

Pixel Translations Display ActiveX Control

User's Guide



DIAMONDHEAD SOFTWARE, INC.
1217 DIGITAL DRIVE STE. 125
RICHARDSON, TEXAS 75081
PHONE: (972) 479-9205
FAX: (972) 479-0219

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.

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,© 1993 - 1996 Pixel Translations, Inc. All rights reserved.

Company and product names mentioned in this documentation are trademarks or registered trademarks of their respective companies. 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 Configuration and Verification.....	3
Displaying an Image.....	6
Image Characteristics.....	6
Display Configuration.....	7
Scaling Images for Display.....	7
Caching of Images.....	9
Basic Image Manipulation.....	10
Zooming and Scrolling.....	10
Setting the Display Orientation.....	11
Grayscale for Display.....	12
Inverting Images.....	13
Enabling and Disabling Events.....	14
Chapter 2 : Controlling the Mouse	17
Mouse Functionality in ImageBASIC.....	17
Rubber Banding.....	17
Assigning Functions to the Mouse Buttons.....	18
Mouse Button Option Properties.....	19
Mouse Related Events.....	22
Chapter 3 : Regions in ImageBASIC	27
The Zoom Region.....	27
Zoom Region Properties.....	28
Scrolling and Zooming.....	28
Pan Window.....	31
The Working Region.....	32
Visual Selection of the Working Region.....	32
Programmatic Selection of the Working Region.....	34
Applying the Working Region.....	36
Chapter 4 : Printing from ImageBASIC	37
Overview of Printing.....	37
Image Scaling When Printing.....	37
Printer Selection and Setup.....	38
Standard Printer Dialogs.....	38
Programmatic Selection of a Printer.....	39
Printing a Single Page.....	40
Printing the Whole Image.....	40
Printing a Region of the Image.....	41

Advanced Printing Options.....	42
Print Jobs	42
Printing Multiple Images on One Page.....	44
Chapter 5 : Reference	47
Reference to Properties, Methods and Events.....	47
Appendix A : Error Reporting	103
Error Reporting and Trapping.....	103
The Error Event.....	104
Error Numbers	106
Error Numbers from Pixel Translations Libraries.....	108
Appendix B : Color Definition	115
File and Compression Formats.....	115
File Format Definitions.....	115
Compression Types.....	117
Color Limitations of File Formats	118
Comparative Charts of Colors Supported by File Format.....	118
Index	123

Chapter 1 : Introduction

Linking of ImageBASIC Controls

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 Pixel Display control cannot directly access image files on disk.
- An ImageBASIC File or Document control must read the image files into memory where they can be retrieved by the Display control.
- Therefore, the source of images for the Pixel Display control may be the Pixel File control or the TMS Sequoia File control. The communication between the controls is enabled by setting the Pixel Display control's **ImageDataSource** to specify the 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 **DisplaySource** property must specify the Pixel 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 Pixel Display control -- to the **Link** property of the source control. For example, for a Pixel Display control to accept image data from a Pixel File control, set the **ImageDataSource** property as illustrated:

```
PixDispl.ImageDataSource = PixFile1.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 Pixel Display control is performed by setting the Annotation control's **DisplaySource** property as shown here:

```
Annotel.DisplaySource = PixDispl.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.

Changes of Image Data

Each time the image data supplied by one ImageBASIC control changes, the recipient control will automatically refresh with the new image. A change in the image will occur when the File control opens a new file or changes to another page in the same image file. Changes in the supplied image data also occur in other controls when scanning, pasting from the clipboard, and other actions. Each time the image data that is being supplied changes, the recipient control's **ImageDataChanged** event occurs. The **ImageDataChanged** event has no parameters but simply occurs once each time the incoming image changes.

Licensing Configuration and Verification

In order to run, each ImageBASIC control must be able to verify the presence of a valid license token. These license tokens are stored on either a hardware key (shipped with each toolkit) or in a licensing database (the standard runtime distribution format).

Utilizing these tokens, licensing in ImageBASIC is based on enabling a set number of concurrent seats. Each license token allows a single PC to run any number of instances of a single control. For example, a single token for a Display will allow one workstation to concurrently run multiple applications, each of which employs any number of Display controls.

A unique token is required for each different type of ImageBASIC component that you have licensed:

- There is a special type of token for the Pixel Translations Display control, another for the TextBridge control, another for the ScanFix control, and yet more token types for each additional control. The 16-bit and 32-bit versions of each control are also licensed separately.
- Each one of these licenses also comes in two varieties, *runtime* and *development*.

As suggested by the names, runtime licenses are necessary for an executable to function, and development licenses are necessary to develop an application using ImageBASIC.

A design time license will function as a runtime license, removing the need to add runtime tokens for application testing during development.

Where the Licenses are Kept

ImageBASIC will find licenses stored in either one of two locations -- in a licensing database or on a hardware key.

- The hardware key is plugged into a parallel port on the computer using ImageBASIC and will be automatically found each time an ImageBASIC component is used.

Note: When using Windows NT, the parallel port must be configured using the Sentinel drivers that are shipped with ImageBASIC.

- The licensing database must be created on the site where it will be used and may not be moved from the location in which it is installed.

The inability to move a licensing database is one of its protection features. This attachment to a single location is formed when the

licensing database is first created. Although it may not be moved once created, the licensing database can be written to a network drive to which all the machines running ImageBASIC have access.

A file called IMGBASIC.INI must be on each of these networked machines. This file contains an entry pointing to the location of the licensing database. ImageBASIC can now search the licensing database for an available copy of each license it needs to run.

The licensing database may also be created on a local drive, activating ImageBASIC on only that one machine. The same INI file is still necessary to pinpoint the location of the database.

How the License Tokens are Used

As each application that uses ImageBASIC is initiated, or the control is loaded into the development environment, the ImageBASIC licensing server attempts to find the proper token for each component.

For example, if an executable has been developed that uses ImageBASIC to display existing images, and then calls on TextBridge to perform OCR on that image, that application must be able to find one *Display runtime license*, one *File runtime license* and one *TextBridge runtime license*.

- If the application is successful in finding these licenses, it will load normally and function exactly as it was programmed.

When the application finds these licenses and is thereby informed that the correct licenses are available, it simultaneously locks the licenses so that no other application can use the same licenses to run at the same time.

If two copies of these same licenses are available, another computer will be able to run the same application and will then lock the second license.

- If the application cannot verify the presence of the required tokens, the ImageBASIC components will not operate.

The **Active** property of the controls that did not find license tokens will be set to False. The state of the **Active** property can be verified during Form Load.

A runtime error that can be trapped during Form Load will also occur.

Licensing Token Release

When an application that has locked one or more tokens terminates normally, the tokens are released and can immediately be taken by another user if a network licensing database is being used. This is the process by which concurrent licensing for any number of seats may be enabled.

- If the application ends abnormally -- the user might reboot or a concurrently running Windows application might lock up -- then the release of the licenses is conditional on the network or disk operating system.
- For Novell networks, the default time for releasing a file lock after a connection is lost is 5 (five) minutes.

For other networks, your system administrator should be able to tell you how long the NOS takes to release the lock.

The system administrator should be able to change the default release time to satisfy your particular needs.

- If the licensing database has been installed on a stand-alone machine, the locks are immediately released and will again be available when the application is started again.

Displaying an Image

The Pixel Display control will display the image data that is provided from the control named in the **ImageDataSource** property. Each time the source control changes the image data that it supplied, the display control will be updated and the **ImageDataChanged** event will occur.

Selecting an Image for Display

To display an image file from disk, the Pixel Display control must be linked to a file access control. When the file access control loads an image file, the image will be immediately displayed, and the **ImageDataChanged** event of the Pixel Display control will occur. The following code segment links the two controls and then loads a disk file for display:

```
PixDispl.ImageDataSource = PixFile1.Link  
PixFile1.LoadFile "c:\images\1001.tif"
```

Once an image is displayed, the manner in which the image is displayed can be altered. "Basic Image Manipulation" on page 10 provides details on the available options which include rotating the image, inverting the displayed colors, and applying a gray scaling algorithm to improve the readability of most text documents.

Image Characteristics

Every time that an image is loaded for display in the Pixel Display control, several property values are updated with descriptive details of the image. These properties report characteristics such as the size and resolution of the image and its color definition. The image description properties are as follows:

ImageBitsPerSample	Reports the bit depth of each color sample in the currently displayed image. For bitonal images, this property will always report a value of one (1). For grayscale and color images, values of four (4) or eight (8) are also possible.
ImageHeight	Reports the number of pixels in the height of the original image. All references to regions of the image or image coordinates are specified in terms of these pixels, not the pixel height of the displayed, scaled copy of the image.
ImageSamplesPerPixel	Reports the number of samples defining the color of each pixel. Bitonal, grayscale, and paletted color images report a value of one (1), while RGB images report a value of three (3).

ImageWidth	Reports the number of pixels in the width of the image. All references to regions of the image or image coordinates are specified in terms of these pixels, not the pixel width of the displayed, scaled copy of the image.
ImageXRes	Reports the horizontal resolution of the image in dots per inch (DPI). For most images, the horizontal and vertical resolution will be identical.
ImageYRes	Reports the vertical resolution of the image in DPI.

Display Configuration

The programmatic interface to the Pixel Display control allows the developer control over how an image is presented to the user. All scaling, zooming, and display enhancements are performed through this control. This control also includes basic Working Region definition. Any portion of an image defined as a Working Region can be sent to another ImageBASIC control for further processing -- OCR, output to file, etc.

Once an image is displayed, the Pixel Display control offers access to these display features and functions:

- Grayscale the displayed image for improved readability and reduced eye strain
- Zooming to any portion of the image with scroll bars for navigation within the zoomed image
- Scaling the image to fit the display window fully or to the image width
- Inverting the colors of the image
- Orienting the image to the primary rotations of 0°, 90°, 180°, or 270°
- Printing the image or any portion of the image
- Defining a Working Region for output to another ImageBASIC control

Scaling Images for Display

Black and white images are usually scanned between 200 DPI (dots per inch) and 400 DPI. Color images are generally scanned at 200 DPI or less. An 8.5" X 11" image scanned at 300 DPI will have a height of 3300 pixels and a width of 2550 pixels.

When you display an image on your monitor, the resolution that you can achieve is a function of the capability of your monitor. It is rarely possible to display the entire

image in an accurate fashion. Usually the display area is smaller than the image, or the monitor is only capable of a lesser image resolution than would be necessary to capture the image completely.

Display software deals with this by scaling down the image. Given the size of the display window available, the actual size in pixels of the image data, and the resolution capabilities of the display monitor, the software will pick a scaling factor. It will then implement it by sampling the pixel content of each small region of the image, then throwing away a certain percentage of the pixels and representing each sampled region with a set of pixels chosen to best represent the sampled area.

The default in ImageBASIC is to present the image in the available window at the largest size possible that permits all image material to be displayed. When you first work with images, you might initially be surprised at the way the display scaling works. If you set up a window on your screen that has the approximate proportions of an 8.5" X 11" page, and display a regular letter size page image, it will seem very natural, displayed at the biggest size possible. If you now rotate the image (by changing the **Rotation** property), you will find that it is now reduced in size, taking up less of the screen and leaving a large blank area. This is because the scaling fits the image to the window in such a way as to display the entire image.

In the illustrations below, you can see the result of this practice when the same image is first shown upright, completely filling the display window, and then rotated 90°. After rotation, the entire image is still displayed, but a significant portion of the display window is left blank.

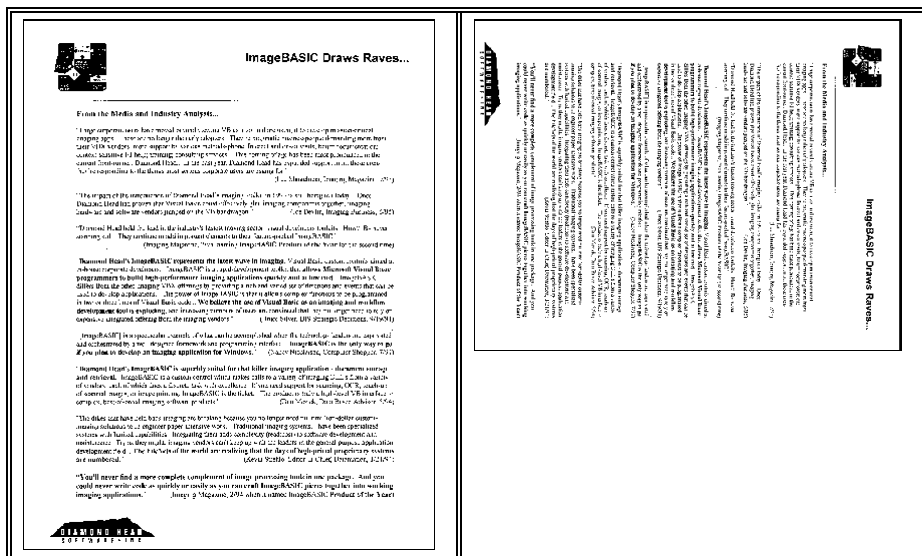


Figure 1. Scaling changes when rotating image.

The proportions, or aspect ratio, of the display window, and the fit of the display window with the aspect ratio of the image itself, dictate the maximum size of the display. If the image is presented such that the longer dimension of the image is squeezed into the shorter dimension of the window, then the displayed image will be scaled small and there will be unused area in the display window.

Caching of Images

ImageBASIC supports file caching, keeping multiple image files open in memory to display more quickly switching between images. Caching keeps the file open for a quicker second display. Caching usually has no negative consequences and can offer real speed benefits. Although caching the image provides performance benefits by reducing the time necessary to change views, problems can arise. A common problem arises if another tool manipulates an image and saves it to file. If the display control then reads from memory when changing its view, an unprocessed image can be retrieved, creating some confusion.

When displaying an image on the screen, it is first decompressed and held in memory as an RLE compressed bitmap. Whenever you zoom into an image the display engine refers back to the original image still held in memory, maps the coordinates of the Zoom Region onto the image, and passes the data held within those coordinates to the screen buffer for scaling and presentation on the screen.

If you rotate the image, then the display engine creates and holds in memory a rotated version of the same file to which it can apply coordinates. These concepts of image data processing will be of value in future chapters as we present some of the functions of ImageBASIC and start making more use of the coordinates.

Basic Image Manipulation

The most common changes that are made to an image while it is displayed have been simplified as much as possible in ImageBASIC. The frequently used options of zooming, rotating, inverting and grayscaling are all definable through the setting of just one property value for each.

Zooming and Scrolling

When you first load an image into a Pixel Display control, the entire image is loaded into memory, and the image is displayed so that all of it is visible in the Display control. The following properties and methods can be used to zoom in or out on the displayed image:

Mouse Options	The mouse buttons can be set to allow the user to select a region and zoom to it
ZoomIn Method	Zooms in by the percentage specified in the ZoomInOutChange property
ZoomOut Method	Zooms out by the percentage specified in the ZoomInOutChange property
ZoomTo Method	Zooms to the regions specified as parameters to the method call
ZoomPercent Property	Specifies the percentage zoom relative to the original image data; equivalent to ZoomRatio / 100).
ZoomRatio Property	Specifies the zoom ratio relative to the original image data

When the zoom region is changed by any method, two sets of properties report the coordinates of the displayed region in image pixels. The following properties reports the exact coordinates to which the zoom region is set:

ZoomTop
ZoomLeft
ZoomBottom
ZoomRight

The following properties report the coordinates of the Display window in image coordinates. These will differ from the Zoom... properties, above, if the aspect ratio of the selected region is different from the aspect ratio of the Display window.

DisplayLeft

DisplayTop

DisplayRight

DisplayBottom

Note: If enabled through the **ScrollBars** property, scroll bars will be displayed any time the display window is showing a zoomed portion of an image.

Setting the Display Orientation

Once the image for display is selected and called to the screen, there are a number of simple manipulations that we may wish to perform on the displayed image.

For example, it is often useful or necessary to display an image in a different orientation than it exists in the file. ImageBASIC makes this easy to accomplish using the **Rotation** property. The **Rotation** property can be set to rotate the image 0°, 90°, 180° or 270° relative to the incoming image. Incremental rotation of an image, such as at a 15° orientation, is not allowed.

The **Rotation** property may be set to these integer values:

0

90

180

270

Sometimes a scanned or faxed image is turned on its side, or is upside down, and the user needs a simple way to display it right side up. You can add this functionality to your program and attach it to a menu item or to a button with one line of code, such as the following one which will rotate the image an addition 90 degrees each time the button is clicked:

```
PixDispl.Rotation = (PixDispl.Rotation + 90) Mod 360
```

Grayscale for Display

Grayscale is an image enhancement feature that is extremely helpful in improving the contrast and readability of document image files while they are displayed. It is especially important when you are applying high scaling factors, such as when you are displaying images on a low resolution monitor or are displaying thumbnail images.

The basic approach for displaying typical document images is to sample small portions of the image and determine, on a binary basis, whether the sampled part of the image should be either black or white. With grayscale, ImageBASIC adds the ability to employ shades of gray instead of the simple black or white. This results in darker, more highly contrasted display, because around text characters in particular, many small white areas pick up darkness and become much more visible on the screen.

When you change to a grayscaled display, the image will often jump right off the screen. These two figures demonstrate the effect of activating **ScaleToGray**. As you can see, the figure on the right would be much easier to read and would cause less eye strain for the user.

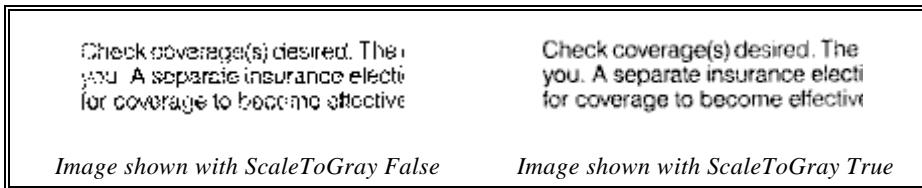


Figure 2. Image text with and without grayscale.

Enabling Grayscaled Display

The implementation of grayscale for display can easily be added to your program for automatic activation or for activation through use of a menu or a button. Setting a single Boolean property **ScaleToGray**, to True will enable grayscale.

```
PixDisp1.ScaleToGray = True
```

Note that grayscale will reduce display performance, so we recommend that the user always be permitted to disable the feature when maximum performance is required. The **ScaleToGray** property takes values of True and False.

Optimizing Grayscaled Display

When grayscaling is enabled, the **ScaleToGrayLevel** property is applied to select the type of scaling that is performed. **ScaleToGrayLevel** is an enumerated property that specifies the number of tones of gray that will be generated for the scaling operation. The options for this property are as follows:

- 0 Low
- 1 Medium
- 2 High

The lower values result in a less well defined grayscale image, but the display is calculated and loaded more quickly. Using more tones of gray results in greater image improvement and readability but is slightly slower.

```
PixDispl.ScaleToGrayLevel = 2
```

Inverting Images

Inverting an image causes all colors to be displayed as the opposite of their original definition. For example, all black pixels are displayed as white and all white pixels displayed as black. For grayscale and color images, the result is somewhat more complicated, but the effect is identical to that observed in film negatives; e.g. lighter colors appear darker and darker colors appear light.

Enabling Image Inversion

The Pixel Display control's **Invert** property toggles between the original display colors and their inverse. This is a Boolean property and changes the colors of all image data passed from the display control. For example, **Invert** is enabled and the image data is saved through the File control, the disk image will be inverted from the original.

