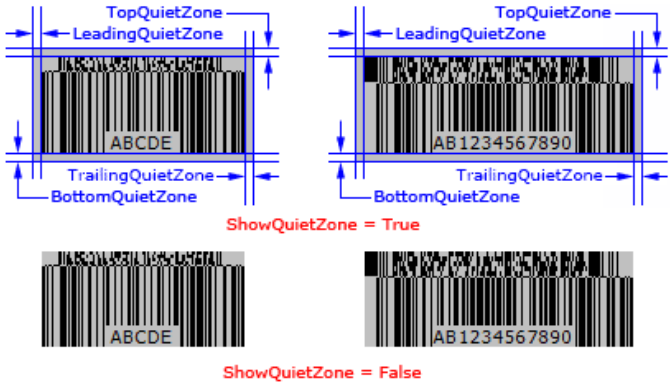
- **ShowQuietZone**: Boolean; Specifies whether to represent the leading, trailing, top and bottom quiet zones. If the parameter value is set to true, these quiet zones are represented. Otherwise, they don't be represented. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



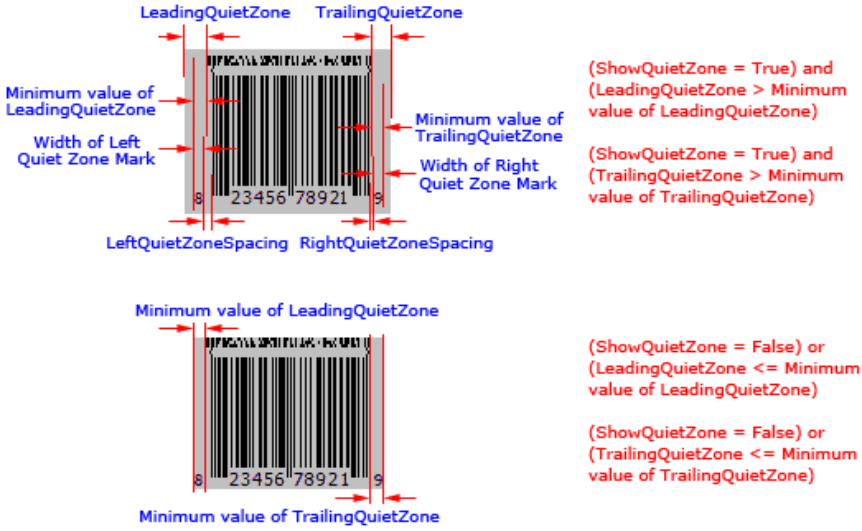
  For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be represented always, even if the ShowQuietZone parameter value is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



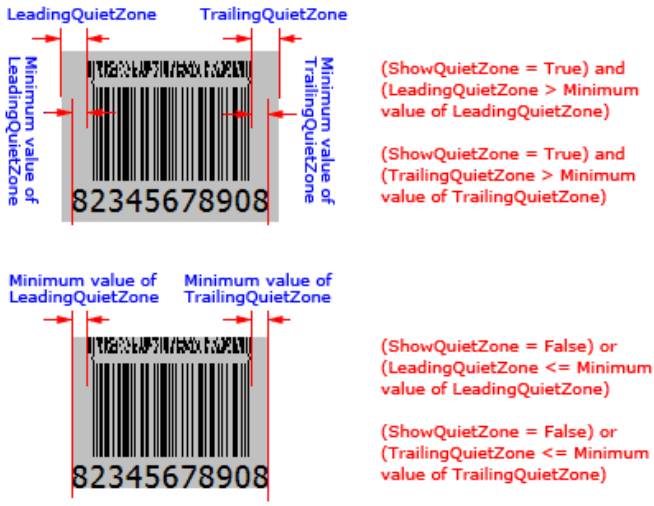
  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



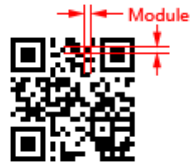
For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



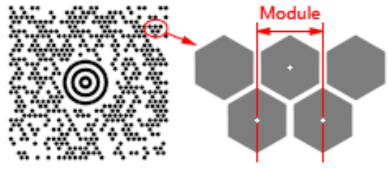
In general, the Inversed property is set to false. In this case, these quiet zones are represented using the color specified by SpaceColor parameter. If the Inversed property is set to true, these quiet zones are represented using the color specified by BarColor parameter.

See also the "ShowQuietZone" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:

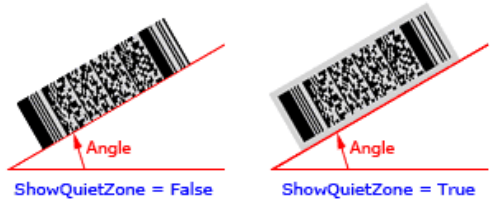


  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

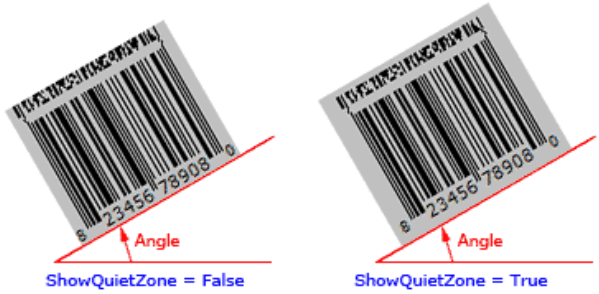
  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the

17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

B.1.4 Create

Creates and initializes a barcode component.

Syntax:

```
constructor Create(Owner: TComponent); override;
```

Description:

Call Create to instantiate a barcode object at runtime. Barcode components added at design time are created automatically.

Parameters:

- **Owner**: TComponent; It is the component that is responsible for freeing the barcode component instance. Typically, this is the form. It becomes the value of the Owner property.

B.1.5 Destroy

Disposes of the instance of the barcode object.

Syntax:

```
destructor Destroy; override;
```

Description:

Destroy is the destructor for a barcode object.

Do not call the destructor directly in an application. Instead, call Free. The Free verifies that the barcode object is not nil before it calls Destroy.

B.1.6 Draw

Redraws current barcode symbol in the TImage, TQRImage, or TQRGzImage control that's specified by the Image property.

Syntax:

```
function Draw(ReEncode: Boolean = False): Boolean; virtual;
```

Description:

The method redraws the barcode symbol to the TImage, TQRImage, or TQRGzImage control that's specified by the Image property. The barcode symbol is specified in the properties of this barcode component.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be drawn in the TImage, TQRImage, or TQRGzImage control.

Parameter:

- **ReEncode**: Boolean; Normally, the barcode text will not be encoded again when call the method, the previous encoded data will be used to present the barcode symbol. If the parameter is set to true, the barcode text will be encoded again before present the barcode symbol. It defaults to false if the parameter is not provided.

  If you changed the OnEncode event handle, please call the method and set the parameter to true (please call the Clear method firstly).

Return:

- If the method succeeds, the return value is true.
- For the Delphi/C++ Builder 2007 or early, if the length of Barcode property value is invalid, or there is any invalid character in the Barcode property value, the return value is false. Corresponding to the OnInvalidLength event or the OnInvalidChar event will occur.

  For Delphi/C++ Builder 2009 or later, if you use the Barcode property to specify the barcode text, when the length of Barcode property value is invalid, or there is any invalid character in the Barcode property value, the return value is false. Corresponding to the OnInvalidLength event or the OnInvalidChar event will occur. If you use the Data property to specify the barcode text, when the length of Data property value is invalid, or there is any invalid character in the Data property value, the return value is false. Corresponding to the OnInvalidDataLength event or the OnInvalidDataChar event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, and the length of its Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value is invalid, the return value is false too, corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur. If there is any invalid character in the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value, the return value is false too, corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

B.1.7 DrawTo

Draws a barcode symbol on the specified canvas. There are several different overloading methods, Syntax 1, Syntax 2, and Syntax 3 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Draws the barcode symbol that is specified by the properties of this barcode component.
- **Syntax 2**: Draws the barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the DrawTo (Syntax 3) overloading method

and specify the converted string in its Data parameter. If you want to encode a block of binary (bytes) data, please use the DrawTo (Syntax 3) overloading method.

- **Syntax 3**: Draws the barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you want to encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the DrawTo (Syntax 2) overloading method.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be drawn on the specified canvas.

B.1.7.1 DrawTo - Syntax 1

Draws a barcode symbol on the specified canvas. The barcode symbol is specified by the properties of this barcode component.

Syntax:

```
function DrawTo(Canvas: TCanvas; Left, Top: Integer; Module: Integer = 0; Angle: Integer
  = -1; HDPI: Integer = 0; VDPI: Integer = 0): Integer; overload; virtual;
```

Description:

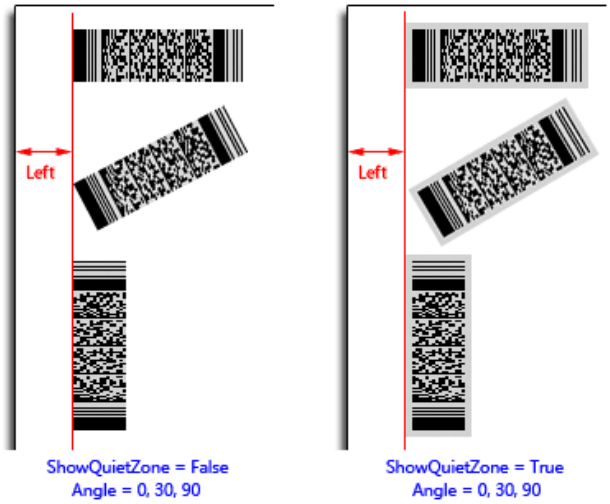
On the specified canvas, draws current barcode symbol that is specified by the properties of this barcode component.

Parameters:

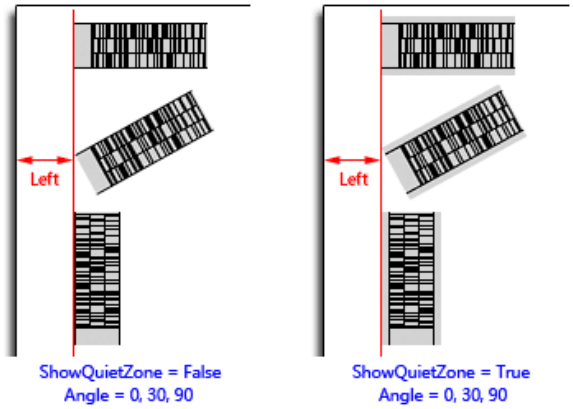
- **Canvas**: TCanvas; Specifies target canvas to represent the barcode symbol on it.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be drawn on the specified canvas.

- **Left**: Integer; Specifies the margin between the rotated barcode symbol and the left side of the canvas in logical dots or pixels in the horizontal direction. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

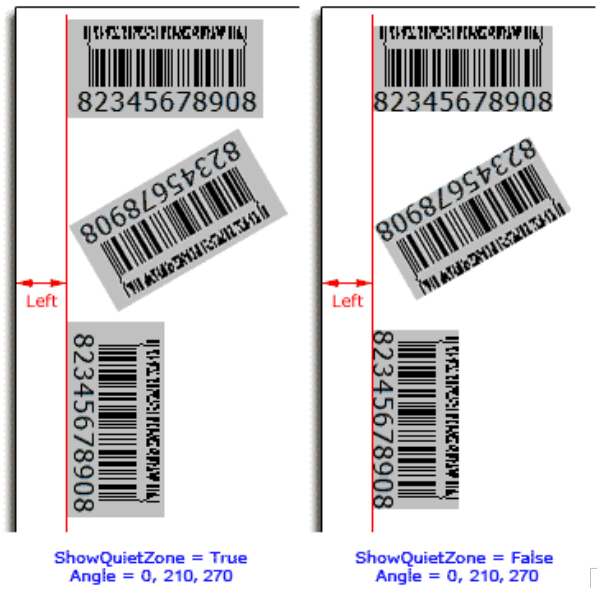


For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



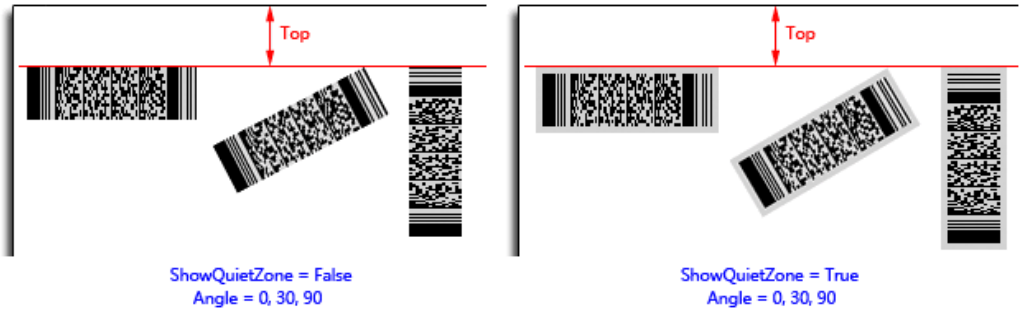
For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the left margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the left margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

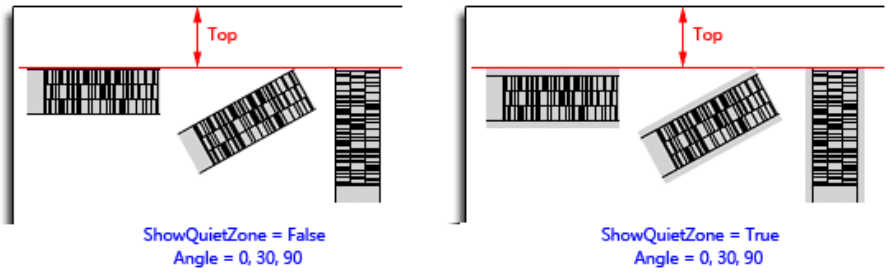


See also the "LeftMargin" property.

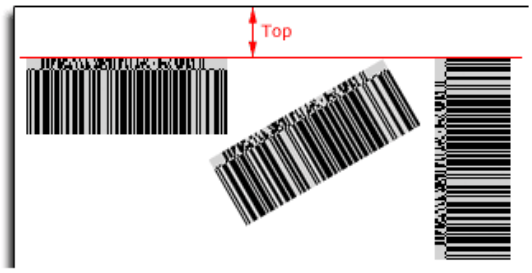
- **Top**: Integer; Specifies the margin between the rotated barcode symbol and the top side of the canvas in logical dots or pixels in the vertical direction. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



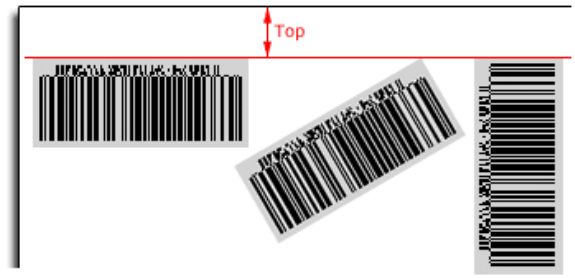
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the top margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



ShowQuietZone = False
Angle = 0, 30, 90



ShowQuietZone = True
Angle = 0, 30, 90

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the top margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

ShowQuietZone = False
Angle = 180, 210, 90

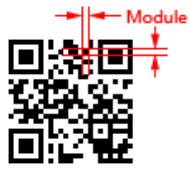
ShowQuietZone = True
Angle = 180, 210, 90

ShowQuietZone = False
Angle = 180, 210, 90

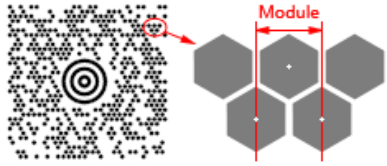
ShowQuietZone = True
Angle = 180, 210, 90

See also the "TopMargin" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction. It defaults to 0 if the Module parameter is not provided, and the value of Module property will be used.
    - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



- For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:

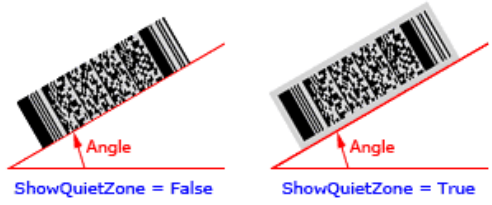


- For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

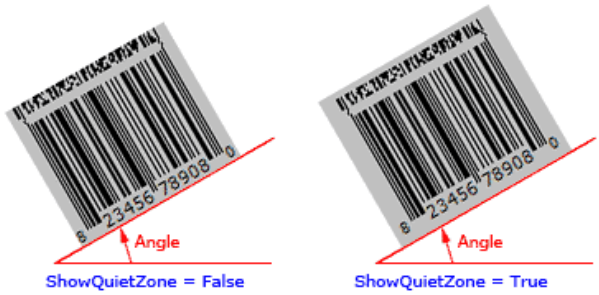
- For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone property about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone property to specify whether to represent these quiet zones. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



The parameter defaults to -1 if the Angle is not provided, and the barcode symbol will be rotated based on the value of the Orientation property:

- boLeftRight: 0 degrees
- boRightLeft: 180 degrees
- boTopBottom: 270 degrees
- boBottomTop: 90 degrees

If you want to use the -1 degrees, the 359 degrees can be used instead.

- **HDPI:** Integer, Specifies the horizontal resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the HDPI is not provided, and the physical horizontal resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROP map mode, and the horizontal units or scaling is changed, the parameter is required in order to represent the correct barcode symbol when the symbol is rotated.
- **VDPI:** Integer, Specifies the vertical resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the VDPI is not provided, and the physical vertical resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROPIC m mode, and the vertical units or scaling is changed, the parameter is required in order to represent the correct barcode symbol when the symbol is rotated.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property is invalid, the return valus is -1 or -65537. Corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.
- If there is any invalid character in the the barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character. Corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid

character. For example, the return value 65537 denotes that the first character is invalid character).

For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.7.2 DrawTo - Syntax 2

Draws a barcode symbol on the specified canvas. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function DrawTo(Canvas: TCanvas; Barcode: string; BarColor, SpaceColor: TColor;
  ShowQuietZone: Boolean; Left, Top, Module: Integer; Angle: Integer = 0; HDPI: Integer =
  0; VDPI: Integer = 0): Integer; overload; virtual;
```

Description:

On the specified canvas, draws a barcode symbol that is specified by the parameters of this method.

Parameters:

- **Canvas**: TCanvas; Specifies target canvas to represent the barcode symbol on it.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be drawn on the specified canvas.

- **Barcode**: String; Specifies the barcode text. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the DrawTo (Syntax 3) overloading method and specify the converted string in its Data parameter. If you want to encode a block of binary (bytes) data, please use the DrawTo (Syntax 3) overloading method.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **BarColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

  If the Inversed property is set to true, it specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

See also the "BarColor" property.

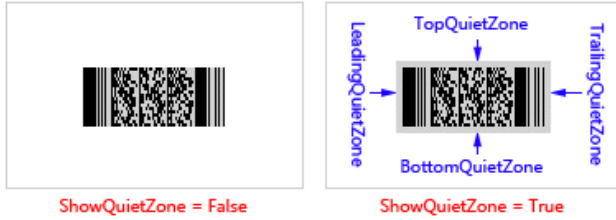
- **SpaceColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  If the Inversed property is set to true, it specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

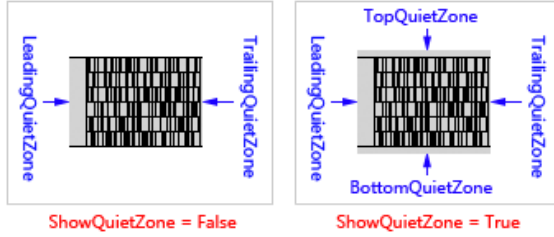
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "SpaceColor" property.

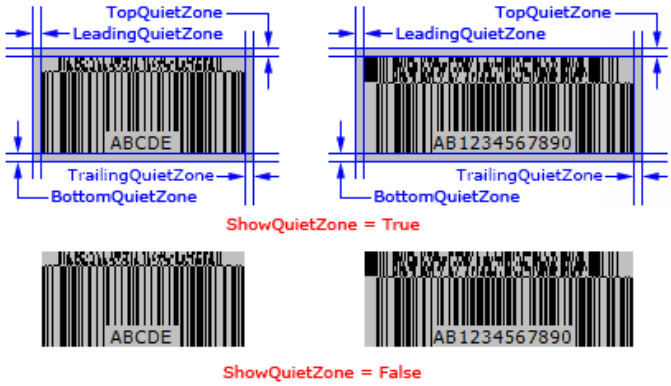
- **ShowQuietZone**: Boolean; Specifies whether to represent the leading, trailing, top and bottom quiet zones. If the parameter value is set to true, these quiet zones are represented. Otherwise, they don't be represented. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



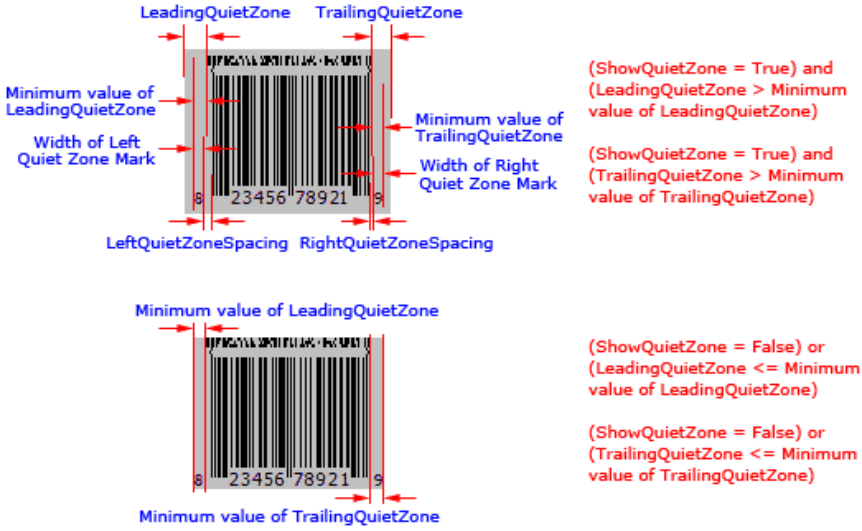
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be represented always, even if the ShowQuietZone parameter value is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



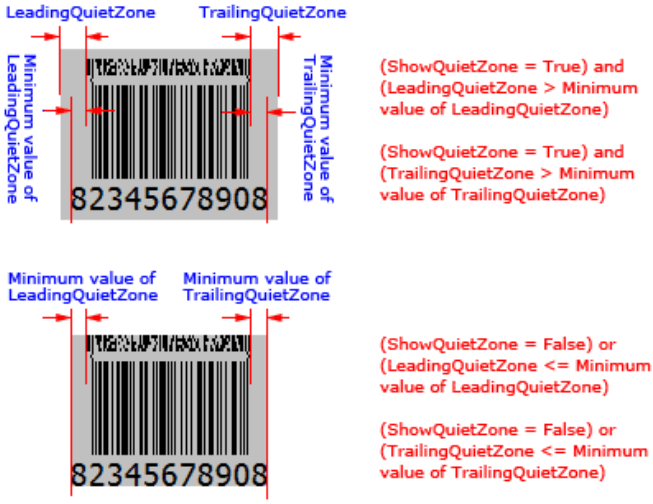
For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



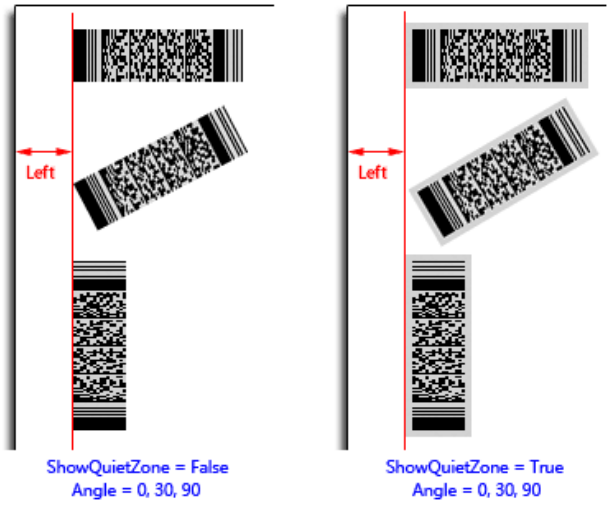
For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



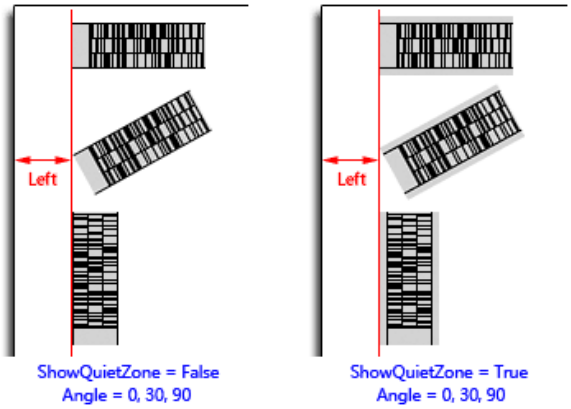
In general, the Inversed property is set to false. In this case, these quiet zones are represented using the color specified by SpaceColor parameter. If the Inversed property is set to true, these quiet zones are represented using the color specified by BarColor parameter.

See also the "ShowQuietZone" property.

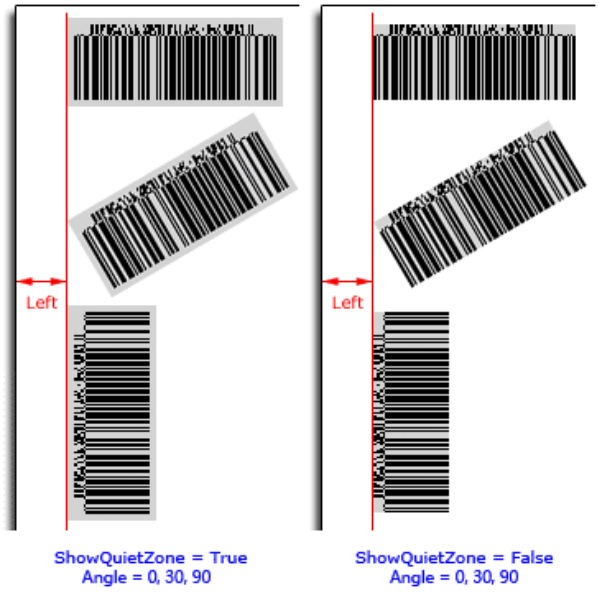
- **Left**: Integer; Specifies the margin between the rotated barcode symbol and the left side of the canvas in logical dots or pixels in the horizontal direction. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



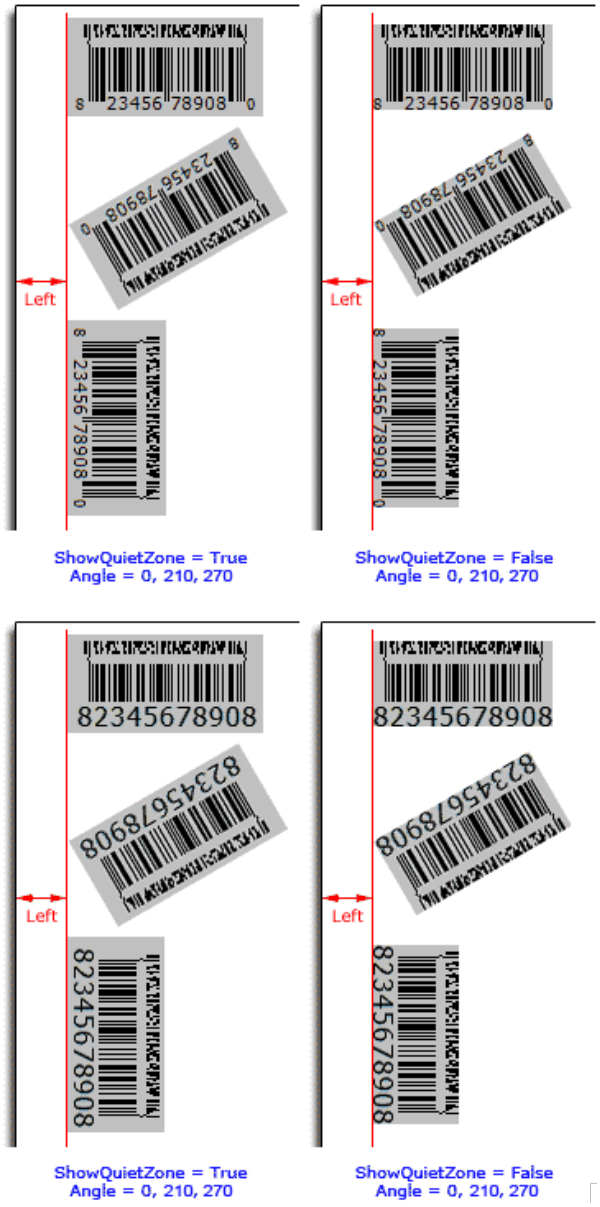
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the left margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

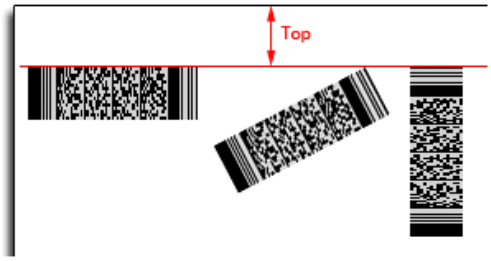


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the left margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

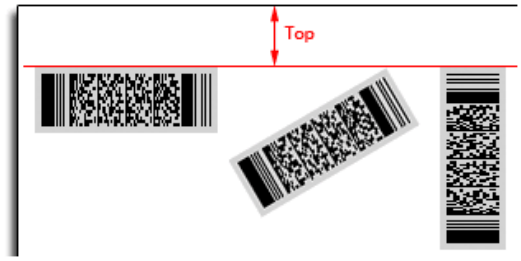


See also the "LeftMargin" property.

- **Top**: Integer; Specifies the margin between the rotated barcode symbol and the top side of the canvas in logical dots or pixels in the vertical direction. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

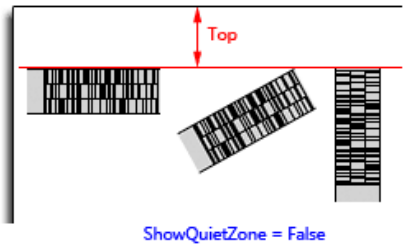


ShowQuietZone = False
Angle = 0, 30, 90

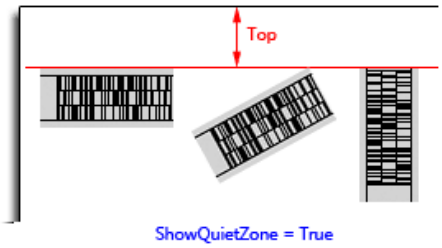


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

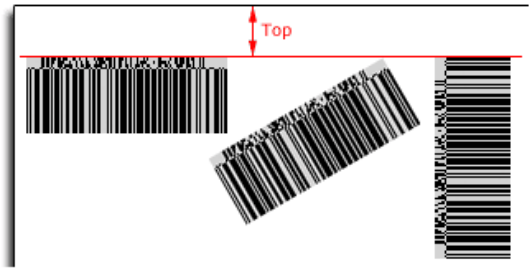


ShowQuietZone = False
Angle = 0, 30, 90

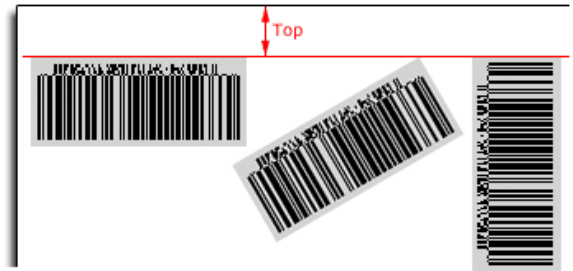


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the top margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

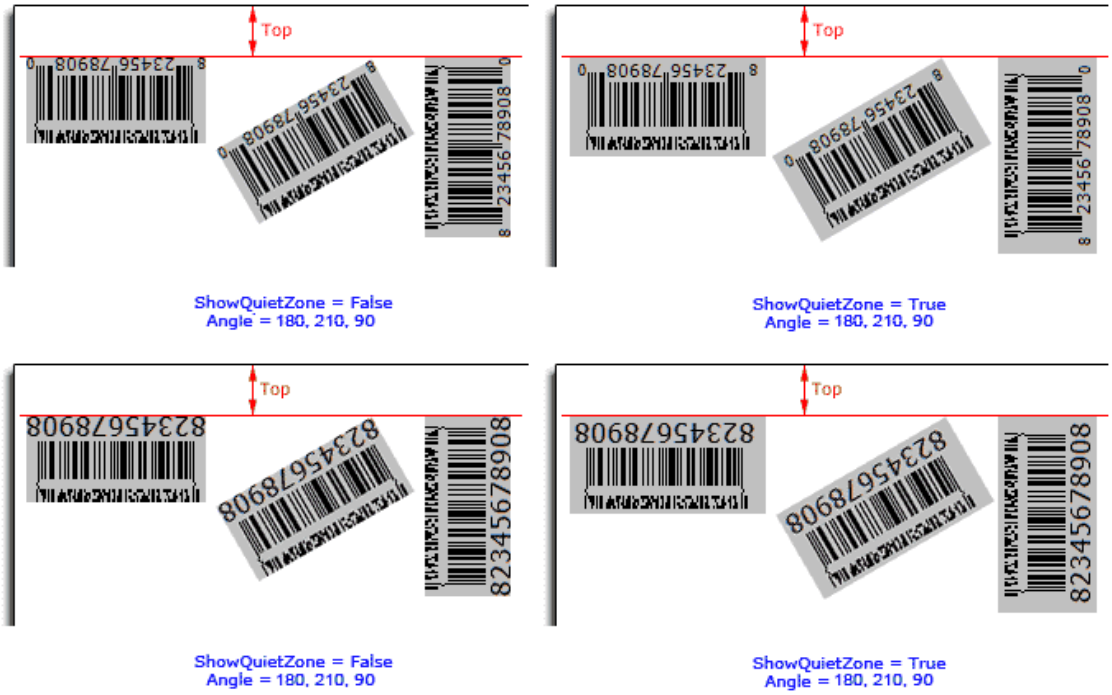


ShowQuietZone = False
Angle = 0, 30, 90



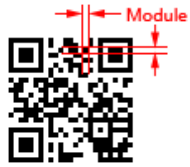
ShowQuietZone = True
Angle = 0, 30, 90

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the top margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

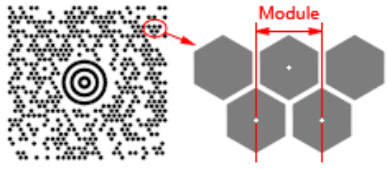


See also the "TopMargin" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



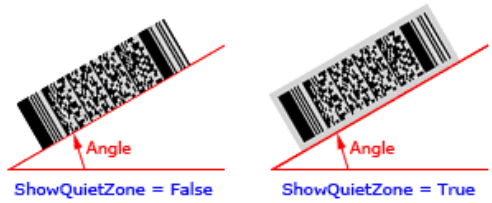
  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the

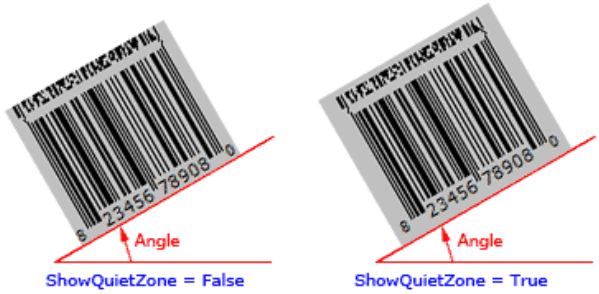
2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **HDPI:** Integer, Specifies the horizontal resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the HDPI is not provided, and the physical horizontal resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROP map mode, and the horizontal units or scaling is changed, the parameter is required in order to represent the correct barcode symbol when the symbol is rotated.

- **VDPI:** Integer, Specifies the vertical resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the VDPI is not provided, and the physical vertical resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROPIC m mode, and the vertical units or scaling is changed, the parameter is required in order to represent the correct barcode symbol when the symbol is rotated.

Return:

- If the method succeeds, the return value is zero.

- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.7.3 DrawTo - Syntax 3

Draws a barcode symbol on the specified canvas. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function DrawTo(Canvas: TCanvas; Data: AnsiString; BarColor, SpaceColor: TColor;
  ShowQuietZone: Boolean; Left, Top, Module: Integer; Angle: Integer = 0; HDPI: Integer =
  0; VDPI: Integer = 0): Integer; overload; virtual;
```

Description:

On the specified canvas, draws a barcode symbol that is specified by the parameters of this method.

Parameters:

- **Canvas**: TCanvas; Specifies target canvas to represent the barcode symbol on it.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be drawn on the specified canvas.

- **Data**: AnsiString; Specifies the barcode text. It is of type AnsiString.

  For the Delphi/C++ Builder 2009 or later, you can use the method to encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you want to encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the DrawTo (Syntax 2) overloading method.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter

only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **BarColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

  If the Inversed property is set to true, it specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "BarColor" property.

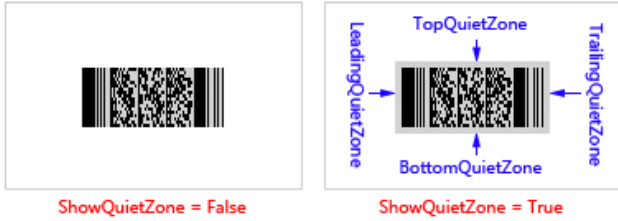
- **SpaceColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  If the Inversed property is set to true, it specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

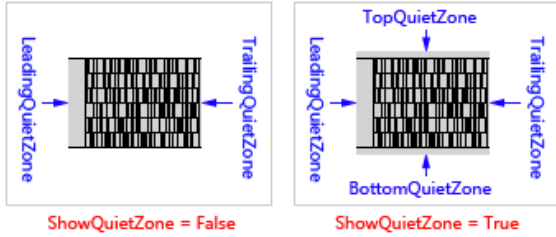
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "SpaceColor" property.

