

TMSSequoia Display ActiveX Control

User's Guide



DIAMOND HEAD 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, © 1994, 1995, 1996, 1997 Diamond Head Software, Inc. All rights reserved.

Portions of this product, © 1996, 1997 TMSSequoia, 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
Licensing Configuration and Verification	1
Linking of ImageBASIC Controls.....	3
Displaying an Image	5
Image Characteristics.....	5
Display Configuration.....	7
Scaling Images for Display	7
Caching of Images.....	8
Chapter 2: Controlling the Display	11
Optimizing the Display Window	11
Specifying the Image Type	11
Selecting a Scaling Quality	12
Selecting a Color Display Format	12
Creating Thumbnails	14
Basic Image Manipulation.....	16
Zooming and Scrolling.....	16
Setting the Display Orientation	17
Grayscale for Display.....	18
Inverting Images.....	19
Enabling and Disabling Events.....	20
Chapter 3: Controlling the Mouse	23
Mouse Functionality in ImageBASIC.....	23
Rubber Banding.....	23
Assigning Functions to the Mouse Buttons.....	24
Mouse Button Option Properties.....	25
Mouse Related Events.....	28
Chapter 4: Regions in ImageBASIC	33
The Zoom Region.....	33
Zoom Region Properties.....	34
Scrolling and Zooming	34
ZoomIn and ZoomOut Methods.....	35
Fitting the Image to the Display Window.....	35
Enabling and Disabling the Scroll Bars	36
Pan Window	36
Panning the Main Window	37
The Working Region	37
Visual Selection of the Working Region	37
Programmatic Selection of the Working Region.....	39

Applying the Working Region.....	41
Chapter 5: Printing from ImageBASIC	43
Overview of Printing	43
Image Scaling When Printing.....	43
Printer Selection and Setup.....	44
Standard Printer Dialogs.....	44
Programmatic Selection of a Printer	45
Printing a Single Page.....	46
Printing the Whole Image	46
Printing a Region of the Image.....	47
Advanced Printing Options.....	47
Print Jobs	48
Printing Multiple Images on One Page.....	50
Chapter 6: Reference	53
Properties, Methods and Events.....	53
Appendix A: Error Reporting	151
Error Reporting and Trapping.....	151
The Error Event	152
Error Numbers	154
Error Numbers -- TMSSequoia Display Control.....	154
Error Numbers -- TMSSequoia Libraries.....	158
Appendix B: Color Definition	163
File and Compression Formats	163
File Format Definitions.....	164
Compression Types.....	165
Color Limitations of File Formats	167
Index	169

Chapter 1: Introduction

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 typical toolkit evaluation and runtime distribution format).

Using 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 control 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 unique token for the TMSSequoia 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

The ImageBASIC licensing server 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 installed with ImageBASIC in the HARDKEY subdirectory.

- 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.

A file called IMGBASIC.INI must be in the Windows directory of each PC that is using an license database. This file contains an entry pointing to the database.

Creating a Licensing Database

Licensing databases can be installed in two ways:

- 1) If you are installing an evaluation copy of the ImageBASIC CD, run the registration utility that is accessible through the CD Navigator.
- 2) If you are installing a runtime database or are modifying an existing database, run a utility called the License Configuration Manager (LCM). The LCM is installed along with all ImageBASIC toolkit in the TOOLS subdirectory.

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.

When the ImageBASIC controls are loaded into a development environment, the licensing server will always immediately attempt to verify that the controls are licensed.

When the controls are loaded at runtime, the licensing server can be instructed to delay the verification of available licenses. For this purpose, each control has a property named **Active**. The **Active** property can be changed to False only at design time. If the property is False when a control initializes at runtime, licensing will not be verified and the control's technology libraries will not be loaded. This will make the initialization of an application somewhat faster.

Before using any control that was loaded without full initialization, the **Active** property must be set to True by the application. At this time, the licensing server will find and lock the requisite token and the technology libraries will be loaded. If the proper token cannot be found **Active** will be set back to False and the control cannot be used.

Licensing Token Release

When an application that has locked one or more tokens terminates normally, the tokens are released and can 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.
- If the licensing database has been installed locally, the locks are immediately released and will again be available when the application is started again.

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 TMSSequoia Display control cannot directly access image files on disk.
- The TMSSequoia File control must read the image files into memory where they can be retrieved by the Display control.
- Therefore, the source of images for the TMSSequoia Display control is the TMSSequoia File control. The communication between the controls is enabled by setting the TMSSequoia Display control's ImageDataSource to specify the TMSSequoia 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 TMSSequoia 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 TMSSequoia Display control -- to the Link property of the source control. For example, for a TMSSequoia Display control to accept image data from a TMSSequoia File control, set the ImageDataSource property as shown here for Visual Basic :

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

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

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

The links between ImageBASIC controls are made either at design time or at runtime. Changes may be made at any time during runtime if your application requires receiving image data or services from multiple controls.

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.

Displaying an Image

The TMSSequoia 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.

Selecting an Image for Display

To display an image file from disk, the TMSSequoia 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 TMSSequoia Display control will occur. The following code segment links the two controls and then loads a disk file for display:

```
TMSDispl.ImageDataSource = TMSFile1.Link  
TMSFile1.InputFileName = "c:\data\images\1001.tif"
```

Optimizing the Image Display

Because of differences in the size and complexity of image objects, applying the same scaling algorithm to textual documents and to photographs will result in less than ideal appearance on the screen. For this reason, the TMSSequoia Display control can perform customized image and color scaling for display depending upon the type of image that is selected. Refer to Chapter 2 for a discussion of the available options.

Once an image is displayed, the manner in which the image is displayed can be changed by inverting, rotating, zooming, and other similar options. Chapter 2 provides an introduction to all of the basic display options.

Image Characteristics

Every time that an image is loaded for display in the TMSSequoia 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 TMSSequoia 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 TMSSequoia 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
- Fitting the image to the full display window, or to its width or height
- 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 for saving, OCR, or other processing

Scaling Images for Display

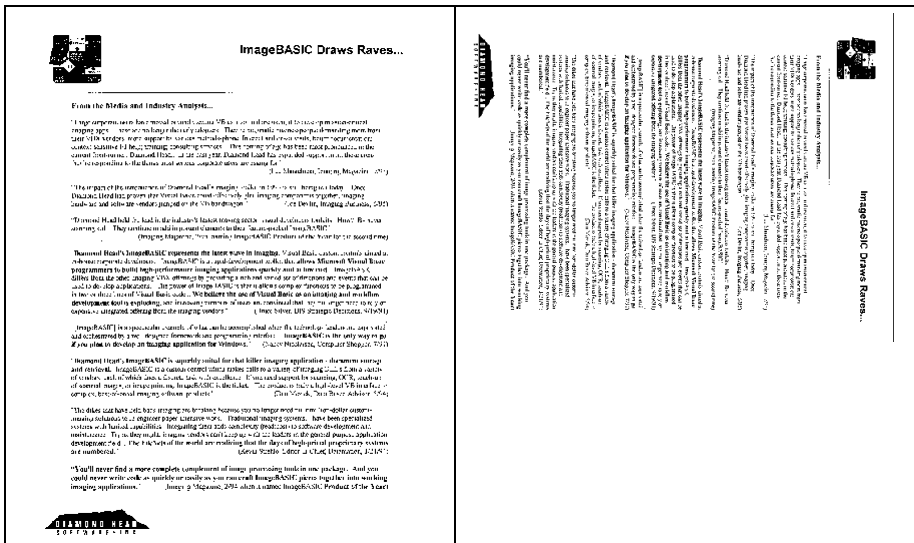
Black and white images are usually scanned between 200 DPI (dots per inch) and 400 DPI. Color images are generally scanned at 200 DPI or less. An 8.5" X 11" image scanned at 300 DPI will have a height of 3300 pixels and a width of 2550 pixels.

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.



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.

Chapter 2: Controlling the Display

Optimizing the Display Window

The TMSSequoia Display control can be optimized for the display of different image file types and color levels. Part of the optimization process is dependent upon the capabilities of the display and display controller, but the control performs all of the calculations necessary to present the image in the best possible manner.

The developer must tell the TMSSequoia Display control what type of image is being displayed and must select the balance between the appearance of the image and the speed with which it is displayed. The following options are available when optimizing the display window:

- Specifying the image type
- Specifying the scaling quality
- Selecting the color palette

Specifying the Image Type

Because of differences in the types of data that are supplied and the complexity of that data, different types of scaling need to be performed when the image is displayed. The **ImageType** property allows you to specify which type of image is displayed. This is an enumerated property with the following values:

- 0 Text Document
- 1 Line Art
- 2 Dithered (Photograph)

0--Textual Document is the default and should be used for most images that are composed primarily of text.

1--Line Art optimizes the display for finely detailed, bitonal images.

2--Dithered (Photograph) should be selected when displaying complex graphics composed of nearly continuous tones.

Selecting a Scaling Quality

When documents are scaled for display, very fine details can often be left out of the display image in favor of a quicker display time. When displaying bitonal images, the TMSSequoia Display control offers you the choice of displaying images at the fastest rate possible or slowing the display slightly but showing more of the fine detail. This selection is made through the **ScaleQuality** property, an enumerated property with these options:

- 0 Fastest
- 1 Slowest

0--Fastest allows the quickest display and refresh times.

1--Slowest can require a little more time to load an image, but the display is better when the image contains substantial fine detail.

Selecting a Color Display Format

When a color image is displayed, the screen colors may not match the original image colors due to the mapping of image colors to a system palette. For this reason, the following properties are available to modify the color conversion and display process:

ColorPaletteType	Specifies the type of display palette
ColorScalingMethod	Specifies the matching of the original image colors to the palette selected in ColorPaletteType
ColorReductionType	Specifies how color images are converted to bitonal

By default, the TMSSequoia Display control displays each image color as the nearest match in the system palette.

Select a Display Palette

The developer may select a different display palette to display color images as best suits a given application. The enumerated property **ColorPaletteType** allows the selection of the display palette, according to the following values:

- 0 System Palette
- 1 Optimal Palette
- 2 Rainbow Palette
- 3 Grayscale

0--System Palette is the default and displays each color in the original image as the closest match in the system palette. For high color displays or for images created on the same system, this option will generally result in nearly true color representation. This is the fastest option.

1--Optimal Palette instructs the Display control to calculate a new display palette that comes closest to matching both the system's display capabilities and the image's colors. This option is slower than the others but can supply good results when color representation is otherwise not accurate. Caution must be used when applying this option as some systems will encounter problems updating and maintaining the multiple palettes that are generated.

