ImageLib

Portfolio V 3.1
Portfolio V 95

Help!

Software License Agreement

This is a legal agreement between you and Skyline Tools.

ImageLib DLL/VCL

Rights and Limitations

This agreement becomes effective when you install all or any part of the software contained on the disks included with this documentation. By using our software, you agree to the terms of this license. If the terms of this agreement are unacceptable, you may return this package (including all documentation, diskettes, and promotional materials) within 30 days of the purchase date for a refund.

SkyLine Tools provides this program and grants non-exclusive use of its contents. The software which accompanies this license ("ImageLib") is the property of SkyLine Tools and is protected by copyright law. By using ImageLib you agree to abide by the terms of this agreement. Only one user may use this software on a single computer. This software may not be used on a network or any other multi-user platform. Contact SkyLine Tools to arrange for a SITE or network license. You may make backup copies for your own use only. When ImageLib is being compiled into an executable application with the extension “.exe”, then there are no licensing fees or royalties for distribution of the executable and the DLL. Should any part of ImageLib , either the VCL or the DLL be used in a non-compiled application, such as: a value added VCL, VBX, OCX, royalties apply. FAILURE TO COMPLY WITH THE TERMS OUTLINED IN THIS AGREEMENT WILL RESULT IN TERMINATION OF YOUR SOFTWARE LICENCE.

YOU MAY NOT

- Use the product or make copies except as provided by this license.
- Translate, reverse engineer, decompile, or disassemble this software package, except to the extent this restriction is prohibited by applicable regulations.
- Use this software package in a manner that violates any law in the jurisdiction of its use or selling market.
- Rent, lease, assign or transfer this program.

Limited Warranty

THE IMAGELIB SOFTWARE CONTAINED IN THIS BOX IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED. THIS

U.S. Government Restricted Rights:

Important notice

Gif and TIFF use LZW compression which is patented by Unisys. On CompuServe use GO PICS to obtain information about the Unisys patents. **By using ImageLib's GIF and TIF - Read and Write features you acknowledge that SkyLine has notified you about the LZW patent and will not hold SkyLine liable for any legal actions.** This work "JPEG file i/o" is based in part on the Independent JPEG Group.

Introducing ImageLib 3.1 and ImageLib 95

ImageLib is a professional software development tool that allows a programmer to implement **BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, THB, TIF,** and **WMF** images into his/her applications. In addition **AVI, MOV, MID, WAV**, and **RMI** multimedia formats are supported in Delphi. These image and multimedia formats can be implemented to/from a **file** or **database BLOB field**. This **Twain compliant** version of ImageLib is a **VCL/DLL** for use with Delphi. Included in this package are **sixteen Delphi components**, a **Dynamic Link Library** (DLL), **VCL source code** for the Delphi components, programming **examples (C++, Delphi, VB, and VC++)**, electronic **help files**, and software **documentation**.

ImageLib Portfolio 3.1 includes 16-Bit VCL/DLL components. ImageLib Portfolio 95 includes 32-Bit VCL/DLL components. The ImageLib Combo Version includes both the 16-Bit and 32-Bit versions of Delphi.

Other image and multimedia development tools are far more expensive than ImageLib. When users compare ImageLib's color resolution with other imaging tools, they find that ImageLib yields superior results. We proudly invite you to compare our product to our competitions more expensive image libraries.

SkyLine Tools stands behind its product with its highly responsive technical support. If you have a problem that cannot be answered from this manual, drop us an e-mail and ask for assistance. Customers have expressed delight in our prompt useful support.

International developers are able to display strings in the DLL as a resource file, thereby enabling translation into foreign languages.

ImageLib has enhanced TImage and TDBImage VCL/DLL with the following added features:

- Corrected Palette and Stretching of the Image Canvas (Work Around for Delphi Color Palate Limitations);
- Reading and writing of BMP, CMS, GIF, JPG, PCX, PNG, SCM, and TIF images to/from a file or a TBlobField ;
- Color Reduction with Dithering for BMP, GIF, JPG, PCX, PNG, and TIF Images (Dithering for 4 and 8 Bit Indexed Color Output);
- Thumbnail (THB) Image Support;
- BMP 1, 4, 8 and 24 Bit Support;
- GIF 4 and 8 Bit Support;
- GIF 4 and 8 Bit Read Interlaced Support;
- JPG 0 to 100% Save Quality;
- JPG 0 to 100% Smoothing;
- JPG 24 Bit Color and 8 Bit Gray Scale Support;

- PCX 1, 4, 8 and 24 bit Support;
- PNG 1, 2, 4, 8, 24 and 32 Bit Support;
- PNG 1, 2, 4, 8, 24, and 32 Bit Interlaced Support;
- RTF Support to/from a BLOB field (32-Bit DLL Only)
- TIF (Baseline) 1, 4, 8, and 24 Bit Support;
- TIF Compression - CCIT, LZW, and PACKBITS ;
- Twain Support;
- Capture Image from Video ;
- Reading and Writing of Horizontal Credit Messages (CMS) and Vertical Scrolling Messages (SCM) to/from a file or a TBlobField ;
- Reading and Writing of AVI, MOV, WAV, RMI and MID multimedia images to/from a file or a TBlobField ;
- Reading and Writing of ICO and WMF Images to/from a File (Delphi inherited);
- CUT, COPY and PASTE to/from the Clipboard ;
- Full Print Support with One Line Code Implementation;
- Internal Scrolling and Credit Message Editor;
- DLL Callback Function (shows progress bar and processes messages);
- No Code (VCL) to Display Image Formats from a TBlobField ;
- Loading/Saving all TBlobField Images to/from File ;
- Conversion of all TBlobField images to BMP/GIF/JPG/PCX/PNG/TIF File ;
- Images Pasted from the Clipboard can be stored as a BMP/GIF/JPG/PCX/PNG/TIF file or TBlobField;
- Open and Save Dialog Boxes with Thumbnail for Previewing Video, Images, and Sound;
- Enhanced Image Manipulation, Including Zoom, Flip, and Mirror Tools;
- Optional Toolbars for Rapid Application Development;
- Powerful Text Features that allow Text Over Image and Text Rotation;
- Many Useable Code Examples including VCLs;

  and
- Foreign Error Strings Support. DLL strings are stored in the DLL Resource.

+ **Should you have a need to make calls directly to the DLL, we listed all the Pascal interface calls in DLL30.INT or DLL30.PAS.**

ImageLib 3.1 and ImageLib 95 includes **sixteen Delphi components.** These incredible components are summarized below.

1. TDBIconComboBox

TDBIconComboBox provides a means of listing and displaying icons from a BLOB field. TDBIconComboBox is a component derived from TComboBox and has several of the same functions and properties.

2. TDBIconEditor

TDBIconEditor provides a means of editing icons from a BLOB field.

3. TDBIconListBox

TDBIconListBox provides a means of listing and displaying icons from a BLOB field. TDBIconListBox is a component derived from the Delphi tListBox and has several of the same functions and properties.

4. TMImageToolBar

TMImageToolBar is a toolbar for use with TPMultiImage. This toolbar can be used on images during design time, (limited) and run time to acquire, copy, cut, flip, paste, print, open, reset, rotate, save, stretch, and zoom images. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette corrector.

5. TMIMediaPlayer

TMIMediaPlayer is a component derived from the Delphi MediaPlayer and has all the same functions and properties. When using the TMIMediaPlayer it is not necessary to assign anything to the TMIMediaPlayer directly, TPMultiMedia will take care of it. TPMultiMedia will automatically enable/disable the playback of:

AVI:	If video for windows is not installed;
MID:	If no midi playback drivers are installed;
MOV:	If quicktime for windows is not installed;
RMI:	If no midi playback drivers are installed;
WAV:	If no sound support is installed;

Thus your program will not crash if no sound card is installed or Video for Windows is not present.