- **ShowQuietZone**: Boolean; Specifies whether to represent the leading, trailing, top and bottom quiet zones. If the parameter value is set to true, these quiet zones are represented. Otherwise, they don't be represented. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

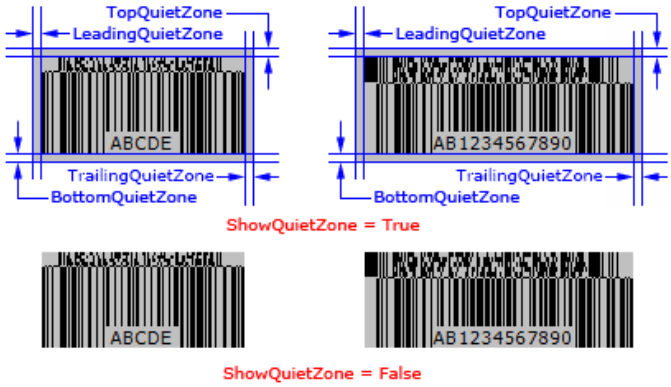


  For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be represented always, even if the ShowQuietZone parameter value is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

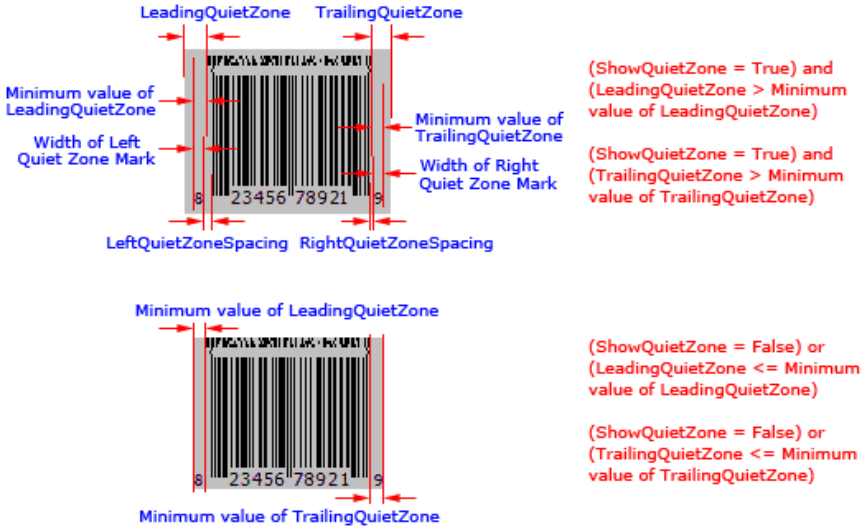


  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set,

the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

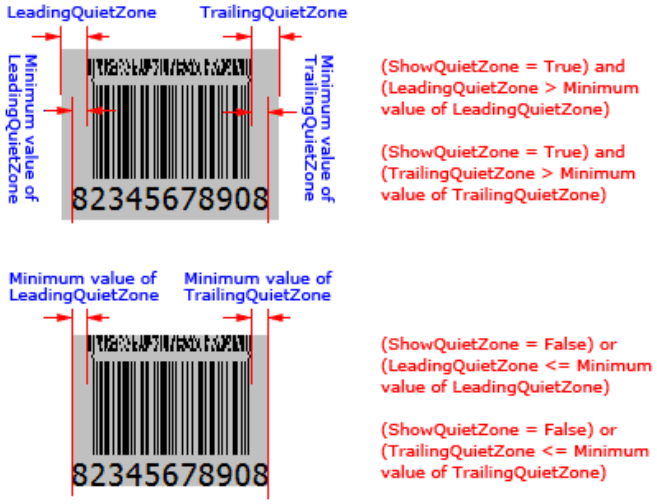


For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is

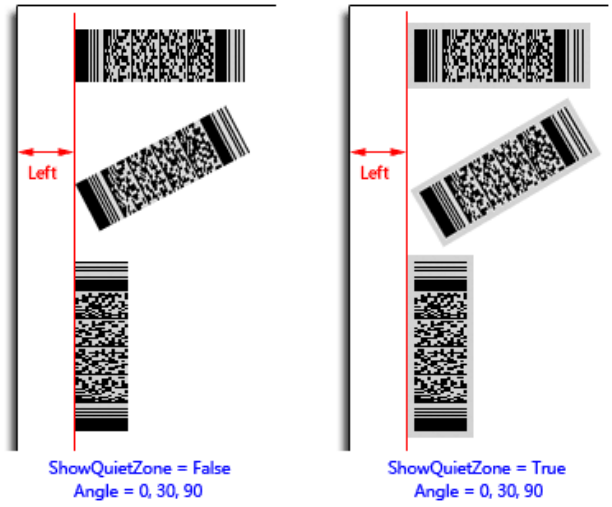
set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



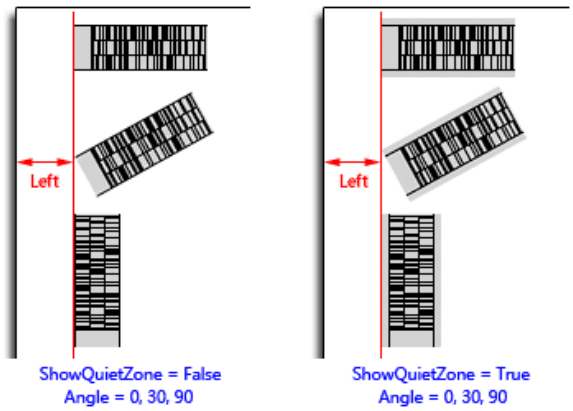
In general, the Inversed property is set to false. In this case, these quiet zones are represented using the color specified by SpaceColor parameter. If the Inversed property is set to true, these quiet zones are represented using the color specified by BarColor parameter.

See also the "ShowQuietZone" property.

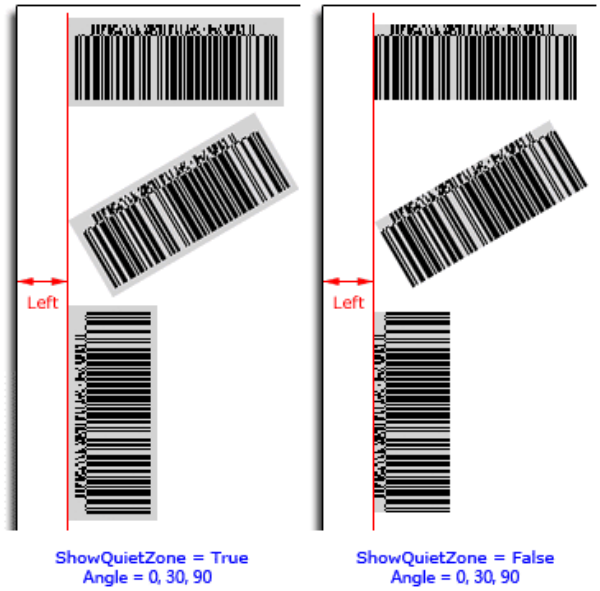
- **Left**: Integer; Specifies the margin between the rotated barcode symbol and the left side of the canvas in logical dots or pixels in the horizontal direction. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



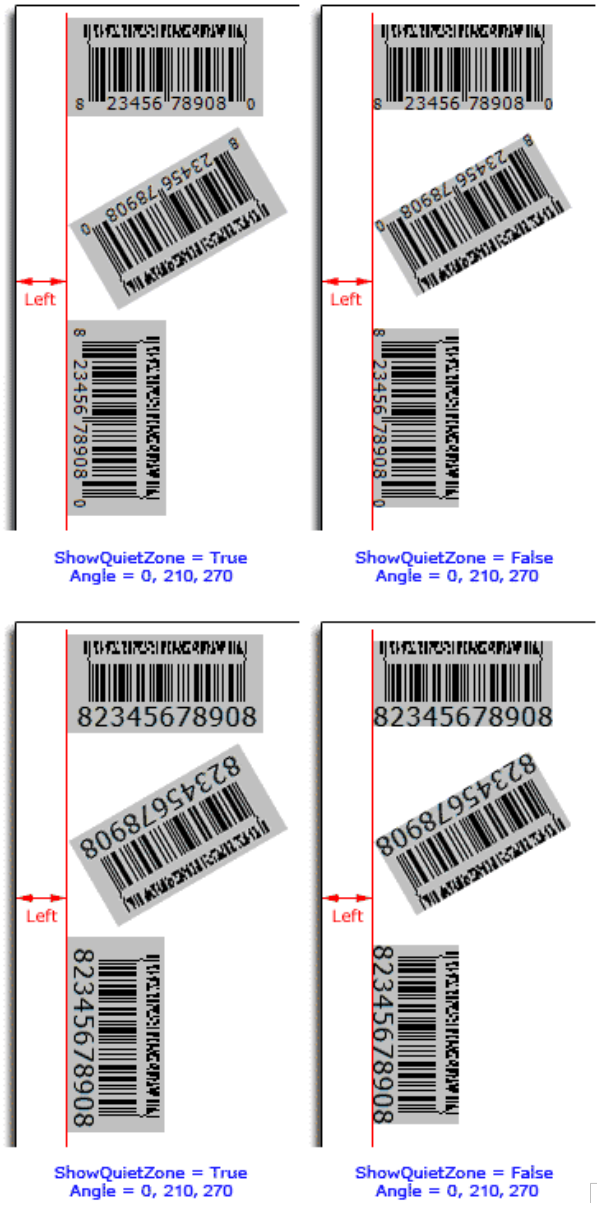
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the left margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

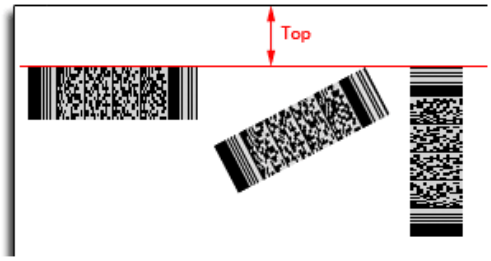


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the left margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

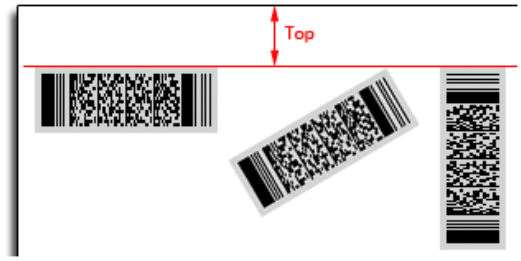


See also the "LeftMargin" property.

- **Top**: Integer; Specifies the margin between the rotated barcode symbol and the top side of the canvas in logical dots or pixels in the vertical direction. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

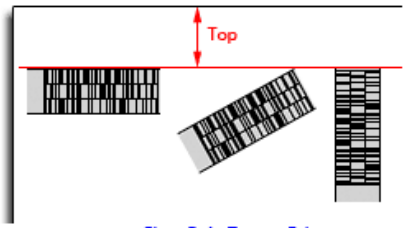


ShowQuietZone = False
Angle = 0, 30, 90

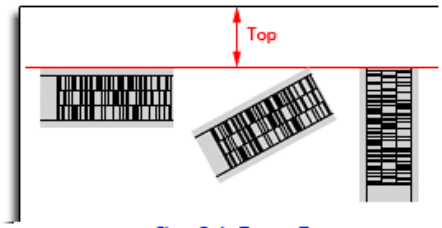


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

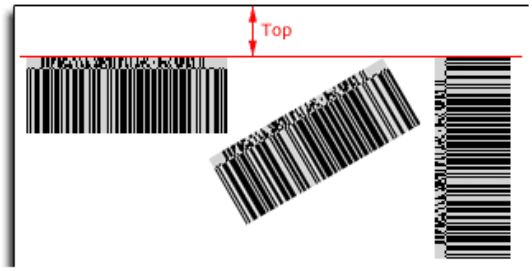


ShowQuietZone = False
Angle = 0, 30, 90

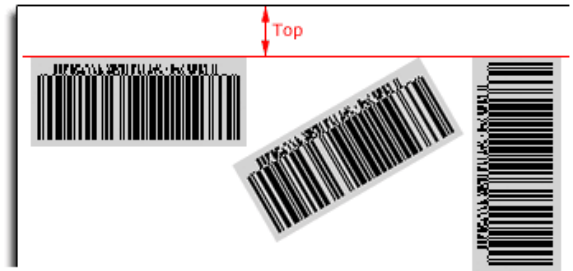


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the top margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



ShowQuietZone = False
Angle = 0, 30, 90



ShowQuietZone = True
Angle = 0, 30, 90

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the top margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

ShowQuietZone = False
Angle = 180, 210, 90

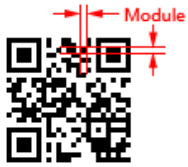
ShowQuietZone = True
Angle = 180, 210, 90

ShowQuietZone = False
Angle = 180, 210, 90

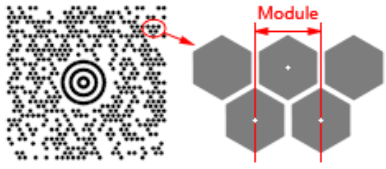
ShowQuietZone = True
Angle = 180, 210, 90

See also the "TopMargin" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



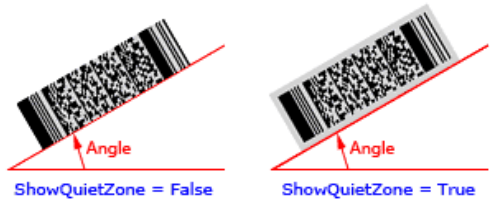
  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the

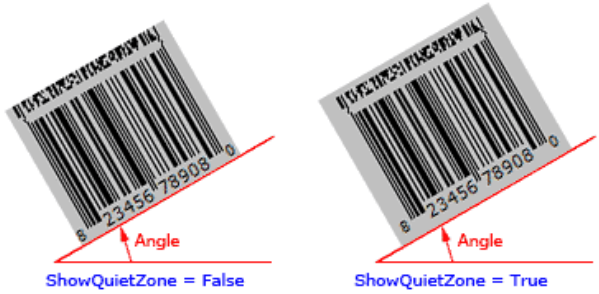
2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **HDPI:** Integer, Specifies the horizontal resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the HDPI is not provided, and the physical horizontal resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROP map mode, and the horizontal units or scaling is changed, the parameter is required in order to represent the correct barcode symbol when the symbol is rotated.

- **VDPI:** Integer, Specifies the vertical resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the VDPI is not provided, and the physical vertical resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROPIC m mode, and the vertical units or scaling is changed, the parameter is required in order to represent the correct barcode symbol when the symbol is rotated.

Return:

- If the method succeeds, the return value is zero.

- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

B.1.8 DrawToSize

Returns the horizontal width and vertical height of a rotated barcode symbol in pixels. There are several different overloading methods, Syntax 1, Syntax 2, and Syntax 3 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Returns the horizontal width and vertical height of the rotated barcode symbol that is specified by the properties of this barcode component.
- **Syntax 2**: Returns the horizontal width and vertical height of the rotated barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method if you encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the DrawToSize (Syntax 3) overloading method and specify the converted string in its Data parameter. If you encode a block of binary (bytes) data, please use the DrawToSize (Syntax 3) overloading method.

- **Syntax 3**: Returns the horizontal width and vertical height of the rotated barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method if you encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the

DrawToSize (Syntax 2) overloading method.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol.

B.1.8.1 DrawToSize - Syntax 1

Returns the horizontal width and vertical height of a rotated barcode symbol in pixels. The barcode symbol is specified by the properties of this barcode component.

Syntax:

```
function DrawToSize(var Width, Height, SymbolWidth, SymbolHeight: Integer; Module:
  Integer = 0; Angle: Integer = -1; Canvas: TCanvas = nil; HDPI: Integer = 0; VDPI:
  Integer = 0): Integer; overload; virtual;
```

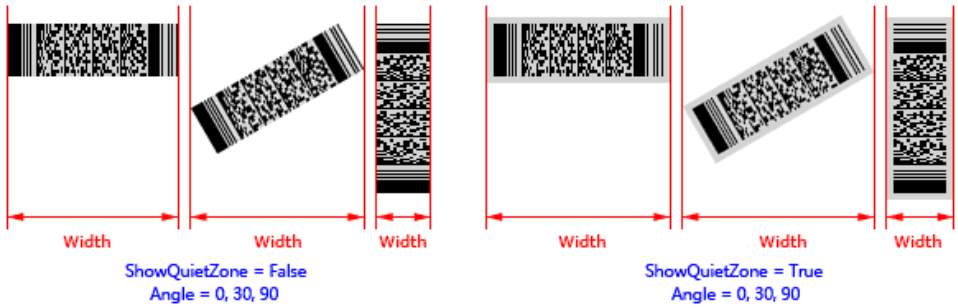
Description:

The method returns the horizontal width and vertical height of a rotated barcode symbol that is specified by properties of this barcode component, in pixels.

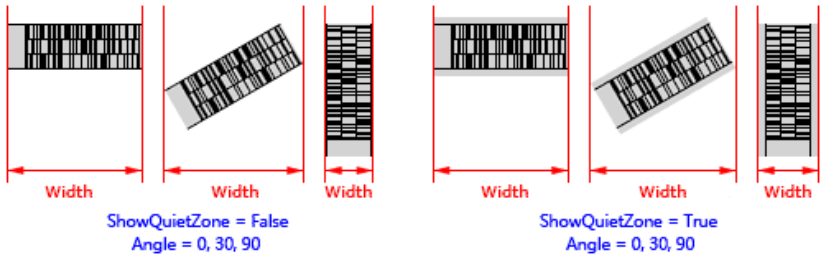
Parameters:

- **Width**: Integer; Returns the horizontal width of the rotated barcode symbol in logical dots or pixels in the horizontal direction.

  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

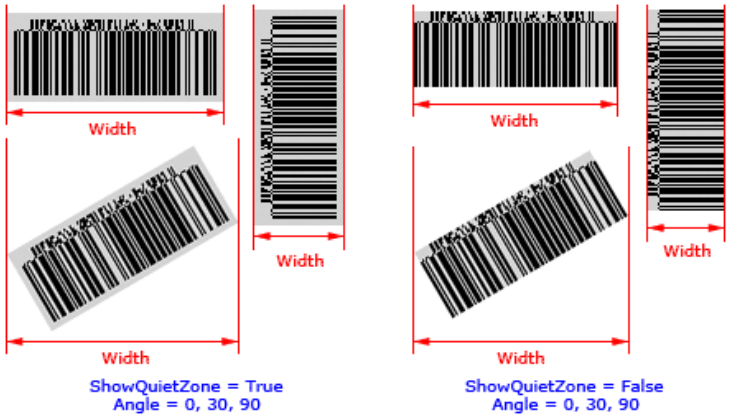


  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

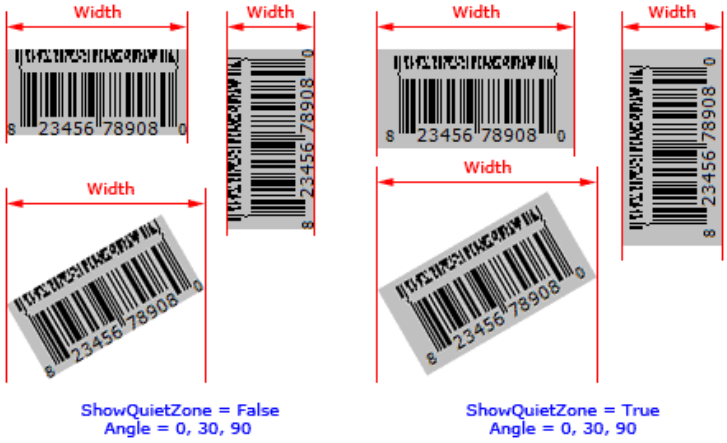


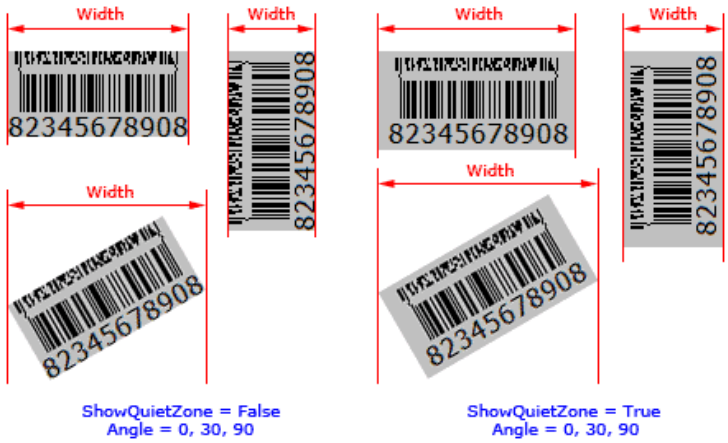
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D

symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



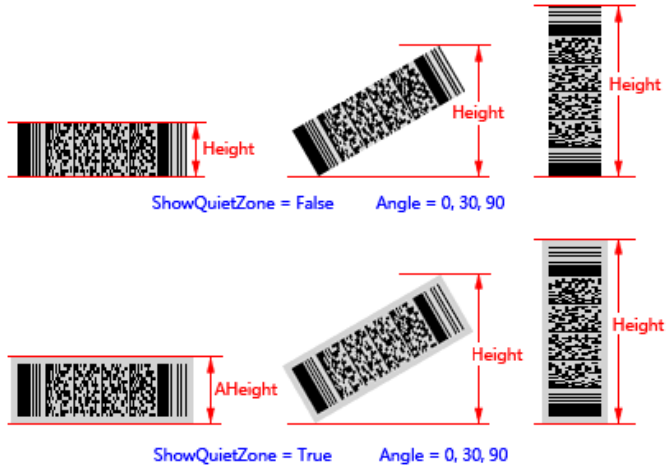
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



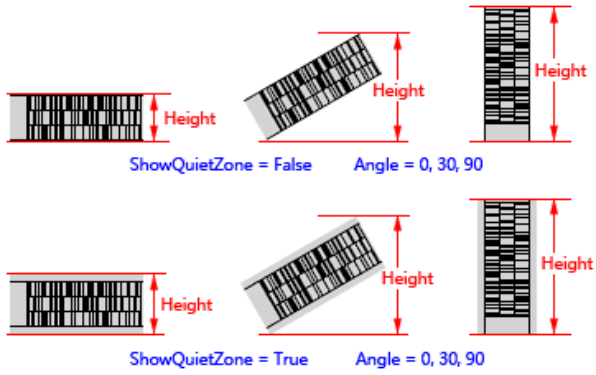


- **Height**: Integer; Returns the vertical height of the rotated barcode symbol in logical dots or pixels in the vertical direction.

  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

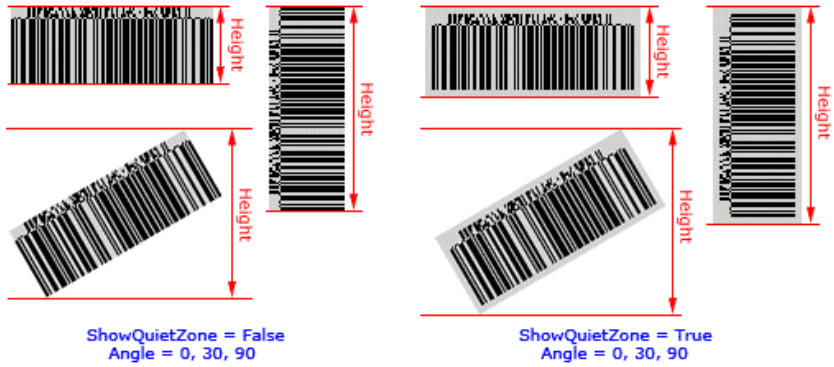


  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

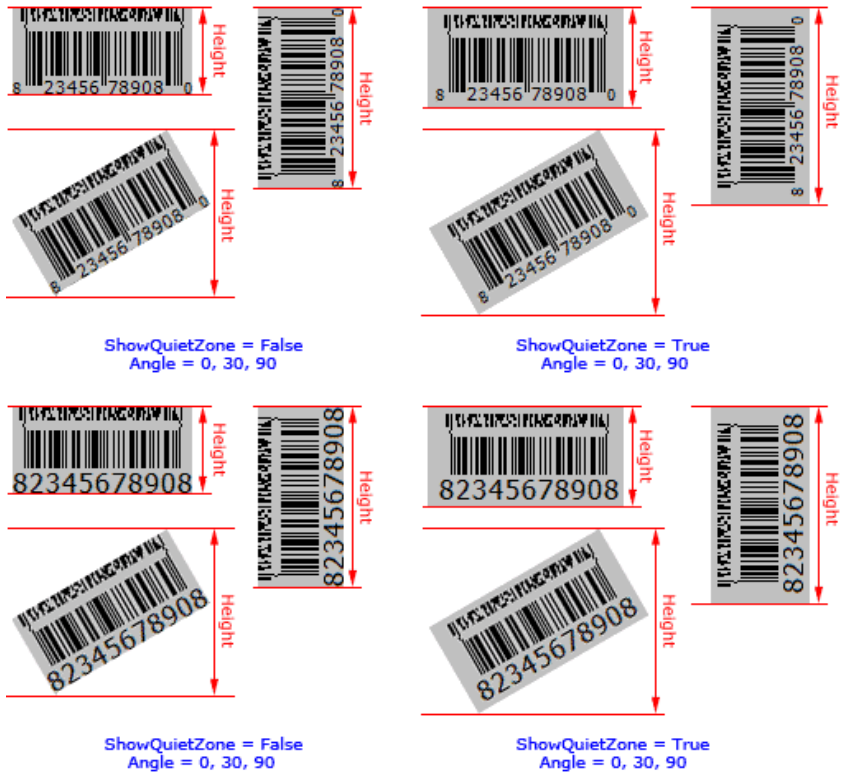


  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is

included in the composite barcode symbol, and the parameter returns the height of entire EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



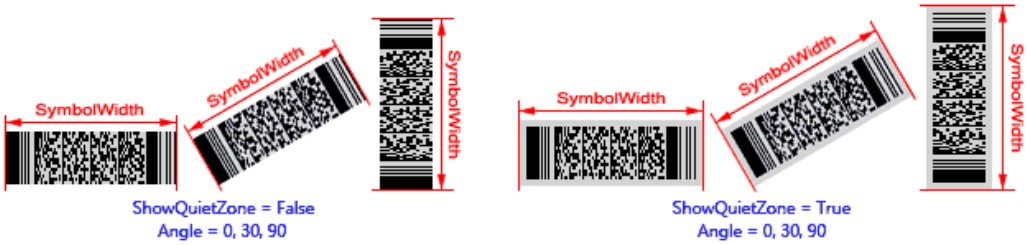
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



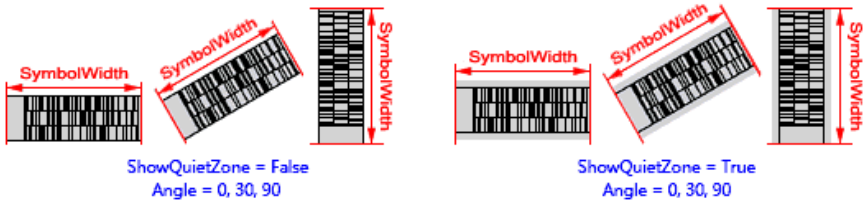
- **SymbolWidth**: Integer; Returns the distance between the leading and trailing of the rotated barcode symbol in logical dots or pixels in the horizontal direction.

  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be

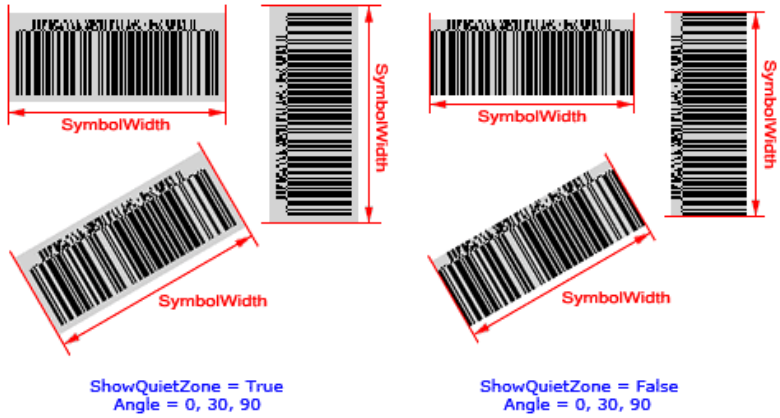
represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire rotated EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

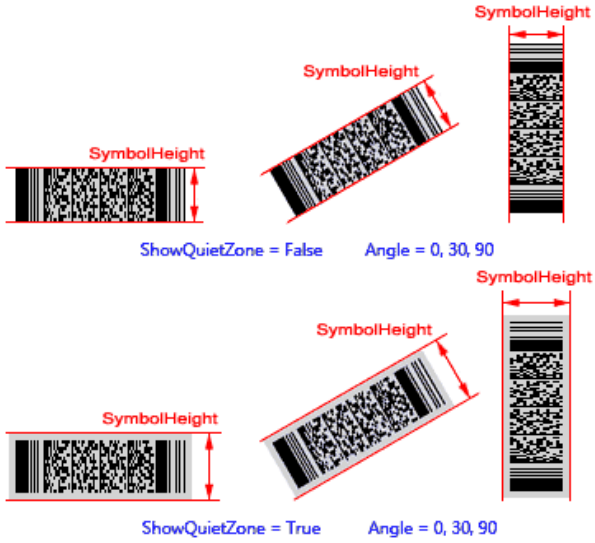


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

See also the "BarcodeWidth" property.

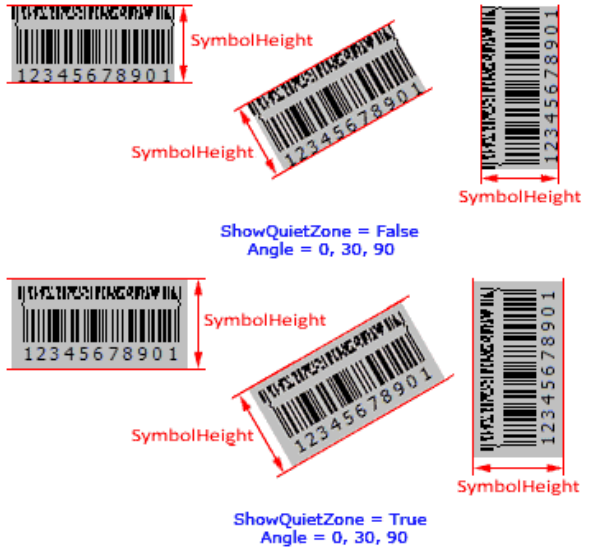
- **SymbolHeight**: Integer; Returns the distance between the top and bottom of the rotated barcode symbol in logical dots or pixels in the vertical direction.

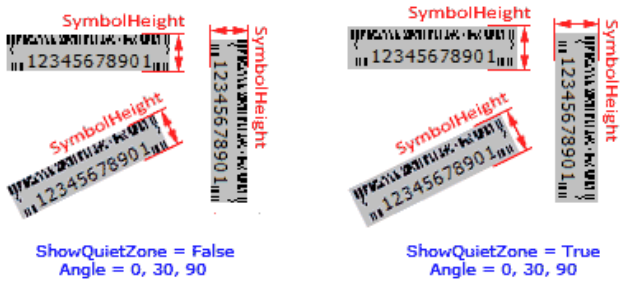
  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire rotated EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

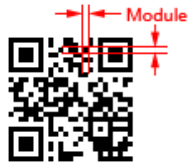
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented, the human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



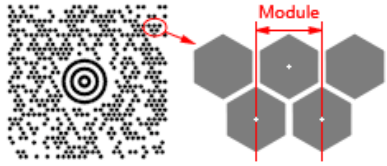


See also the "BarcodeHeight" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction. It defaults to 0 if the Module parameter is not provided, and the value of Module property will be used.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:

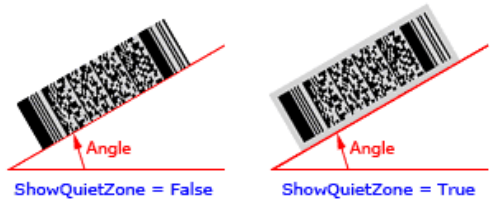


  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

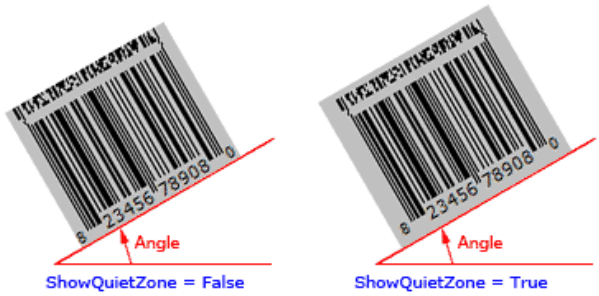
  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone property about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone property to specify whether to represent these quiet zones. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



The parameter defaults to -1 if the Angle is not provided, and the barcode symbol will be rotated based on the value of the Orientation property:

  - boLeftRight: 0 degrees
  - boRightLeft: 180 degrees
  - boTopBottom: 270 degrees
  - boBottomTop: 90 degrees

If you want to use the -1 degrees, the 359 degrees can be used instead.

- **Canvas**: TCanvas; Specifies target canvas to represent the barcode symbol on it.

  The parameter is useful only for the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC. If its Linear property is set to the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and the linear component's DisplayText property isn't set to dtNone, the parameter is required in order to calculate the width of human readable text. Otherwise, it will be ignored, so you can set it to nil.

- **HDPI:** Integer, Specifies the horizontal resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the HDPI is not provided, and the physical horizontal resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROP map mode, and the horizontal units or scaling is changed, the parameter is required in order to obtain the correct sizes when the symbol is rotated.

  When the Canvas parameter is set to nil, if both HDPI and VDPI are set to 0, it indicates the horizontal resolution is equal to the vertical resolution.

- **VDPI:** Integer, Specifies the vertical resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the VDPI is not provided, and the physical vertical resolution obtained from the Canvas parameter will be used.

So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROPIC m mode, and the vertical units or scaling is changed, the parameter is required in order to obtain the correct sizes when the symbol is rotated.

When the Canvas parameter is set to nil, if both HDPI and VDPI are set to 0, it indicates the horizontal resolution is equal to the vertical resolution.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property is invalid, the return valus is -1 or -65537. Corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.
- If there is any invalid character in the the barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character. Corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.8.2 DrawToSize - Syntax 2

