

VisionShape BarCode ActiveX Control

User's Guide
ImageBASIC 3.1

IMAGE  *BASIC*

Diamond Head Software, Inc.
1217 Digital Drive Ste. 125
Richardson, Texas 75081-1970
(972) 479-9205

Copyright Notices

Under the copyright laws, neither the documentation nor the software may be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of Diamond Head Software, Inc., except in the manner described in the documentation.

This software product contains proprietary software components developed by a number of different software companies, referred herein as "Third Party Licensors". This documentation and the software that you purchased are protected by one or more of the following copyright notices:

Portions of this product, ©1993 - 1996 Diamond Head Software, Inc. All rights reserved.

Portions of this product, ©1990 - 1996 VisionShape Inc. All rights reserved.

Company and product names mentioned in this documentation are trademarks or registered trademarks of their respective companies. Lotus and Lotus Notes are registered trademarks of Lotus Development Corporation. Windows is a trademark and Microsoft is a registered trademark of Microsoft Corporation.

DIAMOND HEAD SOFTWARE INC. AND ITS THIRD PARTY LICENSORS MAKE NO WARRANTIES, EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, REGARDING THE SOFTWARE. DIAMOND HEAD SOFTWARE, INC. AND ITS THIRD PARTY LICENSORS DO NOT WARRANT, GUARANTEE OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE SOFTWARE IN TERMS OF ITS CORRECTNESS, ACCURACY, RELIABILITY, CURRENTNESS OR OTHERWISE. THE ENTIRE RISK AS TO THE RESULTS AND PERFORMANCE OF THE SOFTWARE IS ASSUMED BY YOU. THE EXCLUSION OF IMPLIED WARRANTIES IS NOT PERMITTED BY SOME JURISDICTIONS. THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

IN NO EVENT WILL DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS AND/OR THEIR DIRECTORS, OFFICERS, EMPLOYEES OR AGENTS BE LIABLE TO YOU FOR ANY CONSEQUENTIAL, INCIDENTAL OR INDIRECT DAMAGES (INCLUDING DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, AND THE LIKE) ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE EVEN IF DIAMOND HEAD SOFTWARE INC. OR ITS THIRD PARTY LICENSORS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU. Diamond Head Software Inc.'s and its Third Party Licensors' liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort (including negligence), product liability or otherwise), will be limited to \$50.

Contents

| | |
|--|----|
| Chapter 1: Introduction | 1 |
| Linking of ImageBASIC Controls | 1 |
| Licensing Verification..... | 1 |
| Chapter 2: Using the Barcode Control | 3 |
| Introduction to the VisionShape Barcode Control..... | 3 |
| Performing Recognition | 3 |
| Improving Recognition Results..... | 6 |
| Chapter 3: VisionShape Barcode Reference | 11 |
| Properties, Events and Methods..... | 11 |
| Appendix A: Error Codes | 19 |
| Possible VisionShape Barcode Errors..... | 19 |
| Appendix B: Barcode Types | 21 |
| Supported Barcodes | 21 |
| Index | 25 |

Chapter 1: Introduction

Linking of ImageBASIC Controls

Virtually all ImageBASIC controls can accept image data from other ImageBASIC controls. The process of designating where each ImageBASIC component gets its image data is referred to as linking of controls. With the exception of those controls that can directly access files or scanners, all ImageBASIC controls must be linked to another ImageBASIC control to receive any image data.

Linking of controls is the primary method of moving an image through a series of processing steps. For example, an image may be originally captured through the PixScan ActiveX, passed to a TMS Display control for operator verification, optionally routed through the ScanFix control for enhancement, then to a TMSFile control to be written to disk, and finally to a VisionShape Barcode control for barcode recognition processing to generate indexing information for that file.

Creating the Link

All of the ImageBASIC controls have a property named **ImageDataSource**. To create the data link between controls, this property is set to the Name of the source ImageBASIC control. Any image that is received by the named control will also be sent to the linked control. For example, if the **ImageDataSource** property of a TMS Display control is set to the Name of a PixScan control, each time a new image is scanned, the display control will receive that image.

