Contents

BarCode/VBX is a custom control that is used in Visual Basic and Visual C++ to display and print barcodes. Several types of barcodes are supported, some of which allow several variations. BarCode/VBX is a bound control allowing you to bind the control to a data field. You have control of how the bar code is displayed and printed through the use of various properties, events, and functions.

Properties
Events
Functions

Events

The following events are generated by BarCode/VBX. You may choose to respond to these events in your code to provide increased functionality.

Changed
Resized
Click
DblClick
DragDrop
DragOver
LinkClose
LinkError
LinkNotify
LinkOpen
MouseDown
MouseMove
MouseUp

Properties

The following properties are available for your use. By setting these properties, you have complete control over how the barcode is displayed and printed.

BarCodeType
Checksum
Rotation
Text
NarrowBarWidth
Ratio
ChecksumString
PrinterTop
PrinterLeft
PrinterHeight
PrinterBarWidth
PrinterScaleMode
PrinterHDC

Functions

The following functions are available for use with BarCode/VBX. They are defined in the 'barcode.bas' file.

BarCodePrint
BarCodeGetLastErrorCode
BarCodeGetLastErrorString
BarCodeGetLastErrorStringC

Changed

This event is fired when the text property is changed.

Resized

This event is fired when the size of the barcode is changed. BarCode/VBX will automatically resize the conatiner in order to display the entire barcode. This event can be useful if you need to reposition other objects on your form based on the size of the barcode.

Click

This event is fired when the user clicks the left mouse button. This is a standard event, see the Visual Basic users manual for more information.

DblClick

This event is fired when the user double clicks the left mouse button. This is a standard event, see the Visual Basic users manual for more information.

DragDrop

This event is fired when the control is the target of a drop operation. This is a standard event, see the Visual Basic users manual for more information.

DragOver

This event is fired when the control is the target of a drop operation. This is a standard event, see the Visual Basic users manual for more information.

LinkClose

This event is fired when a DDE conversation is terminated. This is a standard event, see the Visual Basic users manual for more information.

LinkOpen

This event is fired when a DDE conversation is initiated. This is a standard event, see the Visual Basic users manual for more information.

LinkError

This event is fired when a DDE error occurs. This is a standard event, see the Visual Basic users manual for more information.

LinkNotify

This event is fired when a DDE conversation has been initiated and the data in the server has changed. This is a standard event, see the Visual Basic users manual for more information.

MouseDown

Fired when a mouse button is pressed. The mouse is captured as a result of this event. This is a standard event, see the Visual Basic users manual for more information.

MouseMove

Fired when mouse movement occurs. This is a standard event, see the Visual Basic users manual for more information.

MouseUp

Fired when a mouse button is released. The mouse capture is released as a result of this event. This is a standard event, see the Visual Basic users manual for more information.

BarCodeType Property

Description

Sets the type of barcode used.

Usage

[*form*.]*control*.BarCodeType[= *setting*]

Remarks

By setting this property, you select the type of barcode that is displayed or printed. the following is a list of the possible types of barcodes.

Setting	Description
0	Code 3 of 9
1	Extended Code 3 of 9
2	Interleaved 2 of 5
3	Code 93
4	Extended Code 93
5	UPCA
6	UPCE 10 digit
7	UPCE0 6 digit
8	UPCE1 6 digit
9	EAN 13
10	EAN 8
11	Code 128 Auto
12	Code 128 A
13	Code 128 B
14	Code 128 C
15	Codabar
16	MSI Plessey
17	UCC-128
18	POSTNET (Zip + 4 PostalCode)

Data Type

Integer

Checksum Property

Description

Controls how the checksum is created

Usage

[*form*.]*control*.Checksum[= *setting*]

Remarks

Checksums can be optionally added to some barcodes. See the description on the particular bar code type for more information.

Setting	Description
0	(Default) No checksum.
1	See the appropriate barcode type for more information.
2	See the appropriate barcode type for more information.

Data Type

Integer

Rotation Property

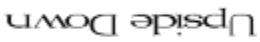
Description
Controls the rotation of the barcode.

Usage
[*form*.]*control*.Rotation[= *setting*]

Remarks
The bar code can be rotated by setting this property to the proper value. The following list shows the different orientations.

Setting	Description
0	Normal
1	Right
2	Upside Down
3	Left

Normal



Data Type
Integer

Text Property

Description

Sets the text to be used in creating the bar code.