Returns the horizontal width and vertical height of a rotated barcode symbol in pixels. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function DrawToSize(var Width, Height, SymbolWidth, SymbolHeight: Integer; Barcode:
  string; ShowQuietZone: Boolean; Module: Integer; Angle: Integer = 0; Canvas: TCanvas =
  nil; HDPI: Integer = 0; VDPI: Integer = 0): Integer; overload; virtual;
```

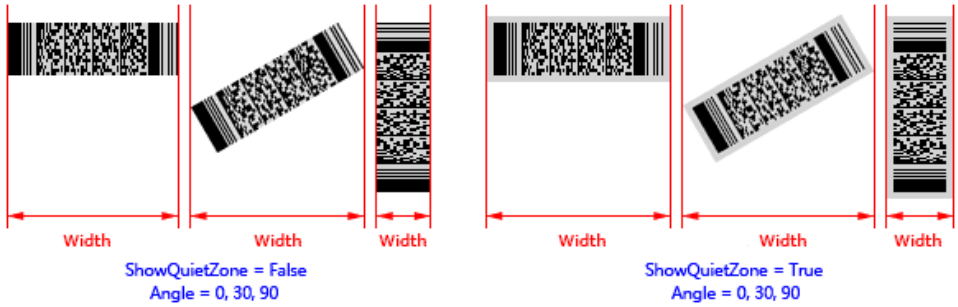
Description:

The method returns the horizontal width and vertical height of a rotated barcode symbol that is specified by parameters of this method, in pixels.

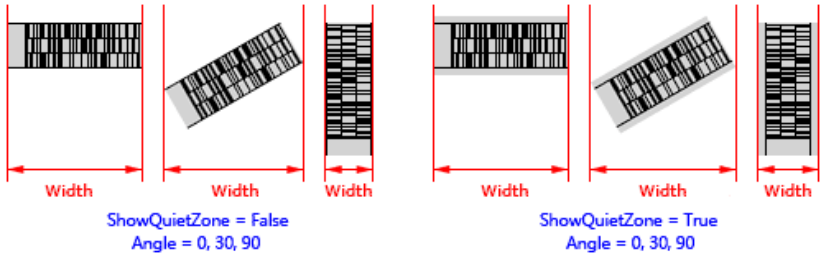
Parameters:

- **Width**: Integer; Returns the horizontal width of the rotated barcode symbol in logical dots or pixels in the horizontal direction.

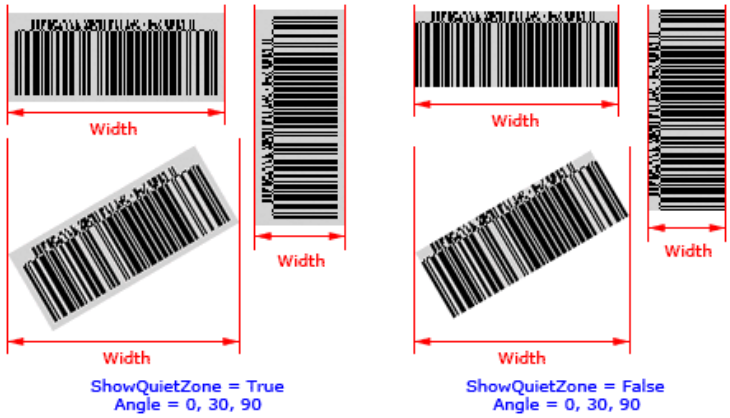
If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



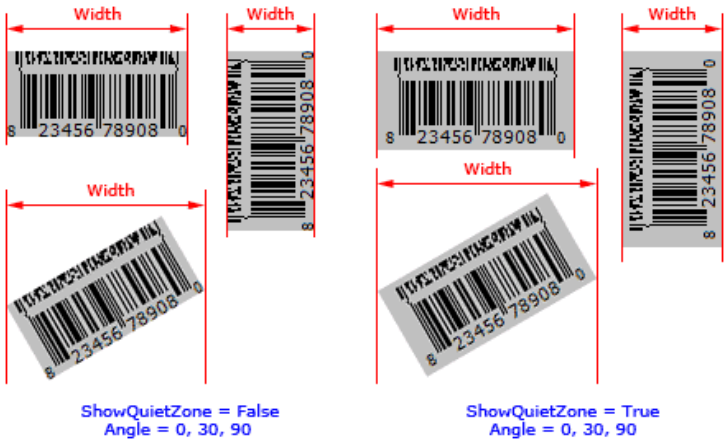
For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

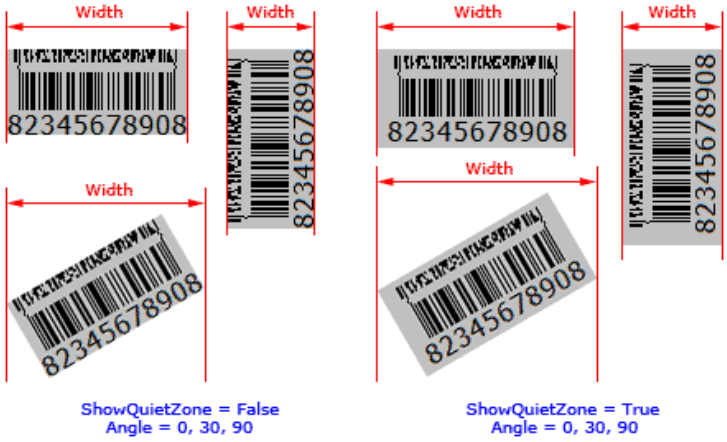


For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



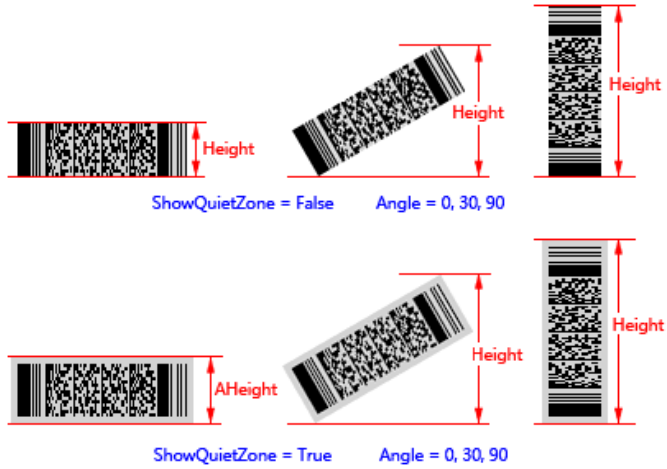
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



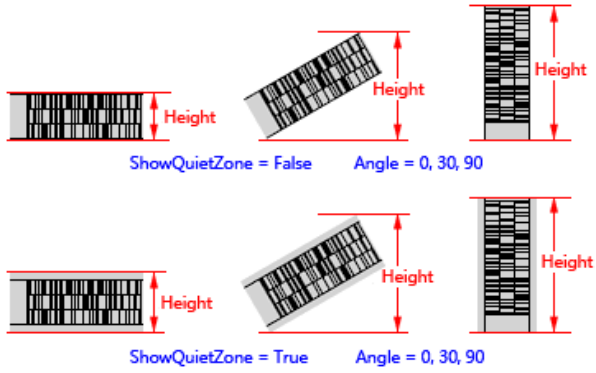


- **Height**: Integer; Returns the vertical height of the rotated barcode symbol in logical dots or pixels in the vertical direction.

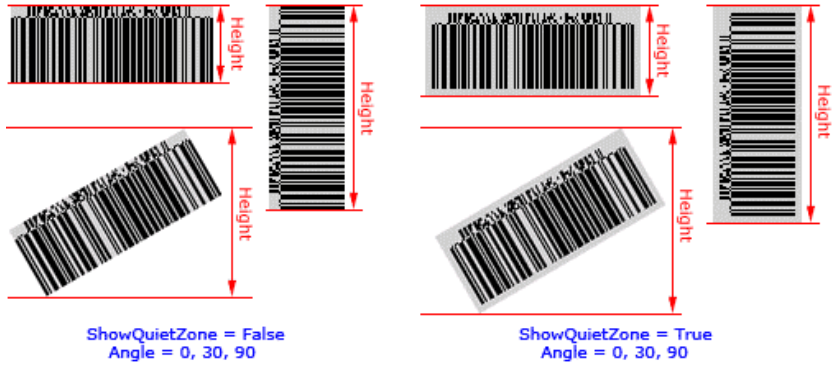
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



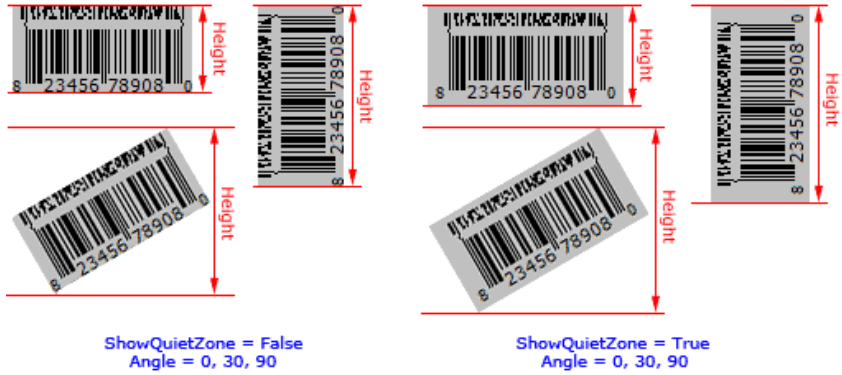
  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

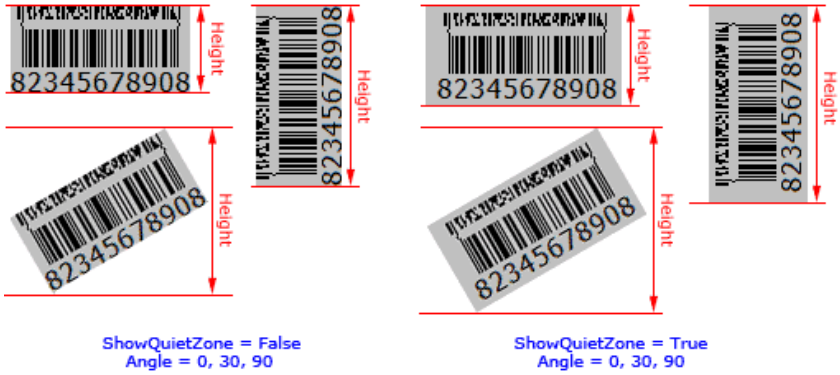


For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



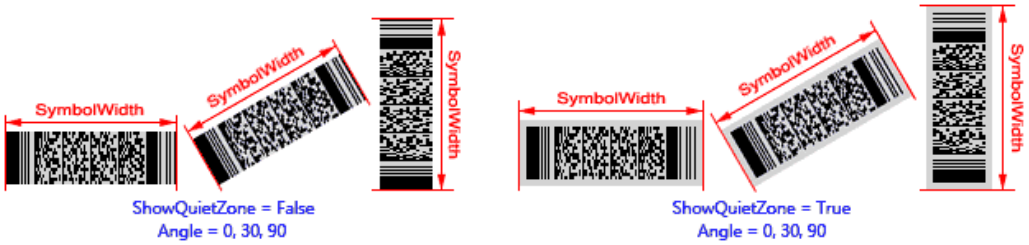
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



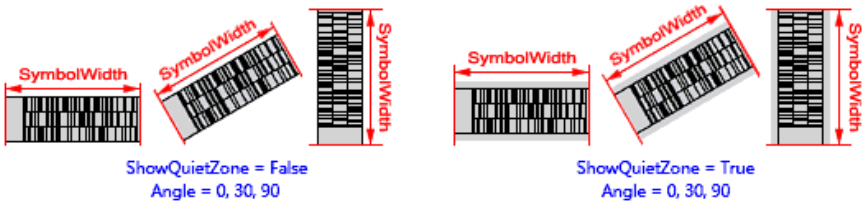


- **SymbolWidth**: Integer; Returns the distance between the leading and trailing of the rotated barcode symbol in logical dots or pixels in the horizontal direction.

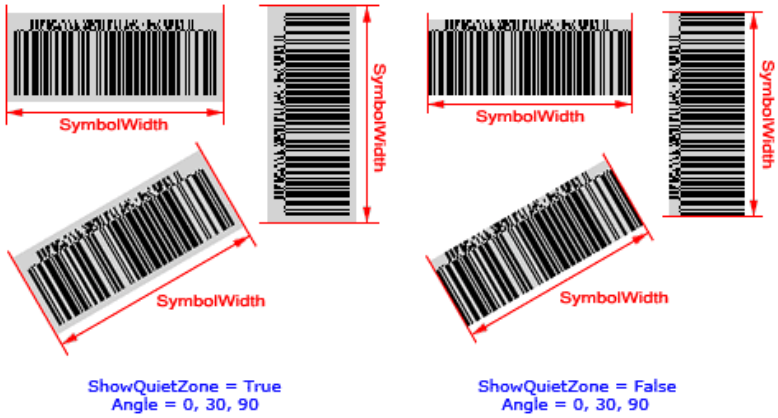
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire rotated EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

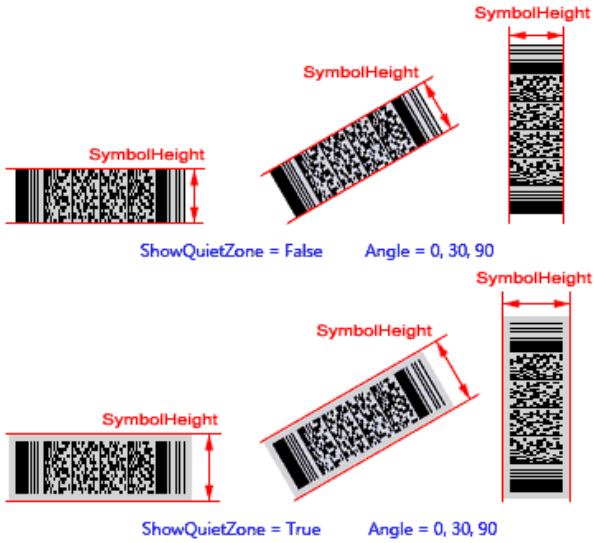


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

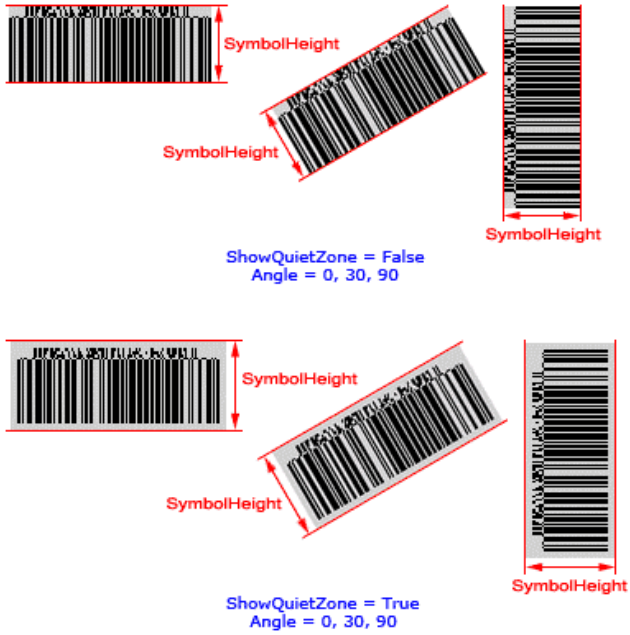
See also the "BarcodeWidth" property.

- **SymbolHeight**: Integer; Returns the distance between the top and bottom of the rotated barcode symbol in logical dots or pixels in the vertical direction.

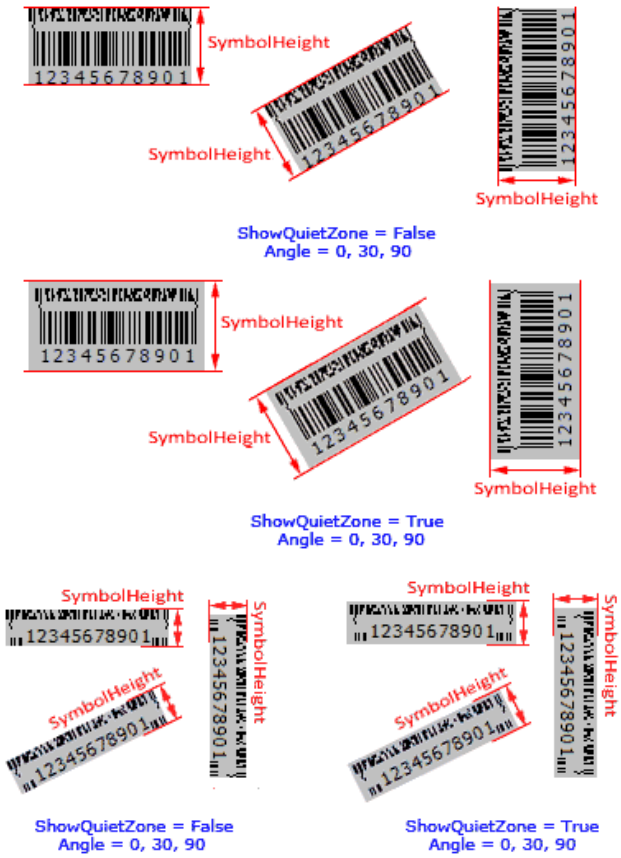
If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire rotated EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented, the human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



See also the "BarcodeHeight" property.

- **Barcode**: String; Specifies the barcode text. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method if you encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the DrawToSize (Syntax 3) overloading method and specify the converted string in its Data parameter. If you encode a block of binary (bytes) data, please use the DrawToSize (Syntax 3) overloading method.

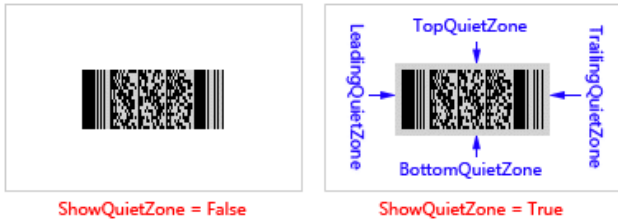
  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

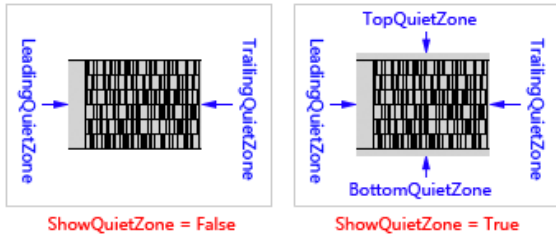
  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **ShowQuietZone**: Boolean; Specifies whether to include the leading, trailing, top and bottom quiet zones in barcode symbol. If the parameter value is set to true, these quiet zones are included in the barcode symbol. Otherwise, they don't be included. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of

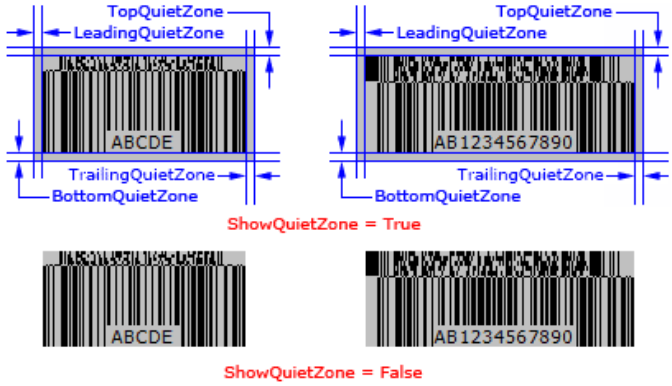
these quiet zones in modules. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



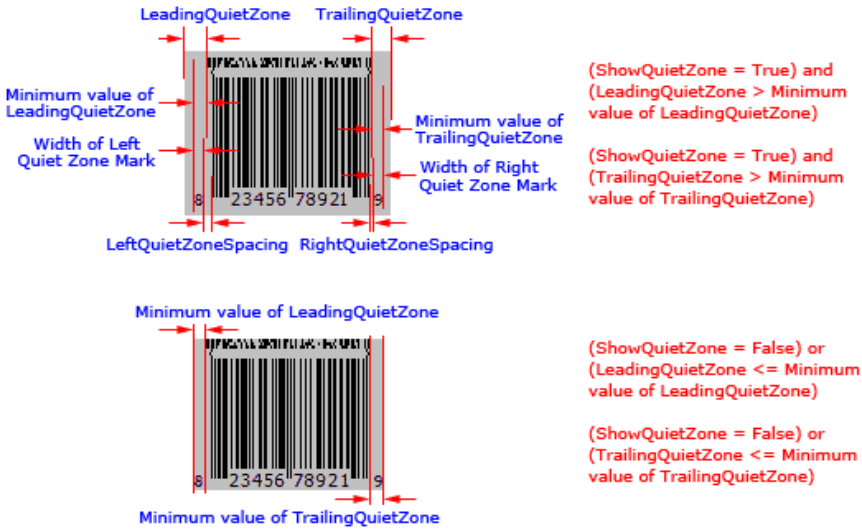
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be included always, even if the ShowQuietZone parameter value is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



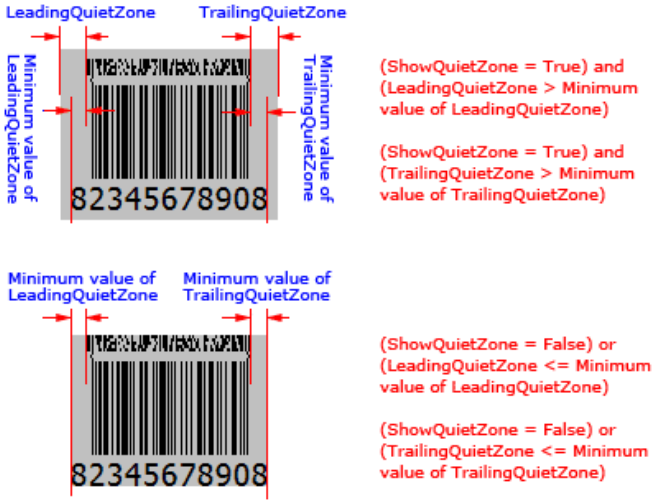
For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

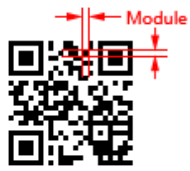


For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

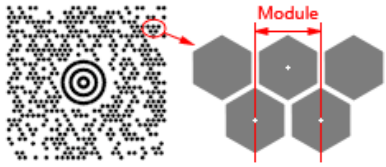


See also the "ShowQuietZone" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction.
  - For the Matrix 2D barcode symbology (excluding theTBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



- For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:

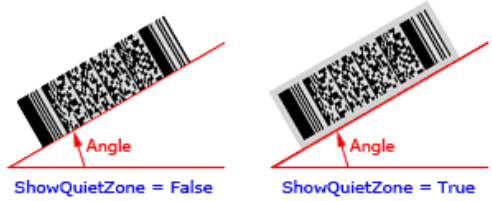


- For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

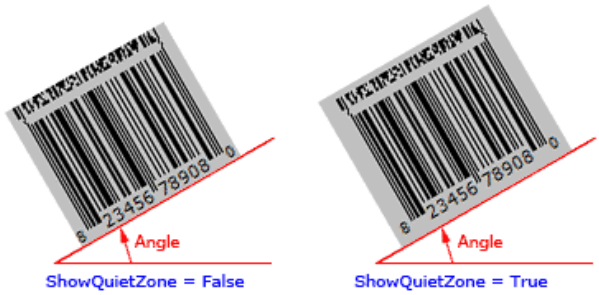
- For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **Canvas**: TCanvas; Specifies target canvas to represent the barcode symbol on it.

  The parameter is useful only for the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC. If its Linear property is set to the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and the linear component's DisplayText property isn't set to dtNone, the parameter is required in order to calculate the width of human readable text. Otherwise, it will be ignored, so you can set it to nil.

- **HDPI:** Integer, Specifies the horizontal resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the HDPI is not provided, and the physical horizontal resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROP map mode, and the horizontal units or scaling is changed, the parameter is required in order to obtain the correct sizes when the symbol is rotated.

  When the Canvas parameter is set to nil, if both HDPI and VDPI are set to 0, it indicates the horizontal resolution is equal to the vertical resolution.

- **VDPI:** Integer, Specifies the vertical resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the VDPI is not provided, and the physical vertical resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROPIC m mode, and the vertical units or scaling is changed, the parameter is required in order to obtain the correct sizes when the symbol is rotated.

  When the Canvas parameter is set to nil, if both HDPI and VDPI are set to 0, it indicates the horizontal resolution is equal to the vertical resolution.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid

character. For example, the return value 65537 denotes that the first character is invalid character).

For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.8.3 DrawToSize - Syntax 3

Returns the horizontal width and vertical height of a rotated barcode symbol in pixels. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function DrawToSize(var Width, Height, SymbolWidth, SymbolHeight: Integer; Data:
  AnsiString; ShowQuietZone: Boolean; Module: Integer; Angle: Integer = 0; Canvas:
  TCanvas = nil; HDPI: Integer = 0; VDPI: Integer = 0): Integer; overload; virtual;
```

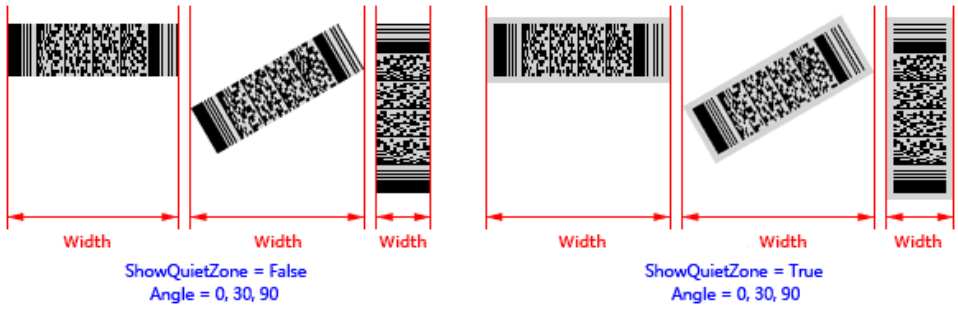
Description:

The method returns the horizontal width and vertical height of a rotated barcode symbol that is specified by parameters of this method, in pixels.

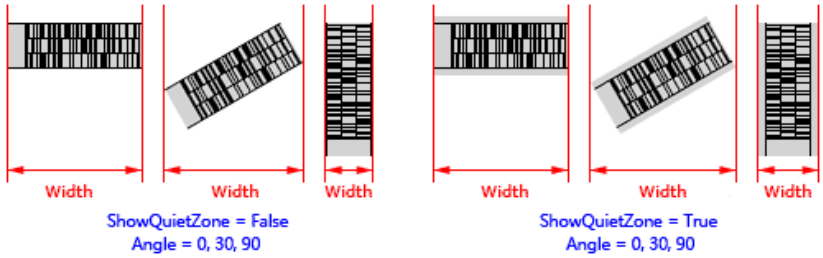
Parameters:

- **Width**: Integer; Returns the horizontal width of the rotated barcode symbol in logical dots or pixels in the horizontal direction.

  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

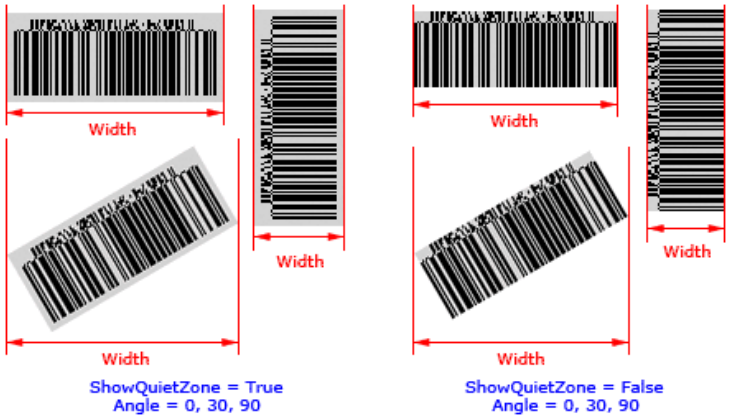


  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

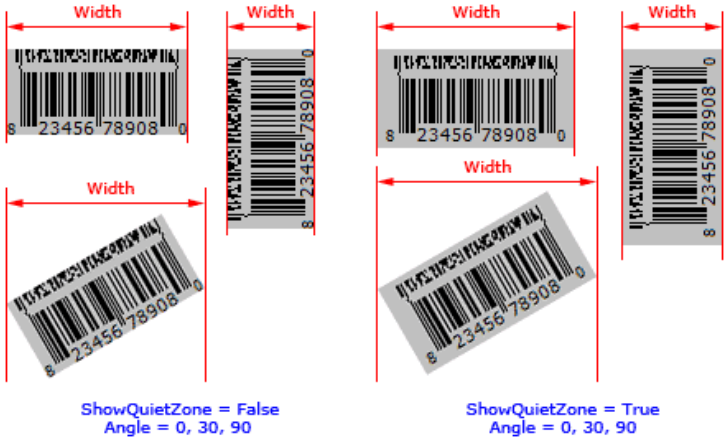


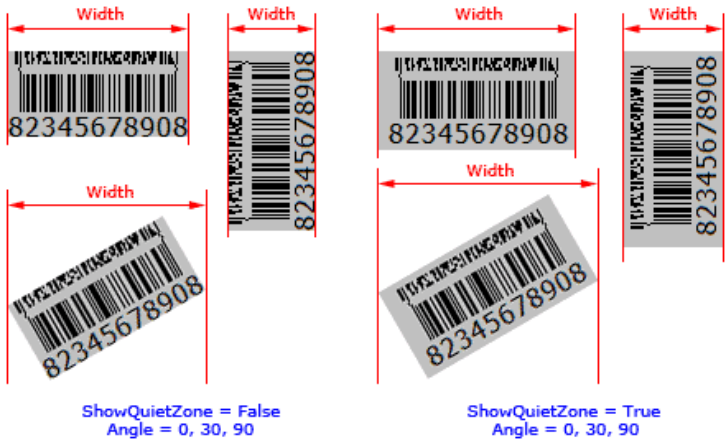
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is

included in the composite barcode symbol, and the parameter returns the width of entire EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



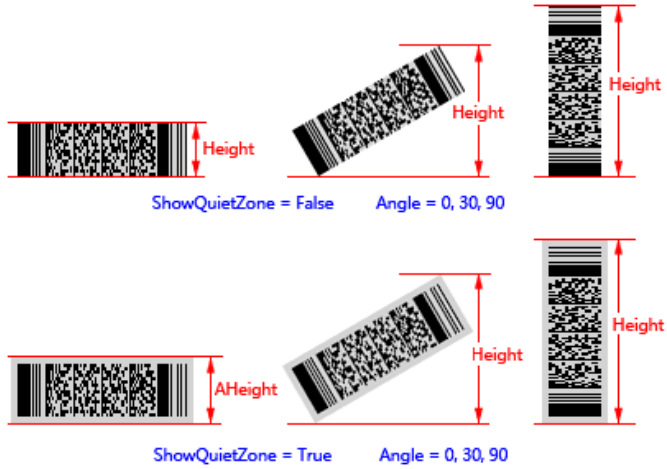
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



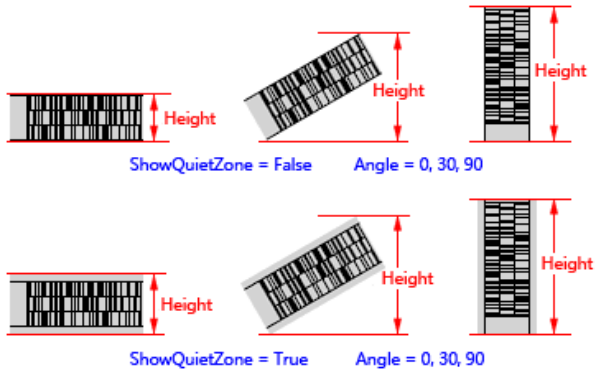


- **Height**: Integer; Returns the vertical height of the rotated barcode symbol in logical dots or pixels in the vertical direction.

  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

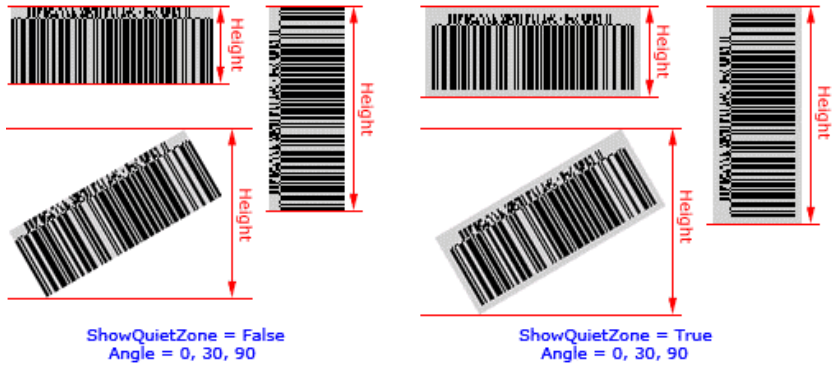


  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

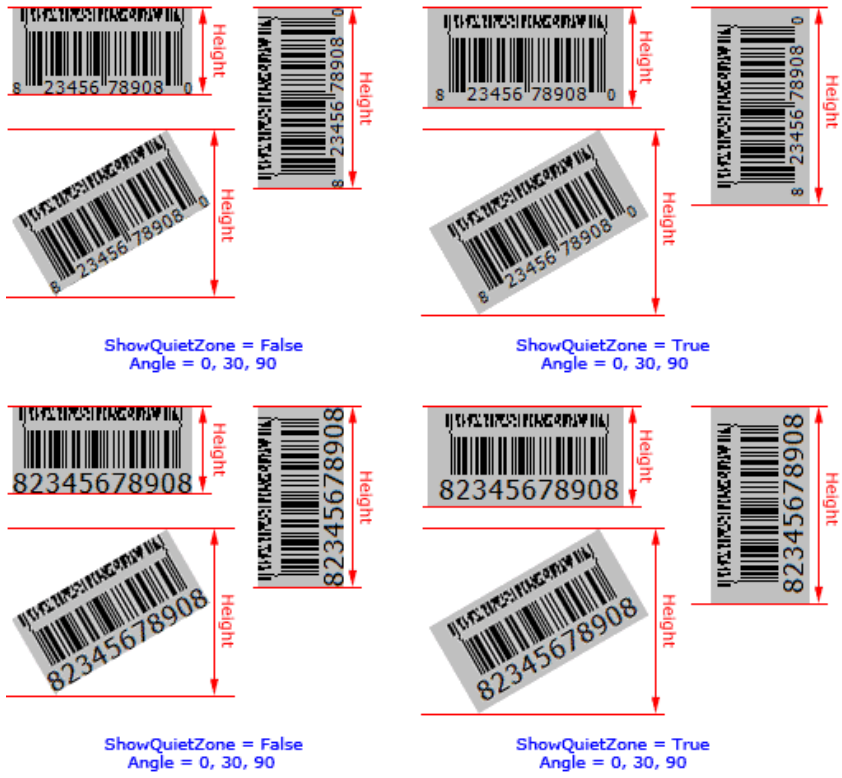


  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is

included in the composite barcode symbol, and the parameter returns the height of entire EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



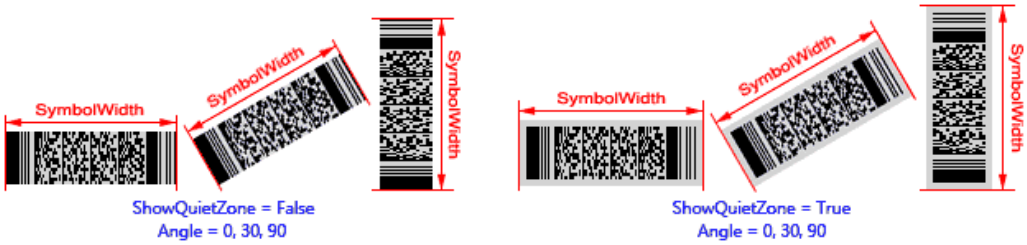
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



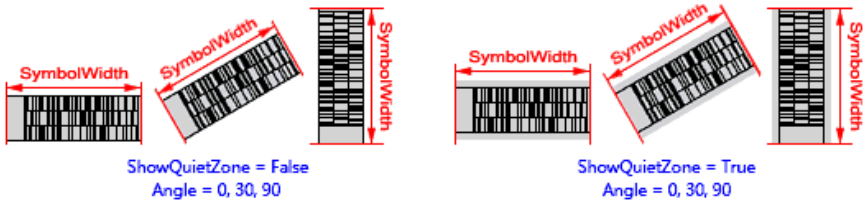
- **SymbolWidth**: Integer; Returns the distance between the leading and trailing of the rotated barcode symbol in logical dots or pixels in the horizontal direction.

  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet

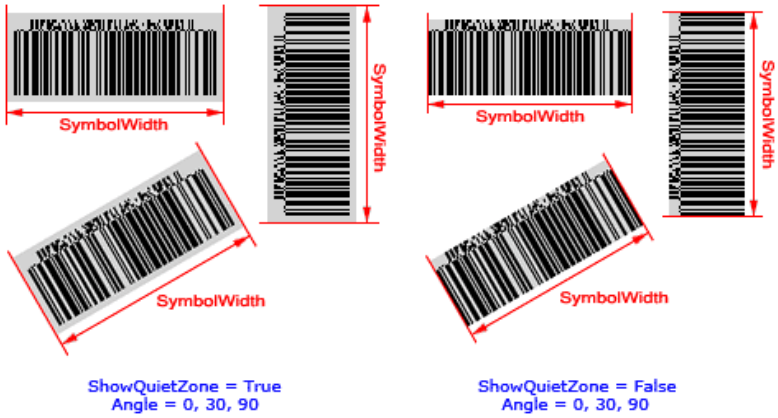
zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire rotated EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

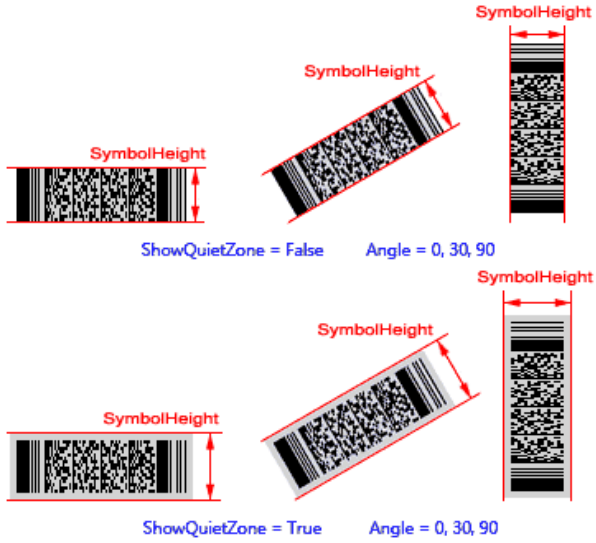


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

See also the "BarcodeWidth" property.

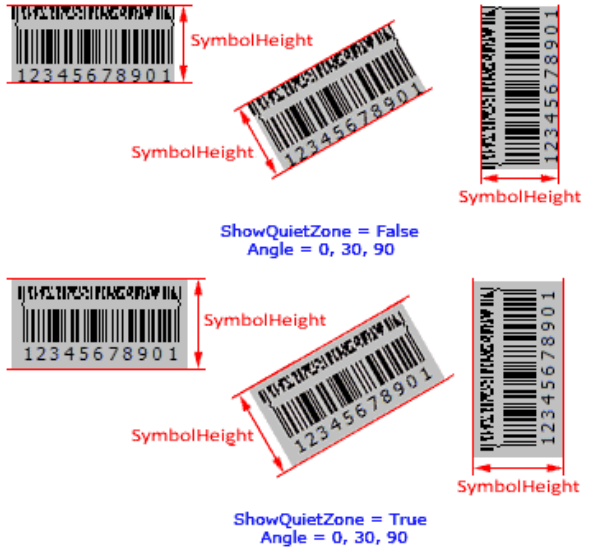
- **SymbolHeight**: Integer; Returns the distance between the top and bottom of the rotated barcode symbol in logical dots or pixels in the vertical direction.

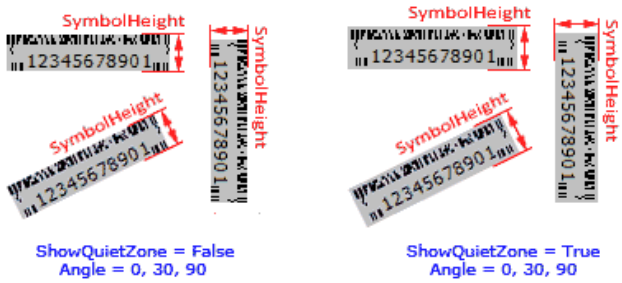
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire rotated EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented, the human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):





See also the "BarcodeHeight" property.

- **Data**: AnsiString; Specifies the barcode text. It is of type AnsiString.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

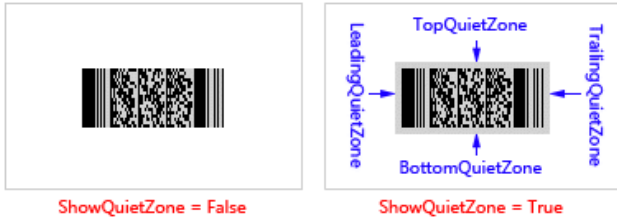
  For the Delphi/C++ Builder 2009 or later, you can use the method if you encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the DrawToSize (Syntax 2) overloading method.

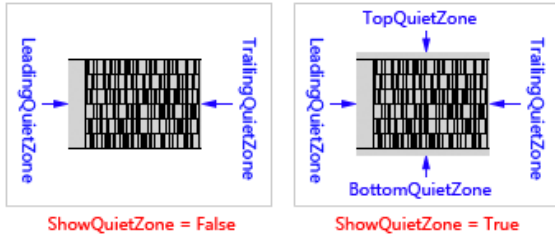
  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **ShowQuietZone**: Boolean; Specifies whether to include the leading, trailing, top and bottom quiet zones in barcode symbol. If the parameter value is set to true, these quiet zones are included in the barcode symbol. Otherwise, they don't be included. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

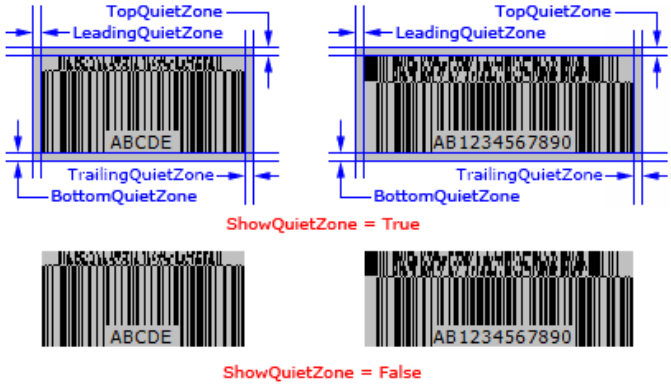


  For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be included always, even if the ShowQuietZone parameter value is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

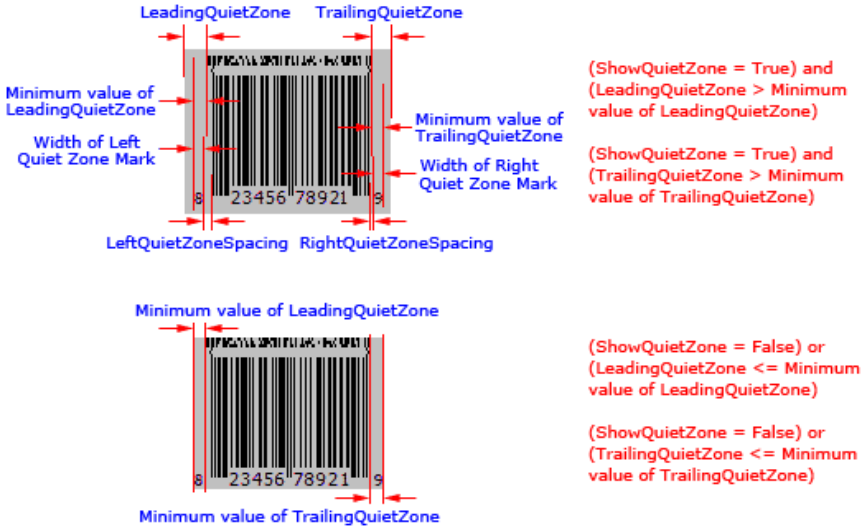


  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for

entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

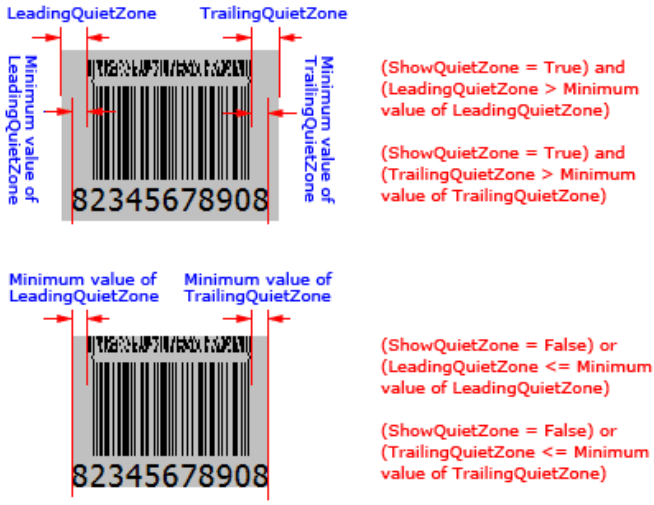


For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



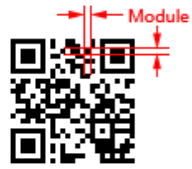
For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if the ShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order

to accentuate them):

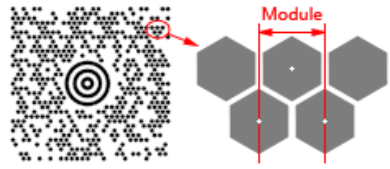


See also the "ShowQuietZone" property.

- **Module**: Integer; Specifies the module size in logical dots or pixels in the horizontal direction.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:

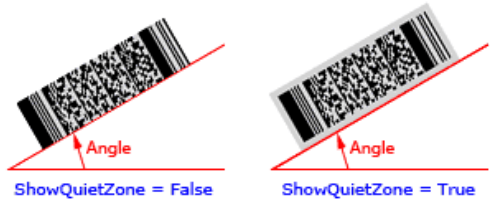


  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

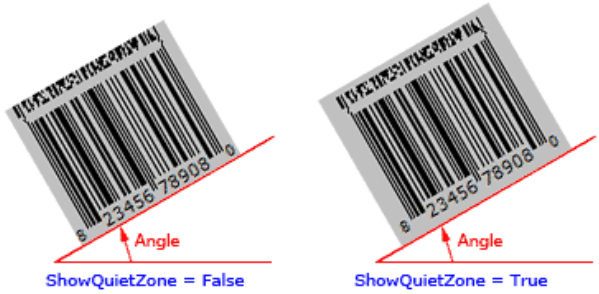
  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **Canvas**: TCanvas; Specifies target canvas to represent the barcode symbol on it.

  The parameter is useful only for the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC. If its Linear property is set to the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and the linear component's DisplayText property isn't set to dtNone, the parameter is required in order to calculate the width of human readable text. Otherwise, it will be ignored, so you can set it to nil.

- **HDPI:** Integer, Specifies the horizontal resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the HDPI is not provided, and the physical horizontal resolution obtained from the Canvas parameter will be used. So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROP map mode, and the horizontal units or scaling is changed, the parameter is required in order to obtain the correct sizes when the symbol is rotated.

  When the Canvas parameter is set to nil, if both HDPI and VDPI are set to 0, it indicates the horizontal resolution is equal to the vertical resolution.

- **VDPI:** Integer, Specifies the vertical resolution of canvas in logical DPI. It's the number of logical dots or pixels per inch. It defaults to 0 if the VDPI is not provided, and the physical vertical resolution obtained from the Canvas parameter will be used.

So if you use the MM_TEXT map mode, you can specify it to 0. If you use the MM_ISOTROPIC or MM_ANISOTROPIC m mode, and the vertical units or scaling is changed, the parameter is required in order to obtain the correct sizes when the symbol is rotated.

When the Canvas parameter is set to nil, if both HDPI and VDPI are set to 0, it indicates the horizontal resolution is equal to the vertical resolution.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.
- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

B.1.9 GetCheckSum

(TBarcode2D_MicroPDF417)

Returns the check sum of a barcode text that will be encoded into a series of structured append symbols. The value will be used in the structured append block, and the structured append block will be used by each symbol in the series of MicroPDF417 structured append symbols. There are two different overloading methods, Syntax 1, and Syntax 2 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Returns the check sum value of a barcode text. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to caculate the check sum of a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then caculate its check sum. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the GetCheckSum (Syntax 2) overloading method and

specify the converted string in its Data parameter. If you want to caculate the check sum of a block of binary (bytes) data, please use the GetCheckSum (Syntax 2) overloading method.

- **Syntax 2**: Returns the check sum value of a barcode text. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to caculate the check sum of a block of binary (bytes) data. Or specifies the barcode text in AnsiString format.

  If you want to caculate the check sum of a block of binary (bytes) data under Delphi/C++ Builder 2007 or early, please use the GetCheckSum (Syntax 1) overloading method.

See also the "Sturctured append" section in the "TBarcode2D_MicroPDF417" article.

B.1.9.1 GetCheckSum - Syntax 1

(TBarcode2D_MicroPDF417)

Returns the check sum of a barcode text that will be encoded into a series of structured append symbols.