6. TMMOpenDialog

TMMOpenDialog is ImageLib's version of a typical dialog box used to open files. Our dialog box plays and displays a thumbnail preview of the image to be opened. TMMOpenDialog can be used with TPMultiImage, TPMultiMedia, TPDBMultiImage, and TPDBMultiMedia.

7. TMMSaveDialog

ImageLib's version of a typical dialog box used to save files. Our dialog box plays and displays a thumbnail preview of the image to be saved. TMMSaveDialog can be used with TPMultiImage, TPMultiMedia, TPDBMultiImage, and TPDBMultiMedia.

8. TMultiMediaToolBar

TMultiMediaToolBar is a toolbar for use with TPMultiMedia. This toolbar can be used on images during design time (Limited) and run time to play multimedia, acquire, open, save, print, cut, copy, paste, zoom, rotate, flip, reset, and stretch images from a file. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette.

9. TPDBMediaPlayer

TPDBMediaPlayer is a component derived from the Delphi MediaPlayer and has all the same functions and properties. When using the TPDBMediaPlayer it is not necessary to assign anything to TPDBMediaPlayer directly, TPDBMultiMedia will take care of it. TPDBMultiMedia will automatically enable/disable the playback of:

AVI:	If video for windows is not installed;
MID:	If no midi playback drivers are installed;
MOV:	If quicktime for windows is not installed;
RMI:	If no midi playback drivers are installed;
WAV:	If no sound support is installed;

Thus your program will not crash if no sound card is installed or Video for Windows is not present.

10. TPDBMImageToolBar

TPDBMMImageToolBar is a toolbar for use with TPDBMultiImage. This toolbar can be used on images during design time (Limited) and run time to acquire, copy, cut, flip, paste, print, open, reset, rotate, save, stretch, and zoom images from a BLOB field. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette corrector.

11. TPDBMMediaToolBar

TPDBMMediaToolBar is a toolbar for use with TPDBMultiMedia. This toolbar can be used on images during design time (Limited) and run time to play multimedia, acquire, copy, cut, flip, paste, print, open, reset, rotate, save, stretch, and zoom images from a BLOB field. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette corrector.

12. TPDBMultiImage

TPDBMultiImage displays and stores BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, TIF and WMF (ICO and WMF are read only) to/from a TBlobField. TPDBMultiImage is the data-aware VCL version of TPMultiImage.

13. TPDBMultiMedia

TPDBMultiMedia has all the same properties and functions as TPDBMultiImage. However, in addition to the storing and displaying of BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, TIF, and WMF (ICO and WMF are read only) from a TBlobField, it also stores and plays AVI, MOV, MID, WAV and RMI multimedia BLOBs. TPDBMediaPlayer is derived from Delphi's MediaPlayer and has the same functions and properties. When using the TPDBMediaPlayer you do not need to assign anything to TPDBMediaPlayer directly, TPDBMultiMedia will take care of it. TPDBMultiMedia will automatically enable/disable the playback of:

AVI:	If video for windows is not installed;
MID:	If no midi playback drivers are installed;
MOV:	If quicktime for windows is not installed;
RMI:	If no midi playback drivers are installed;
WAV:	If no sound support is installed;

Thus your program will not crash if no sound card is installed or Video for

Windows is not present.

14. TPMultiImage

TPMultiImage displays and stores BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, and WMF (ICO and WMF are read only) to/from a file. TPMultiImage is a data-aware VCL.

15. TPMultiMedia

TPMultiMedia has all the same properties and functions as TPMultiImage. However, in addition to the storing and displaying of BMP, CMS, GIF, ICO, JPEG, PCX, PNG, SCM, TIFF, and WMF (ICO and WMF are read only) from a file; TPMultiMedia also stores and plays AVI, MOV, MID, WAV, and RMI multimedia files. When using the TMIMediaPlayer, it is not necessary to assign anything to the TMIMediaPlayer directly, TPMultiMedia will take care of it. TPMultiMedia will automatically enable/disable the playback of:

AVI:	If video for windows is not installed;
MID:	If no midi playback drivers are installed;
MOV:	If quicktime for windows is not installed;
RMI:	If no midi playback drivers are installed;
WAV:	If no sound support is installed;

Thus your program will not crash if no sound card is installed or Video for Windows is not present.

16. TThumbPreview (Component)

ImageLib supports the use of thumbnail images with the TThumbPreview component. Thumbnails are miniature copies of larger image files. The TThumbPreview component uses a thumbnail manager to display multiple thumbnails, create new thumbnails, and remove old thumbnails. Double clicking one of the images on the thumbnail manager will display that image.

What's New in ImageLib 3.1 and ImageLib 95

SkyLine Tools strongly supports the concept of rapid prototyping and constant product improvement. A day rarely passes without new ideas for modification of ImageLib. Concurrent development and product testing are endless processes at SkyLine Tools. You can rest assured that new and better things will come to later versions of ImageLib. It is this constant innovation and desire to be the best that makes our company shine. We value our customers input and look forward to hearing about any suggestion you might have. If you find an error in this manual, drop us an e-mail and we will modify it for the next edition. The following is a list of new and improved capabilities for this version of ImageLib.

- New 32-Bit DLL/VCL (ImageLib 95 and Combo only)
- Reading and writing of TIF (Baseline) images to/from a file or a TBlobField;
- Open and Save Dialog Boxes with Thumbnail for Previewing Video, Images, and Sound;
- Enhanced Image Manipulation, Including Zoom, Flip, and Mirror Tools;
- Optional Toolbars for Rapid Application Development;
- Powerful Text Features that allow Text Over Image and Text Rotation;
- Color Reduction with Dithering for TIF Images (Dithering for 4 and 8 Bit Indexed Color Output);
- BMP 1 Bit Support;
- GIF 4 and 8 Bit Read Interlaced Support;
- JPG 8 Bit Gray Scale Support;
- RTF Support to/from a BLOB field (32-Bit DLL Only)
- Thumbnail (THB) Image Support;
- TIF (Baseline) 1, 4, 8, and 24 Bit Support;
- TIF Compression - CCIT, LZW, and PACKBITS Support;
- Twain Support;
- Capture Image from Video;
- Conversion of all TBlobField images to a TIF File Support;
- Images Pasted from the Clipboard can be stored as a TIF file or TBlobField;
- Code Examples for New Features;
- New and Improved Software Documentation;
  and
- New and Improved Electronic Help File.

ImageLib 3.1 and ImageLib 95 includes eleven new Delphi components and four improved components**.** These incredible components are summarized below.

1. TDBIconComboBox (New!)
TDBIconComboBox provides a means of listing and displaying icons from a BLOB field.

2. TDBIconEditor (New!)
TDBIconEditor provides a means of editing icons from a BLOB field.

3. TDBIconListBox (New!)
TDBIconListBox provides a means of listing and displaying icons from a BLOB field.

4. TMImageToolBar (New!)
TMImageToolBar is a toolbar for use with TPMultiImage.

5. TMIMediaPlayer (New!)
TMIMediaPlayer is a media player for use with TPMultiMedia.

6. TMMOpenDialog (New!)
TMMOpenDialog is ImageLib's version of a typical dialog box used to open files. Our dialog box plays and displays a thumbnail preview of the image to be opened.