2--Rainbow Palette uses a custom palette of evenly differentiated colors that describe the full spectrum. For images with a wide color range that are not accurately represented using the system palette, this is a good option.

3--Grayscale displays color images as grayscale. Colors are matched to the most representative shade of gray for display. Grayscale optimizes the display of color images on monochrome monitors.

Match Image Colors to the Display Palette

When the image's color palette cannot be matched to the display system's color palette, the TMSSequoia Display control can be instructed how to apply the available colors through the **ColorScalingMethod** property. This is an enumerated property with these options:

- 0 Nearest
- 1 Linear
- 2 Bilinear

0--Nearest is the default and simply displays the image with the closest system color available.

1--Linear is similar to the above option except that next highest palette entry is always chosen.

2--Bilinear will dither the colors that do not match exactly, mixing the two closest display colors.

Use the *ColorReductionType* Property

When the TMSSequoia Display control must reduce the number of colors in an image to match the capabilities of the display hardware and software, the **ColorReductionType** property specifies how existing colors are converted.

- 0 Dither
- 1 Halftone
- 2 Threshold

0--Dither creates a dither pattern of the two closest display palette colors.

1--Halftone creates a newspaper-like halftone of black on white dots to simulate tones.

2--Threshold selects the single display palette closest to the original image color.

Creating Thumbnails

Any TMSSequoia Display window can be configured to display a thumbnail rather than a full-scale image. Each display window will display a single thumbnail. The relative resolution of the image that is displayed is configurable. Therefore, the resolution of the original image and the size of the thumbnail control can be accounted for to present the optimum image.

When a Display control shows a thumbnail, the size of the image that it holds in memory is much smaller than the image that would be maintained if a full-scale image were displayed. The scaling from the original image to the thumbnail image that is held in memory is specified by the **ThumbnailByteSize** property.

Display a Thumbnail Image in a TMSSequoia Display Control

Creating a thumbnail image is similar to displaying any image, except that the display control is instructed to read only a small portion of the total image data, thereby greatly reducing the amount of required memory.

To display a thumbnail of any image file, follow these steps:

- 1) Link the TMSSequoia Display control to any valid image data source:
`TMSDispl.ImageDataSource = TMSFile1`
- 2) Set the **ThumbnailByteSize** to specify the total size of the thumbnail bitmap. Values between 1000 and 50 000 are typical for bitonal files.
`TMSDispl.ThumbnailByteSize = 5000`
- 3) Set the **DisplayMode** property to indicate thumbnail display:
`TMSDispl.DisplayMode = 1`

In the example above, when the control named in the **ImageDataSource** property supplies any image data, the thumbnail control will copy 5KB of the image data to its local cache and display the reduced image.

The following sample code will dynamically create a series of thumbnail images, one for each page in a multi-page file. Note that two control arrays are being created -- one array of TMSSequoia Display windows and one array of TMSSequoia File controls.

```
TMSDispl.ImageDataSource = TMSFile1
TMSFile1.InputFileName = "c:\tiff\book3.tif"
intMaxPages = TMSFile1.PageCount
For i = 1 to intMaxPages
    Load thumbWin(i)
    Load thumbFile(i)
    thumbWin(i).ImageDataSource = thumbFile(i)
    thumbWin(i).ThumbnailByteSize = 10000
    thumbWin(i).DisplayMode = 1
    thumbWin(i).Width = 950
    thumbWin(i).Left = (1000 * i) - 850
    thumbWin(i).Visible = True
    thumbFile(i).InputFileName = "c:\tiff\book3.tif"
    thumbFile(i).PageIndex = i
Next i
```

Exporting Image Data to a DIB

The TMSSequoia 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 TMSSequoia Display control, as shown below for Visual Basic:

```
Picture1.Picture = TMSDispl.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 set to the Picture property of another control:

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

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 TMSSequoia 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 that we wish to 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:

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

Grayscaleing for Display

Grayscaleing 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 grayscaleing, 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.

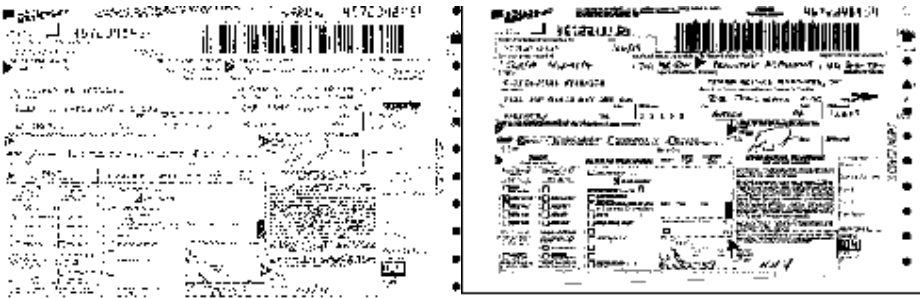


Image shown with ScaleToGray False

Image shown with ScaleToGray True

Enable ScaleToGray Display

The implementation of grayscaleing 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 grayscaleing.

```
TMSDispl.ScaleToGray = True
```

Note that grayscaleing 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.

Optimize 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.

```
TMSDispl.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.

Enable Image Inversion

The TMSSequoia 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, if **Invert** is enabled and the image data is saved through the TMSSequoia File control, the disk image will be inverted from the original.



Image shown before invert



Image shown after invert

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

```
TMSDispl.Invert = True
```

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

```
TMSDispl.Invert = False
```

Note: The TMSSequoia Display control can invert only bitonal (black and white) images. Only true bitonal image can be inverted; i.e., a color image that contains only black and white pixels cannot be inverted.

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 an 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:

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

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

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

The next page lists all of the events that can be enabled and disabled through the **EventMask** property. The table also lists the constants to use with this property.

The following events can be individually enabled and disabled with the **EventMask** property.

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

Chapter 3: 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 Button* on page 24, 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 the section "Zoom Region Properties" in Chapter 4 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 of the region that is defined is set by the **RBandAspect** property. This property defaults to the aspect ratio of the Display window, and is reset each time the window is resized.

3--Zoom

This option paints a standard rubberband in the display window. When the **MouseUp** event is fired, the Zoom Region properties are set. 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 the section "Scaling Images for Display" in Chapter 1.

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 TMSSequoia 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.

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 TMSSequoia 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 TMSSequoia Display control's **Click** event is called when one of the mouse buttons is pressed and released on the window.

The **Click** event has no parameters.

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.

The **DblClick** event has no parameters.

MouseDown Event

The **MouseDown** event is called when either mouse button is depressed while in the boundaries of the TMSSequoia 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 event will occur quite frequently during mouse movement; therefore, if much processing time is taken by the code in this event, system performance can be substantially decreased. Any code in this event should be deactivated, if possible, when it is not immediately in use to avoid unnecessary performance degradation.

Parameters to the **MouseMove** 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 the **MouseUp** event are as follows:

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

DragDrop Event

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

Parameters to the **DragDrop** event are as follows:

- Source
- X
- Y

DragOver Event

This event occurs when the mouse is used to drag any control over the TMSSequoia Display window.

Parameters to the **DragOver** event are as follows:

Source
X
Y
State

Mouse Event Parameters

In the discussion of 'Mouse Related Events' on page 28, 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 4: Regions in ImageBASIC

The Zoom Region

The **Zoom Region** is the portion of the entire image that is currently displayed by the TMSSequoia 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 TMSSequoia 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

Programmatic and GUI control of these coordinates is described in the next section "Scrolling and Zooming".

Scrolling and Zooming

Several techniques are available to control the scale of a displayed image:

- The **ZoomIn** and **ZoomOut** methods are available to allow zooming in or out by a given percentage of the current viewing area.
- The **ZoomToRegion** method will draw the specified region as large as possible in the display window.
- Three methods are also available to fit the image to the window according to different criteria. These methods will scale the image to completely fit the display window, to fit the width of the window, or to fit the height of the window. These methods are **FitToWindow**, **FitToWidth** and **FitToHeight**, respectively.

Scrolling through the image (or, altering the center of focus of the Zoom Region by an equal percentage on all sides without changing its width or height) can be accomplished through three techniques:

- By default, whenever the Zoom Region is set to include less than the entire image, scroll bars appear on the display window. The user can manipulate the scroll bars with the mouse.
- The Pan Window allows the user to drag a region on a thumbnail of the main image and scroll to the specified region.
- One of the mouse button options--*Drag Tool*, allows the user to directly drag the zoomed image around in the Display.

ZoomIn and ZoomOut Methods

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.

Fitting the Image to the Display Window

The TMSSequoia 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.

Three methods are available that scale the image to fit the window 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.

Enabling and Disabling the Scroll Bars

ImageBASIC displays scroll bars whenever the image in the TMSSequoia 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 TMSSequoia 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:
`TMSDispPan.ImageDataSource = TMSDispMain.Link`
- 2) Set the Pan Window's **ThumbnailByteSize** property to specify the image data size of the thumbnail displayed in the Pan Window:
`TMSDispPan.ThumbnailByteSize = 5000`

- 3) Set the Pan Window's **DisplayMode** property to 2--*Pan Window*
`TMSDispPan.DisplayMode = 2`
- 4) Set one of the mouse buttons on the Pan Window to 5--*Drag Region*
`TMSDispPan.LeftButtonOption = 5`
- 5) When any image is displayed in the TMSDispMain control, and the image is zoomed, the user can scroll through the main image by moving the highlighted region in the Pan Window.

Panning the Main Window

The **LeftButtonOption** and **RightButtonOption** properties each have the enumerated option 6--*Drag Tool*. When one of the mouse buttons is set to this option, and the image is zoomed, the user can depress and hold down the mouse button and then drag the image by moving the mouse cursor.

Usually, the image will be moved to maintain the cursor over the same point in the image. However, if the cursor is moved near the edge of the Display control, the image will continue to scroll in that direction.

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 3 shows how the mouse can be configured to select the Working Region. Briefly, if the **LeftButtonOption** or **RightButtonOption** property is set to 1--*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.

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.

Enabling 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.

Highlighting 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.

```
TMSDispl.RegLeft = 300
TMSDispl.RegTop = 300
TMSDispl.RegRight = 600
TMSDispl.RegBottom = 1200
```

Highlighting 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 37), the default behavior of the TMSSequoia 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.

