

# Introduction to ImageBASIC 3.1

---

Including Tutorial and Development Utilities

*IMAGE*  *BASIC*

Diamond Head Software, Inc.  
1217 Digital Drive Ste. 125  
Richardson, Texas 75081  
(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 are protected by one or more of the following copyright notices:

Copyright © 1996 Diamond Head Software, 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.

01/28/97 12:40:44

# Contents

Chapter 1 : Introduction	5
Getting Started with ImageBASIC 3.1.....	5
Installation and Licensing Configuration.....	5
Communication Between ImageBASIC Controls.....	6
Chapter 2 : Tutorial for ImageBASIC	11
Starting a New Project.....	11
Adding Image Display and Basic Management.....	11
Adding File Saving and Printing.....	16
Adding OCR Using the TextBridge ActiveX Control.....	21
Adding Image Enhancement Using the ScanFix Control.....	22
Conclusion to the Tutorial.....	24
Chapter 3 : Development Aids	25
Programming Wizards.....	25
Using Wizards.....	25
Preparing Wizards.....	25
Running Wizards.....	26
Chapter 4 : ImageBASIC Widgets	27
Introduction.....	27
Using an ImageBASIC Widget.....	28
Reference to the Properties and Events of Widgets.....	29
Index	35



# Chapter 1 : Introduction

---

## Getting Started with ImageBASIC 3.1

As you will discover as you begin the short process of learning to use it, ImageBASIC 3.1 is a powerful and flexible development environment that will help you quickly integrate imaging and workflow into your business solutions. You will also discover that, in addition to the many other advantages provided by this high level toolkit, ImageBASIC is relatively easy to learn and easy to implement.

Over the course of the following pages, we will step through the creation of a simple application in Visual Basic 4.0. This application will begin using only the core components necessary for file access and image display. We will then add other functionality using optional components. Because one of the advantages of using component software and designing modular applications is the ability to easily add or remove functionality, if you are not using these tools, simply skip that section and continue with the next. Of course, this simple application will be limited, but it will introduce the core concepts of ImageBASIC and will help you get started on your own development projects.

---

## Installation and Licensing Configuration

### Installation of Software

- 1) If you have not completed the installation process yet, please do so now by running the ImageBASIC installation from the CD Navigator. The Navigator is run when you execute SETUP.EXE in the root directory of the CD.
- 2) The installation program is a wizard that will step you through the selection of all installation parameters. On each screen, select the options appropriate to your system and requirements and then continue to the next screen.
- 3) The final screen allows you to start copying files and performing the other installation steps. The installation of ImageBASIC 3.1 on 16-bit Windows includes the installation and registration of a number of OLE files. After all ImageBASIC files are copied to your local drive, the newly installed controls will be added to the system registry.
- 4) Restart Windows before using ImageBASIC.

## Licensing Configuration

After installation is complete, you must have a development license in order to use any ImageBASIC components.

- *If you are evaluating ImageBASIC* and have installed from the CD, please run the CD Registration and fax the Registration Form to us. We will return to you an authorization code that allows you full development capabilities for thirty (30) days. The only limitation on this license is that you will be unable to compile your project into an executable, but you may run the application inside your development environment.
- *If you have purchased a toolkit*, you will be sent a hardware key that contains licenses for all of the components you now own. This key can be remotely updated if you later purchase additional components. This small device must be attached to a parallel port on your PC. If no parallel port is available, the hardware key may be attached to the port and another device can then connect to the back of the key. The hardware key will allow all messages to and from the attached device to pass through unaffected.

---

## Communication Between ImageBASIC Controls

### Types of Inter-Control Communication

The ImageBASIC controls communicate amongst themselves in a unique manner to get information and data to the control that needs it. There are two times that this communication is performed:

- 1) Image data is passed, or
- 2) Services are performed

Each ImageBASIC control can be either a source or a recipient of one or both of these types of communication. The act of specifying the source and recipient of this communication is called *linking* the controls.

All ImageBASIC controls that can accept image data from another ImageBASIC control have a property called **ImageDataSource**. The ImageDataSource property specifies the ImageBASIC control that is the source for all incoming images.

- For example, the TMS Display control cannot directly access image files on disk.
- The TMS File control must read the image files into memory where they can be retrieved by the Display control.

- Therefore, the source of images for the TMS Display control is the TMS File control. The communication between the controls is enabled by setting the TMS Display control's ImageDataSource to specify the TMS File control.

Some ImageBASIC controls offer services other than image processing. The communication for these controls is enabled in a similar manner, but through properties other than the ImageDataSource property.

- For example, the Annotation control does not process image data and, therefore, does not have an ImageDataSource property.
- Instead, the Annotation control performs a service -- creating and maintaining annotations. The Annotation control must have a Display control on which to show its annotations.
- The Annotation control's DisplaySource property must specify the TMS Display control that will be performing this function.

