ssBarcode v2.3 - *The best barcode VCL for Delphi, hands-down!*

Properties Methods Events

Thank you for your interest in our bar coding VCL. There are many bar coding products for developers in existence, but all are either in DLL or VBX form. With the release of ssBarCode from Soft Sector, Inc., developers now have a Delphi-native option to implement bar coding into their applications.

In this version, ssBarCode supports the six most popular bar code symbologies: Code 128, Code 39 (or 3 of 9), EAN-13, Interleaved 2 of 5, Postnet, and UPC version A. In addition, the 2 and 5 digit addons for UPC and EAN are supported. If you have any symbologies you would like to see implemented, please let us know!

About Soft Sector, Inc.

Soft Sector, Inc. (SSI) is a Cleveland, OH based provider of hardware and software solutions. Our main language for development is Borland Delphi. In the course of our application development, we have created a number of components and tools that would be useful to other developers. We are in the process of packaging these components for retail distribution.

Contacting Soft Sector, Inc.

We can be contacted in a variety of ways:

US Mail: Soft Sector, Inc., PO Box 81480, Cleveland, OH 44181 USA
E-Mail: ssbarcode@softsector.com
CompuServe: 104274,1510
Phone: (216) 777-3612
Fax: (216) 777-3646

Toll-Free Order Line: (888) 533-7987

WWW: http://www.softsector.com

The newest version of ssBarcode, as well as our other components, can always be found at our Web site.

Installation - Delphi 1.x

You can install ssBarcode as you would any other component. Specifically, follow these steps:

** Make sure you unZIPed the ZIP file with the -D option!! **

1. Unzip the .ZIP file into a directory (example: \SSBC) with the -D option. There will be several directories created. The actual components reside in LIB16.

2. Go into Delphi and access the OPTIONS pulldown menu. Then choose INSTALL COMPONENTS...

3. Click on the ADD button. Another dialog box will pop up. Click on the BROWSE button.

4. A file selection common dialog will pop up. Select the directory that contains the files for your version of Delphi (see #1 above). In the “List Files of Type” drop down, you should select "DCU" files.

5. You should see three .DCU files. You should install SSBC.DCU first.

- If you plan to use ssBarcode along with QuickReports v1.x, you should repeat this process and choose QRSSBC.DCU.
- If you plan to use ssBarcode along with QuickReports v2.0, you should repeat this process and choose QR20SSBC.DCU.
- If you plan to use ssBarcode along with ReportPrinter Pro, you should repeat this process and choose RPSSBC.DCU.

6. Now click OK in the Install Components dialog box. Delphi’s VCL will now recompile. When it’s done, you should have a palette page named "Soft Sector". On it will appear an icon for the ssBarCode component. If you installed the either of the QuickReports components, they too will appear on the "Soft Sector" page. If you installed the ReportPrinter Pro component, it will appear on the "Report" page along with the other ReportPrinter components.

That’s all there is to it! You’re now ready to start creating bar codes!

Note to unregistered users: this restricted version of ssBarcode will only run while the Delphi IDE is running. That means in order to include it in your applications, you *must* pay the registration fee! See Ordering Information for more information on this extremely useful, yet affordable product.

Installation of Help File - Delphi 1.x

We have made it possible to link ssBarcode’s help file with Delphi’s help system so you can get context-sensitive help during design-time the same way you would with any Delphi component. To do so, follow these steps:

1. Copy the file "SSBC.KWF" to your \DELPHI\HELP directory.

2. Copy the file "SSBC.HLP" to your \DELPHI\BIN directory.

3. Run the program "HELPINST.EXE" in your \DELPHI\HELP directory. You should also have an icon for this program in your Delphi group.

4. Go to the FILE menu and choose OPEN. Select the file "DELPHI.HDX" which is located in your \DELPHI\BIN directory.

5. Go to the KEYWORDS menu and choose ADD KEYWORD FILE. Select the file "SSBC.KWF" that you copied to your \DELPHI\HELP directory.

6. Go to the FILE menu and choose SAVE. Then you can exit the HelpInst program.

That's it! Now when you're in the Delphi IDE, you can click on a TssBarcode, press [F1] and get ssBarcode's help file.

Installation - Delphi 2.x

You can install ssBarcode as you would any other component. Specifically, follow these steps:

** Make sure you unZIPed the ZIP file with the -D option!! **

1. Unzip the .ZIP file into a directory (example: \SSBC) with the -D option. There will be several directories created. The actual components reside in LIB32.