7. TMMSaveDialog (New!)
ImageLib's version of a typical dialog box used to save files. Our dialog box plays and displays a thumbnail preview of the image to be saved.

8. TMultiMediaToolBar (NEW!)
TMultiMediaToolBar is a toolbar for use with TPMultiMedia.

9. TPDBMImageToolBar (NEW!)
TPDBMImageToolBar is a toolbar for use with TPDBMultiImage.

10. TPDBMMediaToolBar (NEW!)
TPDBMMediaToolBar is a toolbar for use with TPDBMultiMedia.

11. TPDBMultiImage (Improved!)
TIF and Twain support have been added to this component. In addition, improvements have been made to the existing image support.

12. TPDBMultiMedia (Improved!)
TIF and Twain support have been added to this component. In addition, improvements have been made to the existing image support.

13. TPMultiImage (Improved!)
TIF and Twain support have been added to this component. In addition, improvements have been made to the existing image support.

14. TPMultiMedia (Improved!)
TIF and Twain support have been added to this component. In addition, improvements have been made to the existing image support.

15. TThumbPreview (New!)
ImageLib supports the use of thumbnail images with the TThumbPreview

component. Thumbnails are miniature copies of larger image files. The TThumbPreview component uses a thumbnail manager to display multiple thumbnails, create new thumbnails, and remove old thumbnails. Double clicking one of the images on the thumbnail manager will display that image

Installation Instructions

This procedure need be done only once for owners that have purchased only ImageLib 3.1 or ImageLib 95. Be sure that you are installing the proper version of ImageLib with the equivalent 16-Bit or 32-Bit version of Delphi. ImageLib Combo Version owners will have to complete this process twice. Once for the ImageLib 95 (32-Bit) and again for ImageLib 3.1 (16-Bit). Delphi 2.0 owners should consult the Delphi documentation for additional information.

1. Run Setup.EXE from the disk.

2. BACKUP YOUR \DELPHI\BIN\COMPLIB.DCL (Better safe than sorry).

3. Copy the IMAGELIB VCL files into the directory containing your 3rd party added VCLs: (If you don't have a directory yet please, make one).

4. Execute Delphi. From the menu bar in Delphi, select Options\Install components\Add and browse your 3rd party added VCLs directory. Select PMREG.PAS and press the OK button.

RUN THE DEMOALL.DPR FOR ALL NEW FEATURES

+ **The main example project DEMOALL.DPR contains a button for help. If the help file is not in the same directory as the project, the example might not find the help file.**

Installation Troubleshooting

The Delphi Library searchpath is very short (127 characters). The more VCL components you add, the larger your searchpath. Should you get a message PMREG.PAS or PMREG.DCU not found, then your path is being truncated, the solution is to copy several 3rd party VCLs into one directory and delete the freed directories from your searchpath.

If COMPLIB cannot find SKY16V3C.DLL/SKY32V3C.DLL, you will notice that all Icons are gone from your Delphi toolbar and the message “COMPLIB.DCL NOT FOUND” or “COULD NOT OPEN COMPLIB.DCL” will appear. Do not Panic. Copy SKY16V3C.DLL/SKY32V3c.DLL to a directory on your path or to the windows\system directory and restore your backed up COMPLIB.

Technical Support

For responsive technical support*, please E-Mail your questions to the appropriate person listed below.   If your question is better explained over the telephone, please call (818)766-3900 and ask for technical support.

Technical support* for C, C++, VB applications:
Kevin Adams: CompuServe 74742,1444 or
Internet: 74742.1444@compuserve.com

Technical support* for Delphi, Pascal and VB applications:
Jan Dekkers: CompuServe 72130,353 or
Internet: 72130.353@compuserve.com

For custom contract services, please E-Mail your requests to the person listed below or telephone (800) 404-3822 and ask for contract services.

Custom Contract Services:
Chris W. Giggey: CompuServe 76016.2201 or
Internet: 76016.2201@compuserve.com

***No cost technical support is offered for current versions of ImageLib only.   No cost technical support for older versions of ImageLib will be at SkyLine Tools' option.**

TDBIconComboBox (Component)

The TDBIconComboBox provides a means of listing and displaying icons from a BLOB field.   TDBIconComboBox is a component derived from TComboBox and has several of the same functions and properties (SEE Delphi Documentation for the properties not listed here).   The database used to store icons needs to have a string field with a description.   This string field MUST be indexed.   What follows are properties unique to TDBIconComboBox.   For example examine the unit UIconLB.

BlobField (Property)
ShowIcons (Property)
Table (Property)

BlobField (Property)

Value

The name of the BLOB field

Purpose

To provide the name of the BLOB field that contains the icons to be displayed in the combo box

Example

```
DBIconComboBox1.BlobField := ‘ICONBLOB’;
```

ShowIcons (Property)

Value

True or False

Purpose

Turns the icon display on and off in the combo box

Example

```
DBIconComboBox1.ShowIcons := True;
```

Table (Property)

Value

The name of the table that contains the BLOB field

Purpose

To provide the name of the table that contains the icons to be displayed in the combo box

Example

```
DBIconComboBox1.Table := ‘C:\DELPH\EXAMPLES\ICONDB.DB’;
```

TDBIconEditor (Component)

The TDBIconEditor provides a means of editing icons from a BLOB field. The database used to store icons needs to have a string field with a description. What follows are properties for the TDBIconEditor.

+ **IndexFieldName must contain the name of the index for the icon description field. TDBIconEditor will not function properly if this is not done.**

IconBlobFieldName (Property)
IndexFieldName (Property)
Table (Property)

IconBlobFieldName (Property)

Value

The name of the BLOB field

Purpose

To provide the name of the BLOB field that contains the icons to be edited

Example

```
DBIconEditor.BlobField := ‘iconfield’;
```

IndexFieldName (Property)

Value

The name of the index field

Purpose

To provide the name of the index for the icon description field.   This is needed by the icon editor to function properly.

Example

```
DBIconEditor.IndexFieldName := ‘ICONDESC’;
```

Table (Property)

Value

The name of the table that contains the BLOB field

Purpose

To provide the name of the table that contains the icons to be edited

Example

```
DBIconEditor.Table := ‘C:\DELPHI\EXAMPLES\ICONDB.DB’;
```

TDBIconListBox (Component)

The TDBIconListBox provides a means of listing and displaying icons from a BLOB field.   TDBIconListBox is a component derived from the Delphi tListBox and has several of the same functions and properties (SEE Delphi Documentation for the properties not listed here).   The database used to store the icons needs to have a string field with a description.   This string field MUST be indexed.   What follows are properties unique to TDBIconListBox.   For example see the unit UIconLB.

BlobField (Property)
ShowIcons (Property)
Table (Property)

BlobField (Property)

Value

The name of the BLOB field

Purpose

To provide the name of the BLOB field that contains the icons to be displayed in the list box

Example

```
DBIconListBox1.BlobField := ‘IconBlob’;
```

ShowIcons (Property)

Value

True or False

Purpose

Turns the icon display on and off in the list box

Example

```
DBIconListBox1.ShowIcons := True;
```

Table (Property)

Value

The name of the table that contains the BLOB field

Purpose

To provide the name of the table that contains the icons to be displayed in the list box

Example

```
DBIconListBox1.Table := 'C:\DELPHI\EXAMPLES\ICONDB.DB';
```

TMImageToolBar (Component)

Toolbar for use with TPMultiImage. This toolbar can be used on images during design time (limited) and run time to acquire, copy, cut, flip, paste, print, open, reset, rotate, save, stretch images, and zoom. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette corrector.