Usage

[*form*.]*control*.Text[= *stringexpression*]

Remarks

The text property allows you to set the text that will be used to generate the bar code itself. If you are in design mode, changing this property will cause the BarCode/VBX to draw the barcode, if the text entered is valid for the bar code type that is currently selected.

Data Type

String

NarrowBarWidth Property

Description

Sets the width, in pixels, of the narrow bar.

Usage

[*form*.]*control*.Rotation[= *numericexpression*]

Remarks

This property controls the width of the narrow bars, in dots. BarCode/VBX defaults to 2 dots, which means that the narrow bars in each bar code will be two dots wide. Use this property to change the width of the narrow bars. Valid ranges are from 1 thru 6.

Data Type

Integer

Ratio Property

Description
Sets the ratio of the barcode.

Usage
[*form*.]*control*.Ratio[= *setting*]

Remarks
The ratio of the wide bars to narrow bars can be controlled using this property. BarCode/VBX defaults to a 3:1 ratio. Valid selections for this property are listed below.

Setting	Description
0	3:1
1	2.5:1
2	2:1

Data Type
Integer

ChecksumString Property

Description
Contains the cheskum string

Usage
[*stringexpression* =][*form*.]*control*.Checksum

Remarks
This property is a string that contains the value of the checksum character. This property is read-only at run time and not available at design time.

Data Type
String

PrinterTop Property

Description

Sets the top position of the barcode when printing.

Usage

[*form*.]*control*.PrinterTop[= *numericexpression*]

Remarks

The PrinterTop, PrinterLeft, PrinterHeight, PrinterBarWidth, PrinterScaleMode and PrinterHDC properties are used to print a barcode on the printer. The PrinterTop property allows you to specify the top position of the bar code.

Data Type

Single

PrinterLeft Property

Description

Sets the left position of the barcode when printing.

Usage

[*form*.]*control*.PrinterLeft[= *numericexpression*]

Remarks

The PrinterTop, PrinterLeft, PrinterHeight, PrinterBarWidth, PrinterScaleMode and PrinterHDC properties are used to print a barcode on the printer. The PrinterLeft property allows you to specify the left position of the bar code.

Data Type

Single

PrinterHeight Property

Description

Sets the height of the barcode when printing.

Usage

[*form*.]*control*.PrinterHeight[= *numericexpression*]

Remarks

The PrinterTop, PrinterLeft, PrinterHeight, PrinterBarWidth, PrinterScaleMode and PrinterHDC properties are used to print a barcode on the printer. The PrinterHeight property allows you to specify the height of the bar code.

Data Type

Single

PrinterBarWidth Property

Description

Sets the narrow bar width of the barcode when printing.

Usage

[*form*.]*control*.PrinterBarWidth[= *numericexpression*]

Remarks

The PrinterTop, PrinterLeft, PrinterHeight, PrinterBarWidth, PrinterScaleMode and PrinterHDC properties are used to print a barcode on the printer. The PrinterBarWidth property allows you to specify the width of the narrow bar, in pixels, of the bar code.

Data Type

Integer

PrinterScaleMode

Description

Determines the unit of measurement for sizing and positioning the barcode on the printer.

Usage

[*form*.]*control*.PrinterScaleMode[= *setting*]

Remarks

The PrinterTop, PrinterLeft, PrinterHeight, PrinterBarWidth, PrinterScaleMode and PrinterHDC properties are used to print a barcode on the printer. The PrinterScaleMode property allows you to specify the unit of measurement that will be used when sizing and positioning the barcode on the printer.

Setting	Description
0	(Default) Pixel (smallest unit of printer resolution).
1	Twip (1440 twips per logical inch; 567 twips per logical centimeter).
2	Point (72 points per logical inch).
3	Pixel (smallest unit of printer resolution).
4	Character (horizontal = 120 twips per unit; vertical = 240 twips per unit).
5	Inch.
6	Millimeter.
7	Centimeter.

Data Type

Integer

PrinterHDC Property

Description

Sets the handle to the printer device context.

Usage

[*form*.]*control*.PrinterHDC[= **hDC**]

Remarks

The PrinterTop, PrinterLeft, PrinterHeight, PrinterBarWidth, PrinterScaleMode and PrinterHDC properties are used to print a barcode on the printer. The PrinterHDC property allows you to specify the handle of the device context for the printer. When this property is set, the barcode will be printed on the printer.