2. Go into Delphi and access the COMPONENT pulldown menu. Then choose INSTALL...

3. Click on the ADD button. Another dialog box will pop up. Click on the BROWSE button.

4. A file selection common dialog will pop up. Select the directory that contains the files for your version of Delphi (see #1 above). In the "List Files of Type" drop down, you should select "DCU" files.

5. You should see three .DCU files. You should install SSBC.DCU first.

- If you plan to use ssBarcode along with QuickReports v1.x, you should repeat this process and choose QRSSBC.DCU.
- If you plan to use ssBarcode along with QuickReports v2.0, you should repeat this process and choose QR20SSBC.DCU.
- If you plan to use ssBarcode along with ReportPrinter Pro, you should repeat this process and choose RPSSBC.DCU.

6. Now click OK in the Install Components dialog box. Delphi's VCL will now recompile. When it's done, you should have a palette page named "Soft Sector". On it will appear an icon for the ssBarCode component. If you installed either of the QuickReports components, they too will appear on the "Soft Sector" page. If you installed the ReportPrinter Pro component, it will appear on the "Report" page along with the other ReportPrinter components.

Installation of Help File - Delphi 2.x

We have made it possible to link ssBarcode's help file with Delphi's help system so you can get context-sensitive help during design-time the same way you would with any Delphi component. To do so, follow these steps:

1. Copy the files "SSBC.KWF" and "SSBC.HLP" to your \DELPHI\HELP directory.

3. Run the program "HELPINST.EXE" in your \DELPHI 2.0\HELP\TOOLS directory. You may also have an icon for this program in your Delphi group.

4. Go to the FILE menu and choose OPEN. Select the file "DELPHI.HDX" which is located in your \DELPHI 2.0\BIN directory.

5. Go to the KEYWORDS menu and choose ADD KEYWORD FILE. Select the file "SSBC.KWF" that you copied to your \DELPHI 2.0\HELP directory.

6. Go to the FILE menu and choose SAVE. Then you can exit the HelpInst program.

That's it! Now when you're in the Delphi IDE, you can click on a TssBarcode, press [F1] and get ssBarcode's help file.

Demo Programs

We've written four simple demo programs to show off some of the features of ssBarcode, including the different symbologies, bar width specification, and barcode printing. The first project name is DEMOBC.DPR, and it is located in the DEMOS directory. All of the features of this demo program are very easy to understand, and some of the code is commented. The second demo program is named QRDEMO.DPR. This is a simple project that illustrates how to create a QuickReports v1.x report including a barcode. It makes use of the DBDEMO database files that came with Delphi. The third demo is named RPDEMO.DPR. It is identical to DEMOBC, except that it uses ReportPrinter when it needs to print the barcode. The fourth demo program is named QR20DEMO.DPR. It's functionality is almost exactly the same as QRDEMO, except that it uses QuickReports v2.0.

Note that when you open any of the demo projects, you may get the error message "Error reading symbol file". This is completely normal; we don't distribute the symbol file since it can be upwards of 500k and it is rebuilt when you recompile the project anyway

Using ssBarcode with QuickReports v1.0

We have done the necessary work to make ssBarcode completely compatible with the very popular report writer QuickReports. If you are unfamiliar with QR, the following may not make much sense to you. If you have upgraded to QuickReports v2.0, you will need to use the QR20ssBarcode component.

You will remember that you actually installed two components, ssBarcode and QRssBarcode. BOTH components are necessary to make your program work with both QR and ssBarcode. QRssBarcode is what some call a "wrapper" component. It needs to be linked to an actual instance of ssBarcode, either at design time or run time. QRssBarcode doesn't have any properties for changing the barcode parameters. It simply has a property called "Barcode" that you link to an instance of ssBarcode that actually does the work.

In other words, you create an ssBarcode that **controls** the QRssBarcode. Whatever changes you make to the ssBarcode will be reflected in the QRssBarcode. I know this sounds a little confusing, but it really is rather simple once you play with it.

Why did we do it this way? We were having problems with people using different versions of QuickReports than the one that we compiled with. So we've included the full source code to QRssBarcode, which means it will compile no matter what version of QR you are using.

NOTE ABOUT BARCODES WITH QUICKREPORTS v1.0: QuickReports does its own scaling based on the number of pixels the screen can display per inch. Obviously, most printers can display many more pixels (or dots) per inch. As such, bar codes printed using QuickReports tend to be of slightly lower quality than the same barcode printed using the ssBarcode.PrintBarcode method. Nevertheless, we have done extensive testing with barcodes printed with QR and have had favorable results.

