HALO Visual Image

Programmer Guide & Reference

Table of Contents

HALO Image Control (HImage)

HALO Gallery Control (HGallery)

HALO Acquire Control (HAcquire)

HALO Image Control

Description

The HALO Image Control (HImage) allows you to include images on your form. It is similar to Visual Basic's "PictureBox" control, but is much more powerful. In addition to all of the raster-image capabilities of the "PictureBox" control, the HALO Image Control provides you with:

- Class Conversion
- Error Trapping
- Support for all popular File Format and Compression Methods
- Support for Multiple Image Files
- Enhancement Filters
- Support for image information (e.g., Date, Artist, Title)
- Image Printing
- Image Rotation and Orientation
- Partial Image Selection
- Palette Optimization
- Zoom (In and Out)

Furthermore, using the HALO "FileCommand" property you can direct the control to automatically load a selected image file.

File Name

The HALO Image Control is defined by file HIC.VBX. This file is located in your Windows SYSTEM directory.

Distribution Note

When you distribute applications that use the HImage control you must include the HIC.VBX file with it. This file should be installed in your customer's Microsoft Windows SYSTEM subdirectory (the Visual Basic Setup Kit, included with the Professional Visual Basic product, provides tools to help you write installation programs for your application). You must also distribute all of the HALO Visual Image DLLs, which are located in your Windows SYSTEM subdirectory.

See Also

To learn about the specific programming attributes associated with the HALO Image Control, see the following topics:

Properties

Events

Methods

Functions

Examples

Error Codes

File Formats

Properties, HImage

The following table lists the properties associated with the HALO Image Control.

Note - properties that are not highlighted are ones that conform to standard Visual Basic behavior. For information about these properties, refer to your Visual Basic Language Reference or your Visual Basic on-line documentation.

About	FilePage	LinkMode	RotateAngle
AutoRedraw	FilePages	LinkTimeout	RotateCommand
AutoSize	FilePath	LinkTopic	SelCommand
BackColor	FilterCommand	LutBrightness	SelhDIB
BorderStyle	FilterSize	LutContrast	SelHeight
ConvertArg1	FilterStrength	LutGamma	SelImage
ConvertArg2	hDIB	MergeMode	SelLeft
ConvertCommand	Height	MousePointer	SelPicture
DataChanged	HelpContextID	Name	SelTop
DataField	hWnd	Parent	SelWidth
DataFormat	Image	Picture	SmallChange
DataSource	ImageArtist	PrintCommand	TabIndex
DragIcon	ImageClass	PrintDitherOption	TabStop
DragMode	ImageDate	PrintDitherType	Tag
DrawMode	ImageDescription	PrinthDC	Top
Enabled	ImageHeight	PrintHeight	Version
ErrorCode	ImageResX	PrintPosX	ViewFixedPalette
ErrorShow	ImageResY	PrintPosY	ViewPosX
ErrorString	ImageTitle	PrintSmooth	ViewPosY
FileCommand	ImageWidth	PrintUnit	ViewScrollBars
FileCompression	Index	PrintWidth	ViewZoom
FileFormat	LargeChange	ResizeCommand	Visible
FileJPEGQuality	Left	ResizeHeight	Width
FileName	LinkItem	ResizeWidth	

Events, HImage

The following table lists the events associated with the HALO Image Control.

Note - events that are not highlighted are ones that conform to standard Visual Basic behavior. For information about these events, refer to your Visual Basic Language Reference or your Visual Basic on-line documentation.

Click	GotFocus	LinkClose	MouseDown
DblClick	KeyDown	LinkError	MouseMove
DragDrop	KeyPress	LinkNotify	MouseUp
DragOver	KeyUp	LostFocus	Select

Methods, HImage

The following table lists the methods supported by the HALO Image Control.

Note - methods that are not highlighted are ones that conform to standard Visual Basic behavior. For information about these methods, refer to your Visual Basic Language Reference or your Visual Basic on-line documentation.

Clear	LinkExecute	LinkSend	ZOrder
Drag	LinkPoke	Refresh	
Move	LinkRequest	SetFocus	

Functions, HImage

The following table describes the functions associated with the HALO Image Control. To learn more about a specific function, just click its name below.

HicGetGammaIndex

HicGetGammaValue

HicTwipsToImageCoord

HicImageCoordToTwips

Examples, HImage

Click a topic to view example code for the HImage control:

- **File Input/Output**
- **Database**
- **Printing**
- **Copy & Paste**
- **Image Processing**
- **Control Size**

Examples

In this document, the "Example" symbol indicates that sample code relating to the current topic is available. Click this symbol to access the example code.

File Input/Output Examples, HImage Control

The following example opens two images, one after another, using "auto-open" mode.

```
HImage1.FileCommand = 1                    ' Enable auto open
HImage1.FileName = "e:\images\box.tif"     ' Load TIFF image
HImage1.FileName = "c:\vgalogo.bmp"        ' Load BMP image
HImage1.FileCommand = 0                    ' Disable auto open
```

The following example explicitly opens a file in .BMP format.

```
HImage1.FileName = "c:\vgalogo.bmp"        ' Set file name
HImage1.FileFormat = 2                     ' BMP file format
HImage1.FileCommand = 2                    ' Load file
```

The following example saves an image in JPEG format using a quality factor of 40%.

```
HImage1.FileName = "e:\images\box.jpg"     ' Set file name
HImage1.FileJPEGQuality = 40               ' JPEG quality factor
HImage1.FileFormat = 2                     ' JPEG file format
HImage1.FileCommand = 3                    ' Save as JPEG file
```

The following example appends an image to a TIFF file.

```
HImage1.FileFormat = 1                     ' TIFF file format
HImage1.FileName = "e:\images\album.tif"   ' Set file name
HImage1.FileCommand = 4                    ' Append to file
```

The following example opens the third image in a multi-image file.

```
HImage1.FileCommand = 1                    ' Enable auto open
HImage1.FileName = "i:\dcx\fax.dcx"        ' Load DCX file
HImage1.FilePage = 2                       ' Open the third image
```

Data Control Example, HImage Control

The example below shows how an HImage control would be bound to the Data control of a referenced-image database.

```
HImage1.DataSource = Data1          ' Data control name
HImage1.DataField = "Pic"           ' Image field in database
HImage1.DataFormat = 0              ' Field contains a file name
```

The example below shows how an HImage control would be bound to the Data control of an embedded-image database, in which the images are stored in DIB format.

```
HImage1.DataSource = Data1          ' Data control name
HImage1.DataField = "Pic"           ' Image field in database
HImage1.DataFormat = 1              ' Field contains a DIB blob
```

Print Examples, HImage Control

There are three ways of printing a HALO Image Control:

Visual Basic PrintForm

This standard Visual Basic method sends the content of the entire form to the printer.

```
Sub mnuPrintForm_Click ()
  PrintForm                         ' Print entire form
End Sub
```

HImage PrintCommand Property

Using the HImage **PrintCommand** property you can print the image to the default printer (just the image, not the rest of the form). When **PrintCommand** is set to 1, the image is printed using the current print option settings (e.g., **PrintWidth**, **PrintDitherType**, **PrintPosX**).

```
Sub mnuPrintImage_Click ()
  HImage1.PrintUnit = 0             ' Units are Inches
  HImage1.PrintWidth = 2.25         ' Set image width
  HImage1.PrintHeight = 3.5         ' Set image height
  HImage1.PrintPosX = 1             ' Set left margin
  HImage1.PrintPosY = 1             ' Set top margin
  HImage1.PrintDitherType = 2       ' Use angle dither
  HImage1.PrintDitherOption = 0     ' Use finest screen
  HImage1.PrintSmooth = True        ' Smooth enlargement
  HImage1.PrintCommand = 1          ' Print image w/current options
End Sub
```

When **PrintCommand** is set to 2, the "Print" dialog box is issued at run time, so that print-size, position, dither and smoothing options can be obtained from the user. Once the user has specified these values, your program can set **PrintCommand** to 1 to print the image. The following pair of statements show you how this would be coded:

```
HImage1.PrintCommand = 2            ' Obtain options from user
HImage1.PrintCommand = 1            ' Print the image
```

HImage Printer.hDC Property

With the HImage **PrinthDC** property you can print the image to the printer object. This allows you to add the image to a page composed in memory, and lets you combine the image with other data elements (e.g., text, graphics, data) before sending it to the printer.

```
Printer.CurrentX = 1440          ' Set horz position
Printer.CurrentY = 1400          ' Set vert position
Printer.Print "Front View"       ' Put string on printer object

HImage1.PrintUnit = 0            ' Units are Inches
HImage1.PrintWidth = 3.0         ' Set image width
HImage1.PrintHeight = 2.5        ' Set image height
HImage1.PrintPosX = 2            ' Set left margin
HImage1.PrintPosY = 2            ' Set top margin
HImage1.PrintDitherType = 2      ' Use angle dither
HImage1.PrintDitherOption = 0    ' Use finest screen
HImage1.PrintSmooth = True       ' Smooth enlargement
HImage1.PrinthDC = Printer.hDC   ' Put image on printer object

Printer.EndDoc                   ' Print page (text & image)
```

Note that all of the HImage print options are set prior to setting the **PrinthDC** property, since the image data sent to the printer object is formatted and positioned according to the HImage print properties, not the printer object properties (e.g., not **CurrentX** and

CurrentY).

Copy & Paste Examples, HImage Control

The following example copies HImage2 to HImage1.

```
HImage1.Image = HImage2.Image        ' Copy image to HImage1
```

The following example defines a selection box, then copies it to another HImage control.

```
HImage2.SelLeft = 100                ' Left edge of box
HImage2.SelTop = 200                 ' Top edge of box
HImage2.SelWidth  = 50               ' Box width
HImage2.SelHeight = 50               ' Box height
HImage1.Image = HImage2.SelImage     ' Copy box to HImage1
```

The following example defines two selection boxes, one in each image control (HImage1 and HImage2), then copies the box from HImage2 to the box in HImage 1.

```
HImage2.SelLeft = 100                ' Left edge of box
HImage2.SelTop = 200                 ' Top edge of box
HImage2.SelWidth  = 50               ' Box width
HImage2.SelHeight = 50               ' Box height

HImage2.SelLeft = 100                ' Left edge of box
HImage2.SelTop = 200                 ' Top edge of box
HImage2.SelWidth  = 50               ' Box width
HImage2.SelHeight = 50               ' Box height

HImage1.SelImage = HImage2.SelImage  ' Copy box to HImage box
```

The following example defines a selection box, then copies it to its own image control. This causes the entire image to be replaced by just the selection, and is quick and easy way to crop an image.

```
HImage2.SelLeft = 100                ' Left edge of box
HImage2.SelTop = 200                 ' Top edge of box
HImage2.SelWidth  = 50               ' Box width
HImage2.SelHeight = 50               ' Box height
HImage2.Image = HImage2.SelImage     ' Crop image to 50x50
```

The following example loads an image into the control using the Visual Basic **LoadPicture** function.

```
HImage1.Picture = LoadPicture("xx.bmp")     ' Load image w/LoadPicture
```

The following example copies an image from a Visual Basic Picture Box control to an HImage control.

```
HImage1.Picture = Picture1.Picture   ' Copy Picture to HImage1
```

The following example copies a selection box to the Clipboard.

```
Clipboard.Clear                        ' Clear the Clipboard
Clipboard.SetData HImage1.SelPicture   ' Put selection box on Clipboard
```

The following example copies the Clipboard to an HImage control.

```
HImage1.Picture = Clipboard.GetData()  ' Copy Clipboard to image
```

Image Processing Example, HImage Control

The following example illustrates how easy it is to apply a filter to an image using the "FilterCommand" property. Before setting the "FilterCommand" property (which invokes the HALO filtering process) the "FilterStrength" property has been set to specify the level at which the filtered effect will be applied (a high strength value produces a more pronounced effect).

```
HImage1.FilterStrength = 100          ' Set 100% strength
HImage1.FilterCommand = 20            ' Sharpen image
```

Control Size Examples, HImage Control

The following example sets the image and control to a static size, enables scroll bars, and positions the image such that pixel 50,75 is in the upper-left corner of the control.

```
HImage1.AutoSize = 0                ' Set static size
HImage1.ViewScrollBars = True       ' Enable scroll bars
HImage1.ViewPosX = 50               ' Move to column 50
HImage1.ViewPosY = 75               ' Move to row 75
```

The following example zooms in and out on an image -- when the left mouse button is pressed, it zooms-in, and when the right mouse button is pressed, it zooms-out.

```
Sub HImage1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As
Single)
   If (Button And 2) Then
      HImage1.ViewZoom = HImage1.ViewZoom / 2
   else
      HImage1.ViewZoom = HImage1.ViewZoom * 2
   End If
End Sub
```

(About) Property, HImage Control

Description

The **About** property is used to display information about the HALO Image Control, including its version number, release date and so forth. To display this information, select the **About** property, then click the "..." button at the top of the "Properties" window. The HImage "About" box will be displayed. Click "OK" when you are finished viewing its contents.

This property is available at design time, only.

Note - use the **About** property to display HImage version information at design time. Use the **Version** property to display version information at run time.

AutoRedraw Property, HImage Control

Description

This property determines whether the image in the HImage control is automatically redrawn when it is modified.

When **AutoRedraw** is on, the control is automatically updated anytime the image data within it is changed (e.g., filtered, rotated, resized).

When **AutoRedraw** is off, the display is not updated until a Refresh is performed. You might suppress drawing if your program performs a sequence of operations (e.g., open, resize and rotate), and you do not want the results of the intermediate steps to appear on the screen.

Data Type

Integer (Boolean)

Data Value

The **AutoRedraw** settings are:

Setting	Description
True (-1)	(Default). Redraws the control when any visual aspect of the image is changed.
False (0)	Redraws the control only when the Refresh method is performed.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**AutoRedraw**[= {**True** | **False**}]

AutoSize Property, HImage Control

Description

This property determines whether the HImage control is automatically resized to fit the image, whether the image is resized to fit within the control or whether the control and image are static.

Examples

Data Type

Integer (enumerated)

Data Value

The **AutoSize** property settings are:

Setting	Description
0	**None.** The control and image size are static; no automatic sizing of the control or the image occurs. An image opened into the control is displayed at actual size (i.e., the **ViewZoom** property is automatically initialized to 100%). If the image is oversized, only a portion of the image will be visible in the control (you can enable scroll bars to pan such an image -- see **ViewScrollBars**).
1	(Default) **Fit picture to control.** Resizes the image to fit within the control by adjusting the **ViewZoom** property, such that the entire image fits within the control frame. This setting does not affect the internal size of the image.
2	**Fit control to picture.** Opens the image at actual size (i.e., the **ViewZoom** property is automatically initialized to 100%) and resizes the control to fit around the entire image.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**AutoSize**[= *setting%*]

ConvertArg1, ConvertArg2, ConvertCommand Properties, HImage Control

Description

These properties are used to convert an image to another image class. You might use it to convert a 24-bit, True Color image to an 8-bit Palette image. Or to convert a Palette image to a Gray Scale image. When an image is converted to Bilevel (Black & White), you can select the halftoning method that is to be used.

For some conversion types, the **ConvertArg1** and **ConvertArg2** properties are used to specify special conversion options. In instances where they are required, you must set these values before setting **ConvertCommand** (see "Data Value" below for allowed values). For example, you would use the following sequence to convert an image to Bilevel using the fine, vertical-line halftone screen.

```
HImage1.ConvertArg1  =  4           ' Set vertical line halftone
HImage1.ConvertArg2  =  0           ' Set finest screen size
HImage1.ConvertCommand  =  2        ' Convert image to Bilevel
```

The first two statements select the halftone screen type and size, and the last statement performs the actual conversion.

Note - an image's class designation is contained in its **ImageClass** property.

Data Type

Integer

Data Value

The allowed values for the **ConvertCommand**, **ConvertArg1** and **ConvertArg2** properties are as follows:

Command	Arg1	Arg2	Description
0	Not used	Not used	**Nothing.** Do nothing
1	Not used	Not used	**Bilevel Halftone.** Converts the image to Bilevel using the threshold method. 128 is the threshold value.
2	**0**	**0-3**	**Bilevel Halftone.** Converts the image to Bilevel using the Angle-Dot halftone screen at the coarseness level specified in **ConvertArg2**.
	1	**0-3**	**Bilevel Halftone.** Converts the image to Bilevel using the Flat-Dot halftone screen at the coarseness level specified in **ConvertArg2**.
	2	**0-3**	**Bilevel Halftone.** Converts the image to Bilevel using the Angle-Line halftone screen at the coarseness level specified in **ConvertArg2**.
	3	**0-3**	**Bilevel Halftone.** Converts the image to Bilevel using the Horizontal-Line halftone screen at the coarseness level specified in **ConvertArg2**.
	4	**0-3**	**Bilevel Halftone.** Converts the image to Bilevel using the Vertical-Line halftone screen at the coarseness level specified in **ConvertArg2**.
	5	**0**	**Bilevel Halftone.** Converts the image to Bilevel using the Floyd Steinberg (4 Weights) error-diffusion halftone method.
		1	**Bilevel Halftone.** Converts image to Bilevel using the

			Stucki (12 Weights) error-diffusion halftone method.
		2	**Bilevel Halftone.** Converts image to Bilevel using the Floyd Steinberg (Fuzzy) error-diffusion halftone method.
		3	**Bilevel Halftone.** Converts image to Bilevel using the Random White Noise error-diffusion halftone method.
	6	Not used	**Bilevel Halftone.** Converts the image to Bilevel using the threshold method. 128 is the threshold value.
3	Not used	Not used	**Gray Scale.** Converts the image to a Gray Scale image.
4	Not used	Not used	**Palette MColor.** Converts the image to a Palette-class image using the MColor method.
5	Colors	Not used	**Palette Color Reduction.** Converts image to a Palette-class image using the Color Reduction method. **ConvertArg1** determines the number of color indices in the new image's palette.
6	Not used	Not used	**True Color.** Converts the image to a True Color image.
7	Not used	Not used	**Palette 16 Colors.** Converts the image to 16-color, Palette-class image using the VGA palette.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ConvertArg1**[= *setting%*]
[*form*.]*HImage*.**ConvertArg2**[= *setting%*]
[*form*.]*HImage*.**ConvertCommand**[= *setting%*]

DataFormat Property, HImage Control

Description

This property is used to specify the content of the binding field when HImage is bound to a Data control. In the example below, the binding field contains an image file name.

```
HImage1.DataSource = "Data1"      ' Control is bound to Data1
HImage1.DataFormat = 0            ' Binding field contains file name
HImage1.DataField = "PicFile"     ' Binding field is called PicFile
```

Examples

Data Type

Integer (enumerated)

Data Value

The allowed **DataFormat** values are:

Setting	Description
0	(Default). **File Name.** The binding field contains a file name.
1	**DIB.** The binding field contains an image bitmap in DIB (Device Independent Bitmap) format.
2	**Picture.** The binding field contains an image bitmap in Visual Basic Picture format.
3	**Memory File.** The binding field contains an image bitmap in one of the supported file formats (e.g., JPEG, TIFF, PCX). When a memory file is read from a database record, its format is automatically discerned from its header information. When the image is updated and written back to the database, it is encoded using the format and compression method specified by the **FileFormat** and **FileCompression** properties, which, if they haven't been changed by your program, will automatically be set to the format and compression at which the image was initially read.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**DataFormat**[= *setting%*]

DrawMode Property, HImage Control

Description

This property is used to specify how an image is to be combined with the background or brush pixels. Using this property, image and background pixels can be combined using bitwise operations to produce various visual effects.