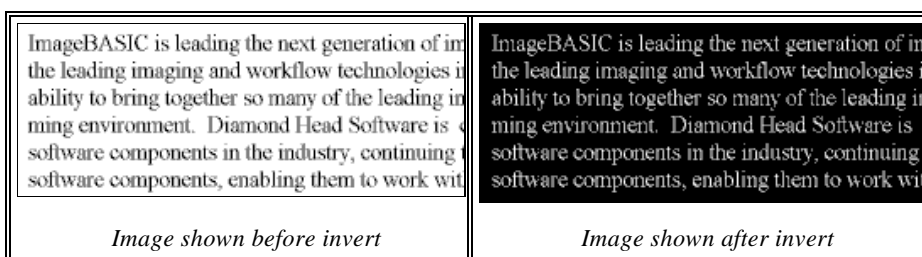


Figure 3. Image text with and without invert.

To display an image inverted, set the **Invert** property to True:

```
PixDispl.Invert = True
```

To display an image with its original colors, set the **Invert** property to False:

```
PixDispl.Invert = False
```

Export Image Data

The Pixel Display control will export a DIB of any image that it is displaying and supply the Windows handle to that bitmap. The method used to export the image data can convert the image to any specified height and width, quickly creating thumbnails or full resolution images which are then available to other, non-ImageBASIC controls.

For example, a Picture control could display a thumbnail of the image in a Pixel Translations Display control, as shown below for Visual Basic:

```
Picture1.Picture = PixDispl.GetScaledPicture(75, 95)
```

The parameters to the **GetScaledPicture** method are the pixel width and height of the DIB that is to be created. The return value from the method can be passed to the Picture property of another Visual Basic control:

```
Picture = PixDispl.GetScaledPicture( width, height)
```

Enabling and Disabling Events

Performance of many applications can be slightly optimized by disabling the more frequently occurring display events. Other applications will require these events to adequately process all incoming data.

The **EventMask** property enables the developer to enable only those events that are needed, possibly improving application performance.

- The **EventMask** property is a Boolean array property.
- The indices for the elements are the *eventIDs* listed in the table below.
- To enable an event, set its **EventMask** index to *ibEventMaskEnable*.
- To disable an event, set its **EventMask** index to *ibEventMaskDisable*.
- The *ibEventMaskEnable* and *ibEventMaskDisable* constants are defined within the control.

For example, to enable the **Paint** event, set the **EventMask** property as shown here:

```
PixDispl.EventMask(dspEventPaint) = ibEventMaskEnable
```

For example, to disable the **MouseMove** event, set the **EventMask** property as shown here:

```
PixDispl.EventMask(dspEventMouseMove) = ibEventMaskDisable
```

The following table lists the events that can be individually enabled and disabled through the **EventMask** property. This table also shows the constant defined for each event. This constant should be used in the *eventID* parameter to the property, as shown on the previous page.

Event	eventID Constant
Click	dspEventClick
DblClick	dspEventDblClick
Error	dspEventError
FilesDropped	dspEventFilesDropped
ImageDataChanged	dspEventImageDataChanged
MouseDown	dspEventMouseDown
MouseMove	dspEventMouseMove
MouseUp	dspEventMouseUp
Paint	dspEventPaint

Chapter 2 : Controlling the Mouse

Mouse Functionality in ImageBASIC

ImageBASIC imaging applications, like most GUI (Graphical User Interface) applications, will usually make use of the mouse as their primary user input mechanism. This means the programmer must have very sophisticated control over how the user can use the mouse. Mouse functionality is one of the hardest programming issues when you are dealing in a low level programming environment like C. ImageBASIC has several features to simplify implementing mouse functionality.

There are two ways to implement mouse functionality with ImageBASIC

- Using the preprogrammed functions of the Mouse Button Options properties.
- Adding operational routines to the Mouse Events

Combinations of the two are typical, such as using a pre-programmed function like a rubber band, then making appropriate use of the image coordinates.

A major advantage when implementing mouse functionality with mouse events is that when your code is called, it is passed the mouse coordinates both as screen coordinates and as image coordinates. This will save you much time, calculation and code since it takes the responsibility for image coordinate mapping away from the programmer.

Rubber Banding

Most of the mouse options involve the use of a common Windows technique called *Rubber Banding*. Rubber Banding is positioning the mouse over a location that is intended to be a corner of a region and then depressing one of the mouse buttons. The mouse is then moved while the button is held down and an animated rectangle is drawn from the point where the button was first pressed to the current location of the mouse pointer. When the mouse button is released, the region is set.

As this rectangle is being drawn, the application can force it to match a certain width-to-height ratio, or the rectangle can be drawn without limitations on this aspect ratio. The first option is known as proportional rubber banding, and the second is standard rubber banding. As will be seen in the section *Assigning Functions to the Mouse Buttons* on page 18, both of these options are available when the mouse is used to define either the Zoom Region or the Working Region.

Standard Rubber Band

Often, an ImageBASIC programmer will wish to perform an operation on an area of the image that has no pre-determined aspect ratio. An example could be to provide the user with the ability to crop an image or split out a portion for printing or for saving as a separate file. This would be a typical time to use the Standard Rubber Band technique. When Standard Rubber Banding is employed as an option for either the left or right mouse button, the user will be able to select any chosen rectangular portion or region of the displayed image using that mouse button.

Proportional Rubber Band

In situations where a specific area of the image should be chosen and a desired shape for the selected portion is known, then ImageBASIC offers a different version of the rubber band. The Proportional Rubber Band will allow the user to select an area of the image in much the same way as before, but the proportions of the rectangular region will match the proportions of the display window.

Assigning Functions to the Mouse Buttons

A common design choice is to set the left mouse button to select the Zoom Region and the right mouse button to select Working Region. The exact behavior of the left and right mouse buttons may be set independently through the **LeftButtonOption** and **RightButtonOption** properties, but the same options are available for both.

If one of the Zoom Region selection options is selected for a mouse button, the area that is selected by dragging out a region will be scaled to fill the display window and the properties that define this region will be populated with new values. Refer to "Zoom Region Properties" on page 28 for details on these properties.

If one of the Working Region selection options is selected for a mouse button, the Working Region properties that define the region will be populated with the new values. The Working Region may be processed, printed, OCR'ed, etc., independently of the remainder of the image. For details, refer to Chapter 5 : The Working Region.

When they are used, the mouse buttons can also activate certain ImageBASIC events called the Mouse Events. These events can be used to hold code for program operations that you wish to associate with the actions of the mouse buttons.

Mouse Button Option Properties

The two mouse buttons may be assigned different, pre-programmed functionality by setting the two Mouse Button Option Properties. The two Mouse Button Option Properties are **LeftButtonOption** and **RightButtonOption**

Drawing Styles

All of these options will draw a rectangle on the display window as the mouse is moved. By setting the **LeftMouseBoxStyle** and **RightMouseBoxStyle** properties, the developer can specify whether the rectangle is drawn hollow or filled. Both of these properties are enumerated and have the following options:

- 0 None
- 1 Hollow
- 2 Solid

0--None does not visibly indicate the position of the Working Region.

1--Hollow indicates the Working Region by drawing a single, solid line around all sides.

2--Solid draws the Working Region with a transparent, inverse color rectangle over the entire region.

Mouse Button Options

The values for the **LeftButtonOption** property and the **RightButtonOption** property can be independently set according to the following:

- 0 Normal
- 1 Rubber Band
- 2 Proportional Rubber Band
- 3 Zoom
- 4 Proportional Zoom
- 5 Drag Region
- 6 Drag Tool

0--Normal

Option *0--Normal* can trigger all of the mouse events, but no other action, such as setting the Working Region, is performed. This option allows the developer the most freedom, but at the cost of limited preprogrammed functionality.

1--Rubber Band

1--Rubber Band paints a rectangle from the point where the **MouseDown** event took place to wherever the mouse is moved. When the button is released, the selected portion of the image is set as the **Working Region**

The Working Region Properties are populated with the new values and , and the region coordinates can be used in your code. This methodology is often employed in passing data to another ImageBASIC control.

If the mouse button is released outside the Image Display Window, drawing the rectangle and setting the **Working Region** coordinates are both abandoned, with no changes made to the image.

2--Proportional Rubber Band

2--Proportional Rubber Bands similar to the Standard Rubber Band option, except that the rectangle drawn by the mouse movement is drawn with the same aspect ratio no matter how the mouse is moved. The aspect ratio always matches the aspect ratio of the display window.

3--Zoom

This option paints a standard rubberband in the display window. When the **MouseUp** event is fired, the Zoom Region properties are set. See "Zoom Region Properties" on page 28 for details. The selected region will then be scaled to fit the display window as well as possible. For details on scaling images to fit the display window, refer to "Scaling Images for Display" on page 7.

4--Proportional Zoom

This option, like *3--Zoom*, sets the Zoom Region and automatically redraws the display window. The advantage of Proportional Zoom is that the rectangle that is drawn by the mouse is forced to be the same aspect ratio as the display window.

By forcing the optimum aspect ratio, Proportional Zoom solves a problem with proportions that crops up when using the standard rubber band method for zoom.

With the standard zoom, if the window is twice as high as it is wide and you draw a rubber band that is wider than it is tall, then the image display engine has to make an intelligent guess as to how it should adjust that information to make it fit.

ImageBASIC will settle this choice in favor of giving you more information rather than less, and the area that you will get will be the largest section of the image that will fit in the Image Display Window, as determined by fitting the most difficult dimension. Although the area will include the area selected, it will not be identical to the selected area -- it will usually be larger.

4--Proportional Zoom restricts the dimensions of the region which may be dragged out by the user by maintaining a fixed aspect ratio between the width and height of the region. The aspect ratio at which the region is drawn will match the aspect ratio of the Pixel Display window.

5--Drag Region

This option allows the user to move, but not resize, the current Working Region. To move the region, simply depress the appropriate mouse button while over the region and then move the mouse cursor without releasing the button. The Working Region will be updated as the cursor is moved.

5--Drag Region is designed primarily for use in the Pan Window where the user can scroll through a parent Display control's image by moving the Working Region in the Pan Window control. See "Pan Window" on page 31 for details on this procedure.

6--Drag Tool

When this option is selected, the user can click and hold down the mouse button. When the mouse is moved while the button is held depressed, the image will be scrolled along with the cursor. This option was designed to maintain approximately the same image position for the cursor wherever it moves on the screen. If the cursor is held near the edge of the Display control with the button depressed, the image will continue to scroll in that direction.

When the image is scrolled, the **ZoomTop**, **ZoomLeft**, **ZoomRight** and **ZoomBottom** properties are updated with the current image coordinates describing the displayed region.

Mouse Related Events

Several events of the Pixel Display control are triggered by mouse actions. These events are the standard Windows mouse events, but ImageBASIC includes additional parameters to the events to address some of the special needs of document imaging applications. The mouse related events are as follows:

- Click
- DblClick
- MouseDown
- MouseUp
- MouseMove
- DragDrop
- DragOver

All mouse events are fired by both mouse buttons, without regard to the option selected for that button. For example, the Click event will occur any time that either mouse button is depressed and then released.

Click Event

The Pixel Display control's **Click** event is called when one of the mouse buttons is pressed and released on the window. There are no parameters to this event.

DblClick Event

Triggering of the **DblClick** event is identical to the **Click** event except that the button must be double clicked as defined in the Windows configuration. There are no parameters to this event.

MouseDown Event

The **MouseDown** event is called when either mouse button is depressed while in the boundaries of the Pixel Display window. Parameters to the **MouseDown** event are as follows:

- Button
- Shift
- ScreenX
- ScreenY
- ImageX
- ImageY

MouseMove Event

The **MouseMove** event is called whenever the mouse is moved inside the boundaries of the Image Display Window. Care should be taken about how complicated a function should be implemented in the **MouseMove** event. This function will be called constantly during mouse sweeps and if much processing time is taken by your code it will adversely affect the performance of the system. If you place code in this event you may want to deactivate it when it is not immediately in use to avoid unnecessary performance degradation.

Parameters to this event are as follows:

Button
Shift
ScreenX
ScreenY
ImageX
ImageY

MouseUp Event

The **MouseUp** event occurs when the user releases either button while in the display window. Parameters to this event are as follows:

Button
Shift
ScreenX
ScreenY
ImageX
ImageY

DragDrop Event

This event occurs when a control is dragged over the Pixel Display window and released.

The parameters to this event are as follows:

Source
X
Y

DragOver Event

This event occurs when the mouse is used to drag any control over the Pixel Display window. The parameters to this event are as follows:

Source

X

Y

State

Mouse Event Parameters

In the discussion of Mouse Related Events on page 22, the parameters for each event are listed. The following paragraphs detail these parameters and the information that they provide.

The event parameters supplied by the ImageBASIC mouse events are as follows:

Button

Shift

ScreenX

ScreenY

ImageX

ImageY

Source

The *Button* parameter reports an integer indicating which button was pressed or released:

- 1 Left Button
- 2 Right Button

The *Shift* parameter holds the Control-and-Shift key states at the time the event occurred. The parameter is a bit flag, so you should use the OR operand to perform your testing.

- 1 Shift
- 2 Control
- 3 Shift & Control

The *ScreenX* and *ScreenY* parameters report the pixel location of the mouse cursor in the screen window for the Mouse Event.

The *ImageX* and *ImageY* parameters report the pixel location of the mouse cursor relative to the unrotated and unscaled image.

The *Source* parameter supplies the Name of the control that was dropped onto the display window.

The *State* parameter reports an integer specifying the movement of the cursor:

- 0 Enter -- The source control is being dragged into this control
- 1 Leave -- The source control is being dragged out of this control
- 2 Over -- The source control has moved from one position to another

Chapter 3 : Regions in ImageBASIC

The Zoom Region

The **Zoom Region** is the portion of the entire image that is currently displayed by the Pixel Display control. The Zoom Region defaults to the entire image, but can be modified by a number of means:

- The mouse may be configured to allow the operator to drag out a region that will be zoomed to fit the window. The configuration of the mouse buttons is controlled through the **LeftButtonOption** and **RightButtonOption** properties.
- Three methods are available to fit the image to the display window. These methods are **FitToWindow**, **FitToWidth**, and **FitToHeight**.
- Two additional methods are available to zoom into and out of the image by a specified percentage. These methods are **ZoomIn** and **ZoomOut**.
- When an image is zoomed, scroll bars are displayed on the window that allow movement within the image.
- A second Pixel Display control can act as a Pan Window. A Pan Window displays a thumbnail of the main window and allows the user to move the main window's displayed region.
- The region is described by several properties which can be directly manipulated to change the displayed region. This set of properties is collectively referred to as the Zoom Region Properties.

As you can see, several interrelated levels of control are available for Zoom Region control. These options allow the developer the most possible freedom in application design while still simplifying implementation a great deal. Whenever any of these options is used to change the Zoom Region, the properties that describe this region are updated to reflect the actual display.

Zoom Region Properties

Any time an image is displayed by a Pixel Display control, two sets of properties describe the displayed region. One set describes the image coordinates of the region selected by the user to display:

ZoomTop
ZoomLeft
ZoomBottom
ZoomRight

The second set describes the coordinates of the Display window itself, in image coordinates. Because the image does not necessarily completely fill the Display window, these coordinates will sometimes be larger than the dimensions of the image, and they can also be negative:

DisplayLeft
DisplayTop
DisplayRight
DisplayBottom

Scrolling and Zooming

When you first load an image into a Pixel Display control, the entire image is loaded into memory, and the image is displayed so that all of it is visible in the Display control. The following properties and methods can be used to zoom in or out on the displayed image:

Mouse Options	The mouse buttons can be set to allow the user to select a region and zoom to it
ZoomIn Method	Zooms in by the percentage specified in the ZoomInOutChange property
ZoomOut Method	Zooms out by the percentage specified in the ZoomInOutChange property
ZoomTo Method	Zooms to the regions specified as parameters to the method call
ZoomPercent Property	Specifies the percentage zoom relative to the original image data; equivalent to ZoomRatio / 100).
ZoomRatio Property	Specifies the zoom ratio relative to the original image data

Note: If enabled through the **ScrollBars** property, scroll bars will be displayed any time the display window is showing a zoomed portion of an image.

Zooming and Fitting the Image to the Display Window

Rather than specifically set the zoom region coordinates, the **ZoomIn** and **ZoomOut** methods provide a simple way to offer the user a Zoom In / Zoom Out capability. These methods can be called in a radio button, a mouse event, or to some other user interface.

When called, these methods change the Zoom Region by a percentage of the image's height and width. This percentage is specified in the **ZoomInOutChange** property.

On initial loading, the Pixel Display control always attempts to display any image as large as possible, given the size of the window and the height-to-width ratio (called the aspect ratio) of the image and the display window. The default display of the image always shows the entire image scaled as large as possible. This may cause the right or bottom edge of the window to be left blank if the aspect ratio of the image does not match the aspect ratio of the display window.

The operator may zoom into the image using any of several different techniques:

- 1) Set one of the mouse button to a zoom option:

The **LeftButtonOption** or **RightButtonOption** property may be set to *3--Zoom* or *4--Proportional Zoom*

- 2) Execute one of the methods to zoom in or out of the image by a fixed percentage:

ZoomIn	Zooms into the image (enlarges the image) by a fixed percentage. The percentage is set in the ZoomInOutChange property which defaults to 15 (fifteen).
---------------	---

ZoomOut	Zooms out of the image by a fixed percentage. The percentage is set in the ZoomInOutChange property.
----------------	---

- 3) Execute the **ZoomToRegion** method, specifying the image coordinates to which the image should be zoomed:

`PixDisp1.ZoomToRegion left, top, right, bottom`

This method will cause the image to be zoom as nearly as possible to the specified coordinates. Any differences between the specified coordinates and the actual Zoom Region will be because the Display control will fit the specified region as well as possible in the window, possibly showing more of the image.

- 4) Three methods are available that scale the image to fit the widow in different fashions:

FitToWindow	Scales the image to fit it to the display window as described in the paragraph above.
FitToWidth	Scales the image to fill the entire width of the display window, allowing the bottom portion of the image to extend beyond the display window, if necessary. If the image does extend beyond the window, scroll bars will be displayed.
FitToHeight	Scales the image to fill the entire height of the display window, allowing the right side of the image to extend beyond the display window, if necessary. If the image does extend beyond the window, scroll bars will be displayed.

Note: While the image is enlarged, ImageBASIC defaults to displaying scroll bars on the Display window so that the operator can navigate within the zoomed image. For more information on navigation within the zoomed image, refer to the following sections:

"Enabling and Disabling the Scroll Bar\$ on page30

"Pan Window" on page31

Enabling and Disabling the Scroll Bars

ImageBASIC displays scroll bars whenever the image in the Pixel Display control is zoomed (i.e., is displaying less than the entire image. The developer has the option of separately enabling the vertical and horizontal scroll bars or of disabling both.

ScrollBars is an enumerated property that specifies how the scroll bars are displayed when the image is zoomed.

The options for the**ScrollBars** property are as follows:

- 0 No Scroll Bars
- 1 Vertical Scroll Bar Only
- 2 Horizontal Scroll Bar Only
- 3 Both Scroll Bars

When the scroll bars are enabled and the image is zoomed, using the scroll bars to navigate within the image will update the Zoom Region properties each time the image is moved.

Pan Window

A Pan Window is a second display window that displays a thumbnail of the entire image in the main viewing window.

- When the main viewing window zooms into a portion of the image, the Pan Window's Working Region coordinates are set to match the main window's Zoom Region coordinates. Changing either set of coordinate properties causes the other window to update.
- This highlighted region in the Pan Window may be dragged around using the mouse when the **LeftButtonOption** or **RightButtonOption** property is set to *5--DragRegion*.
- When the Pan Window's region is moved, the displayed region of the main image will move with the region on the Pan Window.