Filling 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 Black
- 1 White

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

```
TMSDispl.FillWorkingRegion 0
```

Note: A behavior has been reported under Windows 95 using any HP LaserJet 4 printer driver in which the filled region is not printed as filled immediately after this method is executed. The region is printed as it was originally present in the

image. To correct this behavior, either save and load the image before printing or set the printer's Graphics Mode to Raster graphics (defaults to Vector graphics).

Analyzing 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 = TMSDispl.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, but the values are generally very high because most colors will be thresholded to black.

Applying the Working Region

When image data is passed from the TMSSequoia 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:

- The file access controls have a **Save** method, which writes the entire image to file, and the **SaveRegion** method, which writes only the image defined as part of the Working Region.

- The recognition controls have an **OCRPage** and an **OCRRegion** method which will process the specified portion of the image received from the Display control.
- The Pan Window, as described in the section titled 'Pan Window' on page 36, is based in part on the position of the Working Region in the Pan Window control.

Chapter 5: 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 Chapter 2.

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.

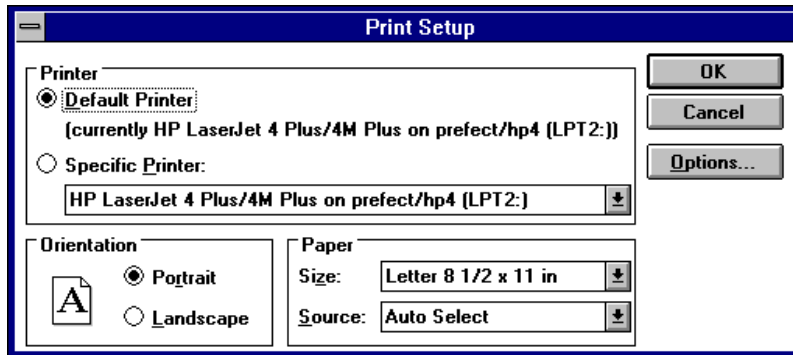
Standard Printer Dialogs

The printer selection and setup dialogs used by the TMSSequoia 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.

As detailed in the following sections, the application can select a printer through either a standard, Windows interface or through a series of properties that specify the printer driver, port and device. In either case, the selected printer must be installed according to its manufacturers instructions and an appropriate Windows printer driver installed.

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:



The printer may also be selected through code, rather than through the GUI shown above. Refer to "Programmatic Selection of a Printer" for details on this procedure.

The printer setup options are available through an 'Options...' or 'More...' button in the printer selection dialog. A standard Windows printer setup dialog, dependent on the current printer, will be displayed when this button is clicked.

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 instance of an application.

Any programmatically selected printer must be installed and configured properly before it is specified through these properties.

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:

```
TMSDispl.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:

```
TMSDispl.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:

```
TMSDispl.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 TMSSequoia Display control will use the Windows default printer.

The valid values for these properties can be found by using the **SelectPrinter** dialog to select a printer and then querying these properties. Alternatively, all installed printers are recorded in WIN.INI in the [Devices] section for Windows 3.x and Windows 95. These entries are as follows:

```
HP LaserJet 4 Plus=HPPCL5MS,LPT2:,15,45
```

The text to the left of the equal sign (=) is the Printer Device, in this case "HP LaserJet 4 Plus".

The first entry to the right of the equal sign is the Printer Driver, in this case "HPPCL5MS".

The Printer Driver is separated from the Printer Port ("LPT2:") by a comma.

Printing a Single Page

The basic printing functions of the TMSSequoia Display window allow printing of either the entire image or a portion of the image. If only a portion, or region, of the image is to be printed, it is the Working Region that is printed, and not any arbitrary region.

Printing the Whole Image

The **PrintImage** method sends the entire image in the TMSSequoia 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.

When sent to the printer, the image will be scaled as large as possible to fit the page size. The image will be printed as it is displayed -- inverted, rotated, etc. The command to print the currently displayed image is as follows:

```
TMSDispl.PrintImage
```

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 Chapter 4.

Unless the **StartPrintJob** method has been called, each region that is printed using this method will be sent to the print queue as an individual print job.

The image sent to the printer will be scaled as large as possible to fit the page size while maintaining its aspect ratio. 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.

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  
    TMSDispl.RegTop = 0  
    TMSDispl.RegLeft = 0  
    TMSDispl.RegRight = TMSDispl.ImageWidth  
    TMSDispl.RegBottom = (TMSDispl.ImageHeight / 2)  
    ' print the selected region  
    TMSDispl.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 an entire 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.

Operating on the same principles as cut and paste operations, the TMSSequoia Display window 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.

Print Jobs

To help prevent the mixing of print pages on a shared printer, 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.

```
TMSDispl.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 50.

EndPrintJob Signals completion of the print job and releases all pages to the printer.

```
TMSDispl.EndPrintJob
```

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

```
TMSDispl.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, the **AbortPrintJob** 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 Display control to a File control
TMSDispl.ImageDataSource = TMSFile1.Link

' load an image file which will display the
first

' page in the Display window
TMSFile1.LoadFile "c:\multpage.tif"

' start the print job -- this will hold all
print

' commands sent to the printer as a single job
TMSDispl.StartPrintJob

' sequentially load all of the images in the
' current file and then print each page
For intLoop = 1 to TMSFile1.PageCount
    TMSFile1.PageIndex = intLoop
    TMSDispl.Refresh
    TMSDispl.PrintImage
Next intLoop

' after all pages are in the print queue, send
the

' entire batch to the printer as a single job
TMSDispl.EndPrintJob
```

Printing Multiple Images on One Page

The TMSSequoia Display window 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 virtual page is created in memory by the **StartPrintPage** method. Once this method is invoked, all regions and pages printed using **PrintImage** and **PrintRegion** methods will be printed to the same virtual page, until there is a call to **EndPrintPage**.

When the **EndPrintJob** method is invoked, the page will be sent to the print spool as any other page. If called within a print job, the page will be held until

the job is finished. If no print job has been started, the page will be sent immediately to the printer.

Example: Specifying 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

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
TMSDispl.ImageDataSource = TMSFile1.Link

' begin the print page
TMSDispl.StartPrintPage

' load and display an image
TMSFile1.LoadFile "c:\1001.tif"

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

```

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

        ' load another image
TMSFile1.LoadFile "c:\tiff\2002.tif"

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

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

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

```


Chapter 6: Reference

Properties, Methods and Events

AboutBox Method

Definition

Displays the ImageBASIC About box supplying version and contact information.

Syntax

```
TMSDispl.AboutBox
```

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.

AbortPrintJob Method

Definition

Ends the current print job and removes existing pages from the print spool without sending to the printer.

Note: If printing to a network print spool, the pages already in the spool cannot be removed by this method.

Parameters

None

Syntax

```
TMSDispl.AbortPrintJob
```

Data Type

Short Integer

Returns

1 on success

0 on error

See Also

StartPrintJob Method, EndPrintJob Method

Comments

Each time the **StartPrintJob** method is executed, it must be followed by either a call to **EndPrintJob** (which will send the job to the printer) or **AbortPrintJob**

Note: Under Windows NT 4.0, calling this method during a virtual print job invalidates the printer device context (see **PrinterDC** property) held by the control. The TMSSequoia Display control then creates a new device context for the printer specified in the properties **PrinterDevice**, **PrinterDriver**, and **PrinterPort**. The new printer DC takes on the default settings for that printer, and does not inherit the settings (e.g., Orientation, Paper Source, etc.) from the old printer DC.

Under Windows 95, the printer DC is not invalidated, and the current printer settings are maintained.

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.

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

Data Type

Boolean

Syntax

```
TMSDispl.AcceptDragFiles = boolOption
```

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

If set to True at design time, the control will fully initialize and verify licensing immediately upon initialization of the runtime application.

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.

Data Type

Boolean

Syntax

```
TMSDispl.Active = boolOption
```

Design Access

Read/Write

Runtime Access

Read/Write (see limits below)

Comments

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.

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:

- 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.

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.

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.

AutoRefresh Property

Definition

Specifies whether or not the Display control will automatically refresh the screen when changes occur to the image.

Data Type

Boolean

Syntax

```
TMSDispl.AutoRefresh = True
```

Design Access

Read/Write

Runtime Access

Read/Write

Possible Values

True (Default)

False

See Also

Refresh Method

Comments

If this property is False, all screen refreshes of the Display window must be performed manually by executing the **Refresh** method.

If this property is True, most changes to the image will be automatically reflected on the screen. A notable exception is changes made to annotations which always require an explicit refresh before they will be displayed.

Click Event

Definition

Occurs when the user presses and then releases a mouse button over the TMSSequoia Display window.

Parameters

None

See Also

DbClick Event, MouseDown Event, MouseUp Event

Comments

This event will occur for any mouse button click, including dragging to define a region on the displayed image.

This event can be enabled and disabled through the **EventMask** property.

ColorPaletteType Property

Definition

Specifies the color palette that is applied to the displayed image.

Data Type

Enumerated

Syntax

```
TMSDispl.ColorPaletteType = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ColorScalingMethod Property

Possible Values

- 0 System
- 1 Optimal
- 2 Rainbow
- 3 Grayscale

Comments

When an image contains more colors than the display is capable of representing, the display window must calculate the palette to use for the display of that image. If extensive calculation is required to build a palette, the display of the image will be slower. The valid options for this property are as follows:

0--System uses the current system palette to display the image.

1--Optimal calculates the best possible palette for the image. This is the slowest option as the entire image must be analyzed prior to display.

2--Rainbow applies a palette of evenly separated colors.

3--Grayscale displays the image as grayscale. This affects only the display of the image but not the image data itself.

ColorReductionType Property

Definition

Specifies the method used by the control to reduce the number of displayed colors when the original image file contains more colors than the monitor can display.

Data Type

Enumerated

Syntax

```
TMSDispl.ColorReductionType = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ColorScalingMethod Property

Possible Values

- 0 Dither
- 1 Halftone
- 2 Threshold

Comments

0--Dither creates a dither pattern of the two closest display palette colors.

1--Halftone creates a newspaper-like halftone of dots to simulate tones.

2--Threshold selects the single display palette closest to the original image color.

ColorScalingMethod Property

Definition

Specifies the type of color conversion performed when the display colors do not exactly match the image colors.