MultiImage (Property)
OpenDialogDir (Property)
OpenDialogFilter (Property)
SaveDialogDir (Property)
SaveDialogFilter (Property)
ShowToolBar (Property)
tbCaption (Property)
tbColorPalatte (Property)
tbCopyImage (Property)
tbCreditMessage (Property)
tbCutImage (Property)
tbFlipImage (Property)
tbImageOpen (Property)
tbImageSave (Property)
tbLeft (Property)
tbPasteImage (Property)
tbPrintImage (Property)
tbResetImage (Property)
tbRotateImage (Property)
tbScanImage (Property)
tbScrollMessage (Property)
tbStretchImage (Property)
tbStretchImageratio (Property)
tbThumbs (Property)
tbTop (Property)
tbZoomInImage (Property)
tbZoomOutImage (Property)
UseMMOpenDialog (Property)
UseMMSaveDialog (Property)

MultiImage (Property)

Value

The name the TPMultiImage or TPDBMultiImage used with the toolbar

Purpose

To connect TPMultiImage or TPDBMultiImage with the toolbar used

Example

For use with TMImageToolBar

```
    MImageToolBar1.MultiImage := PMultiImage1;
```

For use with TPDBMImageToolBar

```
    PDBMImageToolBar1.MultiImage := PDBMultiImage1;
```

OpenDialogDir (Property)

Value

The name of the default directory

Purpose

To specify the name of the default directory

Example

For use with TMImageToolBar
```
MImageToolBar1.OpenDialogDir := C:\IMAGES;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.OpenDialogDir := C:\IMAGES;
```

For use with TPDBMImageToolBar
```
PDBMediaToolBar1.OpenDialogDir := C:\IMAGES;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.OpenDialogDir := C:\IMAGES;
```

OpenDialogFilter (Property)

Value

The file extensions to be displayed in the dialog box

Purpose

To specify the file extension that can be displayed as choices in the open dialog box

Example

For use with TMImageToolBar

```
MImageToolBar1.OpenDialogFilter := ‘JPG Files|*.jpg’;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.OpenDialogFilter := ‘JPG Files|*.jpg’;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.OpenDialogFilter := ‘JPG Files|*.jpg’;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.OpenDialogFilter := ‘JPG Files|*.jpg’;
```

SaveDialogDir (Property)

Value

The name of the default directory

Purpose

To specify the name of the default directory

Example

For use with TMImageToolBar

```
MImageToolBar1.SaveDialogDir := ‘C:\IMAGES’
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.SaveDialogDir := ‘C:\IMAGES’
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.SaveDialogDir := ‘C:\IMAGES’
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.SaveDialogDir := ‘C:\IMAGES’
```

SaveDialogFilter (Property)

Value

The file extensions to be displayed in the dialog box

Purpose

To specify the file extension that can be displayed as choices in the save dialog box

Example

For use with TMImageToolBar
```
MImageToolBar1.SaveDialogFilter := ‘JPG Files|*.jpg’;
```

For use with TMultiImageToolBar
```
MultiMediaToolBar1.SaveDialogFilter := ‘JPG Files|*.jpg’;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.SaveDialogFilter := ‘JPG Files|*.jpg’;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.SaveDialogFilter := ‘JPG Files|*.jpg’;
```

ShowToolBar (Property)

Value

True or False

Purpose

Turns the tool bar on and off.   When true is selected at design time, the tool bar is available for use.

Example

For use with TMImageToolBar

```
    MImageToolBar1.ShowToolBar := True;
```

For use with TMultiMediaToolBar

```
    MultiMediaToolBar1.ShowToolBar := True;
```

For use with TPDBMImageToolBar

```
    PDBMImageToolBar1.ShowToolBar := True;
```

For use with TPDBMMediaToolBar

```
    PDBMMediaToolBar1.ShowToolBar := True;
```

tbCaption (Property)

Value

The name as displayed in the title bar

Purpose

Allows the programmer to specify the name displayed in title bar

Example

For use with TMImageToolBar

```
    MImageToolBar1.tbCaption := ‘ImageLib Tools’;
```

For use with TMultiMediaToolBar

```
    MultiMediaToolBar1.tbCaption := ‘ImageLib Tools’;
```

For use with TPDBMImageToolBar

```
    PDBMImageToolBar1.tbCaption := ‘ImageLib Tools’;
```

For use with TPDBMMediaToolBar

```
    PDBMMediaToolBar1.tbCaption := ‘ImageLib Tools’;
```

tbColorPalatte (Property)

Value

True or False

Purpose

Toggles the Color Palette button on the tool bar between on and off. The function of the color palette button is to switch between the Delphi color palette and the ImageLib color palette.   This is a work around for a problem Delphi has with some 256 color VGA cards. By setting the ImageLib Palette to True, you will bypass the Delphi way of painting. Instead the ImageLib painting method will be used, which is basically using the windows StretchBlt and BitBlt API calls.

+ **ImageLibPalette should always be set to true is you want to ZOOM, Transform, Flip, Rotate and StretchRatio**.

Example

For use with TMImageToolBar
```
MImageToolBar1.tbColorPalette := True;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.tbColorPalette := True;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.tbColorPalette := True;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.tbColorPalette := True;
```

tbCopyImage (Property)

Value

True or False

Purpose

Toggles the copy image button on the tool bar between on and off. The function of the copy image button is to copy the contents of a MultiImage to the clipboard.

Example

For use with TMImageToolBar

```
MImageToolBar1.tbCopyImage := True;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.tbCopyImage := True;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbCopyImage := True;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbCopyImage := True;
```

tbCreditMessage (Property)

Value

True or False

Purpose

Toggles the credit message editor button on the tool bar between on and off.   The function of credit message editor button is to open the credit message editor.

Example

For use with TMImageToolBar
```
    MImageToolBar1.tbCreditMessage := True;
```

For use with TMultiMediaToolBar
```
     MultiMediaToolBar1.tbCreditMessage := True;
```

For use with TPDBMImageToolBar
```
    PDBMImageToolBar1.tbCreditMessage := True;
```

For use with TPDBMMediaToolBar
```
    PDBMMediaToolBar1.tbCreditMessage := True;
```

tbCutImage (Property)

Value

True or False

Purpose

Toggles the cut image button on the tool bar between on and off. The function of the cut image button is to cut the contents of a MultiImage an place it on the clipboard.

Example

For use with TMImageToolBar

```
    MImageToolBar1.tbCutImage := True;
```

For use with TMultiMediaToolBar

```
     MultiMediaToolBar1.tbCutImage := True;
```

For use with TPDBMImageToolBar

```
    PDBMImageToolBar1.tbCutImage := True;
```

For use with TPDBMMediaToolBar

```
    PDBMMediaToolBar1.tbCutImage := True;
```

tbFlipImage   (Property)

Value

True or False

Purpose

Toggles the flip image button on the tool bar between on and off.   The function of the flip image button is to flip the contents of a MultiImage from left to right (mirror).

+ **The ImageLib color palette and stretch/stretchratio image must both be in use for this feature to function properly.**

Example

For use with TMImageToolBar
```
MImageToolBar1.tbFlipImage := True;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.tbFlipImage := True;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.tbFlipImage := True;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.tbFlipImage := True;
```

tbImageOpen (Property)

Value

True or False

Purpose

Toggles the image open button on the tool bar between on and off. The function of the image open button is to activate an open dialog box. If MImageToolBar.MMOpenDialog is set to true, an ImageLib open dialog box is used, otherwise a Delphi open dialog box is used.