Any Pixel Display control can be made into a Pan Window through the following steps:

- 1) Link the Pan Window to the main display window by setting the Pan Window's **ImageDataSource** property to the main display control.
`PixDispPan.ImageDataSource = PixDispMain.Link`
- 2) Set the Pan Window's **ThumbnailByteSize** property to specify the image data size of the thumbnail displayed in the Pan Window:
`PixDispPan.ThumbnailByteSize = 5000`
- 3) Set the Pan Window's **DisplayMode** property to *2--Pan Window*
`PixDispPan.DisplayMode = 2`
- 4) Set one of the mouse buttons on the Pan Window to *5--Drag Region*
`PixDispPan.LeftButtonOption = 5`

The Working Region

ImageBASIC offers the ability to read, write, examine, modify and recognize image data with very little coding. In most cases, a portion of the image known as the **Working Region** is the only part of image that is of interest. The Working Region does default to the entire image, but the region is frequently changed to define a smaller section of the image.

By setting this region to a smaller subset of the image, portions of the image can be manipulated or passed to another ImageBASIC control for OCR or other processing. The Working Region may be set interactively by the mouse user or through code that sets the properties that define the Working Region.

Visual Selection of the Working Region

"Chapter 2 : Controlling the Mouse" on page 17 details how the mouse can be configured to select the Working Region. Briefly, if the **LeftButtonOption** or **RightButtonOption** property is set to *--Rubber Band*, the Working Region properties will automatically be set when a region is selected by holding down that mouse button and dragging out a rectangle. The portion of the image that was selected by the mouse will be specified in unscaled image coordinates in the Working Region properties:

RegBottom

RegLeft

RegRight

RegTop

After the Working Region is selected, either by mouse interaction or by explicitly setting the Working Region properties, detailed below, this region may be printed or processed by another ImageBASIC control independent of the rest of the image.

Enable Visual Selection of the Working Region

The options for both the **LeftButtonOption** and the **RightButtonOption** properties are as follows:

- 0 Normal
- 1 Rubber Band
- 2 Proportional Rubber Band
- 3 Zoom
- 4 Proportional Zoom
- 5 Drag Region

1--Rubber Band and *2--Proportional Rubber Band* both allow the visual definition of the Working region. The only difference between the options is that *2--Proportional Rubber Band* forces the region that is selected to match the aspect ratio of the display window.

Highlight the Working Region When Visually Defining

When a Working Region is selected by dragging out a region using the mouse options described above, region may be visually indicated on the display window or it may be left unmarked. The default behavior, as assigned to the right mouse button, is to show the Working Region in inverted colors.

This behavior may be modified by changing the **LeftMouseBoxStyle** or **RightMouseBoxStyle** property according to the following enumerated list:

- 0 None
- 1 Hollow
- 2 Solid

0--None makes no visible change to the displayed image.

1--Hollow draws a solid, single line around the perimeter of the Working Region as it is being drawn.

2--Solid shows the Working Region in inverse colors.

Programmatic Selection of the Working Region

The image coordinates that define the Working Region are specified by four properties. These properties may be set either in code or using the mouse. To set them with the mouse, one of the mouse buttons must be set to a rubber band option using the **LeftButtonOption** or **RightButtonOption** property. The Working Region properties are as follows:

RegBottom

RegLeft

RegRight

RegTop

RegBottom Property

Specifies the lower edge of the Working Region. The coordinate supplied by this property is the number of pixels from the top edge of the original image (i.e., the image file, not the scaled display copy of the image).

RegLeft Property

Specifies the left edge of the Working Region. The coordinate supplied by this property is the number of pixels from the left edge of the original image (i.e., the image file, not the scaled display copy of the image).

RegRight Property

Specifies the right edge of the Working Region. The coordinate supplied by this property is the number of pixels from the left edge of the original image (i.e., the image file, not the scaled display copy of the image).

RegTop Property

Specifies the upper edge of the Working Region. The coordinate supplied by this property is the number of pixels from the top edge of the original image (i.e., the image file, not the scaled display copy of the image).

Assuming the image in question has been scanned at 300 DPI, the property values shown below will define the **Working Region** as an area one inch wide and three inches tall starting one inch from the top and left edges of the image. This small portion of the image could be passed to an OCR engine to extract indexing information for storage and retrieval of the image file.

```
PixDispl.RegLeft = 300
```

```
PixDispl.RegTop = 300
```

```
PixDispl.RegRight = 600
```

```
PixDispl.RegBottom = 1200
```

Highlight the Working Region

A Working Region is always defined when an image is displayed, but defaults to the entire image when each image is first opened. When a Working Region is defined by dragging out a region using one of the mouse buttons (refer to Visual Selection of the Working Region on page 32), the default behavior of the Pixel Display control is to highlight the Working Region in inverted colors.

After a Working Region has been defined either by explicitly setting the Working Region coordinate properties or by using the mouse to draw the region, it may be hidden and displayed at any time using the **WorkingRegionStyle** property.

WorkingRegionStyle is an enumerated property with the following options:

- 0 None
- 1 Hollow
- 2 Solid

0--None makes no visible change to the displayed image.

1--Hollow draws a solid, single line around the perimeter of the Working Region.

2--Solid shows the Working Region in inverse colors.

Fill the Working Region

The Working Region may be filled in white or black to remove or hide the image data. Unlike using annotations to fill or cover a region, filling the Working Region through the display control's method will permanently destroy any existing image data in the region. Of course, a previously saved copy of the image may be reloaded to restore the original image.

To color the Working Region either all white or all black, execute the **FillWorkingRegion** method. This method accepts a single enumerated parameter which specifies the color of the fill:

- 0 Fill Black
- 1 Fill White

Call this method as shown below to color the entire Working Region black:

```
PixDispl.FillWorkingRegion 0
```

Analyze the Working Region

The relative density of colored pixels in the Working Region may be determined. This function was created for those users who needed a quick way to recognize separator sheets and microfilm image blips.

The **GetRegBlackPercent** method analyzes the Working Region and determines what percentage of the region is comprised of black pixels. Because of the rounding errors in the function, always expect to get at least 1% returned by the method.

One possible use of this function is the identification of separator sheets. Separator sheets may be printed with a single square of black in one corner and a bar coded index in the middle of the page. An application can select a region that is within the expected location of black square and text the pixel density in that region for all pages. If this is a separator sheet, the value will be very high. If a separator page is found, the application searches for and decodes the bar code.

```
intPercent = PixDisp1.GetRegBlackPercent
```

Different types of image data generally return values in specific ranges. A region of normal text usually returns a value of 20% to 30%, bar codes return values of 40% to 60%, and pictures may vary from 15% to 70% depending on their complexity.

If a region is tested on a color image, the image data will be converted to bitonal before the density is calculated. Analysis of pixel density in color images can be unpredictable because of the thresholding that is applied.

Applying the Working Region

When image data is passed from the Pixel Display control to another ImageBASIC control -- for example, to a TextBridge control for OCR or to a TMSSequoia File control for saving -- either the entire image may be used by the recipient control, or only the Working Region may be used.

Each of the ImageBASIC controls that can take advantage of the Working Region is designed with complementary features:

- When linked to a Display control, the File control can save the Display control's entire image through the **Save** method or can save only the Working Region through the **SaveRegion** method.
- When linked to a Display control, the recognition controls can process the entire page or only the Working Region through the **OCRPage** and **OCRRegion** methods, respectively.

Chapter 4 : Printing from ImageBASIC

Overview of Printing

Adding printing functionality to your ImageBASIC programs is as easy as adding scanning. The basic printing functionality sends a single image to the printer as a single print job. There is a set of more powerful printing capabilities that permit you to size multiple images and "paste them up" on a single page sent to the printer.

ImageBASIC uses the standard Windows interface and Windows drivers to communicate with the printer. In this interface, we are sending uncompressed image data to the printer, and this can affect performance. Even with only a modest CPU, generating the print file does not in itself take up much time. Sending the data across to the printer will take time if your printer does not have much memory or if you are communicating with your printer by serial cable. A parallel printer or Ethernet connection will be significantly faster.

All printing operation default to use the Windows Print Manager to queue the information on disk and not require you to wait for the printer to accept it all. This can be changed only through the Print Manager or Control Panel.

Image Scaling When Printing

ImageBASIC sends image data to the printer in a fashion analogous to how the data is sent to a display window: it is automatically scaled to fit. If you are sending a region of the image to the printer, then it will be scaled to print as large as possible. This will be done by fitting the hardest dimension. If the aspect ratio (proportion of width to height) of the Working Region is the same as that of the printer output, then the selected region will use the entire paper.

If, for example, the paper is 8.5" X 11", and the image is 4.25" X 3", then ImageBASIC will print the image at 8.5" X 6". This is the same technique of scaling used for display, as discussed in *Scaling Images for Display* on page 7.

Adjusting for Laser Printer Margins

Most laser printers have dead space of 1/4 inch around the perimeter of the page where they do not print. You may have had the experience of laying out a document and having the printer driver tell you that you were printing too close to the edge of the page. In order to avoid having this problem cause a loss of image data, we have built in a scaling factor into ImageBASIC to correct for this.

If you are printing an image that is 8.5" X 11", ImageBASIC will automatically scale it down to 8" X 10.5", so that it will print in its entirety. This permits the laser printer to maintain its 1/4 inch border at the top, bottom, right and left edges without compromising the image data.

Printer Selection and Setup

Printer selection can be accomplished either through standard dialogs or through property settings. The **SelectPrinter** method opens a standard printer selection dialog. Alternatively, several properties specifying the print driver, the print port and the printer hardware may be explicitly set by the developer.

Standard Printer Dialogs

The printer selection and setup dialogs used by the Pixel Display control are standard Windows dialogs. The exact information displayed in these dialogs is controlled by Windows based on the printers installed on the local system and the capabilities of those scanners.

Printer Selection Dialog

When called, the **SelectPrinter** method will display a standard printer selection dialog, allowing the user to select from all currently installed printers. A typical dialog is shown below:

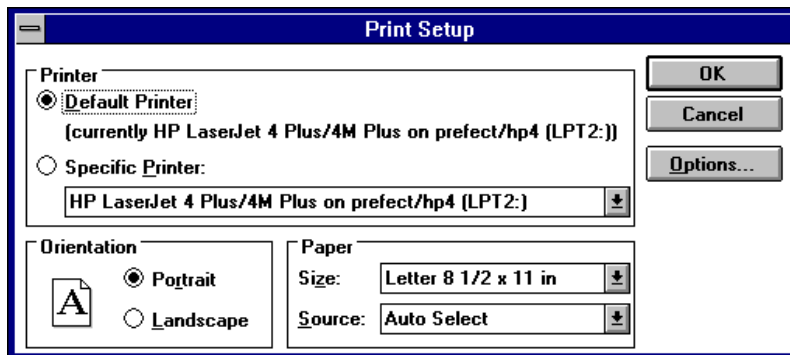


Figure 4. Typical printer selection dialog box.

Programmatic Selection of a Printer

If you prefer to limit the printing options available to the user, or your application must automatically select from one or more printers, several properties are available to select the printer. Print setup cannot be performed programmatically, but any changes that are made through the setup dialog will be maintained throughout a single instantiation of an application.

Programmatic printer selection is based on the following properties:

PrinterPort

PrinterDevice

PrinterDriver

The **PrinterPort** property specifies the port that will accept the printer commands. This property must specify the full port name:

```
PixDispl.PrinterPort = "LPT1"
```

The **PrinterDevice** property must specify the Windows name for the printer hardware. This name may be found in the Print Manager or printer selection dialog. A typical setting for this string property is as follows:

```
PixDispl.PrinterDevice = "HP LaserJet 4 Plus/4M Plus"
```

The **PrinterDriver** property specifies the name of the printer control language driver file with no path and no extension, as shown below:

```
PixDispl.PrinterDriver = "HPPCL5E"
```

If any of these properties is set to an invalid value, or the application cannot verify the presence of the specified device, the Pixel Display control will use the Windows default printer.

Printing a Single Page

The basic printing functions of the Pixel Display window allow printing of either the entire image or a portion of the image. The portion of the image that is printed is the Working Region.

Printing the Whole Image

The **PrintImage** method sends the entire image in the Pixel Display window to the printer:

- This method always prints one entire page, even if the Working Region has been set or one page of a multi-page image file is being displayed. The image will be printed as it is displayed -- inverted, rotated, etc.
- When sent to the printer, the image will be scaled as large as possible to fit the page size. To print the image at a certain scaling factor, use the virtual print page capabilities as discussed in the section **Printing Multiple Images on One Page** on page 44.
- Used by itself, the **PrintImage** method will send the current image as the only page in a new print job. Refer to the section **Print Jobs** on page 42 for details on the creation of multiple page print jobs.

For example, the following code segment will load all pages of a multiple page image file and print each page. Each page will be sent to the printer as a separate job so that, if any problems are encountered, only the single problem page must be printed again:

```
        ' declare necessary variables
Dim intLoop as Integer

        ' create the necessary data link
PixDispl.ImageDataSource = PixFile1.Link

        ' load an image file
PixFile1.LoadFile "c:\images\file0001.tif"

        ' loop through all pages of the file and
        ' print each as it is displayed
For intLoop = 1 to PixFile1.PageCount
    PixFile1.PageIndex = intLoop
    PixDispl.PrintImage
Next intLoop
```

Printing a Region of the Image

The **PrintRegion** method sends the image data defined as the Working Region to the printer. If no Working Region has been specified, the region defaults to the entire image. For details on selecting a Working Region, refer to "The Working Region" on page 32.

- The region will be printed as it is displayed; e.g., if **Invert** is True, the printed image will be inverted; if the image is displayed rotated, it will be printed rotated.
- When sent to the printer, the image region will be scaled as large as possible to fit the page size. To print the image at a certain scaling factor, use the virtual print page capabilities as discussed in the section "Printing Multiple Images on One Page" on page 44.
- Used by itself, the **PrintImage** method will send the current image region as the only page in a new print job. Refer to the section "Print Jobs" on page 42 for details on the creation of multiple page print jobs.

For example, to print the top half of an image, the following routine may be run:

```
Private Sub PrintTopofImage()  
    ' set the Working Region to the top half of the  
    image  
    PixDispl.RegTop = 0  
    PixDispl.RegLeft = 0  
    PixDispl.RegRight = PixDispl.ImageWidth  
    PixDispl.RegBottom = (PixDispl.ImageHeight / 2)  
    ' print the selected region  
    PixDispl.PrintRegion  
End Sub
```

Advanced Printing Options

Two advanced printing options are available -- sending multiple pages to the printer as a single print job, and printing multiple images on a single page.

The basic printing functions send a single page, whether it contains the entire image or just a portion, as its own print job. Some circumstances require the user to compose a print job consisting of multiple pages that are sent to the printer as a batch. All of the component pages of a print job are kept grouped together, even on a network, where various people are competing for a printer. This keeps the job from being interrupted by another user's print request.

The Pixel Display control can build up a single print page that contains data from several different images. This one page can then be sent to the printer as a job or it can be included as part of a larger print job. While it is being created, the single page is generally referred to as *virtual print page*.

Print Jobs

To help prevent the mixing of print pages on a shared printer or to maximize printer performance, a single print job composed of multiple pages may be created within ImageBASIC. After initializing the new print job, all pages printed from ImageBASIC will be held until the print job is completed. At that time, all of the pages will be sent to the printer as a single print job.

The following methods are used to start and end a print job:

StartPrintJob	Initiates a new print job and holds all pages printed from ImageBASIC in the print spool until the job is completed, and all pages can be set as a single job.
---------------	--

`PixDispl.StartPrintJob`

After this method is executed, the Display control can load and print any number of images or image regions. The only limitations on the number of pages and complexity of the print job are the memory and drive space on the PC that is required to hold the entire print job.

The print job can include image pages printed using the **PrintImage** and **PrintRegion** methods as well as virtual print pages as described in the section "Printing Multiple Images on One Page" on page 44.

EndPrintJob	Signals completion of the print job and allows all pages to be released to the printer.
-------------	---

`PixDispl.EndPrintJob`

AbortPrintJob Cancels the current print job and removes any existing pages from the print spool.

`PixDispl.AbortPrintJob`

Note: If the workstation is connected to a network printer, and Windows is configured to send print jobs directly to the network print spool, this method *cannot* delete the print pages already on the network print server. To allow ImageBASIC to remove partial print jobs when this method is executed, Windows must be configured to spool locally.

For example, the following code will load an image file and print all of the pages in that file as a single print job:

```
' link the Pixel Display control to a File control
PixDispl.ImageDataSource = PixFile1.Link

' load an image file which will display the first
' page in the Pixel Display window
PixFile1.LoadFile "c:\multpage.tif"

' start the print job -- this will hold all future
' print commands to be sent to the printer as a
' single job
PixDispl.StartPrintJob

' sequentially load all of the images in the current
' file and then print each page
For intLoop = 1 to PixFile1.PageCount
    PixFile1.PageIndex = intLoop
    PixDispl.Refresh
    PixDispl.PrintImage
Next intLoop

' after all pages are in the print queue, send the
' entire batch to the printer as a single job
PixDispl.EndPrintJob
```

Printing Multiple Images on One Page

The Pixel Display control can print multiple images on a single page that is sent to the printer. The process of creating this multi-image page is frequently referred to as printing to a virtual page. The concept of a virtual page arises because the display window creates a blank image in memory, and subsequent images printed from the display window will all be sent to this blank image.

The following methods are used to control the creation of the virtual print page:

StartPrintPage	Initializes a new virtual print page. Several properties describing the virtual print page are then valid and may be used to select a region of the virtual page. These properties are listed on the next page. <code>PixDispl.StartPrintPage</code>
PrintImage	The standard printing method that will print the entire current image; the only difference is that within a print page, the image will be made part of the virtual page rather than an entire page of its own. <code>PixDispl.PrintImage</code>
PrintRegion	The standard printing method that will print the Working Region of the current image; the only difference is that within a print page, the image will be made part of the virtual page rather than an entire page of its own. <code>PixDispl.PrintRegion</code>
EndPrintPage	Completes the virtual print page and sends it to the printer. If called within a print job, the page will be held until the job is complete. If no print job has been started, the page will be sent immediately to the printer. <code>PixDispl.EndPrintPage</code>

Specify the Print Region on a Virtual Page

StartPrintPage is primarily used in conjunction with the Print Target Properties to print image data from multiple images to different areas of the same page. When **StartPrintPage** is called, several Print Target properties are populated and describe the virtual print page. These properties are as follows:

PrintTargetBottom	Specifies the bottom of the portion of the page into which the next PrintImage or PrintRegion command will be directed
PrintPageHeight	Reports the number of pixels in the height of the printable area of the page
PrintTargetLeft	Specifies the left edge of the portion of the page into which the next PrintImage or PrintRegion command will be directed
PrintTargetRight	Specifies the right edge of the portion of the page into which the next PrintImage or PrintRegion command will be directed
PrintTargetTop	Specifies the top of the portion of the page into which the next PrintImage or PrintRegion command will be directed
PrintPageWidth	Reports the number of pixels in the width of the printable area of the page

Virtual Print Page -- Example

For example, the following code will print two images to a single page. The first image will be scaled to fit in the top half of the page, and the second image will be fit to the bottom half of the page.

```
' link the display control to a file control
PixDispl.ImageDataSource = PixFile1.Link

' begin the print page
PixDispl.StartPrintPage

' load and display an image
PixFile1.LoadFile "c:\1.tif"

' select the top half of the print page
PixDispl.PrintTargetLeft = 0
PixDispl.PrintTargetTop = 0
PixDispl.PrintTargetRight = PixDispl.PrintPageWidth
PixDispl.PrintTargetBottom = PixDispl.PrintPageHeight / 2
```

```

        ' print the displayed image to the above region
PixDispl.PrintImage

        ' load another image
PixFile1.LoadFile "c:\tiff\2.tif"

        ' select the bottom half of the virtual print page
PixDispl.PrintTargetTop = PixDispl.PrintPageHeight / 2
PixDispl.PrintTargetBottom = PixDispl.PrintPageHeight

        ' send the image to the virtual print page
PixDispl.PrintImage

        ' send the page with both images to the printer
PixDispl.EndPrintPage

```

Chapter 5 : Reference

Reference to Properties, Methods and Events

AboutBox Method

Definition:	Displays the ImageBASIC About box supplying version and contact information.
Parameters:	None
Syntax:	<code>PixDispl.AboutBox</code>
Return Values:	None
Comments:	This message box is application modal and displays version, copyright, licensing, and legal notices. The message box will be unloaded when the OK button is clicked.