Note - These operations only affect the way in which an image is displayed on the screen; its pixel values are not actually modified.

Data Type

Integer (enumerated)

Data Value

The allowed **DrawMode** values are:

Setting	Description
0	(Default) **SRCCOPY.** The image pixels replace the background pixels.
1	**SRCPAINT.** The image pixels are ORed with the background pixels.
2	**SRCAND.** The image pixels are ANDed with the background pixels.
3	**SRCINVERT.** The image pixels are XORed with the background pixels.
4	**SRERASE.** The image pixels are ANDed with the background pixels.
5	**NOTSRCOPY.** Inverted image pixels replace the background pixels.
6	**NOTSRCERASE.** Inverted image pixels are ANDed with inverted background pixels.
7	**MERGECOPY.** The image pixels are ANDed with the current brush pattern, and the result replaces the background pixels.
8	**MERGEPAINT.** Inverted image pixels are ORed with the background pixels.
9	**PATCOPY.** Background pixels are replaced with the current brush pattern.
10	**PATPAINT.** Inverted image pixels are ORed with the current brush pattern. This result is ORed with the background pixels.
11	**PATINVERT.** The current brush pattern is XORed with the background pixels.
12	**DSTINVERT.** Background pixels are inverted.
13	**BLACKNESS.** All image pixels are set to black.
14	**WHITENESS.** All image pixels are set to white.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**DrawMode**[= *setting%*]

ErrorCode Property, HImage Control

Description

This property holds the error code returned by the last HImage control operation. Both the **ErrorCode** and **ErrorString** properties are automatically set by the HImage control when an error occurs (**ErrorString** will hold a brief description of the error). To clear the **ErrorCode** property after an error has occurred, set it to 25000.

Data Type

Integer (enumerated)

Data Value

The **ErrorCode** value is an integer representing the error generated by the last HImage operation. If the last operation did not cause an error, this property will contain the value, 25000.

See HImage Error Messages for a complete list of HImage error codes and messages.

Visual Basic

[*form*.]*HImage*.**ErrorCode**[= *setting%*]

ErrorShow Property, HImage Control

Description

This property determines whether an error is returned to Visual Basic or is simply noted in the **ErrorCode** and **ErrorString** properties.

See Error Messages for a complete list of error codes and messages.

Data Type

Integer (Boolean)

Data Value

The **ErrorShow** property settings are:

Setting	Description
True (-1)	(Default). Specifies that the control is to immediately return an error code and error message to Visual Basic, when an error occurs. The code and message is contained in the **ErrorCode** and **ErrorString** properties, respectively.
False (0)	Specifies that the control is to set the **ErrorCode** and **ErrorString** properties, but is not to return an error code to Visual Basic. This setting lets you decide whether (and how) your program will respond to HImage errors when they occur.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ErrorShow**[{**True** | **False**}]

ErrorString Property, HImage Control

Description

This property contains a string describing the error associated with the error code contained in the **ErrorCode** property. The **ErrorString** and **ErrorCode** properties are automatically written by the HImage control whenever an error occurs within it.

ErrorString is a read-only property.

See Error Messages for a complete list of error codes and messages.

Data Type

String

Visual Basic

[*form*.]*HImage*.**ErrorString**

FileCommand Property, HImage Control

Description

This property performs image-file input and output operations. It can be used to explicitly load or save an image; or, it can be set to automatically load an image whenever the **FileName** property is changed.

In the following example, **FileCommand** is used to save an image.

```
HImage1.FileCommand = 0              ' Disable auto open
HImage1.Filename = "C:\balloon.tif"  ' Identifies file
HImage1.FileFormat = 1               ' Sets TIF format
HImage1.FileCompression = 1          ' Sets default compression
HImage1.FileCommand = 3              ' Saves image
```

The first statement disables auto-open mode (in case it had been enabled earlier); the next three statements specify the name of the file and the format in which it is to be encoded. The last statement performs the actual save process.

Examples

Data Type

Integer (enumerated)

Data Value

The allowed **FileCommand** values are:

Setting	Description
0	**Assignment Only.** Does nothing. This option is usually set to disable the "auto-open" mode in preparation for a file save or append operation. See setting 1, below.
1	**Auto open.** Specifies that the "auto-open" mode is to be utilized. In "auto-open" mode, the image specified in **FileName** is automatically opened whenever that property (the **FileName** property) is changed. The file's format and compression mode are automatically determined based upon the contents of the file's header, and are automatically written to the **FileFormat** and **FileCompression** properties once the file is successfully opened. **Note -** if you have opened a file with "auto-open" and want to save it to a file of a different name, you must **1.** Set **FileCommand** to 0 to disable "auto-open", **2.** Change the **FileName** property, then **3.** Set **FileCommand** to 3 to save the file (or 4 to append).
2	**Open using file format.** Opens the file specified in the **FileName** property. The file is opened using the format specified by the **FileFormat** property.
3	**Save using file format.** Saves the image to the file specified in the **FileName** property. The file will be saved using the format and compression methods specified in the **FileFormat** and **FileCompression** properties, respectively. If you have enabled the "auto-open" option, it must be disabled before you can save the image to a new file name. See steps under setting 1, above.
4	**Append using file format.** Appends the image to the file specified in the **FileName** property using file format specified in the **FileFormat** property.

This option is used to create a multi-image file.

Note - currently, only TIFF and OS/2 BMP formats support multi-image files.

If you have enabled the "auto-open" option, it must be disabled before you can append the image to a new file name. See steps under setting 1, above.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**FileCommand**[= *setting%*]

FileCompression Property, HImage Control

Description

This property determines the file compression method used when an image is saved or appended using the **FileCommand** property. Your program should ensure that the **FileCompression** and **FileFormat** properties are correctly set before setting the **FileCommand** property to save or append an image to a file.

The **FileCompression** property need not be set before opening an image file, as the compression mode is automatically derived from the file itself. Once a file has been successfully opened, the **FileCompression** property will indicate the compression method in which it was encoded.

See the File Format Compression Table for a list of supported file formats and the compression methods supported by each.

Data Type

Integer (enumerated)

Data Value

The allowed **FileCompression** values are:

Setting	Description
0	**No Compression.** Use no compression. **Note -** some image formats do not support an uncompressed form. Refer to the File Format Compression Table for the modes supported by each.
1	(Default). **Default Compression.** Use the default compression method for the specified format and image class.
2	**RLE.** Use the Run Length Encoding (RLE) compression method.
3	**CCITT 1D.** Use CCITT Modified Huffman, 1-Dimensional encoding. Use this method with Bilevel TIFF files only.
4	**CCITT Group 3.** Use CCITT Group3 Fax encoding. Use this method with Bilevel TIFF files only.
5	**CCITT Group 4.** Use CCITT Group4 Fax encoding. Use this method with Bilevel TIFF files only.
6	**LZW.** Use modified Lempel-Zif encoding. Use this method with TIFF and GIF formats only.
7	**LZW Horz Predictor.** Use modified Lempel-Zif encoding with horizontal differencing predictor. Use with 8 bits-per-pixel TIFF files only.
8	**JPEG.** Use JPEG compression with the quality factor specified in the **FileJPEGQuality** property.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**FileCompression**[= *setting%*]

File Format Compression Table, HImage Control

The following table shows the image classes and file compression methods supported for each file format.

File Format	Image Class	Default Compression	Other Compression
1 (TIFF)	0	3 (CCITT 1D)	0 , 2 , 4 , 5
	1	7 (LZW Horz)	0 , 2 , 6
	2	7 (LZW Horz)	0 2 , 6
	3	7 (LZW Horz)	0 , 2 , 6
2 (BMP/DIB)	0 , 3	0 (No compression)	-
	1 , 2	2 (RLE)	0
3 (HALO Cut)	0 , 3	Not supported	-
	1 , 2	2 (RLE)	-
4 (PCX)	0 , 1 , 2 , 3	2 (RLE)	-
5 (Adobe EPS)	0 , 1 , 2 , 3	0 (No compression)	-
6 (Targa)	0	Not supported	-
	1 , 2 , 3	2 (RLE)	0
7 (GIF)	0 , 1 , 2	6 (LZW)	-
	3	Not supported	-
8 (Sun Raster)	0 , 1 , 2 , 3	2 (RLE)	0
9 (JPEG)	0 , 2	Not supported	-
	1 , 3	8 (JPEG)	0 - 100%
10 (PICT)	0 , 1 , 2 , 3	2 (RLE)	0

FileFormat Property, HImage Control

Description

This property determines the format used when an image is explicitly loaded, saved or appended using the **FileCommand** property.

If you are explicitly loading an image file, i.e., not using "auto-open", your program should set the **FileFormat** property before setting **FileCommand** (when a file is opened in "auto-open" mode, the **FileFormat** property is ignored, and the format is automatically determined from the file's header). If you don't know the format of the file you are opening, you can set **FileFormat** to 0. This instructs the HImage control to automatically detect the file's format from its header information.

Once an image has been opened, its format and compression are written to the **FileFormat** and **FileCompression** properties, respectively.

When you are saving or appending an image to a file, be sure to set the **FileCompression** and **FileFormat** properties before setting the **FileCommand** property.

Examples

See the File Format Compression Table for a list of supported file formats and the compression methods allowed by each.

Data Type

Integer (enumerated)

Data Value

The allowed **FileFormat** values are:

Setting	Description
0	(Default). **Unknown.** Unknown file format. Will auto-detect the format from the file's header. Use when loading a file only; this is not a valid value for save or append operations.
1	**TIFF.** TIFF file format. TIFF supports multi-image files.
2	**BMP, DIB, RLE.** BMP or DIB formats including Windows, OS/2 1.2 and 2.0 formats (HImage reads both Windows and OS/2 formats, and writes in Windows format). OS/2 2.0 format supports multi-image files.
3	**HALO Cut.** Media Cybernetics HALO Cut file format.
4	**PCX.** ZSoft PCX file format.
5	**EPS.** Adobe Postscript file format (write only).
6	**Targa.** Truevision Targa file format.
7	**GIF.** CompuServe GIF file format.
8	**Raster.** Sun Rasterfile format.
9	**JPEG.** JPEG file format. Conforms to JFIF V1.02.
10	**PICT.** Apple PICT file format (Bitmap only).

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**FileFormat**[= *setting%*]

FileJPEGQuality Property, HImage Control

Description

Use this property to set the quality factor that is to be used when an image is stored in JPEG format (**FileFormat** = 9 and **FileCompression** = 8).

Because JPEG is a "lossy" compression method, a certain amount of information is always lost when an image is stored in this format. The **FileJPEGQuality** property lets you control the amount of information loss -- high **FileJPEGQuality** values retain more image data, but do not result in small files, whereas low **FileJPEGQuality** values create very compact files, but eliminate more visual information (highly compressed JPEG files are usually good for thumbnails and icons).

Examples

Data Type

Integer

Data Value

The **FileJPEGQuality** is an integer from 0 to 100 representing a quality factor from 0% to 100%. The higher the percentage, the better the quality of the compressed image. Since the JPEG quality factor cannot be determined from an image file, the **FileJPEGQuality** property is always set to 75 when a JPEG file is loaded.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**FileJPEGQuality**[= *setting%*]

FileName Property, HImage Control

Description

The **FileName** property specifies the name of the image file from which an image is to be read, or to which an image is to be saved, when the **FileCommand** property is set.

Important - if the **FileCommand** property is set to "auto-open", any change to the **FileName** property causes the specified file to be immediately loaded.

Examples

Data Type

String

Data Value

The **FileName** property contains a string specifying the name of the file to be opened.

Note - if "auto-open" mode is enabled and you need to change the content of **FileName** without triggering the automatic load process (in preparation for saving the image to another file, for example), you must disable "auto-open" before changing the **FileName** property. This is done by setting **FileCommand** to 0.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**FileName**[= *setting$*]

FilePage, FilePages Properties, HImage Control

Description

The **FilePage** and **FilePages** properties are used to load an image from a multi-image file.

Important - not all file formats support multi-image files. Currently, multiple-image files are supported by TIFF, OS/2 BMP and DCX file formats.

When a file is loaded, the number of images contained in the file (normally, 1) is written to the **FilePages** property. Your program can query this property to determine whether the file contains multiple images, and if so, how many.

FilePages is a read-only property.

The **FilePage** property is used to select a specific image in a multiple-image file, where 0 specifies the first image, 1 specifies the second image and so forth (up to **FilePages**-1). You must set the **FilePage** property after (not before) the file is opened (when an image is initially loaded, **FilePage** is automatically set to 0, and the first image in the file is automatically opened). Once the first image has been loaded, your program can select subsequent images in that file by simply changing the **FilePage** property.

Note - the **FilePage** and **FilePages** properties are only used during file open operations -- they are not used during file save or append operations. Multi-image files are created (written) using the **FileCommand** append setting.

These values are not saved with your Visual Basic form.

Examples

Data Type

Integer

Data Value

The **FilePage** property value can be set to a value from 0 to **FilePages**-1. In most cases **FilePages** will be 1, since few file formats support multi-image files.

Visual Basic

[*form*.]*HImage*.**FilePage**[= *setting%*]
[*form*.]*HImage*.**FilePages**

FilePath Property, HImage Control

Description

The **FilePath** property is used to specify the path of the file specified in **FileName**. It is only used when the string in **FileName** contains *no* path information.

If you want **FilePath** to be used, be sure that your **FileName** string contains *nothing but the file name* (e.g., "filename.bmp"); the presence of *any* path information will cause **FilePath** to be ignored. The following strings are all examples of names that contain path information:

```
HImage1.FileName = "C:filename.bmp"
HImage1.FileName = "IMG\filename.bmp"
HImage1.FileName = "\filename.bmp"
HImage1.FileName = "C:\IMG\filename.bmp".
```

FilePath is used when reading *and* storing an image file.

Data Type

String

Data Value

The **FilePath** value is a string that specifies the path to the image file. The following example shows how the **FilePath** property would be set if the image file were in D:\ images (notice that the path *does not* include the backslash that usually delimits the subdirectory from the file name).

```
HImage1.FilePath = "D:\images"       ' Sets the path
HImage1.FileName = "balloon.tif"     ' Assigns the file name
```

The following strings are also valid **FilePath** settings:

```
HImage1.FilePath = "D:"              ' Specifies the current
                                     ' directory on the D: drive.
HImage1.FilePath = "D:\"             ' Specifies the root directory
                                     ' on the D: drive.
HImage1.FilePath = "IMG"             ' Specifies the IMG subdirectory
                                     ' within the current directory
                                     ' on the current drive.
HImage1.FilePath = "\IMG"            ' Specifies the \IMG directory
                                     ' on the current drive.
```

Visual Basic

[*form*.]*HImage*.**FilePath**[= *setting$*]

FilterCommand, FilterSize, FilterStrength Properties, HImage Control

Description

The **FilterCommand**, **FilterSize**, and **FilterStrength** properties are used to apply HALO enhancement filters to an image. They can be used to enhance an image (e.g., sharpen or soften it) or produce unusual visual effects (e.g., metallize or sculpt).

By assigning a value to the **FilterCommand** property, you instruct the control to apply the filter associated with that value (e.g., a value of 16 applies the median filter, which removes spot noise from an image). See "Data Value" below for a list of the allowed filter values.

The **FilterSize** and/or **FilterStrength** properties specify options that are used by certain filters (see "Data Value" below for the filter types that use these properties). As shown by the following example, these properties, when required, must be set before setting **FilterCommand**.

```
HImage1.FilterStrength = 100          'Use full-strength filter
HImage1.FilterCommand = 20            'Apply sharpening filter to image
```

In this example, the first statement sets the option required by the sharpening filter, and the last statement performs the filtering operation.

Data Type

Integer (enumerated)

Data Value

The **FilterCommand** property settings are:

Command	Strength	Size	Description
0	Not used	Not used	Does nothing.
1	Not used	Not used	**Thin Edge Detection.** Produces thin edges.
2	Not used	Not used	**Thick Edge Detection.** Produces thick edges.
3	Not used	Not used	**Horz Edge Detection.** Detects and traces horizontal edges.
4	Not used	Not used	**Vert Edge Detection.** Detects and traces vertical edges.
5	Not used	Not used	**Sculpt Above.** Produces a sculpted look with the light source from above.
6	Not used	Not used	**Sculpt Left.** Produces a sculpted look with the light source from left.
7	Not used	Not used	**Sculpt Diag.** Produces a sculpted look with the light source from upper left corner.
8	Not used	Not used	**Sculpt Metallic.** Produces a shiny, metallic look.
9	Not used	Not used	**Emboss Above.** Produces an embossed look with the light source from above.
10	Not used	Not used	**Emboss Left.** Produces an embossed look with the light source from the left side.
11	Not used	Not used	**Emboss Diag.** Produces an embossed look with the light source from the upper-left corner.
12	Not used	Not used	**Horz Line Detection.** Accentuates horizontal lines and de-emphasizes all other parts of the image.

13	Not used	Not used	**Vert Line Detection.** Accentuates vertical lines and de-emphasizes all other parts of the image.
14	Not used	Not used	**Horz/Vert Line Detection.** Accentuates vertical and horizontal lines and de-emphasizes all other parts of the image.
15	%	Not used	**Blur.** Softens or blurs the image. **FilterStrength** should specify a value from 0 to 100, representing the strength at which the filter is to be applied (100 applies the filter at full strength).
16	Sensitivity	Cell size	**Despeckle.** Removes "noise" from the image. Noise is defined as any pixel exceeding the median kernel value by more than the specified sensitivity. Sensitivity is specified in **FilterStrength**, and is expressed as a percentage (0 to 100). Kernel size is specified in **FilterSize** and must be a value of 3, 5, or 7.
17	Not used	Not used	**Invert.** Inverts the image values (produces a negative).
18	Not used	Cell Size	**Pixelize.** Pixelizes the image by averaging the contents of **FilterSize** x **FilterSize** areas.
19	Not used	Bits	**Posterize.** Reduces the number of levels in an image by eliminating (setting to zero) the least-significant bits in each pixel. **FilterSize** must specify the number of most-significant bits to retain.
20	%	Not used	**Sharpen.** Sharpens the image. **FilterStrength** should specify a value from 0 to 100, representing the strength at which the filter is to be applied (100 applies the filter at full strength).