### Methodology of Linking

As each instance of an ImageBASIC control is created on a Form, a unique identification string is calculated for that control. This string, or Link ID, is stored in the **Link** property of each control. This property is not visible at design time and cannot be changed by the application designer. The unique Link ID is calculated each time a control is created, whether it is created statically at design time or dynamically at runtime.

The assignment of a source control is made by setting the Source property of the receiving control -- perhaps the ImageDataSource of a TMS Display control -- to the Link property of the source control. For example, for a TMS Display control to accept image data from a TMS File control, set the ImageDataSource property as shown here for Delphi:

```
TMSDispl.ImageDataSource := TMSFile1.Link
```

Other forms of inter-control communication are established in the same manner. For example, allowing an Annotation control to display its objects on a TMS Display control is performed by setting the Annotation control's DisplaySource property as shown here:

```
Annotel.DisplaySource = TMSDispl.Link
```

The links between ImageBASIC controls are made either at design time or at runtime. Changes may be made at any time during runtime if your application requires receiving image data or services from multiple controls. The following chart shows which controls can accept and provide the different services.

### Table of Linking Properties

<b>Control Name</b>	<b>Does it have ImageDataSource property? (Can it accept image data?)</b>	<b>Is it a Source of Image Data? (Can it be named in ImageDataSource?)</b>	<b>Other Linking Properties and their purpose?</b>	<b>Other service provisions? How used?</b>
TMS Display	Yes	Yes	None	Can be named in the RegionSource property of other controls
TMS File	Yes; only for saving to file	Yes	None	Can read/write annotations to TIFF header if named in Annotation control's AnnoteSource
Pixel Scan	No	Yes	None	None
Annotation	No	No	DisplaySource: Display control for showing annotations  AnnoteSource: File control for read/write of TIFF header	None
ScanFix	Yes	Yes; output of only the results of enhancement	None	None
TextBridge	Yes	Yes; only during Verify event	RegionSource: Display control for region definition	None
VS Bar Code	Yes	No	RegionSource: Display control for region definition	None

As each control is added to our simple application, the linking of the controls and the flow of image data and services will be noted. Although the process may sound



a bit confusing at first, it is readily mastered when you use ImageBASIC. If you work your way through the tutorial that begins on the next page, you will quickly see the implementation of linking and other major features of ImageBASIC 3.1.



# Chapter 2 : Tutorial for ImageBASIC

---

## Starting a New Project

Begin by running Visual Basic 4.0 and starting a new project. The application that will be built in the following pages is based upon Visual Basic 4.0 16-bit. If you are using Visual Basic 4.0 32-bit, there may be occasional small differences. All ImageBASIC controls, however, are identical in the 16-bit and 32-bit versions.

As we add functionality to this small application, we will be using a few standard Visual Basic controls. If these are not loaded at startup, please add them now:

Command Button

Common Dialog

Picture Box

Text Box

---

## Adding Image Display and Basic Management

### *Adding the First ImageBASIC Components*

Open the Custom Controls dialog and add the first two ImageBASIC controls to the project. These ImageBASIC components are used for file access (both reading and writing) and for image display. They appear in the Custom Controls dialog as shown below:

ImageBASIC Display - TMS

ImageBASIC File - TMS

These components will be referred to as TMS Display and TMS File throughout this document and the various user's guides.

#### **1) Add one instance of each of these controls to your Form.**

Add the control by clicking once on the toolbox icon and dragging out the control on the Form or by double clicking on the toolbox icon.

- a) Place a TMS File control on the Form

Be sure to add the TMS File control first. This will allow the TMS Display control, added next, to automatically link to the TMS File control for file access. If the TMS Display control is added to the Form first, then the link may be set up by simply selecting from the drop-down list box shown in the Properties Window for the ImageDataSource property.

- b) Place a TMS Display control on the Form.

Size the TMS Display control large enough that the image displayed in it will be readable. Four to six inches on each side is a reasonable minimum dimension for the display of a letter size page. A larger window will be able to display the image more clearly.

### **2) Add one command button and change the Caption to *Load Image*.**

In the command button's Click event, add a single line of code:

```
TMSFile1.InputFileName = "c:\images\form1.tif"
```

Change the file name in this example to the name of a TIFF file that you have available.

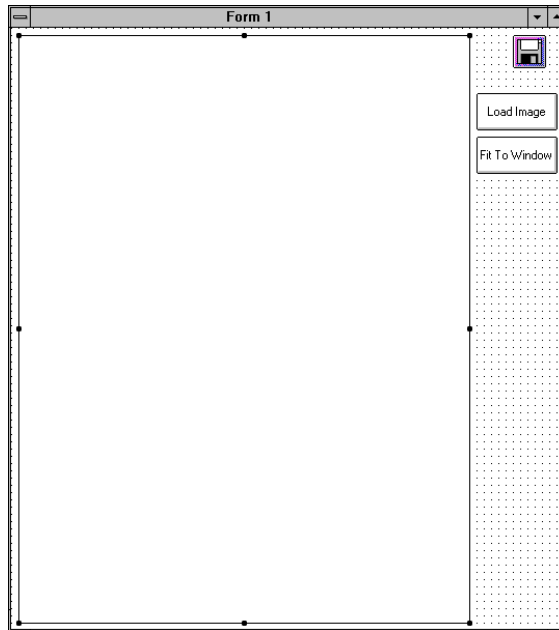
### **3) Add a second command button and change the Caption to *Fit to Window*.**

In the button's Click event and one line of code:

```
TMSDispl.FitToWindow
```

When this button is clicked, the image will be returned to a full-size display after zooming in or out.