AcceptDragFiles Property

Definition:	If True, the Display control's FilesDropped event is enabled and the control will accept files dragged from the File Manager or Explorer and dropped onto the control. If False, the event will not occur.
Data Type:	Boolean
Syntax:	<code>PixDispl.AcceptDragFiles = <i>boolOption</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	FilesDropped Event
Possible Values:	True False
Comments:	When files are dropped on the Display control after dragging from the File Manager or Explorer, the FilesDropped event occurs, reporting the names of the files. If this property is False, the event will not occur.

Active Property

Definition:	<p>If set to True at designtime, 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
Syntax:	<code>PixDispl.Active = <i>boolOption</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write (see limits below)
See Also:	"Licensing Configuration and Verification on page 3
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>

BackColor Property

Description:	Determines the background color of an object.
Syntax:	<code>PixDispl.BackColor = lngColor</code>
Remarks:	Visual Basic uses the Microsoft Windows environment RGB scheme for colors. Each property has the following ranges of settings:

Range of settings	Description
Normal RGB colors	Colors specified by using the Color palette, or by using the RGB or QBColor functions in code.
System default colors	Colors specified with system color constants from CONSTANT.TXT, a Visual Basic file that specifies system defaults. The Windows environment substitutes the user's choices as specified in the user's Control Panel settings.

BackColor set to the WINDOW_BACKGROUND system default color as specified in CONSTANT.TXT.

In labels and shapes, the BackColor property is ignored if the BackStyle property is 0 (Transparent).

If you set the BackColor property on forms or picture boxes, all graphics and print output, including the persistent bitmap, are erased. Setting the ForeColor property does not affect graphics or print output already drawn. On all other controls, the screen color changes immediately.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF). If the high byte is not 0, Visual Basic uses the system colors, as defined in the user's Control Panel and enumerated in CONSTANT.TXT.

To display text in the Windows environment, both the text and background colors must be solid. If the text or background colors you've selected are not displayed, one of the selected colors may be dithered that is, comprised of up to three different-colored pixels. If you choose a dithered color for either the text or background, the nearest solid color will be substituted.

This information taken from the Microsoft Visual Basic 4.0 Help. Copyright 1995 by Microsoft Corporation. All rights reserved.

BlueBrightness Property

Definition:	Specifies the brightness of the blue component of the display.
Data Type:	Integer
Syntax:	<code>PixDispl.BlueBrightness = <i>intBright</i></code>
Possible Values:	Integers, 0 to 256 Default is 127
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	BlueContrast Property, Brightness Property
Comments:	Higher values increase the brightness of the blue component of all colors in the image. Lower values darken the blue component of all colors in the image. To change the overall brightness of the image, use the Brightness property.

BlueContrast Property

Definition:	Specifies the contrast of the blue component of the display.
Data Type:	Integer
Syntax:	<code>PixDispl.BlueContrast = <i>intContrast</i></code>
Possible Values:	Integers, 0 to 256 Default is 127
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	BlueBrightness Property, Brightness Property
Comments:	Higher values increase the contrast of all the blue component in all colors. Lower values reduce the blue contrast. To change the overall contrast of the image, use the Contrast property.

BorderStyle Property

Description:	Determines the border style for an object. For forms and text boxes, read-only at run time.						
Syntax:	<code>PixDispl.BorderStyle = enumOption</code>						
Settings:	The BorderStyle property settings for this control are: <table><tr><th>Setting</th><th>Description</th></tr><tr><td>0</td><td>None</td></tr><tr><td>1</td><td>(Default) Fixed Single</td></tr></table>	Setting	Description	0	None	1	(Default) Fixed Single
Setting	Description						
0	None						
1	(Default) Fixed Single						
Remarks:	<p>For a form, the BorderStyle property determines key characteristics that visually identify a form as either a general-purpose window or a dialog box. Setting 3 (Fixed Double) is useful for standard dialog boxes.</p> <p>MDI child forms set to 2 (Sizable) will appear within the MDI form in a default size defined by the Windows environment at run time. For any other setting, the form will appear in the size specified at design time.</p> <p>If a form with a menu is set to 3 (Fixed Double), it will appear with a setting 1 (Fixed Single) border instead.</p> <p>At run time, a form is either modal or modeless. Use the Show method to specify whether a form is modal or modeless.</p>						
Data Type:	Integer (Enumerated) <i>This information taken from the Microsoft Visual Basic 4.0 Help. Copyright 1995 by Microsoft Corporation. All rights reserved.</i>						

Brightness Property

Definition:	Specifies the overall brightness of the displayed image.
Data Type:	Integer
Syntax:	<code>PixDispl.Brightness = intBright</code>
Possible Values:	Integers, 0 to 256 Default is 127
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	Contrast Property, BlueBrightness Property, GreenBrightness Property, RedBrightness Property
Comments:	Higher values in this property make the image brighter. Lower values make it darker.

Click Event

- Definition:** Occurs when the user presses and then releases a mouse button over the Pixel Display window.
- Parameters:** None
- See Also:** DbClick Event
- Comments:** This event can be triggered by any of the mouse button options if the mouse is moved less ten pixels. When the mouse is used to define a region, the **MouseUp** and **MouseDown** events are triggered.

Contrast Property

- Definition:** Specifies the overall contrast of the displayed image. Contrast cannot apply to bitonal images, only gray or color.
- Data Type:** Integer
- Syntax:** `PixDispl.Contrast = intBright`
- Possible Values:** Integers, 0 to 256
Default is 127
- Design Access:** Read/Write
- Runtime Access:** Read/Write
- See Also:** Brightness Property, BlueContrast Property, GreenContrast Property, RedContrast Property
- Comments:** Higher values in this property increase the contrast of the displayed image. Lower values in this property decrease the apparent contrast.

DbClick Event

- Definition:** Occurs when either mouse button double clicks on the Pixel Display window.
- Parameters:** None
- See Also:** Click Event, 'Mouse Button Option Properties' on page 19
- Comments:** The time in which the double click must occur is defined in the Windows setup.

DisplayBottom Property

Definition:	Reports the image coordinates of the bottom edge of the Display window relative to the top edge of the image data.
Data Type:	Long Integer
Syntax:	<i>lngPosition</i> = PixDispl.DisplayBottom
Design Access:	Not Available
Runtime Access:	Read-only
Possible Values:	Any integer value, including negative values
See Also:	DisplayLeft Property, DisplayRight Property, DisplayTop Property, ZoomBottom Property, The Zoom Region on page 27
Comments:	The displayed image will not necessarily completely fill the Display window. In this case, the value of this property can be larger than the image height.

DisplayLeft Property

Definition:	Reports the image coordinates of the left edge of the Display window relative to the left edge of the image data.
Data Type:	Long Integer
Syntax:	<i>lngPosition</i> = PixDispl.DisplayLeft
Design Access:	Not Available
Runtime Access:	Read-only
Possible Values:	Any integer value, including negative values
See Also:	DisplayBottom Property, DisplayRight Property, DisplayTop Property, ZoomLeft Property, The Zoom Region on page 27
Comments:	<p>The displayed image will not necessarily completely fill the Display window. In this case, the value of this property can be less than zero.</p> <p>A negative value indicates that there is space between the left edge of the image data and the left edge of the Display window. A positive value indicates that some image data extends to the left of the Display window. By default in this case, scroll bars will be displayed.</p>

DisplayRight Property

Definition:	Reports the image coordinates of the right edge of the Display window with 0 (zero) at the left edge of the image data.
Data Type:	Long Integer
Syntax:	<i>lngPosition</i> = PixDispl.DisplayRight
Design Access:	Not Available
Runtime Access:	Read-only
Possible Values:	Any integer value, including negative values
See Also:	DisplayBottom Property, DisplayLeft Property, DisplayTop Property, ZoomRight Property, The Zoom Region on page 27
Comments:	The displayed image will not necessarily completely fill the Display window. In this case, the value of this property can be larger than the image width.

DisplayTop Property

Definition:	Reports the image coordinates of the top edge of the Display window relative to the top edge of the image.
Data Type:	Long Integer
Syntax:	<i>lngPosition</i> = PixDispl.DisplayTop
Design Access:	Not Available
Runtime Access:	Read-only
Possible Values:	Any integer value, including negative values
See Also:	DisplayBottom Property, DisplayRight Property, DisplayLeft Property, ZoomTop Property, The Zoom Region on page 27
Comments:	<p>The displayed image will not necessarily completely fill the Display window. In this case, the value of this property can be less than zero.</p> <p>A negative value indicates that there is some space between the top of the image data and the top of the Display window. A positive value indicates that some of the image data extends above the Display window, when scroll bars will be displayed by default.</p>

DoClick Method

Definition:	Causes the Click event to occur.
Parameters:	None
Syntax:	<code>PixDispl.DoClick</code>
Return Values:	None
See Also:	Click Event, LeftButtonOption Property, RightButtonOption Property
Comments:	Calling this method is equivalent to triggering the Click event. This method may be used at any time, even when the mouse buttons are set to an option that would prevent them from triggering the Click event.

EndPrintJob Method

Definition:	Ends the creation of a print job, allowing the job to be released to the printer.
Parameters:	None
Syntax:	<code>PixDispl.EndPrintJob</code>
Returns:	None
See Also:	StartPrintJob Method, AbortPrintJob Method
Comments:	<p>This method must be called once for each StartPrintJob method that is called. Refer to the StartPrintJob method for details on the creation of multi-page print jobs.</p> <p>The only other method available to end a print job is the AbortPrintJob method which will end the job and remove all pages from the print spool, if possible.</p>

EndPrintPage Method

Definition:	Ends the creation of a virtual print page, signaling its completion.
Parameters:	None
Syntax:	<code>PixDispl.EndPrintPage</code>
Returns:	None
See Also:	StartPrintPage Method
Comments:	A virtual print page is created by calling the StartPrintPage method. This virtual page is held in memory and can have multiple images printed to it. Refer to the StartPrintPage method for more details on this process.

Error Event

Definition:	Occurs once for each error internal to this control.
Parameters:	See below
Comments:	This event occurs only for errors internal to the specific control. If no code is placed in this event, the standard dialog is displayed and a runtime exception occurs.

Number

Reports the error number; see 'Appendix A : Error Reporting' on page 103

Description

Reports a descriptive string of the error

SCode

VB runtime error number; equal to *Number* plus `vbObjectError`

Source

Reports a string of the source of the error

HelpFile

Reports the name of a help file that should explain this error

HelpContext

Reports the help context ID in the *HelpFile* that matches this error

CancelDisplay

Specifies whether or not a built-in message box and/or a runtime exception will report the error. Can be set to any of the following constants to specify the error reporting method:

ibErrNoAction

No display or runtime error

ibErrDisplay

Only display built-in message

ibErrDisplayAndRaise (Default)

Display message and raise runtime exception

ibErrRaise

Only raise runtime exception

EventMask Property

- Definition:** Enables and disables select events.
- Data Type:** Enumerated Integer
- Syntax:** `PixDispl.EventMask(eventID) = ibEventMaskEnable`
- Design Access:** Not Available
- Runtime Access:** Read/Write
- Possible Values:** `ibEventMaskEnable`
`ibEventMaskDisable`
- Comments:** If set to `ibEventMaskEnable` , the specified event will occur.
If set to `ibEventMaskDisable` , the event will not occur.
Application performance can sometimes be notably improved by disabling unused events. The **EventMask** property is used to enable and disable certain events that are not be used in the application.

For example, the **MouseMove** event may need to be disabled. The following code will do this:

```
PixDispl.EventMask(dspEventMouseMove) =  
    ibEventMaskDisable
```

To enable, for example, the **Paint** event, execute this code:

```
PixDispl.EventMask(dspEventPaint) =  
    ibEventMaskEnable
```

The events that can be enabled and disabled are as follows:

Event	eventID Constant
Click	<code>dspEventClick</code>
DblClick	<code>dspEventDblClick</code>
Error	<code>dspEventError</code>
FilesDropped	<code>dspEventFilesDropped</code>
ImageDataChanged	<code>dspEventImageDataChanged</code>
MouseDown	<code>dspEventMouseDown</code>
MouseMove	<code>dspEventMouseMove</code>
MouseUp	<code>dspEventMouseUp</code>
Paint	<code>dspEventPaint</code>

The numerical values assigned to these **EventMask** constants are subject to change in future revisions of ImageBASIC. Therefore, it is advisable to always use the constants when referencing these events in the **EventMask** property.

Note: The **EventMask** property supersedes the **FireGUIEvents** and **AcceptDragFiles** properties. These properties are maintained for backward compatibility.

FilesDropped Event

Definition: Occurs when files are dragged from the Explorer or File Manager and dropped on the Display control.

Parameters: FileCount Long integer reporting the total number of files
FileName Variant array of the file names

See Also: AcceptDragFiles Property

Comments: By default, this method will not occur. The **AcceptDragFiles** property must be set to True to allow this event to occur.

In order for the Display control to show the dropped files, they must be opened by a File control that is specified in the Display control's **ImageDataSource** property.

The following sample code displays the names of all of the dropped files in a list box. Code may be placed elsewhere to instruct the File control to open one of these files for display.

```
Private Sub PixDispl_FilesDropped(ByVal  
    FileCount As Long, ByVal FileName As  
    Variant)  
    Dim intLoop as Integer  
    For intLoop = 1 to FileCount  
        List1.AddItem FileName(intLoop)  
    Next intLoop  
End Sub
```

FillWorkingRegion Method

Definition:	Fills the current Working Region with either white or black pixels.
Parameters:	<i>FillColor</i> Enumerated specification of the fill color
Syntax:	<code>PixDispl.FillWorkingRegion <i>FillColor</i></code>
Returns:	None
See Also:	"The Working Region" on page 32
Comments:	The currently defined Working Region will be filled with the specified color. The FillColor parameter to this method accepts the following values: <div style="margin-left: 40px;">0 Black 1 White</div>

FireGUIEvents Property

Definition:	Enables or disables the Paint and MouseMove events.
Data Type:	Boolean
Syntax:	<code>PixDispl.FireGUIEvents = <i>boolOption</i></code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	Paint Event, MouseMove Event
Possible Values:	True False (Default)
Comments:	The Paint and MouseMove events require substantial processor time when executed because of the frequency with which they occur. If these events are not required, some processor time may be freed by disabling the events through the FireGUIEvents property. This property defaults to False, thereby disabling the events.

FitToHeight Method

Definition:	Scales the image to fit the height of the image to the height of the display window.
Parameters:	None
Syntax:	<code>PixDispl.FitToHeight</code>
Returns:	None
See Also:	FitToWindow Method, FitToWidth Method
Comments:	Calling this method resets the scaling factor to display the entire height of the image in the display window. If the image is wider than the window, the excess image data will be allowed to pass the right edge of the display.

FitToWidth Method

Definition:	Scales the image to fit the width of the image to the width of the display window.
Parameters:	None
Syntax:	<code>PixDispl.FitToWidth</code>
Returns:	None
See Also:	FitToWindow Method, FitToHeight Method
Comments:	Calling this method resets the scaling factor to display the entire width of the image in the display window. If the image does not fit the window, the image will extend past the bottom of the window.

FitToWindow Method

Definition:	Scales the image to display the entire image in the display window.
Parameters:	None
Syntax:	<code>PixDispl.FitToWindow</code>
Returns:	None
See Also:	FitToHeight Method, FitToWidth Method
Comments:	The image will be scaled to fit the window completely, identically to the default display of the image when first loaded.

GetRegBlackPercent Method

Definition:	Determines the ratio of black pixels in the currently defined Working Region.
Parameters:	None
Syntax:	<i>lngDensity</i> = PixDispl.GetRegBlackPercent
Returns:	Long integer percentage of black pixels in region
Comments:	<p>This method analyzes the Working Region and determines what percentage of the region is comprised of black pixels. If a region is tested on a color image, the image data will be converted to bitonal before the density is calculated.</p> <p>Different types of image data generally return values in limited ranges:</p> <ul style="list-style-type: none">• Normal text usually returns a value of 20% to 30%• Bar codes return values of 40% to 60%• Pictures and complex graphics may vary from 15% to 70% depending on their complexity• Because of the rounding errors in the function, always expect to get 1 to 3% error; i.e., a white region may return 2 or 3, and a black region may return 97 or 98.

GetScaledPicture Method

Definition:	Creates a bitmap of the specified dimensions based on the currently displayed image and returns the Windows handle to the bitmap.
Parameters:	<p><i>width</i> Integer pixel width of the bitmap to create</p> <p><i>height</i> Integer pixel height of the bitmap to create</p>
Syntax:	<i>Picture</i> = PixDispl.GetScaledPicture(<i>width</i> , <i>height</i>)
Data Type:	Integer
Returns:	hDIB Windows handle to the DIB that is created
See Also:	SetPicture Method, DisplayMode Property
Comments:	<p>Typically, this method will be used to create a thumbnail image for display in another Visual Basic control such as a Picture Box.</p> <p>For example, the following code snippet will scale the currently displayed image to relatively small dimensions and display all</p>

pages of the current document in a control array of Picture Box controls:

```
Dim intLoop as Integer
PixDispl.ImageDataSource = PixFile1.Link
PixFile1.LoadFile "c:\images\1001.tif"
For intLoop = 1 to PixFile1.PageCount
    PixFile1.PageIndex = intLoop
    Load Picture(intLoop)
    Picture(intLoop).Left = intLoop * 125
    Picture(intLoop).Width = 100
    Picture(intLoop).Picture =
        PixDispl.GetScaledPicture(75, 90)
    Picture(intLoop).Visible = True
Next intLoop
```

GreenBrightness Property

Definition:	Specifies the brightness of the green component of the display.
Data Type:	Integer
Syntax:	<code>PixDispl.GreenBrightness = <i>intBright</i></code>
Possible Values:	Integers, 0 to 256 Default is 127
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	GreenContrast Property, Brightness Property
Comments:	Higher values increase the brightness of the green component of all colors in the image. Lower values darken the green component of all colors in the image. To change the overall brightness of the image, use the Brightness property.

GreenContrast Property

Definition:	Specifies the contrast of the green component of the display.
Data Type:	Integer
Syntax:	<code>PixDispl.GreenContrast = <i>intContrast</i></code>
Possible Values:	Integers, 0 to 256 Default is 127
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	GreenBrightness Property, Brightness Property
Comments:	Higher values increase the contrast of all the green component in all colors. Lower values reduce the green contrast. To change the overall contrast of the image, use the Contrast property.

hWnd Property

Definition:	Reports the handle to the Pixel Display window.
Data Type:	Long
Syntax:	<code><i>lngHandle</i> = PixDispl.hWnd</code>
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	Name Property
Comments:	The value that is returned should not be stored for future use as the handle can change; instead the property should be queried each time the control's handle is required.

ImageBitsPerSample Property

Definition: Reports the bits per sample in the currently displayed image's color definition.