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**FilterStrength**[= *setting%*]
[*form*.]*HImage*.**FilterSize**[= *setting%*]
[*form*.]*HImage*.**FilterCommand**[= *setting%*]

hDIB Property, HImage Control

Description

The **hDIB** property is used to copy image data between an HImage control and the Clipboard in DIB (Device Independent Bitmap) format. When copying an image to the Clipboard, you must use **hDIB** (or **Picture**) instead of **Image** so that the data are copied in a Clipboard-compatible format.

```
Clipboard.Clear                     ' Initialize Clipboard
Clipboard.SetData HImage1.hDIB      ' Copy image in DIB format
```

This property is available at run time, only.

Data Type

Integer

Visual Basic

[*form*.]*HImage*.**hDIB**[= *setting%*]

Image Property, HImage Control

Description

The **Image** property contains the virtual-image handle representing the image in the control (this handle *is not* a memory handle). This property is generally used to copy image data from one HImage control to another. The following example shows how the **Image** property is used to do this.

```
HImage1.Image  =  HImage2.Image      ' Copy HImage1 to HImage2
```

You can also use the **Image** property to copy an image from one control to a selection box in another (the image will be cropped if it is too large to fit within the selected destination). The example below shows how this is done:

```
HImage1.Image  =  HImage2.SelImage   ' Copy selection box to HImage1
```

This property is available at run time, only.

Note - if you want to copy an image from an HImage control to a Visual Basic Picture Box or to the Clipboard, you must use the HImage **Picture** or **hDIB** (Clipboard only) properties, not the **Image** property.

Examples

Data Type

Integer

Data Value

The **Image** value is a positive integer (including 0) that uniquely identifies the image in the control. When you assign this property to another HImage **Image** property, a copy of the source image is placed in the destination control (note, the destination receives a copy of the source image; it does not obtain ownership of the actual source data).

The **Image** and **SelImage** properties are compatible, which means you can copy image data directly between the two as shown in the example above.

When there is no image in the control, the **Image** property will contain a negative value.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**Image**[= *setting%*]

ImageArtist Property, HImage Control

Description

This property is used to assign the name of an artist to an image in a TIFF file (currently only TIFF files record an artist name with an image). When you want to include an artist name with the TIFF file, you must set this property before you save or append the image with the **FileCommand** property.

Note - when a TIFF file is opened, the artist name, if there is one, is automatically written to this HImage property.

Data Type

String

Data Value

The **ImageArtist** value is a string representing the name of the artist. If no artist is associated with the image, this property will be null.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ImageArtist**[= *setting$*]

ImageClass Property, HImage Control

Description

The **ImageClass** property describes the organization and bit depth of the image, according to a set of predefined HALO image classes (see table below).

ImageClass is a read-only property.

Note - the **ImageClass** property merely describes the class of the image within the control; it is not used to convert the image to another class. To perform a class conversion, use the **ConvertCommand** property.

Data Type

Integer (enumerated)

Data Value

The allowed **ImageClass** values are:

Setting	Description
0	(Default). **Bilevel.** Each pixel is stored as a bit. A bit value of 0 is black and a bit value of 1 is white.
1	**Gray Scale.** Each pixel is stored as a byte (8-bits) representing a level of grayness from black (0) to white (255). There is no palette associated with the image.
2	**Palette.** Each pixel is stored as a byte (8-bits). There is a color palette associated with this kind of image. Each pixel value is an index to the R,G,B palette.
3	**True Color.** Each pixel is stored as an RGB triple -- i.e., 24 bits, where the first 8 bits specifies Red, the second 8 bits is Green and the last 8 bits is Blue. There is no padding between pixels.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ImageClass**

ImageDate Property, HImage Control

Description

The **ImageDate** property contains the date the image was last modified. When an image is saved in a format that supports this property, the contents of **ImageDate** is automatically written to the file.

ImageDate is a read-only property.

Data Type

String

Data Value

The **ImageDate** value is a string representing the date of last modification. If no modification date is associated with the image, this property will be null.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ImageDate**[= *setting$*]

ImageDescription Property, HImage Control

Description

This property is used to record a description with an image in a TIFF file (currently only TIFF files allow you to include a description with an image). When you want to include a description with your TIFF file, set the **ImageDescription** property before saving or appending the image using the **FileCommand** property.

Note - when a TIFF file is opened, the description, if there is one, is automatically written to this HImage property.

Data Type

String

Data Value

The **ImageDescription** value is a string representing a comment or description. If no description is associated with the image, this property will be null.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ImageDescription**[= *setting$*]

ImageHeight, ImageWidth Properties, HImage Control

Description

The **ImageHeight** and **ImageWidth** properties contain the size of the image, in pixels. These properties are maintained by the HImage control -- they are set when the image is initially opened, and are automatically modified whenever the image size is changed (e.g., when rotated or resized).

ImageHeight and **ImageWidth** are read-only properties.

Data Type

Integer

Data Value

The **ImageHeight** and **ImageWidth** values represent the number of pixels in each dimension (y and x, respectively).

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ImageHeight**
[*form*.]*HImage*.**ImageWidth**

ImageResX, ImageResY Properties, HImage Control

Description

The **ImageResX** and **ImageResY** properties contain the "physical" resolutions, expressed in dots per inch (DPI), associated with an image. These values are used to determine the real-world size at which an image is displayed or printed.

Note - not all file formats record **ImageResX** and **ImageResY** data. For those that do, these values will be automatically assigned when the image is read, and will be automatically stored when the image is saved.

Data Type

Integer

Data Value

The **ImageResX** and **ImageResY** values represent dots-per-inch in the horizontal and vertical directions, respectively.

Important - never set these values to 0! To obtain the physical size of an image your application (or other applications) must divide **ImageWidth** and **ImageHeight** by **ImageResX** and **ImageResY**, respectively. If the resolution properties have been set to 0, a divide-by-zero error will result.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ImageResX**[= *setting%*]
[*form*.]*HImage*.**ImageResY**[= *setting%*]

ImageTitle Property, HImage Control

Description

This property is used to record a title with an image. To include a title with your image file, set the **ImageTitle** property before saving or appending the image using the **FileCommand** property.

Note - not all file formats record title data. However, if a title is detected when an image is opened, it will be automatically written to the **ImageTitle** property. And, whenever an image is saved in a format that supports title data, HImage will automatically save the contents of **ImageTitle** with the image.

Data Type

String

Data Value

The **ImageTitle** value is a string representing a user-assigned, image name. If no name is associated with the image, this property will be null.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ImageTitle**[= *setting* $]

LutBrightness Property, HImage Control

Description

The **LutBrightness** property is used to adjust the brightness of an image. In color images (True Color or Palette) the **LutBrightness** property can be applied to the entire image or to a selected color channel (i.e., the Red, Green or Blue channel).

This property is available at run time, only.

Data Type

Integer (array)

Data Value

LutBrightness values range from 0 to 100, where 50 represents no change to the image brightness. Values less than 50 reduce brightness (darken the image) and values greater than 50 increase brightness (lighten the image).

LutBrightness is a property array. The number of elements in the array depends upon the "class" of image in the HImage control.

If the control contains a color image (i.e., Palette or True Color -- **ImageClass** 2 or 3, respectively), the array will contain 4 elements. Each element is associated with a specific color channel, as follows:

LutBrightness(0) is the Luminance channel

LutBrightness(1) is the Red channel

LutBrightness(2) is the Green channel

LutBrightness(3) is the Blue channel

By assigning a brightness value to the appropriate element, you can adjust the brightness of a specific color channel in a True Color or Palette image.

If the control contains a Gray Scale image (**ImageClass** 1), the array will contain a single element, 0, representing overall intensity:

LutBrightness(0) is the Luminance channel

If the control contains a Bilevel image (**ImageClass** 0), the array will contain a single element, 0, however changing the brightness of a Bilevel image has no effect. If you read **LutBrightness** for a Bilevel image, you will be given a value of 50.

Visual Basic

[*form*.]*HImage*.**LutBrightness(0)**[= *setting%*]
[*form*.]*HImage*.**LutBrightness(1)**[= *setting%*]
[*form*.]*HImage*.**LutBrightness(2)**[= *setting%*]
[*form*.]*HImage*.**LutBrightness(3)**[= *setting%*]

LutContrast Property, HImage Control

Description

The **LutContrast** property is used to adjust the difference between the lightest and darkest values in an image. It can be used to exaggerate differences in an image that is made of very similar tones, or reduce the harshness of an image containing strong dark and light transitions. In color images (True Color or Palette class) the **LutContrast** property can be applied to the entire image or to a selected color channel (i.e., the Red, Green or Blue channel).

This property is available at run time, only.

Data Type

Integer (array)

Data Value

LutContrast values range from 0 to 90, where 45 represents no change to the image contrast. Values less than 45 reduce contrast and values greater than 45 increase contrast.

LutContrast is a property array. The number of elements in the array depends upon the "class" of image in the HImage control.

If the control contains a color image (Palette or True Color; **ImageClass** 2 or 3, respectively), the array will contain 4 elements. Each element is associated with a specific color channel, as follows:

LutContrast(0) is the Luminance channel

LutContrast(1) is the Red channel

LutContrast(2) is the Green channel

LutContrast(3) is the Blue channel

By assigning a contrast value to the appropriate element, you can adjust the contrast of a specific color channel in a True Color or Palette image.

If the control contains a Gray Scale image (**ImageClass** 1), the array will contain a single element, 0, representing image intensity:

LutContrast(0) is the Luminance channel

If the control contains a Bilevel image (**ImageClass** 0), the array will contain a single element, 0, however changing the contrast of a Bilevel image has no effect on it. If you read **LutContrast** for a Bilevel image, you will be given a value of 45.

Visual Basic

[*form*.]*HImage*.**LutContrast(0)**[= *setting%*]
[*form*.]*HImage*.**LutContrast(1)**[= *setting%*]
[*form*.]*HImage*.**LutContrast(2)**[= *setting%*]
[*form*.]*HImage*.**LutContrast(3)**[= *setting%*]

LutGamma Property, HImage Control

Description

The **LutGamma** property is used to increase the contrast in the image midtones without affecting its highlight (white) and shadow (black) points. It can be used to optimize contrast in the very dark or very light regions of an image, which is often required when "correcting" an image for a particular device.

In color images (True Color or Palette class) the **LutGamma** property can be applied to the entire image or to a selected color channel (i.e., the Red, Green or Blue channel).

This property is available at run time, only.

Data Type

Integer

Data Value

The **LutGamma** value is an integer from 0 to 40, representing an index into a table of predefined, logarithmic gamma scales. A gamma value of 20 (default) represents no change to the image. Values less than 20 reduce gamma, which will increase contrast in the lighter regions, and darken the image overall. Gamma values greater than 20 will increase the contrast in the darker regions, and brighten the image overall.

Note - you can convert this index to a floating-point gamma value using the **HicGetGammaValue** function. Conversely, a floating-point gamma value can be converted to a gamma index using **HicGetGammaIndex**.

LutGamma is a property array. The number of elements in the array depends upon the "class" of image in the HImage control.

If the control contains a color image (Palette or True Color; **ImageClass** 2 or 3, respectively), the array will contain 4 elements. Each element is associated with a specific color channel, as follows:

LutGamma(0) is the Luminance channel

LutGamma(1) is the Red channel

LutGamma(2) is the Green channel

LutGamma(3) is the Blue channel

By assigning a gamma value to the appropriate element, you can adjust the gamma of a specific color channel in a True Color or Palette image.

If the control contains a Gray Scale image (**ImageClass** 1), the array will contain a single element, 0, representing image intensity:

LutGamma(0) is the Luminance channel

If the control contains a Bilevel image (**ImageClass** 0), the array will contain a single element, 0, however changing the gamma of a Bilevel image has no effect on it. If you read **LutGamma** for a Bilevel image, you will be given a value of 20.

Visual Basic

[*form*.]*HImage*.**LutGamma(0)**[= *setting%*]
[*form*.]*HImage*.**LutGamma(1)**[= *setting%*]
[*form*.]*HImage*.**LutGamma(2)**[= *setting%*]
[*form*.]*HImage*.**LutGamma(3)**[= *setting%*]

MergeMode Property, HImage Control

Description

The **MergeMode** property is used to specify how a selection box is to be combined with the pixels in the destination image. For example, the source pixels can be directed to replace the destination pixels, or the pixel values can be combined using bitwise operations.

Data Type

Integer (enumerated)

Data Value

The allowed **MergeMode** property values are as follows:

Setting	Description
0	(Default) **Copy.** Source pixel values replace destination pixel values.
1	**And.** Source and destination pixel values are ANDed. Only bit values that are "on" in both images are "on" in the result.
2	**Or.** Source and destination pixel values are ORed. Bit values that are "on" in either image are "on" in the result.
3	**Nand.** Source and destination pixel values are NANDed. Bit values that are "off" in either or both images are "on" in the result.
4	**Nor.** Source and destination pixel values are NORed. Bit values that are "off" in both images are "on" in the result.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**MergeMode**[= *setting%*]

Picture Property, HImage Control

Description

The **Picture** property is used to copy image data between an HImage control and a Picture Box or the Clipboard. When copying an image to these objects, you must use **Picture** instead of **Image** so that the data are copied in a compatible format (the data are copied in Visual Basic Picture format). The example below shows how the **Picture** property would be used to copy image data to a Picture Box control.

```
Picture1.Picture = HImage1.Picture   ' Copy HImage1 to a Picture Box
```

Note - the **hDIB** property can also be used to copy data between the Clipboard and an HImage control. It copies the data in DIB format.

The **Picture** property is available at run time, only.

Examples

Data Type

Integer

Visual Basic

[*form*.]*HImage*.**Picture**[= *setting%*]

PrintCommand Property, HImage Control

Description

The **PrintCommand** property is used to print the image in the HImage control. The value assigned to this property determines whether the image is printed automatically using the existing position-related print properties, or whether the user will pass these values in the "Print" dialog box.

Note - printing via the **PrintCommand** property is just one of three ways in which an image can be printed. You can also use Visual Basic's standard PrintForm method to print the entire form, or print the image to a given device context using the **PrinthDC** property.

Examples

Data Type

Integer (enumerated)

Data Value

The **PrintCommand** property settings are:

Setting	Description
0	(Default). **None.** Does nothing.
1	**Print.** Prints the image using the current settings specified by the **PrintPosX**, **PrintPosY**, **PrintWidth**, **PrintHeight**, **PrintDitherType**, **PrintDitherOption**, and **PrintSmooth** properties.
2	**Setup position.** Issues the "Print" dialog box from which the user can select a printer and specify size, position and halftone options. The **PrintPosX**, **PrintPosY**, **PrintWidth**, **PrintHeight**, **PrintDitherType**, **PrintDitherOption**, and **PrintSmooth** properties are set when the user clicks "OK" in this dialog box.
3	**Setup and print.** Issues the Print dialog box, allowing the user to select a printer and specify size, position and halftone options. The image is immediately printed with the selected options when the user clicks Print.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**PrintCommand**[= *setting%*]

PrintDitherOption, PrintDitherType Properties, HImage Control

Description

Specifies the dithering and halftoning techniques used when a Gray Scale, Palette or True Color image is printed using the **PrintCommand** property. These properties are ignored when a Bilevel image is printed.

Examples

Data Type

Integer

Data Value

The **PrintDitherType** and **PrintDitherOption** settings are:

DitherType	DitherOption	Description
0	0-3	**Angle Dot Screen.** Prints the image using the Angle-Dot halftone screen at the coarseness level specified in **PrintDitherOption**.
1	0-3	**Flat Dot Screen.** Prints the image using the Flat-Dot halftone screen at the coarseness level specified in **PrintDitherOption**.
2	0-3	**Angle Line Screen.** Prints the image using the Angle-Line halftone screen at the coarseness level specified in **PrintDitherOption**.
3	0-3	**Horz Line Screen.** Prints the image using the Horizontal-Line halftone screen and the coarseness level specified in **PrintDitherOption**.
4	0-3	**Vert Line Screen.** Prints the image using the Vertical-Line halftone screen at the coarseness level specified in **PrintDitherOption**.
5	0	**Error Diffusion.** Prints the image using the Floyd Steinberg (4 Weights) error-diffusion halftone method.
	1	**Error Diffusion.** Prints the image using the Stucki (12 Weights) error-diffusion halftone method.
	2	**Error Diffusion.** Prints the image using the Floyd Steinberg (Fuzzy) error-diffusion halftone method.
	3	**Error Diffusion.** Prints the image using the Random White Noise error-diffusion halftone method.
6	Not used	**Threshold.** Prints the image using the threshold method. 128 is the threshold value.
7	Not used	**Printer Halftone.** Prints using the printer's halftone.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**PrintDitherType**[= *setting%*]
[*form*.]*HImage*.**PrintDitherOption**[= *setting%*]

PrinthDC Property, HImage Control

Description

The **PrinthDC** property is used to send an image to a printer-object hDC. This property allows you to combine the image in the HImage control with other program-defined output elements (e.g., text, data or other images).

PrinthDC is a write-only property. This property is available at run time, only.

Examples

Data Type

Integer

Data Value

The **PrinthDC** property is a number that represents a Device Context.

Visual Basic

[*form*.]*HImage*.**PrinthDC**[= *setting%*]

PrintHeight, PrintWidth Properties, HImage Control

Description

The **PrintWidth** and **PrintHeight** properties are used to specify the size at which an image is to be printed. These properties are utilized when the image is printed using the **PrintCommand** property.

Examples

Data Type

Single

Data Value

The **PrintWidth** and **PrintHeight** values represent image width and height (respectively), and are expressed in terms specified by the **PrintUnit** property.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**PrintWidth**[= *setting!*]
[*form*.]*HImage*.**PrintHeight**[= *setting!*]

PrintPosX, PrintPosY Properties, HImage Control

Description

The **PrintPosX** and **PrintPosY** properties are use to specify the printed image's position on the printed page. These properties are utilized when the image is printed with the **PrintCommand** property.

In the following example, the unit-of-measure is set to inches, and the printed image is positioned 1.5" from the top edge and 2.5" from the left edge of the page:

```
HImage1.PrintUnit = 0          'Set unit to inches
HImage1.PrintPosX = 2.5        'Set Left Margin
HImage1.PrintPosY = 1.5        'Set Top Margin
```

Examples

Data Type

Single

Data Value

The **PrintPosX** and **PrintPosY** values represent the positions of the left and top edges (respectively) of the image. These properties are expressed in terms specified by **PrintUnit**.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**PrintPosX**[= *setting!*]
[*form*.]*HImage*.**PrintPosY**[= *setting!*]

PrintSmooth Property, HImage Control

Description

The **PrintSmooth** property is used to enable and disable the smoothing feature, which is used when the image is printed with **PrintCommand** or **PrinthDC**.

Smoothing compensates for pixels that are added or lost during scaling. When smoothing is enabled, a bilinear smoothing technique is applied, which interpolates pixels between adjacent lines in the enlarged or reduced image. This reduces "jaggies" in the output. Smoothing increases processing time, but produces the highest-quality output.