For more info on QuickReports, see their Web page at http://www.qusoft.no

Symbologies

A symbology is a set of rules that govern how to create a bar code based on what data is being coded. There are many symbologies in existence, some general purpose and others for very specific applications. Each bar code has its pro's and con's. This version of ssBarCode contains the six major symbologies, as described below.

Code 128
Code 39
EAN-13
Interleaved 2 of 5
PostNet
UPC-A

Code 128

Code 128 is an extremely flexible symbology and can be used to create alphanumeric barcodes. It derives its name from the fact that it can represent all 128 ASCII symbols. If you are unfamiliar with Code 128, I will try to give you a quick tutorial, as Code 128 is more complicated than most other one-dimensional barcodes.

The first thing you must know about Code 128 is that it has three different character sets (A, B, and C). That means one set of bars can mean three different things based on which set is currently in use. The other tricky bit is that you have the option of switching character sets **in the middle of the code**. I think you see now why this is a more difficult code.

Character set A can encode most symbol characters (!,$,/, etc.) as well as all uppercase letters and the ten digits. In addition, it can encode the "special" ASCII characters, such as carriage return, line feed, null, escape, etc.

Character set B encodes all symbol characters, **upper and lowercase** letters and the ten digits.

Character set C encodes 100 **pairs of numbers**. That is, one set of bars could mean "05" or "43", etc. This effectively doubles the density of Code 128 when using numbers only. Also note, because they are numeric pairs, the Code C portion must have an even number of digits.

Now that you know what the differences in the codes are, you now have to know how to use them with ssBarcode. Since all normal ASCII characters are used by Code 128, we must use special characters to tell the component what to do. ssBarcode uses ASCII values #208 through #242 for this purpose. If you're using the Object Inspector to type the Data property, you hold down the Alt key, then type (on the numeric keypad) 0 + the number (Alt-0208 for instance). The table below lists all of the special characters.

#208 START A
#209 START B
#210 START C

These are the special START CODE characters. Each Code 128 barcodes MUST START with one of the three above. Which start code you use is obviously which character set is active. You cannot use a start code in the middle of a symbol. See the table below for the special codes used to switch to a different character set.

Special Char	Character Set A	Character Set B	Character Set C
#201	Function 3	Function 3	<none>
#202	Function 2	Function 2	<none>
#203	TempShift to B	TempShift to A	<none>
#204	**Switch to C**	**Switch to C**	<none>
#205	**Switch to B**	Function 4	<none>
#206	Function 4	**Switch to A**	**Switch to A**
#207	Function 1	Function 1	Function 1
#211	NUL	<none>	<none>
#212	SOH	<none>	<none>
#213	STX	<none>	<none>
#214	ETX	<none>	<none>
#215	EOT	<none>	<none>
#216	ENQ	<none>	<none>
#217	ACK	<none>	<none>

#218	BEL	<none>	<none>
#219	BS	<none>	<none>
#220	HT	<none>	<none>
#221	LF	<none>	<none>
#222	VT	<none>	<none>
#223	FF	<none>	<none>
#224	CR	<none>	<none>
#225	SO	<none>	<none>
#226	SI	<none>	<none>
#227	DLE	<none>	<none>
#228	DC1	<none>	<none>
#229	DC2	<none>	<none>
#230	DC3	<none>	<none>
#231	DC4	<none>	<none>
#232	NAK	<none>	<none>
#233	SYN	<none>	<none>
#234	ETB	<none>	<none>
#235	CAN	<none>	<none>
#236	EM	<none>	<none>
#237	SUB	<none>	<none>
#238	ESC	<none>	<none>
#239	FS	<none>	<none>
#240	GS	<none>	<none>
#241	RS	<none>	<none>
#242	US	<none>	<none>

Got all that? <smile> Ok, so here's what that table means: if the current character set is A, then a #205 means Switch to Code B. If the current set was B, that same #205 means Function 4.

The Switch codes are all boldfaced because you'll probably use those the most. A Switch code means "Switch to Character Set X for the remainder of the code, or until another Switch code is encountered."

The Temp-Shift codes means "Shift to Code X for **the next character only**."

The "Function" codes mean different things based on your barcode reader. Generally they are explained in the reader's documentation.

Ok, I think a few examples are in order:

Let's say you want to encode the text "CODE 128". Character set A is sufficient. Here's how:

```
ssBarCode1.Data := #208 + 'CODE 128';
```

If you were using the Object Inspector, you would've typed {Alt-0208}CODE 128.