Syntax:

```
type
  { Defined in the pPDF417Com unit }
  TPDF417_Option = (poIgnoreShiftBeforeECI, poFirst903TextAlphaLatch,
    poFirst904TextMixedLatch, po906TextAlphaLatch, po907TextAlphaLatch,
    po908TextAlphaLatch, po910TextAlphaLatch, po912TextAlphaLatch, po914TextAlphaLatch,
    po915TextAlphaLatch, poFirstFNC1MatchAI01, poMicroPDF417Explicit901);
  { Defined in the pPDF417Com unit }
  TPDF417_Options = set of TPDF417_Option;
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_Options = TPDF417_Options;
function GetCheckSum(Barcode: string; AllowEscape: Boolean; var InvalidIndex: Integer;
  Options: TMicroPDF417_Options = []): string; virtual;
```

Description:

The method returns the check sum of a barcode text that will be encoded into a series of structured append symbols. The value will be used in the structured append block, and the structured append block will be used by each symbol in the series of MicroPDF417 structured append symbols. See also the "Structured append" section in the "TBarcode2D_MicroPDF417" article.

Parameters:

- **Barcode**: string; It's the original input text before division into the each symbol in the series of structured append symbols.It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to caculate the check sum of a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then caculate its check sum. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the GetCheckSum (Syntax 2) overloading method and specify the converted string in its Data parameter. If you want to caculate the check sum of a block of binary (bytes) data, please use the GetCheckSum (Syntax 2) overloading method.

- **AllowEscape**: Boolean; Specifies whether to allow users to insert the escape sequences to the Barcode parameter value, in order to place the function characters and additional control information. See also the "Escape sequences" section in the "TBarcode2D_MicroPDF417" article.
- **InvalidIndex**: Integer; If there is any invalid character in the barcode text that is specified by the Barcode parameter, the parameter returns the position index of first invalid character, the index 1 denotes that the first character is invalid character. Otherwise, it returns the zero.
- **Options**: TMicroPDF417_Options; The Options parameter is an advanced feature which allows low level control over data encoding. The parameter can be used to change the encoding algorithm and meanings of some fucntion codewords, in order to match the reader. The values that can be included in the Options parameter are defined in the pPDF417Com unit. See also the "Options" property article.

Return:

If the method succeeds, it returns the check sum value, it's a 16-bit word value, in decimal string format. And the InvalidIndex parameter returns zero. If the method fails, the InvalidIndex parameter returns a position index of first invalid character, it's an integer value greater than 0.

The result string can be directly used as the check sum in the structured append block. For example:

```
var
  full_barcode, check_sum, structuredappend_block: string;
  invalid_index: integer;
begin
  full_barcode := '......';
  check_sum := GetCheckSum(full_barcode, false, invalid_index, []);
  if invalid_index > 0 then exit;
  structuredappend_block := '\s[1,001287023,,5,2008-12-03 05:30:00,,USA,,' + check_sum +
    ']';
  //.....
```

See also the "Sturctured append" section in the "TBarcode2D_MicroPDF417" article.

B.1.9.2 GetCheckSum - Syntax 2

(TBarcode2D_MicroPDF417)

Returns the check sum of a barcode text that will be encoded into a series of structured append symbols.

Syntax:

```
type
  { Defined in the pPDF417Com unit }
  TPDF417_Option = (poIgnoreShiftBeforeECI, poFirst903TextAlphaLatch,
    poFirst904TextMixedLatch, po906TextAlphaLatch, po907TextAlphaLatch,
    po908TextAlphaLatch, po910TextAlphaLatch, po912TextAlphaLatch, po914TextAlphaLatch,
    po915TextAlphaLatch, poFirstFNC1MatchAI01, poMicroPDF417Explicit901);
  { Defined in the pPDF417Com unit }
  TPDF417_Options = set of TPDF417_Option;
  { Defined in the pMicroPDF417 unit }
  TMicroPDF417_Options = TPDF417_Options;
function GetCheckSum(Data: AnsiString; AllowEscape: Boolean; var InvalidIndex: Integer;
  Options: TMicroPDF417_Options = []): AnsiString; virtual;
```

Description:

The method returns the check sum of a barcode text that will be encoded into a series of structured append symbols. The value will be used in the structured append block, and the structured append block will be used by each symbol in the series of MicroPDF417 structured append symbols. See also the "Structured append" section in the "TBarcode2D_MicroPDF417" article.

Parameters:

- **Data**: AnsiString; It's the original input text before division into the each symbol in the series of structured append symbols. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to caculate the check sum of a block of binary (bytes) data. Or specifies the barcode text in AnsiString format.

  If you want to caculate the check sum of a block of binary (bytes) data under Delphi/C++ Builder 2007 or early, please use the GetCheckSum (Syntax 1) overloading method.

- **AllowEscape**: Boolean; Specifies whether to allow users to insert the escape sequences to the Data parameter value, in order to place the function characters and additional control information. See also the "Escape sequences" section in the "TBarcode2D_MicroPDF417" article.

- **InvalidIndex**: Integer; If there is any invalid character in the barcode text that is specified by the Data parameter, the parameter returns the position index of first invalid character, the index 1 denotes that the first character is invalid character. Otherwise, it returns the zero.

- **Options**: TMicroPDF417_Options; The Options parameter is an advanced feature which allows low level control over data encoding. The parameter can be used to change the encoding algorithm and meanings of some fucntion codewords, in order to match the reader. The values that can be included in the Options parameter are defined in the pPDF417Com unit. See also the "Options" property article.

Return:

If the method succeeds, it returns the check sum value, it's a 16-bit word value, in decimal AnsiString format. And the InvalidIndex parameter returns zero. If the method fails, the InvalidIndex parameter returns a position index of first invalid character, it's an integer value greater than 0.

The result string can be directly used as the check sum in the structured append block. For example:

```
var
  full_barcode, check_sum, structuredappend_block: string;
  invalid_index: integer;
begin
  full_barcode := '......';
  check_sum := GetCheckSum(full_barcode, false, invalid_index, []);
  if invalid_index > 0 then exit;
  structuredappend_block := '\s[1,001287023,,5,2008-12-03 05:30:00,,USA,,' + check_sum +
    ']';
  //.....
```

See also the "Sturctured append" section in the "TBarcode2D_MicroPDF417" article.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

B.1.10 GetCheckSum

(TBarcode2D_PDF417)

Returns the check sum of a barcode text that will be encoded into each symbol in a Macro PDF417 set. The value will be used in the macro PDF417 control information block, and the macro PDF417 control information block will be used by each symbol in the Macro PDF417 set. There are two different overloading methods, Syntax 1, and Syntax 2 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Returns the check sum value of a barcode text. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to caculate the check sum of a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact a UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then caculate its check sum. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the GetCheckSum (Syntax 2) overloading method and specify the converted string in its Data parameter. If you want to caculate the check sum of a block of binary (bytes) data, please use the GetCheckSum (Syntax 2) overloading method.

- **Syntax 2**: Returns the check sum value of a barcode text. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to caculate the check sum of a block of binary (bytes) data. Or specifies the barcode text in AnsiString format.

  If you want to caculate the check sum of a block of binary (bytes) data under Delphi/C++ Builder 2007 or early, please use the GetCheckSum (Syntax 1) overloading method.

See also the "Macro PDF417" section in the "TBarcode2D_PDF417" article.

B.1.10.1 GetCheckSum - Syntax 1

(TBarcode2D_PDF417)

Returns the check sum of a barcode text that will be encoded into each symbol in a Macro PDF417 set.

Syntax:

```
type
  { Defined in the pPDF417Com unit }
  TPDF417_Option = (poIgnoreShiftBeforeECI, poFirst903TextAlphaLatch,
    poFirst904TextMixedLatch, po906TextAlphaLatch, po907TextAlphaLatch,
    po908TextAlphaLatch, po910TextAlphaLatch, po912TextAlphaLatch, po914TextAlphaLatch,
    po915TextAlphaLatch, poFirstFNC1MatchAI01, poMicroPDF417Explicit901);
  { Defined in the pPDF417Com unit }
  TPDF417_Options = set of TPDF417_Option;

function GetCheckSum(Barcode: string; AllowEscape: Boolean; var InvalidIndex: Integer;
  Options: TPDF417_Options = []): string; virtual;
```

Description:

The method returns the check sum of a barcode text that will be encoded into a Macro PDF417 set. The value will be used in the macro PDF417 control information block, and the macro PDF417 control information block will be used by each symbol in the Macro PDF417 set. See also the "Macro PDF417" section in the "TBarcode2D_PDF417" article.

Parameters:

- **Barcode**: string; It's the original input text before division into the each symbol in the Macro PDF417 set. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to caculate the check sum of a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then caculate its check sum. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the GetCheckSum (Syntax 2) overloading method and specify the converted string in its Data parameter. If you want to caculate the check sum of a block of binary (bytes) data, please use the GetCheckSum (Syntax 2) overloading method.

- **AllowEscape**: Boolean; Specifies whether to allow users to insert the escape sequences to the Barcode parameter value, in order to place the function characters and additional control information. See also the "Escape sequences" section in the "TBarcode2D_PDF417" article.

- **InvalidIndex**: Integer; If there is any invalid character in the barcode text that is specified by the Barcode parameter, the parameter returns the position index of first invalid character, the index 1 denotes that the first character is invalid character. Otherwise, it returns the zero.

- **Options**: TPDF417_Options; The Options parameter is an advanced feature which allows low level control over data encoding. The parameter can be used to change the encoding algorithm and meanings of some fucntion codewords, in order to match the reader. The values that can be included in the Options parameter are defined in the pPDF417Com unit. See also the "Options" property article.

Return:

If the method succeeds, it returns the check sum value, it's a 16-bit word value, in decimal string format. And the InvalidIndex parameter returns zero. If the method fails, the InvalidIndex parameter returns a position index of first invalid character, it's an integer value greater than 0.

The result string can be directly used as the check sum in the Macro PDF417 control information block. For example:

```
var
  full_barcode, check_sum, macro_block: string;
  invalid_index: integer;
begin
  full_barcode := '......';
  check_sum := GetCheckSum(full_barcode, false, invalid_index, []);
  if invalid_index > 0 then exit;
  macro_block := '\s[1,001287023,,5,2008-12-03 05:30:00,,USA,,' + check_sum + ']';
  //.....
```

See also the "Macro PDF417" section in the "TBarcode2D_PDF417" article.

B.1.10.2 GetCheckSum - Syntax 2

(TBarcode2D_PDF417)

Returns the check sum of a barcode text that will be encoded into each symbol in a Macro PDF417 set.

Syntax:

```
type
  { Defined in the pPDF417Com unit }
  TPDF417_Option = (poIgnoreShiftBeforeECI, poFirst903TextAlphaLatch,
    poFirst904TextMixedLatch, po906TextAlphaLatch, po907TextAlphaLatch,
    po908TextAlphaLatch, po910TextAlphaLatch, po912TextAlphaLatch, po914TextAlphaLatch,
    po915TextAlphaLatch, poFirstFNC1MatchAI01, poMicroPDF417Explicit901);
  { Defined in the pPDF417Com unit }
  TPDF417_Options = set of TPDF417_Option;

function GetCheckSum(Data: AnsiString; AllowEscape: Boolean; var InvalidIndex: Integer;
  Options: TPDF417_Options = []): AnsiString; virtual;
```

Description:

The method returns the check sum of a barcode text that will be encoded into a Macro PDF417 set. The value will be used in the macro PDF417 control information block, and the macro PDF417 control information block will be used by each symbol in the Macro PDF417 set. See also the "Macro PDF417" section in the "TBarcode2D_PDF417" article.

Parameters:

- **Data**: AnsiString; It's the original input text before division into the each symbol in the Macro PDF417 set. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to caculate the check sum of a block of binary (bytes) data. Or specifies the barcode text in AnsiString format.

  If you want to caculate the check sum of a block of binary (bytes) data under Delphi/C++ Builder 2007 or early, please use the GetCheckSum (Syntax 1) overloading method.

- **AllowEscape**: Boolean; Specifies whether to allow users to insert the escape sequences to the Data parameter value, in order to place the function characters and additional control information. See also the "Escape sequences" section in the "TBarcode2D_PDF417" article.

- **InvalidIndex**: Integer; If there is any invalid character in the barcode text that is specified by the Data parameter, the parameter returns the position index of first invalid character, the index 1 denotes that the first character is invalid character. Otherwise, it returns the zero.

- **Options**: TPDF417_Options; The Options parameter is an advanced feature which allows low level control over data encoding. The parameter can be used to change the encoding algorithm and meanings of some fucntion codewords, in order to match the reader. The values that can be included in the Options parameter are defined in the pPDF417Com unit. See also the "Options" property article.

Return:

If the method succeeds, it returns the check sum value, it's a 16-bit word value, in decimal AnsiString format. And the InvalidIndex parameter returns zero. If the method fails, the InvalidIndex parameter returns a position index of first invalid character, it's an integer value greater than 0.

The result string can be directly used as the check sum in the Macro PDF417 control information block. For example:

```
var
  full_barcode, check_sum, macro_block: string;
  invalid_index: integer;

begin
  full_barcode := '......';
  check_sum := GetCheckSum(full_barcode, false, invalid_index, []);
  if invalid_index > 0 then exit;
  macro_block := '\s[1,001287023,,5,2008-12-03 05:30:00,,USA,,' + check_sum + ']';
  //.....
```

See also the "Macro PDF417" section in the "TBarcode2D_PDF417" article.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

B.1.11 GetParity

(TBarcode2D_QRCode)

Returns the parity value of a barcode text that will be encoded into a series of structured append symbols. The value will be used in the structured append block, and the structured append block will be used by each symbol in the series of QRCode structured append symbols. There are two different overloading methods, Syntax 1, and Syntax 2 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Returns the parity value of a barcode text. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to caculate the parity of a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then caculate its parity. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the GetParity (Syntax 2) overloading method and specify the converted string in its Data parameter. If you want to caculate the parity of a block of binary (bytes) data, please use the GetParity (Syntax 2) overloading method.

- **Syntax 2**: Returns the parity value of a barcode text. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to caculate the parity of a block of binary (bytes) data. Or specifies the barcode text in AnsiString format.

  If you want to caculate the parity of a block of binary (bytes) data under Delphi/C++ Builder 2007 or early, please use the GetParity (Syntax 1) overloading method.

See also the "Structured append" section in the "TBarcode2D_QRCode" article.

B.1.11.1 GetParity - Syntax 1

(TBarcode2D_QRCode)

Returns the parity value of a barcode text that will be encoded into a series of structured append symbols.

Syntax:

```
function GetParity(Barcode: string; AllowEscape: Boolean; var InvalidIndex: Integer):
  Byte; virtual;
```

Description:

The method returns the parity value of a barcode text that will be encoded into a series of structured append symbols. The value will

be used in the structured append block, and the structured append block will be used by each symbol in the series of QRCode structured append symbols. See also the "Structured append" section in the "TBarcode2D_QRCode" article.

Parameters:

- **Barcode**: string; It's the original input text before division into the each symbol in the series of structured append symbols.It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to caculate the parity of a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact a UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then caculate its parity. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the GetParity (Syntax 2) overloading method and specify the converted string in its Data parameter. If you want to caculate the parity of a block of binary (bytes) data, please use the GetParity (Syntax 2) overloading method.

- **AllowEscape**: Boolean; Specifies whether to allow users to insert the escape sequences to the Barcode parameter value, in order to place the function characters and additional control information. See also the "Escape sequences" section in the "TBarcode2D_QRCode" article.

- **InvalidIndex**: Integer; If there is any invalid character in the barcode text that is specified by the Barcode parameter, the parameter returns the position index of first invalid character, the index 1 denotes that the first character is invalid character. Otherwise, it returns the zero.

Return:

If the method succeeds, it returns the parity value, it's an 8-bit byte value. And the InvalidIndex parameter returns zero. If the method fails, the InvalidIndex parameter returns a position index of first invalid character, it's an integer value greater than 0.

See also the "Structured append" section in the "TBarcode2D_QRCode" article.

B.1.11.2 GetParity - Syntax 2

(TBarcode2D_QRCode)

Returns the parity value of a barcode text that will be encoded into a series of structured append symbols.

Syntax:

```
function GetParity(Data: AnsiString; AllowEscape: Boolean; var InvalidIndex: Integer):
  Byte; virtual;
```

Description:

The method returns the parity value of a barcode text that will be encoded into a series of structured append symbols. The value will be used in the structured append block, and the structured append block will be used by each symbol in the series of QRCode structured append symbols. See also the "Structured append" section in the "TBarcode2D_QRCode" article.

Parameters:

- **Data**: AnsiString; It's the original input text before division into the each symbol in the series of structured append symbols. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to caculate the parity of a block of binary (bytes) data. Or specifies the barcode text in AnsiString format.

If you want to caculate the parity of a block of binary (bytes) data under Delphi/C++ Builder 2007 or early, please use the GetParity (Syntax 1) overloading method.

- **AllowEscape**: Boolean; Specifies whether to allow users to insert the escape sequences to the Data parameter value, in order to place the function characters and additional control information. See also the "Escape sequences" section in the "TBarcode2D_QRCode" article.
- **InvalidIndex**: Integer; If there is any invalid character in the barcode text that is specified by the Data parameter, the parameter returns the position index of first invalid character, the index 1 denotes that the first character is invalid character. Otherwise, it returns the zero.

Return:

If the method succeeds, it returns the parity value, it's an 8-bit byte value. And the InvalidIndex parameter returns zero. If the method fails, the InvalidIndex parameter returns a position index of first invalid character, it's an integer value greater than 0.

See also the "Structured append" section in the "TBarcode2D_QRCode" article.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

B.1.12 Print

Prints specified barcode symbol to printer. There are several different overloading methods, Syntax 1, Syntax 2, and Syntax 3 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Prints the barcode symbol that is specified by the properties of this barcode component.
- **Syntax 2**: Prints the barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the Print (Syntax 3) overloading method and specify the converted string in its Data parameter. If you want to encode a block of binary (bytes) data, please use the Print (Syntax 3) overloading method.

- **Syntax 3**: Prints the barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method to encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you want to encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the Print (Syntax 2) overloading method.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Both of them will be printed to the printer.

B.1.12.1 Print - Syntax 1

Prints a barcode symbol to printer. The barcode symbol is specified by the properties of this barcode component. Please use the method between Printer.BeginDoc and Printer.EndDoc methods.

Syntax:

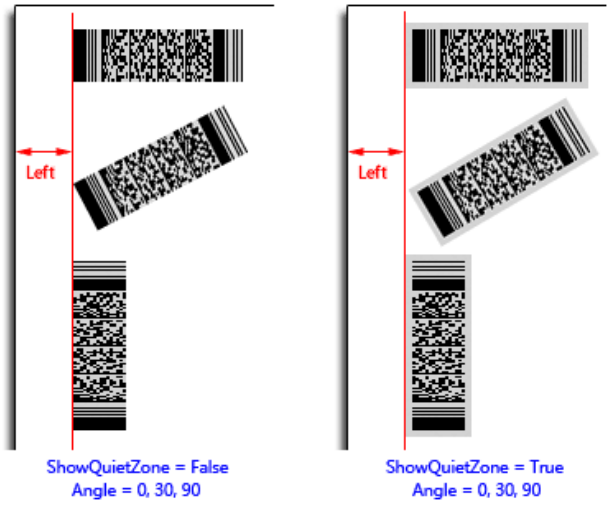
```
function Print(Left, Top, Module: Double; Angle: Integer = -1; BarcodeWidth: Double = 0;
  BarcodeHeight: Double = 0): Integer; overload; virtual;
```

Description:

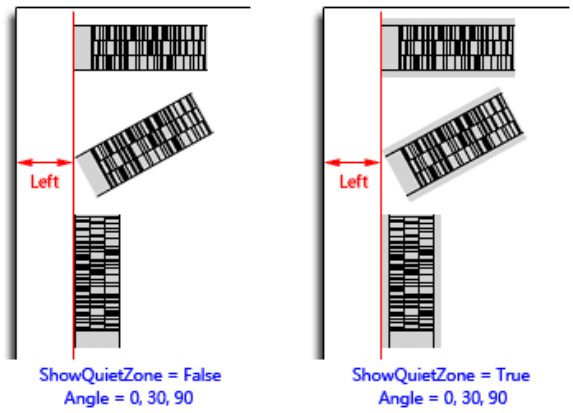
Prints current barcode symbol that is specified by the properties of this barcode component to the printer.

Parameters:

- **Left**: Double; Specifies the margin between the rotated barcode symbol and the left side of the paper in millimeters. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

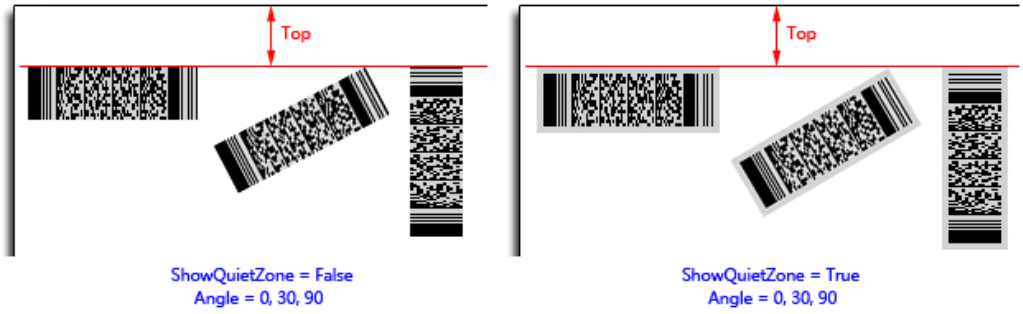


For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the left margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the left margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

See also the "LeftMargin" property.

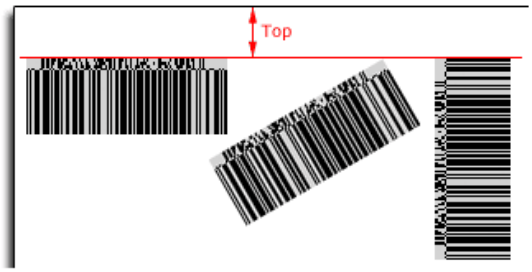
- **Top**: Double; Specifies the margin between the rotated barcode symbol and the top side of the paper in millimeters. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



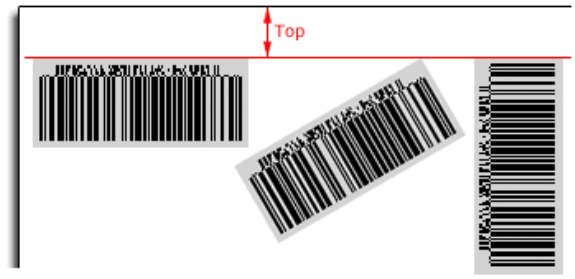
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the top margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor property is set to clSilver in order to

accentuate the quiet zones):



ShowQuietZone = False
Angle = 0, 30, 90



ShowQuietZone = True
Angle = 0, 30, 90

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the top margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

ShowQuietZone = False
Angle = 180, 210, 90

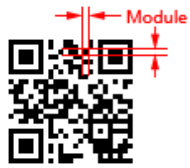
ShowQuietZone = True
Angle = 180, 210, 90

ShowQuietZone = False
Angle = 180, 210, 90

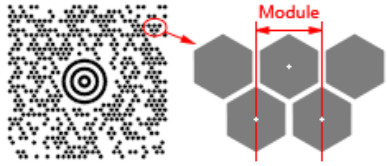
ShowQuietZone = True
Angle = 180, 210, 90

See also the "TopMargin" property.

- **Module**: Double; Specifies the module size in millimeters.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



- For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



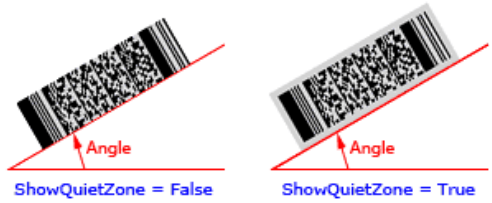
- For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

- For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

If any one of the BarcodeWidth parameter and BarcodeHeight parameter is provided and it isn't zero, the value of Module parameter will be ignored, the module size will be calculated based on the BarcodeWidth parameter value or the BarcodeHeight parmeter value. If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

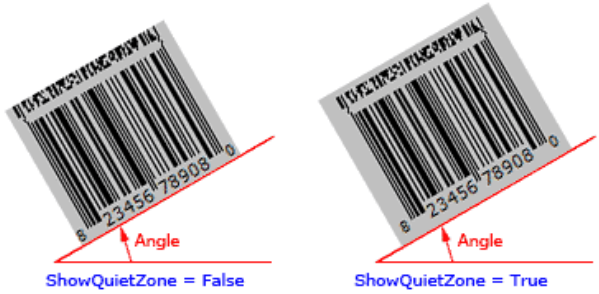
See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone property about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone property to specify whether to represent these quiet zones. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet azones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D

symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

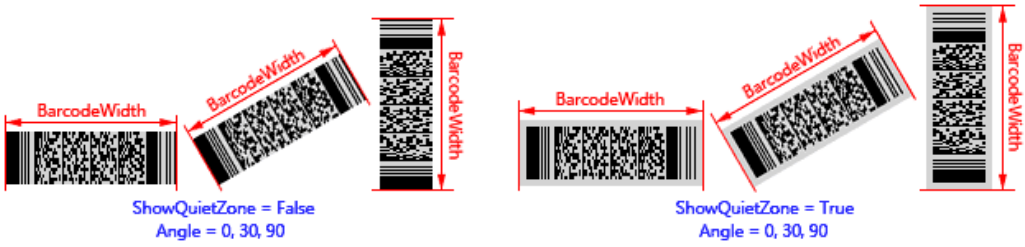


The parameter defaults to -1 if the Angle is not provided, and the barcode symbol will be rotated based on the value of the Orientation property:

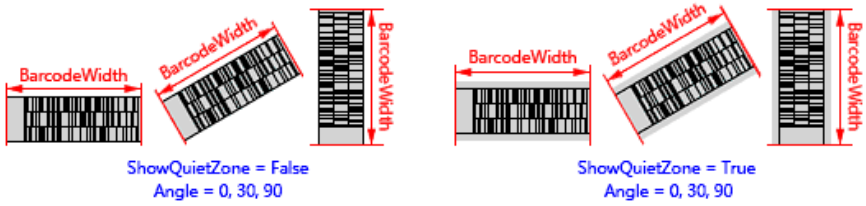
- boLeftRight: 0 degrees
- boRightLeft: 180 degrees
- boTopBottom: 270 degrees
- boBottomTop: 90 degrees

If you want to use the -1 degrees, the 359 degrees can be used instead.

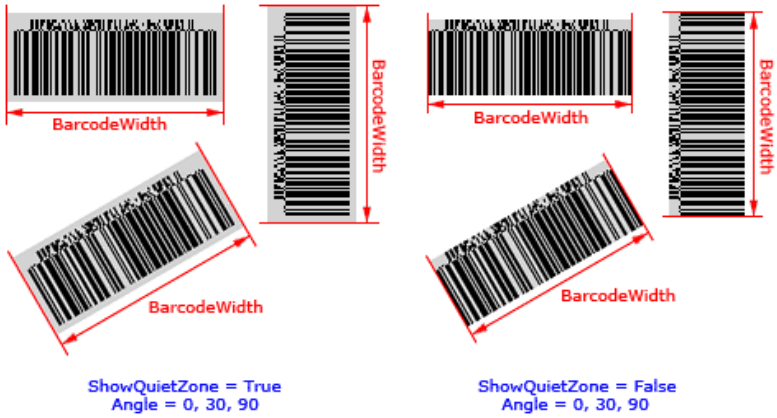
- **BarcodeWidth**: Double, Specifies the barcode symbol width (the distance between the beginning and end of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), the leading quiet zone and the trailing quiet zone are included in the barcode symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the width for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



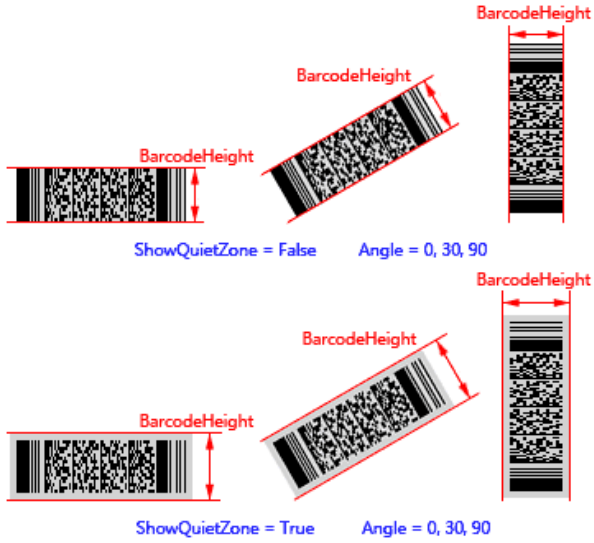
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess isn't included in the width value. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):

Note, if the parameter is provided and it isn't zero, the values in Module and BarcodeHeight parameters will be ignored, the module size will be calculated based on the BarcodeWidth value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol width will be determined by the Module parameter value.

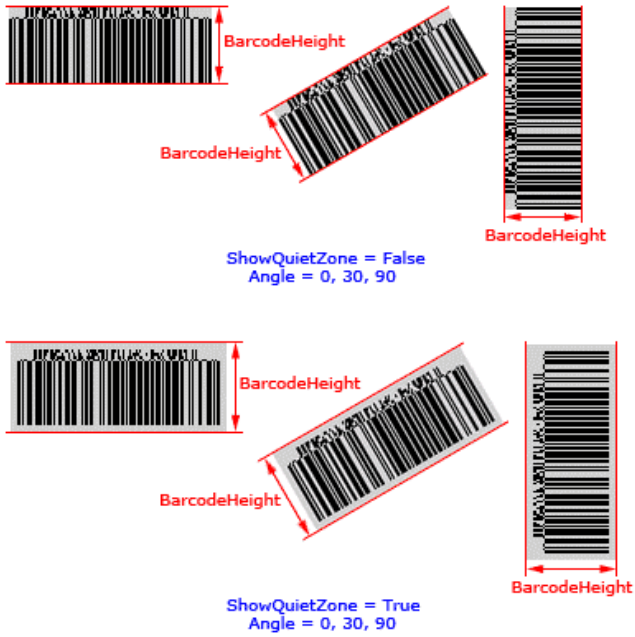
If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeWidth" property.

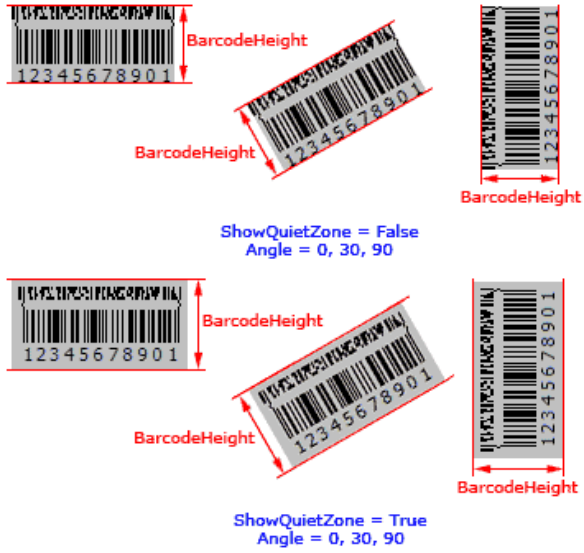
- **BarcodeHeight**: Double, Specifies the barcode symbol height (the distance between the top and bottom of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), the top quiet zone and the bottom quiet zone are included in the barcode symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the height for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the SpaceColor property is set to clSilver in order to accentuate the quiet zones):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. the human readable text and the vertical spacing between the text and linear symbol are included in the height value. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



Note, if the parameter is provided and it isn't zero, and the BarcodeWidth parameter isn't provided or it's zero, the value in Module parameter will be ignored, the module size will be calculated based on the BarcodeHeight value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol height will be determined by the Module parameter value.

If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeHeight" property.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property is invalid, the return valus is -1 or -65537. Corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.
- If there is any invalid character in the the barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character. Corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the

17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

Note:

Pelase use the method between Printer.BeginDoc and Printer.EndDoc methods. For example:

```
Printer.BeginDoc;
... { Print other content }
Barcode_QRCode1.Print(...); { Print the barcode symbol }
... { Print other content }
Printer.EndDoc;
```

B.1.12.2 Print - Syntax 2

Prints a barcode symbol to printer. The barcode symbol is specified by the parameters of this method. Please use the method between the Printer.BeginDoc and Printer.EndDoc methods.

Syntax:

```
function Print(Barcode: string; BarColor, SpaceColor: TColor; ShowQuietZone: Boolean;
  Left, Top, Module: Double; Angle: Integer = 0; BarcodeWidth: Double = 0; BarcodeHeight:
  Double = 0): Integer; overload; virtual;
```

Description:

Prints a barcode symbol that is specified by the parameters of this method to the printer.

Parameters:

- **Barcode**: String; Specifies the barcode text. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method to encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the Print (Syntax 3) overloading method and specify the converted string in its Data parameter. If you want to encode a block of binary (bytes) data, please use the Print (Syntax 3) overloading method.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **BarColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all bars

(Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

If the Inversed property is set to true, it specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

See also the "BarColor" property.

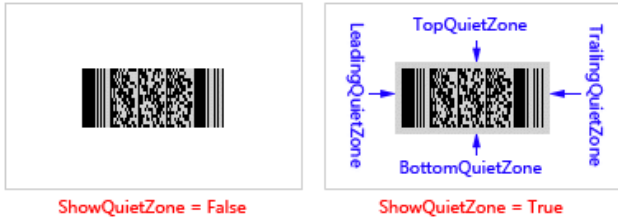
- **SpaceColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  If the Inversed property is set to true, it specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

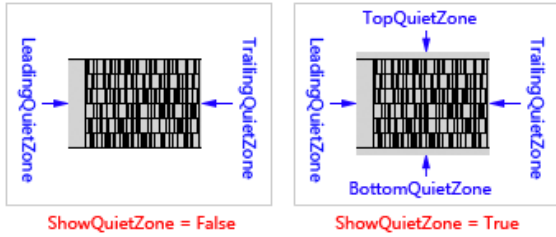
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "SpaceColor" property.

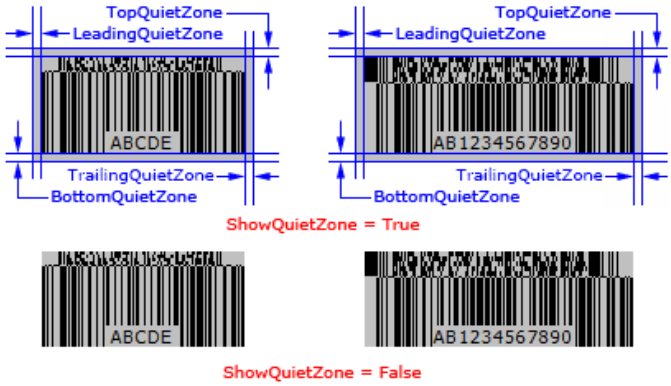
- **ShowQuietZone**: Boolean; Specifies whether to represent the leading, trailing, top and bottom quiet zones. If the parameter value is set to true, these quiet zones are represented. Otherwise, they don't be represented. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



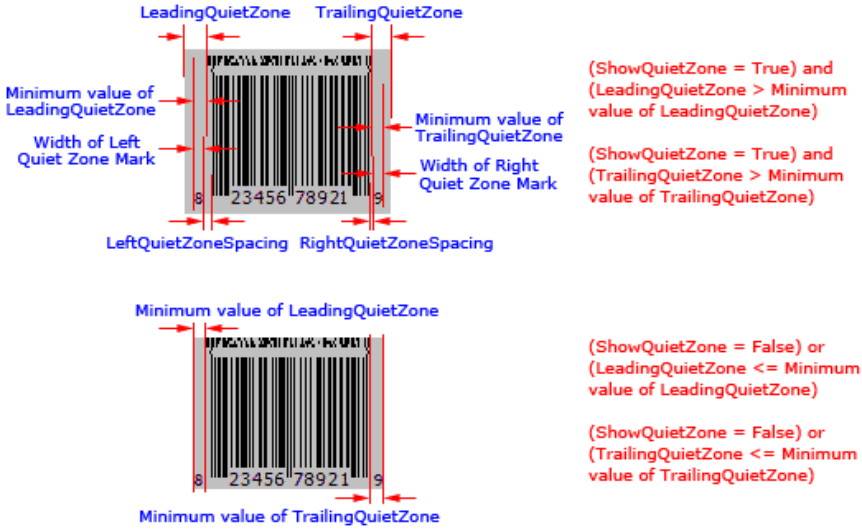
  For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be represented always, even if the ShowQuietZone parameter value is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



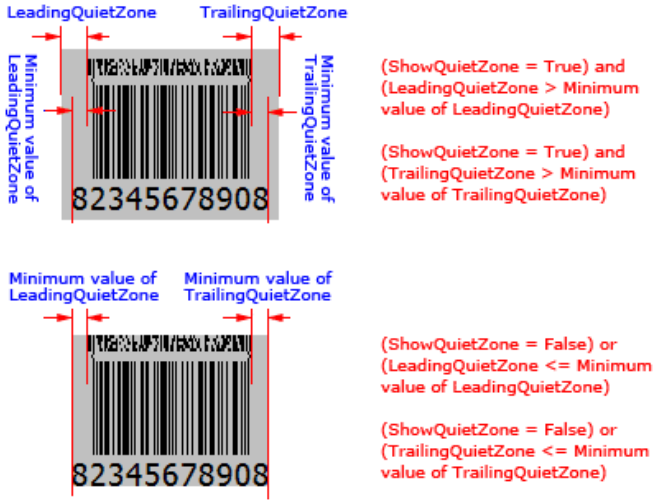
  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



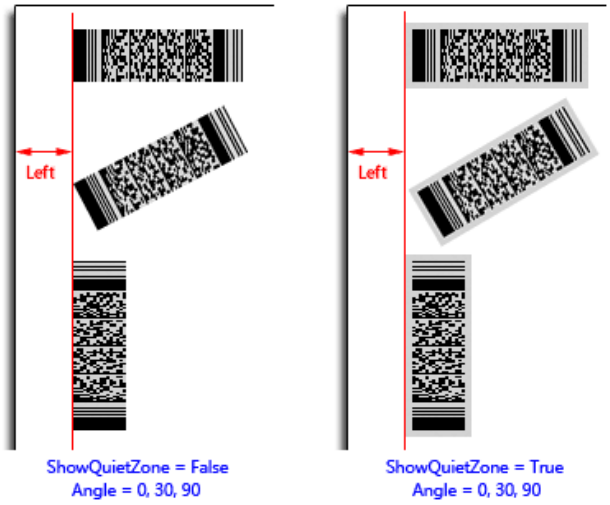
For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



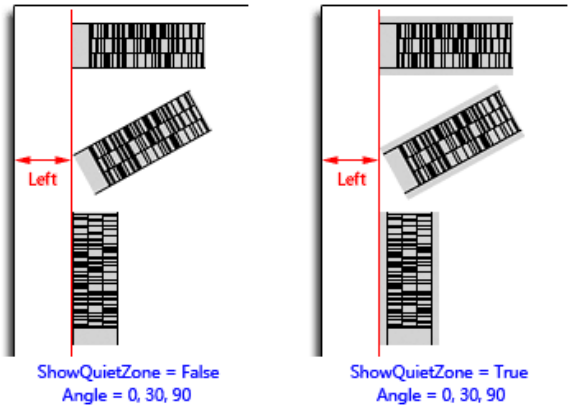
In general, the Inversed property is set to false. In this case, these quiet zones are represented using the color specified by SpaceColor parameter. If the Inversed property is set to true, these quiet zones are represented using the color specified by BarColor parameter.

See also the "ShowQuietZone" property.