Data Type

Enumerated

Syntax

```
TMSDispl.ColorScalingMethod = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ColorPaletteType Property

Possible Values

- 0 Linear
- 1 Dither
- 2 Halftone

Comments

When color images are displayed, the colors defined in the image file frequently cannot be exactly duplicated by the video adapter or monitor. The closest match that can be displayed is usually substituted. Other options are available through this property:

0--Nearest is the default and simply displays the image with the closest system color available.

1--Linear is similar to the above option except that next highest palette entry is always chosen.

2--Bilinear will dither the colors that do not match exactly, mixing the two closest display colors.

DblClick Event

Definition

Occurs when either mouse button double clicks on the TMSSequoia Display window.

Parameters

None

See Also

Click Event

Comments

The time in which the double click must occur is defined in the Windows setup.

This event can be enabled and disabled through the **EventMask** property.

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

```
lngPosition = TMSDispl.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

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

```
lngPosition = TMSDispl.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

Comments

The displayed image will not necessarily completely fill the Display window. In this case, the value of this property can be less than zero.

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.

DisplayMode Property

Definition

Specifies how the Display control will be used -- normally, as a thumbnail display or as a Pan Window.

Data Type

Enumerated

Syntax

```
TMSDispl.DisplayMode = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ThumbnailByteSize Property, LeftButtonOption Property

Possible Values

- 0 Normal
- 1 Thumbnail
- 2 Pan Window

Comments

Used in conjunction with the **ThumbnailByteSize** property, **DisplayMode** causes the Display control to show only a thumbnail of the image received by the control or allows the normal, full resolution display of the image data.

0--Normal allows regular display of images and is the default.

1--Thumbnail causes the control to scale the incoming image data to the size specified in the ThumbnailByteSize property and to display this lower-resolution image. By reducing the size of the image data, system resources are saved.

2--Pan Window causes scaling just like *1--Thumbnail*, but additionally allows the linking of this control to another Display control which must be specified in the **ImageDataSource** property. Refer to the section "Pan Window" in Chapter 4 for implementation details.

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

```
lngPosition = TMSDispl.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

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

```
lngPosition = TMSDispl.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

Comments

The displayed image will not necessarily completely fill the Display window. In this case, the value of this property can be less than zero.

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 the some of the image data extends above the Display window, when scroll bars will be displayed by default.

DoClick Method

Definition

Causes the **Click** event to occur.

Parameters

None

Syntax

```
TMSDispl.DoClick
```

Data Type

None

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

```
TMSDispl.EndPrintJob
```

Returns

None

See Also

StartPrintJob Method

Comments

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.

EndPrintPage Method

Definition

Ends the creation of a virtual print page, signaling its completion.

Parameters

None

Syntax

```
TMSDispl.EndPrintPage
```

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 for any error internal to this control.

Parameters

See below

Comments

Each error internal to this control causes the Error event to occur. In the Error event, additional error reporting can be allowed through the *CancelDisplay* parameter. See the section "Error Reporting and Trapping in Appendix A for details.

This event can be enabled and disabled through the **EventMask** property.

Parameters for this event are as follows:

<i>Number</i>	Reports the error number; see the section "Error Numbers in Appendix A for a list of possible values.
<i>Description</i>	Reports a descriptive string of the error
<i>SCode</i>	Reports the Visual Basic runtime error number, equal to Number plus the constant <code>vbObjectError</code> .
<i>Source</i>	Reports a string of the source of the error
<i>HelpFile</i>	Reports the name of a help file that should explain this error
<i>HelpContext</i>	Reports the help context ID in HelpFile for this error
<i>CancelDisplay</i>	Specifies whether or not a built-in message box and/or a runtime exception will report the error.

The following constants are defined for the *CancelDisplay* parameter:

ibErrNoAction (256)

No display or runtime error; all code in this event is processed normally, so error recovery can be performed in the **Error** event.

ibErrDisplay (1)

Only display built-in message; all code in the **Error** event is still processed.

ibErrDisplayAndRaise (3)

This is the default behavior. Display message box and raises runtime exception.

ibErrRaise (2)

Only raise runtime exception

EventMask Property

Definition

Enables and disables select events.

Data Type

Enumerated Integer

Syntax

```
TMSDispl.EventMask( eventID) = ibEventMaskEnable
```

Design Access

Not Available

Runtime Access

Read/Write

Possible Values

ibEventMaskEnable

ibEventMaskDisable

Comments

If set to *ibEventMaskEnable*, the event specified by the *eventID* will occur.

If set to *ibEventMaskDisable*, the event specified by the *eventID* 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:

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

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

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

The events that can be enabled and disabled through the **EventMask** property are shown in Table 1, below.

Table 1. *EventMask Constants*

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

The numerical values assigned to these 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 either mouse button double clicks on the TMSSequoia Display window.

Parameters

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

See Also

EventMask Property

Comments

Any dropped file must be opened by a linked File control to display it. The following code snippet will instruct the File control to build a new document composed of all of the files that are dropped:

```
Private Sub TMSDisp1_FilesDropped(ByVal FileCount As Long,
    ByVal FileName As Variant)
    For intLoop = 1 to FileCount
        If intLoop = 1 Then
            TMSFile1.LoadFile FileName(intLoop)
        Else
            TMSFile1.AppendFile FileName(intLoop)
        End If
    Next intLoop
End Sub
```

This event can be enabled and disabled through the **EventMask** property.

FillWorkingRegion Method

Definition

Fills the current Working Region with either white or black pixels.

Parameters

FillColor Enumerated specification of the fill color:

- 0 Black
- 1 White

Syntax

```
TMSDispl.FillWorkingRegion    FillColor
```

Returns

None

See Also

RegBottom Property

Comments

The currently defined Working Region will be filled with the specified color.

Note: A behavior has been reported under Windows 95 with an HP LaserJet 4 printer: The filled region is printed as originally present, not filled. To correct this behavior, either save and load the image before printing or set the printer's Graphics Mode to Raster graphics (defaults to Vector graphics).

FireGUIEvents Property

Definition

Enables or disables the **Paint** and **MouseMove** events.

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

Data Type

Boolean

Syntax

```
TMSDispl.FireGUIEvents = boolOption
```

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

```
TMSDispl.FitToHeight
```

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

```
TMSDispl.FitToWidth
```

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

```
TMSDispl.FitToWindow
```

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

```
lngPercent = TMSDispl.GetRegBlackPercent
```

Data Type

Long Integer

Returns

Integer percentage of black pixels in region

Comments

This method analyzes the Working Region and determines what percentage of the region is comprised of black pixels. Color image data will be converted to bitonal before the density is calculated.

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. Because of the rounding errors in the function, always expect to get at least 1% returned by the method.

GetScaledPicture Method

Definition

Creates a bitmap of the specified dimensions based on the currently displayed image and returns the Windows handle to the bitmap (DIB).

Parameters

<i>width</i>	Integer pixel width of the bitmap to create
<i>height</i>	Integer pixel height of the bitmap to create

Syntax

```
hDIB = TMSDispl.GetScaledPicture( width, height)
```

Data Type

Integer

Returns

hDIB Windows handle to the DIB that is created

See Also

SetPicture Method, DisplayMode Property

Comments

Typically, this method will be used to create a thumbnail image for display in another Visual Basic control such as a Picture Box. For example, the following code snippet will scale the currently displayed image to relatively small dimensions and display all pages of the current document in a control array of Picture Box controls:

```
Dim intLoop as Integer
TMSDispl.ImageDataSource = TMSFile1.Link
TMSFile1.InputFileName = "c:\images\1001.tif"
For intLoop = 1 to TMSFile1.PageCount
    TMSFile1.PageIndex = intLoop
    Load Picture(intLoop)
    Picture(intLoop).Left = intLoop * 125
    Picture(intLoop).Width = 100
    Picture(intLoop).Picture = TMSDispl.GetScaledPicture(75,
        90)
    Picture(intLoop).Visible = True
Next intLoop
```

hWnd Property

Definition

Reports the handle to the Display control.

Data Type

Long

Syntax

```
hWindow = TMSDispl.hWnd
```

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 = TMSDispl.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 three characteristics:

Bits Per Sample

Samples Per Pixel

Photometric Interpretation

Table 2, below, shows the most common color configurations.

Table 2. Common Color 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

SPP *Samples Per Pixel*

BPS *Bits Per Sample*

PI *Photometric Interpretation*

For a more detailed discussion of color definition, refer to the section "Color Limitations of File Format\$ in Appendix B.

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.

This event can be enabled and disabled through the **EventMask** property.

ImageDataSource Property

Definition

Specifies the control from which this Display control will received image data.

Data Type

String

Syntax

```
TMSDispl.ImageDataSource = ibControl.Link
```

Design Access

Read / Write

Runtime Access

Read / Write

See Also

ImageDataChanged Event

Comments

This property must specify an ImageBASIC control that can serve as the source of image data. The most common image sources are File and Scan controls, but other Display controls and TextBridge (during verification) are also valid sources.

ImageHeight Property

Definition

Reports the number of pixels in the height of the currently displayed image.

Data Type

Integer

Syntax

```
intHeight = TMSDispl.ImageHeight
```

Design Access

Not Available

Runtime Access

Read-only

See Also

ImageWidth Property, ZoomHeight Property

Comments

The height of the 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

Integer

Syntax

```
intSPP = TMSDispl.ImageSamplesPerPixel
```

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 the section "Color Limitations of File Formats" in Appendix B.

ImageType Property

Definition

Optimizes the display of images based on the type of data contained in them.

Data Type

Enumerated

Syntax

```
enumType = TMSDispl.ImageType
```

Design Access

Read/Write

Runtime Access

Read/Write

Possible Values

- 0 Textual Document
- 1 Line Art
- 2 Photograph

Comments

The TMSSequoia Display control will use one of three display scaling algorithms, each of which is optimized to display a different type of image. The default *Textual Document* is the most commonly used, but continuous tone image such as photographs will be more clear and more appealing if displayed after setting **ImageType** to 2--*Photograph*

ImageWidth Property

Definition

Reports the number of pixels in the width of the currently displayed image.

Data Type

Integer

Syntax

```
intWidth = TMSDispl.ImageWidth
```

Design Access

Not Available

Runtime Access

Read-only

See Also

ImageHeight Property, ImageYRes Property

Comments

The width that is supplied 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

```
intRes = TMSDispl.ImageXRes
```

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

```
intRes = TMSDispl.ImageYRes
```

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

```
TMSDispl.Invert = boolOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

Rotation Property, ScaleToGray Property

Comments

Any image data passed from the TMSSequoia 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.

Note: Only true bitonal image can be inverted; i.e., bits per sample must be 1 (one), samples per pixel must be 1 (one), and photometric interpretation may be 1 (one) or 0 (zero).

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

```
TMSDispl.LeftButtonOption = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

LeftMouseBoxStyle Property, RightButtonOption Property

Possible Values

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

Comments

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.