Here's a little more advanced example. We want to encode "1234 abcd". We can use Set C, since we have an even number of digits. Then we have to switch to Set B for the lowercase letters.

```
ssBarCode1.Data := #210 + '1234' + #205 + ' abcd';
```

Ok, one last example. We want to encode "Soft Sector 1996", but we want a Carriage Return in between "Soft" and "Sector". Here goes:

```
   ssBarCode1.Data := #209 + 'Soft' + #203 +   #224 + 'Sector ' + #204 +
'1996';
```

See what happened there? We started off in Set B and wrote "Soft", then we Temp Shifted to A to encode the CR. The we wrote "Sector" in Set B. Finally, we switched to Set C to encode "1996."

I realize that this looks very complex. But spend some time playing around and it will get much easier.

Code 39

Code 39 was the first alphanumeric symbology to be developed. It is now in wide use and is the de facto standard for non retail applications. It can encode the 26 letters of the alphabet (in upper case), the 10 digits, and the symbols "- . $ / + %" Code 39 gets its name because each character has five bars and four spaces (nine elements); of those nine, three are wide. Thus, "3 of 9." If you need to encode any non-numeric data, its a choice between 3 of 9 and Code 128. In most cases, a proper implementation of Code 128 is more efficient. However, 3 of 9 is more universally accepted by barcode readers.

EAN-13

EAN stands for the European Article Numbering. It is also compatible with JAN and IAN. EAN is the standard for retail applications in countries other than the United States. EAN-13 encodes 13 digits, but does so in the same amount of space that UPC encodes 12 digits. This is accomplished because the first digit is actually encoded in the parity of the left side bars, rather than by actual bars.

The makeup of an EAN code is similar to a UPC, except that it starts with a 1 to 3 digit code identifying the country of origin. The standard EAN country codes can be found in numerous places, including several Web sites. Note that an EAN code with a first digit of "0" is exactly the same as a UPC code encoding the same digits.

EAN codes are becoming widespread even in the U.S., especially on books. An ISBN number can be encoded in the EAN main code, then the price of the book can be encoded in the 5 digit addon. This is commonly called "Bookland EAN". Note that all Bookland EAN codes must start with "978".

Interleaved 2 of 5

Interleaved 2 of 5 (sometimes abbreviated I2of5 or ITF) is a high-density, continuous numeric symbology mainly used in the distribution industry. Many packages you receive have ITF bar codes on them. ITF is a very efficient symbology because it encodes data both in the bars and spaces. Each digit is made up of five bars, of which two are wide; thus the name "2 of 5". Since data is encoded in both bars and spaces, all ITF bar codes must have an even number of digits! Many applications will add a trailing zero if the number to be encoded contains an odd number of digits. If you are encoding less than ten digits, ITF is the most efficient bar code.

One of the problems with ITF is that a partial scan has a high probability of decoding as a valid, but shorter, ITF symbol. To minimize this risk, ITF bar codes are often used with bearer bars (sometimes called protection stripes). Bearer bars are horizontal bars running along the top and bottom of the symbol. They decrease the probability that a partial scan will decode as valid.

PostNet

This symbology was developed by the United States Post Office for the purpose of marking postal items so that they could be sorted by automatic equipment. In the strictest sense, Postnet is not a bar code, since information is not encoded into the widths of the bars. Postnet encodes using the heights of the bars instead. Postnet codes generally match the length of a Zip code; that is, either 5 or 9 digits. Recently, the Post Office has accepted a Postnet code with 11 digits, the last two being used as the first two digits of a street address, PO Box, apartment number, etc. For more information about the proper use of Postnet codes, contact your local Post Office.

UPC-A

UPC (Universal Product Code) is the bar code of choice for the retail industry in the United States. UPC is a coding system as well as a symbology; it is designed to uniquely identify a product and its manufacturer. The actual UPC code is a 10-digit code: the first five digits represent the manufacturer, the next five as a unique product identifier code assigned by the manufacturer. This 10-digit code is preceded with a "number system" digit and followed by a check digit. Thus the UPC-A bar code encodes 12 digits of data.

When you apply for a UPC manufacturer number, the UCC (Uniform Code Council) assigns you a six digit number; the first digit is a "Number System" digit from 0 to 9. The meanings of each of these digits is listed below. The next five digits is your actual manufacturer number.

The Number System assignments are as follows:

0 - 92,000 manufacturer identification numbers; 8,000 locally assigned numbers
1 - Reserved
2 - Random weight consumer packages
3 - Drug products
4 - In-store marking without format
5 - UPC coupons
6 - 100,000 manufacturer identification numbers
7 - 100,000 manufacturer identification numbers
8 - Reserved
9 - Reserved

Since all UPC-A codes encoded 12 digits of data, UPC-A is a fixed-width symbology.