Data Type: Integer

Syntax: `intBPS = PixDispl.ImageBitsPerSample`

Design Access: Not Available

Runtime Access: Read-only

See Also: ImageSamplesPerPixel Property

Comments: The level and type of color in an image is defined by the Bits Per Sample (BPS), Samples Per Pixel (SPP), and Photometric Interpretation (PI). The following table shows the most common configurations:

Color Definition	Description
SPP = 1 BPS = 1 PI = 0	Typical Bitonal White encoded as 0
SPP = 1 BPS = 4 PI = 0	16-level gray White encoded as 0
SPP = 1 BPS = 8 PI = 0	256-level gray White encoded as 0
SPP = 3 BPS = 8 PI = 2	24-bit (16.7 million colors) RGB
SPP = 1 BPS = 4 PI = 3	16-color Paletted
SPP = 1 BPS = 8 PI = 3	256-color Paletted

For a more detailed discussion of color definition, refer to "Appendix B : Color Definition" on page 115.

ImageDataChanged Event

- Definition:** Occurs when the image data that is being received from the control specified in the **ImageDataSource** property changes.
- Parameters:** None
- See Also:** ImageDataSource Property
- Comments:** This event is a vital component in ImageBASIC's linking of controls. When the image data that is being supplied to this control changes, this event occurs.
- Batch processing operations that perform the same operations on a series of images will almost always make use of this event to begin the processing on each new image as it is received.

ImageDataSource Property

- Definition:** Specifies another ImageBASIC control from which the Display control will receive image data.
- Data Type:** String
- Syntax:** `PixDispl.ImageDataSource = IBControl.Link`
- Design Access:** Read / Write
- Runtime Access:** Read / Write
- See Also:** ImageDataChanged Event, Linking of ImageBASIC Controls on page 1
- Comments:** This property must specify an ImageBASIC control that can serve as the source of image data. The most commonly specified image sources are File and Scan controls, but other Display controls and TextBridge (during verification) are also valid image sources.
- When the Display controls in the Pan Window mode, the parent Display control must be specified in this property.

ImageHeight Property

Definition:	Reports the number of pixels in the height of the currently displayed image.
Data Type:	Integer
Syntax:	<code>intHeight = PixDispl.ImageHeight</code>
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	ImageWidth Property, ZomHeight Property
Comments:	The height of the original image as it was supplied to the display control is reported here.

ImageSamplesPerPixel Property

Definition:	Reports the samples per pixel in the currently displayed image's color definition.
Data Type:	Long Integer
Syntax:	<code>intSPP = PixDispl.ImageSamplesPerPixel</code>
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	ImageBitsPerSample Property
Comments:	The level and type of color in an image is defined by the Bits Per Sample (BPS), Samples Per Pixel (SPP), and Photometric Interpretation (PI). For a more detailed discussion of color definition, refer to 'Comparative Charts of Colors Supported by File Format' on page 118.

ImageWidth Property

Definition:	Reports the pixel width of the currently displayed image.
Data Type:	Long Integer
Syntax:	<code>intWidth = PixDispl.ImageWidth</code>
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	ImageHeight Property, ImageYRes Property
Comments:	The width reported in this property is the width of the original image as it was supplied to this control. The image will be scaled for display, but that scaling will not be reflected in this property.

ImageXRes Property

Definition:	Reports the horizontal resolution of the currently displayed image in DPI.
Data Type:	Integer
Syntax:	<code>intRes = PixDispl.ImageXRes</code>
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	ImageYRes Property, ImageWidth Property
Comments:	For some file types such as BMP and GIF, resolution is not stored in the image and cannot be calculated. In this case, the ImageXRes property will report 300, the value to which ImageBASIC defaults.

ImageYRes Property

Definition:	Reports the vertical resolution of the currently displayed image in DPI.
Data Type:	Integer
Syntax:	<code>intRes = PixDispl.ImageYRes</code>
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	ImageXRes Property, ImageHeight Property
Comments:	For some file types such as BMP and GIF, resolution is not stored in the image and cannot be calculated. In this case, the ImageYRes property will report 300, the value to which ImageBASIC defaults.

Invert Property

Definition:	If True, reverses all colors in the displayed image from the colors defined in the original image.
Data Type:	Boolean
Syntax:	<code>PixDispl.Invert = <i>boolOption</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	Rotation Property, ScaleToGray Property
Comments:	Any image data passed from the Pixel Display control will reflect the setting in this property. For example, suppose that an image is displayed with white text on a black background (inverted). Any other ImageBASIC component that is linked to this display control will receive the inverted data.

LeftButtonOption Property

Definition:	Specifies the operation that will be performed when the left mouse button is used to drag out an area of the image.
Data Type:	Enumerated
Syntax:	<code>PixDispl.LeftButtonOption = <i>enumOption</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	LeftMouseBoxStyle Property, RightButtonOption Property
Possible Values:	<ul style="list-style-type: none">0 Normal1 Rubber Band2 Proportional Rubber Band3 Zoom4 Proportional Zoom (Default)5 Drag Region6 Drag Tool
Comments:	<p>When an image region is selected using the mouse to drag out a rectangle on the display window, this property specifies what type of region is specified and how the region is drawn.</p> <p><i>0--Normal</i> does not automatically perform any action, but all of the mouse related events are enabled and the application developer is free to perform any operation.</p>

1--Rubber Band selects a new Working Region when the mouse button is released. The region can be drawn with any aspect ratio.

2--Proportional Rubber Band selects a new Working Region, and the rectangle that is drawn will always have the aspect ratio that matches the aspect ratio of the display window.

3--Zoom sets the Zoom Region when the mouse button is released, enlarging the selected region to fill the display window. The region may be drawn with any aspect ratio, and the selected region will be fit to the display window as fully as possible.

4--Proportional Zoom sets the Zoom Region properties when the mouse button is release, enlarging the selected region to fill the display window. The region is drawn with the same aspect ratio and the display window, so the selected region always exactly fits the display window when it is enlarged.

5--Drag Region allows the user to move the currently defined Working Region by dragging it while depressing the mouse button.

6--Drag Tool allows the user to move the zoomed image by dragging it while holding the depressed mouse button.

LeftMouseButtonStyle Property

Definition: When a rectangle is being drawn by dragging with the left mouse button, this property specifies whether it will be filled or hollow.

Data Type: Enumerated

Syntax: `PixDispl.LeftMouseButtonStyle = enumOption`

Design Access: Read/Write

Runtime Access: Read/Write

See Also: LeftButtonOption Property, RightMouseButtonStyle Property, WorkingRegionStyle Property

Possible Values:

0	None
1	Hollow (Default)
2	Solid

Comments: In order for this property to have any effect, the **LeftButtonOption** must be set to a value other than *0--Normal*. *0--None* makes no visible change in the Display control to indicate the Working Region. *1--Hollow* draws an outline around the region that is being selected. *2--Solid* draws the rectangle as an inverted region on the image.

Link Property

Definition:	Reports the Link ID calculated for this control at its creation.
Data Type:	String
Syntax:	<i>IBControl.Source = PixDispl.Link</i>
Design Access:	Not Available
Runtime Access:	Read-only
Comments:	Each ImageBASIC control is assigned a unique Link ID at its creation. This Link ID can be specified in the ImageDataSource , DisplaySource , RegionSource , and AnnoteSource properties of various ImageBASIC controls. These source properties specify the ImageBASIC control that is supplying information or services to a control.

MergeNotes Method

Definition:	Merges all currently displayed annotation objects into the underlying image data.
Parameters:	None
Syntax:	<i>PixDispl.MergeNotes</i>
Data Type:	None
Return Value:	None
Comments:	<p>One or more annotation objects must be displayed to allow this event to operate. Merging annotation objects converts the annotations to actual image data which cannot then be modified by the Annotation control.</p> <p>After this method completes, the displayed image page must be saved to persist the changes to the image file.</p>

MouseDown Event

Definition:	Occurs each time either of the mouse buttons is depressed on the Pixel Display window.	
Parameters:	Button	Reports which mouse button was depressed
	Shift	Reports the state of the Shift and Control keys
	ScreenX	Reports the horizontal position of the mouse pointer
	ScreenY	Reports the vertical position of the mouse pointer
	ImageX	Reports the horizontal pixel position in the image
	ImageY	Reports the vertical pixel position in the image
See Also:	MouseUp Event, LeftButtonOption Property	
Comments:	This event will occur for all mouse button options.	

MouseIcon Property

Definition:	Specifies the icon to display for the mouse cursor. Valid only if MousePointer property is <i>99--Custom</i> .
Data Type:	Long Integer
Syntax:	<code>PixDispl.MouseIcon = LoadPicture(<i>filename</i>)</code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	MousePointer Property
Possible Values:	Any object expression that evaluates to a Picture
Comments:	This property has the same behavior as the MouseIcon property of a Visual Basic form.

MouseMove Event

- Definition:** Occurs when the mouse is moved on top of the Pixel Display window.
- Parameters:**
- | | |
|---------|--|
| Button | Reports which mousebutton was depressed |
| Shift | Reports the state of the Shift and Control keys |
| ScreenX | Reports the horizontal position of the mouse pointer |
| ScreenY | Reports the vertical position of the mouse pointer |
| ImageX | Reports the horizontal pixel position in the image |
| ImageY | Reports the vertical pixel position in the image |
- See Also:**MouseDown Event, MouseUp Event, Click Event
- Comments:** Any code placed in this event should be limited as much as possible because of the frequency with which this event occurs. This method must be enabled through the **FireGUIEvents** property.

MousePointer Property

- Definition:** Specifies the mouse pointer style.
- Data Type:** Enumerated
- Syntax:** `PixDispl.MousePointer = enumOption`
- Design Access:** Read/Write
- Runtime Access:** Read/Write
- See Also:** MouseIcon Property
- Possible Values:** See below
- Comments:** The valid values for this property are listed in the table below.

Constant	Value	Description
ibDefault	0	Default
ibArrow	1	Arrow
ibCross	2	Cross
ibIBeam	3	I-beam
ibIcon	4	Icon
ibSizeCur	5	Size
ibSizeNESW	6	Size NE to SW
ibSizeNS	7	Size N to S

Constant	Value	Description
ibSizeNWSE	8	Size NW to SE
ibSizeWE	9	Size W to E
ibUpArrow	10	Up Arrow
ibHourglass	11	Hourglass
ibNoDrop	12	No drop
ibArrowHourglass	13	Arrow and hourglass.
ibArrowQuestion	14	Arrow and question mark.
ibSizeAll	15	Size all.
ibCustom	99	Custom icon specified by the MouseIcon property.

MouseUp Event

Definition:	Occurs when either mouse button is released if the mouse button option that is selected for that button enables this event.	
Parameters:	Button	Reports which mouse button was depressed
	Shift	Reports the state of the Shift and Control keys
	ScreenX	Reports the horizontal position of the mouse pointer
	ScreenY	Reports the vertical position of the mouse pointer
	ImageX	Reports the horizontal pixel position in the image
	ImageY	Reports the vertical pixel position in the image
See Also:	RightButtonOption Property, LeftButtonOption Property	
Comments:	This event will occur for all mouse button options.	

NewImage Method

Definition:	Creates a blank image page in the Display control.	
Parameters:	Width	Pixel width of the image to create
	Height	Pixel height of the image to create
	XRes	Horizontal resolution in DPI of the new image
	YRes	Vertical resolution in DPI of the new image
Returns:	None	
Syntax:	PixDispl.NewImage <i>width, height, xres, yres</i>	
Comments:	The image page that is created is bitonal.	

Paint Event

- Definition:** Occurs each time the Pixel Display window is drawn.
- Parameters:** None
- See Also:** Refresh Method, FireGUIEvents Property
- Comments:** The display window is repainted each time that any visible changes are made to the image and each time the Refresh method is called. Any code placed in this event should be limited as much as possible because of the frequency with which the event occurs.
- By default, this event will not occur. It must be enabled by setting the **FireGUIEvents** property to True.

Picture Property

- Definition:** Reports the handle to a scaled DIB that the control creates from the currently displayed image.
- Note:** This property has been superseded by the **GetScaledPicture** and **SetPicture** methods.
- Data Type:** Picture Object
- Syntax:** `Picture = PixDispl.Picture`
- Design Access:** Not Available
- Runtime Access:** Read-only
- See Also:** DisplayMode Property, ThumbnailByteSize Property
- Comments:** The Pixel Display control will create a DIB of any specified size based on the currently displayed image. This may be used as an alternative method for creating thumbnail images and displaying them in a Visual Basic or third party control that can accept image data as a DIB handle.
- A typical use for the **Picture** property would be to create a control array of Picture Box controls and display each page in a multiple page file in them.

PrinterDC Property

Definition:	Reports the device context of the current printer.
Data Type:	Long Integer
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	PrinterDevice Property, PrinterDriver Property, PrinterPort Property, StartPrintJob Method
Comments:	<p>This property will hold a valid value only during printing or during the construction of a print job; i.e., between calls to the StartPrintJob and EndPrintJob methods.</p> <p>The currently selected printer is reported in the PrinterDevice, PrinterDriver, and PrinterPort properties.</p>

PrinterDevice Property

Definition:	Specifies the name of the currently selected printer device.
Data Type:	String
Syntax:	<code>PixDispl.PrinterDevice = strDevice</code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	PrinterDriver Property, PrinterPort Property, SelectPrinter Method
Comments:	<p>When any of the printing related methods is called, the Pixel Display control will attempt to use the printer specified in these three properties:</p> <ul style="list-style-type: none">PrinterDevicePrinterDriverPrinterPort <p>If these properties do not point to a valid printer device -- for example, no when no printer has been selected -- the Windows default printer will be used. When the SelectPrinter method dialog is used to select a printer, these properties are updated to reflect the selection.</p>

PrinterDriver Property

Definition: Specifies the driver name for the currently selected printer device.

Data Type: String

Syntax: `PixDispl.PrinterDriver = strDriver`

Design Access: Not Available

Runtime Access: Read/Write

See Also: PrinterDevice Property, PrinterPort Property, SelectPrinter Method

Comments: When any of the printing related methods is called, the Pixel Display control will attempt to use the printer specified in these three properties:

PrinterDevice

PrinterDriver

PrinterPort

If these properties do not point to a valid printer device -- for example, no when no printer has been selected -- the Windows default printer will be used. When the **SelectPrinter** method dialog is used to select a printer, these properties are updated to reflect the selection.

PrinterPort Property

Definition:	Specifies the output port for the currently selected printer device.
Data Type:	String
Syntax:	<code>PixDispl.PrinterPort = <i>strPort</i></code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	See below
Comments:	<p>When any of the printing related methods is called, the Pixel Display control will attempt to use the printer specified in these three properties:</p> <ul style="list-style-type: none">PrinterDevicePrinterDriverPrinterPort <p>If these properties do not point to a valid printer device -- for example, no when no printer has been selected -- the Windows default printer will be used. When the SelectPrinter method dialog is used to select a printer, these three properties are updated to reflect the selection.</p>

PrintImage Method

Definition:	Prints the entire image, regardless of any Working Region or Zoom Region definition.
Parameters:	None
Syntax:	<code>PixDispl.PrintImage</code>
Returns:	None
See Also:	PrintRegion Method, SelectPrinter Method
Comments:	The image will be printed as it is displayed; that is, the values of the Invert and Rotation properties will be applied to the printed data.

PrintPageHeight Property

Definition:	Reports the height of a virtual print page created by the StartPrintPage method.
Data Type:	Long Integer
Syntax:	<i>lngHeight</i> = PixDispl.PrintPageHeight
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	StartPrintPage Method, StartPrintJob Method, PrintPageWidth Property
Comments:	Use this property to calculate PrintTargetTop and PrintTargetBottom values when adding image data to a virtual print page.

PrintPageWidth Property

Definition:	Reports the width of a virtual print page created by the StartPrintPage method.
Data Type:	Long Integer
Syntax:	<i>lngWidth</i> = PixDispl.PrintPageWidth
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	StartPrintPage Method, StartPrintJob Method, PrintPageHeight Property
Comments:	Use this property to calculate PrintTargetLeft and PrintTargetRight values when adding image data to a virtual print page.

PrintRegion Method

Definition:	Prints the current Working Region.
Parameters:	None
Syntax:	<code>PixDispl.PrintRegion</code>
Returns:	None
See Also:	PrintImage Method, 'The Working Region' on page 32, SelectPrinter Method
Comments:	<p>The Working Region is defined by the RegTop, RegBottom, RegLeft, and RegRight properties. The portion of the image that is defined by these coordinates will be printed when this method is invoked. The region will be scaled to fit the output page as well as possible.</p> <p>The image will be printed as it is displayed; that is, the values of the Invert and Rotation properties will be applied to the printed data.</p>

PrintTargetBottom Property

Definition:	Specifies the bottom edge of the destination region when adding image data to a virtual print page.
Data Type:	Long Integer
Syntax:	<code>PixDispl.PrintTargetBottom = lngPosition</code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	StartPrintPage Method, StartPrintJob Method, PrintPageHeight Property
Comments:	Total virtual page height can be read from the PrintPageHeight property. Print Target properties are available only after calling the StartPrintPage method.

PrintTargetLeft Property

Definition:	Specifies the left edge of the destination region when adding image data to a virtual print page.
Data Type:	Long Integer
Syntax:	<code>PixDispl.PrintTargetLeft = lngPosition</code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	StartPrintPage Method, StartPrintJob Method, PrintPageWidth Property
Comments:	Use PrintPageWidth to calculate valid settings for PrintTargetLeft . Available only after calling the StartPrintJob method.

PrintTargetRight Property

Definition:	Specifies the right edge of the destination region when adding image data to a virtual print page.
Data Type:	Long Integer
Syntax:	<code>PixDispl.PrintTargetRight = lngPosition</code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	StartPrintPage Method, StartPrintJob Method, PrintPageHeight Property
Comments:	Use PrintPageWidth to calculate valid settings for PrintTargetRight . Available only after calling StartPrintJob method.

PrintTargetTop Property

Definition:	Specifies the top edge of the destination region when adding image data to a virtual print page.
Data Type:	Long Integer
Syntax:	<code>PixDispl.PrintTargetTop = lngPosition</code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	StartPrintPage Method, StartPrintJob Method, PrintPageHeight Property
Comments:	Use PrintPageHeight to calculate valid settings for PrintTargetTop . Available only after calling the StartPrintJob method.

RedBrightness Property

Definition:	Specifies the brightness of the red component of the display.
Data Type:	Integer
Syntax:	<code>PixDispl.RedBrightness = intBright</code>
Possible Values:	Integers, 0 to 256 Default is 127
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	RedContrast Property, Brightness Property
Comments:	Higher values increase the brightness of the red component of all colors in the image. Lower values darken the red component of all colors in the image. To change the overall brightness of the image, use the Brightness property.

RedContrast Property

Definition:	Specifies the contrast of the red component of the display.
Data Type:	Integer
Syntax:	<code>PixDispl.RedContrast = <i>intContrast</i></code>
Possible Values:	Integers, 0 to 256 Default is 127
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	RedBrightness Property, Brightness Property
Comments:	Higher values increase the contrast of all the red component in all colors. Lower values reduce the red contrast. To change the overall contrast of the image, use the Contrast property.