Linking each control requires only one line of code that can be executed at any time. The source of image data can be changed during program execution by naming another ImageBASIC control in the **ImageDataSource** property, as shown here:

```
VSBarcode1.ImageDataSource = TMSDispl.Link
```

Licensing Verification

In order for any application using the VisionShape Barcode ActiveX control to function, it must be able to verify that the control is licensed. This verification usually takes the form of a search for a license in a database or on a hardware key. In all cases, however, the developer is given some control over when the control requests licensing verification.

Chapter 2: Using the Barcode Control

Introduction to the VisionShape Barcode Control

The VisionShape Barcode control provides you with several options for recognizing barcodes. With ImageBASIC 3.1, you are given the option of recognizing just the first barcode on an image or all of the barcodes which appear on the image. If the TMS Display component is used, working regions can define barcodes for recognition. Additionally, if a display component is not in the application, the recognition output can be saved to a text file.

Performing Recognition

Before performing recognition, it is helpful to know what type of barcodes you are trying to identify. For a list of supported barcodes, see Appendix B. Once you have determined the type of barcode you are using, set the **BarCodeType** property to the appropriate setting. However, if you can not specifically identify the barcode, you can set **BarCodeType** to *Auto Detect*. The auto detect setting will slow the recognition engine down, but is a convenient setting for when multiple types of barcodes are used or the barcode type can not be identified.

Recognizing Barcodes

The barcode engine allows you to perform recognition on barcodes without setting working regions. There are two options when performing recognition in this manner: you can recognize just the first barcode which appears on the image or you can recognize all of the barcodes on the image. The **ReadMultiple** property enables and disables this functionality.

The following text outlines the basic steps needed to perform recognition without specifying working regions in the display control:

1. Set the **ImageDataSource** to the display control from which you want to recognize the barcode:

```
VSBarcode1.ImageDataSource = TMSDisp1
```
2. Set any recognition options. See the following section 'Improving Recognition Results' on page 6 for more information on these options.

3. Set the **ReadMultiple** property to read either all the barcode present on the image or just the first one. To read just the first barcode on the image. Set **ReadMultiple** to False:

```
VSBarcode1.ReadMultiple = False
```

To read all the barcodes on the image set the **ReadMultiple** property to True. When retrieving the results from a multiple barcode read, the **Result** property will display the information with a pipe separating the recognition output for each of the barcodes.

4. Set the **Recognize** method. This will trigger the VisionShape Barcode engine recognition attempt:

```
VSBarcode1.Recognize
```

The results of the recognition attempt can be found in the **Result** property. This property must be queried in order to retrieve the results. See the section "Retrieving Recognition Output" on page 5 for more details.

When recognizing the entire page with multiple barcodes (**ReadMultiple** property set to True), you may receive better results if you increase the passes the recognition engine makes over the image. This can be accomplished through the **SearchMode** property. For more information on this property, refer to the section "Improving Recognition Results" on page 6.

Recognizing Barcodes Set in Working Regions

One of the most convenient ways to read barcodes in an ImageBASIC application is to set the barcodes in working regions. Working regions are active regions set in the display window, in this case TMS Display control. This control allows users to specify which regions to recognize with the VisionShape Barcode engine. Users simply draw a box around the barcode that they need to recognize and the recognition engine will catch those areas. However, if you are designing an application in which barcodes will appear in the same place on each image, you can design TMS Display control to automatically draw working regions around the barcodes. This method speeds up production time and reduces operator error.

Whichever method you decide to use, if you use working regions, the code for the VisionShape Barcode control will be the same. You can learn more about setting working regions in the *TMS Display's User Guide*.

In order for the VisionShape Barcode control to read working regions in the display control:

1. Set the **ImageDataSource** to the display control from which you want to recognize the barcode:

```
VSBarcode1.ImageDataSource = TMSDispl
```


2. Set any recognition options. See the section **Improving Recognition Results** on page 6 for more information.
3. Set the **ReadMultiple** property to read either all the barcode present on the image or just the first one. To read just the first barcode on the image. Set **ReadMultiple** to False:

```
VSBarcode1.ReadMultiple = False
```

To read all the barcodes on the image set the **ReadMultiple** property to True. When retrieving the results from a multiple barcode read, the **Result** property will display the information with a pipe separating the recognition output for each of the barcodes.

Note: The 32 bit VisionShape Barcode ActiveX does not support multiple working regions at this time. You can work around this situation by not using working regions and setting the property **ReadMultiple** to True.