Future Enhancements

ssBarcode has grown enormously since its 1.0 release. We believe that ssBarcode rivals any competitor, be it VCL, VBX, or DLL. If you have any features you'd like to see, please let us know. Specifically, if there are symbologies youd like use to implement, drop us a line.

- Additional symbologies: UPC-E, EAN-8, Codabar, FIM, Code 93, Code 49, 2-dimensional codes?

Revision History

02-17-97 Version 2.3
- Added property AutoSizeFont to specify whether you want ssBarcode to size the human readable font automatically
- Added property FontAlignment to allow you to specify an alignment for the human readable text for some symbologies
- Added property MeasurementUnits to allow you to specify the BarWidth property in either Inches or Millimeters
- Added support for QuickReports v2.0 with the QR20ssBarcode component
- Added new demo program QR20DEMO.DPR illustrating how to use ssBarcode with QuickReports v2.0
- Added XBDE compiler directive, allowing you to compile a version of ssBarcode that does not use the BDE
- Bug fix: the component now fills in the background at run-time properly
- Bug fix: the human readable font on UPC and EAN symbologies now correctly appears when using the Right to Left orientation
- Bug fix: the component now correctly recognizes the values of the Font.Style property (e.g. fsBold, fsItalic, etc.)
- Bug fix: fixed numerous problems with the strange font sizing some people were experiencing
- Added logical limits on the BarWidth property to protect against entering a large value that would hang the system
- Change: the Orientation property is now of type TBCOrientation, so as to not conflict with Delphi's TOrientation

09-12-96 Version 2.21
- Updated RpssBarcode component
- Added KWF so that help file can be linked to Delphi’s help system
- Corrected some errors in help file
- Added MANUAL.WRI with proper formatting for Windows Write or WordPad
- Updated Soft Sector’s phone numbers in help file, manual

07-23-96 Version 2.2
- Added RPssBarcode component for full compatibility with ReportPrinter Pro from Nevrona Designs
- Added Windows HLP file
- Bug fix: corrected problem where existing controls on a form were being scaled because of the ssBarcode control
- Bug fix: now respects the BearerBars property at run-time
- Change: new barcodes dropped on a form now have the Color property equal to clWhite

05-15-96 Version 2.1
- Changed the way ssBarcode interacts with QuickReports
- Added a new component, QRssBarcode
- Added 32-bit version
- Bug fix: barcodes printed with QR now respect the AutoSize property correctly

- Changed demo program, added new QRDEMO program

05-07-96 Version 2.0
- Added two symbologies: Code 128, EAN-13
- Added 2 and 5 digit AddOn codes
- Added Font property for human readable
- Made data-aware (added DataField and DataSource properties)
- Made QuickReports compatible
- Added additional error checking
- Streamlined internal code

04-15-96 Version 1.5
- Changed the BarWidth to type Single. Now you specify the BarWidth in Inches (generally fractions thereof)
- Added property WideBarRatio, allowing you to specify the narrow-to-wide bar ratio for symbologies that support it (Code 39 and Int2of5)
- Added method PrintBarcode, allowing you to easily print barcodes on any Windows-supported printer

03-24-96 Version 1.0
Initial public release

Methods

PrintBarcode

Properties

AddOn
AutoSize
AutoSizeFont
BarCodeType
BarColor
BarWidth
BearerBars
CalcCheckDigit
Canvas
Color
Data
DataField
DataSource
Font
FontAlignment
MeasurementUnits
Orientation
PrintHumanReadable
Visible
WideBarRatio

Events

OnClick
OnDblClick
OnMouseDown
OnMouseMove
OnMouseUp

AddOn property

Declaration

```
property AddOn : string
```

Description

The AddOn property allows you to implement either 2 or 5 digit addon codes for UPC or EAN. If you try to enter an AddOn code with the BarCodeType property set to either bcUPC_A or bcEAN_13, you will raise an exception.

AddOn codes are used for a few rather specific purposes. The 2 digit addon code is used to track the serial number of periodicals, mainly magazines. The 5 digit addon has several purposes, but mainly it is used in conjunction with EAN-13 for bar coding books. A books ISBN number can be encoded in the actual EAN code. Then the addon code is used to indicate the price of the book and the currency. This form of using EAN is commonly known as "Bookland EAN."