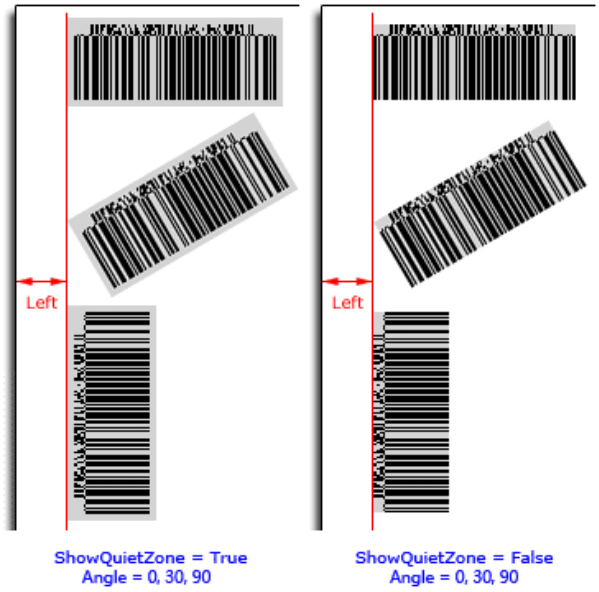
- **Left**: Double; Specifies the margin between the rotated barcode symbol and the left side of the paper in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



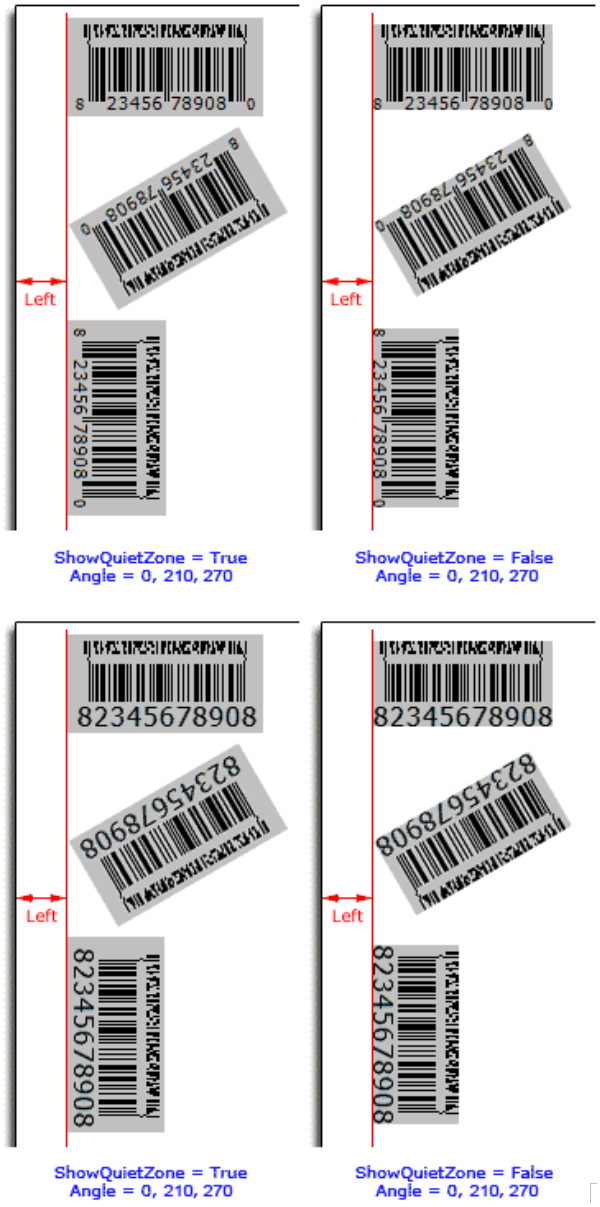
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the left margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

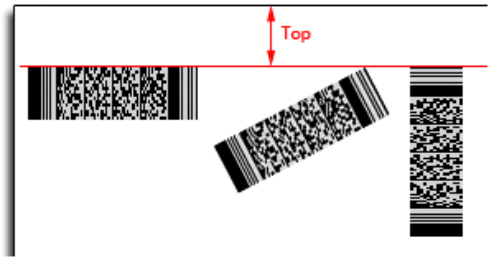


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the left margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

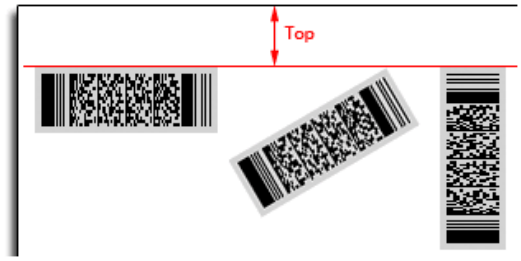


See also the "LeftMargin" property.

- **Top**: Double; Specifies the margin between the rotated barcode symbol and the top side of the paper in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

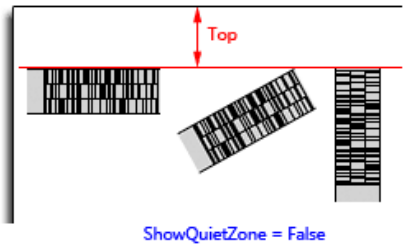


ShowQuietZone = False
Angle = 0, 30, 90

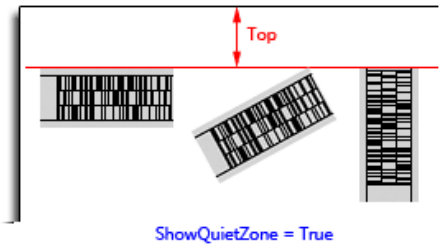


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

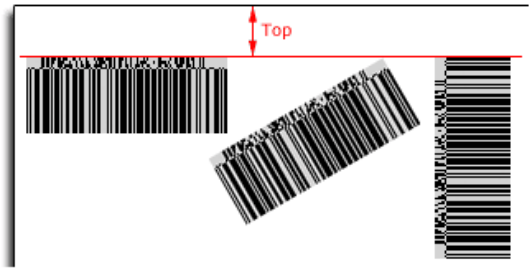


ShowQuietZone = False
Angle = 0, 30, 90

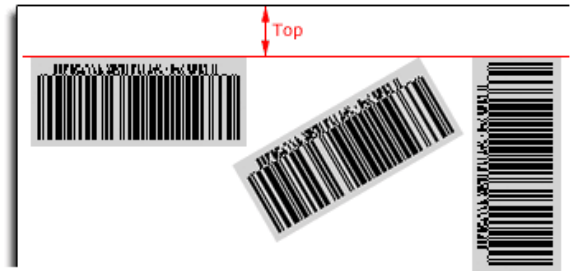


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the top margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



ShowQuietZone = False
Angle = 0, 30, 90



ShowQuietZone = True
Angle = 0, 30, 90

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the top margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

ShowQuietZone = False
Angle = 180, 210, 90

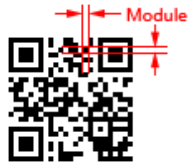
ShowQuietZone = True
Angle = 180, 210, 90

ShowQuietZone = False
Angle = 180, 210, 90

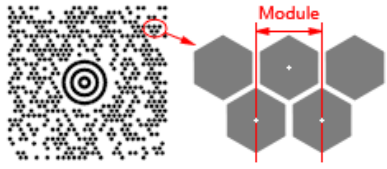
ShowQuietZone = True
Angle = 180, 210, 90

See also the "TopMargin" property.

- **Module**: Double; Specifies the module size in millimeters.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

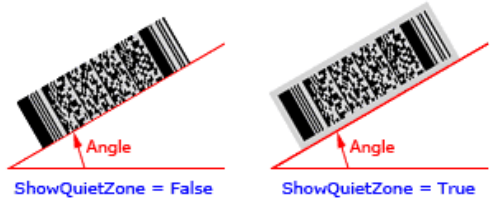
  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the

2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

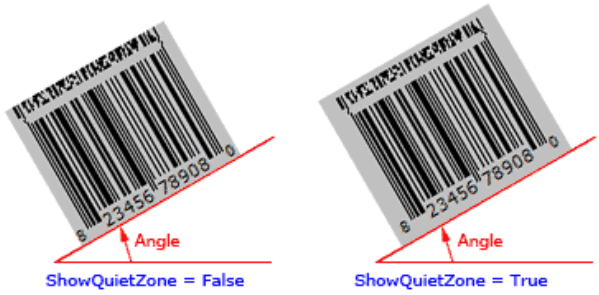
If any one of the BarcodeWidth parameter and BarcodeHeight parameter is provided and it isn't zero, the value of Module parameter will be ignored, the module size will be calculated based on the BarcodeWidth parameter value or the BarcodeHeight parmeter value. If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



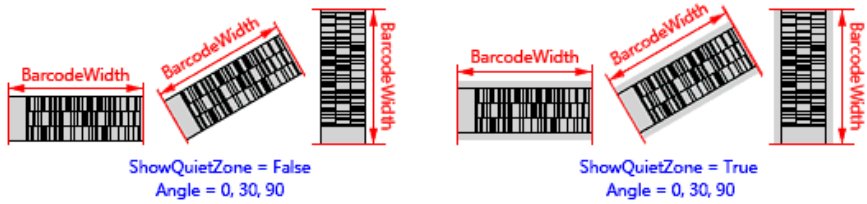
For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



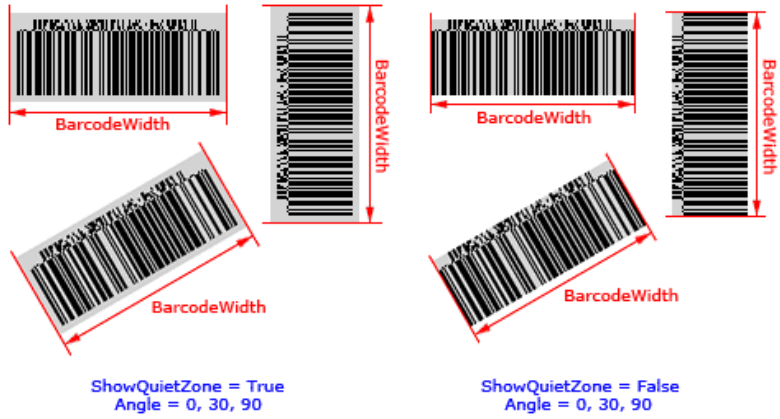
The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **BarcodeWidth**: Double, Specifies the barcode symbol width (the distance between the beginning and end of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the leading quiet zone and the trailing quiet zone are included in the barcode symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the width for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



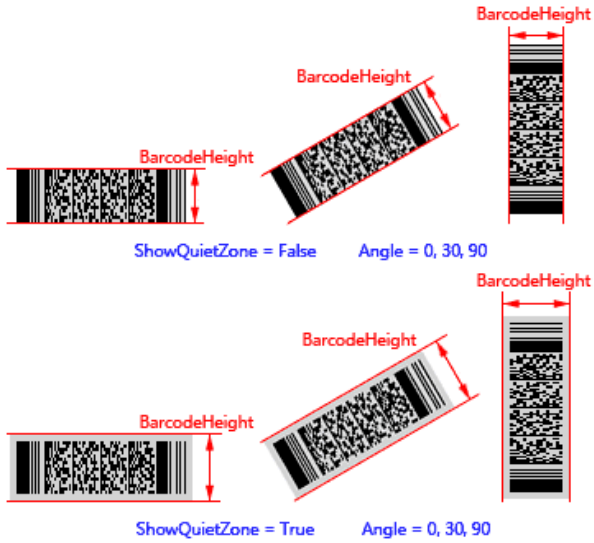
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess isn't included in the width value. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

Note, if the parameter is provided and it isn't zero, the values in Module and BarcodeHeight parameters will be ignored, the module size will be calculated based on the BarcodeWidth value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol width will be determined by the Module parameter value.

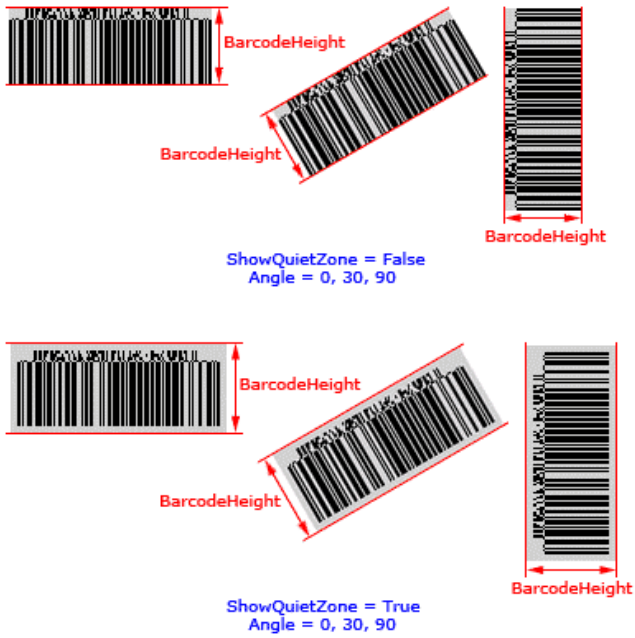
If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeWidth" property.

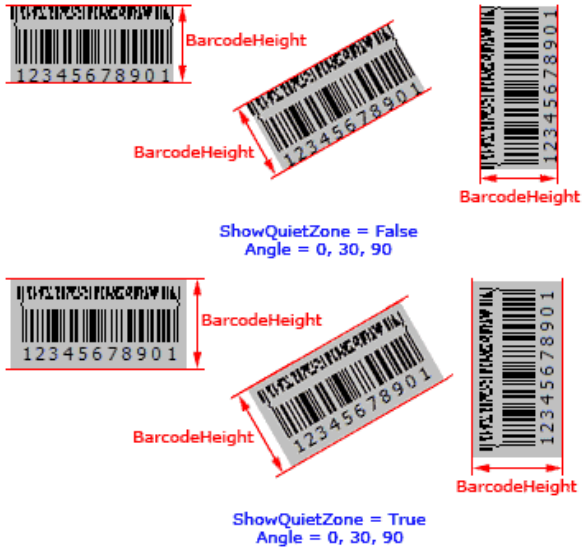
- **BarcodeHeight**: Double, Specifies the barcode symbol height (the distance between the top and bottom of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the top quiet zone and the bottom quiet zone are included in the barcode symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the height for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. the human readable text and the vertical spacing between the text and linear symbol are included in the height value. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



Note, if the parameter is provided and it isn't zero, and the BarcodeWidth parameter isn't provided or it's zero, the value in Module parameter will be ignored, the module size will be calculated based on the BarcodeHeight value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol height will be determined by the Module parameter value.

If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeHeight" property.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

Note:

Pelase use the method between Printer.BeginDoc and Printer.EndDoc methods. For example:

```
Printer.BeginDoc;
```

```
... { Print other content }
Barcode_QRCode1.Print(...); { Print the barcode symbol }
... { Print other content }
Printer.EndDoc;
```

B.1.12.3 Print - Syntax 3

Prints a barcode symbol to printer. The barcode symbol is specified by the parameters of this method. Please use the method between the Printer.BeginDoc and Printer.EndDoc methods.

Syntax:

```
function Print(Data: AnsiString; BarColor, SpaceColor: TColor; ShowQuietZone: Boolean;
  Left, Top, Module: Double; Angle: Integer = 0; BarcodeWidth: Double = 0; BarcodeHeight:
  Double = 0): Integer; overload; virtual;
```

Description:

Prints a barcode symbol that is specified by the parameters of this method to the printer.

Parameters:

- **Data**: AnsiString; Specifies the barcode text. It is of type AnsiString.

  For the Delphi/C++ Builder 2009 or later, you can use the method to encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you want to encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the Print (Syntax 2) overloading method.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **BarColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

  If the Inversed property is set to true, it specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

  See also the "BarColor" property.

- **SpaceColor**: TColor; In general, the Inversed property is set to false. In this case, the parameters specifies the color for all spaces (Stacked 2D barcode symbology and Linear 1D barcode symbology) or light modules (Matrix 2D barcode

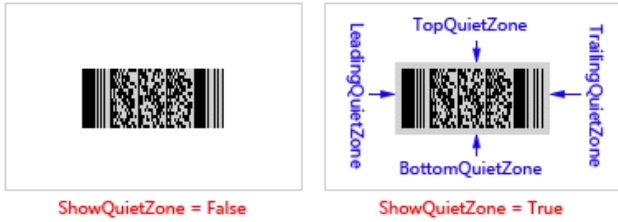
symbology) in the barcode symbol. Also, if the ShowQuietZone parameter value is set to true, the leading, trailing, top, and bottom quiet zones are represented using the color.

If the Inversed property is set to true, it specifies the color for all bars (Stacked 2D barcode symbology and Linear 1D barcode symbology) or dark modules (Matrix 2D barcode symbology) in the barcode symbol.

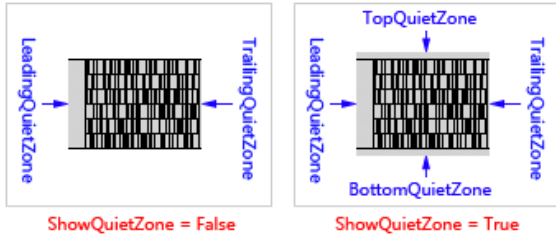
For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter is in effect for entire EAN.UCC composite symbol (including both 2D and linear symbols).

See also the "SpaceColor" property.

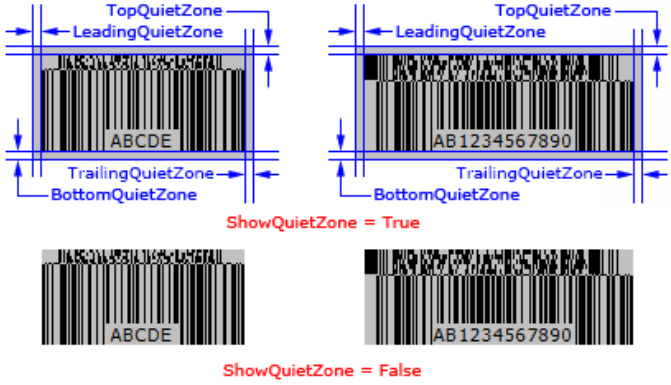
- **ShowQuietZone**: Boolean; Specifies whether to represent the leading, trailing, top and bottom quiet zones. If the parameter value is set to true, these quiet zones are represented. Otherwise, they don't be represented. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be represented always, even if the ShowQuietZone parameter value is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

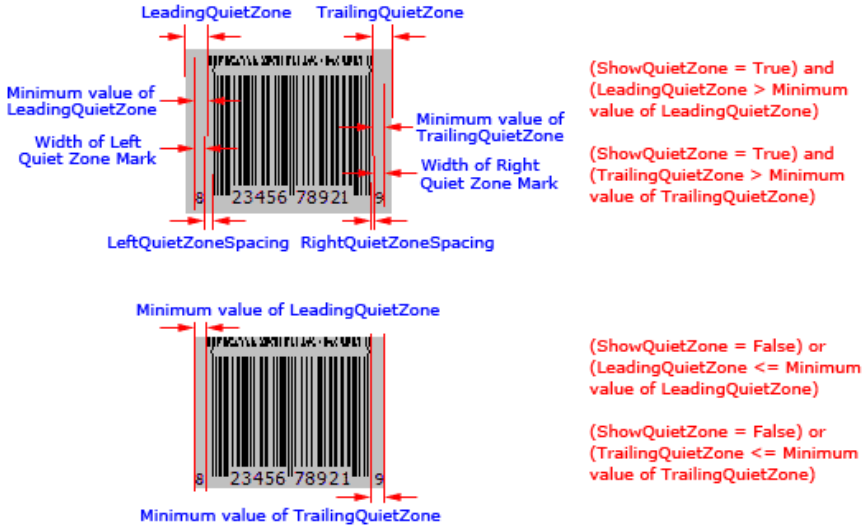


For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

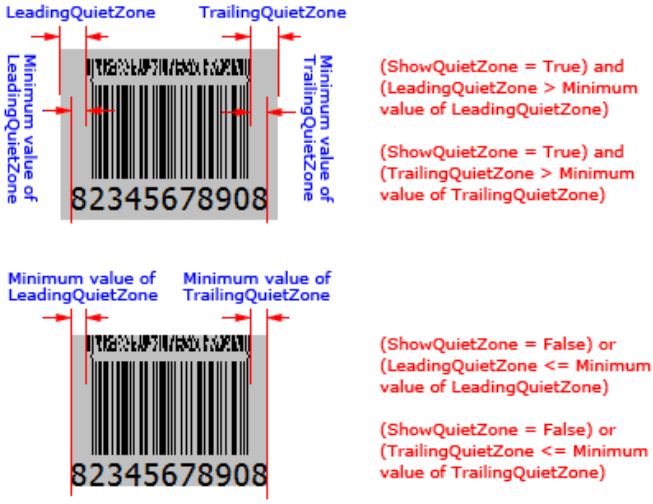


For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE,

TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be represented even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



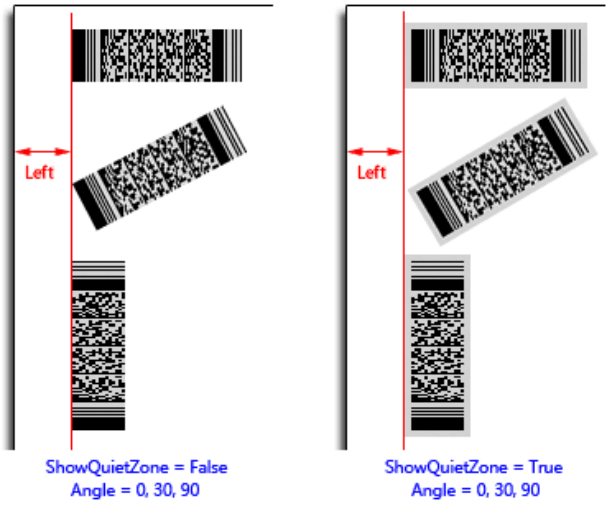
For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



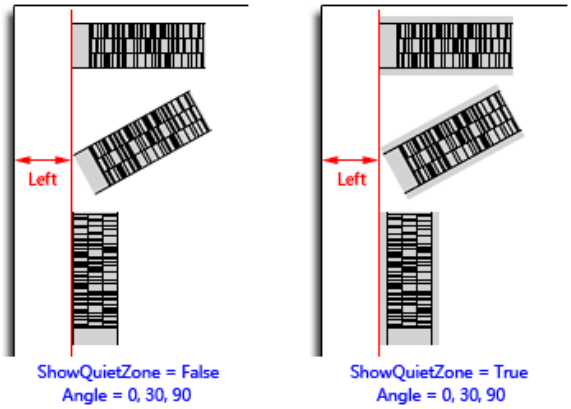
In general, the Inversed property is set to false. In this case, these quiet zones are represented using the color specified by SpaceColor parameter. If the Inversed property is set to true, these quiet zones are represented using the color specified by BarColor parameter.

See also the "ShowQuietZone" property.

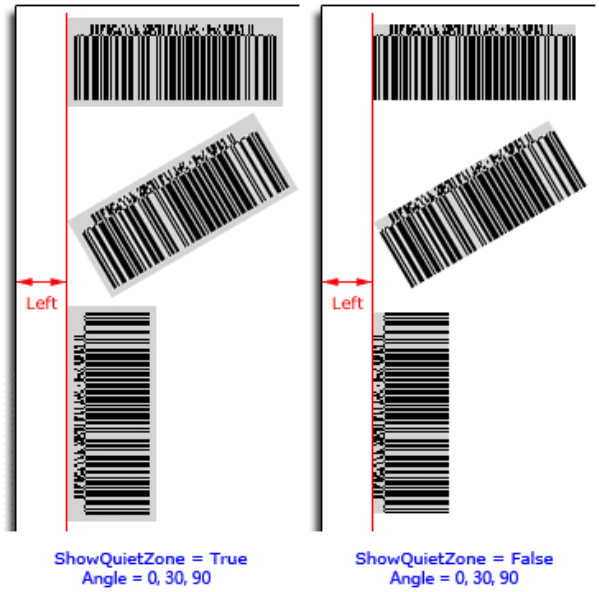
- **Left**: Double; Specifies the margin between the rotated barcode symbol and the left side of the paper in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



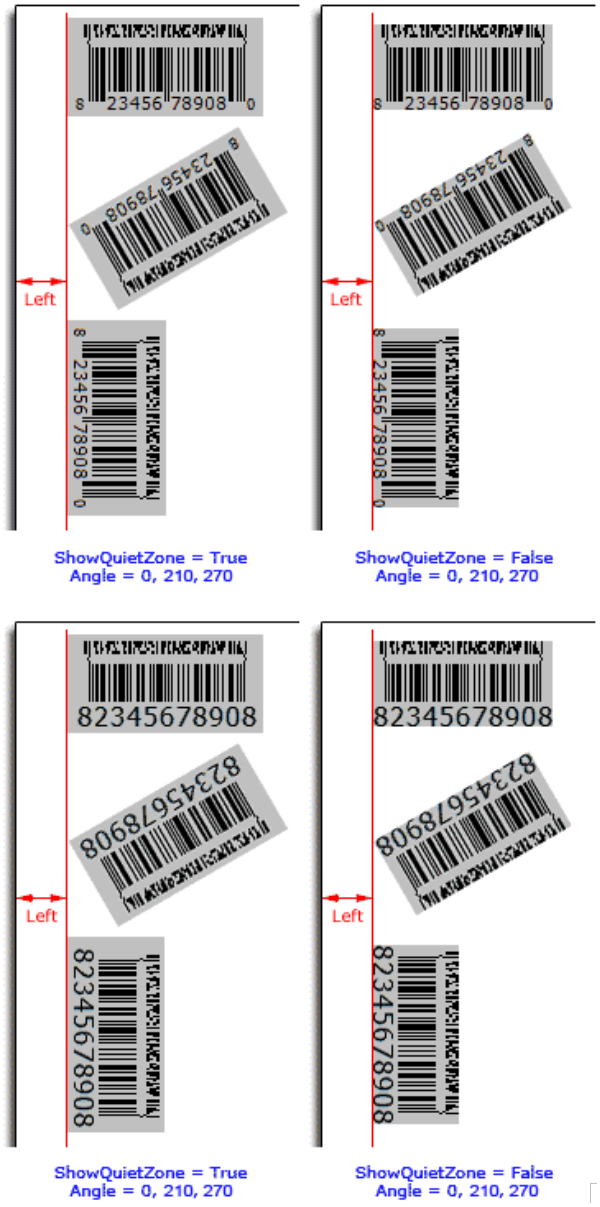
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the left margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

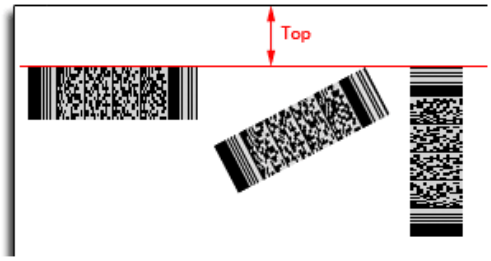


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the left margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

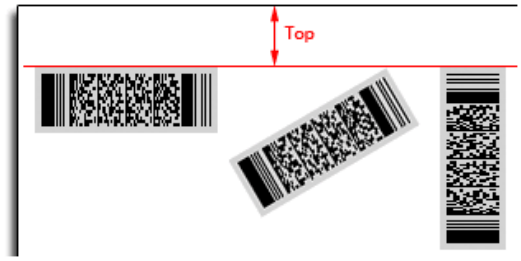


See also the "LeftMargin" property.

- **Top**: Double; Specifies the margin between the rotated barcode symbol and the top side of the paper in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

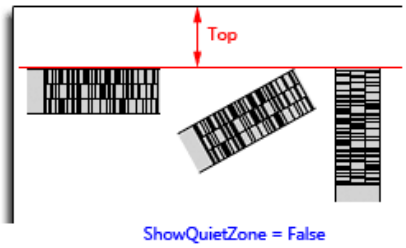


ShowQuietZone = False
Angle = 0, 30, 90

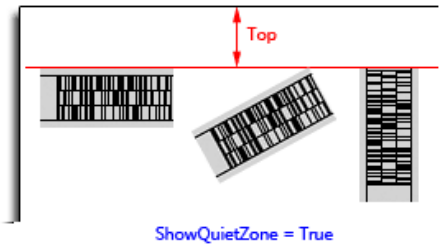


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

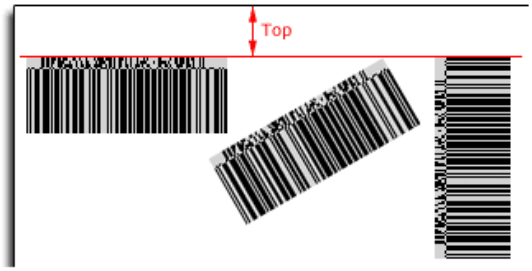


ShowQuietZone = False
Angle = 0, 30, 90

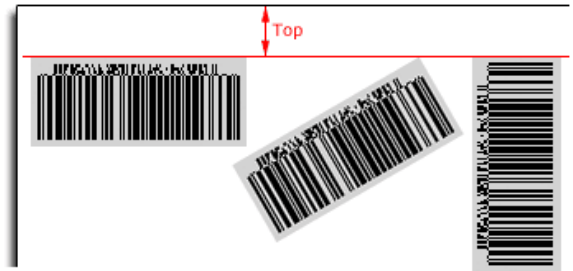


ShowQuietZone = True
Angle = 0, 30, 90

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The linear symbol is included in the composite barcode symbol, and the parameter specifies the top margin for entire rotated EAN.UCC composite barcode symbol not for the 2D symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



ShowQuietZone = False
Angle = 0, 30, 90



ShowQuietZone = True
Angle = 0, 30, 90

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included in the symbol too (if they are represented, this does not apply to TBarcode1D_EAN128). The parameter always specifies the top margin for entire rotated EAN.UCC composite symbol. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):

ShowQuietZone = False
Angle = 180, 210, 90

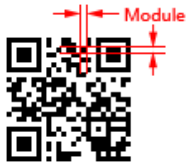
ShowQuietZone = True
Angle = 180, 210, 90

ShowQuietZone = False
Angle = 180, 210, 90

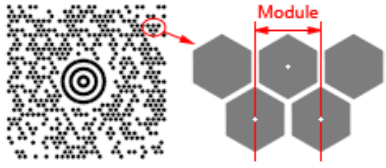
ShowQuietZone = True
Angle = 180, 210, 90

See also the "TopMargin" property.

- **Module**: Double; Specifies the module size in millimeters.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

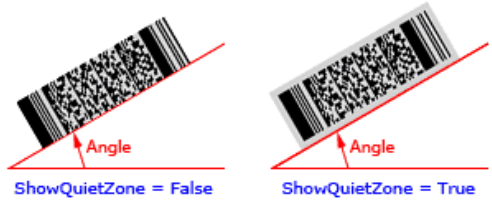
  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the

2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

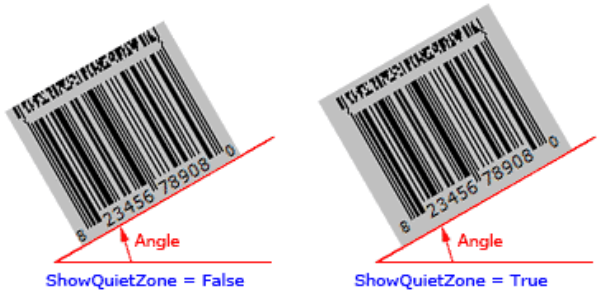
If any one of the BarcodeWidth parameter and BarcodeHeight parameter is provided and it isn't zero, the value of Module parameter will be ignored, the module size will be calculated based on the BarcodeWidth parameter value or the BarcodeHeight parmeter value. If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



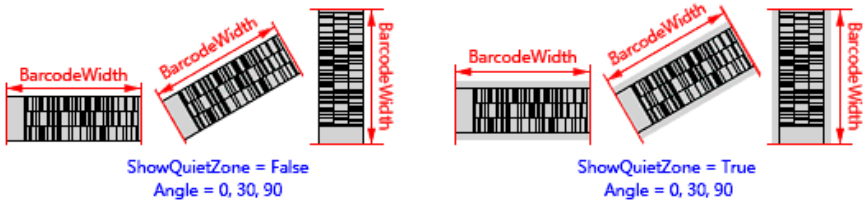
For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the SpaceColor parameter value is set to clSilver in order to accentuate the quiet zones):



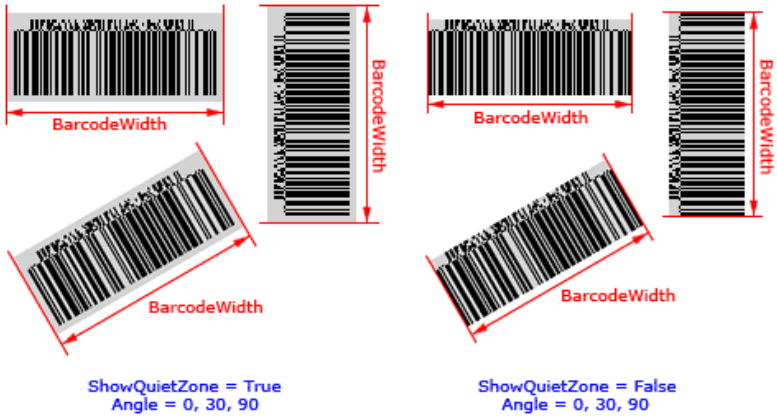
The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **BarcodeWidth**: Double, Specifies the barcode symbol width (the distance between the beginning and end of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the leading quiet zone and the trailing quiet zone are included in the barcode symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the width for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



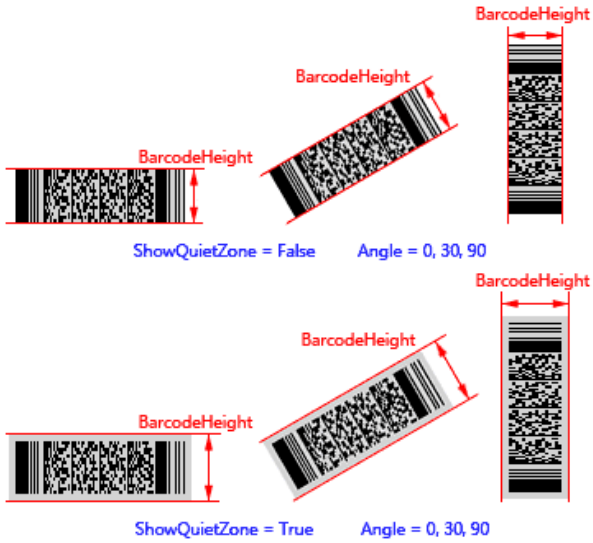
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess isn't included in the width value. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):

Note, if the parameter is provided and it isn't zero, the values in Module and BarcodeHeight parameters will be ignored, the module size will be calculated based on the BarcodeWidth value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol width will be determined by the Module parameter value.

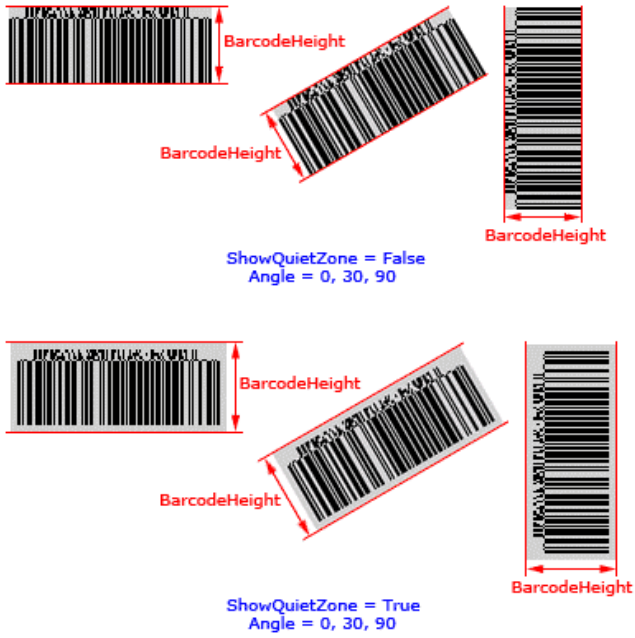
If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeWidth" property.

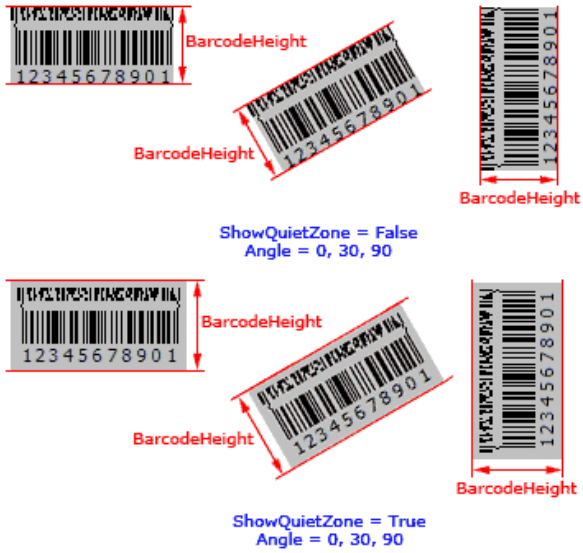
- **BarcodeHeight**: Double, Specifies the barcode symbol height (the distance between the top and bottom of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the top quiet zone and the bottom quiet zone are included in the barcode symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the height for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. the human readable text and the vertical spacing between the text and linear symbol are included in the height value. See diagram (the SpaceColor parameter is set to clSilver in order to accentuate the quiet zones):



Note, if the parameter is provided and it isn't zero, and the BarcodeWidth parameter isn't provided or it's zero, the value in Module parameter will be ignored, the module size will be calculated based on the BarcodeHeight value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol height will be determined by the Module parameter value.

If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeHeight" property.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.
- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

Pelase use the method between Printer.BeginDoc and Printer.EndDoc methods. For example:

```
Printer.BeginDoc;
... { Print other content }
Barcode_QRCode1.Print(...); { Print the barcode symbol }
... { Print other content }
Printer.EndDoc;
```

B.1.13 PrintSize

Returns actual horizontal width and vertical height of a rotated barcode symbol in millimeters. There are several different overloading methods, Syntax 1, Syntax 2, and Syntax 3 (only for Delphi/C++Builder 2009 or later):

- **Syntax 1**: Returns the actual print horizontal width and vertical height of the rotated barcode symbol that is specified by the properties of this barcode component.
- **Syntax 2**: Returns the actual print horizontal width and vertical height of the rotated barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Barcode parameter. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method if you encode a block of binary (bytes) data.

  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the PrintSize (Syntax 3) overloading method and specify the converted string in its Data parameter. If you encode a block of binary (bytes) data, please use the PrintSize (Syntax 3) overloading method.

- **Syntax 3**: Returns the actual print horizontal width and vertical height of the rotated barcode symbol that is specified by the parameters of this method. The barcode text is specified in the Data parameter. It is of type AnsiString. The method is available only for the Delphi/C++ Builder 2009 or later.

  For the Delphi/C++ Builder 2009 or later, you can use the method if you encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the PrintSize (Syntax 2) overloading method.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol.

B.1.13.1 PrintSize - Syntax 1

Returns actual horizontal width and vertical height of a rotated barcode symbol in millimeters. The barcode symbol is specified by the properties of this barcode component.

Syntax:

```
function PrintSize(var Width, Height, SymbolWidth, SymbolHeight: Double; Module: Double;
```

```
Angle: Integer = -1; BarcodeWidth: Double = 0; BarcodeHeight: Double = 0; HDPI: Integer
= 0; VDPI: Integer = 0): Integer; overload; virtual;
```

Description:

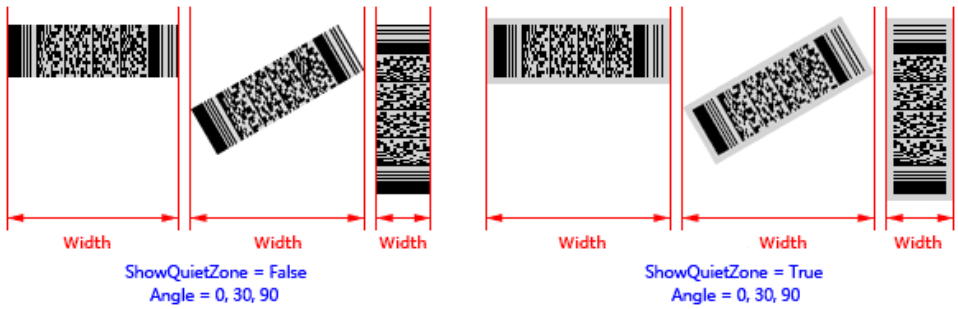
The method returns the actual horizontal width and vertical height of a rotated barcode symbol that is specified by properties of this barcode component, in millimeters.

For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC. If its Linear property is set to the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and the linear component's DisplayText property isn't set to dtNone, Please use the method between the Printer.BeginDoc and Printer.EndDoc methods.