Data Type

Integer

Example:

```
BarCode1.PrinterTop = 100
BarCode1.PrinterLeft = 0
BarCode1.PrinterHeight = 100
BarCode1.PrinterBarWidth = 4
BarCode1.PrinterScaleMode = 5
BarCode1.PrinterHDC = Printer.hDC
If BarCodeGetLastErrorCode() <> False Then
    MsgBox BarCodeGetLastErrorString()
End If
```

Code3 of 9

This bar code is an alphanumeric bar code allowing uppercase letters and numbers. BarCode/VBX will convert any lower case letters into upper case before printing the bar code. Each character consists of nine elements. 3 of the nine elements are wide, hence the name '3 of 9'.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Extended 3 of 9

Extended 3 of nine is similar to Code 3 of 9 except that it allows the full 128 ASCII character set to be encoded by printing two bar code characters for each text character.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Interleaved 2 of 5

This is strictly a numeric bar code. Each encoded character is made up of five elements, two are wide and three are narrow. The number of characters to be printed must be an even number. If the number of characters to be printed is odd a zero will be appended to the beginning of the code.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Code 93

Code 93 is an alpha-numeric bar code allowing upper case letters and numbers. BarCode/VBX will convert lower case letters to upper case before encoding them.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Extended Code 93

Extended Code 93 is similar to Code 93 except that it allows the full 128 character ASCII character set to be encoded.

Set the checksum property to a non-zero value to add a checksum to the barcode.

UPCA

UPC (Universal Product Code) version A is used to encode an 11 digit number. The first digit is the system number and the rest are data characters. Both 2 and 5 digit supplementals are also supported.

The value of the checksum property is not used.

UPCE 10 digit

UPCE is a zero suppressed version of the UPCA barcode. This version allows 10 digits to be encoded. The first digit must be zero. Both 2 and 5 digit supplementals are also supported.

The value of the checksum property is not used.

UPCE0 6 digit

UPCE is a zero suppressed version of the UPCA barcode. This version allows 6 digits to be encoded. The first digit must be zero. Both 2 and 5 digit supplementals are also supported.

The value of the checksum property is not used.

UPCE1 6 digit

UPCE is a zero suppressed version of the UPCA barcode. This version allows 6 digits to be encoded. The first digit must be zero. Both 2 and 5 digit supplementals are also supported.

The value of the checksum property is not used.

EAN 13

EAN barcodes are used when the country origin needs to be known. There are 13 digits in EAN 13 where the first two characters are used to define the country of origin,. the next ten are data, followed by the checksum. Both 2 and 5 digit supplementals are also supported.

The value of the checksum property is not used.

EAN 8

EAN barcodes are used when the country origin needs to be known. There are 8 digits in EAN 8 where the first two characters are used to define the country of origin,. the next 5 are data, followed by the checksum. Both 2 and 5 digit supplementals are also supported.

The value of the checksum property is not used.

Code 128 Auto

Code 128 is a variable length bar code that is capable of encoding the entire 128 character ASCII character set. Code 128 allowsthree subsets, A, B and C. This version, 'Code 128 Auto', will automatically select the subset that will produce the smallest bar code.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Code 128 A

Code 128 is a variable length bar code that is capable of encoding the entire 128 character ASCII character set. Code 128 allowsthree subsets, A, B and C. This subset (A) allows all standard upper case alpha-numeric keyboard characters plus control characters.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Code 128 B

Code 128 is a variable length bar code that is capable of encoding the entire 128 character ASCII character set. Code 128 allowsthree subsets, A, B and C. This subset (B) allows all standard upper case alpha-numeric keyboard characters and lower case alpha characters.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Code 128 C

Code 128 is a variable length bar code that is capable of encoding the entire 128 character ASCII character set. Code 128 allowsthree subsets, A, B and C. This subset (C) includes a set of 100 digit pairs from 00 to 99 inclusive. This allows double density numeric digits, two digits per bar coded character.

Set the checksum property to a non-zero value to add a checksum to the barcode.

Codabar

Codabar is a variable length barcode that can encode 16 data characters including 0-9, plus the symbols - $; / . +. Codabar is used primarily for numeric data.

Set the checksum property to a non-zero value to add a checksum to the barcode.

MSI Plessey

This barcode is a variable length barcode that can encode up to 15 numeric digits. Checksum generation is dependent on the value of the checksum parameter. The following table indicates the value of the checksum property and the type of checksum created.