Your Form should now look something like the following illustration, although the exact position of the controls is not important:



### 3) View the Properties Window for the TMS Display control.

Find the following properties and verify that they are set as shown here:

LeftButtonOption	4--Proportional Zoom
RightButtonOption	1--Rubber Band
ImageDataSource	TMSFile1

The ImageDataSource property specifies the ImageBASIC control from which the TMS Display control will get the image data that it displays. When the TMS Display control is drawn on the Form, it searches for other ImageBASIC controls already on the Form to use as the default ImageDataSource.

- At design time, this property may be changed through the dropdown list box shown in the Properties Window.
- At runtime, the property specifies the control that will provide the data.

For example, in the current project, the TMS File control will supply image data to the TMS Display control. Therefore, the ImageDataSource would be set as shown here:

```
TMSDispl.ImageDataSource = TMSFile1.Link
```

### 4) Run the application.

As soon as the application is running, click the "Load Image" button. The specified file will be read from disk and displayed.

- The image will be centered on the Display control, possibly leaving some portion of the control visible outside the bounds of the image either on the top and bottom or on the left and right. The image will be fit to the window as well as possible without distorting the width-to-height ratio of the image.
- The portion of the Display window that is not covered by image data is often referred to as the overscan area. The color of this area is controlled by setting the BackgroundColor property of the TMS Display control.
- It is possible to define regions and to draw annotations that extend into the overscan area. However, anything outside the bounds of the image itself will be lost when the image or annotations are saved or transferred to another ImageBASIC control.

The left and right mouse buttons each have some predefined functionality. Each mouse button is used in the definition of an image region. Define a region by depressing the mouse button at one corner of the region and then holding the button down while moving the cursor to the opposite corner and releasing the button.

- The region defined by the left mouse button is referred to as the Zoom Region.  
If you will now drag out a region using the left mouse button, you will see that the defined region is enlarged to fill the display window.  
Using the scroll bars that appear, you can scroll through the image. You may continue to zoom in more by dragging out smaller regions using the left mouse button.

Click the "Fit to Window" button to return the image to a full display.

- The right mouse button defines a region called the Working Region. The Working Region is a subsection of the image that can be processed independently of the whole image. For example, the Working Region can be read by the TextBridge OCR control or can be saved as a separate image by the TMS File control.

Each of the ImageBASIC controls that can process either the Working Region or the entire image has a pair of complementary methods to initiate the action. For example, the TMS File control has the Save method and the SaveRegion method. The TextBridge control has the OCRPage method and the OCRRegion method.

- The functionality assigned to each of the mouse buttons can be modified through the LeftButtonOption and RightButtonOption properties.

## **5) Stop the application.**

We can now add some more features to our small application to expand its usefulness and to introduce more of the features of ImageBASIC. Initially, we will add printing and saving of files.

- Any time a change is made to an image or a new image is captured, the new image needs to be saved. Alternatively, a portion of the image may be saved separate from the rest of the image.
- Any image that can be displayed can also be printed. Any portion of the image (down to a certain minimum region size that depends upon the resolution of the image and the resolution of the print device) may also be independently printed.

Because the processes of printing and saving are both relatively quick and simple, we will add options for both to our growing application in the next step, along with a few other improvements for the display of varying images.

---

## Adding File Saving and Printing

A small amount of additional code will allow the addition of saving and printing capabilities. The following features will be added in the next few pages to make our application more usable:

- Basic image manipulation -- grayscaled display, image rotation and inversion
- Saving the displayed image
- Saving a region of the displayed image
- Printing the displayed image
- Printing a region of the displayed image

To simplify future expansion and work toward a standard Windows interface, we will remove the buttons that were added in the previous step and create menu items to replace them.

