ssBarCode version 1.5

The Delphi Native Bar code Component - 16-bit version

Soft Sector, Inc.
PO Box 81480
Cleveland OH 44181

See the document ORDER.TXT for ordering information.

See the end of this document for the new features in this version.

Introduction

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 first version, ssBarCode supports the four most popular bar code symbologies: Code 39 (or 3 of 9), Interleaved 2 of 5, Postnet, and UPC version A. In the near future we will be implementing many more symbologies. Please see the section of this file entitled "Future Enhancements."

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: ssi@en.com
Phone/fax: (216) 777-3612

Keep an eye out for our Web page, coming soon!

Installation

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

1. Place all files that were contained in the .ZIP file into a directory (example: \SSBC)

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 where you put the ssBarCode files. In the "List Files of Type" drop down, you should select "DCU" files.

5. You will see "SSBC.DCU" appear in the files list box. Select it and click OK.

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.

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

Note that 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 the file ORDER.TXT for more information on this extremely useful, yet affordable product.

Demo Program

We've written a simple demo program to show off some of the features of ssBarcode, including the different symbologies, bar width specification, and barcode printing. The project name is DEMOBC.DPR. All of the features of this demo program are very easy to understand, and some of the code is commented.

Properties

AutoSize
BarCodeType
BarColor
BarWidth
BearerBars
CalcCheckDigit
Canvas (*)
Color (*)
Data
Orientation
PrintHumanReadable
Visible (*)
WideBarRatio

(*) These are inherited properties that are not described in this document. See the Delphi help file for details.

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.

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:

{bcCode39, bcInt2of5, bcPostnet, bcUPC_A}

See the section below entitled “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 inches, of each bar. 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 39 = 0.010 inches
Interleaved 2 of 5 = 0.015 inches
Postnet = 0.020 ** (We strongly recommend using this value for Postnet)
UPC A = 0.013 inches

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 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 section below entitled "Symbologies" for a description of acceptable data for each type of bar code.

Orientation property

Declaration
property Orientation : TOrientation

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 TOrientation 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 should always have this set to True. As for the other bar codes, it is completely up to you.

Note that, in this version, all human readable text is printed using the Courier font.

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.

Methods

There is only one available method, aside from the constructor for this component. The constructor's implementation is as follows:

constructor Create(AOwner : TComponent);

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.

To use this method, supply X and Y values for the upper left hand corner of the barcode, in inches. The Height parameter allows you to specify an exact height of the barcode, in inches. 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.Canvas.Font := 'Courier New'; {****}
  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.

Events

The events defined for ssBarCode are inherited from standard Delphi events. See Delphi's help file for details.

OnClick
OnDblClick
OnMouseDown
OnMouseMove
OnMouseUp

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 four major symbologies, as described below.

Code 39 (or 3 of 9)

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, this is the code to choose.

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 version 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

This is version 1.0 of ssBarCode. As such, we have many ideas and features that we will be adding the near future. If you have any comments or suggestions, feel free to let us know. The rate at which we add these new features will be directly proportional to the amount of registrations we receive (hint, hint :))

- Additional symbologies: UPC-E, EAN-8, EAN-13, Codabar, Code 128, FIM, Bookland, UPC add on codes
- Data aware: allowing you to assign a Datasource and Datafield to the bar code component
- Auto detecting symbologies based on the data to be encoded
- Allowing the component user to set the font to use for human readable text
- Windows .HLP file

Revision History

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