5. In order for the barcode engine to read working regions from the display, you must set the **RegionSource** Property to the name of the display control that you will be setting working regions:
6. Set the **Recognize** method. This will trigger the VisionShape Barcode engine recognition attempt:

```
VSBarcode1.Recognize
```

The results of the recognition attempt can be found in the **Result** property. This property must be queried in order to retrieve the results. See the following section "Retrieving Recognition Output" for more details.

Retrieving Recognition Output

The results of the recognition attempt can be found in the **Result** property. This property must be queried in order to retrieve the results:

```
Text1.Text = VSBarcode1.Result
```

If the recognition attempt fails a tilde (~) will be returned. There are several factors which may be involved for the failure of a recognition attempt. For more information on improving recognition results, please refer to the section "Improving Recognition Results" on page 6.

Sending Recognition Output to a File

Your barcode application can be written to send recognition data directly to a file. This process can greatly speed up the recognition process if no display control is used in the application.

In the following example, the VisionShape Barcode ActiveX will receive a file directly from the TMS File control, perform recognition and send the information to an output file.

1. Set the VisionShape Barcode's **ImageDataSource** to TMSFile so that the image can be passed to it:

```
VSBarcode1.ImageDataSource = TMSFile1
```

2. Set the TMSFile control's **InputFileName** property to the name of a barcode file:

```
TMSFile1.InputFileName = "c:\info\barcod1.tif"
```

3. Indicate whether you would like the control to read multiple barcodes. In this case, VisionShape will read multiple:

```
VSBarcode1.ReadMultiple = True
```

4. Set any recognition options which will improve accuracy. In this case, the **SearchMode** property will be set to 3 to make the recognition engine make 99 passes at recognition.

```
VSBarcode1.SearchMode = 3
```

To learn more about improving recognition accuracy, please refer to the section, "Improving Recognition Results" on page 6.

5. Set the recognition engine to read the image:

```
VSBarcode1.Recognize
```

6. Save the recognition results out to a file:

```
Open "c:\images\result.txt" for append as #1  
Print #1, VSBarcode1.Result  
Close#1
```

This basic coding routine will save the recognition results in the specified text file.

Improving Recognition Results

If you are having difficulty with the recognition process, there are several properties within the VisionShape Barcode ActiveX that you can use to improve accuracy. Finding the correct setting for each of these properties takes some time and varies with each barcode. If none of these properties seem to be working, recheck the type of barcode you are trying to recognize for compatibility with

VisionShape. A list of these barcodes is provided in Appendix B. If the barcode is supported and VisionShape is still not recognizing it, you may need to perform some clean-up techniques to sharpen the image. VisionShape can read barcodes from many different angles, but sometimes, especially when barcodes are extremely wide, the recognition engine may have problems interpreting them. In these cases, an image enhancing engine would become most helpful.

Beam Width

For wider barcodes, VisionShape offers the **BeamWidth** property which can be adjusted to provide a more accurate reading. The default value of the barcode recognition beam may be too small to read the entire barcode. This property adds additional width to the reader beam in 1/100th inch segments. A wider beam will be slower but more accurate. However, if the **BeamWidth** property is set too high, nearby text in the image may interfere with barcode readings.

If the recognition engine is having difficulty recognizing a barcode, calling for a recognition attempt in a loop, and incrementing the **BeamWidth** on each pass, will often solve the difficulty. The following code segment provides an example of the necessary settings to use for this technique. Setting the upper limit to the value of the **BeamWidth** is necessary to prevent an infinite loop in case the barcode is unreadable. The final If/Then statement will inform the user if the barcode is unrecognizable and proceeds to exit the subroutine:

```
Sub StartRecognition_Click ()
    Dim BarValue$

    Do
        BarValue$ = VSBarcode1.Result
        VSBarcode1.BeamWidth = VSBarcode1.BeamWidth + 1
        Loop Until BarValue$ <> "~" or (VSBarcode1.BeamWidth
            >=50)

        If VSBarcode1.Result = "~" Then
            Text1.Text = "Unable to read barcode"
        End If

        VSBarcode1.BeamWidth = 0

    End Sub
```

Quiet Zones

The quietzone is the white space just before and after the barcode. Barcode standards generally require a quiet zone of 10 times the smallest bar used in the barcode or 1/10 of an inch. However, most users have found that a quiet zone 1/4 of an inch tends to work best. If the VisionShape Barcode engine continually returns a tilde (~) upon recognition you might need to adjust the **QuietZone** property for readings.