Data Type

Integer (Boolean)

Data Value

The **PrintSmooth** property settings are:

Setting	Description
False (0)	(Default). Do not apply smoothing.
True (-1)	Apply the smoothing algorithm during printing.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**PrintSmooth**[{**True** | **False**}]

PrintUnit Property, HImage Control

Description

The **PrintUnit** property determines the unit-of-measure in which the **PrintPosX**, **PrintPosY**, **PrintWidth**, and **PrintHeight** values are expressed.

Data Type

Integer (enumerated)

Data Value

The allowed **PrintUnit** values are:

Setting	Description
0	(Default). Inches
1	Centimeters
2	Pixels
3	TWIPS

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**PrintUnit**[= *setting%*]

ResizeCommand Property, HImage Control

Description

The **ResizeCommand** property is used to change the size of an image (i.e., scale the image) to the dimensions specified by the **ResizeWidth** and **ResizeHeight** properties.

To produce the highest-quality result, you can optionally apply the bilinear smoothing algorithm. This feature eliminates the "jaggies", which often occur when an image is significantly enlarged.

The following example shows how the resize command would be used to change the image size to 200 x 300 pixels.

```
HImage1.ResizeWidth = 200            'Set New Width
HImage1.ResizeHeight = 300           'Set New Height
HImage1.ResizeCommand = 2            'Performs Resize w/Smoothing
```

The first two statements specify the dimensions to which the image will be resized, and the last statement resizes the image.

Note - if a selection box is active when the **ResizeCommand** is set, it will be reset (i.e., deactivated).

Data Type

Integer (enumerated)

Data Value

The allowed **ResizeCommand** values are:

Setting	Description
0	(Default). **None.** Does nothing.
1	**Decimate.** Uses pixel decimation when scaling the image. This method is fast, but when the image is substantially resized, "jaggies" may result.
2	**Bilinear Smoothing.** Uses bilinear smoothing when scaling the image. Smoothing compensates for pixels that are added or lost during scaling, by interpolating pixels between adjacent lines in the enlarged or reduced image. Smoothing increases processing time, but produces the highest-quality result.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ResizeCommand**[= *setting%*]

ResizeHeight, ResizeWidth Properties, HImage Control

Description

The **ResizeHeight** and **ResizeWidth** properties specify the dimensions (in pixels) to which an image will be resized when the **ResizeCommand** property is set.

Data Type

Integer

Data Value

The **ResizeWidth** and **ResizeHeight** values represent new width and height dimensions (respectively) in pixels.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ResizeWidth**[= *setting%*]
[*form*.]*HImage*.**ResizeHeight**[= *setting%*]

RotateAngle Property, HImage Control

Description

The **RotateAngle** property specifies the number of degrees by which an image is to be rotated, when the **RotateCommand** is set to 1 (rotate with angle).

Data Type

Single

Data Value

The **RotateAngle** value represents degrees in the counterclockwise direction. For example, a **RotateAngle** value of 30 represents a rotation of 30 degrees, counterclockwise.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**RotateAngle**[= *setting!*]

RotateCommand Property, HImage Control

Description

The **RotateCommand** is used to rotate or reorient (e.g., flip) an image.

Note - if a selection box is active when **RotateCommand** is set, the selection box will be reset (deactivated).

Data Type

Integer (enumerated)

Data Value

The allowed **RotateCommand** values are:

Setting	Description
0	(Default). **None.** Does nothing.
1	**Rotate with Angle.** Rotates the image counterclockwise by the number of degrees specified in **RotateAngle**.
2	**Flip Left/Right.** Flips the image from left to right.
3	**Flip Top/Bottom.** Flips the image from top to bottom.
4	**Transpose.** Transposes the image, such that the upper-right corner becomes the lower-left corner.
5	**Rotate 90 deg CW.** Rotates the image counterclockwise by 270 degrees.
6	**Rotate 90 deg Counter CW.** Rotates the image counterclockwise by 90 degrees.
7	**Rotate 180 deg.** Rotates the image 180 degrees.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**RotateCommand**[= *setting%*]

SelCommand Property, HImage Control

Description

This property is used to hide, show, create or reset a selection box. It also allows the user to create a selection box by drawing it with their mouse.

Data Type

Integer (enumerated)

Data Value

The **SelCommand** property settings are:

Setting	Description
0	(Default). **Hide.** Hides the selection box border.
1	**Show.** Displays a border around the current selection box.
2	**Create.** Enables "selection mode". When this mode is enabled, the **MouseDown** and **MouseUp** events set the **SelTop/SelLeft** and **SelWidth/SelHeight** properties, respectively (this sequence will also trigger the **Select** event). This lets your user "draw" a selection box with their mouse. **Note -** if your program responds to a **MouseDown** event in the HImage control, be sure it first checks and, if necessary, disables "selection mode" (i.e., sets **SelCommand** to 0, 1 or 3).
3	**Reset.** Resets the selection box. Sets the **SelTop/SelLeft** and **SelWidth/SelHeight** properties such that they define the dimensions of the entire image.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**SelCommand**[= *setting%*]

SelhDIB Property, HImage Control

Description

The **SelhDIB** property is used to copy a selection box between an HImage control and the Clipboard, in DIB (Device Independent Bitmap) format. When copying a selection box to the Clipboard, you must use **SelhDIB** (or **SelPicture**) instead of **SelImage** so that the data are copied in a Clipboard-compatible format.

```
Clipboard.Clear                     ' Initialize Clipboard
Clipboard.SetData HImage1.SelhDIB   ' Copy image in DIB format
```

This property is available at run time, only.

Data Type

Integer

Visual Basic

[*form*.]*HImage*.**SelhDIB**[= *setting*%]

SelHeight, SelWidth Properties, HImage Control

Description

The **SelWidth** and **SelHeight** properties (along with **SelTop** and **SelLeft**) contain the dimensions and position of the current "selection box". A selection box is a rectangular area, equal to or smaller than the image. It is used to copy/paste a portion of an image to/from another control (or the Clipboard).

The **SelWidth** and **SelHeight** properties define the width and height of the selection box, in pixels. The **SelTop** and **SelLeft** properties define the position of the box within the image. These properties can be explicitly set by your program, or can be interactively set by your user using the **SelCommand** property's "create" mode (2).

Examples

Data Type

Integer

Data Value

The **SelWidth** and **SelHeight** values represent the horizontal and vertical length of the selection box, in pixels.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**SelWidth**[= *setting%*]
[*form*.]*HImage*.**SelHeight**[= *setting%*]

SelImage Property, HImage Control

Description

The **SelImage** property identifies the current selection box. A selection box is a rectangular area, equal to or smaller than the image. It is used to copy/paste a portion of an image to/from another control.

A selection box is created by setting the **SelLeft**, **SelTop**, **SelWidth**, and **SelHeight** properties. Therefore, your program must set these properties before referencing or assigning **SelImage** (your program can set these properties explicitly, or use the **SelCommand's** "create" mode (2) to set them interactively at run time).

Once a selection box is defined, its contents can be copied to another HImage control by assigning it to the other control's **Image** property, as shown in the following statement:

```
HImage2.Image = HImage1.SelImage     ' Copy sel box to HImage2
```

When a selection box is copied to an **Image** property, it replaces the image in the receiving control (the disposition of the control afterward is determined by the **AutoSize** property, just as if a new image had been loaded into it).

If you want to copy a selection box to just part of another HImage control, you must define a selection box in the receiving control and assign the source selection box to the receiving control's **SelImage** property. This is illustrated by the following example:

```
HImage2.SelImage = HImage1.SelImage  ' Copy to sel box in HImage2
```

If the receiving selection box is smaller than the source selection box, the bottom and right edges will be cropped to fit the destination.

You can also copy an entire image from one control to a selection box in another by assigning the **Image** property to the receiving control's **SelImage** property, as shown below

```
HImage2.SelImage = HImage1.Image     ' Copy HImage to sel box
```

The right and bottom edges of the source image will be cropped to fit the destination selection box, if necessary (tip - assigning the **SelImage** property to the **Image** property in its own control can be used to quickly crop an image).

This property is available at run time, only.

Examples

Data Type

Integer

Data Value

The **SelImage** value is a positive integer (including 0) representing the virtual image index of the current selection box. When you assign this property to another HImage **Image** or **SelImage** property, a copy of the selection box is placed in the destination control (note, the destination receives a copy of the source image; it is not given ownership of the actual source data).

The **Image** and **SelImage** properties are compatible, which means you can copy image data directly between the two as shown in the examples above. These properties are not compatible with the Visual Basic Picture Box or the Clipboard, however. To copy image data to/from these objects, use the **Picture** (or **hDIB**) and **SelPicture** (or **SelhDIB**) properties.

When there is no selection box, the **SelImage** property will contain a negative value. Any HImage operation that changes the size of an image (e.g., resize, rotate) will automatically set **SelImage** to a negative value (i.e., remove the selection box) when it

is finished.

Visual Basic

[*form*.]*HImage*.**SelImage**[= *setting%*]

SelLeft, SelTop Properties, HImage Control

Description

The **SelTop** and **SelLeft** properties (along with **SelWidth** and **SelHeight**) contain the dimensions and position of the current "selection box". A selection box is a rectangular area, equal to or smaller than the image. It is used to copy/paste a portion of an image to/from another control (or the Clipboard).

The **SelTop** and **SelLeft** properties define the position of the box within the control, while the **SelWidth** and **SelHeight** properties determine its size. Your program can set these properties explicitly, or use the **SelCommand's** "create" mode (2) to set them interactively at run time.

Examples

Data Type

Integer

Data Value

The **SelLeft** and **SelTop** values identify the position of the upper-left corner of the selection box with respect to the upper-left corner (position 0,0) of the entire image. For example, values of `SelLeft = 50` and `SelTop = 20` specifies position 50,20 (in other words, the 51st pixel from the left edge, and the 21st pixel from the top).

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**SelLeft**[= *setting%*]
[*form*.]*HImage*.**SelTop**[= *setting%*]

SelPicture Property, HImage Control

Description

The **SelPicture** property is used to copy a selection box between an HImage control and a Visual Basic Picture Box or the Clipboard. When copying a selection box to these objects, you must use **SelPicture** instead of **SelImage** so that the selection box is copied in a compatible format (the data are copied in Visual Basic Picture format). The example below shows how the **SelPicture** property would be used to copy image data to a Picture Box control.

```
Picture1.Picture = HImage1.SelPicture ' Copy sel box to Picture
```

Note - the **SelhDIB** property can also be used to copy data between the Clipboard and an HImage control. It copies the data in DIB format.

The **SelPicture** property is available at run time, only.

Data Type

Integer

Visual Basic

[*form*.]*HImage*.**SelPicture**[= *setting*%]

Version Property, HImage Control

Description

The **Version** property holds a string that describes the HImage control version.

Version is a read-only property.

Note - use the **About** property to display HImage version information at design time. Use the **Version** property to display version information at run time.

Data Type

String

Data Value

The **Version** property contains a single string, formatted as follows: "HIC Version #.#.#". Where "#.#.#" indicates the version number associated with the control.

Visual Basic

[*form*.]*HImage*.**Version**

ViewFixedPalette Property, HImage Control

Description

The **ViewFixedPalette** property selects the display palette that is to be used when an image is displayed on an 8-bit (256 color) display device (e.g., a VGA-class monitor). This property lets you choose between using the HImage control's "fixed" palette, or a display palette specifically tailored to the image in focus.

The HImage control's "fixed" palette optimizes the display when multiple images are open. It renders all open images (even those not in focus) as close to their actual color as possible, and eliminates the obvious repainting effect that occurs when focus is transferred among them.

The image-specific palette customizes the display palette to that of the image in focus. If several images are open, they will all be rendered with the active image's palette. This can cause significant color distortion in the background images, and produce a disturbing repaint "ripple" whenever the focus is changed.

If your application involves multiple, open images, and you want to achieve the best overall display, use the fixed-palette mode. If your application does not involve simultaneous display of multiple images, or if fidelity to the active image's data is more important than the appearance of the inactive images, use the image-specific display mode.

Note - if your system is equipped with a 16- or 24-bit display device, you will not need to use "fixed" palette mode -- your display palette can easily accommodate the color requirements of multiple open images.

Data Type

Integer (Boolean)

Data Value

The **ViewFixedPalette** values are:

Setting	Description
True (-1)	(Default) Use the HImage "fixed" palette to display the image.
False (0)	Use the palette associated with the image in focus.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ViewFixedPalette**[= {**True** | **False**}]

ViewPosX, ViewPosY Properties, HImage Control

Description

These properties contain the coordinates of the image pixel situated in the upper-left corner of the control. By changing their values, you can reposition an image within the control (when the **ViewPosX** or **ViewPosY** value is changed, the specified pixel is moved to the upper-left corner of the control).

Important - these properties will only affect an image's position when the **AutoSize** is set to 0 (none); they are ignored is the image is fit to the control, or the control is fit to an image.

Hint - consider using the **ViewScrollBars** property to let your users easily reposition ("pan") an oversized image. When an image is repositioned using the scroll bars, the **ViewPosX** and **ViewPosY** properties are automatically set by the control itself.

Data Type

Integer

Data Value

The **ViewPosX** and **ViewPosY** values represent the X and Y coordinates of the image pixel occupying the upper-left corner of the control. For example, `ViewPosX = 20` and `ViewPosY = 50` would move the image such that pixel 20,50 occupied the upper-left corner of the control.

When an image is loaded into an HImage control, the **ViewPosX** and **ViewPosY** properties are automatically initialized to 0.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ViewPosX**[= *setting%*]
[*form*.]*HImage*.**ViewPosY**[= *setting%*]

ViewScrollBars Property, HImage Control

Description

The **ViewScrollBars** property is used to enable or disable the scroll bars. The scroll bars can be used to pan an oversized (or magnified) image within the control.

Note - scroll bars will only be displayed when the **AutoSize** property is set to 0 (none); they will not appear when the image is fit to the control, or the control is fit to an image.

Data Type

Integer (Boolean)

Data Value

The **ViewScrollBars** property settings are:

Setting	Description
False (0)	(Default) Do not display scroll bars.
True (-1)	Display scroll bars when **AutoSize** property is set to 0.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ViewScrollBars**[= {**True** | **False**}]

ViewZoom Property, HImage Control

Description

The **ViewZoom** property specifies the zoom factor at which an image is displayed. By changing its value, you can magnify or reduce the image -- when the **ViewZoom** value is changed, the image is automatically displayed at the specified size.

Important - the **ViewZoom** property can only be set when **AutoSize** is set to 0 (none) or 2 (fit to image). This property is "read-only" when **AutoSize** is set to 1 (fit image to control).

Note - this property only affects the size at which the image is *displayed*. It does not actually shrink or enlarge the actual bitmap. To permanently change the size (spatial resolution) of an image, use the **ResizeCommand**.

Examples

Data Type

Integer

Data Value

The **ViewZoom** value represents a "zoom factor", expressed as a percentage of the original image size. Allowed values are from 1 to 1600. A value of 50 reduces the displayed image to half its original size...a value of 200 increases it to twice its original size, and so forth.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HImage*.**ViewZoom**[= *setting%*]

Select Event, HImage Control

Description

The **Select** event occurs when the user draws a selection box using the **SelCommand's** "create" mode (2).

Visual Basic

Sub *HImage1*_**Select**([*Index* **As Integer**], *X* **As Integer**, *Y* **As Integer**, *Cx* **As Integer**, *Cy* **As Integer**)

Argument	Description
Index	Uniquely identifies a control if it is in an array.
X,Y	The coordinates of the upper-left corner of the selection box.
Cx, Cy	The width (*Cx*) and height (*Cy*) of the selection box.

Note - if your program responds to a **MouseDown** event in the HImage control, be sure that it first checks and, if necessary, disables the **SelCommand's** "create" mode (i.e., sets **SelCommand** to 0, 1 or 3).

Clear Method, HImage Control

Description

The **Clear** method deletes the image in the control and resets certain properties to their default values.

Visual Basic

[*form*.]*HImage*.**Clear**

HicGetGammaIndex Function, HImage Control

Description

The **HicGetGammaIndex** function is used to convert an actual gamma value to an index. You will need to use this function if your program displays and accepts gamma values, instead of indexes (the **LutGamma** property uses *index* values, not gamma values). An index can be converted to a gamma value using the **HicGetGammaValue** function.

Note - gamma is measured on a logarithmic scale from 0.1 to 10, where a value of 1 represents neutrality. Values less than 1 increase the contrast in the lighter regions, darkening the image overall, and values greater than 1 increase the contrast in the darker regions, brightening the image overall.

Syntax

integer **HicGetGammaIndex(***GammaValue***)**

Parameters

Name	Type	Description
GammaValue	**single**	The gamma value that is to be converted to an index.

Return Value

HicGetGammaIndex returns an index.

HicGetGammaValue Function, HImage Control

Description

The **HicGetGammaValue** function is used to convert an HImage gamma index value to an actual gamma value. You will need to use this function if your program displays and accepts gamma values, instead of indexes (the **LutGamma** property uses *index* values, not gamma values). A gamma value can be converted to an index using the **HicGetGammaIndex** function.

Note - gamma is measured on a logarithmic scale from 0.1 to 10, where a value of 1 represents neutrality. Values less than 1 increase the contrast in the lighter regions, darkening the image overall, and values greater than 1 increase the contrast in the darker regions, brightening the image overall.

Syntax

single **HicGetGammaValue(***GammaIndex***)**

Parameters

Name	Type	Description
GammaIndex	**integer**	The index that is to be converted to a gamma value.

Return Value

HicGetGammaIndex returns the gamma value in floating-point format.

integer **HicTwipsToImageCoord Function, HImage Control**

Description

The **HicTwipsToImageCoord** function is used to convert a coordinate expressed in TWIPs, to one expressed in image pixels.

Syntax

integer **HicTwipsToImageCoord(***hctl, Twips, bVert***)**

Parameters

Name	Type	Description
hctl	**HCTL**	The name of the HImage control for which the TWIPs will be converted.
Twips	**long**	The TWIPs value that is to be converted.
bVert	**Boolean**	A value of True or False that determines whether the value is a measurement of the vertical or horizontal dimension, as follows: **True -** the coordinate is for the vertical dimension. **False -** the coordinate is for the horizontal dimension.

Return Value

HicTwipsToImageCoord returns an integer that represents the specified image coordinate, in pixels.

Example

```
Sub HImage1_MouseMove(Button, Shift, X, Y)
   ImageX = HicTwipsToImageCoord(HImage1, X, False)
   ImageY = HicTwipsToImageCoord(HImage1, Y, True)
End Sub
```

HicImageCoordToTwips Function, HImage Control

Description

The **HicImageCoordToTwips** function is used to convert an image coordinate from pixels to TWIPs.

Syntax

long **HicImageCoordToTwips(***hctl, ImageCoord, bVert***)**

Parameters

Name	Type	Description
hctl	**HCTL**	The name of the HImage control for which the image coordinate will be converted.
ImageCoord	**integer**	The image coordinate, in pixels, that is to be converted to TWIPs.
bVert	**Boolean**	A value of True or False that determines whether the coordinate is a measurement of the vertical or horizontal dimension, as follows: **True -** the coordinate is for the vertical dimension. **False -** the coordinate is for the horizontal dimension.