**1) Open the Menu Editor window and add menu items as shown in this table:**

Menu Item	Object Name
File	mnuFile
Open Image	mnuFile_Open
Close Image	mnuFile_Close
-	mnuFile_Sep1
Save Image	mnuFile_SaveImage
Save Region	mnuFile_SaveRegion
-	mnuFile_Sep2
Exit	mnuFile_Exit
Display	mnuDisp
Display as Gray	mnuDisp_Gray
Rotate Clockwise	mnuDisp_RotClock
Rotate Counter-clockwise	mnuDisp_RotCounter
Inverse Colors	mnuDisp_Inverse
Fit Image To Window	mnuDisp_FitToWindow
Printing Options	mnuPrint
Printer Selection	mnuPrint_Select
Print Image	mnuPrint_Image
Print Region	mnuPrint_Region



**2) Click the OK button to close the Menu Editor after all of these items are added.**

We can now place a small bit of code in each item:

- a) To open any image file, we will use the Common Dialog control to allow the user to browse all available drives and directories. In design time, click on the **File, Open Image** menu item to open the code window and add the following:

```
Sub mnuFile_Open_Click ()  
    CommonDialog1.Filter = "TIFF Files|*.tif|All Files|*.*"  
    CommonDialog1.FilterIndex = 1  
    CommonDialog1.ShowOpen  
    TMSFile1.InputFileName = CommonDialog1.FileName  
End Sub
```

- b) Closing an image file and clearing the TMS Display window requires only a single line of code to clear the TMS File control:

```
Sub mnuFile_Close_Click ()  
    TMSFile1.InputFileName = ""  
End Sub
```

- c) To save the currently displayed image file, we will again use the Common Dialog control to select the output file name. For now, we save all files as TIFF Group 4 which is the default and supports only bitonal images. If you wish to save other file types, set the TMS File control's OutputFileFormat property to specify the preferred format before saving the file.

To select a file name and then save to that file, only the following code is required:

```
Sub mnuFile_SaveImage_Click ()  
    CommonDialog1.ShowSave  
    TMSFile1.Save CommonDialog1.FileName  
End Sub
```

To save the area of the displayed image that is defined as the Working Region, execute the SaveRegion method instead of the Save method:

```
Sub mnuFile_SaveRegion_Click ()  
    CommonDialog1.ShowSave  
    TMSFile1.SaveRegion CommonDialog1.FileName  
End Sub
```

- d) The Exit menu item will end the application, releasing all memory used for image display and manipulation and unloading all of the dynamically linked libraries:

```
Sub mnuFile_Exit_Click ()  
    End  
End Sub
```

- e) The following menu items will all be used to perform some basic image manipulation and optimize the image display if it is difficult to read or is rotated. Each of these items requires only a few lines of code:

Displaying a bitonal image as grayscale makes the text more readable and reduces eye strain:

```
Sub mnuDisp_Gray_Click ()  
    TMSDisp1.ScaleToGray = Not TMSDisp1.ScaleToGray  
    mnuDisp_Gray.Checked = TMSDisp1.ScaleToGray  
End Sub
```

Image files may have been scanned and saved in any orientation, so the next two menu items allow the operator to rotate them to a readable orientation. Each time these routines are executed, the image is rotated to the next 90 degree increment, either clockwise or counter-clockwise:

```
Sub mnuDisp_RotClock_Click ()  
    With TMSDisp1  
        Select Case .Rotation  
            Case 0  
                .Rotation = 90  
            Case 90  
                .Rotation = 180  
            Case 180  
                .Rotation = 270  
            Case 270  
                .Rotation = 0  
        End Select  
    End With  
End Sub
```

```

Sub mnuDisp_RotCounterClock_Click ( )
    With TMSDisp1
        Select Case .Rotation
            Case 0
                .Rotation = 270
            Case 90
                .Rotation = 0
            Case 180
                .Rotation = 90
            Case 270
                .Rotation = 180
        End Select
    End With
End Sub

```

Images may be supplied with inverted colors (white text on a black background) either because the originals were printed that way or because the scanning process introduced the inversion. The next option allows the image to be displayed with inverse colors and will place a check mark by the menu item to indicate when this is being done.

```

Sub mnuDisp_Inverse_Click ( )
    TMSDisp1.Invert = Not TMSDisp1.Invert
    mnuDisp_Inverse.Checked = TMSDisp1.Invert
End Sub

```

After zooming and scrolling in the TMS Display window, return the image to a full-sized display through this menu item:

```

Sub mnuDisp_FitToWindow_Click ( )
    TMSDisp1.FitToWindow
End Sub

```

- f) Printer selection, setup and printing the image page each require only one additional line of code. The first step is to select and setup the printer. The following code segment will open a standard Windows printer selection dialog. This dialog will allow the user to select the printer of choice and to make any setup changes before printing begins.

```

Sub mnuPrint_Select_Click ( )
    TMSDisp1.SelectPrinter
End Sub

```

After the printer is selected and setup is performed-- all Windows defaults will be used if this step is omitted -- the image may be printed. Add this code to the specified menu to print the entire image page that is currently displayed. When printed, the image will be scaled to fit the printed page just as images are scaled to fit the display window.

```
Sub mnuPrint_Image_Click ()  
    TMSDispl.PrintImage  
End Sub
```

Rather than printing the entire image, it may be necessary to print only a smaller portion of the image. Any portion of the image defined as the Working Region may be printed separate from the remainder of the image. The region that is selected will be scaled to fit the entire output page just as the entire image would be.

```
Sub mnuPrint_Region_Click ()  
    TMSDispl.PrintRegion  
End Sub
```

### **3) Run the application and experiment with these new features.**

You can load image files in any of the supported formats and can perform basic manipulation including rotation and inversion. In the next section, we will be adding OCR capability using the TextBridge ActiveX control. This control will accept image data from the TMS Display window and output its results directly to the screen for review.