CheckDigits

The **CheckDigit** property reads the checksum number encoded into the barcode. If this property is set to true, the barcode engine runs a comparison of its recognition output with the checkdigit. If the checksum does not match, the barcode will be reported as unreadable and the output will report a tilde (~). Because not all barcode types include a checkdigit, the recognition output will not be affected if the barcode does not contain one.

Searching Modes

The **SearchMode** property is available to users who are reading multiple barcodes on an image. The value of this property dictates to the recognition engine how to search for barcodes on the image. The choice of a search pattern should be based on the size of the image passed to the recognition engine.

Smaller regions usually require fewer search passes. Large image regions require more search passes to ensure that any barcodes present will not be missed. The values of this property follow:

- 0--Adaptive Search
- 1--3 Horizontal Scans
- 2--33 Horizontal Scans
- 3--99 Horizontal Scans
- 4--99 Horizontal and Vertical Scans
- 5--3 Vertical Scans
- 6--33 Vertical Scans
- 7--99 Vertical Scans
- 8--200 Horizontal and Vertical Scans

The number of steps shown above indicates the number of sections into which the image data passed to the VisionShape engine is divided. The engine then searches along each dividing line for evidence of a barcode. As you might expect, more search passes generally result in a higher recognition rate, but require more time.

In general, if you are passing small regions of image data to the recognition engine, fewer search passes are necessary. However, if an entire page is being searched by VisionShape, more passes are likely to be necessary.

Chapter 3: VisionShape Barcode Reference

Properties, Events and Methods

Active Property

| | |
|------------------------|--|
| Definition: | <p>If set to True at design time, the control will fully initialize and verify licensing immediately upon initialization of the runtime application.</p> <p>If set to False at design time, full initialization of the control will be delayed at initialization of the runtime application. In this case, this property must be explicitly set to True at runtime before the control is used.</p> |
| Data Type: | Boolean |
| Design Access: | Read/Write |
| Runtime Access: | Read/Write (see limits below) |
| Comments: | <p>If this property is set to True (the default) at design time, the control is fully initialized and licensing is verified immediately upon initialization of the application at runtime. The related technology libraries are loaded and the control is ready to be used.</p> <p>If this property is set to False at design time, the control will only partially initialize when the application loads at runtime. By delaying these two actions, the application should be able to load more quickly:</p> <ol style="list-style-type: none">1) The related technology libraries for the control will not be loaded.2) The licensing server will not verify an available token for the control. <p>If the control initializes with Active set to False, this property must be explicitly set to True by the application. Until Active is set to True, the control will ignore all instructions to it.</p> <p>If the control fails to find a license token, the Active property will be automatically set to False. The application can check this value on Form Load to determine if each control is licensed and can be used.</p> |

AboutBox Method

| | |
|----------------------|---|
| Definition: | Displays a message box containing the name of the control, a copyright message and an OK button. Pressing the button will unload the message box. |
| Parameters: | None |
| Syntax: | <code>Annotel>AboutBox</code> |
| Return Value: | None |
| Comments: | This message box provides information about the Barcode control. |

BarcodeType Property

| | |
|---------------------|--|
| Definition: | Determines the type of barcode the VSBarcode engine will interpret. |
| Data Type: | Integer (Enumerated) |
| D/T Access: | Read/Write |
| R/T Access: | Read/Write |
| Description: | <p>This property gives fourteen different choices for barcode interpretation. The default, <i>0--Auto Detect</i>, will interpret all of the barcodes listed. When set to <i>0--Auto Detect</i>, the VisionShape Barcode ActiveX will take significantly longer to process information. This property gives the following options:</p> <ul style="list-style-type: none">0--Auto Detect1--Codabar2--Code 393--Code 39 Ext4--Code 935--Code 1286--Code 2 of 57--Code 2 of 5 Int8--UPC-A9--UPC-E10--EAN-811--EAN-1312--Batch Card13--POSTNET |