Return Value

HicImageCoordToTwips returns a long value that represents the specified coordinate, in TWIPs.

Example

```
X = HicImageCoordToTwips(HImage1, HImage1.SelLeft, False)
Y = HicImageCoordToTwips(HImage1, HImage1.SelTop, True)
```

HImage Error Messages

ErrorCode	ErrorMessage
25000	*This ErrrorCode value indicates that no error has occurred. When this value is present, ErrorMessage will be null.*
25001	Operation canceled.
25002	Insufficient memory to complete the request.
25003	Cannot lock an image line.
25004	Cannot reallocate cache when opening an image instance.
25005	Error closing an instance.
25006	Invalid width or height parameter.
25007	Illegal line number.
25008	Invalid rectangle coordinate.
25009	Error opening disk cache.
25010	Error reading disk cache.
25011	Error writing to disk cache.
25012	Invalid image handle.
25013	Maximum number of images exceeded.
25014	Operation not valid for this image class.
25015	Internal IMCMD error.
25016	Invalid parameter passed as an argument.
25017	Unsupported feature.
25151	Too many files open at the same time.
25152	Invalid internal file i/o parameter.
25153	Unsupported file format feature.
25154	Cannot complete valid file i/o function.
25155	Insufficient memory for file i/o function.
25156	Bad image data.
25157	Invalid file header.
25158	Error opening the file.
25159	Error closing the file.
25160	Error reading the file.
25161	Error writing the file.
25162	Error seeking in the file.
25163	File not found.
25164	Unknown file format.

The Print Dialog Box

The "Print" dialog box is presented when **PrintCommand** is set to 2. This box allows the user to specify print size, position and halftone options before the image is printed.

Position Group Box

Top - this field lets the user specify the top margin on the page. This field is automatically labeled in the units (e.g., inches, pixels, centimeters) specified by **PrintUnit**. When the dialog box is closed, the **Top** value is assigned to the **PrintPosY** property.

Left - this field lets the user specify the left margin on the page. This field is automatically labeled in the units (e.g., inches, pixels, centimeters) specified by **PrintUnit**. When the dialog box is closed, the **Left** value is assigned to the **PrintPosX** property.

Size Group Box

Width/Height - these fields let the user scale the printed output to a specific width and height. If the "Allow Distortions" box is checked, each field can be modified independently. Otherwise, the "Height" and "Width" fields are linked, such that a change to one automatically generates a proportional change in the other. When the dialog box is closed, the values in the "Height" and "Width" fields are assigned to the **PrintHeight** and **PrintWidth** properties, respectively.

Allow Distortions - this check box lets you scale the "Height" and "Width" fields independently. If it is not checked, the "Height" and "Width" fields are linked -- a change in one automatically generates a proportional change in the other, which maintains the aspect ratio of the image.

Smoothing - this check box is used to enable/disable the bilinear smoothing process. When "Smoothing" is enabled, the control interpolates pixels between adjacent lines in the scaled output. This reduces the "jaggies" that often occur when an image is enlarged significantly. If this box is checked, the **PrintSmooth** property is set to True when the dialog box is closed. Otherwise, **PrintSmooth** is set to False.

Fit To Page - this button can be used to automatically scale the image to its largest possible dimensions, given the current printer page size. If the "Allow Distortions" option is checked, the image will fill the entire page from margin to margin. Otherwise, the image will be scaled to its largest possible dimensions without changing its aspect ratio.

Actual - this button can be used to return the "Width" and "Height" fields to their original, unaltered, values.

Halftone Group Box

The "Type" and "Screen" list boxes let the user select a halftone style and screen size. When the dialog box is closed, the options selected in the "Type" and "Screen" fields are assigned to the **PrintDitherType** and **PrintDitherOption** properties, respectively.

Auto-Open Mode
"Auto-Open" mode that is enabled by setting the **FileCommand** property to 1. This instructs the HImage control to automatically load an image whenever the **FileName** property is changed. When an image is opened in "auto-mode", its format is automatically detected from information in its header.

TWIP
A TWIP is the standard unit of measure that Windows applies to all output devices. There are 1440 TWIPS in an inch.

Halftone

Halftoning is a method of simulating continuous tones of gray using small patterns of black dots. Dark tones are created using many closely spaced dots, and lighter tones, with fewer dots. Halftoning is used when an image is converted to Bilevel, or when a Gray Scale or color image is printed to a black & white output device.

Threshold

Thresholding is a method of converting continuous-tone images to Bilevel by simply converting a pixel to black (0) or white (1) depending upon whether its value greater than or equal to the threshold value. In HALO Visual Image, the threshold value is 128.

Image Class
Image class is a term describing the general category to which an image belongs. HALO Visual Image considers an image to belong to one of the following 4 "classes": *Bilevel*, *Gray Scale*, *Palette* or *True Color*.

A class is characterized by the number of bits-per-pixel used to represent a pixel, and the color conventions by which this value is interpreted. For example, pixels in a Bilevel image are represented by a single bit (1 BPP), which stands for black (0) or white (1). True Color images use 24 bits-per-pixel, allowing over 16 million colors to be represented.

True Color
"True Color" refers to a class of images that use 24 bits-per-pixel to represent RGB data. The first 8 bits represent a pixel's Redness, the second 8 bits, its Greenness and the last 8 bits, its Blueness (the 24-bit segments in a True Color image are often referred to as "RGB triples" or "RGB Chunks").

This image class can represent over 16 million colors, which is virtually every color in the visible spectrum.

Palette
"Palette-class" images are color images in which the value of a pixel is represented with 8 bits-per-pixel. However, unlike other image classes, this value *does not* represent a brightness level. Instead, it represents an index to the image's "palette", which is a 256-entry table of 24-bit RGB values. The entry in the table contains the actual color associated with the pixel's value.

Gray Scale
"Gray Scale" is a class that uses 8 bits to represent a pixel's "brightness". In a Gray Scale image, a value of 0 is completely black (zero intensity) and a value of 255 is completely white. Values between 0 and 255 represent gradation of gray, where 127 is a gray halfway between black and white, (medium gray), 63 is a dark-gray, midway between medium gray and black, and so forth.

Bilevel
"Bilevel" refers to the kind of image that uses only 1 bit to represent a pixel. It is the simplest way to represent an image digitally -- each bit is either 0 (black) or 1 (white). This kind of image is often used to record simple black and white information (faxes, for example). Images of this nature are often referred to as "Line Art" or "Halftones".

MColor

MColoring is a method of converting an image to Palette class using the MColor palette. This technique, developed by Media Cybernetics, uses a fixed palette containing colors from all across the spectrum. The conversion is performed using a dithering technique that converts the pixel values into MColor palette indices.

Color Reduction (also called **Popular Color Reduction**)
Color Reduction is a method of converting an image to Palette class based on the most popular colors in the image. During color reduction, a palette is created from the specified number of most-frequently-occurring colors in the image. This procedure is often performed to reduce the amount of space needed to store a color image.

Run Length Encoding (**RLE**)
Run Length Encoding is a compression technique that encodes data "runs" (stretches of identical information).

Luminance
Luminance is the overall intensity of a pixel, irrespective of color. In a Gray Scale image, luminance is the same as the pixel's intensity. In a color image, luminance is derived from the mean value of the pixel's RGB values.

Selection Box
A selection box defines a rectangular area, equal to or smaller than the image, that is isolated so that it can be copied or manipulated separately. A selection box is defined by the **SelTop, SelLeft, SelWidth** and **SelHeight** properties, and is referenced using the **SelImage** property.

Contrast

Contrast is the degree of difference between the lightest and darkest values in an image. An image containing pixels with very similar values will have poor contrast, and will appear flat and uninteresting. When contrast is increased, the values in an image are changed so that there is a greater difference between light and dark values. Conversely, when contrast is reduced values are made more similar.

The Example Symbol

The "Example" symbol is used to indicate that sample code is available for the current topic. Click the Example "button" to access the sample code.

TWAIN
TWAIN is an industry-standard, raster-image acquisition protocol for Windows and Macintosh-based systems. With TWAIN, images can be acquired from any device having TWAIN-compliant source software (e.g., scanners and digital cameras) using any application that is TWAIN compatible.

The HAcquire control lets you develop "TWAIN-compatible" applications.

Bilevel

Gray Scale

Palette

True Color

No Compression Applied

Default Compression

RLE Compression

CCITT Modified-Huffman, 1-Dimensional Encoding
(for Bilevel, TIFF only)

CCITT Group 3 Encoding
(for Bilevel, TIFF only)

CCITT Group 4 Encoding
(for Bilevel, TIFF only)

Modified Lempel-Zif encoding
(for TIFF and GIF only)

Modified Lempel-Zif encoding w/horizontal-differencing predictor
(for 8-bit TIFF only)

JPEG Compression Quality Factor
(default is 75%)

ConvertArg2
This property specifies the screen size to be used during the halftoning process, where 0 represents the finest screen and 3 represents the coarsest screen.

PrintDitherOption
This property specifies the screen size to be used during the halftoning process, where 0 represents the finest screen and 3 represents the coarsest screen.

Angle Dot Screen

Flat Dot Screen

Angle Line Screen

Horizontal Line Screen

Vertical Line Screen

Error Diffusion Halftone

Threshold

File Formats

The following file formats are supported by *HALO Visual Image*.

- **BMP**
- **CUT**
- **EPS** (write only)
- **GIF**
- **JPEG**
- **PCX**
- **PICT**
- **Sun Rasterfile**
- **TIFF**
- **TGA**

Bitmap File Format (BMP)

Originators

Microsoft Corporation
16011 NE 36th Way, Box 97017
Redmond WA 98073-9717

IBM Corporation

Specification

Microsoft Windows 3.1
Programmer's Reference - Volume 3

OS/2 2.0 Presentation Manager
Programming Reference Volume III

HALO Device Independent Image File Format (CUT)

Originators

Media Cybernetics
8484 Georgia Avenue
Silver Spring MD 20910

Specification

HALO Professional Reference Manual

Encapsulated Postscript File Format (EPS)

Originators

Adobe Systems Inc.
1505 Charleston Rd
P.O. Box 7900
Mountain View, CA 94039-7900

Specification

V2.0

Notes

EPS files can be written, but not read by *HALO Visual Image*.

Graphics Interchange Format (GIF)

Originators

CompuServe Inc.
Graphics Technology Dept.
5000 Arlington Center Blvd
Columbus, OH 43220

Specification

GIF89a Programming Reference and Specification

JPEG File Interchange Format (JPEG)

Originators

C-Cube MicroSystems
1770 McCarthy Blvd
Milpitas, CA 95035

JPEG software is based, in part, on the work of the Independent JPEG Group.

Specification

JPEG File Interchange Format V1.02

ZSoft Image File Format (PCX)

Originators

ZSoft Corporation
450 Franklin Road, Suite 100
Marietta GA 30067

Specification

ZSoft Technical Reference Manual

Apple Macintosh PICT Format (PICT)

Originators

Apple Computers Inc.
Mail Stop 3T
20525 Mariani Ave
Cupertino, CA 95014

Specification

Inside Macintosh, Vol. 5, Chap. 4

Notes

HALO Visual Image supports reading and writing of PICT Versions 1 and 2, but reads only the bitmap (raster, not vector) data.

Sun Rasterfile Format

Specification

rasterfile.h

Tag Image File Format (TIFF)

Originators

Aldus Corporation
411 First Avenue South, Suite 200
Seattle WA 98104

Specification

Tag Image Format Specification Revision 5.0 - Final

Notes

HALO Visual Image is compatible with Revision 5 of the TIFF specification, but there can be no more than 64 tags per image file.

HALO Visual Image fully supports TIFF classes X, B, G, R and P.

Truevision Targa (TGA)

Originators

Truevision Inc.
7351 Shadeland Station
Indianapolis, IN 46256-3925

Specification

Truevision File Format Specification V2.0

HALO Gallery Control

Description

This control allows the user to create and manipulate a collection of images arranged in rows and columns. The gallery control may be bound to a field in a data control that contains either the name of an image file or an actual image bitmap.

File Name

The HALO Gallery Control is defined by file HGC.VBX. This file is located in your Windows SYSTEM directory.

Distribution Note

When you distribute applications that use the HGallery control you must include the HGallery.VBX file with it. This file should be installed in your customer's Microsoft Windows SYSTEM subdirectory (the Visual Basic Setup Kit, included with the Professional Visual Basic product, provides tools to help you write installation programs for your application). You must also distribute all of the HALO Visual Image DLLs, which are located in your Windows' SYSTEM directory.

See Also

To learn about the specific programming attributes associated with the HALO Gallery Control, see the following topics:

Properties

Events

Examples

Error Codes

Properties, HGallery

The following table lists the properties associated with the HALO Gallery Control. To learn more about a specific property, just click its name in the table.

Note - properties that are not highlighted are ones that conform to standard Visual Basic behavior. For information about these properties, refer to your Visual Basic Language Reference or your Visual Basic on-line documentation.

About	ErrorString	LargeChange	ThumbStore
AutoRedraw	FileName	Left	ThumbWidth
BackColor	FilePath	LinkItem	Title
BorderStyle	FillOrder	LinkMode	TitleAlignment
CellSpacing	FontBold	LinkTimeout	TitleStyle
CellStyle	FontItalic	LinkTopic	Top
Columns	FontName	MousePointer	Version
DataChanged	FontSize	Name	ViewColumns
DataField	FontUnderline	Parent	ViewLeftColumn
DataFormat	Height	PrintCommand	ViewRows
DataSource	HelpContextID	Rows	ViewScrollBars
DataThumbField	hWnd	SelectedColor	ViewTopIndex
DataThumbFormat	ImeMode	SmallChange	ViewTopRow
DragIcon	ImgCount	TabIndex	Visible
DragMode	ImgLastSelected	TabStop	Width
Enabled	ImgSelCount	Tag	
ErrorCode	ImgSelected	ThumbFileName	
ErrorShow	Index	ThumbHeight	

Events, HGallery

The following table lists the events associated with the HALO Gallery Control. To learn more about a specific event, just click its name below.

Note - events that are not highlighted are ones that conform to standard Visual Basic behavior. For information about these events, refer to your Visual Basic Language Reference or your Visual Basic on-line documentation.

ClickThumb	DragOver	KeyPress	MouseDown
DblClick	GotFocus	KeyUp	MouseMove
DragDrop	KeyDown	LostFocus	MouseUp

Methods, HGallery

The following table lists the methods supported by the HALO Gallery Control. To learn more about a specific event, refer to your Visual Basic Language Reference or your Visual Basic on-line documentation.

AddItem	Move	SetFocus
Clear	Refresh	ZOrder
Drag	RemoveItem	

Examples, HGallery

Click a topic to view example code for the HGallery control:

- **Creating a Gallery and a Viewer**
- **Binding HGallery to a Data Control**
- **Tagging Gallery Images**
- **Working with Thumbnails**
- **Printing**

Examples

In this document, the "Example" symbol indicates that sample code relating to the current topic is available. Click this symbol to access the example code.

(About) Property, HGallery Control

Description

The **About** property is used to display information about the HALO Gallery Control, including its version number, release date and so forth. To display this information, select the **About** property, then click the "..." button at the top of the "Properties" window. The HGallery "About" box will be displayed. Click "OK" when you are finished viewing its contents.

This property is available at design time, only.

Note - use the **About** property to display HGallery version information at design time. Use the **Version** property to display version information at run time.

AutoRedraw Property, HGallery Control

Description

This property determines whether the gallery is redrawn when its contents are modified.

When **AutoRedraw** is on, the control is automatically updated whenever the gallery contents are changed (e.g., images added, replaced or deleted).

When **AutoRedraw** is off, the display is not updated until a Refresh is performed. You might do this when your program performs a sequence of gallery operations (e.g., adding several images in sequence) and you do not want the results of the intermediate steps to appear on the screen.

Data Type

Integer (Boolean)

Data Value

The **AutoRedraw** Property settings are:

Setting	Description
True (-1)	(Default). Redraws the control when any visual aspect of the gallery is changed.
False (0)	Redraws the control only when the Refresh method is performed.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HGallery*.**AutoRedraw**[= *setting%*]

CellSpacing Property, HGallery Control

Description

The **CellSpacing** property is used to specify the amount of space that exists between image cells in the viewer.

Examples

Data Type

Integer

Data Type

The **CellSpacing** value represents the number of Twips that exist between the image cells in the viewer.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HGallery*.**CellSpacing**[= *setting%*]

CellStyle Property, HGallery Control

Description

This property determines the type of border that surrounds the individual image cells in the viewer.

Examples

Data Type

Integer (enumerated)

Data Value

The **CellStyle** Property settings are:

Setting	Description
0	(Default) **Gray 3D In.** Displays the image cell with an engraved 3-Dimensional border.
1	**Gray 3D Out.** Displays the image cell with a raised 3-Dimensional border.
2	**Simple Frame.** Displays the image cell with a single, thin-line border.
3	**None.** Displays the image cell without a border.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HGallery.***CellStyle**[= *setting%*]

Columns, Rows Properties, HGallery Control

Description

The **Columns** and **Rows** properties specify the vertical and horizontal dimensions of your gallery, respectively. Combined with the **FillOrder** property, they define the tabular structure imposed upon your linear set of images.

At run time, only one of these dimensions can be set by your program; the other is maintained by the control, and adjusts automatically to accommodate the number of images in your image set. The fixed dimension is determined by the **FillOrder** value for example, when **FillOrder** specifies "Fill Horizontally" (0), the **Columns** property must be set by your program, and the **Rows** value is maintained by the control (i.e., it is read-only). Conversely, when **FillOrder** specifies "Fill Vertically" (1), the **Rows** property must be set by your program, and the **Columns** value is read-only.

At design time, the **Rows** and **Columns** properties are read/write. At run time, one of the properties becomes read-only, as described above.

Note - do not confuse these properties with **ViewRows** and **ViewColumns**. Rows and Columns are used to define the size of the gallery. The **ViewRows** and **ViewColumns** properties are used to define the dimensions of the viewer.

Examples

Data Type

Integer

Data Value

The **Columns** and **Rows** values represent the number of image cells contained in a single column or row (respectively) of the Gallery.

These values are saved with your Visual Basic form.

Visual Basic

[*form*.]*HGallery*.**Columns**[= *setting%*]
[*form*.]*HGallery*.**Rows**[= *setting%*]

DataFormat Property, HGallery Control

Description

This property is used to specify the content of the binding field (i.e., the field specified by **DataField**) when HGallery is bound to a data control.

Examples

Data Type

Integer (enumerated)

Data Value

The **DataFormat** Property settings are:

Setting	Description
0	(Default) **File Name.** The binding field contains a file name.
1	**DIB.** The binding field contains an image bitmap in DIB (Device Independent Bitmap) format.
2	**Picture, DIB File.** The binding field contains an image bitmap in Visual Basic Picture format.
3	**Memory File.** The binding field contains an image bitmap in one of the supported file formats (e.g., JPEG, TIFF, PCX).