0--Normal 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.

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 specified by the

RBandAspect property, which defaults to the same aspect ratio as 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.

LeftMouseBoxStyle 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

```
TMSDispl.LeftMouseBoxStyle = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

LeftButtonOption Property, RightMouseBoxStyle 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

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

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 the currently displayed annotations into the underlying image.

Important: If the underlying image is not bitonal, it will be converted to bitonal by the merge.

Parameters

None

Syntax

```
TMSDispl.MergeNotes
```

Data Type

None

Returns

None

Comments

An Annotation control that is currently displaying notes must be linked to the Display control before this method is called. After merging, the new image page should be saved.

Note: The merged annotation objects are completely merged into the image data, and cannot be further manipulated as annotation objects.

MouseDown Event

Definition

Occurs each time either of the mouse buttons is depressed on the TMSSequoia 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.

This event can be enabled and disabled through the **EventMask** property.

MouseIcon Property

Definition

Specifies the icon to display for the mouse cursor. Valid only **MousePointer** property is *99--Custom*.

Data Type

Long Integer

Syntax

```
TMSDispl.MouseIcon = LoadPicture( filename)
```

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 TMSSequoia 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

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.

This event can be enabled and disabled through the **EventMask** property.

MousePointer Property

Definition

Specifies the mouse pointer style.

Data Type

Integer (Enumerated)

Syntax

```
TMSDispl.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 Table 3, below.

Table 3. Valid values for the *MousePointer* property.

Constant	Value	Description
ibDefault	0	Default (Defined by parent or Screen)
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
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. (Only in 32-bit VB 4.0. and later)
ibArrowQuestion	14	Arrow and question mark. (Only in 32-bit VB 4.0 and later)
ibSizeAll	15	Size all. (Only in 32-bit VB 4.0. and later)
ibCustom	99	Custom icon specified by the <i>MouseIcon</i> 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.

This event can be enabled and disabled through the **EventMask** property.

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

Syntax

```
TMSDispl.NewImage width, height, xres, yres
```

Data Type

Long Integer

Returns

None

Comments

The image page that is created is bitonal.

Paint Event

Definition

Occurs each time the TMSSequoia 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 through the **EventMask** property.

PrinterDC Property

Definition

Reports the device context of the current printer.

Data Type

Long

Syntax

```
lngDC = TMSDispl.PrinterDC
```

Design Access

Not Available

Runtime Access

Read-only

See Also

PrinterDevice Property, PrinterDriver Property, PrinterPort Property,
StartPrintJob Method

Comments

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. The currently selected printer is reported in the **PrinterDevice**, **PrinterDriver**, and **PrinterPort** properties.

PrinterDevice Property

Definition

Specifies the name of the currently selected printer device.

Data Type

String

Syntax

```
TMSDispl.PrinterDevice = strDevice
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

PrinterDriver Property, PrinterPort Property, SelectPrinter Method

Comments

When any of the printing related methods is called, the TMSSequoia 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.

PrinterDriver Property

Definition

Specifies the driver name for the currently selected printer device.

Data Type

String

Syntax

```
TMSDispl.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 Display control will attempt to use the printer specified in these 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

```
TMSDispl.PrinterPort = strPort
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

PrinterDriver Property, PrinterDevice Property, SelectPrinter Method

Comments

When any of the printing related methods is called, the TMSSequoia 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.

PrintImage Method

Definition

Prints the entire image, regardless of any Working Region or Zoom Region definition.

Parameters

None

Syntax

```
TMSDispl.PrintImage
```

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.

The Windows default printer is used if none has been selected by the following techniques:

The **SelectPrinter** method opens a standard printer selection dialog, allowing the user to graphically select the printer of choice.

If the following properties are populated with valid values, the printer described will be used. These properties are populated when the **SelectPrinter** method is used to select a printer.

PrinterDevice

PrinterDriver

PrinterPort

This method can be used within either a print page or a print job. Refer to the section "Advanced Printing Options" in Chapter 5 for more information on these subjects.

PrintPageHeight Property

Definition

Reports the height of a virtual print page created by the **StartPrintPage** method.

Data Type

Long

Syntax

```
lngHeight = TMSDispl.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

Integer

Syntax

```
lngWidth = TMSDispl.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 using the current printer.

Parameters

None

Syntax

```
TMSDispl.PrintRegion
```

Return Values

None

See Also

PrintImage Method, SelectPrinter Method

Comments

The region will be scaled to fit the output page as well as possible. 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.

The Windows default printer is used if none has been selected by the following techniques:

The **SelectPrinter** method opens a standard printer selection dialog, allowing the user to graphically select the printer of choice.

If the following properties are populated with valid values, the printer described will be used:

PrinterDevice

PrinterDriver

PrinterPort

This method can be used within either a print page or a print job. Refer to the section "Advanced Printing Options" in Chapter 5 for more information on these subjects.

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

```
TMSDispl.PrintTargetBottom = lngPosition
```

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

Syntax

```
TMSDispl.PrintTargetLeft = lngPosition
```

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

Syntax

```
TMSDispl.PrintTargetRight = lngPosition
```

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

Syntax

```
TMSDispl.PrintTargetTop = lngPosition
```

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.

RBandAspect Property

Definition

Specifies the aspect ratio of the region that is drawn when a mouse button is set to rubberband.

Data Type

Single precision floating point

Syntax

```
TMSDispl.RBandAspect = sngRatio
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

LeftButtonOption Property, RightButtonOption Property

Comments

This property defaults to the aspect ratio of the Display control, and it is reset to that value every time the control is sized.

Refresh Method

Definition

Displays all changes made to the image.

Parameters

None

Syntax

```
TMSDispl.Refresh
```

Returns

None

See Also

Paint Event

Comments

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.

RegBottom Property

Definition

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

Data Type

Long Integer

Syntax

```
TMSDispl.RegBottom = lngPosition
```

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 TMSSequoia 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.

RegLeft Property

Definition

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

Data Type

Long Integer

Syntax

```
TMSDispl.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 TMSSequoia 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

```
TMSDispl.RegRight = lngPosition
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

RegLeft Property, RegBottom 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 TMSSequoia 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

```
TMSDispl.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 TMSSequoia 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

```
TMSDispl.RightButtonOption = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

RightMouseBoxStyle Property, LeftButtonOption Property

Possible Values

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

Comments

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.

0--Normal 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.

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 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.
- 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.

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

```
TMSDispl.RightMouseBoxStyle = enumOption
```

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 *0--Normal*. *1--Hollow* draws an outline around the region that is being selected. *2--Solid* draws the rectangle as an inverted region on the image. *0--None* 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

```
TMSDispl.Rotation = enumOption
```

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 TMSSequoia 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.

ScaleQuality Property

Definition

Specifies the quality of scaling algorithm applied when displaying bitonal images.

Data Type

Enumerated

Syntax

```
TMSDispl.ScaleQuality = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ColorPaletteType Property, ScaleToGray Property

Possible Values

0 Fastest

1 Slowest

Comments

0--Fastest displays the image as quickly as possible, possibly at the loss of some detail in the scaling process.

1--Slowest requires slightly more time to display a bitonal image, but the final display quality is slightly better in most cases.

ScaleToGray Property

Definition

If True, bitonal images are displayed as grayscale to enhance readability.

Data Type

Boolean

Syntax

```
TMSDispl.ScaleToGray = boolOption
```

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 Integer

Syntax

```
TMSDispl.ScaleToGrayLevel = enumOption
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ScaleToGray Property

Possible Values

- 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 the position of scroll bars on the Display window.

Data Type

Enumerated Integer

Syntax

`TMSDispl.ScrollBars` *enumOption*

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ZoomLeft Property., ZoomRight Property, ZoomTop Property, ZoomBottom Property

Comments

Unless otherwise specified, scroll bars are drawn on the Display window whenever the image is zoomed. However, other options for scrolling (i.e., the Pan Window and the Grab Tool) are available. Scroll bars, therefore, can be disabled if appropriate for the application.

The valid values for the **ScrollBars** property are as follows:

- 0 No Scroll Bars
- 1 Vertical Scroll Bar Only
- 2 Horizontal Scroll Bar Only
- 3 Both Vertical and Horizontal Scroll Bars (Default)

ScrollBy Method

Definition

Scrolls the image by the specified distance.

Parameters

HorzDistance Horizontal distance to scroll in image pixels
VertDistance Vertical distance to scroll in image pixels

Syntax

```
TMSDispl.ScrollBy HorzDistance, VertDistance
```

Data Type

Boolean

Return Values

True on success
False on error

Comments

This method scrolls the image in the display window by the specified horizontal or vertical distances, specified in image coordinates, if the image is displayed zoomed; i.e., not fit to window.

The method returns a non-zero value if it succeeds in scrolling in at least one of the directions (horizontal or vertical).

The method returns 0 if it failed to scroll at all. This could definitely happen under the following conditions:

- If an image was being displayed but the image was zoomed out or was displayed fit to window
- If both the *HorzDistance* and the *VertDistance* specified to it are 0 (zero)
- If no image was being displayed

A positive value for *HorzDistance* scrolls the image to the left in relation to the display window, thereby revealing more of the image on the right.

A negative value for *HorzDistance* scrolls the image to the right relative to the display window, thereby revealing more of the image on the left.

A positive value for *VertDistance* scrolls the image up relative to the display window, thereby revealing more of the image towards its bottom.

A negative value for *VertDistance* scrolls the image down relative to the display window, thereby revealing more of the image towards its top.

SelectPrinter Method

Definition

When called, a standard Windows printer selection dialog is displayed.

Parameters

None

Syntax

```
TMSDispl.SelectPrinter
```

Return Values

None

See Also

PrintImage Method, PrintRegion Method

Comments

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.

After a printer is selected through the dialog, the following properties are updated to reflect the selected printer:

PrinterDevice

PrinterDriver

PrinterPort

SetPicture Method

Definition

Displays the specified DIB.

Parameters

hPicture Windows handle to the bitmap to display

Syntax

```
TMSDispl.SetPicture    hPicture
```

Returns

None

See Also

GetScaledPicture Method, ImageDataSource Property

Comments

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.