BeamWidth Property

| | |
|--------------------|---|
| Definition: | Specifies the size of the barcode that the VisionShape engine is to identify. |
|--------------------|---|

| | |
|---------------------|---|
| Data Type: | Integer |
| D/T Access: | Read/Write |
| R/T Access: | Read/Write |
| Description: | The BeamWidth Property is used mainly for troubleshooting barcode readings. The value set in this property corresponds to 1/100th inch segments. If VisionShape is having difficulty recognizing a barcode, this property can be set to a larger value for a more accurate reading. The wider the BeamWidth property is set, the slower, but more accurate, the engine will be. However, if the BeamWidth is set too high, nearby text in the image may interfere with the reading. |

CheckDigit Property

| | |
|---------------------|---|
| Definition: | The CheckDigit property enables the checksum verification encoded into the barcode. |
| Data Type: | Integer (Enumerated) |
| D/T Access: | Read/Write |
| R/T Access: | Read/Write |
| Description: | Checksum verification is an encoded some barcodes to help double-check the accuracy of recognition attempts. When this property is set to True, the VisionShape control will verify is recognition output against the checksum encoded in the barcode. If the checksum does not match, the barcode will be reported as unreadable and the output string will display a tilde (~). The recognition output will not be affected if a barcode does not contain a checksum. |

Error Event

| | | |
|---------------------|--|---|
| Definition: | Occurs for each error internal to the control. | |
| Parameters: | Number | A long error code that identifies the error |
| | Description | Descriptive string of the error |
| | SCode | A composite long number indicating the severity of the error, the facility code, the origin of the error, and the native error code |
| | Source | Descriptive string of the source of the error |
| | HelpFile | Suggested help file name that should have a detailed explanation of the error |
| | HelpContext | Context ID in the help file |
| | CancelDisplay | If set to True during this event, the standard error dialog will not be displayed |
| Description: | Any time an error occurs inside the VisionShape Barcode control, the Error event is triggered. | |

ImageDataSource Property

| | |
|---------------------|---|
| Definition: | Specifies the name of the ImageBASIC control that will supply image data to this control. |
| Data Type: | String |
| D/T Access: | Read/Write |
| R/T Access: | Read/Write |
| Description: | This property may be set to the name of any ImageBASIC control that is capable of outputting image data from the VisionShape Barcode control. |

Progress Event

| | |
|---------------------|--|
| Definition: | This event is fired at the beginning of a recognition attempt. |
| Description: | The Progress Event is a good place to place code which informs users how much time is left on their recognition attempt. |

QuietZone Property

| | |
|---------------------|---|
| Definition: | Specifies the amount of whitespace that the VisionShape engine should expect to find at the ends of barcodes |
| Data Type: | Integer |
| D/T Access: | Read/Write |
| R/T Access: | Read/Write |
| Description: | This property is set in pixels units. The default value is zero, but recognition will proceed more quickly if the white space is at least twice as large as the maximum space between bars. |

ReadMultiple Property

| | |
|---------------------|---|
| Definition: | When True, activates the control's ability to recognize multiple barcodes in a single operation. |
| Data Type: | Integer (Enumerated) |
| D/T Access: | Read/Write |
| R/T Access: | Read/Write |
| Description: | If this property is set to True when recognition is performed, and more than one barcode is present in the recognition region, the VisionShape control will attempt to read and report all of the barcodes. If multiple barcodes are found, the return string will report the values of all of the codes, separated by pipe () characters (ASCII 124). |

Result Property

| | |
|---------------------|--|
| Definition: | Holds the recognition output of the most current recognition attempt. |
| Data Type: | String |
| D/T Access: | N/A |
| R/T Access: | Read-only |
| Description: | This property should be queried to retrieve the results of the most current recognition attempt. If the ReadMultiple property is set to True and more than one barcode is read, the Result property will report the values of barcodes, separated by pipe () characters (ASCII 124). If a recognition attempt is made, but the controls fails to recognize a barcode, the returned string will be populated with a tilde (~) character. |

Recognize Method

- Definition:** Triggers a recognition attempt.
- Description:** This method will trigger the VisionShape engine to attempt recognition. All of the parameter properties, such as ReadMultiple, QuietZone, BeamWidth, CheckDigit, BarCodeType, and Search Mode properties must be set before this method is triggered. The result of this recognition attempt will be reported in the Result property.