Example

For use with TMImageToolBar

```
MImageToolBar1.tbImageOpen := True;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.tbImageOpen := True;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbImageOpen := True;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbImageOpen := True;
```

tbImageSave (Property)

Value

True or False

Purpose

Toggles the image save button on the tool bar between on and off. The function of the image save button is to activate a save dialog box. If MImageToolBar.MMSaveDialog is set to true, an ImageLib save dialog box is used, otherwise a Delphi save dialog box is used.

Example

For use with TMImageToolBar
```
MImageToolBar1.tbImageSave := True;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.tbImageSave := True;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.tbImageSave := True;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.tbImageSave := True;
```

tbLeft (Property)

Value

Integer

Purpose

Left position of the toolbar

Example

For use with TMImageToolBar

```
MImageToolBar1.tbLeft := 10;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.tbLeft := 10;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbLeft := 10;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbLeft := 10;
```

tbPasteImage (Property)

Value

True or False

Purpose

Toggles the paste image button on the tool bar between on and off.   The function of the paste image button is to place the contents of the clipboard in a MultiImage.

Example

For use with TMImageToolBar

```
MImageToolBar1.tbPasteImage := True;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.tbPasteImage := True;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbPasteImage := True;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbPasteImage := True;
```

tbPrintImage (Property)

Value

True or False

Purpose

Toggles the print image button on the tool bar between on and off. The function of the print image button is to print the contents of a MultiImage.

Example

For use with TMImageToolBar

```
MImageToolBar1.tbPrintImage := True;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.tbPrintImage := True;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbPrintImage := True;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbPrintImage := True;
```

tbResetImage (Property)

Value

True or False

Purpose

Toggles the reset image button on the tool bar between on and off. The function of the reset image button is to reset the contents of a MultiImage to its original configuration and orientation.

Example

For use with TMImageToolBar
```
MImageToolBar1.tbResetImage := True;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.tbResetImage := True;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.tbResetImage := True;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.tbResetImage := True;
```

tbRotateImage (Property)

Value

True or False

Purpose

Toggles the rotate image button on the tool bar between on and off. The function of the rotate image button is to rotate the contents of a MultiImage 180 degrees (Upside down).

+ **The ImageLib color palette and stretch/stretchratio image must both be in use for this feature to function properly.**

Example

For use with TMImageToolBar

```
MImageToolBar1.tbRotateImage := True;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.tbRotateImage := True;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbRotateImage := True;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbRotateImage := True;
```

tbScanImage (Property)

Value

True or False

Purpose

Toggles the scan image button on the tool bar between on and off. The function of the scan image button is to utilize a Twain compliant device to acquire an image and place it in a MultiImage.

Example

For use with TMImageToolBar
```
MImageToolBar1.tbScanImage := True;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.tbScanImage := True;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.tbScanImage := True;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.tbScanImage := True;
```

tbScrollMessage (Property)

Value

True or False

Purpose

Toggles the scroll message editor button on the tool bar between on and off.   The function of scroll message editor button is to open the scroll message editor.

Example

For use with TMImageToolBar
```
    MImageToolBar1.tbScrollMessage := True;
```

For use with TMultiMediaToolBar
```
     MultiMediaToolBar1.tbScrollMessage := True;
```

For use with TPDBMImageToolBar
```
    PDBMImageToolBar1.tbScrollMessage := True;
```

For use with TPDBMMediaToolBar
```
    PDBMMediaToolBar1.tbScrollMessage := True;
```

tbStretchImage (Property)

Value

True or False

Purpose

Toggles the stretch image button on the tool bar between on and off. The function of stretch image button is to display an image such that it fills a MultiImage component .   The aspect ratio is not maintained for this type of stretch (image distortion).

Example

For use with TMImageToolBar
```
    MImageToolBar1.tbStretchImage := True;
```

For use with TMultiMediaToolBar
```
    MultiMediaToolBar1.tbStretchImage := True;
```

For use with TPDBMImageToolBar
```
    PDBMImageToolBar1.tbStretchImage := True;
```

For use with TPDBMMediaToolBar
```
    PDBMMediaToolBar1.tbStretchImage := True;
```

tbStretchImageratio (Property)

Value

True or False

Purpose

Toggles the stretch image ratio button on the tool bar between on and off.   The function of stretch image ratio button is to display an image such that it fills a MultiImage component .   The aspect ration is maintained for this type of stretch (no image distortion).

+ **The ImageLib color palette and stretch image must be in use for this feature to function properly.**

Example

For use with TMImageToolBar
```
MImageToolBar1.tbStretchImageratio := True;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.tbStretchImageratio := True;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.tbStretchImageratio := True;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.tbStretchImageratio := True;
```

tbThumbs (Property)

Value

True or False

Purpose

Toggles the thumbnail manager button on the tool bar between on and off.   The function of the thumbnail manager button is to activate the thumbnail manager.   For more information on the thumbnail manager, read the section on the TThumbPreview component.

Example

For use with TMImageToolBar
```
    MImageToolBar1.tbThumbs := True;
```

For use with TMultiMediaToolBar
```
     MultiMediaToolBar1.tbThumbs := True;
```

For use with TPDBMImageToolBar
```
    PDBMImageToolBar1.tbThumbs := True;
```

For use with TPDBMMediaToolBar
```
    PDBMMediaToolBar1.tbThumbs := True;
```

tbTop (Property)