AutoSize Property

Declaration

```
property AutoSize : boolean
```

Description

The AutoSize property will automatically size the bar code to certain proportions based a number of calculations. The first factor is the BarWidth property; the width of the component will be the BarWidth property times the actual number of bars and spaces. The height of the component will be based on the standard accepted height to width ratio for each symbology.

If you set AutoSize to false, you can size the component as you see fit. However, there is no guarantee that the resulting code will fit in the space you allot.

AutoSizeFont Property

Declaration

```
property AutoSizeFont : boolean
```

Description

By default, ssBarcode will auto size the human readable font to a standard accepted size based on the size of your barcode. However, if you would rather input the font size yourself, you can set AutoSizeFont to FALSE. Then simply type a value in the Font.Size property and the human readable font will reflect your selection.

BarCodeType Property

Declaration

```
property BarCodeType : TBarCodeType
```

Description

This is the main property of the component. It defines which symbology you want to use to create the bar code. The possible values for TBarCodeType are:

{bcCode128, bcCode39, bcEAN_13, bcInt2of5, bcPostnet, bcUPC_A}

See the section Symbologies for a more in-depth description of each symbology.

When you set the BarCodeType property, the program will check the Data property to see if the data entered is acceptable for the chosen bar code type. For example, you cannot have alpha characters in an Interleaved 2 of 5 code. If the data is not acceptable, an exception of type EBarCodeError will be raised.

BarColor Property

Declaration

```
property BarColor : TColor
```

Description

Use this property to set a color for the actual bars themselves as well as the human readable text beneath the bar code. Note that for best scanning, bar codes should be black bars on a white background, but you do have the option to change this.

BarWidth Property

Declaration

```
property BarWidth : single
```

Description

This property allows you to change the width, in either inches or millimeters, of each bar. Set the MeasurementUnits property to specify if you are going to use Inches or Millimeters. For bar codes that have a narrow to wide bar ratio, the wide bar width is calculated based on the WideBarRatio property. If you have the AutoSize property set to True, the component will resize when you change this property.

Each symbology has a standard accepted BarWidth, listed below. Of course, these are only recommendations and you are free to use whatever width you choose.

Code 128 = varies, based on use. Generally 0.0070 inches to 0.015 inches. (0.1778 mm to 0.381 mm)
Code 39 = 0.010 inches (0.254 mm)
EAN-13 = 0.013 inches (0.3302 mm)
Int. 2 of 5 = 0.015 inches (0.381 mm)
Postnet = 0.020 ** (We strongly recommend using this value for Postnet) (0.508 mm)
UPC-A = 0.013 inches (0.3302 mm)

NOTE: We have successfully printed a Code 128 barcode on a 300 dpi laser printer with a bar width of 0.0070 inches. That is an extremely efficient code!

BearerBars Property

Declaration

```
property BearerBars : boolean
```

Description

Certain bar code symbologies call for bearer bars, or horizontal lines that run along the top and bottom of the code. The reason for these bars is that a partial scan can sometimes be interpreted as a complete scan. Bearer bars greatly reduce the chance of this happening.

As of version 1.0, only Interleaved 2 of 5 bar codes can have bearer bars. The other symbologies do not need them. If you try to set BearerBars equal to True for any other symbology, an EBarCodeError exception will be raised.

CalcCheckDigit Property

Declaration

```
property CalcCheckDigit : boolean
```

Description

Set this property to True if you want the component to automatically generate the check digit for UPC-A and EAN-13 bar codes. If CalcCheckDigit is False, then you must provide all 12 digits of the UPC code in the Data property, or an exception will be raised.

Data Property

Declaration

```
property Data : string
```

Description

This property is where you set the actual data to be encoded. Data for each bar code symbology must follow certain rules or an exception will be raised. See the topic Symbologies for a description of acceptable data for each type of bar code. Please note that Code 128 requires some special characters to function.

Font Property

Declaration

```
property Font : TFont
```

Description

This is the font that will be used to draw the human readable portion of the barcode. Note that this MUST be a scalable font! (either TrueType or ATM) Also note that you can change the fonts attributes (bold, italic, etc.), but the font's size, by default, is automatically calculated by the component. If you wish to specify a font size yourself, you must set the AutoSizeFont property to False. The Font's color is taken from the BarColor property, as all barcode specifications call for the bars and human readable text to be the same color.

If possible, you should always use the font OCR-B for your human readable. If you do not have this TrueType font, it is available from a variety of sources. Most bar code specification documents call for this font to be used.