Refresh Method

Definition:	Displays all changes made to the image.
Parameters:	None
Syntax:	<code>PixDispl.Refresh</code>
Returns:	None
See Also:	Paint Event
Comments:	<p>It is possible that changes will be made to the image data without the display reflecting those changes. Executing this method will force the display to show the image as it is currently held in memory. The Paint event will occur when this method is called.</p> <p>The most common situation requiring the use of the Refresh method are (1) when using the Annotation control which does not automatically refresh, and (2) when performing many operations, and the paint messages to Windows are delayed in the message queue.</p>

RegBottom Property

Definition:	Specifies the position of the bottom edge of the current Working Region.
Data Type:	Long Integer
Syntax:	<code>PixDispl.RegBottom = lngPosition</code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	RegLeft Property, RegRight Property, RegTop Property, RightButtonOption Property
Comments:	The following properties define the Working Region

RegBottom

RegLeft

RegRight

RegTop

These properties are collectively called the Working Region properties. They define the region of the image that is currently of interest. This region is the only portion of the image that is passed from this control when another ImageBASIC control names the Pixel Display control as its **ImageDataSource**. When an image is first loaded, these properties default to defining the entire image.

These properties may be set explicitly to define a Working Region. They are also updated when a mouse button that is set to either of the Rubber Band options is used to draw a region on the image. See the **LeftButtonOption** and **RightButtonOption** properties for information on these options.

Note: These properties are all in units of original image pixels, as the image was received from the control specified in the **ImageDataSource** property.

RegLeft Property

Definition: Specifies the position of the left edge of the current Working Region.

Data Type: Long Integer

Syntax: `PixDispl.RegLeft = lngPosition`

Design Access: Not Available

Runtime Access: Read/Write

See Also: RegBottom Property, RegRight Property, RegTop Property, RightButtonOption Property

Comments: The following properties define the Working Region

RegBottom

RegLeft

RegRight

RegTop

These properties are collectively called the Working Region properties. They define the region of the image that is currently of interest. This region is the only portion of the image that is passed from this control when another ImageBASIC control names the Pixel Display control as its **ImageDataSource**. When an image is first loaded, these properties default to defining the entire image.

These properties may be set explicitly to define a Working Region. They are also updated when a mouse button that is set to either of the Rubber Band options is used to draw a region on the image. See the **LeftButtonOption** and **RightButtonOption** properties for information on these options.

Note: These properties are all in units of original image pixels, as the image was received from the control specified in the **ImageDataSource** property.

RegRight Property

Definition:	Specifies the position of the bottom edge of the current Working Region.
Data Type:	Long Integer
Syntax:	<code>PixDispl.RegRight = lngPosition</code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	RegLeft Property, RegBottom Property, RegTop Property, RightButtonOption Property
Comments:	<p>The following properties define the Working Region</p> <ul style="list-style-type: none">RegBottomRegLeftRegRightRegTop

These properties are collectively called the Working Region properties. They define the region of the image that is currently of interest. This region is the only portion of the image that is passed from this control when another ImageBASIC control names the Pixel Display control as its **ImageDataSource**. When an image is first loaded, these properties default to defining the entire image.

These properties may be set explicitly to define a Working Region. They are also updated when a mouse button that is set to either of the Rubber Band options is used to draw a region on the image. See the **LeftButtonOption** and **RightButtonOption** properties for information on these options.

Note: These properties are all in units of original image pixels, as the image was received from the control specified in the **ImageDataSource** property.

RegTop Property

Definition: Specifies the position of the bottom edge of the current Working Region.

Data Type: Long Integer

Syntax: `PixDispl.RegTop = lngPosition`

Design Access: Not Available

Runtime Access: Read/Write

See Also: RegLeft Property, RegRight Property, RegBottom Property, RightButtonOption Property

Comments: The following properties define the Working Region

RegBottom

RegLeft

RegRight

RegTop

These properties are collectively called the Working Region properties. They define the region of the image that is currently of interest. This region is the only portion of the image that is passed from this control when another ImageBASIC control names the Pixel Display control as its **ImageDataSource**. When an image is first loaded, these properties default to defining the entire image.

These properties may be set explicitly to define a Working Region. They are also updated when a mouse button that is set to either of the Rubber Band options is used to draw a region on the image. See the **LeftButtonOption** and **RightButtonOption** properties for information on these options.

Note: These properties are all in units of original image pixels, as the image was received from the control specified in the **ImageDataSource** property.

RightButtonOption Property

Definition:	Specifies the operation performed when the right mouse button is used to rubber band an image region.
Data Type:	Enumerated
Syntax:	<code>PixDispl.RightButtonOption = <i>enumOption</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	RightMouseBoxStyle Property, LeftButtonOption Property
Possible Values:	<ul style="list-style-type: none">0 Normal1 Rubber Band (Default)2 Proportional Rubber Band3 Zoom4 Proportional Zoom5 Drag Region6 Drag Tool
Comments:	<p>When an image region is selected using the mouse to drag out a rectangle on the display window, this property specifies what type of region is specified and how the region is drawn.</p> <p><i>0--Normal</i> does not automatically perform any action, but all of the mouse related events are enabled and the application developer is free to perform any operation.</p> <p><i>1--Rubber Band</i> selects a new Working Region when the mouse button is released. The region can be drawn with any aspect ratio.</p> <p><i>2--Proportional Rubber Band</i> selects a new Working Region, and the rectangle that is drawn will always have the aspect ratio that matches the aspect ratio of the display window.</p> <p><i>3--Zoom</i> sets the Zoom Region properties when the mouse button is released, enlarging the selected region to fill the display window. The region may be drawn with any aspect ratio, and the selected region will be fit to the display window as fully as possible.</p> <p><i>4--Proportional Zoom</i> sets the Zoom Region properties when the mouse button is release, enlarging the selected region to fill the display window. The region is drawn with the same aspect ratio and the display window, so the selected region always exactly fits the display window when it is enlarged.</p>

5--Drag Region allows the user to move the currently defined Working Region by dragging it while depressing the mouse button.

6--Drag Tool allows the user to move the zoomed image by dragging it while holding the depressed mouse button.

RightMouseBoxStyle Property

Definition:	If the RightButtonOption property is set to an option that allows rubber banding, this property specifies whether the area that is defined will be shaded or outlined while dragging.
Data Type:	Enumerated
Syntax:	<code>PixDispl.RightMouseBoxStyle = <i>enumOption</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	RightButtonOption Property, WorkingRegionStyle Property
Possible Values:	0 None 1 Hollow 2 Solid
Comments:	In order for this property to have any effect, the RightButtonOption must be set to a value other than <i>0--Normal</i> . <i>1--Hollow</i> draws an outline around the region that is being selected. <i>2--Solid</i> draws the rectangle as an inverted region on the image. <i>0--None</i> makes no visible change in the Display control to indicate the Working Region.

Rotation Property

Definition:	Specifies the orientation of the displayed image relative to the original image.
Data Type:	Enumerated
Syntax:	<code>PixDispl.Rotation = <i>enumOption</i></code>
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	Invert Property
Possible Values:	0 90 180 270
Comments:	If the image is passed from the Pixel Display control to another ImageBASIC component, the value of this property will effect the data. The image will be supplied to the destination control as it is displayed, either rotated or inverted.

ScaleToGray Property

Definition:	If True, bitonal images are displayed as grayscale to enhance readability.
Data Type:	Boolean
Syntax:	<code>PixDispl.ScaleToGray = <i>boolOption</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	ScaleToGrayLevel Property, Invert Property
Comments:	Results in darker, more highly contrasted display, because many small white areas pick up darkness and become much more visible on the screen. The outgoing image data is not changed by enabling this property; it affects only the display quality.

ScaleToGrayLevel Property

Definition:	Specifies the number of shades of gray that will be used when ScaleToGray is enabled.						
Data Type:	Enumerated						
Syntax:	<code>PixDispl.ScaleToGrayLevel = <i>enumOption</i></code>						
Design Access:	Read/Write						
Runtime Access:	Read/Write						
See Also:	ScaleToGray Property						
Possible Values:	<table><tr><td>0</td><td>Low</td></tr><tr><td>1</td><td>Medium</td></tr><tr><td>2</td><td>High</td></tr></table>	0	Low	1	Medium	2	High
0	Low						
1	Medium						
2	High						
Comments:	Using the options that display more shades of gray to represent the bitonal data usually results in a better image, but are slower than using fewer shades of gray.						

ScrollBars Property

Definition:	Specifies when scroll bars are displayed on the Pixel Display window when the image is zoomed.								
Data Type:	Enumerated								
Syntax:	<code>PixDispl.ScrollBars = <i>enumOption</i></code>								
Design Access:	Read/Write								
Runtime Access:	Read/Write								
See Also:	"Zooming and Scrolling" on page 10, ZoomLeft Property., ZoomRight Property, ZoomTop Property, ZoomBottom Property								
Comments:	The possible values for this property are as follows: <table><tr><td>0</td><td>No Scroll Bars</td></tr><tr><td>1</td><td>Vertical Scroll Bar Only</td></tr><tr><td>2</td><td>Horizontal Scroll Bar Only</td></tr><tr><td>3</td><td>Both Vertical and Horizontal Scroll Bars</td></tr></table>	0	No Scroll Bars	1	Vertical Scroll Bar Only	2	Horizontal Scroll Bar Only	3	Both Vertical and Horizontal Scroll Bars
0	No Scroll Bars								
1	Vertical Scroll Bar Only								
2	Horizontal Scroll Bar Only								
3	Both Vertical and Horizontal Scroll Bars								

SelectPrinter Method

Definition:	When called, a standard Windows printer selection dialog is displayed.
Parameters:	None
Syntax:	<code>PixDispl.SelectPrinter</code>
Returns:	None
See Also:	PrintImage Method, PrintRegion Method
Comments:	<p>The dialog that is opened by this method allows access to all of the configuration and setup dialogs that the selected printer driver allows. If no printer is selected before calling the printing methods, the Windows default printer will be used.</p> <p>After a printer is selected through the dialog, the following properties are updated to reflect the selected printer:</p> <ul style="list-style-type: none">PrinterDevicePrinterDriverPrinterPort

SetPicture Method

Definition:	Displays a DIB whose handle is specified as the parameter to the method.
Parameters:	<code>hPicture</code> Windowshandle to the bitmap to display
Syntax:	<code>PixDispl.SetPicture <i>Picture</i></code>
Return Values:	None
See Also:	GetScaledPicture Method, ImageDataSource Property
Comments:	<p>Under most circumstances, the Display control will accept its image data from another ImageBASIC control that is specified in the ImageDataSource property. However, using the SetPicture method, the Display control can also display a bitmap residing in memory whose handle is known.</p> <p>For example, the image displayed in a Picture Box control can be displayed in the Pixel Translations Display control using this method as illustrated here:</p> <pre>PixDispl.SetPicture Picture1.Picture</pre> <p>Note: After using the SetPicture method to display a DIB, the ImageDataSource property must be reset to again accept image data from an ImageBASIC control.</p>

StartPrintJob Method

Definition:	Starts a print job.
Parameters:	None
Returns:	None
See Also:	EndPrintJob Method
Comments:	All pages printed after this method is called will be held by the Print Manager until the EndPrintJob method is called. When that method is invoked, all of the images will be sent to the printer as a single job. When a network printer is used, this will prevent the insertion of other print jobs between the image pages.

StartPrintPage Method

Definition:	Creates a new virtual print page.
Parameters:	None
Syntax:	<code>PixDispl.StartPrintPage</code>
Returns:	None
See Also:	EndPrintPage Method
Comments:	The virtual print page that is created by this method will accept all future print commands and will merge the images that are printed to it. The virtual page will be released to the printer when the EndPrintPage method is called.

Once this method is called, the following properties are enabled. By using these properties, multiple images may be printed to different regions of the virtual page:

- PrintPageHeight
- PrintPageWidth
- PrintTargetTop
- PrintTargetBottom
- PrintTargetLeft
- PrintTargetRight

UsePrintAcceleration Property

Definition:	<p>If True, the Display control attempts to use the print accelerator board in the selected printer during all print operations.</p> <p>If False, the Display control assumes that no print acceleration is possible and uses the printer's standard Windows driver.</p>
Data Type:	Boolean
Syntax:	<code>PixDispl.UsePrintAcceleration = <i>BoolOption</i></code>
Design Access:	Not Available
Runtime Access:	Read/Write
Possible Values:	True False
See Also:	PrintImage Method, PrintRegion Method
Comments:	<p>Only Group 4 compression is currently supported, so this option should be used only with print accelerator hardware that supports Group 4 compression (Xionics XipPrint & XipPrint II do support it).</p> <p>Accelerated printing may not reduce the spooling time on the local machine -- depending upon RAM and hard drive space on the machine -- but it will typically reduce the time that is required to send image data to the printer, thereby allowing the printer to operate at near its rated speed.</p> <p>Note: Do not use accelerated printing for very complex images; e.g., image with a lot of halftoning or Floyd-Steinberg dithering. Complex images files can be larger after compression than after normal printing, thereby taking longer to print.</p> <p>If accelerated printing fails, the control will report an error, but it will print the page normally so that the page will be sent to the printer under all cases.</p> <p>Note: Xionics accelerators support only 300 DPI, so set the printer to 300 DPI for accelerated printing.</p>

WorkingRegionStyle Property

Definition: Specifies how the Working Region is highlighted, if at all.

Data Type: Enumerated

Syntax: `PixDispl.WorkingRegionStyle = enumOption`

Design Access: Not Available

Runtime Access: Read/Write

See Also: LeftMouseBoxStyle Property

Possible Values:

- 0 None
- 1 Hollow
- 2 Solid

Comments: When an image is displayed, the Working Region defaults to the entire image. Therefore, if this property is set to 2--Solid when a new image is loaded, the entire image will be highlighted.

0--None does not indicate the Working Region and clears any current highlighting.

1--Hollow draws a single, solid line around the perimeter of the Working Region.

2--Solid draws the Working Region in inverse color from the original image.

ZoomBottom Property

Definition:	Specifies the position of the bottom edge of the region selected for zooming, in original image pixels.
Data Type:	Long Integer
Syntax:	<i>lngPosition</i> = PixDispl.ZoomBottom
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	ZoomLeft Property, ZoomRight Property, ZoomTop Property, DisplayBottom Property, "Zooming and Scrolling" on page 10
Comments:	The following properties define the region of an image that is displayed:

ZoomBottom

ZoomLeft

ZoomRight

ZoomTop

These properties are collectively called the Zoom Region properties. They define the region of the image that is currently visible in the display window. When an image is first loaded, these properties default to displaying the entire image.

Changing or executing any of the following properties or methods will cause the Zoom Region to change, therefore changing the values of the Zoom Region properties:

ZoomToRegion Method

ZoomIn Method

ZoomOut Method

ZoomPercent Property

Note: The Zoom Region properties are all in units of original image pixels, as the image was originally received.

ZoomIn Method

Definition:	Changes the scaling of the displayed image by the percentage specified in the ZoomInOutChange property to display a smaller but enlarged portion of the image.
Parameters:	None
Syntax:	<code>PixDispl.ZoomIn</code>
Returns:	None
See Also:	ZoomOut Method, ZoomInOutChange Property, ZoomPercent Property
Comments:	When this method is executed, the Zoom Region properties ZoomPercent are updated to reflect the new display region. The Working Region is not affected by this method.

ZoomInOutChange Property

Definition:	Specifies the change in the display area when the ZoomIn or ZoomOut method is called.
Data Type:	Single precision floating point
Syntax:	<code>PixDispl.ZoomInOutChange = <i>sngChange</i></code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	ZoomIn Method, ZoomOutMethod, ZoomPercent Property
Comments:	This percentage is based on the size of the original image.

ZoomLeft Property

Definition: Specifies the position of the left edge of the region selected for zooming, in original image pixels.

Data Type: Long Integer

Syntax: *lngPosition* = PixDispl.ZoomLeft

Design Access: Not Available

Runtime Access: Read/Write

See Also: ZoomBottom Property, ZoomRight Property, ZoomTop Property, DisplayLeft Property, 'Zooming and Scrolling' on page 10

Comments: The following properties define the region of an image that is displayed:

ZoomBottom

ZoomLeft

ZoomRight

ZoomTop

These properties are collectively called the Zoom Region properties. They define the region of the image that is currently visible in the display window. When an image is first loaded, these properties default to displaying the entire image.

Changing or executing any of the following properties or methods will cause the Zoom Region to change, therefore changing the values of the Zoom Region properties:

ZoomToRegion Method

ZoomIn Method

ZoomOut Method

ZoomPercent Property

Note: The Zoom Region properties are all in units of original image pixels, as the image was originally received.

ZoomOut Method

Definition:	Changes the scaling of the displayed image by the percentage specified in the ZoomInOutChange property to display more of the image.
Parameters:	None
Syntax:	<code>PixDispl.ZoomOut</code>
Returns:	None
See Also:	ZoomInOutChange Property, ZoomPercent Property
Comments:	When this method is called, the Zoom Region properties are updated to reflect the currently displayed region. The Working Region is unaffected by this method.

ZoomPercent Property

Definition:	Reports what percentage of the image is currently visible.
Data Type:	Single precision floating point
Syntax:	<code>PixDispl.ZoomPercent = sngZoom</code>
Design Access:	Not Available
Runtime Access:	Read-only
See Also:	"The Zoom Region" on page 27, ZoomRatio Property, ZoomIn Method, ZoomOut Method
Comments:	The value of this property is independent of the size of the display window. Therefore, a very small window displaying an entire image will report a ZoomPercent of 100.

ZoomRatio Property

Definition:	Reports the scaling factor applied to the image for the current display.
Data Type:	Single precision floating point
Syntax:	<code>PixDispl.ZoomRatio = sngZoom</code>
Design Access:	Read/Write
Runtime Access:	Read/Write
See Also:	ZoomPercent Property
Comments:	The value of this property is the ratio between the pixel size of the original image and the pixel size of the displayed image. Because screen pixels are generally larger than image pixels, ZoomRatio of 1 (one) would appear substantially enlarged.

ZoomRight Property

Definition:	Specifies the right edge of the region selected for zooming, in original image pixels.
Data Type:	Long Integer
Syntax:	<i>lngPosition</i> = PixDispl.ZoomRight
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	ZoomLeft Property, ZoomBottom Property, ZoomTop Property, DisplayRight Property, 'Zooming and Scrolling' on page 10
Comments:	<p>The following properties define the region of an image that is selected for display:</p> <ul style="list-style-type: none">ZoomBottomZoomLeftZoomRightZoomTop <p>Changing or executing any of the following properties or methods will cause the Zoom Region to change, therefore changing the values of the Zoom Region properties:</p> <ul style="list-style-type: none">ZoomToRegion MethodZoomIn MethodZoomOut MethodZoomPercent Property

ZoomTop Property

Definition:	Specifies the top edge of the region selected for zooming, in original image pixels.
Data Type:	Long Integer
Syntax:	<i>lngPosition</i> = PixDispl.ZoomTop
Design Access:	Not Available
Runtime Access:	Read/Write
See Also:	ZoomLeft Property, ZoomRight Property, ZoomBottom Property, DisplayTop Property, Zooming and Scrolling on page 10
Comments:	The following properties define the region of an image that is displayed:

ZoomBottom

ZoomLeft

ZoomRight

ZoomTop

These properties are collectively called the Zoom Region properties. They define the region of the image that is currently visible in the display window. When an image is first loaded, these properties default to displaying the entire image.

Changing or executing any of the following properties or methods will cause the Zoom Region to change, therefore changing the values of the Zoom Region properties:

ZoomToRegion Method

ZoomIn Method

ZoomOut Method

ZoomPercent Property

Note: The Zoom Region properties are all in units of original image pixels, as the image was originally received.