---

## Adding OCR Using the TextBridge ActiveX Control

After stopping the application and returning to design mode, open the Custom Controls dialog and add the TextBridge ActiveX control to your project. This control appears in the list as *ImageBASIC Recognition -- TextBridge*

- Add a single instance of the TextBridge control anywhere on the Form.

The ImageDataSource property of the control will automatically be set to the TMS Display control. This setting allows the TextBridge control to accept image data for processing directly from the TMS Display control.

In addition, the TextBridge control has a property called RegionSource. This property is also set to the TMS Display control. This setting allows the TextBridge control to accept region coordinates from the TMS Display control along with the image data. By accepting region data with the image data, the control can recognize portions of the image. In this case, the area of the image defined as the TMS Display's Working Region will be processed.

- Add a Text Box control to the Form

Change the MultiLine property to True. The results of the recognition attempt will be reported to this text box. Because the standard text box can display only ASCII text, the output format of the TextBridge control will be left at its default of Text.

To output any of the word processor formats, set the OutputFormat property to specify the preferred format. It is necessary that output to a word processor format be written directly to file by

- 1) setting the OutputTo property to *I--File*, and
- 2) setting the OutputFileName property to the file name of the target.

- Add a command button and change the caption to *OCR Region*.

Add a single line of code to this button's Click event:

```
Text1.Text = TxtBrg1.OCRRegion
```

- This is the only code necessary to both initiate an OCR attempt and to display the results in the text box. The region that is processed is the Working Region of the TMS Display control.

Because the Working Region defaults to the entire image when it is loaded, an entire page will be processed if recognition is begun without defining a Working Region. Use the right mouse button to drag out a region and then press the "OCR Region" button to start the OCR attempt on the defined region.

---

## Adding Image Enhancement Using the ScanFix Control

After stopping the application and returning to design mode, open the Custom Controls dialog and add the ScanFix ActiveX control to your project. This control appears in the list as *ImageBASIC ScanFix -- Sequoia*.

The ScanFix control performs image enhancement such as deskewing, line removal, shading removal, and others. These options are all necessary for the most accurate positioning and clearest presentation of the data on the image for either manual or automated processing. Any application or series of applications that perform OCR on scanned images should perform ScanFix on the images to ensure the most accurate and most reliable OCR possible under the conditions. To add this vital control to a capture process, follow these steps:

1. Add a single instance of the ScanFix control anywhere on the Form.
2. Add three check box controls to the Form. These will be used to enable and disable three select features of the ScanFix engine. In your production application you may desire to add additional features or allow the user to set processing parameters, for example, the maximum size of specks to remove. For now, we will only enable and disable these three features and use the default parameters. Refer to the *TMSSequoia ScanFix ActiveX Control User's Guide* for details on all of the parameters to these features.

Change the Caption of Check1 to *Deskew Image*

Change the Caption of Check2 to *Remove Lines*

Change the Caption of Check3 to *Remove Specks*

3. For now, we will add a single Command Button to start the image enhancement process. In many cases, the enhancement of images is performed in a batch process in which similar code may be used set the enhancement options for each image.

Change the Name of the new button to *cmdScanFix*

Change the Caption of the button to *Begin Image Enhancement*

4. To begin enhancement with the options selected through the Check Box controls that were added in step (2), add the code shown on the next page to the Click event of the command button added in step (3).
5. After the coding is complete, run the application and experiment with the enhancement features of ScanFix. You will probably want to add some more options and detailed control to any production application, but the addition of other features is similar to the process of adding these few enhancement options.

```

Sub cmdScanFix_Click()
    ' The InputFrom property specifies whether the image
    ' for enhancement will come from another ImageBASIC
    ' control or will be read directly from file.
    ' Set InputFrom to specify the receipt of image data
    ' from another ImageBASIC control. Then set the
    ' ImageDataSource property to specify the control that
    ' will supply the image.
ScanFix1.InputFrom = 0
ScanFix1.ImageDataSource = TMSDispl.Link
    ' Similar to the InputFrom property, the OutputTo
    ' property specifies whether the enhanced image will
    ' be written to a new file or will be returned to the
    ' control that supplied the original image. In our
    ' current case, this will display the enhanced image
    ' in the display window, allowing review and saving.
    ' Set the OutputTo property to 0 to specify that the
    ' enhanced image is to be retained in memory and not
    ' written to file.
ScanFix1.OutputTo = 0
    ' Enable or disable the enhancement options based on
    ' whether the user has checked or cleared the Check
    ' Box controls.
ScanFix1.DeskewActivate = Check1.Value
ScanFix1.HorzLineActivate = Check2.Value
ScanFix1.VertLineActivate = Check2.Value
If Check3.Value = True Then
    ScanFix1.DespeckIsolated = 3
Else
    ScanFix1.DespeckIsolated = 0
End If
    ' Start enhancement on the current image. Set the
    ' mouse pointer to an hourglass while processing and
    ' then back to normal after enhancement is complete.
    ' When enhancement is complete, the image will be
    ' displayed in the TMS Display control specified as
    ' the source.
Screen.MousePointer = 11
ScanFix1.ScanFix
Screen.MousePointer = 0