FontAlignment Property

Declaration

```
property FontAlignment : TAlignment
```

Description

For most symbologies, you can use FontAlignment to set whether you'd like the human readable text to be Left-Justified, Right-Justified, or Centered underneath the barcode. Two symbologies that are not affected by this setting are EAN-13 and UPC-A. These two symbologies have special text placement that must be adheared to.

MeasurementUnits Property

Declaration

```
property MeasurementUnits : TMeasurementUnits
```

Description

This property allows the programmer to choose whether the BarWidth property will be specified in Inches or Millimeters. The possible values for MeasurementUnits are:

{muInches, muMMs}

By default, MeasurementUnits is muInches.

Orientation Property

Declaration

```
property Orientation : TBCOrientation
```

Description

The Orientation property is used to rotate the bar code in 90 degree increments. You would never want to have a bar code that is not rotated in an increment of 90 degrees. The human readable text is obviously also rotated. The possible values for TBCOrientation are:

{orLeft_Right, orRight_Left, orTop_Bottom, orBottom_Top}

orLeft_Right means that the bar code is created from Left to Right (it is the default).
orTop_Bottom means that the bar code is created from Top to Bottom, etc.

PrintHumanReadable Property

Declaration

```
property PrintHumanReadable : boolean
```

Description

This property allows you to set whether the component will output the human readable version of the bar code (i.e. the actual letters or numbers). Bar codes like UPC-A and EAN-13 should always have this set to True. As for the other bar codes, it is completely up to you.

WideBarRatio Property

Declaration

```
property WideBarRatio : single
```

Description

This property affects the narrow-to-wide bar ratio for symbologies that have such elements. The value you give this property will be multiplied by the value of BarWidth to get the wide bar width. Most symbologies dictate a wide bar ratio in the 2.0 to 3.0 range. You will want a higher WideBarRatio if your BarWidth is very small. Most applications call for this ratio to be 3.0.

PrintBarcode Method

Declaration

```
procedure PrintBarcode(X, Y, Height : single);
```

Description

This method makes it extremely easy to print barcodes from your application. Simply create an instance of ssBarcode, either at design-time or run-time, set the properties you need, and then call this method. ssBarcode handles all of the sticky implementations of printing, such as printer resolutions, margins, etc. NOTE: If you dropped ssBarcode on QuickReports band, you do not have to manually issue this command.

To use this method, supply X and Y values for the upper left hand corner of the barcode, in either inches or millimeters (depending on the value of the MeasurementUnits property). The Height parameter allows you to specify an exact height of the barcode, in inches or millimeters. You can also pass 0 (zero) as the Height, and ssBarcode will calculate the Height based on standard height-to-width ratio for the current symbology.

**** IMPORTANT NOTE **** This method uses the Printer variable (defined in the Printers unit that came with Delphi) to handle the physical printer. As such, you must follow the code below to print barcodes:

```
{procedure name}

begin
   Printer.BeginDoc; {****}
   ssBarCode1.PrintBarcode(1.0,1.0,0);
   {any other output to the printer here}
   Printer.EndDoc; {****}
end;
```

The lines with asterisks are absolutely critical. If you do not include them, it is unlikely that the barcode will print.

Another thing to note: the PrintBarcode method pays no attention to the AutoSize or Width properties. PrintBarcode will always print the entire barcode with no clipping.

Ordering Information

To Print This Form: Click on the "File" menu in the upper left, then choose "Print Topic".

-- You can now order through CompuServe's SWREG service --
-- See below for the SWREG # --

Please remit to:

Soft Sector, Inc.
PO Box 81480
Cleveland OH 44181 USA

If paying by credit card:

You can call in your order, toll free, to: (888) 533-7987
or from outside the United States, call: (216) 777-3612

You can fax your order to: (216) 777-3646

Name: _____

Company: _____

Address: _____

_____

City: _____ State: _____

Zip/Postal: _____ Country: _____

Phone # (optional): _____

E-Mail Address: _____

Where did you obtain our product?

_____

Comments or Suggestions:

Product Description	Price Per Unit	Qty	Total
ssBarcode v2.3 w/SOURCE CODE SWREG #11923	$75.00	x	=
ssImage v1.0	$5.00	x	=

ssDBUtil v1.0 $25.00 x =

Shipping/Handling via US Mail $3.00 (flat rate) =
-- or --
Shipping/Handling via E-Mail $1.00 (flat rate) =

Subtotal:

Ohio Residents Add 7% Sales Tax:

Total:

Payment Options

[ ] MasterCard [ ] Visa [ ] American Express

[ ] Check/Money Order (make payable to SOFT SECTOR, INC.)