For example, the image displayed in a Picture Box control can be displayed in the TMSSequoia Display control using this method as illustrated here:

```
TMSDispl.SetPicture Picture1.Picture
```

Note: When the **SetPicture** method is used to display an image, the **ImageDataSource** property is cleared. In order to again accept data from another ImageBASIC control, the **ImageDataSource** property must be set again.

StartPrintJob Method

Definition

Starts a print job.

Parameters

None

Syntax

```
TMSDispl.StartPrintJob
```

Data Type

Long Integer

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

```
TMSDispl.StartPrintPage
```

Data Type

Long Integer

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

ThumbnailByteSize Property

Definition

Specifies the total byte size of the thumbnail image that is held in memory when the **DisplayMode** property specifies a thumbnail or Pan Window mode.

Data Type

Long Integer

Syntax

```
TMSDisp1.ThumbnailByteSize = lngSize
```

Design Access

Not Available

Runtime Access

Read/Write

Possible Values

Positive, non-zero integers

See Also

DisplayMode Property

Comments

This property must be set before **DisplayMode** is changed from the default of *Normal*.

UsePrintAcceleration Property

Definition

If True, the Display control attempts to use the print accelerator board in the selected printer during all print operations.

If False, the Display control assumes that no print acceleration is possible and uses the printer's standard Windows driver.

Data Type

Boolean

Syntax

```
TMSDispl.UsePrintAcceleration = BoolOption
```

Design Access

Not Available

Runtime Access

Read/Write

Possible Values

True

False

See Also

PrintImage Method, PrintRegion Method

Comments

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).

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.

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, taking longer to print.

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.

Note: Xionics accelerators support only 300 DPI, so set the printer to 300 DPI for accelerated printing.

WorkingRegionStyle Property

Definition

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

Data Type

Enumerated Integer

Syntax

```
TMSDispl.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

```
lngPosition = TMSDispl.ZoomBottom
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

ZoomLeft Property, ZoomRight Property, ZoomTop Property, DisplayBottom Property

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

ZoomIn Method

Definition

Zooms in on (enlarges) the displayed image by the percentage specified in the **ZoomInOutChange** property.

Parameters

None

Syntax

```
TMSDispl.ZoomIn
```

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

```
TMSDispl.ZoomInOutChange = sngPercent
```

Design Access

Read/Write

Runtime Access

Read/Write

See Also

ZoomIn Method, ZoomOut Method, ZoomPercent Property

Comments

This percentage is based on the size of the current displayed region, not on the size of the original image. Therefore, executing a **ZoomIn** method and then a **ZoomOut** method with the same **ZoomInOutChange** value will not return the image to exactly the same display scaling.

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 = TMSDispl.ZoomLeft
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

ZoomBottom Property, ZoomRight Property, ZoomTop Property, DisplayLeft Property

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

ZoomOut Method

Definition

Zooms out on the displayed image by the percentage specified in the **ZoomInOutChange** property to display more of the image.

Parameters

None

Syntax

```
TMSDispl.ZoomOut
```

Returns

None

See Also

ZoomIn Method, 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

Returns/sets the current zoom percentage.

Data Type

Single precision floating point

Syntax

```
TMSDispl.ZoomPercent = sngPercent
```

Design Access

Not Available

Runtime Access

Read-only

See Also

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

```
TMSDispl.ZoomRatio = sngPercent
```

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, a **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

```
lngPosition = TMSPixDispl.ZoomRight
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

ZoomLeft Property, ZoomBottom Property, ZoomTop Property, DisplayRight Property

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.

ZoomTop Property

Definition

Specifies the top edge of the region selected for zooming, in original image pixels.

Data Type

Long Integer

Syntax

```
lngPosition = TMSDispl.ZoomTop
```

Design Access

Not Available

Runtime Access

Read/Write

See Also

ZoomLeft Property, ZoomRight Property, ZoomBottom Property, DisplayTop Property

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

Left	Image pixel coordinate of left edge of zoom region
Top	Image pixel coordinate of top edge of zoom region
Right	Image pixel coordinate of right edge of zoom region
Bottom	Image pixel coordinate of bottom edge of zoom region

Data Type

Long Integer

Returns

None

Syntax

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

See Also

ZoomIn Method, ZoomInOutChange Property, ZoomPercent Property

Comments

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.

Appendix A: Error Reporting

Error Reporting and Trapping

Errors that occur in the TMSSequoia Display 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 TMSSequoia 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 TMSSequoia Display 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 -- TMSSequoia Display Control' on page 154.

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 TMSDispl_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 TMSDispl_Error(..., CancelDisplay As Integer)  
    CancelDisplay = ibErrContinue  
    Select Case Number  
        ...perform error detection and recovery  
    End Select  
End Sub
```

Error Numbers

The following section titled "Error Numbers -- TMSSequoia Display Control" lists all of the error codes can be returned in the *Number* parameter to the error event. The "Error Message" that is shown with each error is the string that is reported in the *Description* parameter.

If error number 385, 386 or 391 occurs, that indicates that an error has occurred in the underlying technology libraries and that the error is not mapped to one of the common error numbers. In this case, the *Description* parameter will report the engine's error message and error number together in one string. The engine's error numbers may be found in the section titled "Error Numbers -- TMSSequoia Libraries" on page 158.

Error Numbers -- TMSSequoia Display Control

The following error numbers can be reported in the *Number* parameter of the **Error** event. The "Error Message" that is shown with each error number will be reported in the *Description* parameter to that event. These same data will be shown in the error message box that is displayed if the *CancelDisplay* parameter is left unchanged.

The Visual Basic runtime error that is reported when each of these errors occurs is equal to the "Error Number" plus the Visual Basic constant *vbObjectError*.

Error Number: 375

Error Message:	"Print Error - General Escape Error encountered."
Comments:	A general printing error has caused the print job to abort. Check system configuration and resources. It may be necessary to reboot.

Error Number: 376

Error Message: "Print Error - Not enough disk to spool"
Comments: The hard drive does not have enough free space to spool the print job. The print job may be spooled locally or on a network drive, depending upon system configuration. Free sufficient space on the appropriate drive before printing again.

Error Number: 377

Error Message: "Print Error - Not enough memory to spool"
Comments: The printer driver cannot allocate sufficient memory to print the current job. Free system resources.

Error Number: 378

Error Message: "Print Error - User aborted"
Comments: The operator canceled the print job.

Error Number: 379

Error Message: "Print Error - Unknown escape error."
Comments: An unknown error has caused the print job to abort. Check system configuration and resources. It may be necessary to reboot.

Error Number: 380

Error Message: "Print Error - Escape function not implemented"

Error Number: 381

Error Message: "Print Error - Unable to print page`page_number`"
Comments: The page specified in`page_number`could not be printed. Check for a valid image page and sufficient system resources.

Error Number: 382

Error Message: "Print Error - A print job is already active."
Comments: An attempt was made to start a print job while one is already in process. Finish the current print job (refer to the **EndPrintJob** method) before beginning another.

Error Number: 383

Error Message: "Print Error - Print mode not set. (Software Error. Contact DHS)"

Comments: This error should not occur. If you can reproduce the problem, please forward the steps that produce the error to Diamond Head Software technical support.

Error Number: 384

Error Message: "Print Error - Unknown print mode. (Software Error. Contact DHS)"

Comments: This error should not occur. If you can reproduce the problem, please forward the steps that produce the error to Diamond Head Software technical support.

Error Number: 385

Error Message: "View Director error No:*err_num*, Desc: *err_message*"

Comments: Occurs when an unmapped error is reported from the underlying display libraries. The library's error number (*err_num*) and error message (*err_message*) are reported. The application can retrieve the native error code and message by parsing the main error message. See 'Error Numbers -- TMSSequoia Libraries' on page 158 for a list of the possible values that will be reported.

Error Number: 386

Error Message: "View Director Paint error No:<error_num>; Desc: <descript>"

Comments: Occurs when an unmapped error is reported from the underlying display libraries. The error number that is reported in this error description is read directly from the TMSSequoia libraries. See "Error Numbers -- TMSSequoia Libraries" on page 158 for a list of the possible values that will be reported.

Error Number: 387

Error Message: "Palette Error - Invalid image palette size."

Error Number: 388

Error Message: "Palette Error - Can't allocate local palettes"

Error Number: 389

Error Message: "Palette Error - Error getting system palette"

Error Number: 390

Error Message: "Palette Error - Invalid palette type requested (Software Error. Contact DHS)"

Error Number: 391

Error Message: "View Director Error encountered while retrieving image palette. Error No.: <error_num>; Desc: <str_description>"

Comments: The error number that is reported in this error description is read directly from the TMSSequoia libraries. See "Error Numbers -- TMSSequoia Libraries" on page 158 for a list of the possible values that will be reported.

Error Number: 392

Error Message: "Cannot change/select printer during a virtual printing job."

Comments: An attempt was made to change the printer during a print job or print page. End the current job or page before changing the printer.

Error Number: 393

Error Message: "Unable to create new image. (TMSDispTM::NewImage)"

Comments: Could not allocate sufficient resources for the NewImage method to complete.

Error Numbers -- TMSSequoia Libraries

The following error number can be returned by the TMSSequoia Display control. These error number and their associated strings will be reported in the *Description* parameter to the **Error** event when error number 385, 386 or 391 is reported in the *Number* parameter.

Most of these errors will never be encountered as they are either mapped to one of the other error number listed in the section titled 'Error Numbers -- TMSSequoia Display Control' on page 154 or they are not valid return values from the ImageBASIC implementation of the libraries.