RegionSource Property

- Definition:** Set this property to the display control whose working regions you want to recognize.
- Data Type:** Integer (Enumerated)
- D/T Access:** Read/Write
- R/T Access:** Read/Write
- Description:** This property tells the barcode engine to identify all the working regions in the specified display.
- Note:** Currently multiple regions are not supported with the 32-bit BarCode control. The 16 bit BarCode ActiveX will recognize multiple working regions.

SearchMode Property

- Definition:** Tells the VisionShape engine where to search for the barcode on the current image.
- Data Type:** Integer (Enumerated)
- D/T Access:** Read/Write
- R/T Access:** Read/Write
- Description:** The choice of a search pattern should be based on the size of the image passed to the recognition engine. Smaller regions usually require fewer search passes. Larger image regions require more search passes to ensure that any barcodes present will not be missed. The SearchMode property gives the following options:
- 0--Adaptive Search
 - 1--3 Horizontal Scans
 - 2--33 Horizontal Scans
 - 3--99 Horizontal Scans
 - 4--99 Horizontal and Vertical Scans
 - 5--3 Vertical Scans
 - 6--33 Vertical Scans
 - 7--99 Vertical Scans
 - 8--200 Horizontal and Vertical Scans

The number of steps shown above indicates the number of sections into which the image data passed to the VisionShape engine is divided. The engine then searches along each dividing line for evidence of a barcode. As you might expect, more search passes generally result in a higher recovery rate, but require more time. In general, if you are passing small regions of image data to the recognition engine, fewer search passes are necessary. However, if an entire page is being search by VisionShape, more passes are likely to be necessary.

Appendix A: Error Codes

Possible VisionShape Barcode Errors

| | |
|----------------------------|---|
| -1401 ERR_OCR_ENGINE | Cannot perform barcode recognition. Insufficient memory problem. Free up memory and try the recognition attempt again. |
| -1411 ERR_OCR_ENGINE-10 | Invalid file format or corrupted image. Check the list of valid image files. If this is not an invalid file format, the image is most likely corrupted. For a list of valid file formats, refer to the <i>ImageBASIC TMS Display's User's Guide</i> |
| -1410 ERR_OCR_COMMAND | Memory allocation failure. Free up memory and try the recognition attempt again. |
| -1420 ERR_OCR_SET | Illegal attribute set. Reset recognition attributes and try the recognition attempt again. |
| -1421 ERR_OCR_GET | Illegal attribute set. Reset recognition attributes and try the recognition attempt again. |
| -1422 ERR_OCR_LOCAL_MEMORY | Cannot allocate local memory. Free up memory and try the recognition attempt again. |
| -1498 ERR_ILL_X | Illegal X position or Width of zone. Redefine recognition zone and try again. |
| -1499 ERR_ILL_Y | Illegal Y position or Height of zone. Redefine recognition zone and try again. |
| -1801 ERR_ILL_HANDLE | Internal error. Please contact Diamond Head Software Technical Support. |
| -2311 ERR_ILL_IMAGE_NO | Internal error. Please contact Diamond Head Software Technical Support. |

| | |
|-------------------------|---|
| -2310 ERR_ILL_IMAGE | Internal error. Please contact Diamond Head Software Technical Support. |
| -3652 ERR_ILL_LENGTH | Length of result string is not big enough for the data. |
| -30000 ERR_MEMORY | Out of Global memory |
| -30100 ERR_MEMORY_LOCAL | Out of local memory in the barcode DLL. Please contact Diamond Head Software Technical Support. |

Appendix B: Barcode Types

Supported Barcodes

The following section is an excerpt for the *AutoClass Barcode Version 3.0 User's Guide*, Copyright 1990 by VisionShape, Inc. All rights reserved.