If Credit Card...

Credit Card #: _____

Expiration Date: ____/____

Name on Card: _____

Signature: _____

Registration Benefits

For any of our products that you register, you get the following:

- Current version usable in any of your applications
- Technical support via e-mail for one year
- Bug fixes free via e-mail
- Next major update free
- Notification of updates of all our products
- Discounts on our other products

Using ssBarcode with ReportPrinter Pro

Using ssBarcode with ReportPrinter is very different than using it with QuickReports. If you installed RPSSBC.DCU, you will have a component on your palette named RPssBarcode. RPssBarcode is a descendant of ssBarcode, so it has all of the properties and methods described above. You can set properties the same way you would with just a basic ssBar code. To use an RPssBarcode, you do NOT need an additional ssBarcode as you would with QuickReports.

To print barcodes with ReportPrinter you must first place a RPSSBarCode component on the same form (or a form accessible by) as a ReportPrinter or ReportSystem component. Set the barcode desired using the object inspector or in the source code. To print the bar code, use the following example:

```
procedure TForm1.ReportSystem1Print(Sender: TObject);
begin
   with Sender as TBaseReport do
   begin
      { Print other items on page }

         RPSSBarCode1.PrintBarCode(Sender,2.0,2.0,0.0);

      { Print other items on page }
   end; { with }
end;
```

The first parameter of PrintBarCode is the TBaseReport object which is usually accessible as the Sender parameter of the printing events. The second and third parameters are the X and Y values (in inches or metric) for the upper left corner of the barcode. The last parameter is the height of the barcode (in inches or metric) or pass 0.0 to use the default height for the barcode.

NOTES:

- Barcodes printed with TRPSSBarCode will not suffer any degradation and will appear exactly as if it was printed using the TSSBarCode component.
- As of version 2.21, RPssBarcode **will work** with ReportPrinter's print preview

We would like to thank Jim Gunkel and the rest of the staff at Nevrona Designs for their help in making our products compatible.

For more information on ReportPrinter Pro:

Nevrona Designs
1930 S. Alma School Rd. #C-204
Mesa, AZ 85210
UNITED STATES

Voice: (602) 491-5492
Fax : (602) 530-4823
Email: info@nevrona.com
CIS : 70711,2020
WWW : http://www.nevrona.com/designs

Using ssBarcode with QuickReports v2.0

We have done the necessary work to make ssBarcode completely compatible with the newest version of the very popular report writer QuickReports. If you are unfamiliar with QR, the following may not make much sense to you.

You will remember that you actually installed two components, ssBarcode and QR20ssBarcode. BOTH components are necessary to make your program work with both QR and ssBarcode. QR20ssBarcode is what some call a "wrapper" component. It needs to be linked to an actual instance of ssBarcode, either at design time or run time. QR20ssBarcode doesn't have any properties for changing the barcode parameters. It simply has a property called "Barcode" that you link to an instance of ssBarcode that actually does the work.

In other words, you create an ssBarcode that **controls** the QR20ssBarcode. Whatever changes you make to the ssBarcode will be reflected in the QR20ssBarcode. I know this sounds a little confusing, but it really is rather simple once you play with it.

Why did we do it this way? We were having problems with people using different versions of QuickReports than the one that we compiled with. So we've included the full source code to QR20ssBarcode, which means it will compile no matter what version of QR you are using.

NOTE ABOUT BARCODE QUALITY: There was a problem with barcode quality when using QuickReports v1.0 and ssBarcode. This was because QuickReports forced third-party components to paint themselves for the screen, then scaled it up for the printer. This sometimes resulted in somewhat blurred barcodes. However, with QuickReports v2.0, this is no longer the case. Components paint themselves for the canvas of the printer, so they will be as high quality as if you used the PrintBarcode method.

For more info on QuickReports, see their Web page at http://www.qusoft.no

Using ssBarcode WITHOUT the BDE

It is now possible to use ssBarcode in applications that do not require the BDE. In order to do so, you MUST own the source code version of ssBarcode. There is a compiler directive that we have checked for called "XBDE". If you define this compiler directive and then rebuild your VCL, all the references to the BDE will be removed from ssBarcode. This can be done by typing the following line at the very top of the SSBC.PAS source file:

```
{$DEFINE XBDE}
```

Note that QuickReports also uses the XBDE compiler directive to indicate that all references to the BDE should be removed. We have purposely made these two the same so that you only need to define the directive in one place if you are using both products without the BDE.