TMSSequoia ViewDirector Error Number	Error Message
10000	Null pointer
10001	Bad callback number
10002	Unimplemented function
10003	Unable to allocate memory
10004	Unable to free memory
10005	Unable to lock memory
10006	Unable to unlock memory
10007	Null parameter pointer
10008	Invalid product number
10009	Unable to open file
10010	Unable to close file

TMSSequoia ViewDirector Error Number	Error Message
10011	File seek error
10012	File read error
10013	Partial file read
10014	Fill error
10015	Cannot create DC
10016	Cannot create bitmap
10017	Cannot select object
10018	Cannot set bitmap bits
10019	Cannot bitblt
10020	Cannot delete object
10021	Cannot finish file write (disk full?)
10022	Cannot write to file
10023	Internal error
11000	Invalid image type
11001	Invalid color
11002	Invalid image size
11003	Invalid image resolution
11004	Invalid window size
11005	Invalid window resolution
11006	Invalid compressed direction
11007	Invalid parameter number
11008	Invalid scale value
11009	Invalid image handle
11010	Invalid method
11011	No file
11012	Invalid horizontal method
11013	Invalid vertical method
11014	Invalid amount
11015	Invalid unit

TMSSequoia ViewDirector Error Number	Error Message
11016	Invalid bit direction
11017	Invalid FILL flag
11019	Invalid type
11020	Invalid scale speed
11021	Invalid window grain
11022	Invalid horizontal gravity
11023	Invalid vertical gravity
11024	No file name
11025	Invalid width
11026	Invalid height
11027	Corrupt header
11028	Image not supported
11029	NULL file name pointer
11030	Invalid length
11031	NULL image pointer
11032	Invalid image number
11033	Wrong header
11034	Invalid rotation value
11035	Invalid horizontal mirroring value
11036	Invalid vertical mirroring value
11037	Decompression error
11038	No title
11039	Must redraw image
11040	Invalid from value
11041	Invalid to value
11042	Invalid scrollbar range
11043	Invalid state bounds flag
11044	Demo limit exceeded
11045	Invalid window bits/pixel

TMSSequoia ViewDirector Error Number	Error Message
11046	Invalid window planes
11047	Invalid anti-alias flag
11048	Window palette table
11049	Image must be tiled
11050	Invalid tile index
11051	Invalid tile width
11052	Invalid tile height
11053	Decompression error - invalid symbol
11054	Unknown header
11055	Invalid use parameter
11056	Invalid page number
11057	Invalid group number
11058	Image was corrupted by application
11059	Invalid debug value
11060	Scale device
11061	Internal error - shared)
11062	Image palette
11063	Image bits per pixel
11064	Image planes
11065	Rectangle level
11066	Rectangle item
11067	Rectangle hidden flag
11068	Rectangle not found
11069	Rectangle exists
11070	Rectangle empty
11071	Rectangle placement
11072	No rectangle callback
11073	Invalid image file state
11074	Unsupported PDA image

TMSSequoia ViewDirector Error Number	Error Message
11075	Invalid window bit direction
11076	No current selection
11077	Window width not set
11078	Window height not set
11079	Window X resolution not set
11080	Window Y resolution not set
11081	Image is too wide
11082	DRAW_RUNS callback not set
11083	Invalid draw runs value
11084	Invalid window color format
11085	Invalid VU_PARM_IMAGE_USECACHE value
11086	Requested line out of sync
11087	Compression mode not supported
11088	Unsupported or bad JPEG file type
11089	Unrecognized JPEG marker
11090	Invalid palette type
11091	Invalid palette color format
11092	No window palette defined for this image
11093	Image palette must have 2 or more entries
11094	Both standard and extended callback set
11095	Invalid parameter value
11096	Tag not found
11097	Template file not specified
11098	Vector image support not enabled
11099	No overlays associated with image
11100	An overlay with this ID already exists
11101	Invalid vector view

Appendix B: Color Definition

File and Compression Formats

The following short primer on image data management and compression may be of some service when you select the formats which your application will support.

When an 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, ImageBASIC uses TIFF Group 4 compression in files and an RLE variant in memory. Other options are available, however, and an overview of the various compression methods is given in the next section.

Each of the many file formats and compression types that are supported by ImageBASIC were designed to balance the compression/decompression time with an acceptable file size reduction. Some file/compression types are designed only for bitonal or only for paletted color images, while others can store almost any color format. As may be expected, those file/compression formats that are optimized for a particular color level are typically faster and/or compress better than the more generic options. The following sections describe the various compress algorithms and file formats available in ImageBASIC.

By taking into account the purpose for which each file/compression type was designed and the types of images which your application will most often support, one or more file types may be found to optimize the display and storage performance of the application. If you would like to investigate this topic further, a thorough treatment of the file formats including their structure, history, and applicability can be found in the *Encyclopedia of Graphics File Formats* by James D. Murray and William van Ryper, published by O'Reilly & Associates, Inc.

File Format Definitions

The following sections briefly describe a number of different file formats that are supported by one of the ImageBASIC File controls. By their definitions, some of these file formats use a particular data compression. Some of the formats, however, can select from a variety of different compression algorithms. These compression algorithms are described in the section *Compression Types* on page 165.

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. Type I CALS files, the only type supported by ImageBASIC, use Group 4 compression. The image data is appended to a header block, very much like TIFF files. Although common in U.S. government document imaging applications, CALS is uncommon outside the sphere of direct federal influence.

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 64,000 pixels in either dimension using from 1 to 8 bits of color (bitonal to 256 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 and JFIF 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.

Strictly speaking, JPEG is compression specification, not a file format, but the JPEG File Interchange Format (JFIF) is the most commonly used file format employing JPEG compression, and the terms JPEG and JFIF are largely interchangeable.

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.

CCITT 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. This compression type supports only bitonal data.

CCITT 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. This compression type supports only bitonal data.

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) Compression

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 some BMP files. This compression type is most effective on bitonal images but can be applied to any color depth.

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.

Bits Per Sample (BPS) indicates how many bits define the different values for each sample (see `SamplesPerPixel`) 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. `BitsPerSample` and `PhotometricInterpretation` 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 `PhotometricInterpretation` 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 <code>SamplesPerPixel</code> |
| 3 | Paletted | An array is constructed with a color assigned to each value in the array. This value is generally only applied to color images and is limited to fewer values than RGB. |

Index

A

- AbortPrintJob Method 54
- AboutBox Method 53
- Accelerated Printing
 - (UsePrintAcceleration) 139
- AcceptDragFiles Property 55
- Active Property 56
- Annotation Control 3
- Annotation Merging, MergeNotes 98
- Assigning Mouse Button Functions 24
- AutoRefresh Property 57

B

- Blank Image Creation, NewImage
 - Method 105
- Button Parameter 30

C

- Click Event 21, 28, 58, 73
- Coloring the Working Region 40, 75
- ColorPaletteType Property 12, 59
- ColorReductionType Property 60
- ColorScalingMethod Property 13, 61
- Connecting Controls
 - Link Property 97
- Constants
 - dspEventClick 21, 73
 - dspEventDbtClick 21, 73
 - dspEventError 21, 73
 - dspEventFilesDropped 21, 73
 - dspEventImageDataChanged 21, 73
 - dspEventMouseDown 21, 73
 - dspEventMouseMove 21, 73
 - dspEventMouseUp 21, 73
 - dspEventPaint 21, 73
- Control Communication 3
- Control Handle
 - hWnd Property 82
- Creating a Blank Image
 - NewImage Method 105
- Creating a DIB 15
- Cursor Icon, MousePointer Property 102

D

- Data Input
 - ImageDataSource Property 5, 86
 - SetPicture Property 135
- Database, Creation of Licensing 2
- Database, Licensing 1
- DbtClick Event 21, 28, 62, 73
- Descriptive Properties
 - ImageBitsPerSample 5, 83
 - ImageHeight 5, 87
 - ImageSamplesPerPixel 6, 88
 - ImageWidth 6, 90
 - ImageXRes 6, 91
 - ImageYRes 6, 92
- DIB, Making a
 - GetScaledPicture Method 81
- Dimensions of Thumbnail
 - ThumbnailByteSize Property 138
- Disable Events 20
- Disabling Events, EventMask 72
- Disabling GUI Events 20
- Display Color Reduction 60
- Display Configuration
 - ColorPaletteType Property 12, 59
 - ColorScalingMethod Property 13, 61
 - ImageType Property 11, 89
 - Invert Property 19, 93
 - Rotation 128
 - Rotation Property 8
 - ScaleQuality Property 12, 129
 - ScaleToGray Property 18, 130
 - ScaleToGrayLevel Property 19, 131
 - ZoomPercent Property 146
- Display Performance 20
- Display Properties
 - Invert 19
 - ZoomBottom 16, 34
 - ZoomLeft 16, 34
 - ZoomRight 16, 34
 - ZoomTop 16, 34
- Display Scaling
 - ZoomRatio Property 147
- DisplayBottom Property 63
- Displaying as Thumbnail 14
- DisplayLeft Property 64
- DisplayMode Property 14, 65
- DisplayRight Property 66
- DisplayTop Property 67
- DoClick Method 68

- Drag and Drop Files
 - AcceptDragFiles Property 55
 - FilesDropped Event 74
- DragDrop Event 28
- DragOver Events 28
- dspEventClick Constant 21, 73
- dspEventDblClick Constant 21, 73
- dspEventError Constant 21, 73
- dspEventFilesDropped Constant 21, 73
- dspEventImageDataChanged Constant 21, 73
- dspEventMouseDown Constant 21, 73
- dspEventMouseMove Constant 21, 73
- dspEventMouseUp Constant 21, 73
- dspEventPaint Constant 21, 73

E

- Enable Events 20
- Enabling Events, EventMask 72
- EndPrintJob Method 69
- EndPrintPage Method 50, 70
- Error Dialog
 - CancelDisplay Parameter 153
- Error Event 21, 71, 73, 152
- Error Event Parameters
 - CancelDisplay 153
 - Description 152
 - HelpContext 153
 - HelpFile 153
 - Number 152
 - SCode 152
- Event Parameters
 - Button 30
- Error Event
 - CancelDisplay Parameter 153
 - Description Parameter 152
 - HelpContext Parameter 153
 - HelpFile Parameter 153
 - Number Parameter 152
 - SCode Parameter 152
- ImageX 30
- ImageY 30
- ScreenX 30
- ScreenY 30
- Shift 30
- Source 30
- EventMask Property 72
- Events

- Click 21, 28, 58, 73
- DblClick 21, 28, 62, 73
- DragDrop 28
- DragOver 28
- Error 21, 71, 73, 152
- FilesDropped 21, 73, 74
- ImageDataChanged 5, 21, 73, 85
- MouseDown 21, 28, 73, 99
- MouseMove 21, 28, 73, 101
- MouseUp 21, 28, 73, 104
- Paint 21, 73, 106

- Exporting Image
 - GetScaledPicture Method 81
- Exporting Image Data 15

F

- FilesDropped Event 21, 73, 74
- FillWorkingRegion Method 40, 75
- FireGUIEvents Property 76
- FitToHeight Method 33, 35, 77
- FitToWidth Method 33, 35, 78
- FitToWindow Method 33, 35, 79

G

- GetRegBlackPercent Method 41, 80
- GetScaledPicture Method 15, 81
- Grayscale
 - ScaleToGray Property 18, 130
 - ScaleToGrayLevel Property 19, 131
- Grouping pages for printing
 - StartPrintJob Method 48