Value

Integer

Purpose

Top position of the toolbar

Example

For use with TMImageToolBar

```
MImageToolBar1.tbTop := 20;
```

For use with TMultiMediaToolBar

```
MultiMediaToolBar1.tbTop := 20;
```

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbTop := 20;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbTop := 20;
```

tbZoomInImage (Property)

Value

True or False

Purpose

Toggles the zoom in button on the tool bar between on and off.   The function of zoom in button is to increase the size of the image such that fine details are easier to see.

+ **The ImageLib color palette and stretch/stretchratio image must both be in use for this feature to function properly.**

Example

For use with TMImageToolBar
```
    MImageToolBar1.tbZoomInImage := True;
```

For use with TMultiMediaToolBar
```
    MultiMediaToolBar1.tbZoomInImage := True;
```

For use with TPDBMImageToolBar
```
    PDBMImageToolBar1.tbZoomInImage := True;
```

For use with TPDBMMediaToolBar
```
    PDBMMediaToolBar1.tbZoomInImage := True;
```

tbZoomOutImage (Property)

Value

True or False

Purpose

Toggles the zoom out button on the tool bar between on and off. The function of zoom out button is to decrease the size of the image such that more of the image may be displayed in a smaller space.

+ **The ImageLib color palette and stretch/stretchratio image must both be in use for this feature to function properly.**

Example

For use with TMImageToolBar
```
MImageToolBar1.tbZoomOutImage := True;
```

For use with TMultiMediaToolBar
```
MultiMediaToolBar1.tbZoomOutImage := True;
```

For use with TPDBMImageToolBar
```
PDBMImageToolBar1.tbZoomOutImage := True;
```

For use with TPDBMMediaToolBar
```
PDBMMediaToolBar1.tbZoomOutImage := True;
```

UseMMOpenDialog (Property)

Value

True or False

Purpose

Toggles between the Delphi open dialog box and the ImageLib open dialog box. When the image open button from the MImageToolBar is used, either the Delphi open dialog box or the ImageLib open dialog box is used.

Example

For use with TMImageToolBar
```
    MImageToolBar1.UseMMOpenDialog := True;
```

For use with TMultiMediaToolBar
```
     MultiMediaToolBar1.UseMMOpenDialog := True;
```

For use with TPDBMImageToolBar
```
    PDBMImageToolBar1.UseMMOpenDialog := True;
```

For use with TPDBMMediaToolBar
```
    PDBMMediaToolBar1.UseMMOpenDialog := True;
```

UseMMSaveDialog (Property)

Value

True or False

Purpose

Toggles between the Delphi save dialog box and the ImageLib save dialog box.   When the image save button from the MImageToolBar is used, either the Delphi save dialog box or the ImageLib save dialog box is used.

Example

For use with TMImageToolBar
```
    MImageToolBar1.UseMMSaveDialog := True;
```

For use with TMultiMediaToolBar
```
     MultiMediaToolBar1.UseMMSaveDialog := True;
```

For use with TPDBMImageToolBar
```
    PDBMImageToolBar1.UseMMSaveDialog := True;
```

For use with TPDBMMediaToolBar
```
    PDBMMediaToolBar1.UseMMSaveDialog := True;
```

TMIMediaPlayer (Component)

TMIMediaPlayer is a component derived from the Delphi MediaPlayer and has exactly all the same functions and properties (SEE Delphi Documentation). When using the TMIMediaPlayer it is not necessary to assign anything to TMIMediaPlayer directly, TPMultiMedia will take care of it. TPMULTIMEDIA will automatically enable/disable the playback of:

AVI:	If video for windows isn't installed;
MOV:	If quicktime for windows isn't installed;
WAV:	If no sound support is installed;
RMI:	If no midi playback drivers are installed;
MID:	If no midi playback drivers are installed;

Thus your program will not crash if no sound card is installed or Video for Windows is not present.

+ **The Delphi MultiMediaPlayer is extension sensitive, thus extensions can’t be changed.**

TMMOpenDialog (Component)

ImageLib’s version of a typical dialog box used to open files. TMMOpenDialog can be used with TPMultiImage, TPMultiMedia, TPDBMultiImage, and TPDBMultiMedia. For additional information about opening dialog boxes and typical dialog box properties see Delphi documentation.

AutoDisPlay (Property)
AutoPlay (Property)
Filename (Hidden Property)

AutoDisPlay (Property)

Value

True or False

Purpose

When a file is selected from the open dialog box, a preview of the file contents is displayed.

Example

```
MMOpenDialog1.AutoDisPlay:=True;
```

AutoPlay (Property)

Value

True or False

Purpose

When a multimedia file is selected from the open dialog box, a preview of the file contents is played.

Example

```
MMOpenDialog1.AutoPlay:=True;
```

Filename (Hidden Property)

Value

The name of the file in which the data is to be retrieved.

Purpose

When a file is selected from the open dialog box, a preview of the file contents is displayed.

Example

```
If MMOpenDialog1.Execute then
 PMultiImage1.ImageName(MMOpenDialog1.FileName);
end;
```

TMMSaveDialog (Component)

ImageLib's version of a typical dialog box used to save files. TMMSaveDialog can be used with TPMultiImage, TPMultiMedia, TPDBMultiImage, and TPDBMultiMedia. For additional information about using dialog boxes and typical dialog box properties see Delphi documentation.

AutoDisPlay (Property)
AutoPlay (Property)
Filename (Hidden Property)

AutoDisPlay (Property)

Value

True or False

Purpose

When a file is selected from the save dialog box, a preview of the file contents is displayed.

Example

```
MMSaveDialog1.AutoDisPlay:=True;
```

AutoPlay (Property)

Value

True or False

Purpose

When a multimedia file is selected from the save dialog box, a preview of the file contents is played.

Example

```
MMSaveDialog1.AutoPlay:=True;
```

Filename (Hidden Property)

Value

The name of the file in which the data is to be saved

Purpose

When a file is selected from the save dialog box, a preview of the file contents is displayed.

Example

```
 If MMSaveDialog1.Execute then
  PDBMultiImage1.SaveToFileAsPNG(MMSaveDialog1.FileName);
end;
```

TMultiMediaToolbar (Component)

Toolbar for use with TPMultiMedia. This toolbar can be used on images during design time (Limited) and run time to play multimedia, acquire, open, save, print, cut, copy, paste, zoom, rotate, flip, reset, and stretch images from a file. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette.

MultiMedia (Property)
OpenDialogDir (Property)
OpenDialogFilter (Property)
SaveDialogDir (Property)
SaveDialogFilter (Property)
ShowToolBar (Property)
tbCaption (Property)
tbColorPalatte (Property)
tbCopyImage (Property)
tbCreditMessage (Property)
tbCutImage (Property)
tbFlipImage (Property)
tbImageOpen (Property)
tbImageSave (Property)
tbLeft (Property)
tbMIMediaPlayer (Property)
tbPasteImage (Property)
tbPrintImage (Property)
tbResetImage (Property)
tbRotateImage (Property)
tbScanImage (Property)
tbScrollMessage (Property)
tbStretchImage (Property)
tbStretchImageratio (Property)
tbThumbs (Property)
tbTop (Property)
tbZoomInImage (Property)
tbZoomOutImage (Property)
UseMMOpenDialog (Property)
UseMMSaveDialog (Property)

MultiMedia (Property)

Value

The name of the component that the TMultiMediaToolBar or TPDBMMediaToolBar is going to be used with

Purpose

To connect TPMultiMedia or TPDBMultiMedia with the toolbar

Example

For use with TMultiMediaToolBar

```
    MultiMediaToolBar1.MultiMedia := TPMultiMedia1;
```

For use with TPDBMMediaToolBar

```
    PDBMMediaToolBar1.MultiMedia := PDBMultiMedia1;
```

tbMIMediaPlayer (Property)

Value

True or False

Purpose

This property is a toggle that controls the display of the media player on the tool bar.   If set to false, the feature does not appear on the tool bar

Example

```
PMultiMediaToolBar1.tbMIMediaPlayer := True;
```

TPDBMediaPlayer (Component)

TPDBMediaPlayer   is a component derived from the Delphi MediaPlayer and has exactly all the same functions and properties (SEE Delphi Documentation).   When using the TPDBMediaPlayer, it is not necessary to assign anything to TPDBMediaPlayer directly, TPDBMultiMedia will take care of it.
TPDBMULTIMEDIA will automatically enable/disable the playback of:

AVI:	If video for windows isn't installed;
MOV:	If quicktime for windows isn't installed;
WAV:	If no sound support is installed;
RMI:	If no midi playback drivers are installed;
MID:	If no midi playback drivers are installed;

Thus your program will not crash if no sound card is installed or Video for Windows is not present.

TPDBMImageToolbar (Component)

Toolbar for use with TPDBMultiImage.   This toolbar can be used on images during design time (Limited) and run time to acquire, copy, cut, flip, paste, print, open, reset, rotate, save, stretch, and zoom images from a BLOB field. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette corrector.

tbDBNavigator (Property)

Value

True or False

Purpose

This property is a toggle that controls the display of the database navigator on the tool bar. If a feature is set to false, the feature does not appear on the tool bar. For additional information on the database navigator, see Delphi documentation.

Example

For use with TPDBMImageToolBar