```

---

## Conclusion to the Tutorial

This concludes the tutorial introducing you to ImageBASIC 3.1. As you can see, very little code is required to perform the fundamentals of image processing. In a finished document imaging application, the amount of code dedicated to actual image management is only a small portion of the overall development effort. By reducing the time necessary to create this portion of your application, ImageBASIC allows you to concentrate more on the design and usability of your application and on the organization and retrieval of the images that you need.

Please take some time to look over the sample applications that are installed with ImageBASIC. Each one of these applications highlights a single ImageBASIC control used in conjunction with the core of the ImageBASIC family of controls – display, file, cut & paste. The sample applications are installed under the ImageBASIC installation directory in several different subdirectories depending upon whether they are for Visual Basic 16-bit, 32-bit, or Delphi.

If you have an evaluation CD of ImageBASIC, of course concentrate on the controls with the features that you need now, but take at least a few minutes to look over the other controls with an eye toward future development and integration. Thank you for purchasing ImageBASIC and please let us know if there is anything that we can do to help to successfully and efficiently complete your project.



# Chapter 3 : Development Aids

---

## Programming Wizards

Incorporated into the ImageBASIC 3.1 architecture are Wizards which help make programming of standard ImageBASIC functions easier. Wizards are generally specific to a type of technology. For example, there is one Wizard for the display control and another for the file control. Wizards are actually add-in components that can be ran from inside Visual Basic.

An ImageBASIC Wizard walks you through application setup, asking you questions on what type of feature you would like to incorporate into your application.

Currently, there are two types of ImageBASIC Wizards:

TMSFile

TMSDisplay

Because of the simplicity of the ImageBASIC Wizards, they encompass no ImageBASIC licensing.

---

## Using Wizards

Before you can start using a Wizard, you must run the wizard executable which was installed in your ImageBASIC directory. If you installed both the 16 and 32 bit versions of ImageBASIC, you will have two wizard exe files. However, if you installed either to the 32 or 16 bit ImageBASIC controls you should have the corresponding Wizard control in your ImageBASIC directory.

## Preparing Wizards

Before you can run the wizards, you must prepare them by adding them to the Visual Basic Add-in menu option. To add a the wizards to the Add in Manager:

- 1) Run either SFWiz32.exe or the SFWiz16.exe from the desktop or locate and run these applications in Explorer or File Manager.
- 2) Running one of these executables will add the ImageBASIC Wizards to the Visual Basic **Add In** menu.

## Running Wizards

After you have run the wizard executable, you are ready to use the wizard from inside Visual Basic. To run a wizard:

- 1) Run Visual Basic.
- 2) The wizards will appear in the **Add-in** menu option.
- 3) Choose a wizard. When running both the File and Display wizard, run the File wizard first.
- 4) The wizard will then launch and walk you through the options listed for that control. You should refer to the corresponding component manual for information on the various features.
- 5) After you have finished choosing the options you want, click the Finish button and the wizard will add the features to your Visual Basic form.

**Note:** The wizard may ask you to add a line of code to your application. If so, it will give you detailed information on where to place this information.

In addition, you can not add features twice. For example, if you add a rotate feature and you rerun the wizard, you can not add the rotate feature a second time. However, if you delete that feature from the Form, the wizard will allow you to add the feature.

# Chapter 4 : ImageBASIC Widgets

---

## Introduction

In the new ImageBASIC architecture, Diamond Head Software has created simple controls called Widgets which help simplify ImageBASIC programming. Widgets are different from other ImageBASIC controls in that they do not encompass any particular technology, such as the TMS Display control which handles the ability to display and print image files. Widgets look and act like the textbox and combo box controls on the standard Visual Basic toolbar, but with a catch: they work directly with the ImageBASIC controls. Widgets make setting values in various ImageBASIC controls a simplified task. Programmers can simply place the Widget on the Visual Basic form assign it an ImageBASIC control and an ImageBASIC control property.

There are four type of Widgets for ImageBASIC controls:

- Combo Box

- Check Box

- List Box

- Edit Box

Because of the simplicity of the ImageBASIC Widgets, they do not include ImageBASIC licensing.

Each of the four Widgets operate in a similar manner; however, there are some slight differences. Depending on the type of Widget you use, you will be able to set different types of properties. For example, the CheckBox Widget only recognizes ImageBASIC Boolean properties. This allows the user to check or uncheck the Widget resulting in setting the ImageBASIC property to True when checked and False when unchecked. Each of the Widgets handle different sorts of properties.