ZoomToRegion Method

Definition:	Changes the Zoom Region (the portion of the image that is displayed) to any valid region of the image as specified in the parameter to the method call.
Parameters:	<i>left</i> Image pixel coordinate of left edge of zoom region <i>top</i> Image pixel coordinate of top edge of zoom region <i>right</i> Image pixel coordinate of right edge of zoom region <i>bottom</i> Image pixel coordinate of bottom edge of zoom region
Returns:	None
Syntax:	<code>PixDispl.ZoomToRegion <i>left, top, right, bottom</i></code>
See Also:	ZoomIn Method, ZoomPercent Property
Comments:	<p>The region that is specified for zooming may not be the exact area that is displayed because of different aspect ratios between the Display window and the region. The region will be scaled to fit the window as well as possible, and the entire region will always be displayed with the possibility of additional image data also being visible.</p> <p>After this method successfully executes, the following property values are updated:</p> <ul style="list-style-type: none">ZoomBottomZoomLeftZoomPercentZoomRatioZoomRightZoomTop

Appendix A : Error Reporting

Error Reporting and Trapping

Errors that occur in this control can be reported in two ways:

- A runtime error can be raised
- A built-in dialog can be displayed

The selection of the error reporting method is done in the Error event. In this event, the *CancelDisplay* parameter specifies how the error will be reported to the application.

The *CancelDisplay* parameter can be set to any of the following values:

CancelDisplay Value	Behavior
ibErrNoAction	Application continues without reporting the error; the Error event still occurs and any code in it will be processed.
ibErrDisplay	A built-in dialog box is displayed with the error number and message
ibErrRaise	A runtime error is raised and can be trapped using the development environment's error handling routine; for example, in Visual Basic, the On Error statement would be used.
ibErrDisplayAndRaise	The built-in dialog is displayed and the runtime error is raised

Any other code in the **Error** event is processed independent of other error reporting. Regardless of the trapping and handling of the runtime error, the **Error** event will behave the same. The **Error** event occurs for each error internal to the Display control. The error number and error message that describe the cause of the error are available as parameters to the event.

In addition to the information available as parameters to the **Error** event, ImageBASIC includes a standard error message box that will be displayed unless the *CancelDisplay* parameter of the event is set to True. If you are performing all error trapping using the *On Error* statement, the **Error** event should have no code except setting the *CancelDisplay* parameter to prevent the display of the standard ImageBASIC error message box.

The Error Event

The creation of a useful and powerful application will inevitably require the developer to identify and correct the source of errors that arise during the development process. The **Error** event is triggered whenever any error internal to the control occurs. This event reports the type of error that occurred and an error code.

It is also possible to trap all errors arising from both the ImageBASIC control and any other control or code in a routine by the use of the *On Error* statement. If this method of error trapping is used, the **Error** event can be used only to cancel the display of the standard ImageBASIC error message box.

Using this information, the developer can trap and recover from many errors without the user ever knowing that something has happened. More often, however, the application just requires additional information or configuration to perform the requested action. In this case, the developer can use the **Error** event to inform the operator of this need.

Parameters to the Error Event

The **Error** event is called with the following parameters of the specified data type:

Number As Long
Description As String
SCode As Long
Source As String
HelpFile As String
HelpContext As Long
CancelDisplay As Integer

Number Parameter

This parameter reports ImageBASIC's error code. The possible error numbers are listed in the section "Error Numbers" on page 106.

Description Parameter

This parameter provides a descriptive string that briefly explains the error and how to correct it.

SCode Parameter

This parameter is equal to the Visual Basic runtime error number. It is equal to the value reported in the *Number* parameter plus the constant `vbObjectError`.

HelpFile Parameter

This parameter reports a string of the file name of a help file that should contain more information on this error.

HelpContext Parameter

This parameter reports an integer Context ID in the help file named above where this specific error is discussed.

CancelDisplay Parameter

This parameter specifies how the error that caused this event to occur will be reported to the user/application. The following options are available:

CancelDisplay Value	Behavior
ibErrNoAction	Application continues without reporting the error; the Error event still occurs and any code in it will be processed.
ibErrDisplay	A built-in dialog box is displayed with the error number and message
ibErrRaise	A runtime error is raised and can be trapped using the development environment's error handling routine; for example, in Visual Basic, the On Error statement would be used.
ibErrDisplayAndRaise	The built-in dialog is displayed and the runtime error is raised

For example, if the application will trap and recover from runtime errors, a runtime exception may be the only error report that you wish to receive. In this case, set the *CancelDisplay* parameter as shown here:

```
Private Sub PixDispl_Error(ByVal Number As Long, ...,
    CancelDisplay As Integer)
    CancelDisplay = ibErrRaise
End Sub
```

If the application will perform all error recovery in the **Error** event, the other reporting of errors should be disabled and error handling performed in the event:

```
Private Sub PixDispl_Error(ByVal Number As Long, ...,
    CancelDisplay As Integer)
    CancelDisplay = ibErrNoAction
    Select Case Number
        ...perform error detection and recovery
    End Select
End Sub
```

Error Numbers

The following Error Numbers are the possible return values in the *Number* parameter of the **Error** event.

Error Number: 750

Error Message: *Message directly from technology libraries*

Comments: Any unmapped error from the PixTools libraries will cause this error to occur. The original message from the PixTools libraries will be reported. This message will include the original error number as listed in the section, Error Numbers from Pixel Translations Libraries' on page 108.

Error Number: 751

Error Message: "An image page is required for this method."

Comments: An attempt was made to perform an operation that requires an image, but no image is available. Open an image or create a new image page.

Error Number: 752

Error Message: "Image data is required for this method."

Comments: No image data is available from the ImageDataSource. Set this property to a valid ImageBASIC control and ensure that the source control is supplying an image.

Error Number: 753

Error Message: "Unable to get reference to image object."

Error Number: 754

Error Message: "Image handle lock failed."

Error Number: 755

Error Message: "Image handle unlock failed."

Error Number: 756

Error Message: "Image has bad image data size."

Error Number: 757

Error Message: "Unable to change/select printer during a virtual printing job."
Comments: An attempt was made to change the printer while a print job is being created. Execute the EndPrintJob or EndPrintPage method to clear the current job.

Error Number: 758

Error Message: "Print Error - A print job is already active."
Comments: An attempt was made to start a new print job while one is already in progress. End the current print job to continue.

Error Number: 759

Error Message: "Print Error - Print mode not set. (Software Error. Contact DHS)"
Comments: This error should not occur. Please document the steps to duplicate this error and contact Diamond Head Software Technical Support.

Error Number: 760

Error Message: "Print Error - Unknown print mode. (Software Error. Contact DHS)"
Comments: This error should not occur. Please document the steps to duplicate this error and contact Diamond Head Software Technical Support.

Error Number: 761

Error Message: "Failed to create a new image instance."
Comments: Generally caused by system configuration problems or a lack of system resources. Check hardware setup and free some system resources to continue.

Error Numbers from Pixel Translations Libraries

The following error codes can be returned from virtually any ImageBASIC function or subroutine call and, therefore, no attempt has been made to list the possible sources of difficulty. The solutions suggested with each error will usually resolve the difficulty, but any other solution you might devise is certainly not ruled out.

-4400 Not Ready	A device being requested by this function call is not ready. Check to make sure the device is ready and on-line.
-4401 No Page	The function could not find an image page to process. Make certain a valid Image Display Window is provided as the source of image data.
-4403 Don't Know	An error has occurred that has not been previously documented and identified with a particular error code.
-4404 Bad Setup	The application or function call has detected an invalid configuration. Check the computer configuration first, then the scanner and then the printer.
-4405 No Device	No hardware device (printer or scanner) could be detected. Check hardware connections, setup, and drivers.
-4408 Bad Parameter	One or more of the parameters to this function call are invalid. The parameters may be internal or they may be the arguments to the Visual Basic function call.
-4409 Not Open	This error is returned when a file that needs to be processed is not open. This usually occurs when an image should be displayed in the Image Display Window before processing.
-4410 Time-out	A print or scan function has failed because of a device time-out. Make sure the device has paper if a printer and a page to scan if a scanner.
-4413 Still Open	The function tried to open a file that was already open or detected a file that it expected to be closed.
-4415 File Read Err	An error occurred while reading the file. Make certain the file is not of an invalid format or corrupt.
-4421 EXE Header Bad	The header to an executable file is corrupt or unreadable. To correct the error, recompile the executable from within Visual Basic.
-4423 Bad Handle	An invalid handle has been requested by the function. Check that the control specified in the argument is valid and available.

-4426 Jam	The scanner or printer has suffered a paper jam. Clear the jam and call the function again to continue.
-4427 No Board	The software could not locate the appropriate board. This might occur if the board has been dislodged or its IRQ changed. Check the installation and setup of the board.
-4429 Cover	The cover is off or loose on the scanner or printer. Secure the cover and call the function again to continue.
-4430 Lamp	The lamp in the scanner is not functioning properly. It may be burned out or damaged.
-4431 Scanner	An error internal to the scanner has occurred. The scanner has not reported an exact error condition, only that some error has occurred.
-4432 Communication	A communication error occurred between the scanner and the application. Check installation and configuration of the scanner, scanner driver, and ASPI driver, if applicable.
-4433 Software	The software driver of the scanner is either not accessible or does not support the requested function. Make sure the latest version of the software is installed in the proper directory.
-4434 Bad Card	The scanner card (SCSI or proprietary) is damaged or is otherwise malfunctioning.
-4436 Overheat	The scanner has overheated and stopped. Wait a few minutes for the scanner to cool and try again. If this error occurs repeatedly, the cooling fan may be damaged or jammed.
-4437 Busy	A request for a scan has been received while the scanner is busy. Wait until the scanner has finished its current job before requesting another.
-4438 Not Now	The scanner cannot process the request right now either because it is busy, is warming up, or is self-testing. Check all these conditions and try the scan function again.
-4439 Procedure Halt	The scanning procedure was stopped by the operator or a preprogrammed command.
-4451 FTP Too Long	The File Transfer Protocol is too long to be properly processed by the ImageBASIC application. This error will usually occur only when scanning a large, complex page.

-4453 File Bad Format	The image file received by the application is an unsupported format or is corrupt.
-4454 File Read Error	An error occurred while the application was trying to read the file. Make certain the file is in a supported format and the drive is functioning properly.
-4458 FEOF	The end of the file was reached at an unexpected point.
-4459 Color	An attempt has been made to scan or display a color image when the current configuration will not support it. Or a color file was expected and not found.
-4460 Twain	A Twain driver is being used and either does not support the requested function or is not configured properly.
-4500 No Memory	The application ran out of memory while trying to interpret or display an image. This error can be returned when the image data is too large (the maximum allowable data size is 64K per dimension).
-4501 Parameter	A bad parameter was accessed during the function. This may be an internal error or may be an invalid argument supplied to the function call.
-4502 Bad Tag	An invalid or undefined tag name has been called. This error can be triggered by an attempt to either set or query this tag.
-4504 No PIXDFLT	The application was unable to locate the PIXDFLT.DLL file. Locate this file and ensure that it is in the \WINDOWS\SYSTEM directory and is the most recent version.
-4505 Access	The application does not have valid access rights to a file it tried to open or write.
-4506 Bad Path	The PATH supplied as part of the file name for either input or output is not accessible or does not exist.
-4507 Close	An error has occurred while trying to close a file. This error may be caused by a data error or a drive malfunction.
-4508 Create	An error occurred while the application tried to create a file. This error may be the result of a malfunctioning drive or an attempt to create a file with an invalid name.
-4509 Disk Full	The disk to which the application is trying to write is full. Delete or move files from this disk or write to another disk.

-4510 Handle	An attempt to locate an invalid or inaccessible handle has been made by the application. This error can occur if an attempt is made to use an unavailable or otherwise occupied control.
-4511 Too Many	Windows has run out of handles. Closing other applications will sometimes prevent this error from occurring.
-4512 Write	An error has occurred while writing to a file. This error could be the result of anything from a stray subatomic particle to a malfunctioning disk drive.
-4513 Bad Access	An attempt has been made to access an invalid drive or file. Make certain the application has access rights to the file and the drive is available.
-4514 Not Found	A file or drive that has been requested cannot be located. Make certain the file exists and is not hidden. Make sure network connection is active and properly functioning.
-4515 Open	An error has occurred while opening a file. This error could be the result of a corrupt file, malfunctioning drive, or bad parameters.
-4517 Seek	An attempt to find information inside a file has failed. The information may be absent or bad parameters may have been passed to the function call.
-4518 No Function	The application is unable to locate the function that has been called. Make sure ImageBASIC is properly installed and all files are accessible. This error can occur if a network drive is necessary but is temporarily unavailable.
-4519 Not Loaded	A driver is not loaded. This error could be the result of a printer, scanner, or display driver not being located. Make sure ImageBASIC is installed properly and any additional drivers required by the application are also properly installed and available.
-4520 End Stack	The end of a stack was reached while the application or function was still trying to access information.
-4521 End Page	The end of an image page was reached but the application expected it to continue.
-4523 Unknown	An error has occurred that is not defined by another error code.
-4525 No Current Scanner	An attempt to access a scanner, either for selection/setup or for scanning was made, but no scanner

	is available. Check scanner installation and configuration.
-4526 Cancel	A function was canceled by a command originating outside the function. This command may have been triggered by the operator or by another operation of the application.
-4527 Warning	This error code may be triggered if the application recognizes an old or corrupt file that still works but might not in the future.
-4528 Error	An undefined error has occurred.
-4531 Add Scanner	A scanner or scanner driver needs to be installed or reconfigured.
-4532 Put Environment	An error occurred while writing an environmental variable or setting a path.
-4533 Get Environment	An error occurred while accessing an environmental variable.
-4534 Same Device	An attempt was made to use the same device as the source and destination in a function. Make certain different devices are named as source and destination.
-4536 Can't Find	The application is unable to locate a requested device. Check the installation and configuration of the device that is being requested.
-4537 Restrict	The application has detected restricted access rights, that it does not have, to the requested device.
-4538 Can't Load	The application is unable to load the device driver. If this is a scanner, the LoadScanner method can be used to load the driver.
-4539 Driver Busy	A request was made to the device driver while it was occupied. Wait for the driver to complete its current task and call the function again.
-4540 System Version	An obsolete or invalid system version has been detected. Upgrade to a more recent version.
-4542 Locked	The file or drive the application tried to access is locked. Use another or unlock the desired device/file.
-4547 Pix Version	An outdated or obsolete Pixel file has been detected. Remove all old imaging related files from the computer, or ensure they are not on the Path.

- 4552 ASPI No Driver The application was unable to locate the ASPI driver. This error will be rectified by installing the ASPI driver according to the manufacturer's instructions.
- 4553 ASPI No I/O Control The ASPI driver was located but no I/O control was found. This error could be the result of an incorrect I/O address being supplied at installation or a conflict of I/O address. Check all addresses to make certain none is not duplicated.
- 4554 ASPI No Stack The ASPI driver could not find or access its stack.
- 4555 ASPI Call Fail The call to the ASPI driver failed. This could be the result of an invalid address, bad parameters to the function, or stray radiation.
- 4557 ASPI Aborted A call to the ASPI driver was aborted before it could finish. This error will occur if the function is stopped by the operator or by a programmed abort routine.
- 4558 ASPI Invalid A call was made to an invalid or inaccessible ASPI driver. Make certain the driver is installed and configured according to the manufacturer's instructions.
- 4560 ASPI Not Found This error will be triggered if no ASPI driver can be located by the application. To remedy this problem, check the installation of the driver, particularly the I/O address.
- 4561 ASPI Error An error occurred within the ASPI driver itself. If this error recurs repeatedly, you may need to reinstall the ASPI or contact the card manufacturer.
- 4562 ASPI No Memory The ASPI driver cannot find sufficient memory to properly function. Most ASPI drivers access conventional memory and increasing the free memory may help.

Appendix B : Color Definition

File and Compression Formats

Most of us do not have a lot of experience with image data. We may know that our favorite fax program produces PCX files, and that other programs produce files called TIFF files, but we may not know much beyond that. Many of the features in ImageBASIC will be much more easily understood if we present a short primer on how image data is captured, stored and maintained.

When a black and white image is captured at the scanner, it is held as raster data, or a stream of pixels that map together to form a picture. This data stream is relatively large; e.g., the pixels making up a bitonal 8.5" X 11" image scanned at 300 DPI (dots per inch) resolution will take up about 1 megabyte. The scanner usually sends this data either to the CPU or to a special board for compression.

As the data reaches the CPU there are numerous compression options. Each compression option presents different trade-offs with respect to compactness and decompression speed. For data that is being sent to the screen for display, we use a form of compression called Run Length Encoding (RLE). Run Length Encoding is a form of compressing the data that provides for very rapid decompression for quick display. RLE reduces the file size from 1 megabyte to about 250K, so it is somewhat more costly in terms of memory size than some other options.

By default, Diamond Head Software uses TIFF Group 4 compression in files and RLE in memory. Other options are available, however, and an overview of the various compression methods is given in the next section.

File Format Definitions

BMP File Format

A common Windows graphics format, BMP files can be read by almost any Windows-based image viewer. BMP files are paletted, and can represent up to 24-bit color. However, these files are not compressed and can therefore be very large when storing high color or high resolution images.

CALS File Format

CALS is a raster format for compressing bitonal image data. It was developed by the U.S. Department of Defense and is now in wide use throughout federal applications. The image data is either uncompressed or compressed using CCITT

Group 4 algorithm. The image data is appended to a raster header block, very much like TIFF files. Although common in U.S. government document imaging applications, CALS is uncommon elsewhere.

GIF File Format

GIF has been popularized by years of use on CompuServe and other Internet resources. Typically grayscale or paletted, GIF images are widely supported and provide reasonable compression -- a typical GIF file will be 20% of the uncompressed data size. This file format can encode an image up to 64K X 64K pixels using from 1 to 8 bits of color.

GIF files use LZW compression and are, therefore, not recommended for use unless you have entered into a licensing agreement with UNISYS Corp., holders of rights to LZW.

JPEG File Format

JPEG is a compression method originally developed to compress photographs. As might be expected, most JPEG files are color or grayscale, not bitonal. JPEG is a "lossy" compression, meaning that some resolution and details are lost when an image undergoes compression. However, the loss of resolution is usually not visible and does not discourage the use of JPEG.

JPEG can encode an image of up to 24-bit color to a maximum size of 64K X 64K pixels. Many image files can be compressed to just 10% of their original size with very little net loss.

PCX / DCX File Format

PCX was developed and distributed by Microsoft and ZSoft and is common throughout all Windows installations. Image data is compressed using an RLE scheme, and is therefore quick but not markedly efficient, particularly when storing high color images.

PCX can encode bitonal, 4-bit, 8-bit, or 24-bit color images up to 64K X 64K pixels. Typical compression for images or 16 colors or fewer is approximately 50%, while high color and complex images can actually be increased in size.

DCX is simply a version of PCX that allows for the storage of multiple images in a single file.

TIFF File Format

TIFF was initially developed by Aldus Corporation to standardize the storage of bitonal images in document imaging and desktop publishing applications. TIFF is a versatile format composed of a header record listing details of the image followed by the compressed image data. Image data in a TIFF can be compressed using a number of different algorithms, and the exact compression scheme is recorded in the header to allow for easier decompression.

TIFF Group 4, the default format for ImageBASIC, compresses typical documents to approximately 5% of the original size. Group 4 is a bitonal standard and will not reliably store any grayscale or color images.

Compression Types

The image data that is stored in files may be compressed using a number of different compression algorithms or not compressed at all. Some file types are, by their design or definition, restricted to using only one type of compression. GIF files are an example of this as all GIF files use LZW compression. Other file types, TIFF for example, can contain image data compressed in any one of many different formats. The following sections outline some of the more popular data compression schemes used to maximize the storage efficiency of image data.

Group 3 Compression

Group 3 compression is a standard developed by CCITT, a standards organization committee responsible for the sanctioning of many file compression methods. This technique uses a combination of RLE, differential, and Huffman encoding.