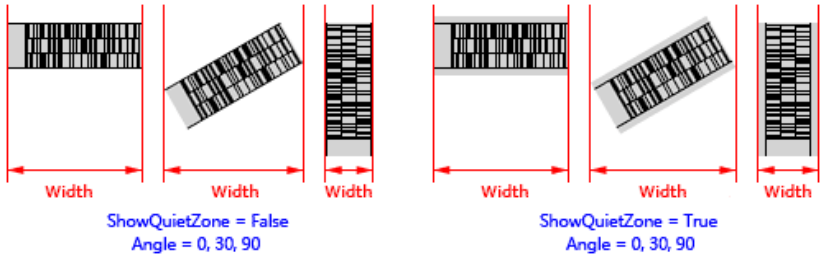
Parameters:

- **Width**: Double; Returns the horizontal width of the rotated barcode symbol in millimeters. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

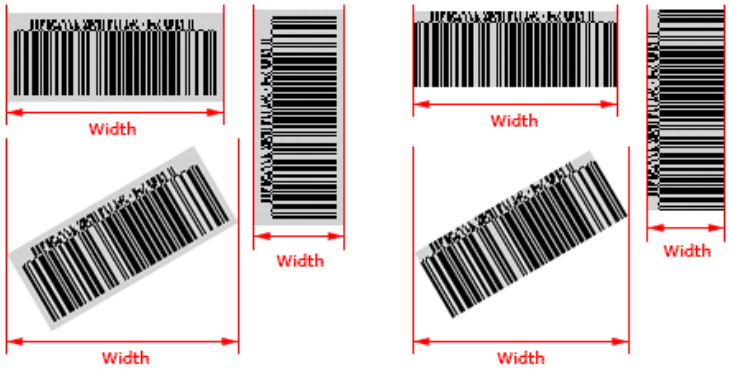
  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



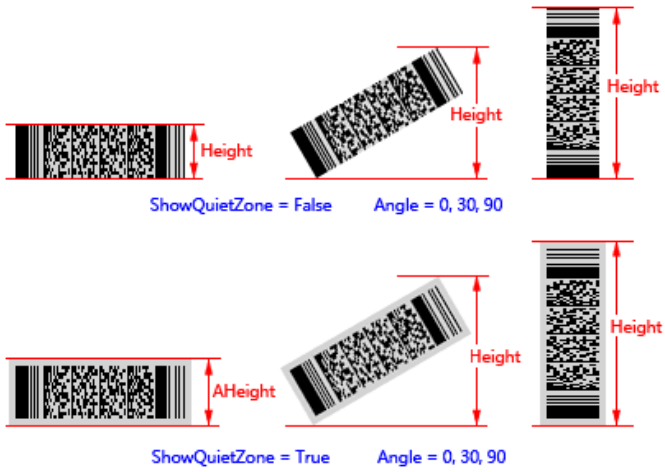
  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



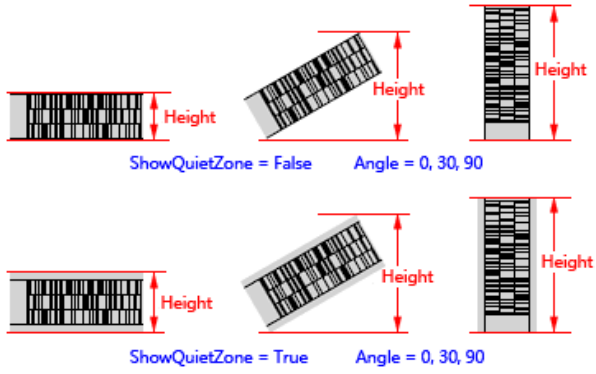
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

- **Height**: Double; Returns the vertical height of the rotated barcode symbol in millimeters. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

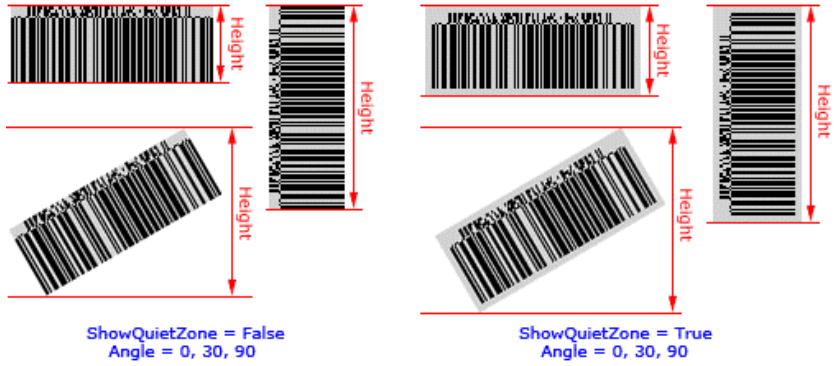
  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



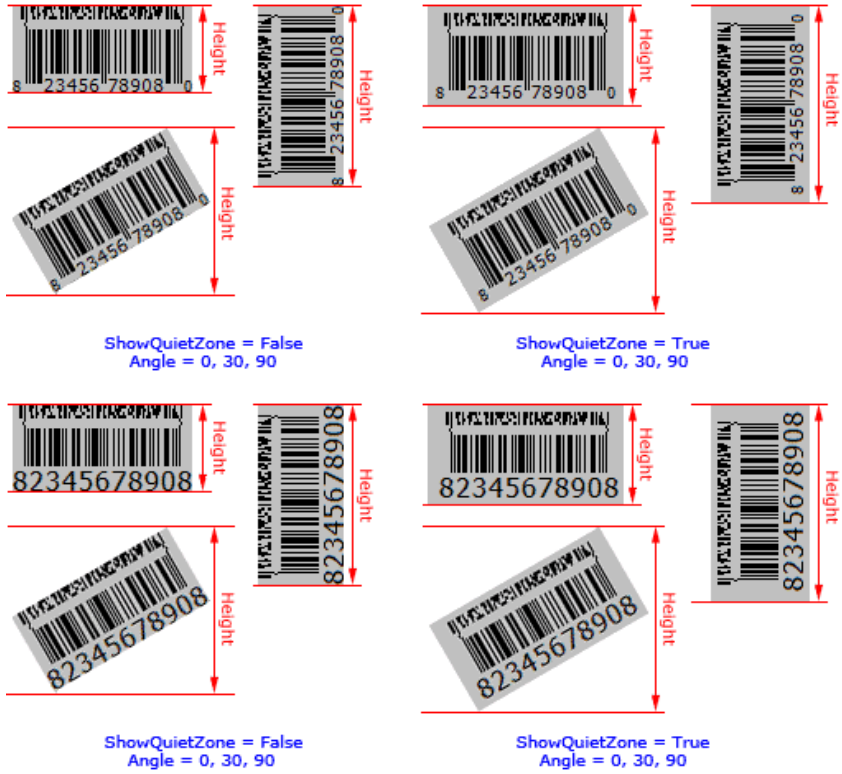
  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

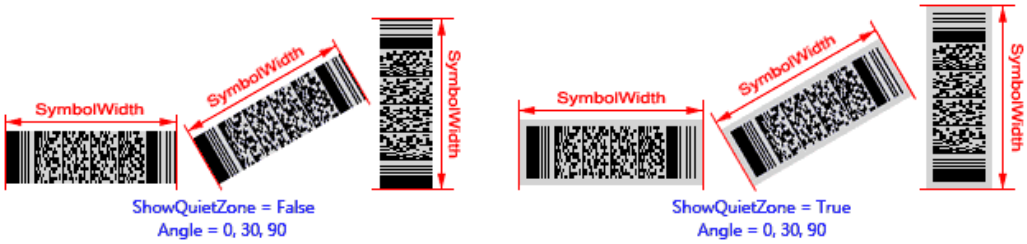


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

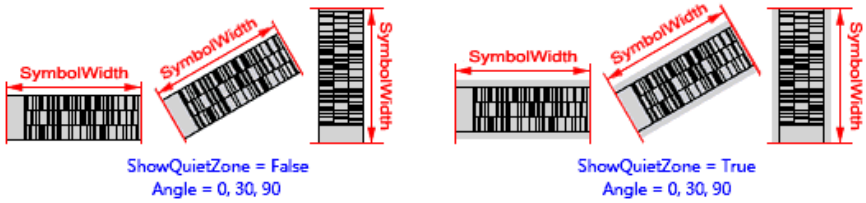


- **SymbolWidth**: Double; Returns the distance between the leading and trailing of the rotated barcode symbol in millimeter. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

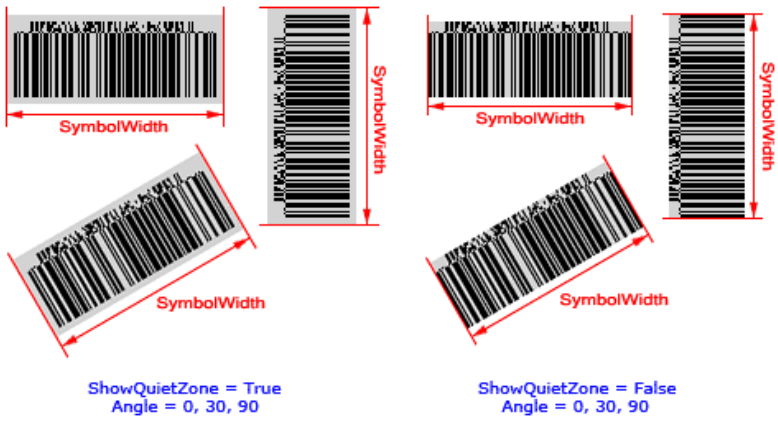
  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire rotated EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

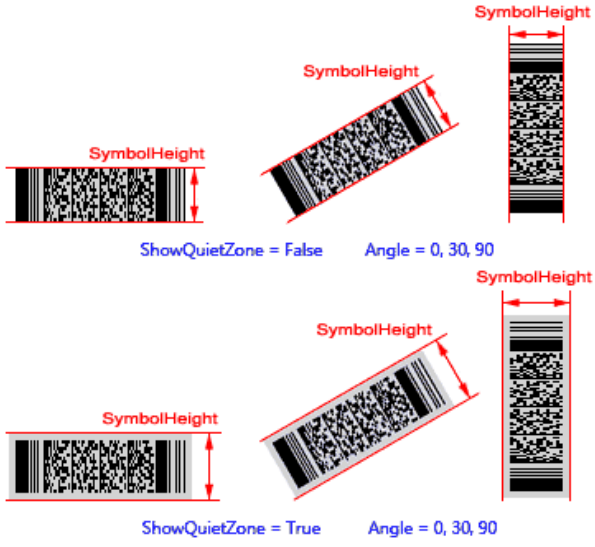


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

See also the "BarcodeWidth" property.

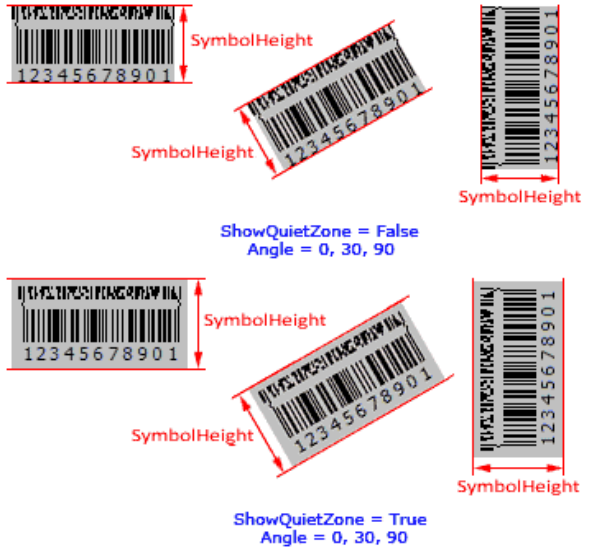
- **SymbolHeight**: Double; Returns the distance between the top and bottom of the rotated barcode symbol in millimeter. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

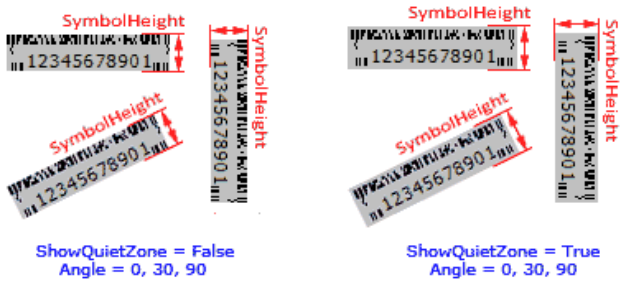
  If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire rotated EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

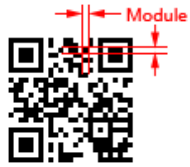
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented, the human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



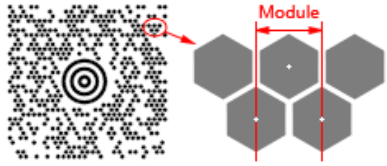


See also the "BarcodeHeight" property.

- **Module**: Double; Specifies the module size in millimeters.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

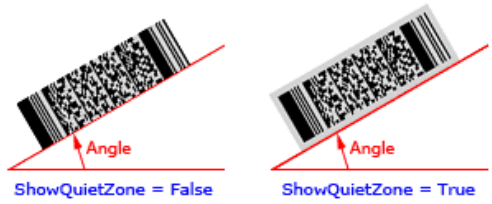
  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

If any one of the BarcodeWidth parameter and BarcodeHeight parameter is provided and it isn't zero, the value of Module parameter will be ignored, the module size will be calculated based on the BarcodeWidth parameter value or the BarcodeHeight parmeter value. If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the

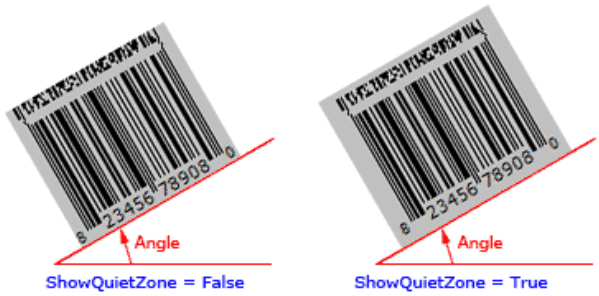
BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone property about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone property to specify whether to represent these quiet zones. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



The parameter defaults to -1 if the Angle is not provided, and the barcode symbol will be rotated based on the value of the Orientation property:

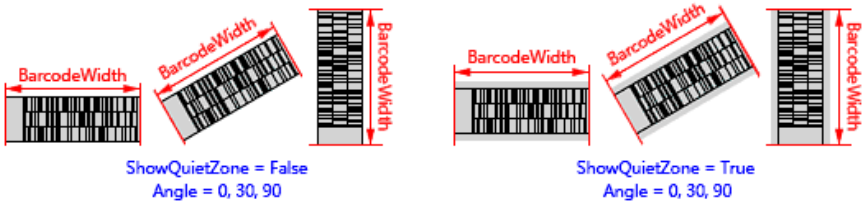
- boLeftRight: 0 degrees
- boRightLeft: 180 degrees
- boTopBottom: 270 degrees
- boBottomTop: 90 degrees

If you want to use the -1 degrees, the 359 degrees can be used instead.

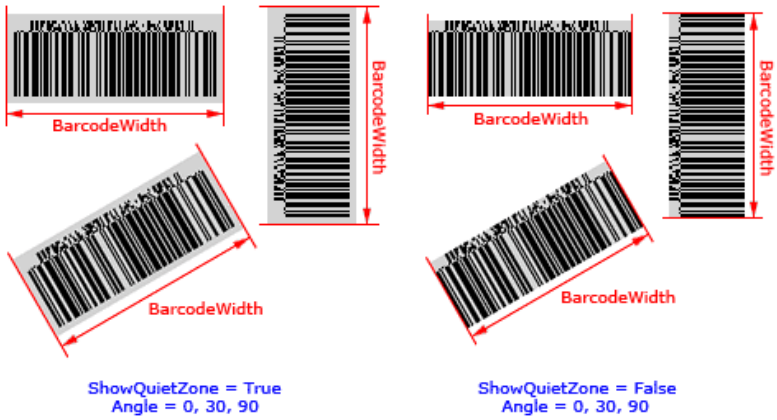
- **BarcodeWidth**: Double, Specifies the barcode symbol width (the distance between the beginning and end of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), the leading quiet zone and the trailing quiet zone are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the quiet zones are drawn in clSilver color in order to

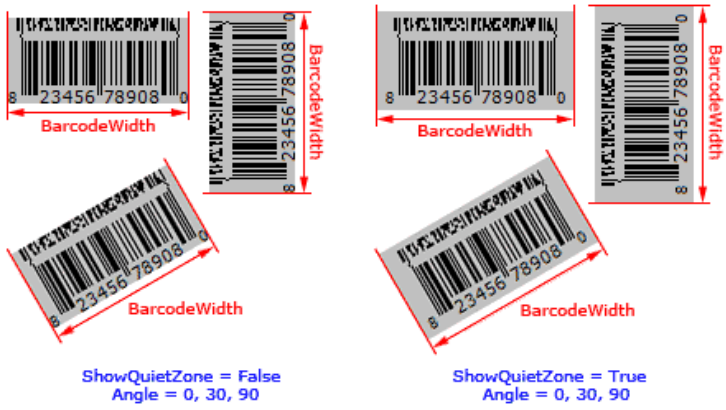
accentuate them):

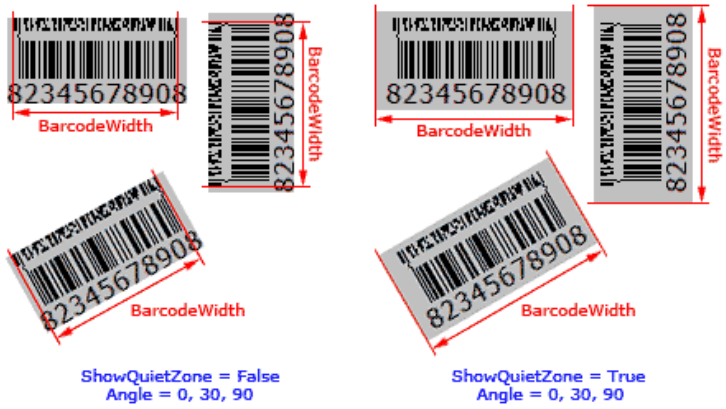


For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the width for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess isn't included in the width value. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



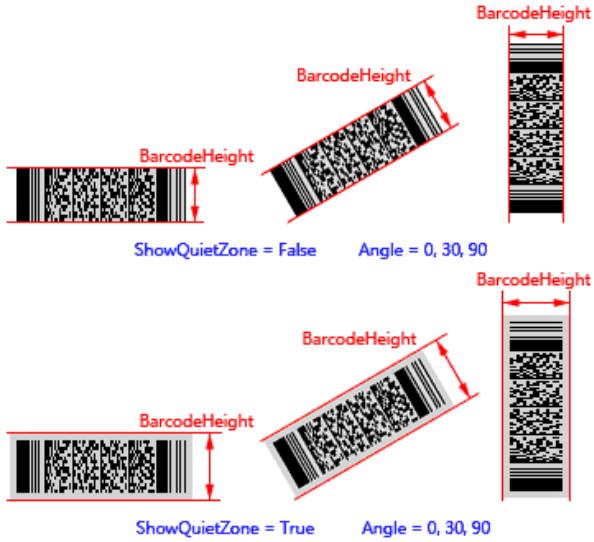


Note, if the parameter is provided and it isn't zero, the values in Module and BarcodeHeight parameters will be ignored, the module size will be calculated based on the BarcodeWidth value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol width will be determined by the Module parameter value.

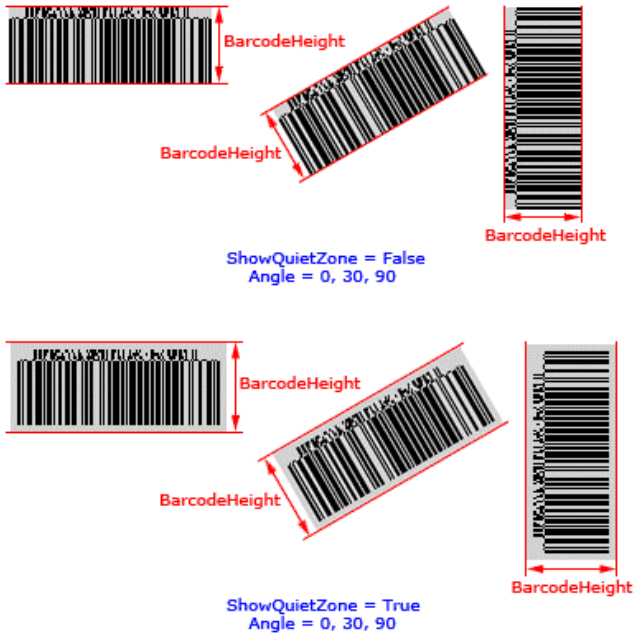
If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeWidth" property.

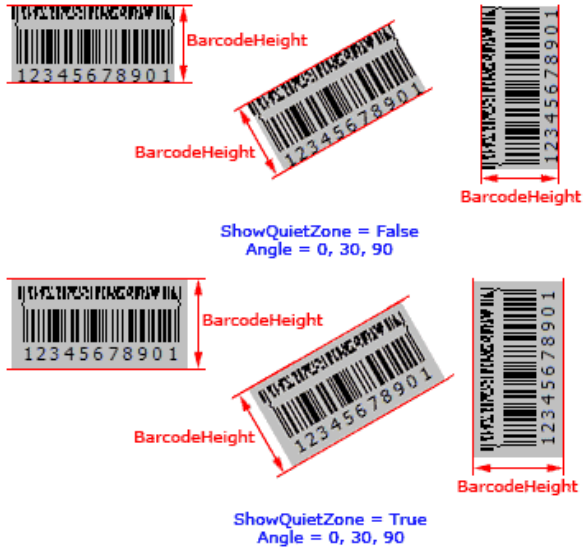
- **BarcodeHeight**: Double, Specifies the barcode symbol height (the distance between the top and bottom of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), the top quiet zone and the bottom quiet zone are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the height for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. the human readable text and the vertical spacing between the text and linear symbol are included in the height value. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



Note, if the parameter is provided and it isn't zero, and the BarcodeWidth parameter isn't provided or it's zero, the value in Module parameter will be ignored, the module size will be calculated based on the BarcodeHeight value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol height will be determined by the Module parameter value.

If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeHeight" property.

- **HDPI**: Integer, Specifies the horizontal resolution of printer in DPI (dots per inch).

  It defaults to 0 if the HDPI is not provided, and the horizontal resolution will be obtained from your printer.

- **VDPI**: Integer, Specifies the vertical resolution of printer in DPI (dots per inch).

  It defaults to 0 if the VDPI is not provided, and the vertical resolution will be obtained from your printer.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property is invalid, the return valus is -1 or -65537. Corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character. Corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.13.2 PrintSize - Syntax 2

Returns the actual horizontal width and vertical height of a rotated barcode symbol in millimeters. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function PrintSize(var Width, Height, SymbolWidth, SymbolHeight: Double; Barcode: string;
  ShowQuietZone: Boolean; Module: Double; Angle: Integer = 0; BarcodeWidth: Double = 0;
  BarcodeHeight: Double = 0; HDPI: Integer = 0; VDPI: Integer = 0): Integer; overload;
  virtual;
```

Description:

The method returns the actual horizontal width and vertical height of a rotated barcode symbol that is specified by parameters of this method, in millimeters.

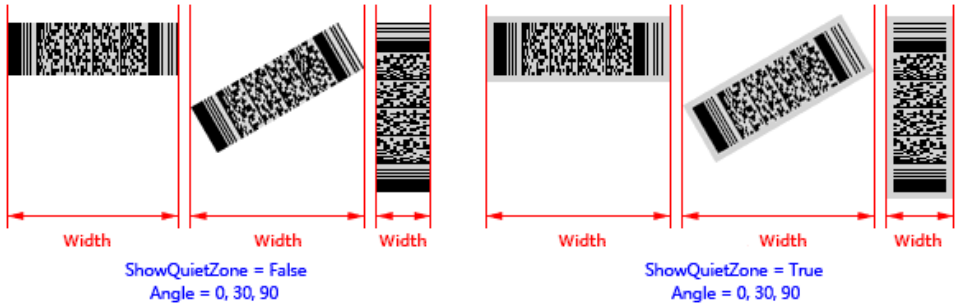
For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC. If its Linear property is set to the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8,

TBarcode1D_EAN13, or TBarcode1D_EAN128, and the linear component's DisplayText property isn't set to dtNone, Please use the method between the Printer.BeginDoc and Printer.EndDoc methods.

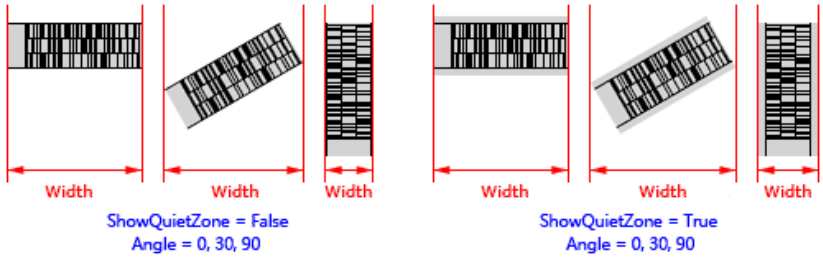
Parameters:

- **Width**: Double; Returns the horizontal width of the rotated barcode symbol in millimeters. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

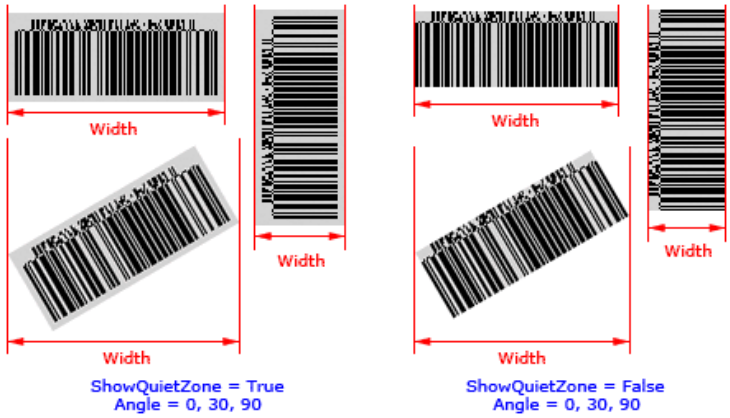
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

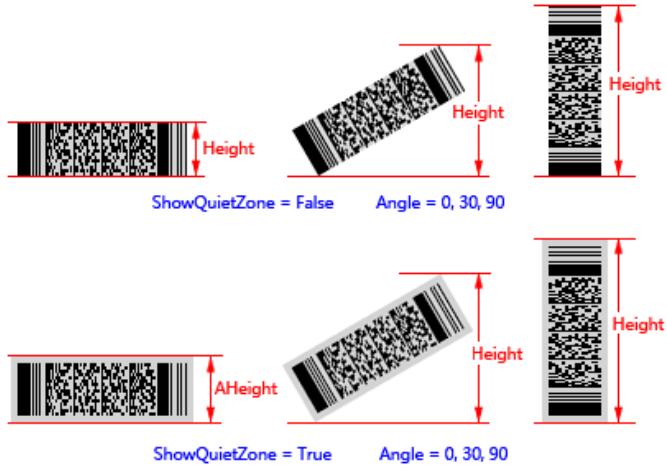


  When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8,

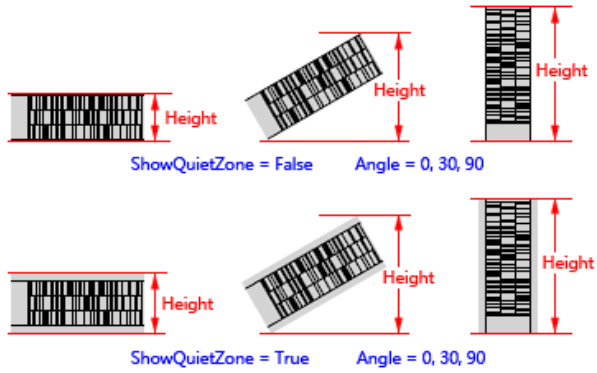
TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

- **Height**: Double; Returns the vertical height of the rotated barcode symbol in millimeters. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

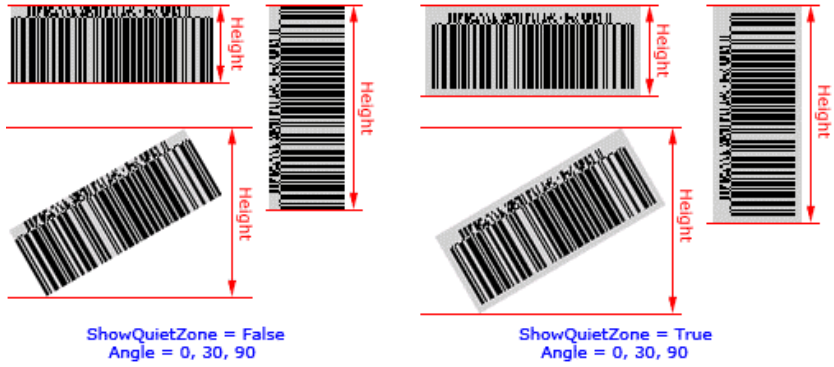
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

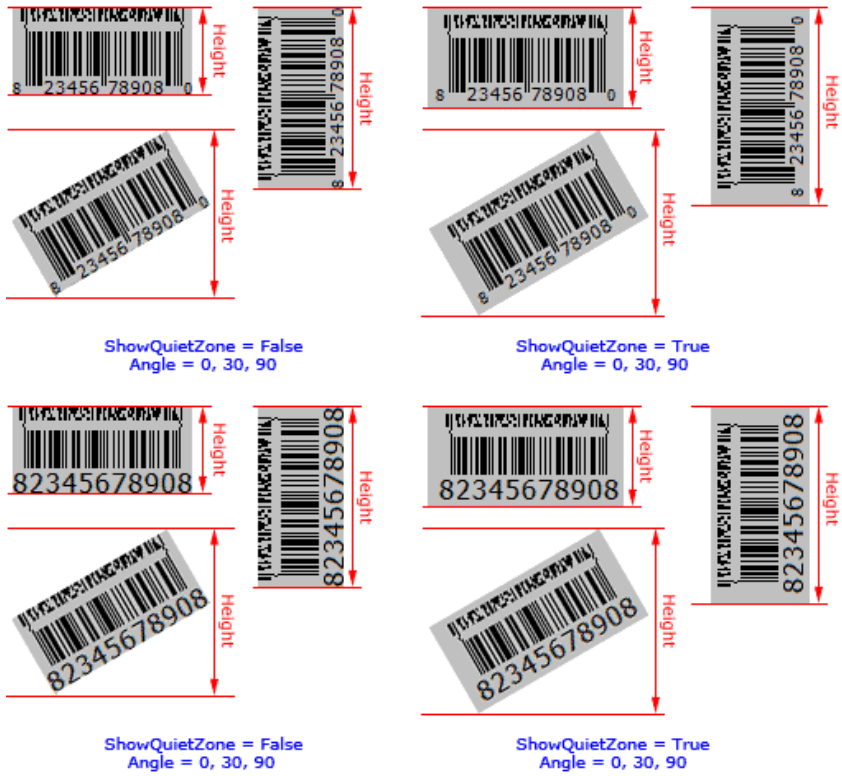


For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



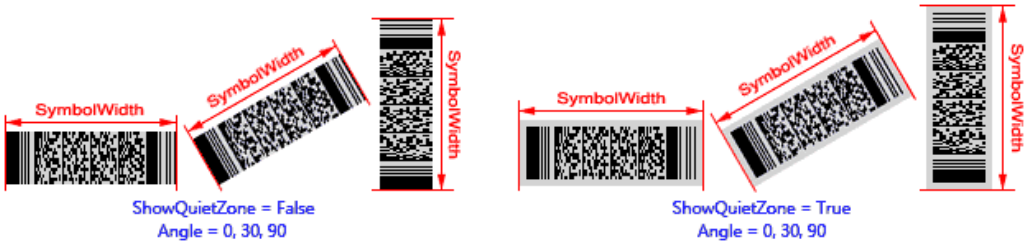
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the

excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

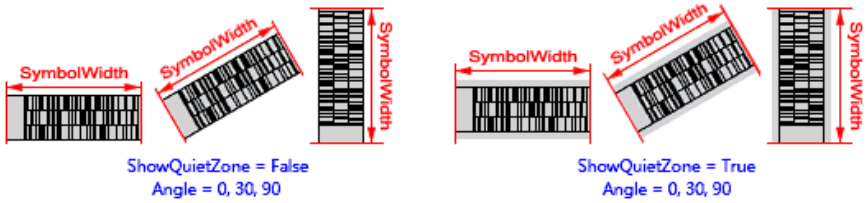


- **SymbolWidth**: Double; Returns the distance between the leading and trailing of the rotated barcode symbol in millimeter. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

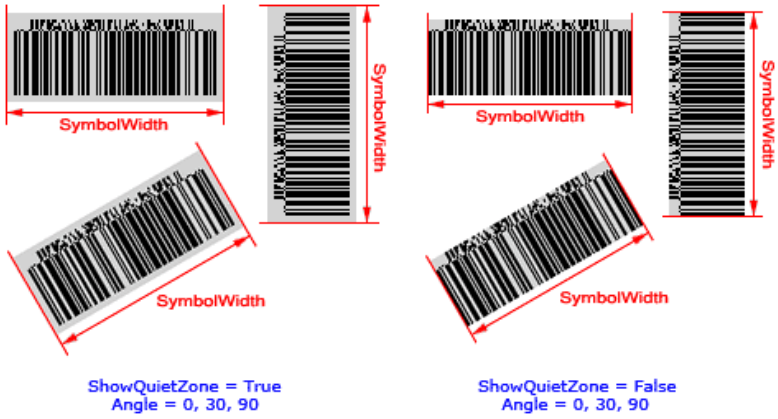
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



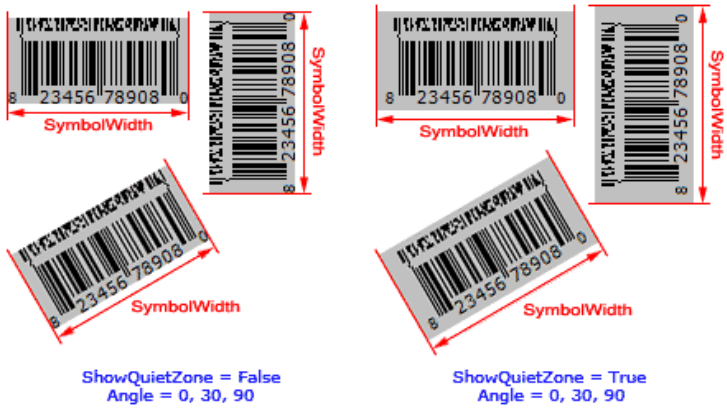
  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

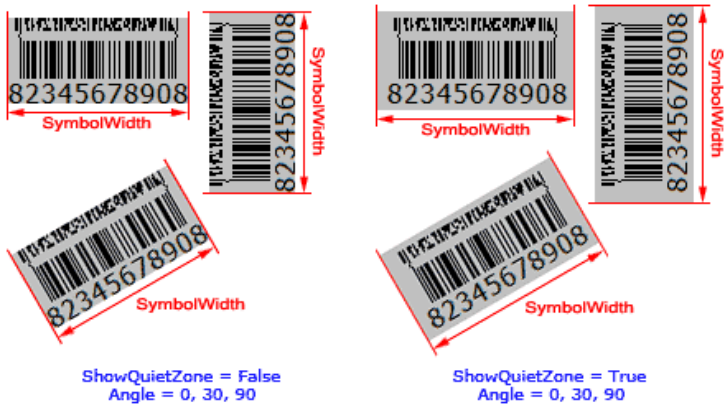


For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire rotated EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

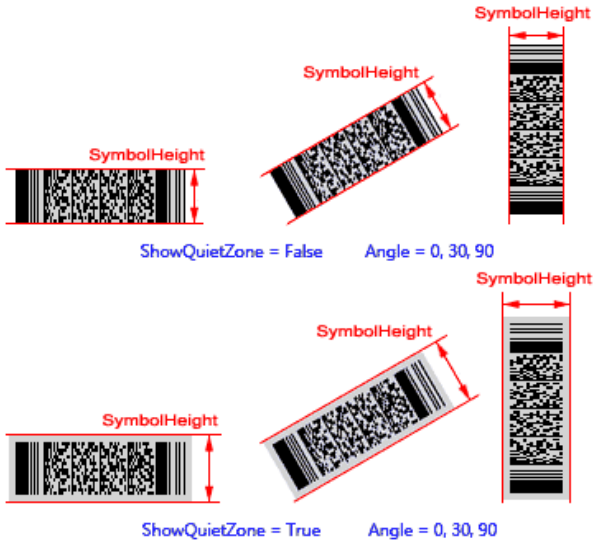




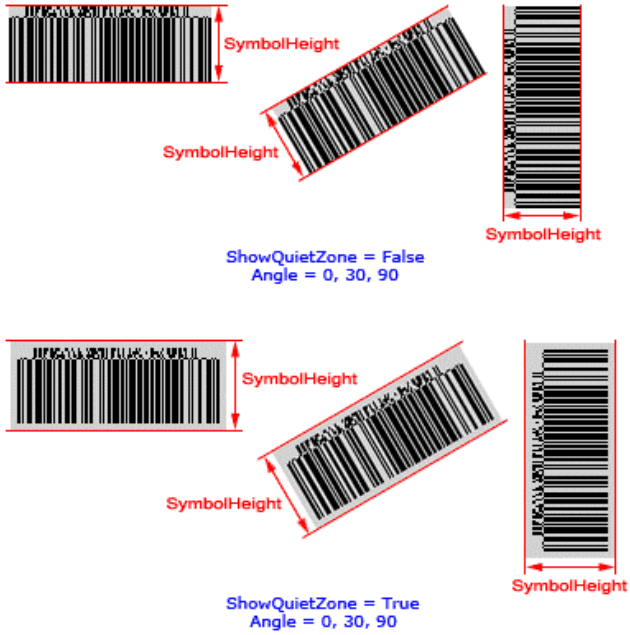
See also the "BarcodeWidth" property.

- **SymbolHeight**: Double; Returns the distance between the top and bottom of the rotated barcode symbol in millimeter. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

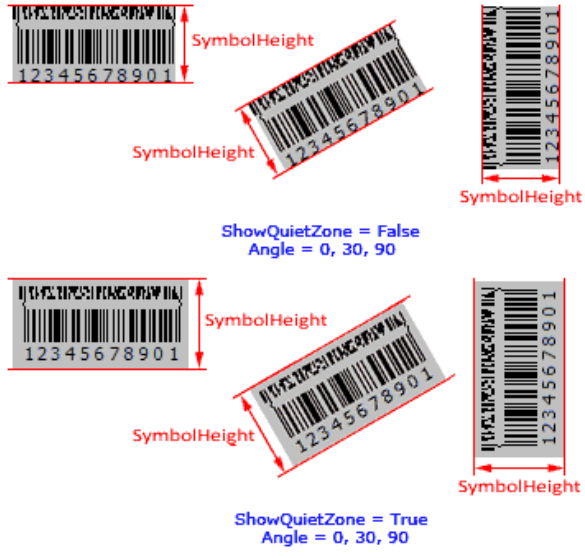
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

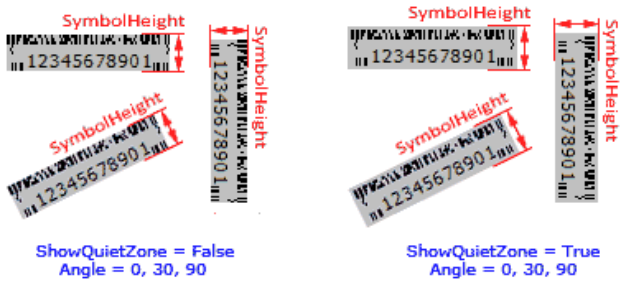


  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire rotated EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented, the human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):





See also the "BarcodeHeight" property.

- **Barcode**: String; Specifies the barcode text. It is of type string.

  For Delphi/C++ Builder 2007 or early, the Barcode parameter is in fact an AnsiString. By default, it is an ANSI encoding string, if you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or specify the converted string in the Barcode parameter. Also you can use the method if you encode a block of binary (bytes) data.

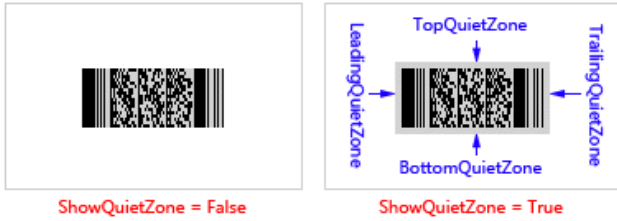
  For Delphi/C++ Builder 2009 or later, it is in fact an UnicodeString instead of AnsiString. By default, the unicode string will be converted to an ANSI encoding string, then be encoded into the barcode symbol. If you want to use other encoding scheme (for example the UTF-8, UTF-16), please convert it in the OnEncode event, or use the PrintSize (Syntax 3) overloading method and specify the converted string in its Data parameter. If you encode a block of binary (bytes) data, please use the PrintSize (Syntax 3) overloading method.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

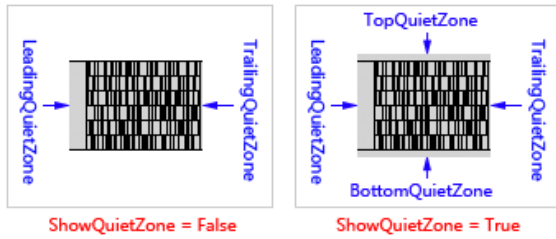
  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

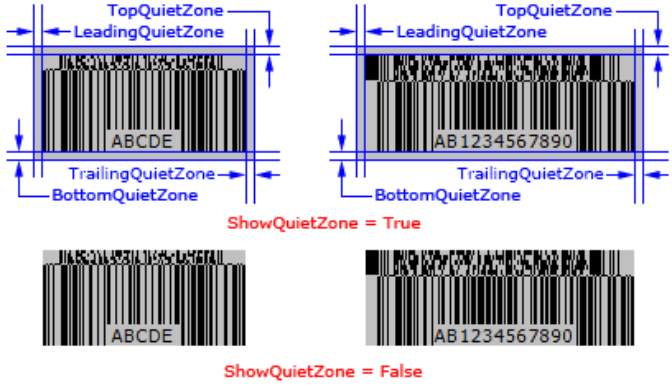
- **ShowQuietZone**: Boolean; Specifies whether to include the leading, trailing, top and bottom quiet zones in barcode symbol. If the parameter value is set to true, these quiet zones are included in the barcode symbol. Otherwise, they don't be included. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



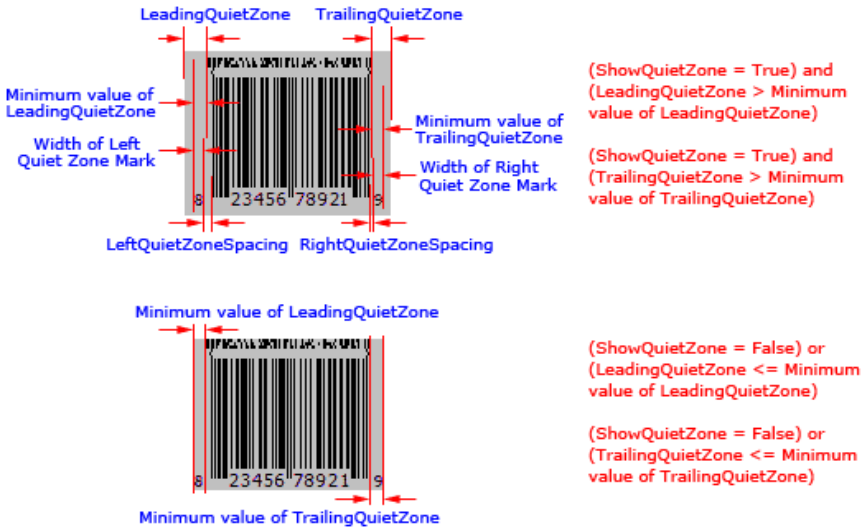
  For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be included always, even if the ShowQuietZone parameter value is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



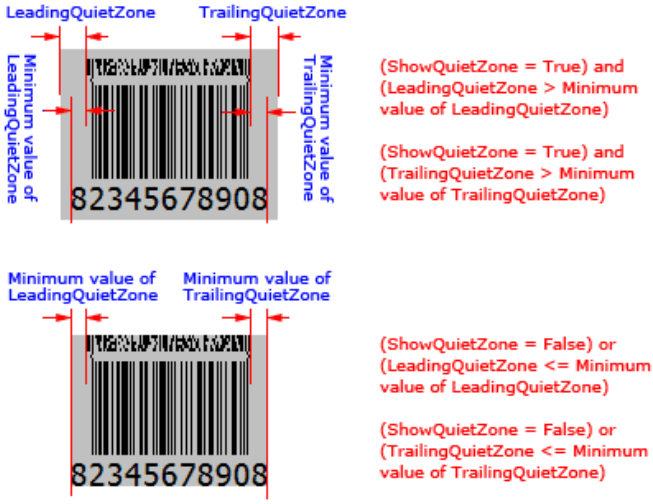
For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

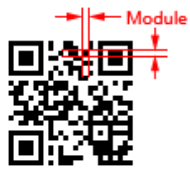


For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

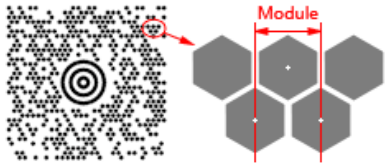


See also the "ShowQuietZone" property.

- **Module**: Double; Specifies the module size in millimeters.
  - For the Matrix 2D barcode symbology (excluding theTBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



- For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



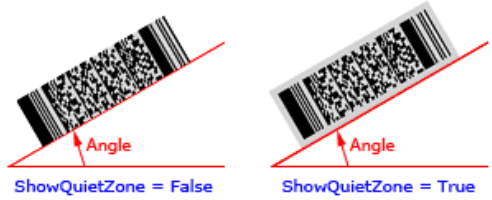
- For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

- For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

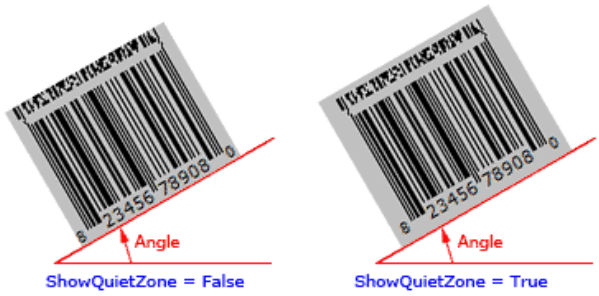
If any one of the BarcodeWidth parameter and BarcodeHeight parameter is provided and it isn't zero, the value of Module parameter will be ignored, the module size will be calculated based on the BarcodeWidth parameter value or the BarcodeHeight parmeter value. If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



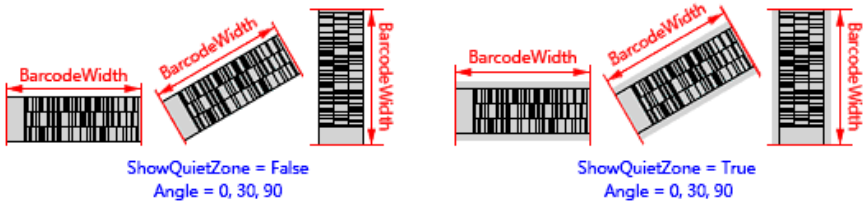
For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



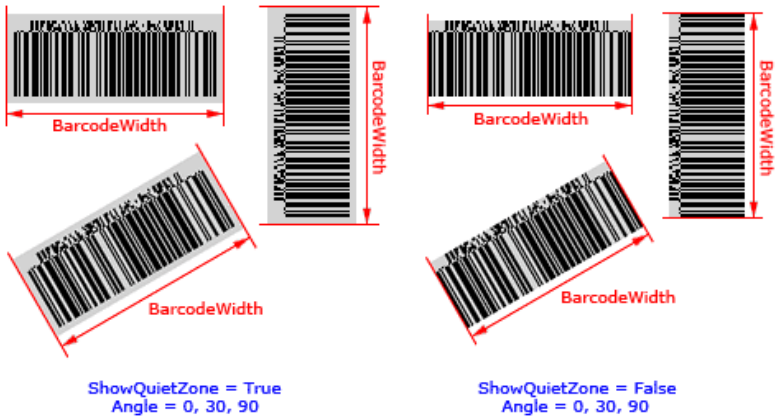
The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **BarcodeWidth**: Double, Specifies the barcode symbol width (the distance between the beginning and end of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the leading quiet zone and the trailing quiet zone are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the width for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



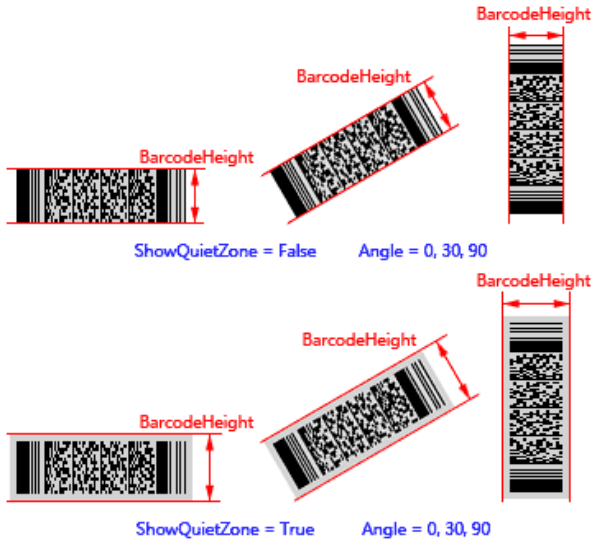
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess isn't included in the width value. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

Note, if the parameter is provided and it isn't zero, the values in Module and BarcodeHeight parameters will be ignored, the module size will be calculated based on the BarcodeWidth value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol width will be determined by the Module parameter value.

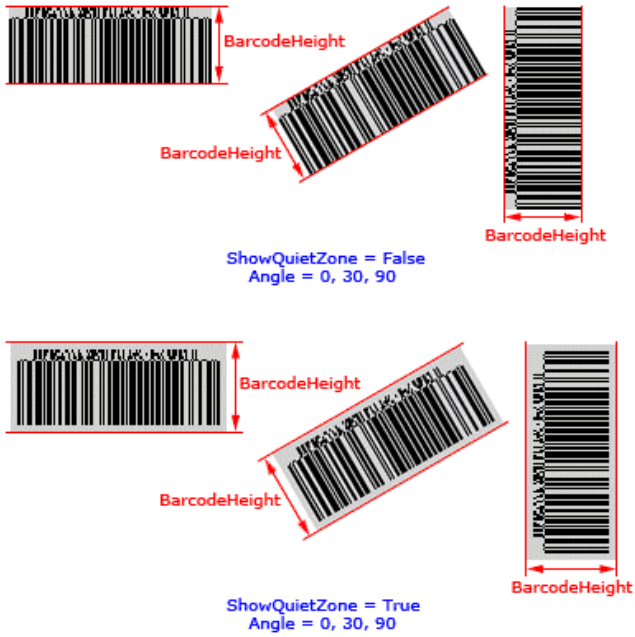
If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeWidth" property.

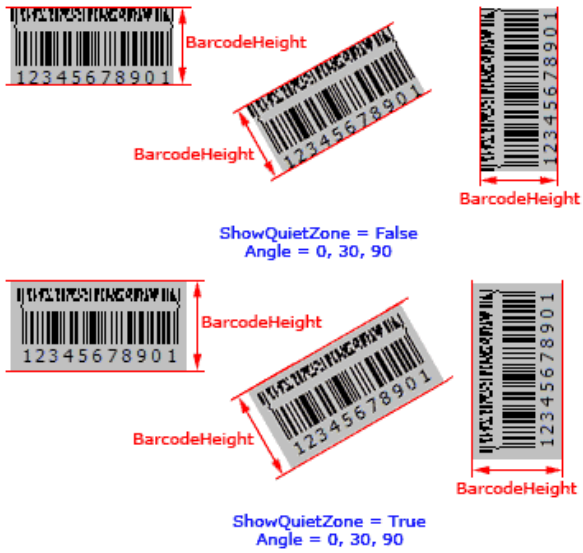
- **BarcodeHeight**: Double, Specifies the barcode symbol height (the distance between the top and bottom of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the top quiet zone and the bottom quiet zone are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the height for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. the human readable text and the vertical spacing between the text and linear symbol are included in the height value. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



Note, if the parameter is provided and it isn't zero, and the BarcodeWidth parameter isn't provided or it's zero, the value in Module parameter will be ignored, the module size will be calculated based on the BarcodeHeight value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol height will be determined by the Module parameter value.

If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeHeight" property.

- **HDPI**: Integer, Specifies the physical horizontal resolution of printer in DPI (dots per inch).

  It defaults to 0 if the HDPI is not provided, and the horizontal resolution will be obtained from your printer.

- **VDPI**: Integer, Specifies the physical vertical resolution of printer in DPI (dots per inch).

  It defaults to 0 if the VDPI is not provided, and the vertical resolution will be obtained from your printer.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

B.1.13.3 PrintSize - Syntax 3

Returns the actual horizontal width and vertical height of a rotated barcode symbol in millimeters. The barcode symbol is specified by the parameters of this method.

Syntax:

```
function PrintSize(var Width, Height, SymbolWidth, SymbolHeight: Double; Data:
  AnsiString; ShowQuietZone: Boolean; Module: Double; Angle: Integer = 0; BarcodeWidth:
  Double = 0; BarcodeHeight: Double = 0; HDPI: Integer = 0; VDPI: Integer = 0): Integer;
  overload; virtual;
```

Description:

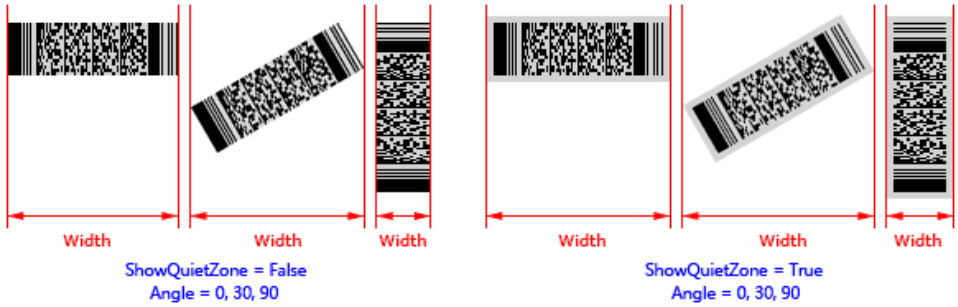
The method returns the actual horizontal width and vertical height of a rotated barcode symbol that is specified by parameters of this method, in millimeters.

For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC. If its Linear property is set to the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and the linear component's DisplayText property isn't set to dtNone, Please use the method between the Printer.BeginDoc and Printer.EndDoc methods.

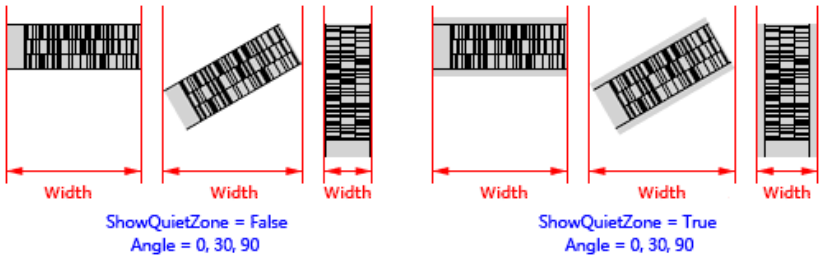
Parameters:

- **Width**: Double; Returns the horizontal width of the rotated barcode symbol in millimeters. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

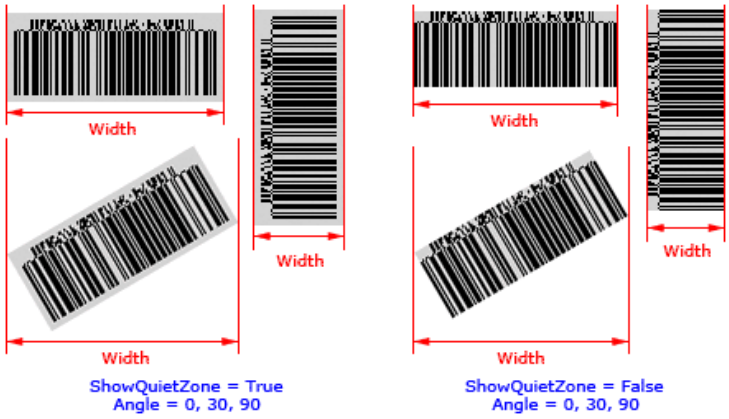
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



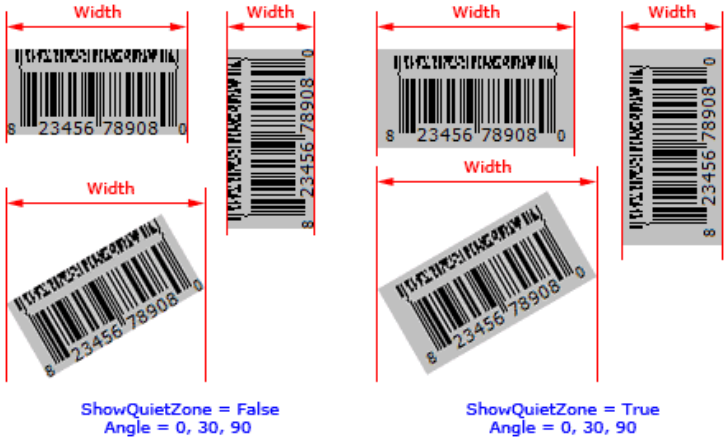
  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

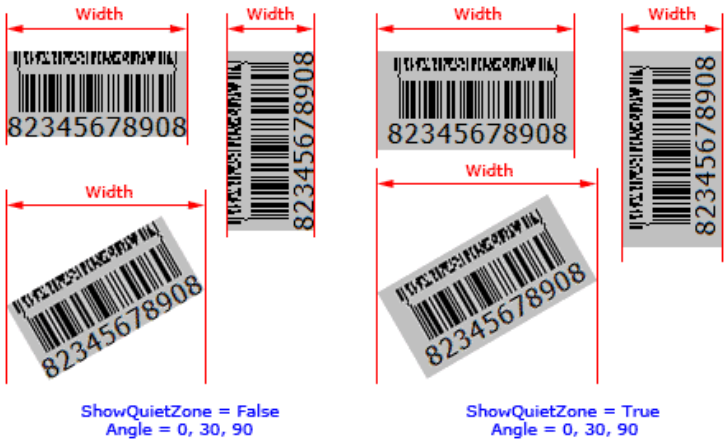


For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



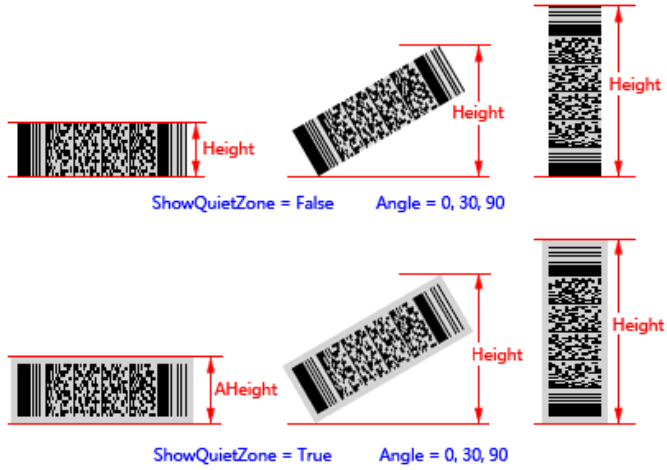
When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



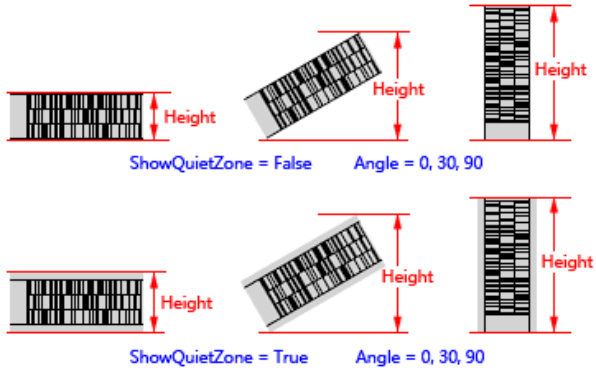


- **Height**: Double; Returns the vertical height of the rotated barcode symbol in millimeters. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

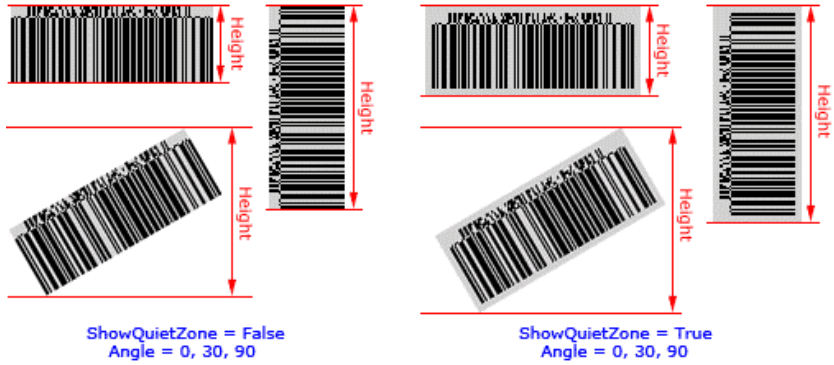


  For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

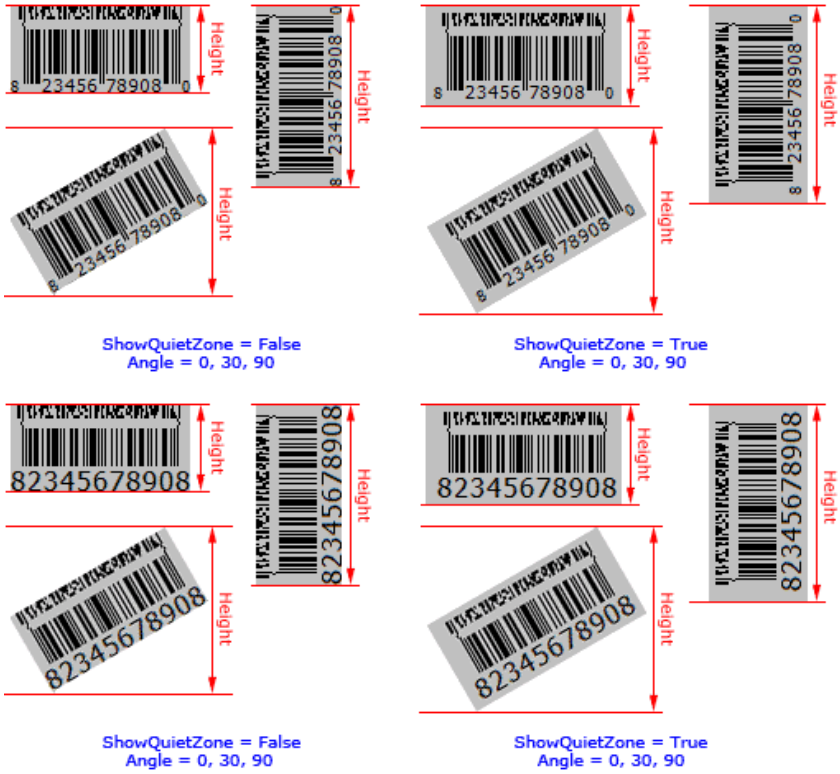


  For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D

symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

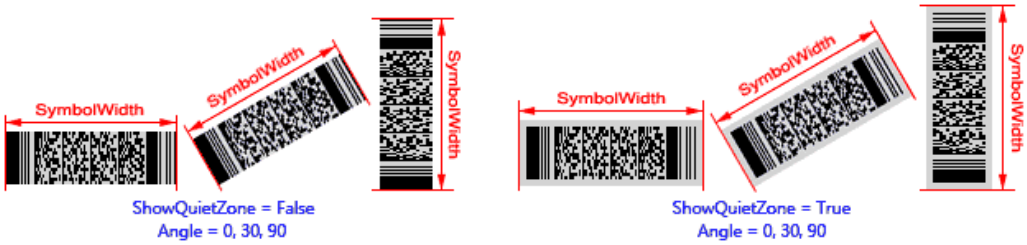


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

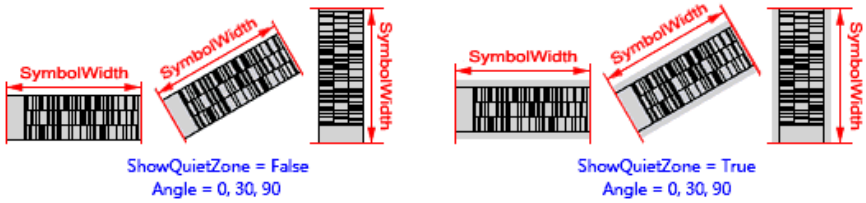


- **SymbolWidth**: Double; Returns the distance between the leading and trailing of the rotated barcode symbol in millimeter. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

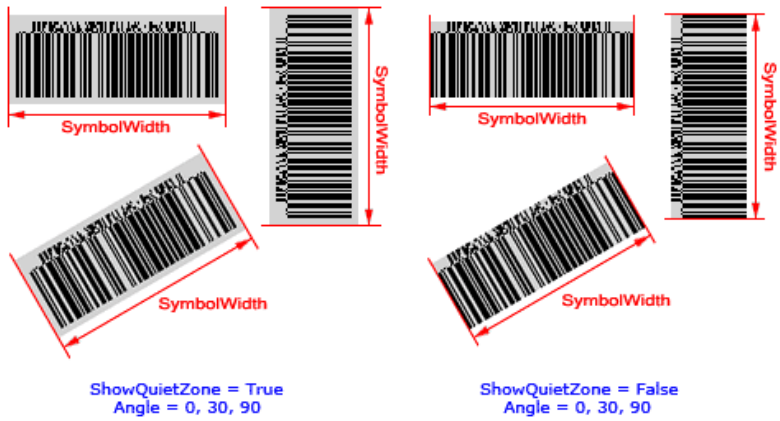
If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the width of entire rotated EAN.UCC composite barcode symbol, not the width of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

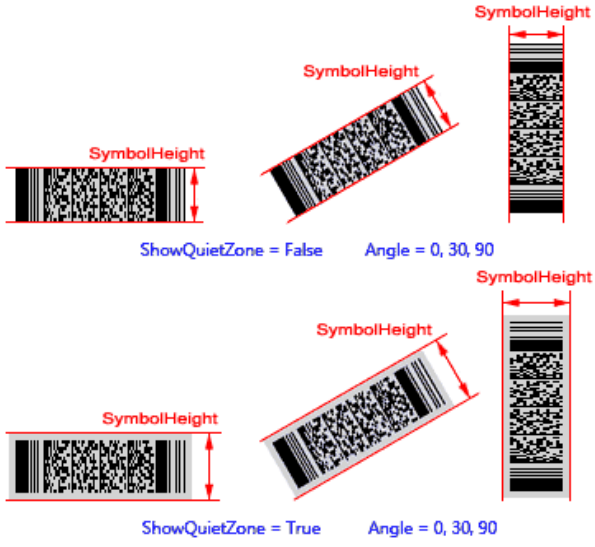


When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

See also the "BarcodeWidth" property.

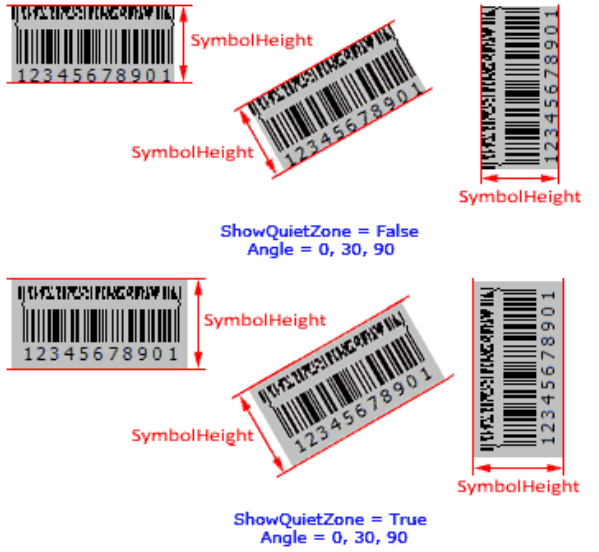
- **SymbolHeight**: Double; Returns the distance between the top and bottom of the rotated barcode symbol in millimeter. The value is calculated based on the module size. You can use one of the Module, BarcodeWidth and BarcodeHeight parameters to specify the module size.

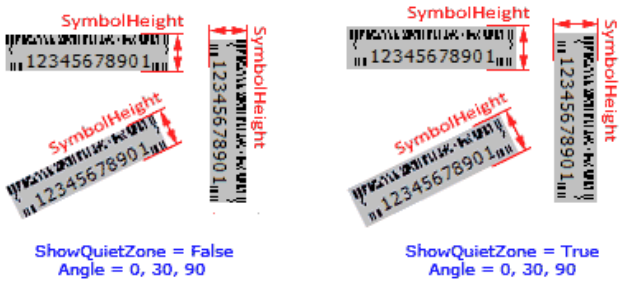
  If the quiet zones are represented (please read the ShowQuietZone parameter section below about whether or not the quiet zones will be represented), they are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter returns the height of entire rotated EAN.UCC composite barcode symbol, not the height of the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented, the human readable text and the vertical spacing between the text and linear symbol are included in the symbol. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):





See also the "BarcodeHeight" property.

- **Data**: AnsiString; Specifies the barcode text. It is of type AnsiString.

  For the TBarcode2D_RSS14 and TBarcode2D_RSSLimited components, if the property AutoCheckDigit is set to true, the check digit doesn't need to be included in the parameter, otherwise the check digit can be specified by you in the parameter.

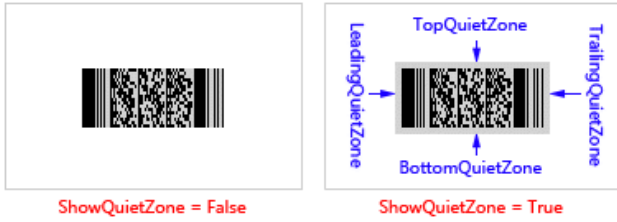
  For the Delphi/C++ Builder 2009 or later, you can use the method if you encode a block of binary (bytes) data into a barcode symbol. Or specifies the barcode text in AnsiString format.

  If you encode a block of binary (bytes) data into a barcode symbol under Delphi/C++ Builder 2007 or early, please use the PrintSize (Syntax 2) overloading method.

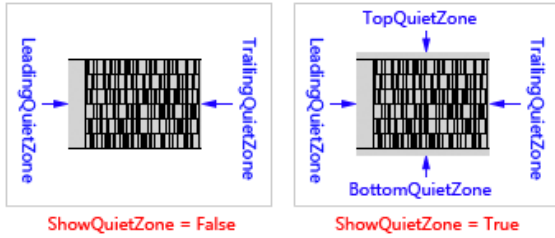
  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter only specifies the barcode text for the 2D symbol, the barcode text of linear symbol is specified by the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property of the linear component.

  See also the "Barcode" and "Data" (only for Delphi/C++ Builder 2009 or later) properties.

- **ShowQuietZone**: Boolean; Specifies whether to include the leading, trailing, top and bottom quiet zones in barcode symbol. If the parameter value is set to true, these quiet zones are included in the barcode symbol. Otherwise, they don't be included. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

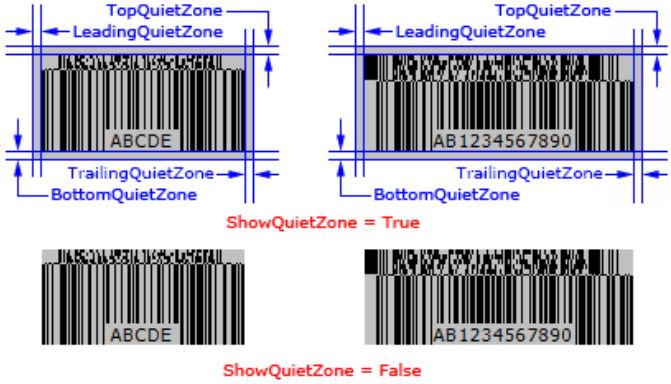


  For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone will be included always, even if the ShowQuietZone parameter value is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

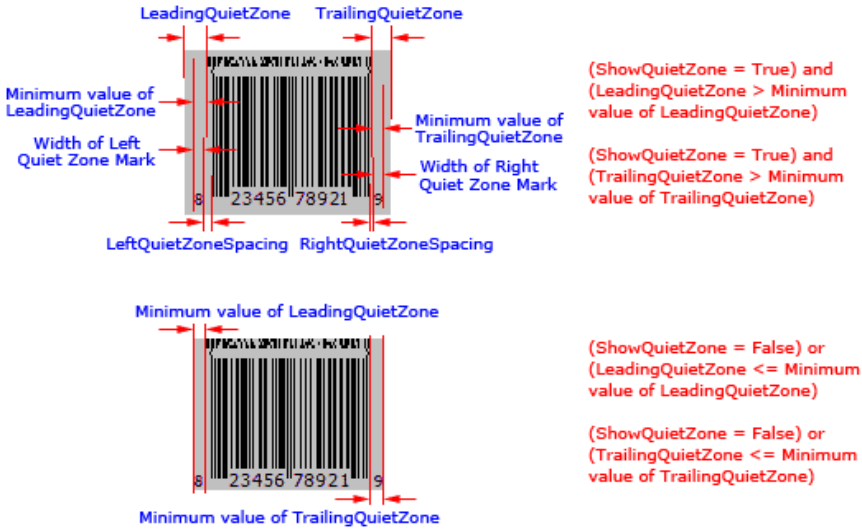


  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. And the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties specify the size of quiet zones for

entire EAN.UCC composite symbol, they are not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

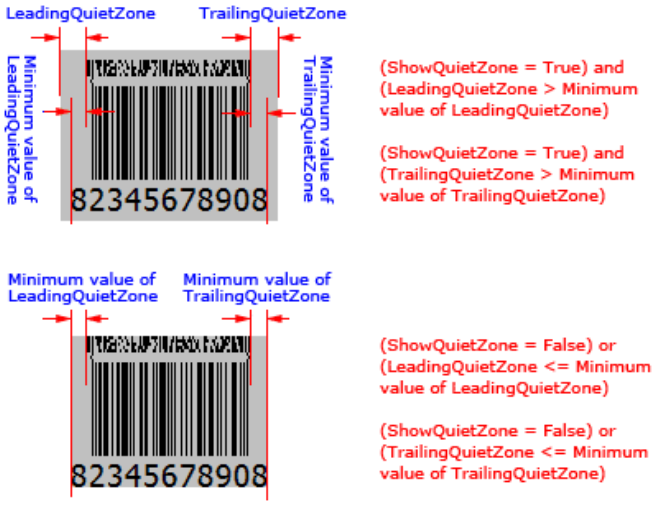


For the TBarcode2D_CCA and TBarcode2D_CCB barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, or TBarcode1D_EAN13, and its left quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 component, the ShowQuietZoneMark property is set to true too), the leading quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of leading quiet zone is the distance between the left side of the left quiet zone mark of the linear symbol and the first bar of the 2D symbol, and the minimum width of leading quiet zone will be used if the value of LeadingQuietZone property is less than it. Similarly, if its right quiet zone mark is represented (the TextAlignment property's value is set to taCustom, and for the TBarcode1D_ENA8 and TBarcode1D_EAN13 components, the ShowQuietZoneMark property is set to true too), the trailing quiet zone will be included even if the ShowQuietZone parameter is set to false, the minimum width of trailing quiet zone is the distance between the last bar of the 2D symbol and the right side of the right quiet zone mark of the linear symbol, and the minimum width of trailing quiet zone will be used if the value of TrailingQuietZone property is less than it. In other words, the leading and trailing quiet zoens will be automatically extanded to cover the left and right quiet zone marks, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



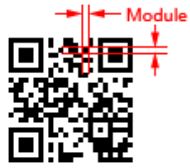
For all of the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the linear component is the TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_UPCA, TBarcode1D_ENA8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented and exceeds the beginning or end of entire EAN.UCC barcode symbol (the leading and trailing quiet zones are included if theShowQuietZone parameter is set to true), the leading and trailing quiet zoens will be automatically extanded to cover the human readable text, regardless of whether the ShowQuietZone parameter is set to true or false. See diagram (the quiet zones are drawn in clSilver color in order

to accentuate them):

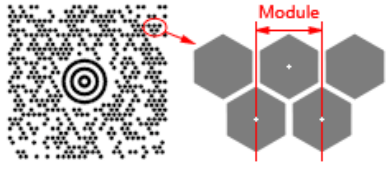


See also the "ShowQuietZone" property.

- **Module**: Double; Specifies the module size in millimeters.
  - For the Matrix 2D barcode symbology (excluding the TBarcode2D_MaxiCode barcode symbology), the module is single cell (a suqare shape) used to encode one bit data. The parameter specifies the module width and height. See diagram:



  - For the TBarcode2D_MaxiCode barcode symbology, the module is single cell (a regular hexagonal shape) used to encode one bit data. The parameter specifies the horizonal width of a module. Also, it's the center to center horizontal distance between adjacent modules. See diagram:



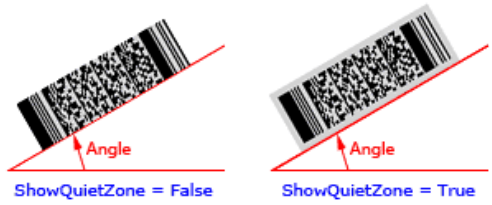
  - For the Stacked 2D barcode symbology and Linear 1D barcode symbology, the module is the thinest bar (or space) in the barcode symbol. The parameter specifies the module width. See diagram:

  - For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the parameter specifies the module width for entire EAN.UCC composite barcode symbol (including both 2D and linear symbols). See diagram:

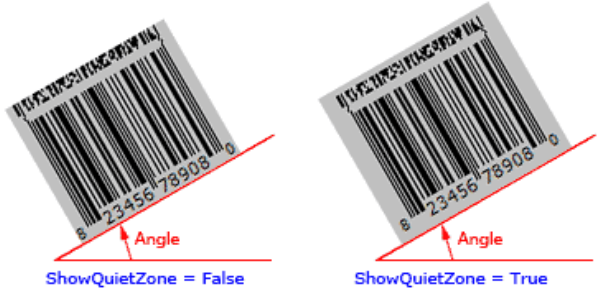
If any one of the BarcodeWidth parameter and BarcodeHeight parameter is provided and it isn't zero, the value of Module parameter will be ignored, the module size will be calculated based on the BarcodeWidth parameter value or the BarcodeHeight parmeter value. If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "Module" property.

- **Angle**: Integer; Specifies an angle in degrees to rotate the barcode symbol and its quiet zones (if they are represented, please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented) anticlockwise. You can use the LeadingQuietZone, TrailingQuietZone, TopQuietZone, and BottomQuietZone properties to specify the size of these quiet zones in modules. And use the ShowQuietZone parameter to specify whether to represent these quiet zones. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



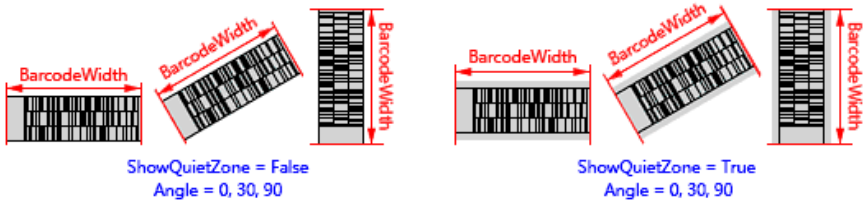
For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, and they will be rotated as an entirety. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



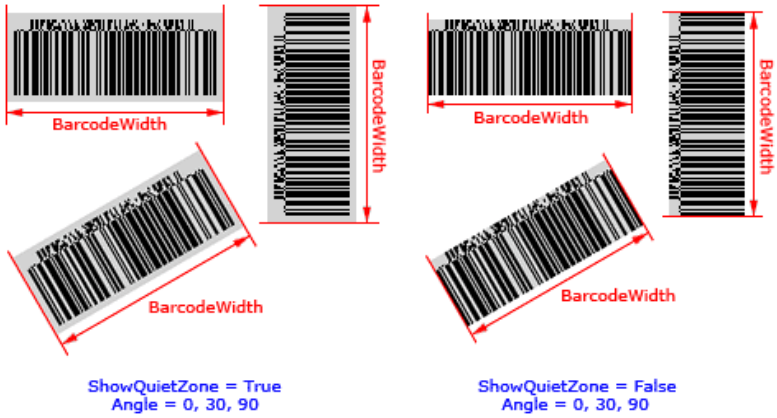
The parameter defaults to 0 if the Angle is not provided, meaning left to right horizontal direction.

- **BarcodeWidth**: Double, Specifies the barcode symbol width (the distance between the beginning and end of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the leading quiet zone and the trailing quiet zone are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

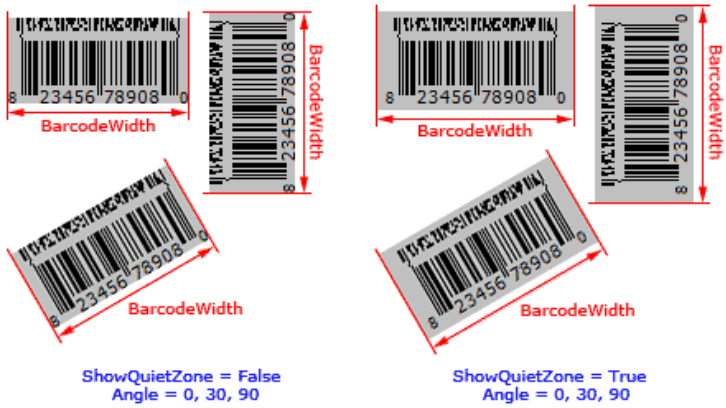
For the TBarcode2D_Code16K barcode component, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone parameter is set to false. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):

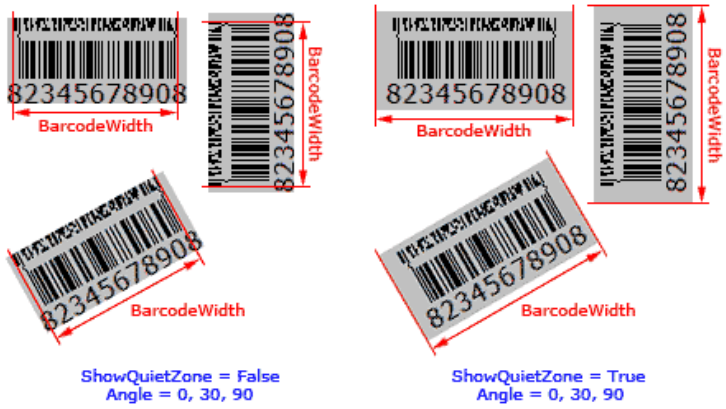


For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the width for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess isn't included in the width value. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



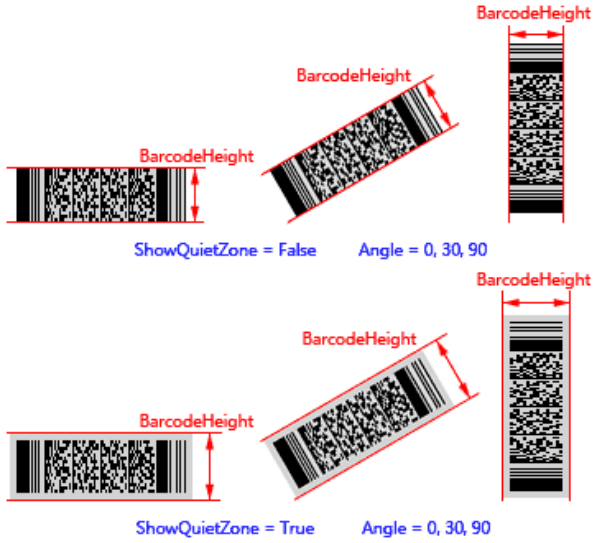


Note, if the parameter is provided and it isn't zero, the values in Module and BarcodeHeight parameters will be ignored, the module size will be calculated based on the BarcodeWidth value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol width will be determined by the Module parameter value.

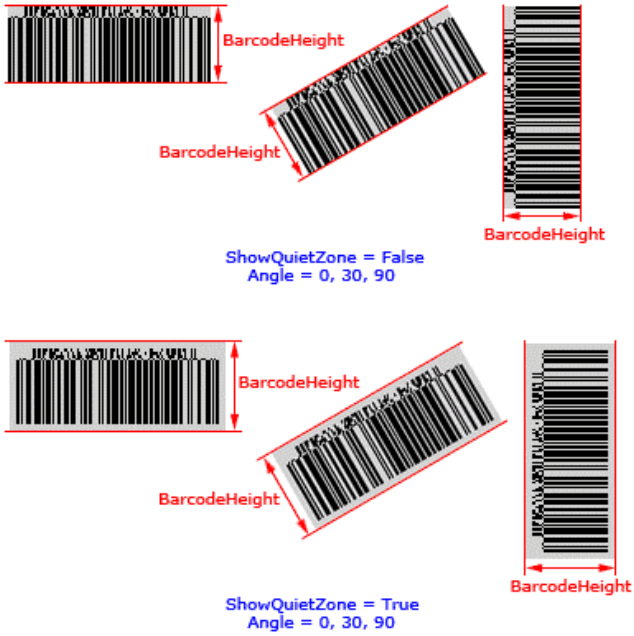
If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeWidth" property.

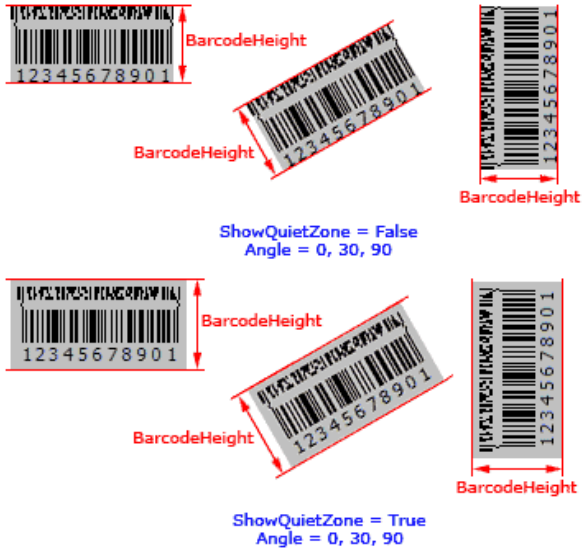
- **BarcodeHeight**: Double, Specifies the barcode symbol height (the distance between the top and bottom of the barcode symbol), in millimeters. If the quiet zones are represented (please read the ShowQuietZone parameter section above about whether or not the quiet zones will be represented), the top quiet zone and the bottom quiet zone are included in the barcode symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol, the linear symbol is included in the composite barcode symbol, and the parameter specifies the height for entire EAN.UCC composite barcode symbol not only for the 2D symbol. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. the human readable text and the vertical spacing between the text and linear symbol are included in the height value. See diagram (the quiet zones are drawn in clSilver color in order to accentuate them):



Note, if the parameter is provided and it isn't zero, and the BarcodeWidth parameter isn't provided or it's zero, the value in Module parameter will be ignored, the module size will be calculated based on the BarcodeHeight value. If both BarcodeHeight and BarcodeWidth parameters aren't provided or they are all zero, the barcode symbol height will be determined by the Module parameter value.

If both BarcodeWidth and BarcodeHeight parameters are provided and aren't zero, only the BarcodeWidth parameter value will be used to calculate the module size, the BarcodeHeight parameter value will be ignored.

See also the "BarcodeHeight" property.

- **HDPI**: Integer, Specifies the physical horizontal resolution of printer in DPI (dots per inch).

  It defaults to 0 if the HDPI is not provided, and the horizontal resolution will be obtained from your printer.

- **VDPI**: Integer, Specifies the physical vertical resolution of printer in DPI (dots per inch).

  It defaults to 0 if the VDPI is not provided, and the vertical resolution will be obtained from your printer.

Return:

- If the method succeeds, the return value is zero.
- If the length of barcode text is invalid, the return valus is -1 or -65537.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the length of barcode text of the 2D symbol is invalid, the return valus is -1. If the length of barcode text of the linear symbol is invalid, the return valus is -65537.

  For other barcode components, if the length of barcode text is invalid, the return valus is -1.

- If there is any invalid character in the the barcode text, the return value is greater than zero. In general, it's the position index of first invalid character, the index 1 denotes that the first character is invalid character.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. If the invalid character is in the 2D symbol, the 17th bit of the return value is set to 0 (the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character). If the invalid character is in the linear symbol, the 17th bit of the return value is set to 1 (the difference between the return value and 65536 is the position index of first invalid character. For example, the return value 65537 denotes that the first character is invalid character).

  For other barcode components, if there is any invalid character in the the barcode text, the return value is the position index of first invalid character, the index 1 denotes that the first character is invalid character.

Note:

The overloading method is available only for the Delphi/C++ Builder 2009 or later.

B.1.14 Size

Returns the height and width of specified barcode symbol, in pixels, that's represented in the TImage, TQRImage, or TQRGzImage control specified by the Image property.

Syntax:

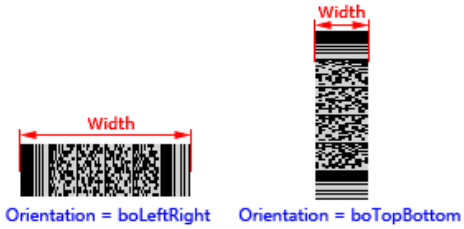
```
function Size(var Width, Height, SymbolWidth, SymbolHeight: Integer): Boolean; virtual;
```

Description:

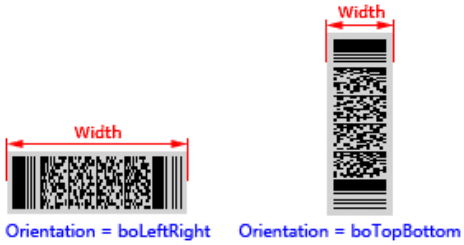
The method returns the width and height of the barcode symbol that's represented in the TImage, TQRImage, or TQRGzImage control specified by the Image property, in pixels. If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included in the barcode symbol.

Parameters:

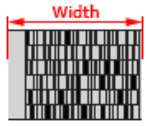
- **Width**: Integer; Returns the horizontal width of the rotated barcode symbol that's displyed in the TImage, TQRImage, or TQRGzImage control specified in the Image property, in pixels. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter returns the width of entire EAN.UCC composite barcode symbol, namely, the linear symbol is included in the barcode symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the barcode symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the barcode symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

The value of Width is calculated using following method:

- When the Stretch property is set to false, it's calculated based on the Module property value.
- When the Stretch property is set to true:
  - When the Orientation property is set to "boLeftRight" or "boRightLeft":

    If the BarcodeWidth property value is greater than zero, it's same to the BarcodeWidth property value.

    If the BarcodeWidth property value is equal to zero, the LeftMargin property value will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

    If the BarcodeWidth property value is less than zero, the LeftMargin property value, and the absolute value of the negative BarcodeWidth property value (i.e. the right margin) will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

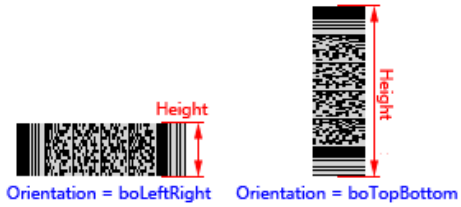
  - When the Orientation property is set to "boTopBottom" or "boBottomTop":

    If the BarcodeHeight property value is greater than zero, it's same to the BarcodeHieght property value.

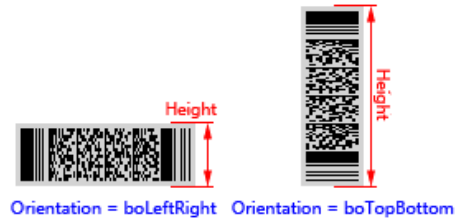
    If the BarcodeHeight property value is equal to zero, the LeftMargin property value will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

    If the BarcodeHeight property value is less than zero, the LeftMargin property value, and the absolute value of the negative BarcodeHeight property value (i.e. the right margin) will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

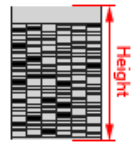
- **Height**: Integer; Returns the vertical height of the rotated barcode symbol that's displyed in the TImage, TQRImage, or TQRGzImage control specified in the Image property, in pixels. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



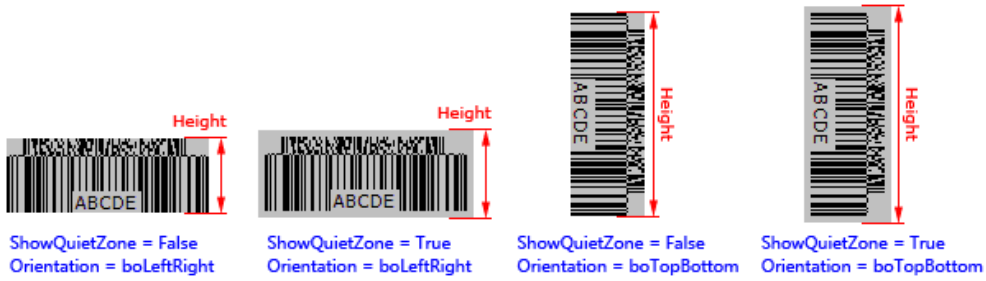
If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), they are included. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the TBarcode2D_Code16K barcode components, if the Orientation property is set to "boTopBottom" or "boBottomTop", the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC components, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter returns the height of entire EAN.UCC composite barcode symbol, namely, the linear symbol is included in the barcode symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the barcode symbol. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the barcode symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

The value of Height is calculated using following method:

- When the Stretch property is set to false, it's calculated based on the Module property value.
- When the Stretch property is set to true:
  - When the Orientation property is set to "boLeftRight" or "boRightLeft":

    If the BarcodeHeight property value is greater than zero, it's same to the BarcodeHeight property value.

    If the BarcodeHeight property value is equal to zero, the TopMargin property value will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

    If the BarcodeHeight property value is less than zero, the TopMargin property value, and the absolute value of the negative BarcodeHeight property value (i.e. the bottom margin) will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

  - When the Orientation property is set to "boTopBottom" or "boBottomTop":

    If the BarcodeWidth property value is greater than zero, it's same to the BarcodeWidth property value.

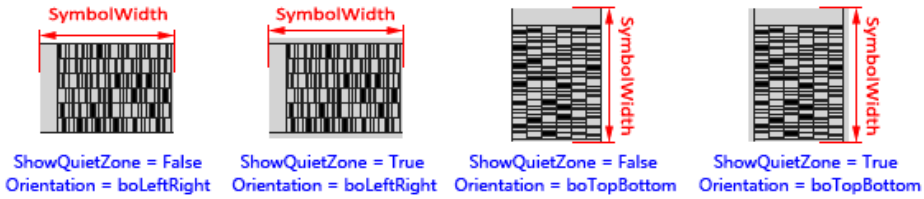
    If the BarcodeWidth property value is equal to zero, the TopMargin property value will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

    If the BarcodeWidth property value is less than zero, the TopMargin property value, and the absolute value of the negative BarcodeWidth property value (i.e. the bottom margin) will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

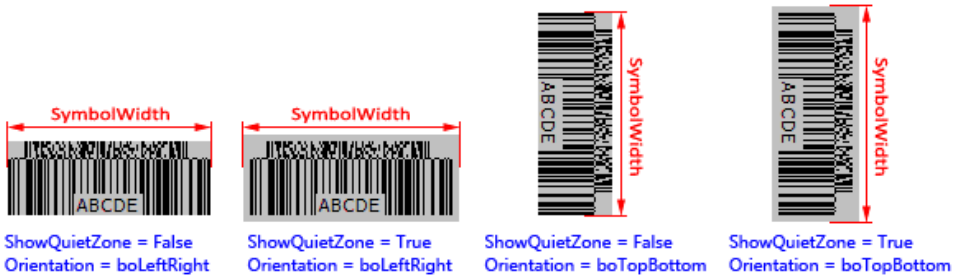
- **SymbolWidth**: Integer; Returns the distance between the leading and trailing of the rotated barcode symbol that's displyed in the TImage, TQRImage, or TQRGzImage control specified in the Image property, in pixels. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), the leading quiet zone and the trailing quiet zone are included. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):

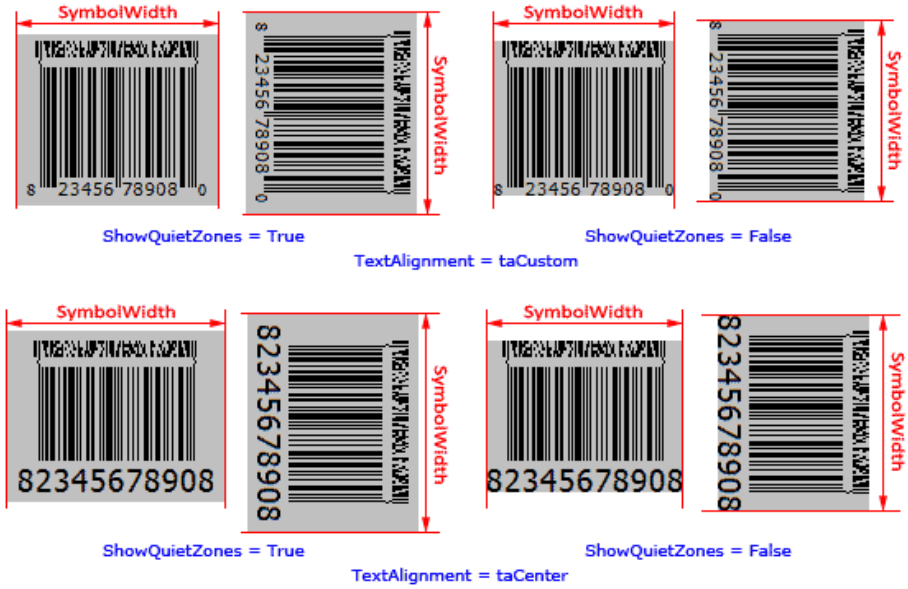
For the TBarcode2D_Code16K barcode components, the leading quiet zone and the trailing quiet zone are included always, even if the ShowQuietZone property is set to false. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if its Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter returns the width of entire EAN.UCC composite barcode symbol, namely, the linear symbol is included in the barcode symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. If the TextAlignment property isn't set to taCustom, and the human readable text exceeds entire EAN.UCC composite barcode symbol in horizontal direction, the excess is included in the barcode symbol. If the TextAlignment property is set to taCustom, the width of quiet zone marks and their horizontal spacing (LeftQuietZoneSpacing and RightQuietZoneSpacing) are included too (if they are represented, this does not apply to TBarcode1D_EAN128). See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



The value of SymbolWidth is calculated using following method:

- When the Stretch property is set to false, it's calculated based on the Module property value.
- When the Stretch property is set to true:
  - If the BarcodeWidth property value is greater than zero, it's same to the BarcodeWidth property value.
  - If the BarcodeWidth property value is equal to zero:

    When the Orientation property is set to "boLeftRight" or "boRightLeft", the LeftMargin property value will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

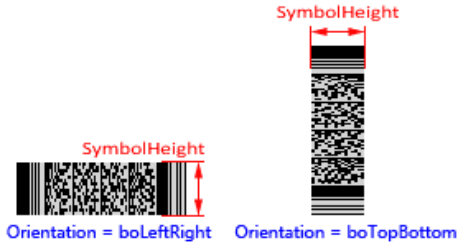
    When the Orientation property is set to "boTopBottom" or "boBottomTop", the TopMargin property value will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.
  - If the BarcodeWidth property value is less than zero:

    When the Orientation property is set to "boLeftRight" or "boRightLeft", the LeftMargin property value, and the absolute value of the negative BarcodeWidth property value (i.e. the right margin) will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

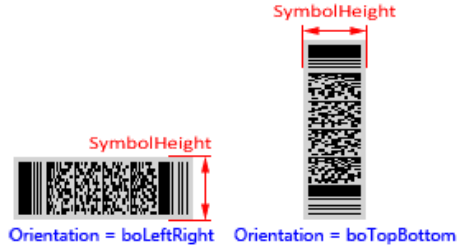
    When the Orientation property is set to "boTopBottom" or "boBottomTop", the TopMargin property value, and the absolute value of the negative BarcodeWidth property value (i.e. the bottom margin) will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

See also the "BarcodeWidth" property.

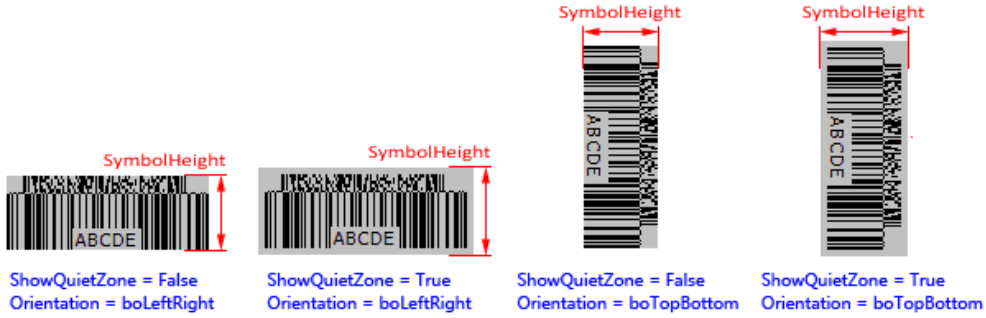
- **SymbolHeight**: Integer; Returns the distance between the top and bottom of the rotated barcode symbol that's displyed in the TImage, TQRImage, or TQRGzImage control specified in the Image property, in pixels. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



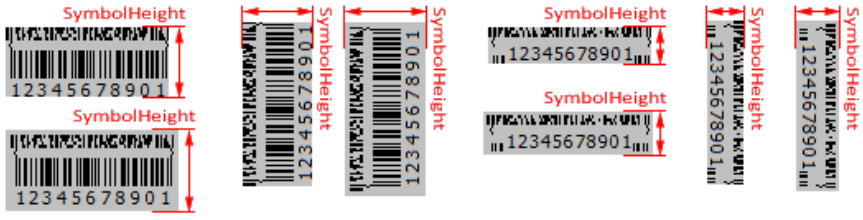
If the quiet zones are represented (please read the ShowQuietZone property about whether or not the quiet zones will be represented), the top quiet zone and the bottom quiet zone are included. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



For the 2D component TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC, if it's Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. The parameter returns the height of entire EAN.UCC composite barcode symbol, namely, the linear symbol is included in the barcode symbol. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



When the linear component of TBarcode2D_CCA, TBarcode2D_CCB, or TBarcode2D_CCC component is TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, or TBarcode1D_EAN128, and its human readable text is represented. The human readable text and the vertical spacing between the text and linear symbol are included in the barcode symbol. If the human readable text exceeds the linear barcode symbol in vertical direction, the excess is included too. See diagram (the SpaceColor property value is set to clSilver in order to accentuate the quiet zones):



The value of SymbolHeight is calculated using following method:

- When the Stretch property is set to false, it's calculated based on the Module property value.
- When the Stretch property is set to true:

- If the BarcodeHeight property value is greater than zero, it's same to the BarcodeHeight property value.
- If the BarcodeHeight property value is equal to zero:

  When the Orientation property is set to "boLeftRight" or "boRightLeft", the TopMargin property value will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

  When the Orientation property is set to "boTopBottom" or "boBottomTop", the LeftMargin property value will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.
- If the BarcodeHeight property value is less than zero:

  When the Orientation property is set to "boLeftRight" or "boRightLeft", the TopMargin property value, and the absolute value of the negative BarcodeHeight property value (i.e. the bottom margin) will be subtracted from the height of TImage, TQRImage, or TQRGzImage control that's specified by Image property, then the result will be returned.

  When the Orientation property is set to "boTopBottom" or "boBottomTop", the LeftMargin property value, and the absolute value of the negative BarcodeHeight property value (i.e. the right margin) will be subtracted from the width of TImage, TQRImage, or TQRGzImage control that's specified by the Image property, then the result will be returned.

See also the "BarcodeHeight" property.

Return:

- If the method succeeds, the return value is true.
- For the Delphi/C++ Builder 2007 or early, if the length ofBarcode property value is invalid, or there is any invalid character in the Barcode property value, the return value is false. Corresponding to the OnInvalidLength event or the OnInvalidChar event will occur.

  For Delphi/C++ Builder 2009 or later, if you use theBarcode property to specify the barcode text, when the length of Barcode property value is invalid, or there is any invalid character in the Barcode property value, the return value is false. Corresponding to the OnInvalidLength event or the OnInvalidChar event will occur. If you use the Data property to specify the barcode text, when the length of Data property value is invalid, or there is any invalid character in the Data property value, the return value is false. Corresponding to the OnInvalidDataLength event or the OnInvalidDataChar event will occur.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components, if the Linear property is set, and the length of its Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value is invalid, the return value is false too, corresponding to the OnInvalidLength or OnInvalidDataLength (only for Delphi/C++ Builder 2009 or later) event will occur. If there is any invalid character in the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value, the return value is false too, corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event will occur.

Note:

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D components, if the Linear property is set to one of TBarcode1D_UPCA, TBarcode1D_UPCE, TBarcode1D_UPCE0, TBarcode1D_UPCE1, TBarcode1D_EAN8, TBarcode1D_EAN13, and TBarcode1D_EAN128 components, and the Image property is not set, the human readable text of linear component will not be consulted.

B.2 TLogo2D

B.2.1 Assign

Copies a TLogo2D component from another TLogo2D component.

Syntax:

```
procedure Assign(Source: TPresistent); override;
```

Description:

If the Source parameter is an object created from a TSave2D component class, Assign copies all property values and event handles. If Source is any other type of object, an ELogo2DError exception occurs.

Parameters:

- **Source**: TPersistent; Specifies the source object.

B.2.2 Create

Creates and initializes a TLogo2D component.

Syntax:

```
constructor Create(Owner: TComponent); override;
```

Description:

Call Create to instantiate a TLogo2D object at runtime. The TLogo2D components added at design time are created automatically.

Parameters:

- **Owner**: TComponent; It is the component that is responsible for freeing the TLogo2D component instance. Typically, this is the form. It becomes the value of the Owner property.

B.2.3 Destroy

Disposes of the instance of the TLogo2D object.

Syntax:

```
destructor Destroy; override;
```

Description:

Destroy is the destructor for a TLogo2D object.

Do not call the destructor directly in an application. Instead, call Free. The Free verifies that the image saving object is not nil before it calls Destroy.

B.3 TSave2D

B.3.1 Assign

Copies an image saving component from another image saving component.

Syntax:

```
procedure Assign(Source: TPresistent); override;
```

Description:

If the Source parameter is an object created from a subclass of TSave2D component class, and the class is same to current image saving component class, Assign copies all property values and event handles. If Source is any other type of object, an ESave2DError exception occurs.

Parameters:

- **Source**: TPersistent; Specifies the source object.

B.3.2 Create

Creates and initializes an image saving component.

Syntax:

```
constructor Create(Owner: TComponent); override;
```

Description:

Call Create to instantiate an image saving object at runtime. Image saving components added at design time are created automatically.

Parameters:

- **Owner**: TComponent; It is the component that is responsible for freeing the image saving component instance. Typically, this is the form. It becomes the value of the Owner property.

B.3.3 Destroy

Disposes of the instance of the image saving object.

Syntax:

```
destructor Destroy; override;
```

Description:

Destroy is the destructor for an image saving object.

Do not call the destructor directly in an application. Instead, call Free. The Free verifies that the image saving object is not nil before it calls Destroy.

B.3.4 Save

Save current barcode symbol in the barcode component to an image file. The barcode component is specified by the Barcode2D property.

Syntax:

```
function Save(FilePath: string): Integer; virtual;
```

Description:

The method saves the barcode symbol to the image file. The barcode symbol is defined in the barcode component that's specified by the Barcode2D property. Also, the TImage component isn't required in the Image property of the barcode component.

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC barcode components, if the Linear property is set, the 2D symbol and the linear symbol are assembled to generate an EAN.UCC composite barcode symbol. Only the 2D portion will be saved in the image file now.

An OnSave event will occur before save the barcode symbol to picture file. And an OnSaved event will occur after the barcode symbol is saved.

Parameters:

- **FilePath**: String; Specifies the file path and name for the image file.

Return:

This method can return one of these values (these consts are defined in the pSave2D unit):

- **ERROR_OK (0)**: The method invocation succeeded.
- **ERROR_NO_BARCODE_COMPONENT (1)**: The method invocation failed because the Barcode2D property is not link to a barcode component.
- **ERROR_INVALID_BARCODE (2)**: The method invocation failed. Because the length of Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value is invalid in the barcode component, corresponding to the OnInvalidLength or OnInvalidDatalength (only for Delphi/C++ Builder 2009 or later) event of barcode component will occur. Or because there is any invalid character in the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value, corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event of barcode component will occur.
- **ERROR_WRITE_PICTURE (3)**: The method invocation failed because the picture file write error. Perhaps the file system fulls, or the access is denied.

Annex C. Events

C.1 TBarcode2D

C.1.1 OnChange

Occurs when the value of Barcode or Data (only for Delphi/C++ Builder 2009 or later) property is changed.

Syntax:

```
property OnChange: TNotifyEvent;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.

Note:

For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D barcode components, if the properties' value of the linear component is changed, only the OnChange event of linear component occurs. The event ofTBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC components doesn't occur.

C.1.2 OnDrawBarcode

Occurs after representing the barcode symbol. Write an OnDrawBarcode event handler to modify the barcode symbol after it was represented.

Syntax:

```
type
  { Defined in the pBarcode2D unit }
  TUserWorkArea = record
    WCanvas: TCanvas;
    WBarcode: string;
    WBarColor: TColor;
    WSpaceColor: TColor;
    WModule: Integer;
    WLeftInPixels: Integer;
    WTopInPixels: Integer;
    WAngle: Integer;
    WShowQuietZone: Boolean;
    WMirror: Boolean;
    WAnsiBarcode: AnsiString;
    WBarcodeLength: Integer;
    WQuietZoneWidthInModules_Left: Integer;
```

```
    WQuietZoneWidthInModules_Top: Integer;
    WQuietZoneWidthInModules_Right: Integer;
    WQuietZoneWidthInModules_Bottom: Integer;
    WSymbolZoneWidthInModules: Integer;
    WSymbolZoneHeightInModules: Integer;
    WTotalWidthInPixels: Integer;
    WTotalHeightInPixels: Integer;
    WSymbolZoneOffsetInPixels_Left: Integer;
    WSymbolZoneOffsetInPixels_Top: Integer;
    WAlpha: Double;
    WOrgin: TPoint;
  end;
  { Defined in the pBarcode2D unit }
  PUserWorkArea = ^TUserWorkArea;
  { Defined in the pBarcode2D unit }
  TOnDrawBarcode = procedure (Sender: TObject; Canvas: TCanvas; PWorkArea: PUserWorkArea)
    of object;
property OnDrawBarcode: TOnDrawBarcode;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **Canvas**: TCanvas; The target canvas, the barcode symbol will be represented in it.
- **PWorkArea**: PUserWorkArea; It points to theTUserWorkArea record. The record contains fields defining the barcode symbol. See also the "TUserWorkArea" record.

C.1.3 OnEncode

Occurs when the value of Barcode property is changed and the component is updating its barcode symbol, or one of the Clear, Draw, Size, DrawTo, DrawToSize, Print, PrintSize, GetParity (only TBarcode2D_QRCode) , GetCheckSum (only TBarcode2D_PDF417), GetCheckSum (only TBarcode2D_MicroPDF417) method is called.

It's useful if you want to encode the barcode text in your own encoding scheme before generate the barcode symbol. See also the "How to encode the UNICODE text in a 2D barcode symbol" article.

Syntax:

```
type
  { Defined in the pCore2D unit }
  TOnEncode = procedure (Sender: TObject; var Data: AnsiString; Barcode: string) of
    object;
property OnEncode: TOnEncode;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **Data**: AnsiString; The barcode text that's encoded by your own encoding scheme should be returned in this parameter.

  The initial value is the barcode text in AnsiString (the barcode text will be converted to AnsiSring if the UNICODE string is supported by the Delphi or C++ Builder). By default, it's in ANSI encoding scheme.

- **Barcode**: String; It is the barcode text in string type (it's an UnicodeString if the UNICODE string is supported by the Delphi or C++ Builder, otherwise it's an AnsiString).

Its value is equal to the Barcode property if the Clear, Draw, Size, DrawTo (syntax 1), DrawToSize (syntax 1), Print (syntax 1), or PrintSize (syntax 1) method is called or a component is updating its barcode symbol. And the value of the parameter is equal to the Barcode parameter if the DrawTo (syntax 2), DrawToSize (syntax 2), Print (syntax 2), PrintSize (syntax 2), GetParity (syntax 1) (only TBarcode2D_QRCode), GetCheckSum (syntax 1) (only TBarcode2D_PDF417), or GetCheckSum (syntax 1) (only TBarcode2D_MicroPDF417) method is called.

Note:

The event doesn't occur when the DrawTo (syntax 3), DrawToSize (syntax 3), Print (syntax 3), PrintSize (syntax 3), GetParity (syntax 2) (only TBarcode2D_QRCode) , GetCheckSum (syntax 2) (only TBarcode2D_PDF417), or GetCheckSum (syntax 2) (only TBarcode2D_MicroPDF417) method is called.

C.1.4 OnInvalidChar

Occurs if there is any invalid character in the Barcode property value.

Syntax:

```
type
  { Defined in the pBarcode2D unit }
  TOnInvalidChar = procedure (Sender: TObject; Index: Integer; BarcodeChar: Char;
    LinearFlag: Boolean) of object;
property OnInvalidChar: TOnInvalidChar;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **Index**: Integer; The index position of first invalid character in the barcode text that is specified by the Barcode property, the index 1 denotes that the first character is invalid character.
- **BarcodeChar**: Char; The first invalid character in the barcode text that is specified by the Barcode property. The character is a WideChar if the UNICODE is supported by the Delphi or C++ Builder. Otherwise it is an AnsiChar.
- **LinearFlag**: Bolean; It denotes whether the invalid character is in the the linear component of an EAN.UCC composite barcode symbol.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D barcode components, If the value of the parameter is true, the invalid character is in the Barcode property's value of the linear component that is specified by the Linear property, and both of the Index and BarcodeChar parameters are depend on the Barcode property's value of the linear component. Otherwise, the invalid char is in the Barcode property's value of the 2D barcode component, and both of the Index and BarcodeChar parameters are depend on the Barcode property's value of the 2D component.

  For other barcode components, the value of the parameter is always false.

Note:

- If the Locked property is set to false, the event occurs when any component property is changed to cause the barcode is redrew, or when the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.
- If the Locked property is set to true, the event occurs when the Locked property is set to false to cause the barcode is redrew, or the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.

C.1.5 OnInvalidDataChar

If you use the Delphi/C++ Builder 2009 or later, you can use the Data property to specify a block of binary (bytes) data, and encode it into a barcode symbol, or specifies the barcode text in AnsiString format. The event occurs if there is any invalid character in the Data property value.

Syntax:

```
type
  { Defined in the pBarcode2D unit }
  TOnInvalidDataChar = procedure (Sender: TObject; Index: Integer; DataChar: AnsiChar;
    LinearFlag: Boolean) of object;
property OnInvalidDataChar: TOnInvalidDataChar;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **Index**: Integer; The index position of first invalid character in the barcode text that is specified by the Data property, the index 1 denotes that the first character is invalid character.
- **DataChar**: AnsiChar; The first invalid character in the barcode text that is specified by the Data property.
- **LinearFlag**: Bolean; It denotes whether the invalid character is in the the linear component of an EAN.UCC composite barcode symbol.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D barcode components, If the value of the parameter is true, the invalid character is in the Data property's value of the linear component that is specified by the Linear property, and both of the Index and DataChar parameters are depend on the Data property's value of the linear component. Otherwise, the invalid char is in the Data property's value of the 2D barcode component, and both of the Index and DataChar parameters are depend on the Data property's value of the 2D component.

  For other barcode components, the value of the parameter is always false.

Note:

- If the Locked property is set to false, the event occurs when any component property is changed to cause the barcode is redrew, or when the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.
- If the Locked property is set to true, the event occurs when the Locked property is set to false to cause the barcode is redrew, or the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.
- The event is available only for the Delphi/C++ Builder 2009 or later.

C.1.6 OnInvalidDataLength

If you use the Delphi/C++ Builder 2009 or later, you can use the Data property to specify a block of binary (bytes) data, and encode it into a barcode symbol, or specifies the barcode text in AnsiString format. The event occurs when the length of the Data property value is invalid.

Syntax:

```
type
  { Defined in the pBarcode2D unit }
  TOnInvalidDataLength = procedure (Sender: TObject; Data: AnsiString; LinearFlag:
    Boolean) of object;
property OnInvalidDataLength: TOnInvalidDataLength;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **Data**: AnsiString; The invalid value of the Data property.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D barcode components, if the value of the LinearFlag parameter is true, it's the Data property's value of the linear barcode component that's specified in the Linear property. Otherwise, it is the Data property's value of the 2D barcode component.

  For other barcode components, it's the Data property's value of the 2D barcode component.
- **LinearFlag**: Bolean; It denotes whether the barcode's length of the linear component is invalid in an EAN.UCC composite barcode symbol.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D barcode components, if the value of the parameter is true, it denotes the length of the Data property's value of the linear barcode component that's specified by the Linear property is invalid. Otherwise, it denotes the length of the Data property's value of the 2D barcode component is invalid.

  For other barcode components, the value of the parameter is always false.

Note:

- In general, it occurs when the length of barcode text exceeds the maximum length limit. For the TBarcode2D_RSSLimited, TBarcode2D_RSS14, and TBarcode2D_AztecRunes components, if the Data property value is an empty string, the event occurs too.
- If the Locked property is set to false, the event occurs when any component property is changed to cause the barcode is redrew, or when the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.
- If the Locked property is set to true, the event occurs when the Locked property is set to false to cause the barcode is redrew, or the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.
- The event is available only for the Delphi/C++ Builder 2009 or later.

C.1.7 OnInvalidLength

Occurs when the length of the Barcode property value is invalid.

Syntax:

```
type
  { Defined in the pBarcode2D unit }
  TOnInvalidLength = procedure (Sender: TObject; Barcode: string; LinearFlag: Boolean) of
    object;
property OnInvalidLength: TOnInvalidLength;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **Barcode**: String; The invalid value of the Barcode property.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D barcode components, if the value of the LinearFlag parameter is true, it's the Barcode property's value of the linear barcode component that's specified in the Linear property. Otherwise, it is the Barcode property's value of the 2D barcode component.

  For other barcode components, it's the Barcode property's value of the 2D barcode component.
- **LinearFlag**: Bolean; It denotes whether the barcode's length of the linear component is invalid in an EAN.UCC composite barcode symbol.

  For the TBarcode2D_CCA, TBarcode2D_CCB, and TBarcode2D_CCC 2D barcode components, if the value of the parameter is true, it denotes the length of the Barcode property's value of the linear barcode component that's specified by the Linear property is invalid. Otherwise, it denotes the length of the Barcode property's value of the 2D barcode component is invalid.

  For other barcode components, the value of the parameter is always false.

Note:

- In general, it occurs when the length of barcode text exceeds the maximum length limit. For the TBarcode2D_RSSLimited, TBarcode2D_RSS14, and TBarcode2D_AztecRunes components, if the Barcode property value is an empty string, the event occurs too.
- If the Locked property is set to false, the event occurs when any component property is changed to cause the barcode is redrew, or when the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.
- If the Locked property is set to true, the event occurs when the Locked property is set to false to cause the barcode is redrew, or the Draw, Clear, Size, DrawTo (Syntax 1), DrawToSize (Syntax 1), Print (Syntax 1), or PrintSize (Syntax 1) method is called. Even if the Image property isn't specified.

C.2 TSave2D

C.2.1 OnSave

Occurs before save the barcode symbol to an image file, when the Save method is called.

Syntax:

```
type
  { Defined in the pSave2D unit }
  TOnSave = procedure (Sender: TComponent; var FilePath: string; var Continue: Boolean)
    of object;
property OnSave: TOnSave;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **FilePath**: string; The file path and name of the image file that you want to save the barcode symbol to. You can change the

parameter value in the event handle.

- **Continue**: Boolean; Specifies whether to save the barcode symbol to an image file. By default, its value is true. Changing it to false will cancel the saving operation, in this case, the OnSaved event will not be occurred.

C.2.2 OnSaved

Occurs after save the barcode symbol to an image file, when the Save method is called.

Syntax:

```
type
  { Defined in the pSave2D unit }
  TOnSaved = procedure (Sender: TComponent; FilePath: string; ErrorCode: Integer) of
    object;
property OnSaved: TOnSaved;
```

Parameters:

- **Sender**: TObject; It is the object whose event handler is called.
- **FilePath**: string; The file path and name of the image file that you want to save the barcode symbol to.
- **ErrorCode**: Integer; Return the error status code of the saving operation. The Save method will return the error code value too. It can be one of these value (these consts are defined in the pSave2D unit):
  - **ERROR_OK (0)**: The saving operation succeeded.
  - **ERROR_NO_BARCODE_COMPONENT (1)**: The saving operation failed because the Barcode2D property is not link to a barcode component.
  - **ERROR_INVALID_BARCODE (2)**: The saving operation failed, because the length of Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value is invalid in the barcode component, corresponding to the OnInvalidLength or OnInvalidDatalength (only for Delphi/C++ Builder 2009 or later) event of barcode component will occur. Or because there is any invalid character in the Barcode or Data (only for Delphi/C++ Builder 2009 or later) property value, corresponding to the OnInvalidChar or OnInvalidDataChar (only for Delphi/C++ Builder 2009 or later) event of barcode component will occur.
  - **ERROR_WRITE_PICTURE (3)**: The saving operation failed because the picture file write error. Perhaps the file system fulls, or the access is denied.

2D Barcode VCL Components User Manual

13.3.0.2668 2025-04-06

Copyright © 2001-2025 Han-soft Corporation. All rights reserved.