## Using an ImageBASIC Widget

To use an ImageBASIC Widget, do the following:

- 1) Place all the desired ImageBASIC control and Widget on the Visual Basic form.
- 2) Set the Widget's **PropertySource** property to the name of the control whose properties you want to set.
- 3) When placed on the Visual Basic form after the ImageBASIC controls, the Widgets automatically default to one of the IB controls. This link can be found in the **PropertySource** property. However, if the Widget is placed on the form before the ImageBASIC controls, you will need to set **PropertySource** in the Properties Window to the desired ImageBASIC control.
- 4) Set the Widget's **PropertyName** property to the name of the ImageBASIC control property you want the user to set during runtime. The following table shows the type of properties that each of the Widgets can expose:

Widget	Properties Set
CheckBox (DHSCkBox)	Boolean (True/False) and Enumerated properties if they contain only two values
ComboBox (DHSCmbBox)	Enumerated
EditBox (DHSEdtBox)	String, Boolean (True/False) and Integer properties
ListBox (DHSLstBox)	Enumerated

- 5) Make any changes to the Widgets appearance properties: Caption, Font, BackColor etc.
- 6) Run the application.

The possible values for the ImageBASIC property will appear in the Widget if using the ListBox or ComboBox. The CheckBox will turn the property on and off, and the EditBox can be used to type in values.

**Note:** Setting the same ImageBASIC property through multiple Widgets may result in a synchronization error. It is recommended that ImageBASIC properties be set by only one widget.

---

## Reference to the Properties and Events of Widgets

The following pages provide a quick reference to the properties and events of ImageBASIC Widgets. If you have any questions about the application of a particular feature, check this section first. If you are just beginning to use Widgets, refer to "Using an ImageBASIC Widget" on page 28 for details of their implementation.

### ***BackColor Property***

**Definition:** Determines the background color of the Widget  
**Data Type:** String  
**Design Access:** Read/Write  
**Runtime Access:** Read/Write  
**Description:** Double-clicking this property in the property window will display a color palette. The color chosen will determine the background color.

### ***Caption Property***

**Definition:** The value of this property is the wording shown in the Widget control.  
**Data Type:** String  
**Design Access:** Read/Write  
**Runtime Access:** Read/Write

### ***DragDrop Event***

**Definition:** This event is fired when a drag and drop operation of the Widget control is completed.  
**Parameters:** Source--the control being dragged in this case the ImageBASIC Widget control  
X and Y--horizontal and vertical position of the mouse within the Widget.  
**See Also:** DragIcon Property, DragMode Property, DragOver Event

### ***DragIcon Property***

<b>Definition:</b>	Specifies the icon that will appear when the mouse pointer is moved over the object.
<b>Data Type:</b>	Integer
<b>Design Access:</b>	Read/Write
<b>Runtime Access:</b>	Read/Write
<b>See Also:</b>	DragDrop Event, DragMode Property, DragOver Event

### ***DragMode Property***

<b>Definition:</b>	Specifies whether a drag and drop operation is automatic or manual.
<b>Data Type:</b>	Integer (Enumerated)
<b>Design Access:</b>	Read/Write
<b>Runtime Access:</b>	Read/Write
<b>See Also:</b>	DragDrop Event, DragIcon Property, DragOver Event

### ***DragOver Event***

<b>Definition:</b>	Is triggered when a drag and drop operation is in progress. This event helps monitor with the Widget control is moved over the target coordinates.
<b>Parameters:</b>	Control--In this case, it is the ImageBASIC widget. X and Y--The target coordinates of the widget control. State--The transition state of the control being dragged in relation to a target form or control: 0 - Enter (source control is being dragged within the range of a target). 1 - Leave (source control is being dragged out of the range of a target). 2 - Over (source control has moved from one position in the target to another).
<b>See Also:</b>	DragDrop Event, DragIcon Property, DragMode Property

### ***DynamicControl Property***

<b>Definition:</b>	Must be set to True at design time if the control will be dynamically cloned, such as part of a control array or member of an MDI child form.  It is recommended that the property is left at its default of False if the control will not be dynamically cloned.
--------------------	---

**Data Type:** Boolean

**Design Access:** Read/Write

**Runtime Access:** Read-only

**See Also:** Link Property, ImageDataSource Property

**Comments:** If this property is set to True at design time, this indicates that the control will be dynamically cloned (e.g., as a control array or as a member of an MDI child form). This forces a new calculation of the **Link** value each time a control initiates.

**DynamicControl** defaults to False and should be left at this value unless the control will be dynamically cloned as part of a control array or a member of an MDI child form.

**Caution:** The effect of setting this property to True is that the **Link** value is not persistent. Any links made at design time will be lost if **DynamicControl** is True. To accommodate this behavior, all links must be made in code.