Code 251

| | |
|---------------------------------|--|
| Character Set: | 0123456789 (Numeric only) 1 start/stop pattern |
| Type: | Binary |
| Length: | Fixed |
| Check Digit: | No required check digit |
| Maximum Density: | 18 chars/inch |
| Minimum Bar Width: | .0075 inch |
| Small to Big Bars Ratio: | 2-3 |
| Description: | Code uses 5 bars and 5 spaces (5 elements)/character; one character size, always 2 thick; 3 thin; two symbols are encoded in one code character. |

Codabar

| | |
|---------------------------------|---|
| Character Set: | 0123456789-\$./+ (16 characters) 4 start/stop characters (can be used for function code) |
| Type: | Binary |
| Length: | Variable |
| Check Digit: | No required check digit |
| Maximum Density: | 12.8 chars/inch |
| Minimum Bar Width: | .0075 inch |
| Small to Big Bars Ratio: | 2-3 |
| Description: | Code uses 4 bars and 3 spaces (7 elements/character) |

Code 39

| | |
|---------------------------------|--|
| Character Set: | A-z 0-9 -\$/.+ (26+10+7 characters) 1 start/stop character |
| Type: | Binary |
| Check Digit: | No required check digit |
| Maximum Density: | 9.8 chars/inch |
| Minimum Bar Width: | .0075 inch |
| Small to Big Bars Ratio: | 2-3 |
| Description: | Code uses 5 bars and 4 spaces (9 elements)/character; 3 elements are always big, the others are small, this results in a fixed length barcode character. |

Code 128

| | |
|-----------------------------|--|
| Character Set: | All 128 ASCII characters 4 non data function characters 4 code set selection characters High density mode available for numeric characters only 3 start/1 stop character |
| Type: | Proportional |
| Check Digit: | Check digit included in the barcode (modulo 103) |
| Maximum Density: | 12.12 chars/inch 24.23 chars/inch if high density numeric |
| Minimum Bar Width: | .0075 inch |
| Number of Bar Types: | 4 |
| Description: | Code uses 3 bars and 3 spaces (6 elements)/character; characters are always 11 units long (except for the stop character) |

Code 93

| | |
|-----------------------|--|
| Character Set: | Includes 43 character. 0-9, A-Z, 6 symbols (-.\$%+?), SPACE and 4 shift characters. The entire 128 ASCII character set is represented using combination of shift characters and basic data characters. 3 start/ 4 stop characters |
| Type: | Proportional |
| Check Digit: | Two check digits included in the barcode (modulo 47), mandatory |

| | |
|-----------------------------|--|
| Maximum Density: | 11.11 chars/inch, alphanumeric 5.56 chars/inch for full ASCII (based on X=0.010 inch or 0.23 mm) |
| Minimum Bar Width: | 0.0075 inches (0.191 mm) |
| Number of Bar Types: | 3 |
| Description: | Code uses 3 bars and 3 spaces (6 elements/character) |

EAN-13

| | |
|---------------------------|---|
| Character Set: | 10 characters (Numeric only) |
| Type: | Binary |
| Check Digit: | One check digit (13th character) |
| Minimum Bar Width: | 0.0132 inch (0.33 mm) |
| Description: | Code uses 2 bars and 2 spaces (4 elements/character) Additional features: In addition to 13 characters, it also consists of two guard characters and a center character. |

EAN-8

| | |
|---------------------------|---|
| Character Set: | 10 characters (Numeric only) |
| Type: | Fixed |
| Check Digit: | One check digit (weighted modulo 10) |
| Minimum Bar Width: | 0.0132 inch (0.33 mm) |
| Description: | Code uses 2 bars and 2 spaces (4 elements/character) Additional features: In addition to 8 characters, it also consists of two guard characters and a center characters. |

UPC-A

| | |
|---------------------------|---|
| Character Set: | 10 characters (Numeric only) |
| Type: | Fixed type, only 12 digits |
| Minimum Bar Width: | 0.032 inch (0.33 mm) |
| Description: | Code uses 2 bars and 2 spaces (4 elements/characters) additional features: In addition to 12 characters, it also consists of two gurard characters |

UPC-E

| | |
|-----------------------------|--|
| Character Set: | 0 characters (Numeric only) |
| Type: | Fixed type, only 6 digits |
| Check Digit: | One check digit |
| Minimum Bar Width: | 0.0132 inch (0.33 mm) |
| Description: | Code uses 2 bars and 2 spaces (4 elements/character) |
| Additional features: | The UPC-E symbol, on being scanned, yields 8 digit values, 6 values directly from the digit bar characters and values from the variable parity |