Setting	Description
0	one modulus 10 checksum
1	two modulus 10 checksums
2	one modulus 11 checksum/one modulus 10 checksum

UCC-128

This bar code is a specially defined subset of Code 128 that is used mostly on shipping containers. It is numeric only having a fixed length of 19 digits.

Set the checksum property to a non-zero value to add a checksum to the barcode.

POSTNET (Zip + 4 Postal Code)

The POSTNET barcode is used on envelopes and postcards that are sent through the U.S. Postal Service. This barcode is placed in the lower right hand corner of the envelope.

The Checksum, Rotation, and NarrowBarWidth properties have no effect on this bar code.

BarCodePrint Function

This function allows you to print a bar code on a device context that you have created before calling this function. Simply obtain a device context for the device that you wish to print on and fill in the members of the BARCODEPRINTDATA structure. The return value of this function is a boolean value that indicates if the function was successful. If the return value is 'False', the function failed. Use the BarCodeGetLastErrorCode or the BarCodeGetLastErrorString functions to retrieve the last error.

This function is declared in the 'BARCODE.BAS' file as:

```
Declare Function   Lib "BARCODE.VBX" (lpBarCodeData As BARCODEPRINTDATA) As Integer
```

See also: PrinterTop, PrinterLeft, PrinterHeight, PrinterBarWidth, and the PrinterHDC properties to print the barcode by setting properties rather than using this function call.

BARCODEPRINTDATA Structure

In Visual Basic this structure is defined as:

```
Type BARCODEPRINTDATA
        BarCodeType As Integer
        Checksum As Integer
        Text As String * 30
        Rotation As Integer
        NarrowBarWidth As Integer
        Ratio As Integer
        hDC As Integer
        x As Integer
        y As Integer
        Height As Integer
End Type
```

In Visual C++ the structure is as follows:

```
typedef struct tagPrintBarCode {
            int BarCodeType;
            int Checksum;
            Byte Text[30];
            int Rotation;
            int NarrowBarWidth;
            int Ratio;
            int hDCPrinter;
            int x;
            int y;
            int Height;
} *LPBARCODEDATA;
```

BarCodeGetLastErrorCode Function

This function allows you to retrieve the last error code. This function can be used in Visual Basic or Visual C++.

In Visual Basic this function is defined as:

```
Declare Function GetLastErrorCode Lib "BARCODE.VBX" () As Integer
```

int BarCodeGetLastErrorCode(void);

This function returns the last error code.

In Visual C++ this function can be called by using the 'Load Library', 'GetProcAddress', and 'FreeLibrary' functions. The following is an example:

```
HINSTANCE hVBXInst;
int TheLastErrorCode;
int (FAR PASCAL *GetErrorCode) (void) = NULL;

hVBXInst = LoadLibrary("BARCODE.VBX");
GetErrorCode = (int (__far __pascal *)(void))GetProcAddress(hVBXInst,"BarCodeGetLastErrorCode");
if(GetErrorCode != NULL)
    TheLastErrorCode = (*GetErrorCode)();
FreeLibrary(hVBXInst);
```

BarCodeGetLastErrorString Function

This function will return a string that describes the last error. This function must not be used in Visual C++ because this function returns a Visual Basic String. See the 'BarCodeGetLastErrorStringC' function.

```
Declare Function GetLastErrorString Lib "BARCODE.VBX" () As String
```

BarCodeGetLastErrorStringC Function

LPSTR BarCodeGetLastErrorStringC(LPSTR Buffer, int BufferLength);

LPSTR *Buffer*; /* pointer to buffer that will hold string */
int *BufferLength*; /* length of buffer */

This function retrieves the last error string into a buffer. In Visual C++ this function can be called by using the 'Load Library', 'GetProcAddress', and 'FreeLibrary' functions. The following is an example:

```
HINSTANCE hVBXInst;
char Str[81] = "";
LPSTR (FAR PASCAL *GetErrorStr) (LPSTR, int) = NULL;

GetErrorStr = (char __far *(__far __pascal *)(LPSTR,
int))GetProcAddress(hVBXInst,"BarCodeGetLastErrorStringC");

if(GetLastErrorString != NULL)
    (*GetErrorStr)((LPSTR)Str, 81);
FreeLibrary(hVBXInst);
```