H

- Handle
 - hWnd Property 82
- Highlighting Region
 - WorkingRegionStyle Property 140
- hWnd Property 82

I

- Icon, Mouse
 - MouseIcon Property 100
- Image Characteristics
 - ImageBitsPerSample Property 5, 83
 - ImageHeight Property 5, 87

- ImageSamplesPerPixel Property 6, 88
- ImageWidth Property 6, 90
- ImageXRes Property 6, 91
- ImageYRes Property 6, 92
- ImageBitsPerSample Property 5, 83
- ImageDataChanged Event 5, 21, 73, 85
- ImageDataSource 3, 4
- ImageDataSource Property 5, 86
- ImageHeight Property 5, 87
- ImageSamplesPerPixel Property 6, 88
- ImageType Property 11, 89
- ImageWidth Property 6, 90
- ImageX Parameter 30
- ImageXRes Property 6, 91
- ImageY Parameter 30
- ImageYRes Property 6, 92
- Invert Property 19, 93

L

- LCM, Using 2
- LeftButtonOption Property 25, 33, 37, 39, 94
- LeftMouseBoxStyle Property 25, 96
- License Storage 1
- License Verification 2
- Licensing
 - Active Property 56
- Link ID 4
- Link property 4, 97
- Linking Controls 3, 4
 - ImageDataChanged Event 5, 85
 - ImageDataSource Property 5, 86
- Load Time Improvement
 - Active Property 56

M

- Manual Refresh
 - AutoRefresh Property 57
- Margins for Printing 43
- MergeNotes Method 98
- Methods
 - AbortPrintJob 54
 - AboutBox 53
 - DoClick 68
 - EndPrintJob 69
 - EndPrintPage 50, 70

- FillWorkingRegion 40, 75
- FitToHeight 77
- FitToWidth 78
- GetRegBlackPercent 41, 80
- GetScaledPicture 15, 81
- MergeNotes 98
- NewImage 105
- PrintImage 46, 50, 111
- PrintRegion 47, 50, 114
- Refresh 120
- ScrollBy 133
- SelectPrinter 44, 134
- StartPrintJob 48, 136
- StartPrintPage 50, 51, 137
- ZoomOut 145
- ZoomToRegion 150
- Mouse Button Options
 - Normal 25
 - Proportional Rubber Band 26
 - Proportional Zoom 26, 27
 - Rubber Band 26
 - Zoom 26
- Mouse Configuration
 - LeftButtonOption 25, 94
- Mouse Events
 - Click 28, 58
 - DbClick 28, 62
 - DragDrop 28
 - DragOver 28
 - MouseMove 28, 99, 101
 - MouseUp 28, 104
- Mouse Functionality 23
- Mouse Related Properties
 - LeftButtonOption 33, 37, 39
 - LeftMouseBoxStyle 25, 96
 - RightButtonOption 25, 33, 37, 39, 125
 - RightMouseBoxStyle 25, 127
- MouseDown Event 21, 28, 73, 99
- MouseIcon Property 100
- MouseMove Event 21, 28, 73, 101
- MouseMove Property, Enabling 76
- MousePointer Property 102
- MouseUp Event 21, 28, 73, 104
- Multi-page print jobs 48

N

- NewImage Method 105

Normal, Mouse Button Option 25

O

OLE Drag and Drop

AcceptDragFiles Property 55

FilesDropped Event 74

P

Paint Event 21, 73, 106

Paint Property, Enabling 76

Painting the Display Window

Paint Event 106

Refresh Method 120

Pan Window 36, 42

Pan Window Size

ThumbnailByteSize Property 138

Percentage of Black Pixels 41, 80

Performance Enhancement

AutoRefresh Property 57

FireGUIEvents Property 76

Performance Improvement 20

Pixel Density

GetRegBlackPercent Method 41, 80

Position of Image

DisplayBottom Property 63

DisplayLeft Property 64

DisplayRight Property 66

DisplayTop Property 67

Print Jobs 48

AbortPrintJob Method 54

Printer Configuration

PrinterDC Property 107

PrinterDevice Property 46, 108

PrinterDriver Property 46, 109

PrinterPort Property 45, 110

SelectPrinter Method 44, 134

Printer Setup 45

PrinterDC Property 107

PrinterDevice Property 46, 108

PrinterDriver Property 46, 109

PrinterPort Property 45, 110

PrintImage Method 46, 50, 111

Printing

Multiple Images on One Page 50,
51, 137

Multiple Page Print Jobs 136

Print Jobs 136

PrintImage Method 46, 50, 111

PrintRegion Method 47, 50, 114

StartPrintJob Method 48

Printing Properties

PrintPageHeight 112

PrintPageWidth 113

PrintTargetBottom 115

PrintTargetLeft 116

PrintTargetRight 117

PrintTargetTop 118

Printing, Accelerated

(UsePrintAcceleration) 139

Printing, Margins 43

Printing, Overview 43

Printing, Scaling 43

PrintPageHeight Property 51, 112

PrintPageWidth Property 51, 113

PrintRegion Method 47, 50, 114

PrintTargetBottom Property 51, 115

PrintTargetLeft Property 51, 116

PrintTargetRight Property 51, 117

PrintTargetTop Property 51, 118

Properties

AcceptDragFiles 55

Active 56

AutoRefresh 57

ColorPaletteType 12, 59

ColorReductionType Property 60

ColorScalingMethod 13, 61

DisplayBottom 63

DisplayLeft 64

DisplayMode 14, 65

DisplayRight 66

DisplayTop 67

EventMask 72

FireGUIEvents 76

ImageBitsPerSample 5, 83

ImageDataSource 5, 86

ImageHeight 5, 87

ImageSamplesPerPixel 6, 88

ImageType 11, 89

ImageWidth 6, 90

ImageXRes 6, 91

ImageYRes 6, 92

Invert 19, 93

LeftMouseBoxStyle 25, 96

Link 97

MouseIcon 100

MousePointer 102

PrinterDC 107

- PrinterDevice 46, 108
- PrinterDriver 46, 109
- PrinterPort 45, 110
- PrintPageHeight 51
- PrintPageWidth 51
- PrintTargetBottom 51
- PrintTargetLeft 51
- PrintTargetRight 51
- PrintTargetTop 51
- RBandAspect 119
- RegBottom 39, 121
- RegLeft 39, 122
- RegRight 39, 123
- RegTop 39, 124
- RightMouseBoxStyle 25, 127
- Rotation 8
- ScaleQuality 12, 129
- ScaleToGray 18, 130
- ScaleToGrayLevel 19, 131
- ScrollBars 132
- SetPicture 135
- ThumbnailByteSize 138
- UsePrintAcceleration 139
- WorkingRegionStyle 140
- ZoomBottom 141
- ZoomInOutChange 143
- ZoomPercent 146
- ZoomRatio 147
- ZoomRight 148
- ZoomTop 149
- Proportional Rubber Band, Mouse
 - Button Option 26
- Proportional Rubber Banding 24
- Proportional Zoom, Mouse Button
 - Option 26, 27

R

- RBandAspect Property 119
- Redaction
 - FillWorkingRegion Method 40, 75
- Reducing Colors for Display 60
- Refresh Method 120
- Refresh, Automatic
 - AutoRefresh Property 57
- Refreshing Display, AutoRefresh 57
- RegBottom Property 39, 121
- Region Highlight
 - WorkingRegionStyle Property 140

- RegLeft Property 39, 122
- RegRight Property 39, 123
- RegTop Property 39, 124
- Releasing License Tokens 2
- RightButtonOption Property 25, 33, 37, 39, 125
- RightMouseBoxStyle Property 25, 127
- Rotating Images
 - Rotation Property 128
- Rotation Property 8, 128
- Routing Data between Controls 3
- Rubber Band, Mouse Button Option 26
- Rubber Banding 23
- Runtime Linking 4

S

- ScaleQuality Property 12, 129
- ScaleToGray Property 18, 130
- ScaleToGrayLevel Property 19, 131
- ScreenX Parameter 30
- ScreenY Parameter 30
- ScrollBars Property 132
- ScrollBy Method 133
- Scrolling
 - Pan Window 36, 42
 - ScrollBy Method 133
- SelectPrinter Method 44, 134
- SetPicture Property 135
- Setup of Printer 45
- Shift Parameter 30
- Source of Image Data 3
- Source Parameter 30
- Speed of Display
 - ScaleQuality Property 12, 129
- Standard Rubber Banding 24
- StartPrintJob Method 48, 136
- StartPrintPage Method 50, 51, 137

T

- Thumbnail Scaling 14
- ThumbnailByteSize Property 14, 138
- Thumbnails 14
 - DisplayMode Property 65
- Timing of License Locking/Unlocking 2
- Timing of Licensing Verification
 - Active Property 56
- TMS Display Control 3

TMS File Control 3

U

UsePrintAcceleration Property 139

Using License Tokens 2

V

Virtual Print Page

EndPrintPage 70

Virtual Print Pages

PrintPageHeight Property 51

PrintPageWidth Property 51

PrintTargetBottom Property 51

PrintTargetLeft Property 51

PrintTargetRight Property 51

PrintTargetTop Property 51

StartPrintPage Method 50, 51, 137

W

Working Region Properties

RegBottom 39, 121

RegLeft 39, 122

RegRight 39, 123

RegTop 39, 124

Working Region, definition 37

WorkingRegionStyle Property 140

Z

Zoom Coordinates

DisplayBottom Property 63

DisplayLeft Property 64

DisplayRight Property 66

DisplayTop Property 67

Zoom Region Properties

ZoomBottom 16, 34, 141

ZoomLeft 16, 34, 144

ZoomRight 16, 34, 148

ZoomTop 16, 34, 149

Zoom, Mouse Button Option 26

ZoomBottom Property 16, 34, 141

ZoomIn Method 33, 35, 142

Zooming

ZoomInOutChange Property 143

Zooming Configuration

FitToHeight Method 33, 35, 77

FitToWidth Method 33, 35, 78

FitToWindow Method 33, 35, 79

ZoomIn Method 33, 35, 142

ZoomOut Method 33, 35, 145

ZoomToRegion Method 150

ZoomInOutChange Property 143

ZoomLeft Property 16, 34, 144

ZoomOut Method 33, 35, 145

ZoomPercent Property 146

ZoomRatio Property 147

ZoomRight Property 16, 34, 148

ZoomTop Property 16, 34, 149

ZoomToRegion Method 150