This value is saved with your Visual Basic form (when the control has been bound to a data control, only).

Visual Basic

[*form*.]*HGallery.***DataFormat**[= *setting%*]

DataThumbField Property, HGallery Control

Description

This property specifies the field containing a "thumbnail" of an image. This field is only used when HGallery is bound to a data control and the database contains both a primary image and a thumbnail.

When a thumbnail field is defined in a database, HGallery uses it instead of the primary image to render the image in the viewer. The use of thumbnails is optional, however, they can significantly improve the performance of a gallery containing very large images.

The content of the thumbnail field, which may be a file name or actual image data, must be specified in the **DataThumbFormat** property.

Examples

Data Type

String

Data Value

The **DataThumbField** property specifies the name of the database field containing a thumbnail of the image, if there is one. Otherwise it should be null

This value is saved with your Visual Basic form (when the control has been bound to a data control, only).

Visual Basic

[*form*.]*HGallery*.**DataThumbField**[= *setting$*]

DataThumbFormat Property, HGallery Control

Description

This property specifies the content of the thumbnail field (i.e., the field specified by **DataThumbField**) when HGallery is bound to a data control and the database contains both a primary image and a thumbnail. The thumbnail field may contain a file name, or it may contain actual image data.

A thumbnail is a reduced-size version of the primary image. When a thumbnail field is defined in a database, HGallery uses it instead of the primary image to render the image in the viewer. The use of thumbnails is optional, however, they can significantly improve the performance of a gallery containing very large images. When used, the name of the thumbnail field must be specified in **DataThumbField**.

Examples

Data Type

Integer (enumerated)

Data Value

The **DataThumbFormat** Property settings are:

Setting	Description
0	(Default) **File Name.** The thumbnail field contains a file name.
1	**DIB.** The thumbnail field contains an image bitmap in DIB (Device Independent Bitmap) format.
2	**Picture, DIB File.** The thumbnail field contains an image bitmap in Visual Basic Picture format.
3	**Memory File.** The thumbnail field contains an image bitmap in one of the supported file formats (e.g., JPEG, TIFF, PCX).

This value is saved with your Visual Basic form (when the control has been bound to a data control, only).

Visual Basic

[*form*.]*HGallery*.**DataThumbFormat**[= *setting%*]

ErrorCode Property, HGallery Control

Description

This property holds the error code returned by the last HGallery control operation. Both the **ErrorCode** and **ErrorString** properties are automatically set by HGallery when an error occurs (**ErrorString** will hold a brief description of the error). To clear the **ErrorCode** property after an error has occurred, set it to 25000.

Data Type

Integer (enumerated)

Data Value

The **ErrorCode** value is an integer representing the error generated by the last HGallery operation. If the last operation did not cause an error, this property will contain the value, 25000.

See HGallery Error Messages for a complete list of error codes and messages.

Visual Basic

[*form*.]*HGallery*.**ErrorCode**[= *setting%*]

ErrorShow Property, HGallery Control

Description

This property determines whether an error is returned to Visual Basic or is simply noted in the **ErrorCode** and **ErrorString** properties.

See HGallery Error Messages for a complete list of error codes and messages.

Data Type

Integer (Boolean)

Data Value

The **ErrorShow** Property settings are:

Setting	Description
True (-1)	(Default). Specifies that the control is to immediately return an error code and error message to Visual Basic, when an error occurs. The code and message is contained in the **ErrorCode** and **ErrorString** properties, respectively.
False (0)	Specifies that the control is to set the **ErrorCode** and **ErrorString** properties, but is not to return an error code to Visual Basic. This setting lets you decide whether (and how) your program will respond to HGallery errors when they occur.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HGallery.***ErrorShow**[{**True** | **False**}]

ErrorString Property, HGallery Control

Description

This property contains a string describing the error associated with the error code contained in the **ErrorCode** property. The **ErrorString** and **ErrorCode** properties are automatically written by the HGallery Control whenever an error occurs within it.

ErrorString is a read-only property.

See HGallery Error Messages for a complete list of error codes and messages.

Data Type

String

Visual Basic

[*form*.]*HGallery.***ErrorString**

FileName Property, HGallery Control

Description

The **FileName** property is used to specify the file names of the images comprising your gallery. This property *must* be set when the HGallery control is not bound to a database.

FileName is a one-dimensional property array, in which each entry specifies the name of a file. The example below shows how the **FileName** property would be set to define a gallery containing two images.

```
HGallery1.FileName(0) = "C:\images\balloon.tif"   ' Assigns file name
HGallery1.FileName(1) = "C:\images\baseball.bmp"  ' Assigns file name
```

When an element in the **FileName** array is changed, the specified file is opened (its format is automatically discerned from its header information), and a thumbnail of the image is created and loaded into the control.

This property is available at run time, only.

Data Type

String (array)

Data Value

Each element in the **FileName** array contains a string that specifies the name of an image file. This property is used in unbound mode only.

Visual Basic

[*form*.]*HGallery*.**FileName(***index***)**[= *setting$*]

FilePath Property, HGallery Control

Description

The **FilePath** property is used to specify the path of the files in the **FileName** and **ThumbFileName** arrays. It is only used when a string in one of these arrays contains *no* path information.

If you want **FilePath** to be used, be sure that your file name string contains *nothing but the file name* (e.g., "filename.bmp"); the presence of *any* path information will cause **FilePath** to be ignored. The following strings are all examples of names that contain path information:

```
HGallery1(0).FileName = "C:filename.bmp"
HGallery1(0).FileName = "IMG\filename.bmp"
HGallery1(0).FileName = "\filename.bmp"
HGallery1(0).FileName = "C:\IMG\filename.bmp".
```

FilePath is used when reading *and* storing an image file.

Data Type

String

Data Value

The **FilePath** value is a string that specifies the path to the image file. The following example shows how the **FilePath** property would be set if the image files were stored in D:\images (notice that the path *does not* include the backslash that usually delimits the subdirectory from the file name).

```
HGallery1.FilePath = "D:\images"        ' Sets path
HGallery1.FileName(0) = "balloon.tif"   ' Assigns files name
HGallery1.FileName(1) = "baseball.bmp"  ' Assigns file name
```

The following strings are also valid **FilePath** settings:

```
HGallery1.FilePath = "D:"          ' Specifies the current
                                   ' directory on the D: drive.
HGallery1.FilePath = "D:\"         ' Specifies the root directory
                                   ' on the D: drive.
HGallery1.FilePath = "IMG"         ' Specifies the IMG subdirectory
                                   ' within the current directory
                                   ' on the current drive.
HGallery1.FilePath = "\IMG"        ' Specifies the \IMG directory
                                   ' on the current drive.
```

Visual Basic

[*form.*]*HGallery.***FilePath**[= *setting$*]

FillOrder Property, HGallery Control

Description

FillOrder determines the order in which images are arranged in the gallery. Images can be loaded from left to right (filled horizontally), or from top to bottom (filled vertically). Combined with the **Columns** and **Rows** properties, this property defines the tabular structure imposed upon your linear image set.

Examples

Data Type

Integer (enumerated)

Data Value

The **FillOrder** Property settings are:

Setting	Description
0	(Default) **Fill Horizontally.** The image cells are filled left-to-right, one row at a time (i.e., when one row becomes full, the control moves to the next row). When this value is used, your program must fix the horizontal dimension of the gallery by setting the **Columns** property. HGallery will automatically adjust the **Rows** value to accommodate the number of images in your image set.
1	**Fill Vertically.** The image cells are filled top-to-bottom, one column at a time (i.e., when one column becomes full, the control moves to the next column). When this value is used, your program must fix the vertical dimension of the gallery by setting the **Rows** property. HGallery will automatically adjust the **Columns** value to accommodate the number of images in your image set.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HGallery.***FillOrder**[= *setting%*]

ImgCount Property, HGallery Control

Description

This property identifies the number of images currently in the Gallery. If the control is bound to a data control, it represents the total number of images in the database. Otherwise, it indicates the number of elements in the **FileName** array.

This property is only available at run time. This property is read-only.

Examples

Data Type

Integer

Data Value

The **ImgCount** values identifies the number of images in the gallery.

Visual Basic

[*form*.]*HGallery*.**ImgCount**[= *setting%*]

ImgLastSelected Property, HGallery Control

Description

The **ImgLastSelected** property identifies the image that was last selected with the **ImgSelected** property, or last clicked by the user.

This property is available at run time, only. This property is read-only.

Data Type

Integer

Data Value

The **ImgLastSelected** value represents a gallery index, where 0 represents the first image in the gallery, 1 represents the second image, and so forth.

Visual Basic

[*form*.]*HGallery1*.**ImgLastSelected**[= *setting%*]

ImgSelCount Property, HGallery Control

Description

This property contains the total number of "tagged" thumbnails in the gallery. Multiple images can be "tagged" for batch processing by "selecting" them with the **ImgSelected** property.

This property is available at run time, only. This property is read-only.

Data Type

Integer

Visual Basic

[*form*.]HGallery.**ImgSelCount**

ImgSelected Property, HGallery Control

Description

The **ImgSelected** property is used to select and deselect images in the gallery. **ImgSelected** is an one-dimensional property array, in which each entry contains a value of 0 or 1, indicating whether an image is "deselected" or "selected" (respectively). You can use the **ImgSelected** property to "tag" specific images for special processing -- for example, you might tag a group of images so that they can be deleted from the gallery or copied to another directory.

This property is only available at run time.

Examples

Data Type

Integer (array)

Data Value

Each entry in the **ImgSelected** array contains an integer value of 1 or 0, indicating whether the corresponding image is selected (1) or not (0). In the array, each element corresponds to a gallery index, i.e., **ImgSelected(0)** represents the first image in the gallery, **ImgSelected(1)** represents the second image, and so forth.

Visual Basic

[*form*.]HGallery.**ImgSelected(***index***)**[= *setting%*]

PrintCommand Property, HGallery Control

Description

The **PrintCommand** property is used to print a gallery. When this property is set to 1, all thumbnails associated with the HGallery control will be printed in "contact-sheet" format (i.e., thumbnails arranged in rows and columns on a printed page).

Note - printing a gallery via the **PrintCommand** property is just one way a gallery can be printed. You can also use Visual Basic's standard **PrintForm** method to print the entire form.

Examples

Data Type

Integer (enumerated)

Data Value

The **PrintCommand** Property settings are:

Setting	Description
0	(Default). **None.** Does nothing.
1	**Print.** Prints the gallery.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**PrintCommand**[= *setting%*]

SelectedColor Property, HGallery Control

Description

This property is used to specify the color of the frame drawn around an image cell containing a "tagged" thumbnail (images are "tagged" with the **ImgSelected** property).

Data Type

4-byte Integer

Data Value

The **SelectedColor** value represents the frame color in standard Visual Basic form (i.e., a 4-byte integer value). Your program can set this value directly, using the hexadecimal syntax: `&HBBGGRR&` where BB, GG and RR specify the amount of Blue, Green and Red, respectively. For example, the following statement would set the frame color to pure green:

```
HGallery1.SelectedColor = &H00FF00&
```

You can also assign color using an indirect method, such as Visual Basic's RGB function. This function lets you specify the color value by supplying the Red, Green and Blue values using a decimal scale of 0 to 255, as shown in the following statement.

```
HGallery1.SelectedColor = RGB(0, 255, 0)
```

For more information about setting color, see *Using Simple Color* and *Using Color Properties* in your *Visual Basic Programmer's Guide*.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**SelectedColor**[= *setting%*]

ThumbFileName Property, HGallery Control

Description

The **ThumbFileName** property is used to specify the file names of the "thumbnails" associated with your gallery. This property is used when a gallery has thumbnail images associated with it, and it is not bound to a database (see **DataThumbField** for using thumbnails in bound mode).

ThumbFileName is a one-dimensional property array, in which each entry specifies the name of an image file. When a thumbnail file name is defined by this property, HGallery uses it instead of the primary image to render the image in the viewer. The use of thumbnails is optional, however, they can significantly enhance the performance of a gallery containing very large images.

The example below shows how the **ThumbFileName** property would be set to define a gallery containing two images.

```
HGallery1.ThumbFileName(0) = "C:\thumbs\balloon.TIF"
HGallery1.ThumbFileName(1) = "C:\thumbs\baseball.bmp"
```

This property is available at run time, only.

Examples

Data Type

String (array)

Data Value

Each element in the **ThumbFileName** array contains a string that specifies the name of the image file that is to be used to render the image in the viewer.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**ThumbFileName(***index***)**[= *setting$*]

ThumbHeight, ThumbWidth Properties, HGallery Control

Description

The **ThumbHeight** and **ThumbWidth** properties specify the size of the thumbnail area in the viewer (note that this is the size of the *thumbnail*, not the *image cell).* They represent the dimensions to which the image will be reduced for display purposes.

Examples

Data Type

Integer

Data Value

The **ThumbHeight** and **ThumbWidth** values represent the cell height and width (respectively), in Twips.

These values are saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**ThumbHeight**[= *setting%*]
[*form*.]HGallery.**ThumbWidth**[= *setting%*]

ThumbStore Property, HGallery Control

Description

The **ThumbStore** property determines whether the gallery thumbnail is stored to the thumbnail field when a record is updated or added to the database to which HGallery is bound.

Data Type

Integer

Data Value

The **ThumbStore** values are:

Setting	Description
0	(Default) **None.** Do not store the thumbnail to the database.
1	**16 Color.** Store the gallery thumbnail to the database in 4-bit (16 color) DIB format. When this option is used, **DataThumbField** must specify a valid blob field, and **DataThumbFormat** must specify **DIB** format.
2	**256 Color.** Store the gallery thumbnail to the database in 8-bit (256 color) DIB format. When this option is used, **DataThumbField** must specify a valid blob field, and **DataThumbFormat** must specify **DIB** format.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**ThumbStore**[= *setting%*]

Title Property, HGallery Control

Description

The **Title** property contains the user-defined image "titles", which are used when the **TitleStyle** property is set to 2 (User Defined). **Title** is a one-dimensional property array, in which each entry specifies the title associated with the corresponding image index. This title is displayed along with the thumbnail in the viewer.

This property is available at run time, only.

Data Type

String (array)

Data Value

Each element in the **Title** array contains a string that specifies the name, or "title" of the image. In the array, each element corresponds to a gallery index, i.e., **Title(0)** contains the title for the first thumbnail in the gallery, **Title(1)** contains the title for the second thumbnail, and so forth.

Visual Basic

[*form*.]HGallery.**Title(***index***)**[= *setting$*]

TitleAlignment Property, HGallery Control

Description

The **TitleAlignment** property is used to position the image "title", which is displayed along with the thumbnail in the viewer. The title, which may be a file name, an index number or a user-defined string, is determined by the **TitleStyle** property setting.

Examples

Data Type

Integer (enumerated)

Data Value

Setting	Description
0	**Bottom Left Justified.** Displays the title below the thumbnail, flush with the left edge of the image cell.
1	**Bottom Centered.** Displays the title below the thumbnail, centered in the image cell.
2	**Bottom Right Justified.** Displays the title below the thumbnail, flush with the right edge of the image cell.
3	**Top Left Justified.** Displays the title above the thumbnail, flush with the left edge of the image cell.
4	**Top Centered.** Displays the title above the thumbnail, centered in the image cell.
5	**Top Right Justified.** Displays the title above the thumbnail, flush with the right edge of the image cell.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**TitleAlignment**[= *setting%*]

TitleStyle, HGallery Control

Description

The **TitleStyle** property is used to specify the kind of information that is to be displayed in the title line of the image cell. You may choose to display the image's file name, its index number or a string of your own design.

Examples

Data Type

Integer (enumerated)

Data Value

Setting	Description
0	(Default) **File Name.** The image's file name (as defined by the **FileName** property array) will be displayed in the title line.
1	**Indexes.** The image's gallery index will be displayed in the title line.
2	**User Defined.** The image's user-defined title (as defined by the **Title** property array) will be displayed in the title line.
3	**None.** The title line will be empty.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**TitleStyle**[= *setting%*]

Version Property, HGallery Control

Description

The **Version** property holds a string that describes the version of the HGallery control.

Version is a read-only property.

Note - use the **About** property to display HGallery version information at design time. Use the **Version** property to view display information at run time.

Data Type

String

Data Value

The **Version** property contains a single string, formatted as follows: "HGC Version #.#.#". Where "#.#.#" indicates the version number associated with the control.

Visual Basic

[*form*.]HGallery.**Version**

ViewColumns, ViewRows Properties, HGallery Control

Description

The **ViewColumns** and **ViewRows** properties are used to define the size of the viewer. They represent the number of image cells contained in its vertical and horizontal dimensions, respectively.

Note - do not confuse these properties with **Rows** and **Columns**. **ViewRows** and **ViewColumns** are used to define the size of the *viewer*. The **Rows** and **Columns** properties are used to define the dimensions of the *entire gallery*.

Examples

Data Type

Integer

Data Value

The **ViewColumns** and **ViewRows** values represent the number of image cells contained in a single column or row (respectively) of the viewer.

These values are saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**ViewColumns**[= *setting%*]
[*form*.]HGallery.**ViewRows**[= *setting%*]

ViewLeftColumn, ViewTopRow Properties, HGallery Control

Description

The **ViewLeftColumn** and **ViewTopRow** properties specify the position of the viewer within the gallery. Their values represent the gallery coordinates of the upper-left cell in the viewer.

You may use either the **ViewLeftColumn**/**ViewTopRow** or **ViewTopIndex** properties to shift the viewer over the gallery. These properties are all linked, such that a change to one is automatically reflected in the other. For example, when the **ViewLeftColumn** or **ViewTopRow** value is changed, the **ViewTopIndex** automatically changes to reflect the index of the image located in the specified cell. Similarly, when **ViewTopIndex** is changed, **ViewLeftColumn** and **ViewTopRow** are updated to reflect the specified image's cell position in the gallery.

Data Type

Integer

Data Value

The **ViewLeftColumn** and **ViewTopRow** values represent the coordinates of the gallery cell over which the upper-left cell of the viewer is positioned (0,0 represents the position of the very first image in the galley). For example, if the viewer were to be positioned over the second image in the third row of the gallery, the **ViewLeftColumn** and **ViewTopRow** properties would be set as follows:

```
HGallery1.ViewLeftColumn = 1         ' Position left edge of viewer
HGallery1.ViewTopRow = 2             ' Position top of viewer
```

The **ViewLeftColumn** and **ViewTopRow** values are not saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**ViewLeftColumn**[= *setting%*]
[*form*.]HGallery.**ViewTopRow**[= *setting%*]

ViewScrollBars Property, HGallery Control

Description

The **ViewScrollBars** property is used to display a scroll bar(s) with the viewer. A scroll bar can be used pan the viewer over the gallery.

When scroll bars are enabled, they are added to the outside of the gallery viewer.

Note - the Pg Up and Pg Dn keys can also be used to scroll through a gallery.