POSTNET (POSTalNumericEncodingTechnique)

| | |
|-----------------------------|--|
| Character Set: | 10 characters (Numeric only) 1 start/ 1 stop character |
| Bar Size: | Width 0.015-0.025 inches Height 0.040 - 0.060 inches (for small bar) Height 0.135 - 0.115 inches (for large bar) |
| Number of Bar Types: | 2 (depending on height) |
| Description: | Code uses 5 bars (5 elements/character) |
| Additional Features: | In this type of code, the width of each bar is not important, height of the bar defines the barsize |

Batch Card

| | |
|-----------------------------|---|
| Character Set: | 10 characters (Numeric only) represents numbers from 0 - 64 1 start/ 1 stop character |
| Bar Width: | 0.5 inch, 1 inch, 1.5 inch and 2 inch |
| Number of Bar Types: | 4 |
| Description: | Code uses 5 bars and 3 spaces (8 elements/character) (These are VisionShape Batch header Cards used by AutoScan-Pro software for batch separation) |

Index

A

AboutBox Method 12
Active Property 11

B

BarCodeType Property 3–12
Batch Card 12
Batch Card Supported Barcode 24
Beam Width 7
BeamWidth Property 7–12

C

CheckDigit Property 8–13
CheckDigits 8
Codabar 12
Codabar Supported Barcode 21
Code 128 12
Code 128 Supported Barcode 22
Code 251 Supported Barcode 21
Code 39 12
Code 39 Supported Barcode 22
Code 93 12
Code 93 Supported Barcode 22

E

EAN-13 12
EAN-13 Supported Barcode 23
EAN-8 12
EAN-8 Supported Barcode 23
Error Event 14
Events
 Error 14
 Process 14

I

ImageDataSource 1–4, 6–14
ImageDataSource Property 1, 3, 4, 14
Improving Recognition Results 3–6
 BeamWidth Property 7
 CheckDigits Property 8
 QuietZone Property 8

SearchingMode Property 8

InputFileName Property 6
Introduction to the VisionShape Barcode
 ActiveX 3

L

Licensing Verification 1
Linking ImageBASIC Controls
 ImageDataSource Property 1
Linking of ImageBASIC Controls 1

M

Methods
 AboutBox 12
 Recognize 4, 5, 6, 16

P

Performing Recognition 3
 BarCodeType Property 3
POSTNET 12
POSTNET Supported Barcode 24
Progress Event 14
Properties
 Active 11
 BarCodeType 3, 12
 BeamWidth 7, 12
 CheckDigit 13
 CheckDigits 8
 ImageDataSource 3, 4, 6, 14
 InputFileName 6
 QuietZone 8, 15
 ReadMultiple 3, 4, 5, 15
 RegionSource 5, 16
 Result 4, 5, 15
 SearchMode 6, 8, 16
 SearchMode Property 4

Q

Quiet Zones 8
QuietZone Property 8–15

R

- ReadMultiple Property 3–15, 3–15, 5, 15
- Recognize Method 4–16, 5, 6, 16
- Recognizing Barcodes 3–4
 - ImageDataSource Property 3
 - ReadMultiple 3, 4
 - Recognize Method 4
 - Result Property 4
 - SearchMode Property 4
- Recognizing Barcodes in Working Regions
 - ImageDataSource Property 4
 - ReadMultiple Property 5
 - Recognize Method 5
 - RegionSource Property 5
 - Result Property 5
- Recognizing Barcodes Set in Working Regions 4
- RegionSource Property 5, 16
- Result Property 4–15, 5, 15
- Retrieving Recognition Output 4–5
 - Result Property 5

S

- Searching Modes 8
- SearchMode Property 4, 6, 8–16
- Sending Recognition Output to a File 6
 - ImageDataSource Property 6
 - InputFileName Property 6
 - Recognize Method 6
 - SearchMode Property 6
- Supported Barcodes 3
 - Batch Card 24
 - Codabar 21
 - Code 128 22
 - Code 251 21
 - Code 39 22
 - Code 93 22
 - EAN-8 23
 - POSTNET 24
 - UPC-A 23
 - UPC-E 24

U

- UPC-A 12
- UPC-A Supported Barcode 23
- UPC-E 12
- UPC-E Supported Barcode 24

V

- VisionShape Barcode Errors 19