Group 4 Compression

CCITT Group 4 compression is similar to Group 3 except that Group 3 limits its compression to one dimension, while Group 4 compresses both horizontally and vertically. Group 4 compression results in the smallest file size of all the options available in ImageBASIC, which is why it is the default compression method for file saving.

LZW Compression

LZW is a compression algorithm first developed in the late 1970's that is dictionary-based. LZW compression substitutes portions of image data that have been seen before with shorter strings stored in its dictionary.

The two file types compressed by this method that ImageBASIC supports are GIF and TIFF LZW. These files can be either bitonal or color, but LZW compresses paletted images best.

Run Length Encoding (RLE)

RLE (Run Length Encoding) is a simple compression algorithm that encodes a run of identical pixels as a count and the pixel value. This compression will generally result in a file size approximately one-quarter the original size. RLE compression is very fast but relatively inefficient; however, because of its speed advantages, it is used as the compression method for images held in memory by ImageBASIC and also in BMP files.

Uncompressed Image Files

Uncompressed images are simply that -- no compression has been applied to the raw image data. Therefore, the image files tend to be very large. For example, the raw data to describe a bitonal letter size image at 300 DPI is about 1 MB. Adding to the area of the image or to the level of color can drastically increase the file size and the time required to transfer and display it.

Color Limitations of File Formats

The level of color that can be saved in each file format is a function of the specification of that format. For example, TIFF Group 4 is a bitonal standard, but JPEG can store up to 24-bit color. The next section introduces the definition of color in image files and contains tables that shown the level of color that can be saved in each file type.

Comparative Charts of Colors Supported by File Format

In the following tables, the entries in the top row of each table indicate the color format of an image file. The three numbers indicate, respectively, Bits Per Sample, Samples Per Pixel, and Photometric Interpretation.

The left-most column indicates the file format. By finding the intersection of the file type and color format, you can find the most appropriate, fully functional combination for your needs.

Bits Per Sample(BPS) indicates how many bits define the different values for each sample (see Samples Per Pixel) that defines a pixel's color. The only valid values are 1, 4, and 8.

Samples Per Pixel(SPP) indicates how many individual values are used to define a pixel's color. The only valid values are 1 and 3. A value of 1 indicates that this image is not RGB, leaving bitonal, grayscale, and paletted as possibilities. Bits Per Sample and Photometric Interpretation will indicate which of these possible formats is actually used.

Photometric Interpretation(PI) indicates the interpretation of color values for display. The valid values of Photometric Interpretation are as follows:

- | | | |
|---|----------|--|
| 0 | White 0 | Typical bitonal/gray interpretation in which 0 indicates white and higher numbers are darker shades. |
| 1 | White 1 | The inverse of White 0, in which 0 is interpreted as black and higher numbers are lighter shades. |
| 2 | RGB | Red, Green, and Blue values of each pixel are specified; this setting requires a value of 3 for Samples Per Pixel |
| 3 | Paletted | An array is constructed with a color assigned to each value in than array. This value is generally only applied to color images and is limited to few values than RGB. |

The key to the results codes in the tables is as follows:

- Y The combination file type and color definition is fully functional
- ~ Recognizable image but palette is not converted properly
- N This combination does not work at all

Bitonal and Grayscale Image File Formats

	Binary BPS 1 SPP 1 PI 0/1	16-level gray BPS 4 SPP 1 PI 0/1	256-level gray BPS 8 SPP 1 PI 0/1
TIFF Group 4	Y	N	N
TIFF Group 3	Y	N	N
TIFF Group 3 Modified	Y	N	N
TIFF Packed	Y	Y	Y
TIFF Uncompressed	Y	Y	Y
TIFF LZW	Y	Y	Y
CALS	Y	N	N
PCX	Y	N	Y
DCX	Y	N	N
PDA Uncompressed	Y	N	N
PDA Group 4	Y	N	N
PDA Group 3	Y	N	N
GIF	Y	N	Y
BMP	Y	N	Y
JPEG	N	N	N

Paletted and RGB Image File Formats

	16-color palette BPS 4 SPP 1 PI 3	256-color palette BPS 8 SPP 1 PI 3	24-bit RGB BPS 8 SPP 3 PI 2
TIFF Group 4	N	N	N
TIFF Group 3	N	N	N
TIFF Group 3 Modified	N	N	N
TIFF Packed	Y	Y	Y
TIFF Uncompressed	Y	Y	Y
TIFF LZW	Y	Y	N
CALS	N	N	N
PCX	N	Y	Y
DCX	N	N	N
PDA Uncompressed	N	N	N
PDA Group 4	N	N	N
PDA Group 3	N	N	N
GIF	Y	Y	N
BMP	Y	Y	N
JPEG	N	N	Y

Index

A

- AboutBox Method 47
- Accelerated Printing
 - (UsePrintAcceleration) 93
- AcceptDragFiles Property 47
- Active Property 48
- Annotation Control 1
- Assigning Mouse Button Functions 18

B

- Color of Fill 49
- Fill Color 49
- BackColor Property 49
- Blank Image Creation, NewImage
 - Method 73
- BlueBrightness Property 50
- BlueContrast Property 50
- BMP File Format 115
- BorderStyle Property 51
- Brightness Property 51
- Button Parameter 24

C

- CALS File Format 115
- Click Event 15, 22, 52, 57
- Color Brightness
 - BlueBrightness 50
 - GreenBrightness 62
 - RedBrightness 81
- Color Contrast
 - BlueContrast 50
 - GreenContrast 63
 - RedContrast 82
- Coloring the Working Region 35, 59
- Compression Types 117
- Connecting ImageBASIC Controls
 - Link Property 70
- Constants
 - dspEventClick 15, 57
 - dspEventDblClick 15, 57
 - dspEventError 15, 57
 - dspEventFilesDropped 15, 57

- dspEventImageDataChanged 15, 57
- dspEventMouseDown 15, 57
- dspEventMouseMove 15, 57
- dspEventMouseUp 15, 57
- dspEventPaint 15, 57

- Contrast Property 52
- Control Communication 1
- Control Handle
 - hWnd Property 63
- Creating a Blank Image
 - NewImage Method 73
- Creating a DIB 14
- Cursor Icon, MousePointer Property 72

D

- Data Input
 - ImageDataSource Property 6, 65
- DblClick Event 15, 22, 52, 57
- DCX File Format 116
- Descriptive Properties
 - ImageBitsPerSample 6, 64
 - ImageHeight 6, 66
 - ImageSamplesPerPixel 6, 66
 - ImageWidth 7, 66
 - ImageXRes 7, 67
 - ImageYRes 7, 67
- DIB Creation
 - GetScaledPicture Method 61
 - Picture Property 74
- Disable Events 14
- Disabling GUI Events 14
- Display Config
 - BlueBrightness 50
 - BlueContrast 50
 - Brightness 51
 - Contrast 52
 - GreenBrightness 62
 - GreenContrast 63
 - RedBrightness 81
 - RedContrast 82
- Display Configuration
 - Invert Property 13, 68
 - Rotation 89
 - Rotation Property 8
 - ScaleToGray Property 12, 89

- ScaleToGrayLevel Property 13, 90
- ZoomPercent Property 98
- Display of DIB
 - SetPicture Method 91
- Display Performance 14
- Display Properties
 - Invert 13
 - ZoomBottom 10, 28
 - ZoomLeft 10, 28
 - ZoomRight 10, 28
 - ZoomTop 10, 28
- Display Scaling
 - ZoomRatio Property 98
- DisplayBottom Property 53
- DisplayLeft Property 53
- DisplayRight Property 54
- DisplayTop Property 54
- DoClick Method 55
- Drag and Drop Files 58
- DragDrop Event 22
- DragOver Events 22
- dspEventClick Constant 15, 57
- dspEventDblClick Constant 15, 57
- dspEventError Constant 15, 57
- dspEventFilesDropped Constant 15, 57
- dspEventImageDataChanged Constant 15, 57
- dspEventMouseDown Constant 15, 57
- dspEventMouseMove Constant 15, 57
- dspEventMouseUp Constant 15, 57
- dspEventPaint Constant 15, 57

E

- Enable Events 14
- EndPrintJob Method 55
- EndPrintPage Method 55
- Error Dialog
 - CancelDisplay Parameter 105
- Error Event 15, 56, 57, 104
- Error Event Parameters
 - CancelDisplay 105
 - Description 104
 - HelpContext 105
 - HelpFile 105
 - Number 104
 - SCode 104
- Event Parameters
 - Button 24

- Error Event
 - CancelDisplay Parameter 105
 - Description Parameter 104
 - HelpContext Parameter 105
 - HelpFile Parameter 105
 - Number Parameter 104
 - SCode Parameter 104

- ImageX 24
- ImageY 24
- ScreenX 24
- ScreenY 24
- Shift 24
- Source 24

- Disable Events 57
- Enable Events 57
- Performance 57
- EventMask Property 57

- Events
 - Click 15, 22, 52, 57
 - DblClick 15, 22, 52, 57
 - DragDrop 22
 - DragOver 22
 - Error 15, 56, 57, 104
 - FilesDropped 15, 57, 58
 - ImageDataChanged 6, 15, 57, 65
 - MouseDown 15, 22, 57, 71
 - MouseMove 15, 22, 57, 72
 - MouseUp 15, 22, 57, 73
 - Paint 15, 57, 74

- Exporting Image Data 14
- Exporting Image Data, Picture Property 74

F

- FilesDropped Event 15, 57, 58
- FillWorkingRegion Method 35, 59
- FireGUIEvents Property 59
- FitToHeight Method 27, 30, 60
- FitToWidth Method 27, 30, 60
- FitToWindow Method 27, 30, 60

G

- GetRegBlackPercent Method 36, 61
- GetScaledPicture Method 61
- Getting Started 6
- GIF File Format 116
- Grayscale

- ScaleToGray Property 12, 89
- ScaleToGrayLevel Property 13, 90
- GreenBrightness Property 62
- GreenContrast Property 63
- Grouping pages for printing
 - StartPrintJob Method 42

H

- Handle
 - hWnd Property 63
- Highlighting Regions
 - Working RegionStylr Property 94
- hWnd Property 63

I

- Icon, Mouse
 - MouseIcon Property 71
- Image Characteristics
 - ImageBitsPerSample Property 6, 64
 - ImageHeight Property 6, 66
 - ImageSamplesPerPixel Property 6, 66
 - ImageWidth Property 7, 66
 - ImageXRes Property 7, 67
 - ImageYRes Property 7, 67
- Image for Display
 - SetPicture Method 91
- ImageBitsPerSample Property 6, 64
- ImageDataChanged Event 6, 15, 57, 65
- ImageDataSource 1, 2
- ImageDataSource Property 6, 65
- ImageHeight Property 6, 66
- ImageSamplesPerPixel Property 6, 66
- ImageWidth Property 7, 66
- ImageX Parameter 24
- ImageXRes Property 7, 67
- ImageY Parameter 24
- ImageYRes Property 7, 67
- Invert Property 13, 68

J

- JPEG File Format 116

L

- LeftButtonOption 27, 32, 34

- LeftButtonOption Property 19, 68
- LeftMouseBoxStyle Property 19, 69
- Licensing

- Active Property 48

- Link ID 2

- Link property 2, 70

- Linking Controls 1, 2

- ImageDataChanged Event 6, 65

- ImageDataSource Property 6, 65

- Load Time Improvement

- Active Property 48

M

- Margins for Printing 37

- Methods

- AboutBox 47

- DoClick 55

- EndPrintJob 55

- EndPrintPage 55

- FillWorkingRegion 35, 59

- FitToHeight 60

- FitToWidth 60

- GetRegBlackPercent 36, 61

- GetScaledPicture 61

- NewImage 73

- PrintImage 40, 77

- PrintRegion 41, 79

- Refresh 82

- SelectPrinter 38, 91

- SetPicture 91

- StartPrintJob 42, 92

- StartPrintPage 45, 92

- ZoomOut 98

- ZoomToRegion 101

- Mouse Button Options

- Normal 19

- Proportional Rubber Band 20

- Proportional Zoom 20, 21

- Rubber Band 20

- Zoom 20

- Mouse Configuration

- LeftButtonOption 19, 68

- Mouse Events

- Click 22, 52

- DbClick 22, 52

- DragDrop 22

- DragOver 22

- MouseMove 22, 71, 72

- MouseUp 22, 73
- Mouse Functionality 17
- Mouse Related Properties
 - LeftButtonOption 27, 32, 34
 - LeftMouseBoxStyle 19, 69
 - RightButtonOption 19, 27, 32, 34, 87
 - RightMouseBoxStyle 19, 88
- MouseDown Event 15, 22, 57, 71
- MouseIcon Property 71
- MouseMove Event 15, 22, 57, 72
- MouseMove Property, Enabling 59
- MousePointer Property 72
- MouseUp Event 15, 22, 57, 73
- Multi-page print jobs 42

N

- NewImage Method 73
- Normal, Mouse Button Option 19

P

- Paint Event 15, 57, 74
- Paint Property, Enabling 59
- Painting the Display Window
 - Paint Event 74
 - Refresh Method 82
- Pan Window 21, 31
- PCX File Format 116
- Percentage of Black Pixels 36, 61
- Performance Improvement 14
- Picture Property 74
- Pixel Density
 - GetRegBlackPercent Method 36, 61
- Pixel Translations Display Control 1
- Position of Image
 - DisplayBottom Property 53
 - DisplayLeft Property 53
 - DisplayRight Property 54
 - DisplayTop Property 54
- Print Jobs 42
- Printer Configuration
 - PrinterDC Property 75
 - PrinterDevice Property 39, 75
 - PrinterDriver Property 39, 76
 - PrinterPort Property 39, 77
 - SelectPrinter Method 38, 91
- PrinterDC Property 75

- PrinterDevice Property 39, 75
- PrinterDriver Property 39, 76
- PrinterPort Property 39, 77
- PrintImage Method 40, 77
- Printing
 - Multiple Images on One Page 45, 92
 - Multiple Page Print Jobs 92
 - Print Jobs 92
 - PrintImage Method 40, 77
 - PrintRegion Method 41, 79
 - StartPrintJob Method 42
- Printing Properties
 - PrintPageHeight 78
 - PrintPageWidth 78
 - PrintTargetBottom 79
 - PrintTargetLeft 80
 - PrintTargetRight 80
 - PrintTargetTop 81
- Printing, Accelerated
 - (UsePrintAcceleration) 93
- Printing, Margins 37
- Printing, Overview 37
- Printing, Scaling 37
- PrintPageHeight Property 45, 78
- PrintPageWidth Property 45, 78
- PrintRegion Method 41, 79
- PrintTargetBottom Property 45, 79
- PrintTargetLeft Property 45, 80
- PrintTargetRight Property 45, 80
- PrintTargetTop Property 45, 81
- Properties
 - AcceptDragFiles 47
 - Active 48
 - BackColor 49
 - BlueBrightness 50
 - BlueContrast 50
 - BorderStyle 51
 - Brightness 51
 - Contrast 52
 - DisplayBottom 53
 - DisplayLeft 53
 - DisplayRight 54
 - DisplayTop 54
 - EventMask 57
 - FireGUIEvents 59
 - GreenBrightness 62
 - GreenContrast 63
 - ImageBitsPerSample 6, 64
 - ImageDataSource 6, 65
 - ImageHeight 6, 66

- ImageSamplesPerPixel 6, 66
- ImageWidth 7, 66
- ImageXRes 7, 67
- ImageYRes 7, 67
- Invert 13, 68
- LeftMouseBoxStyle 19, 69
- MouseIcon 71
- MousePointer 72
- Picture 74
- PrinterDC 75
- PrinterDevice 39, 75
- PrinterDriver 39, 76
- PrinterPort 39, 77
- PrintPageHeight 45
- PrintPageWidth 45
- PrintTargetBottom 45
- PrintTargetLeft 45
- PrintTargetRight 45
- PrintTargetTop 45
- RedBrightness 81
- RedContrast 82
- RegBottom 32, 34, 83
- RegLeft 32, 34, 84
- RegRight 32, 34, 85
- RegTop 32, 34, 86
- RightMouseBoxStyle 19, 88
- Rotation 8
- ScaleToGray 12, 89
- ScaleToGrayLevel 13, 90
- ScrollBars 90
- UsePrintAcceleration 93
- WorkingRegionStyle 94
- ZoomBottom 95
- ZoomInOutChange 96
- ZoomPercent 98
- ZoomRatio 98
- ZoomRight 99
- ZoomTop 100
- Proportional Rubber Band, Mouse
 - Button Option 20
- Proportional Rubber Banding 18
- Proportional Zoom, Mouse Button
 - Option 20, 21

R

- Redaction
 - FillWorkingRegion Method 35, 59
- RedBrightness Property 81

- RedContrast Property 82
- Refresh Method 82
- RegBottom Property 32, 34, 83
- Region Highlight
 - WorkingRegionStyle Property 94
- RegLeft Property 32, 34, 84
- RegRight Property 32, 34, 85
- RegTop Property 32, 34, 86
- RightButtonOption 32, 34
- RightButtonOption Property 19, 27, 87
- RightMouseBoxStyle Property 19, 88
- Rotating Images
 - Rotation Property 89
- Rotation Property 8, 89
- Routing Data between Controls 1
- Rubber Band, Mouse Button Option 20
- Rubber Banding 17
- Runtime Linking 2

S

- ScaleToGray Property 12, 89
- ScaleToGrayLevel Property 13, 90
- ScreenX Parameter 24
- ScreenY Parameter 24
- ScrollBars Property 90
- Scrolling
 - Pan Window 21, 31
- SelectPrinter Method 38, 91
- SetPicture Method 91
- Shift Parameter 24
- Source of Image Data 1
- Source Parameter 24
- Standard Rubber Banding 18
- StartPrintJob Method 42, 92
- StartPrintPage Method 45, 92

T

- Thumbnails 14
 - GetScaledPicture Method 61
 - Picture Property 74
- TIFF File Format 117
- Timing of Licensing Verification
 - Active Property 48

U

- UsePrintAcceleration Property 93

V

Virtual Print Pages

- PrintPageHeight Property 45
- PrintPageWidth Property 45
- PrintTargetBottom Property 45
- PrintTargetLeft Property 45
- PrintTargetRight Property 45
- PrintTargetTop Property 45
- StartPrintPage Method 45, 92

W

Working Region Properties

- RegBottom 32, 34, 83
- RegLeft 32, 34, 84
- RegRight 32, 34, 85
- RegTop 32, 34, 86

Working Region, definition 32

WorkingRegionStyle Property 94

Z

Zoom Coordinates

- DisplayBottom Property 53
- DisplayLeft Property 53
- DisplayRight Property 54
- DisplayTop Property 54

Zoom Region Properties

- ZoomBottom 10, 28, 95
- ZoomLeft 10, 28, 97
- ZoomRight 10, 28, 99
- ZoomTop 10, 28, 100

Zoom, Mouse Button Option 20

ZoomBottom Property 10, 28, 95

ZoomIn Method 27, 29, 96

Zooming

- ZoomInOutChange Property 96

Zooming Configuration

- FitToHeight Method 27, 30, 60
- FitToWidth Method 27, 30, 60
- FitToWindow Method 27, 30, 60
- ZoomIn Method 27, 29, 96
- ZoomOut Method 27, 29, 98
- ZoomToRegion Method 101

ZoomInOutChange Property 96

ZoomLeft Property 10, 28, 97

ZoomOut Method 27, 29, 98

ZoomPercent Property 98

ZoomRatio Property 98

ZoomRight Property 10, 28, 99

ZoomTop Property 10, 28, 100

ZoomToRegion Method 101