Data Type

Integer (enumerated)

Data Value

The **ViewScrollBars** Property settings are:

Setting	Description
0	(Default) **None.** Do not display scroll bars with the viewer.
1	**Horizontal.** Display a scroll bar along the bottom edge of the viewer.
2	**Vertical.** Display a scroll bar along the right edge of the viewer.
3	**Both.** Display a scroll bars along the bottom and right edges of the viewer.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HGallery*.**ViewScrollBars**[= *setting%*]

ViewTopIndex Property, HGallery Control

Description

The **ViewTopIndex** property specifies the position the viewer within the gallery. Its value represents the index of the image in the upper-left cell of the viewer. You might set this value to jump to a specific image in the gallery, as shown in the example below.

```
Sub Button14_Click ()            ' Jump to first name starting with N
   For i = 0 to ImgCount
      If Left$(HGallery1.FileName(i),1) >= "N" Then
         HGallery1.ViewTopIndex = i
         Exit For
      EndIf
   Next i
End Sub
```

Your program can also query this property to determine which image is currently located in the upper-left position of the control.

Note - You may use either the **ViewLeftColumn**/**ViewTopRow** or **ViewTopIndex** properties to shift the viewer over the gallery. These properties are all linked, such that a change to one is automatically reflected in the other. For example, when the **ViewLeftColumn** or **ViewTopRow** value is changed, the **ViewTopIndex** automatically changes to reflect the index of the image located in the specified cell. Similarly, when **ViewTopIndex** is changed, **ViewLeftColumn** and **ViewTopRow** are updated to reflect the specified image's cell position in the gallery.

The **ViewTopIndex** property is available at run time, only.

Data Type

Integer

Data Value

The **ViewTopIndex** value represents the index of the image over which the upper-left cell of the viewer is positioned. For example, if the viewer were to be positioned over the tenth image in the gallery, the **ViewTopIndex** property would be set as follows:

```
HGallery1.ViewTopIndex = 9          ' Move to image 9
```

The **ViewTopIndex** value is not saved with your Visual Basic form.

Visual Basic

[*form*.]HGallery.**ViewTopIndex**[= *setting%*]

ClickThumb Event, HGallery Control

Description

The **ClickThumb** event occurs when the user presses then releases the left mouse button over an image cell in the HGallery control. The event does not occur if the cursor is not within an image cell when the mouse is clicked (e.g. , if it is between cells).

This event sets a variable called CellIndex to the index of the image on which the user clicked. It also updates the **ImgSelected** property array as follows:

- If the user clicks an untagged image, it will be tagged. Any previously tagged images will be untagged.
- If the user clicks a tagged image, it will be untagged. Any previously tagged images will be untagged.
- If the user shift-clicks an untagged image, it will be tagged. Any previously tagged images will remain tagged.
- If the user shift-clicks an tagged image, it will be untagged. Any previously tagged images will remain tagged.

Examples

Visual Basic

Sub *HGallery*_**ClickThumb**([*Index* **As Integer**], *CellIndex* **As Integer**)

Argument	Description
Index	The index of the control if it is part of an array.
CellIndex	The index of clicked image.

Data Control Example, HGallery Control

The following example shows the properties that would be set to bind a HGallery control to a database that references its images by file name. In this example, the control is bound to the Data1 control, which references its images via the ImageFileName field.

```
HGallery1.DataSource = Data1                ' Data control name
HGallery1.DataFormat = 0                    ' Field contains file name
HGallery1.DataField = ImageFileName         ' Field with file name
```

The following example shows the properties that would be set to bind a HGallery control to a database containing embedded images. The control is bound to the Data1 control, which embeds its images in field ImageData.

```
HGallery1.DataSource = Data1                ' Data control name
HGallery1.DataFormat = 1                    ' Field contains DIB data
HGallery1.DataField = "ImageData"           ' Field with image
```

Print Example, HGallery Control

There are two ways of printing a HALO Gallery Control:

Visual Basic PrintForm

The standard Visual Basic **PrintForm** method sends the content of the entire form to the printer. This method can be used when you want to print just the image cells currently visible within the control

```
Sub mnuPrintForm_Click ()
   PrintForm                        ' Print the entire form
End Sub
```

HGallery PrintCommand Property

Using the HGallery **PrintCommand** property you can print the entire gallery in "contact-sheet" format (i.e., thumbnails arranged in rows and columns on a printed page).

```
HGallery1.PrintCommand = 1          ' Print entire gallery
```

Tagging Gallery Images, HGallery Control

Gallery images can be "tagged" for special processing via the **ClickThumb** event, which is invoked when the user clicks an image cell in the viewer. In the following example, the selected image is loaded into an HImage control.

```
Sub HGallery1_ClickThumb (CellIndex As Integer)
   HImage1.FileName = HGallery1.FileName(CellIndex)
End Sub
```

You can also use **ClickThumb** to tag multiple images, then apply a batch process to all the tagged images. In the following example the **ImgSelected** property array is interrogated for "tagged" images and a special process is applied to those that are found.

```
Sub Command1_Click ()
   For i = 0 to HGallery1.ImgCount
      If HGallery1.ImgSelected(i) = 1 Then
         YourSpecialProcess(HGallery1.FileName(i))
      EndIf
   Next
End Sub
```

Gallery and Viewer Size Example, HGallery Control

In this example, a 4-column gallery is created. A 4 x 2 image viewer is defined containing 1.5" x 1" thumbnails, in 3D image cells. The image index will be displayed above the thumbnail.

```
HGallery1.FillOrder = 0             ' Fill gallery row-by-row
HGallery1.Columns = 4               ' Create 4 images per row
HGallery1.ViewColumns = 4           ' Create viewer 4 images across
HGallery1.ViewRows = 2              ' Create viewer 2 images high
HGallery1.ThumbWidth = 2160         ' Display image 1.5" across
HGallery1.ThumbHeight = 1440        ' Display image 1.0" across
HGallery1.CellSpacing = 140         ' Space cells 1/10" apart
HGallery1.CellStyle = 1             ' Use Raised 3D Border
HGallery1.TitleStyle = 1            ' Display index number w/thumbnail
HGallery1.TitleAlignment = 4        ' Center title at top of cell
```

Thumbnail Example, HGallery Control

If your image set is defined in the FileName property array (i.e., you are using HGallery in "unbound" mode), you can specify a thumbnail for each image by specifying the name of the file containing the thumbnail in the corresponding element of the ThumbFileName property array. The example below shows how a gallery of four 4 images would be rendered from their thumbnail files.

```
HGallery1.AutoRedraw = False         ' Turn off screen paint
                                     ' Load image set
HGallery1.FileName(0) = c:\parts\wheel.tif
HGallery1.FileName(1) = c:\parts\gear.tif
HGallery1.FileName(2) = c:\parts\hinge.tif
HGallery1.FileName(3) = c:\parts\spring.tif
                                     ' Load image thumbnails
HGallery1.ThumbFileName(0) = c:\parts\thumb\wheel.tif
HGallery1.ThumbFileName(1) = c:\parts\thumb\gear.tif
HGallery1.ThumbFileName(2) = c:\parts\thumb\hinge.tif
HGallery1.ThumbFileName(3) = c:\parts\thumb\spring.tif

HGallery1.Refresh                    ' Update control
HGallery1.AutoRedraw = True          ' Turn on screen paint
```

The following example illustrates how a thumbnail would be defined in a HGallery control, bound to a database containing embedded image thumbnails in DIB format.

```
HGallery1.DataSource = Data1           ' Data control name
HGallery1.DataFormat = 1               ' Field has DIB image
HGallery1.DataField = "DIBPic"         ' Name of image field
HGallery1.DataThumbFormat = 1          ' Thumb field has DIB image
HGallery1.DataThumbField = "tDIBPic"   ' Name of thumbnail field
```

HGallery Error Messages

ErrorCode	ErrorMessage
25000	*This ErrrorCode value indicates that no error has occurred. When this value is present, ErrorMessage will be null.*
25001	Operation canceled.
25002	Insufficient memory to complete the request.
25003	Cannot lock an image line.
25004	Cannot reallocate cache when opening an image instance.
25005	Error closing an instance.
25006	Invalid width or height parameter.
25007	Illegal line number.
25008	Invalid rectangle coordinate.
25009	Error opening disk cache.
25010	Error reading disk cache.
25011	Error writing to disk cache.
25012	Invalid image handle.
25013	Maximum number of images exceeded.
25014	Operation not valid for this image class.
25015	Internal IMCMD error.
25016	Invalid parameter passed as an argument.
25017	Unsupported feature.
25151	Too many files open at the same time.
25152	Invalid internal file i/o parameter.
25153	Unsupported file format feature.
25154	Cannot complete valid file i/o function.
25155	Insufficient memory for file i/o function.
25156	Bad image data.
25157	Invalid file header.
25158	Error opening the file.
25159	Error closing the file.
25160	Error reading the file.
25161	Error writing the file.
25162	Error seeking in the file.
25163	File not found.
25164	Unknown file format.

ThumbBlob Property, HGallery Control

Description

ThumbBlob contains the handles to each of the thumbnails in the gallery. You might use this property to ??

This property is available at run time, only.

Data Type

Integer (array)

Data Value

ThumbBlob is a property array. Each element in the array contains the handle to the corresponding thumbnail's bitmap. In the array, each element corresponds to a gallery index, i.e., **ThumbBlob(0)** contains the handle for the first thumbnail in the gallery, **ThumbBlob(1)** contains the handle for the second thumbnail, and so forth.

Visual Basic

[*form*.]HGallery.**ThumbBlob**[= *setting%*]

HALO Acquisition Control (HAcquire)

Description

This control allows you to create applications capable of acquiring images from any TWAIN-compliant source device (e.g. scanner, digital camera or frame-grabber). It lets you invoke the TWAIN device, receive the image data it produces and place it in a file, an HImage control or the Clipboard.

File Name

The HALO Acquisition Control is defined by file HAC.VBX. This file is located in your Windows SYSTEM directory.

Distribution Note

When you distribute applications that use the HAcquire control you must include the HAC.VBX file with it. This file should be installed in your user's Microsoft Windows SYSTEM subdirectory (the Visual Basic Setup Kit, included with the Professional Visual Basic product, provides tools to help you write installation programs for your application). You must also distribute all of the HALO Visual Image DLLs, which are located in your Windows' SYSTEM directory.

See Also

To learn about the specific programming attributes associated with the HALO Acquisition Control, see the following topics:

Properties

Events

Examples

Error Codes

Properties, HAcquire

The following table lists the properties associated with the HALO Acquisition Control. To learn more about a specific property, just click its name in the table.

Note - properties that are not highlighted are ones that conform to standard Visual Basic behavior. For information about these properties, refer to your Visual Basic Language Reference or your Visual Basic on-line documentation.

About	FileCompression	SelLeft
AppFamily	FileFormat	SelTop
AppMfr	FileJPEGQuality	SelWidth
AppName	Height	Source
Command	ImageType	SourceCount
Count	Index	SourceNames
DestName	Left	Tag
DestType	Name	Top
ErrorCode	Parent	Unit
ErrorShow	Resolution	Version
ErrorString	SelHeight	Width

Events, HAcquire

The following table lists the event associated with the HALO Acquisition Control. To learn more about this event, just click its name below.

Acquire

Examples, HAcquire

Click a topic to view example code for the HAcquire control:

- **Acquiring An Image**
- **Acquiring Multiple Images**

Examples

In this document, the "Example" symbol indicates that sample code relating to the current topic is available. Click this symbol to access the example code.

(About) Property, HAcquire Control

Description

The **About** property is used to display information about the HALO Acquire Control, including its version number, release date, and so forth. To display this information, select the **About** property, then click the "..." button at the top of the "Properties" window. The HAcquire "About" box will be displayed. Click "OK" when you are finished viewing its contents.

This property is available at design time, only.

Note - use the **About** property to display HAcquire version information at design time. Use the **Version** property to display version information at run time.

AppFamily Property, HAcquire Control

Description

The **AppFamily** property is used to identify the product-family with which your application is associated. This string is incorporated into the dialog boxes of some TWAIN devices (not all TWAIN drivers use it).

The **AppFamily** property is one of three product-identifying strings defined by the TWAIN specification. A TWAIN device driver may use all, one or none of these strings in their dialogs.

Property	Description	Example
AppMfr	Specifies the application manufacturer	"SoftABC Corp."
AppFamily	Specifies a product division	"Optical Products Division"
AppName	Specifies the product name	"ABC QuickScan"

Although the use of this property is optional, we recommend that it be set when you are using the TWAIN user interface. That way your program will be properly identified by any TWAIN device that displays such information.

Data Type

String

Data Value

The **AppFamily** value is a string identifying the family or division to which your application belongs. If this property is not set by your program, the default string, "HALO Visual Image", will be used.

The **AppFamily** property value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**AppFamily**[= *setting$*]

AppMfr Property, HAcquire Control

Description

The **AppMfr** property is used to identify the name of your company. This string is incorporated into the dialog boxes of some TWAIN devices (not all TWAIN drivers use it).

The **AppMfr** property is one of three product-identifying strings defined by the TWAIN specification. A TWAIN device driver may use all, one or none of these strings in their dialogs.

Property	Description	Example
AppMfr	Specifies the application manufacturer	"SoftABC Corp."
AppFamily	Specifies a product division	"Optical Products Division"
AppName	Specifies the product name	"ABC QuickScan"

Although the use of this property is optional, we recommend that it be set when you are using the TWAIN user interface. That way your program will be properly identified by any TWAIN device that displays such information.

Data Type

String

Data Value

The **AppMfr** value is a string identifying the manufacturer of your application. If this property is not set by your program, the default string, "Media Cybernetics, Inc.", will be used.

The **AppMfr** property value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**AppMfr**[= *setting$*]

AppName Property, HAcquire Control

Description

The **AppName** property is used to identify the name of your application. This string is incorporated into the dialog boxes of some TWAIN devices (not all TWAIN drivers use it).

The **AppName** property is one of three product-identifying strings defined by the TWAIN specification. A TWAIN device driver may use all, one or none of these strings in their dialogs.

Property	Description	Example
AppMfr	Specifies the application manufacturer	"SoftABC Corp."
AppFamily	Specifies a product division	"Optical Products Division"
AppName	Specifies the product name	"ABC QuickScan"

Although the use of this property is optional, we recommend that it be set when you are using the TWAIN user interface. That way your program will be properly identified by any TWAIN device that displays such information.

Data Type

String

Data Value

The **AppName** value is a string identifying the name of your application. If this property is not set by your program, the default string, "HALO Acquisition Control", will be used.

The **AppName** property value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**AppName**[= *setting$*]

Command Property, HAcquire Control

Description

The **Command** property is used to select a TWAIN device and perform acquisition operations.

Example

Data Type

Integer (enumerated)

Data Value

The **Command** property settings are:

Setting	Description
0	(Default). **None.** Does nothing.
1	**Select.** Displays the TWAIN "Select Source" dialog box, which allows the user to select a TWAIN source device. When the user closes the dialog box, the name of the selected device is written to the **Source** property.
2	**Acquire.** Invokes the user interface for the selected TWAIN device. Within this interface, the user can select operating parameters and initiate the acquisition process.
3	**Acquire Direct.** Performs a direct acquisition using the parameters set in the **ImageType**, **Count**, **Resolution**, **SelLeft**, **SelTop**, **SelWidth**, and **SelHeight**, properties.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**Command**[= *setting%*]

Count Property, HAcquire Control

Description

The **Count** property specifies the number of images to be captured by a single acquisition operation. By setting **Count** to a value greater than 1, multiple images can be acquired from a device (such as from a scanner with an Automatic Document Feeder).

Example

Important - when **Count** is set to a value greater than 1, your acquisition output *must* be directed to "File Name" (i.e., **DestType** must be set to 1). Multiple images *cannot* be stored to an HImage control or the Clipboard.

Data Type

Integer

Data Value

The **Count** value specifies the number of images to be captured, as follows:

Setting	Description
-1	Capture as many images as the user wants. When this option is used, the user must close the TWAIN dialog when he/she is finished acquiring images.
1	Capture a single image. When the TWAIN user interface is utilized, the TWAIN dialog will automatically close after a single image has been acquired.
any value >1	Capture a single image. When the TWAIN user interface is utilized, the TWAIN dialog will automatically close after the specifies number of images have been acquired.

Visual Basic

[*form*.]*HAcquire*.**Count**[= *setting%*]

DestName Property, HAcquire Control

Description

The **DestName** property specifies the name of the image file or HImage control to which the acquired image will be written.

Data Type

String

Example

Data Value

The string in the **DestName** property specifies the name of the destination file or control to which the acquired data will be written when it is returned by the TWAIN source driver.

- **If the image is directed to a file** (**DestType** = 1), the contents of **DestName** must specify a file name. When multiple files are acquired, this string must specify the pattern to be used for creating their file names. The pattern must be composed of a alphanumeric "prefix" (of 1 to 7 characters) followed by one or more # symbols, which denote the number of positions to be used for digits . For example:

```
IMG####.TIF
```

  generates files: IMG0000.TIF, IMG0001.TIF, IMG0002.TIF...

  You must be sure to allow sufficient digit positions for the number of images that will be acquired (e.g., if twelve images were to be stored, you must define at least 2 digit positions). Also, you must ensure that the prefix and digits combined, do not exceed 8 characters.

  If **DestName** does not include an extension, the file will be stored without an extension; *HAcquire* will not assign default extensions to the files it creates.

- **If the image is directed to a HALO Image Control** (**DestType** = 2), the contents of **DestName** must specify the name of the HImage control to which the acquired image will be written.

```
HAcquire1.DestName = "HImage1"        ' Route output to HImage1
```

  In this example, the acquired output would be written to HImage1 control (the image is copied to the virtual image, i.e., HImage1.Image). The HImage control you specify in **DestName** *must* exist in the same Visual Basic form as the HAcquire control.

- **If the image is written to the Clipboard** (**DestType** = 0), the contents of **DestName** is ignored.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**DestName**[= *setting$*]

DestType Property, HAcquire Control

Description

The **DestType** property specifies whether the acquired image is to be written to a file, an HImage virtual image or the Clipboard.

Example

Data Type

Integer (enumerated)

Data Value

The allowed **DestType** values are:

Setting	Description
0	(Default). **Clipboard.** Writes the acquired image to the Clipboard in DIB format.
1	**File.** Writes the acquired image to a file. When this option is used, you must specify the file name in the **DestName** property. You must also specify its format and compression method in the **FileFormat** and **FileCompression** properties (you can set **FileJPEGQuality** too, if JPEG format is used and the default quality of 75 is unacceptable).
2	**Control.** Writes the acquired image to an HImage control. When this option is used, the name of the HImage control must be specified in the **DestName** property, and the HImage control must exist in the same Visual Basic form as the HAcquire control.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**DestType**[= *setting%*]

ErrorCode Property, HAcquire Control

Description

This property holds the error code returned by the last HAcquire control operation. Both the **ErrorCode** and **ErrorString** properties are automatically set by the HAcquire control when an error occurs (**ErrorString** will hold a brief description of the error). To clear the **ErrorCode** property after an error has occurred, set it to 25000.