```
PDBMImageToolBar1.tbDBNavigator := True;
```

For use with TPDBMMediaToolBar

```
PDBMMediaToolBar1.tbDBNavigator := True;
```

TPDBMMediaToolbar (Component)

Toolbar for use with TPDBMultiMedia. This toolbar can be used on images during design time (limited) and run time to play multimedia, acquire, copy, cut, flip, paste, print, open, reset, rotate, save, stretch, and zoom images from a BLOB field. In addition, this toolbar includes a scrolling message editor, credit message editor, and an ImageLib palette corrector.

MultiMedia (Property)
OpenDialogDir (Property)
OpenDialogFilter (Property)
SaveDialogDir (Property)
SaveDialogFilter (Property)
ShowToolBar (Property)
tbCaption (Property)
tbColorPalatte (Property)
tbCopyImage (Property)
tbCreditMessage (Property)
tbCutImage (Property)
tbDBNavigator (Property)
tbFlipImage (Property)
tbImageOpen (Property)
tbImageSave (Property)
tbLeft (Property)
tbPasteImage (Property)
tbPDBMediaPlayer (Property)
tbPrintImage (Property)
tbResetImage (Property)
tbRotateImage (Property)
tbScanImage (Property)
tbScrollMessage (Property)
tbStretchImage (Property)
tbStretchImageratio (Property)
tbThumbs (Property)
tbTop (Property)
tbZoomInImage (Property)
tbZoomOutImage (Property)
UseMMOpenDialog (Property)
UseMMSaveDialog (Property)

tbPDBMediaPlayer (Property)

Value

True or False

Purpose

This property is a toggle that controls the display of the media player on the tool bar.   If a feature is set to false, the feature does not appear on the tool bar.

Example

```
PDBMMediaToolBar1.tbPDBMediaPlayer := True;
```

TPDBMultiImage (Component)

Displays and stores BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, TIF and WMF (ICO and WMF are read only) to/from a TBlobField. TPDBMutltimage is the data-aware VCL version of TPMultiImage. TPDBMultiImage is derived from TCustomControl and has the same properties as Delphi's TDBImage with the following additions:

Sample project

Blob.dpr

Common Image Properties and Procedures
Image Format Specific Properties and Procedures

Common Image Properties and Procedures (Database)

Common image properties and procedures are generic in the sense that they apply to all image formats. However, exclusion from this section does not indicate that the procedure is unavailable to all formats. For example: SaveToFileAsJPG is available to all image formats, but is listed in the Image Format Specific Properties and Procedures Section for JPG.

General Image Properties and Procedures (Database)

DataField (Property)

See Delphi Documentation

DataSource (Property)

See Delphi Documentation

ImageDither (Property)

Value

True or False

Purpose

To improve the appearance of an image when the stored resolution is higher (more colors) than the system resolution

Dithering is used in conjunction with the ImageReadRes.

Setting for ImageReadRes	Result
lAutoMatic	Depends on Resolution
lColorTrue	No Dithering
lColor256	Dithering Available
lColor16	Dithering Available
lColorVGA	Dithering Available
lMonochrome	No Dithering

Example

For use with TPDBMultiImage
```
   PDBMultiImage1.ImageDither:=True;
   PDBMultiImage1.ImageReadRes:= lColor256;
```

For use with TPDBMultiMedia
```
   PDBMultiMedia1.ImageDither:=True;
   PDBMultiMedia1.ImageReadRes:= lColor256;
```

For use with TPMultiImage
```
   PMultiImage1.ImageDither:=True;
   PMultiImage1.ImageReadRes:= lColor256;
```

For use with TPMultiMedia
```
   PMultiMedia1.ImageDither:=True;
   PMultiMedia1.ImageReadRes:= lColor256;
```

ImageLibPalette (Property)

Value

True or False

Purpose

This is a "work around" for a problem Delphi has with some 256 color VGA cards.   By setting ImageLib palette to true you will bypass the Delphi way of painting and use the ImageLib painting method which is basically using the windows StretchBlt and BitBlt API calls. ImageLibPalette should always be set to true if you want to ZOOM, Transform, Flip, Rotate and StretchRatio

Example

For use with TPDBMultiImage

```
procedure TViewImageForm.CheckBox4Click(Sender: TObject);
begin
 PDBMultiImage1.ImageLibPalette:=CheckBox4.Checked;
 PDBMultiImage1.Invalidate;
end;
```

For use with TPDBMultiMedia

```
procedure TViewImageForm.CheckBox4Click(Sender: TObject);
begin
 PDBMultiMedia1.ImageLibPalette:=CheckBox4.Checked;
 PDBMultiMedia1.Invalidate;
end;
```

For use with TPMultiImage

```
procedure TViewImageForm.CheckBox4Click(Sender: TObject);
begin
 PMultiImage1.ImageLibPalette:=CheckBox4.Checked;
 PMultiImage1.Invalidate;
end;
```

For use with TPMultiMedia

```
procedure TViewImageForm.CheckBox4Click(Sender: TObject);
begin
 PMultiMedia1.ImageLibPalette:=CheckBox4.Checked;
 PMultiMedia1.Invalidate;
end;
```

ImageReadRes (Property)

Value

lAutoMatic, lColorTrue, lColor256, lColor16, lColorVGA, or lMonochrome

lAutoMatic	Checks the system resolution and chooses the appropriate resolution (New!)
lColorTrue	24-Bit true color (16.7 million colors)
lColor256	8-Bit color (256 colors)
lColor16	4-Bit color (16 colors)
lColorVGA	4-Bit system colors (16 system colors only) (New!)
lMonochrome	1-Bit Monochrome (New!)

Purpose

To force an image to be read in a specific resolution.   Lets assume that the VGA display of a particular computer is 16 colors but the Image is a 256 color image.   This image needs to be color reduced to be shown on the 16 color PC.

Example

For use with TPDBMultiImage
```
DBMultiImage1.ImageDither:=True;
DBMultiImage1.ImageReadRes:= lColor256;
```

+Include DLL95V1 in your USES clause

For use with TPDBMultiMedia
```
DBMultiMedia1.ImageDither:=True;
DBMultiMedia1.ImageReadRes:= lColor256;
```

+Include DLL95V1 in your USES clause

For use with TPMultiImage
```
procedure TViewImageForm.resClick(Sender: TObject);
begin
if Res4System.checked   then
 PMultiImage1.ImageReadRes:=lColorVGA;
if res4.checked   then
 PMultiImage1.ImageReadRes:=lColor16;
if res8.checked   then
 PMultiImage1.ImageReadRes:=lColor256;
if res24.checked then
 PMultiImage1.ImageReadRes:=lColorTrue;
if resAuto.checked then
 PMultiImage1.ImageReadRes:=lAutoMatic;
```

```
MultiImage1.ImageDither:=True;
end;
```

+ Include DLL95V1 in your USES clause

For use with TPMultiMedia

```
procedure TViewImageForm.resClick(Sender: TObject);
begin
if Res4System.checked   then
 PMultiMedia1.ImageReadRes:=lColorVGA;
if res4.checked   then
 PMultiMedia1.ImageReadRes:=lColor16;
if res8.checked   then
 PMultiMedia1.ImageReadRes:=lColor256;
if res24.checked then
 PMultiMedia1.ImageReadRes:=lColorTrue;
if resAuto.checked then
 PMultiMedia1.ImageReadRes:=lAutoMatic;
MultiMedia1.ImageDither:=True;
end;
```

+ Include DLL95V1 in your USES clause

ImageWriteRes (Property)

Value

sAutoMatic, sColorTrue, sColor256, sColor16, sColorVGA, sJpegGray, or sMonochrome

sAutoMatic	Checks the system resolution and chooses the appropriate resolution (New!)
sColorTrue	24-Bit true color (16.7 million colors)
sColor256	8-Bit color (256 colors)
sColor16	4-Bit color (16 colors)
sColorVGA	4-Bit system colors (16 system colors only) (New!)
sJpegGray	Jpeg gray scale (New!)
sMonochrome	1-Bit Monochrome (New!)

Purpose

To force an image to be written in a specific resolution (Upscale or Downscale)

Example

For use with TPDBMultiImage