### ***Enabled Property***

**Definition:** Determines whether a Widget is available for user response.

**Data Type:** Integer (Boolean)

**Design Access:** Read/Write

**Runtime Access:** Read/Write

### ***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 TMS Cut and Paste control, the **Error** event is triggered.

### ***Font Property***

**Definition:** Specifies the particular font, size and characteristics such as bold, italic, etc. of the Widget text.

**Data Type:** String

**Design Access:** Read/Write

**Runtime Access:** Read/Write

### ***ForeColor Property***

**Definition:** Specifies the foreground color of the text of the widget.

**Data Type:** String

**Design Access:** Read/Write

**Runtime Access:** Read/Write

**See Also:** BackColor Property

### ***GotFocus Event***

**Definition:** Occurs when the Widget control receives the focus by user action or by setting the SetFocus method.

**See Also:** LostFocus Event



### ***HelpContextID Property***

**Definition:** Specifies the context-sensitive ID number which associates the Widget with the help file.

**Data Type:** Long Integer

**Design Access:** Read/Write

**Runtime Access:** Read-only

### ***Locked Property***

**Definition:** When True, disables the ability to edit the textbox in runtime. This property only applies to the DHSEdtBox Widget.

**Data Type:** Boolean

**Design Access:** Read/Write

**Runtime Access:** Read/Write

### ***LostFocus Event***

**Definition:** Occurs when the Widget control loses the focus either by user action or by setting the SetFocus method.

**See Also:** GotFocus Property

### ***PropertyName Property***

**Definition:** Specifies the ImageBASIC control property that the Widget will set.

**Data Type:** Enumerated

**Design Access:** Read/Write

**Runtime Access:** Read-only

**See Also:** PropertySource Property

**Description:** This property displays the available properties for the ImageBASIC control specified in the PropertySource property. Depending on which Widget is used, different types of properties will be displayed in the PropertyName property. The following table shows the property type that each widget can access:

<b>Widget</b>	<b>Properties Set</b>
CheckBox	Boolean (True/False) and Enumerated properties if they contain only two values
ComboBox	Enumerated
EditBox	String, Boolean (True/False) and Integer properties

***PropertySourceProperty***

<b>Definition:</b>	Specifies the name of the ImageBASIC control to which it is linked.
<b>Data Type:</b>	Enumerated
<b>Design Access:</b>	Read/Write
<b>Runtime Access:</b>	Read-only
<b>See Also:</b>	PropertyName Property
<b>Description:</b>	This property displays all of the ImageBASIC controls displayed on the Visual Basic form. The PropertyName property will display the available properties.

***Visible Property***

<b>Definition:</b>	Determines if the Widget is visible during runtime.
<b>Data Type:</b>	Boolean
<b>Design Access:</b>	Read/Write
<b>Runtime Access:</b>	Read/Write

***WhatsThisHelpID Property***

<b>Definition:</b>	Specifies the help context ID of the Widget.
<b>Design Access:</b>	Read/Write
<b>Runtime Access:</b>	Read/Write

# Index

## A

Annotation Control 7, 8  
AnnotateSource 8

## B

BackColor Property 28, 29, 31

## C

Caption Property 28, 29  
CD Navigator 5  
Control Communication 6

## D

DisplaySource 8  
Dongle, or Hardware Key 6  
DragDrop Event 29–30  
DragIcon Property 29–30  
DragMode Property 29–30  
DragOver Event 29–30

## E

Enabled Property 30  
Error Event 31  
Evaluation CD 6  
Events  
    DragDrop 29  
    DragOver 30  
    Error 31  
    GotFocus 31  
    LostFocus 32

## F

Font Property 28, 31  
ForeColor Property 31

## G

GotFocus Event 31

## H

Hardware Key 6  
HelpContextID Property 32

## I

ImageDataSource 6, 7, 13  
Installation 5

## L

LeftButtonOption 13  
Licensing 6  
Link ID 7  
**Link property**7  
Linking Controls 6, 7  
Locked Property 32  
LostFocus Event 31–32

## N

Navigator 5

## O

OLE Components 5

## P

Pixel Scan Control 8  
Properties  
    BackColor 28, 29  
    Caption 28, 29  
    DragIcon 30  
    DragMode 30  
    Enabled 30  
    Font 28, 31

ForeColor 31  
HelpContextID 32  
Locked 32  
PropertyName 28, 32  
PropertySource 28, 33  
Visible 33  
PropertyName Property 28, 32–33  
PropertySource Property 28, 32–33

## R

RegionSource 8  
RightButtonOption 13  
Routing Data between Controls 6  
**Runtime Linking** 7

## S

ScanFix Control 8  
Setup 5

Source of Image Data 7

## T

TextBridge Control 9  
TMS Display Control 6, 8, 11  
TMS File Control 7, 8, 11  
Tutorial  
Starting Out 11

## U

Using an ImageBASIC Widget 28

## V

Visible Property 33  
Visual Basic 4.0 11