Data Type

Integer (enumerated)

Data Value

The **ErrorCode** value is an integer representing the error generated by the last HAcquire operation. If the last operation did not cause an error, this property will contain the value, 25000.

See HAcquire Error Messages for a complete list of error codes and messages.

Visual Basic

[*form*.]*HAcquire*.**ErrorCode**[= *setting*%]

ErrorShow Property, HAcquire Control

Description

This property determines whether an error is returned to Visual Basic or is simply noted in HAcquire's **ErrorCode** and **ErrorString** properties.

See HAcquire Error Messages for a complete list of error codes and messages.

Data Type

Integer (Boolean)

Data Value

The **ErrorShow** property settings are:

Setting	Description
True (-1)	(Default). Specifies that the control is to immediately return an error code and error message to Visual Basic when an error occurs. The code and message is contained in the **ErrorCode** and **ErrorString** properties, respectively.
False (0)	Specifies that the control is to set the **ErrorCode** and **ErrorString** properties, but is not return an error code to Visual Basic. This setting lets you decide whether (and how) your program will respond to HAcquire errors when they occur.

This value is saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**ErrorShow**[= **True** | **False**]

ErrorString Property, HAcquire Control

Description

This property contains a string describing the error associated with the error code contained in the **ErrorCode** property. The **ErrorString** and **ErrorCode** properties are automatically written by the HAcquire control whenever an error occurs within it.

ErrorString is a read-only property.

See HAcquire Error Messages for a complete list of error codes and messages.

Data Type

String

Visual Basic

[*form*.]*HAcquire*.**ErrorString**

FileCompression Property, HAcquire Control

Description

The **FileCompression** property determines the file compression method used when an acquired image is saved to a file (**DestType** = 1).

See the File Format Compression Table for a list of supported file formats and the compression methods supported by each.

Example

Data Type

Integer

Data Value

The allowed **FileCompression** values are:

Setting	Description
0	**No compression.** Use no compression. **Note -** some image formats do not support an uncompressed form. Refer to the File Format Compression Table for the modes supported by each.
1	(Default). **Default compression.** Use the default compression method for the specified format and image class.
2	**RLE.** Use the Run Length Encoding (RLE) compression method.
3	**CCITT 1D.** Use CCITT Modified Huffman, 1-Dimensional encoding. Use this method with Bilevel TIFF files only.
4	**CCITT Group 3.** Use CCITT Group3 Fax encoding. Use this method with Bilevel TIFF files only.
5	**CCITT Group 4.** Use CCITT Group4 Fax encoding. Use this method with Bilevel TIFF files only.
6	**LZW.** Use modified Lempel-Zif encoding. Use this method with TIFF and GIF formats only.
7	**LZW Horz Predictor.** Use modified Lempel-Zif encoding with horizontal differencing predictor. Use with 8 bits-per-pixel TIFF files only.
8	**JPEG.** Use JPEG compression with the quality factor specified in the **FileJPEGQuality** property.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**FileCompression**[= *setting%*]

File Format Compression Table, HAcquire Control

The following table shows the image classes and file compression methods supported for each file format.

File Format	Image Class	Default Compression	Other Compression
0, 1 (TIFF)	0	3 (CCITT 1D)	0 , 2 , 4 , 5
	1	7 (LZW Horz)	0 , 2 , 6
	2	7 (LZW Horz)	0 2 , 6
	3	7 (LZW Horz)	0 , 2 , 6
2 (BMP/DIB)	0 , 3	0 (No compression)	-
	1 , 2	2 (RLE)	0
3 (HALO Cut)	0 , 3	Not supported	-
	1 , 2	2 (RLE)	-
4 (PCX)	0 , 1 , 2 , 3	2 (RLE)	-
5 (Adobe EPS)	0 , 1 , 2 , 3	0 (No compression)	-
6 (Targa)	0	Not supported	-
	1 , 2 , 3	2 (RLE)	0
7 (GIF)	0 , 1 , 2	6 (LZW)	-
	3	Not supported	-
8 (Sun Raster)	0 , 1 , 2 , 3	2 (RLE)	0
9 (JPEG)	0 , 2	Not supported	-
	1 , 3	8 (JPEG)	0 - 100%
10 (PICT)	0 , 1 , 2 , 3	2 (RLE)	0

FileFormat Property, HAcquire Control

Description

This property determines the format used when saving an acquired image to a file.

Be sure that the **FileFormat** and **FileCompression** properties are set before acquiring an image to a file (set **FileJPEGQuality**, too, if you are using JPEG format).

See the File Format Compression Table for a list of supported file formats and the compression methods allowed by each.

Example

Data Type

Integer (enumerated)

Data Value

The allowed **FileFormat** values are:

Setting	Description
0	(Default) **Default.** HAcquire's default format, which is TIFF.
1	**TIFF.** TIFF file format.
2	**BMP, DIB, RLE.** BMP or DIB formats including Windows, OS/2 1.2 and 2.0 formats (HAcquire reads both Windows and OS/2 formats, and writes in Windows format).
3	**HALO Cut.** Media Cybernetics HALO Cut file format.
4	**PCX.** ZSoft PCX file format.
5	**EPS.** Adobe EPS file format
6	**Targa.** Truevision Targa file format.
7	**GIF.** CompuServe GIF file format.
8	**Raster.** Sun Rasterfile format.
9	**JPEG.** JPEG file format. Conforms to JFIF V1.02
10	**PICT.** Apple PICT file format

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**FileFormat**[= *setting%*]

FileJPEGQuality Property, HAcquire Control

Description

Use this property to set the quality factor that is to be used when an acquired image is stored to a file in JPEG format (**FileFormat** = 9 and **FileCompression** = 8).

Because JPEG is a "lossy" compression method, a certain amount of information is always lost when an image is stored in this format. The **FileJPEGQuality** property lets you control the amount of information loss -- high **FileJPEGQuality** values retain more image data, but do not result in small files, whereas low **FileJPEGQuality** values create very compact files, but eliminate more visual information (highly compressed JPEG files are usually good for thumbnails and icons).

Data Type

Integer

Data Value

The **FileJPEGQuality** property value can be set from 0 to 100, which represents a quality factor of 0% to 100% (the default is 75). The higher the percentage, the better the quality of the compressed image.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**FileJPEGQuality**[= *setting*%]

ImageType Property, HAcquire Control

Description

The **ImageType** property determines the class of image acquired by the device. If you are using the TWAIN user interface, this property *limits* your user to the specified class. If you are acquiring an image directly, this property specifies the kind of image that will be acquired.

Note - if the selected device does not support the requested image class, an error will result.

Data Type

Integer (enumerated)

Data Value

The **ImageType** property settings are:

Setting	Description
0	(Default) **Any.** The acquired image will be of the device's default class. If you are using the TWAIN user interface, this value allows the user to select from all available classes offered by the device. If you using direct acquisition, the device's default class will be returned (in this case, your program must be prepared to accept any image type, since you can never be sure what the source will default to. Many TWAIN devices modify their default values from session to session).
1	**Bilevel.** The image will be acquired as a Bilevel (1-bit) image.
2	**Gray Scale.** The image will be acquired as a Gray Scale (8-bit) image.
3	**True Color.** The image will be acquired as a True Color (24-bit) image.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**ImageType**[= *setting%*]

Resolution Property, HAcquire Control

Description

The **Resolution** property specifies the resolution at which the image is to be acquired. If you are using the TWAIN user interface, this value will be set as the default in the TWAIN dialog. If you are acquiring an image directly, it specifies the resolution at which the image is to be acquired.

Data Type

Integer (enumerated)

Data Value

The **Resolution** property settings are:

Setting	Description
-4	**Draft.** Sets the lowest resolution supported by the source device (even if less than 50 DPI, see **Low**, below).
-3	**Low**. Sets the lowest resolution supported by the source device or 50 DPI, whichever is greater (see also Draft, above).
-2	**Medium.** Sets the median resolution between the Low and High values.
-1	**High.** Sets the highest resolution supported by the source device or 400 DPI, whichever is lesser.
0	(Default) **Default .** Specifies that the device's default resolution is to be used.
any value >1	Sets the resolution to the value specified. The source device will select the nearest supported value if the one specified is not valid for the device. Specified value must not exceed 32767.

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**Resolution**[= *setting%*]

SelHeight, SelWidth Properties, HAcquire Control

Description

The **SelWidth** and **SelHeight** properties define the size of the current area-of-acquisition (AOA). These are used along with **SelLeft** and **SelTop** to acquire data from a rectangular area that is equal to or smaller than the entire acquisition "page".

If you are using the TWAIN user interface, these properties define the *default* AOA size -- a user may resize this area once he/she is within the TWAIN dialog. If you are acquiring an image directly, these properties will determine the actual size of the acquired image.

Example

Data Type

Single

Data Value

The **SelWidth** and **SelHeight** values represent the horizontal and vertical lengths of the AOA. These values are expressed in terms defined by the **Unit** property.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**SelHeight**[= *setting*!]
[*form*.]*HAcquire*.**SelWidth**[= *setting* !]

SelLeft, SelTop Properties, HAcquire Control

Description

The **SelTop** and **SelLeft** properties define the position of the current area-of- acquisition (AOA). These properties are used along with **SelWidth** and **SelHeight** to acquire data from a rectangular area that is equal to or smaller than the entire acquisition "page".

If you are using the TWAIN user interface, these properties define the *default* AOA position -- a user may reposition this area once he/she is within the TWAIN dialog. If you are acquiring an image directly, these properties will determine the actual position of the AOA relative to the acquisition page.

Example

Data Type

Single

Data Value

The **SelLeft** and **SelTop** values identify the position of the upper-left corner of the AOA. These values are expressed in terms defined by the **Unit** property.

These values are not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**SelLeft**[= *setting*!]
[*form*.]*HAcquire*.**SelTop**[= *setting*!]

Source Property, HAcquire Control

Description

This property specifies which of the available TWAIN source devices is to be used for the acquisition. This property is defaulted to the currently selected TWAIN source, and is set when the user selects a device from the TWAIN "Select Source" dialog box using the **Command** property.

Note - you may attempt to set this property directly, however, the string you set here must match a device name in the **SourceNames** array, *exactly*. Unless you have a compelling reason to do otherwise, we recommend that you allow this property to be set by TWAIN's "Select Source" dialog box.

The **Source** property is available at run time, only.

Data Type

String

Data Value

The **Source** value is the name of a TWAIN source device. This string must match one of the source names listed in the **SourceNames** property array.

Visual Basic

[*form*.]*HAcquire*.**Source**[= *setting$*]

SourceCount Property, HAcquire Control

Description

This property specifies the number of TWAIN source devices available on the system. As shown in the example below, it, along with **SourceNames**, can be used to construct a customized source-selection dialog.

```
For i = 0 to HAcquire1.SourceCount
   Combo1.List(i) = HAcquire1.SOurceNames(i)
Next i
Combo1.SelText=HAcquire1.Source
```

Note - The **SourceCount** and **SourceNames** properties have been provided to give you maximum programming flexibility. However, you are not required to use them to select a source device. You may use TWAIN's default "Select Source" dialog by setting the HAcquire **Command** property to 1.

The **SourceCount** property is read-only.

Data Type

Integer

Visual Basic

[*form*.]*HAcquire*.**SourceCount**

SourceNames Property, HAcquire Control

Description

This property contains the names of the TWAIN source devices available on the system. As shown in the example below, it, along with **SourceCount** can be used to construct a customized source-selection dialog.

```
For i = 0 to HAcquire1.SourceCount
  Combo1.List(i) = HAcquire1.SourceNames(i)
Next i
Combo1.SelText=HAcquire1.Source
```

Note - The **SourceNames** and **SourceCount** properties have been provided to give you maximum programming flexibility. However, you are not required to use them to select your source device. You may use TWAIN's default "Select Source" dialog by setting the HAcquire **Command** property to 1.

The **SourceNames** property is available at run time, only. The **SourceNames** property is read-only.

Data Type

String (array)

Data Value

SourceNames is a property array in which each element contains the name of an available TWAIN source device.

Visual Basic

[*form*.]*HAcquire*.**SourceNames**

Unit Property, HAcquire Control

Description

The **Unit** property specifies the unit-of-measure in which the **SelLeft**, **SelTop**, **SelHeight**, and **SelWidth** values are expressed.

Data Type

Integer (enumerated)

Data Value

The **Unit** property settings are:

Setting	Description
0	(Default). **Inches.**
1	**Centimeters.**
2	**Pixels.**
3	**Twips.**

This value is not saved with your Visual Basic form.

Visual Basic

[*form*.]*HAcquire*.**Unit**[= *setting%*]

Version Property, HAcquire Control

Description

The **Version** property holds a string that describes the HAcquire control version.

Version is a read-only property.

Note - use the **About** property to display HAcquire version information at design time. Use the **Version** property to display version information at run time.

Data Type

String

Data Value

The **Version** property contains a single string, formatted as follows: "HAC Version #.#.#". Where "#.#.#" indicates the version number associated with the control.

Visual Basic

[*form*.]*HAcquire.***Version**

Acquire Event, HAcquire Control

Description

The HAcquire event occurs when an acquisition operation is complete (e.g., a scan is performed) or the TWAIN dialog box is closed.

Visual Basic

Sub *HAcquire*_**Acquire**(*Completed* **As Integer**, *Status* **As Integer**, *Count* **As Integer**)

Argument	Description
Completed	Indicates whether the number of requested images (as defined by the Count property) were successfully acquired. A value of 1 indicates that the requested acquisitions were completed (if Count was set to -1, the acquisition of a single image qualifies as successful completion). A value of 0 indicates that the requested number of acquisitions was not completed i.e., the operation was canceled by the user or terminated due to an error condition. If termination was caused by an error, an error code will be contained in *Status* (see below).
Status	Indicates whether the acquisition operation completed normally. A value of 0 indicates normal completion or user cancellation. A non-zero value represents an error code. See HAcquire Error Messages for a complete list of error codes and messages.
Count	Indicates the number of images successfully acquired. If, for example, 4 images were initially requested, but only 2 were acquired, *Count* would be 2 and *Completed* would be 0 (indicating unsuccessful completion of the initial request). *Status* would contain an error code or a 0, indicating whether the early termination was due to an error or user cancellation, respectively

Single Image Acquisition Example, HAcquire Control

The example below shows how the TWAIN user interface is invoked with HAcquire. The first statement directs the acquired data to a file (you can also use this property to direct your output to the Clipboard or an HImage control). The next set of statements specify the file's name and format. The last statement invokes the user interface for the currently selected TWAIN device.

```
HAcquire1.DestType = 0                  ' Routes output to Clipboard
HAcquire1.DestName = "IMG.TIF"          ' Assigns file name
HAcquire1.FileFormat = 1                ' Sets TIF format
HAcquire1.FileCompression = 1           ' Sets default compression
HAcquire1.Command = 2                   ' Invokes TWAIN user interface
```

The example below shows how the direct method would be used to acquire a high-resolution, 3" x 5", Gray Scale image:

```
HAcquire1.DestType = 0                  ' Routes output to Clipboard
HAcquire1.ImageType = 2                 ' Sets gray scale image
HAcquire1.Resolution = -1               ' Sets highest resolution
HAcquire1.Unit = 0                      ' Sets unit to inches
HAcquire1.SelLeft = 1.7                 ' Sets left margin of AOA
HAcquire1.SelTop = 2.5                  ' Sets top margin of AOA
HAcquire1.SelWidth = 3                  ' Sets AOA width
HAcquire1.SelHeight = 5                 ' Sets AOA length
HAcquire1.Command = 3                   ' Performs the acquisition
```

Multiple Image Acquisition Example, HAcquire Control

The example below shows how 3, gray scale images would be acquired using the direct acquisition method. The acquired images would be saved to files: IMG00.TIF, IMG01.TIF and IMG02.TIF.

```
HAcquire1.Count = 3                         ' Sets 3-image acquisition
HAcquire1.DestType = 1                      ' Routes results to disk
HAcquire1.DestName = "IMG##.TIF"            ' Sets file name pattern
HAcquire1.FileFormat = 1                    ' Sets TIF format
HAcquire1.FileCompression = 1               ' Sets default compression
HAcquire1.ImageType = 2                     ' Sets gray scale image
HAcquire1.Resolution = -1                   ' Sets highest resolution
HAcquire1.Command = 3                       ' Performs the acquisition
```

HAcquire Error Messages

ErrorCode	ErrorMessage
25000	*This ErrrorCode value indicates that no error has occurred. When this value is present, ErrorMessage will be null.*
25001	Operation canceled.
25002	Insufficient memory to complete the request.
25003	Cannot lock an image line.
25004	Cannot reallocate cache when opening an image instance.
25005	Error closing an instance.
25006	Invalid width or height parameter.
25007	Illegal line number.
25008	Invalid rectangle coordinate.
25009	Error opening disk cache.
25010	Error reading disk cache.
25011	Error writing to disk cache.
25012	Invalid image handle.
25013	Maximum number of images exceeded.
25014	Operation not valid for this image class.
25015	Internal IMCMD error.
25016	Invalid parameter passed as an argument.
25017	Unsupported feature.
25151	Too many files open at the same time.
25152	Invalid internal file i/o parameter.
25153	Unsupported file format feature.
25154	Cannot complete valid file i/o function.
25155	Insufficient memory for file i/o function.
25156	Bad image data.
25157	Invalid file header.
25158	Error opening the file.
25159	Error closing the file.
25160	Error reading the file.
25161	Error writing the file.
25162	Error seeking in the file.
25163	File not found.
25164	Unknown file format.
25190	Internal error.
25200	TWAIN software is improperly installed.

25201	Cannot select TWAIN source.
25202	TWAIN source cannot be opened.
25203	TWAIN user interface cannot be enabled.
25204	Unsupported transfer or pixel type.
25205	Image file cannot be transferred.
25206	Image cannot be returned by TWAIN source.
25207	Unable to get TWAIN image info.
25208	TWAIN Error; cannot get valid error information.
25209	File name too long for sequential numbering.
25210	Cannot identify default TWAIN source.
25211	No TWAIN sources on system.
25212	Cannot setup file transfer.
25213	Cannot create destination image.
25214	Cannot communicate with Scan Manager.
25215	Pixel type is not supported by TWAIN source.
25216	Palette cannot be returned by TWAIN source.
25230	Cannot open TWAIN.DLL.
25231	Unknown TWAIN error.
25232	Insufficient memory for requested action.
25233	No TWAIN sources in TWAIN directory.
25234	TWAIN Source already in use.
25235	Unknown TWAIN operation error.
25236	Bad capability error.
25239	Protocol error.
25240	Bad value error.
25241	TWAIN triplet out of expected sequence.
25300	Invalid or missing destination file.
25301	Invalid or missing destination file format.
25302	Missing destination control name.
25303	Destination control is not on this form.
25304	Invalid image count.
25305	Cannot communicate with HAC Scan Manager.
25306	Invalid destination type; use DestType File.
25307	File name requires "##" pattern.
25308	Invalid image count for direct acquire.