```
DBMultiImage1.ImageWriteRes:= sColor16;
```

+ **Include DLL95V1 in your USES clause**

For use with TPDBMultiMedia

```
DBMultiMedia1.ImageWriteRes:= sColor16;
```

+ **Include DLL95V1 in your USES clause**

For use with TPMultiImage

```
procedure TViewImageForm.SaveResClick(Sender: TObject);
begin
if SaveResJpegGray.checked   then
 PMultiImage1.ImageWriteRes:=sJpegGray;
if SaveRes4System.checked   then
 PMultiImage1.ImageWriteRes:=sColorVGA;
if Saveres4.checked   then
 PMultiImage1.ImageWriteRes:=sColor16;
if Saveres8.checked   then
 PMultiImage1.ImageWriteRes:=sColor256;
if Saveres24.checked then
 PMultiImage1.ImageWriteRes:=sColorTrue;
if SaveResAuto.checked then
```

```
 PMultiImage1.ImageWriteRes:=sAutoMatic;
end;
```

+ Include DLL95V1 in your USES clause

For use with TPMultiMedia

```
procedure TViewImageForm.SaveResClick(Sender: TObject);
begin
if SaveResJpegGray.checked   then
 PMultiMedia1.ImageWriteRes:=sJpegGray;
if SaveRes4System.checked   then
 PMultiMedia1.ImageWriteRes:=sColorVGA;
if Saveres4.checked   then
 PMultiMedia1.ImageWriteRes:=sColor16;
if Saveres8.checked   then
 PMultiMedia1.ImageWriteRes:=sColor256;
if Saveres24.checked then
 PMultiMedia1.ImageWriteRes:=sColorTrue;
if SaveResAuto.checked then
 PMultiMedia1.ImageWriteRes:=sAutoMatic;
end;
```

+ Include DLL95V1 in your USES clause

MakeThumbNail (HWND); (Procedure)

Value

Window handle of the window to be made into a thumbnail

Purpose

Captures the current frame of a video or image and creates a thumbnail of that frame.

Example

For use with TPDBMultiImage

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
PDBMultiImage1.MakeThumbNail(PMultiMedia1.Handle);
end;
```

+ This procedure should be used to make thumbnails of images that are displayed in any image component.

For use with TPDBMultiMedia

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
PDBMultiMedia1.MakeThumbNail(PDBMultiMedia2.Handle);
end;
```

+ This procedure should be used to make thumbnails of images that are displayed in any image component.

For use with TPMultiImage

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
PMultiImage1.MakeThumbNail(PMultiImage1.Handle);
end;
```

+ This procedure should be used to make thumbnails of images that are displayed in any image component.

For use with TPMultiMedia

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
PMultiMedia1.MakeThumbNail(PMultiMedia1.Handle);
end;
```

+ This procedure should be used to make thumbnails of images that are displayed in any image component.

SaveToFile(filename: TFilename) (Procedure)

Value

The filename of the file to which it is being saved.

Purpose

Saves the current BLOB to a file AS Stored (**No conversion**)

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
 var temp: string;
begin
 temp:=PDBMultiImage1.GetInfoAndType;
 if temp = 'GIF' then begin
   SaveDialog1.filter:='GIF files|*.GIF';
   SaveDialog1.DefaultExt:='GIF';
 end else if temp = 'PCX' then begin
  SaveDialog1.filter:='PCX files|*.PCX';
  SaveDialog1.DefaultExt:='PCX';
 end else if temp = 'PNG' then begin
  SaveDialog1.filter:='PNG files|*.PNG';
  SaveDialog1.DefaultExt:='PNG';
 end else if temp = 'JPG' then begin
  SaveDialog1.filter:='Jpeg files|*.JPG';
  SaveDialog1.DefaultExt:='JPG';
 end else if temp = 'BMP' then begin
  SaveDialog1.filter:='BMP files|*.BMP';
  SaveDialog1.DefaultExt:='BMP';
 end else if temp = SCM' then begin
  SaveDialog1.filter:='SCM files|*. SCM';
  SaveDialog1.DefaultExt:=' SCM ';
 end;
 If SaveDialog1.Execute Then
 PDBMultiImage1.SaveToFile(SaveDialog1.FileName);
end;
```

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
 var temp: string;
begin
 temp:=PDBMultiMedia1.GetInfoAndType;
 if temp = 'GIF' then begin
    SaveDialog1.filter:='GIF files|*.GIF';
   SaveDialog1.DefaultExt:='GIF';
 end else if temp = 'PCX' then begin
  SaveDialog1.filter:='PCX files|*.PCX';
  SaveDialog1.DefaultExt:='PCX';
 end else if temp = 'PNG' then begin
```

```
  SaveDialog1.filter:='PNG files|*.PNG';
  SaveDialog1.DefaultExt:='PNG';
 end else if temp = 'JPG' then begin
  SaveDialog1.filter:='Jpeg files|*.JPG';
  SaveDialog1.DefaultExt:='JPG';
 end else if temp = 'BMP' then begin
  SaveDialog1.filter:='BMP files|*.BMP';
  SaveDialog1.DefaultExt:='BMP';
 end else if temp = SCM' then begin
  SaveDialog1.filter:='SCM files|*. SCM';
  SaveDialog1.DefaultExt:=' SCM ';
 end;
 If SaveDialog1.Execute Then
 PDBMultiMedia1.SaveToFile(SaveDialog1.FileName);
end;
```

VideoToPicture(HWND); (Procedure)

Value

Window handle of the window playing the video

Purpose

Captures the current frame of a video and places it in a TPDBMultiImage, TPDBMultiMedia, TPMultiMedia or TPMultiImage component.

Example

For use with TPDBMultiImage

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
  PDBMultiImage1.VideoToPicture(PMultiMedia1.Handle);
end;
```

For use with TPDBMultiMedia

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
  PDBMultiMedia1.VideoToPicture(PMultiMedia1.Handle);
end;
```

For use with TPMultiImage

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
  PMultiImage1.VideoToPicture(PMultiMedia1.Handle);
end;
```

For use with TPMultiMedia

```
procedure TForm1.CaptureIt(Sender: TObject);
begin
   PMultiMedia1.VideoToPicture(PDBMultiMedia1.Handle);
end;
```

Clipboard

Procedures for use with the clipboard.

CopyToClipboard (Procedure)

Value

None

Purpose

Copy the displayed image to the clipboard

Example

For use with TPDBMultiImage

```
procedure TForm1.Copy1Click(Sender: TObject);
begin
  PDBMultiImage1.CopyToClipboard;
end;
```

+ **CRTL INSERT and CRTL C does the same**

For use with TPDBMultiMedia

```
procedure TForm1.Copy1Click(Sender: TObject);
begin
  PDBMultiMedia1.CopyToClipboard;
end;
```

+ **CRTL INSERT and CRTL C does the same**

For use with TPMultiImage

```
procedure TForm1.Copy1Click(Sender: TObject);
begin
  PMultiImage1.CopyToClipboard;
end;
```

+ **CRTL INSERT and CRTL C does the same**

For use with TPMultiMedia

```
procedure TForm1.Copy1Click(Sender: TObject);
begin
  PMultiMedia1.CopyToClipboard;
end;
```

+ **CRTL INSERT and CRTL C does the same**

CutToClipboard (Procedure)

Value

None

Purpose

Copy the displayed image to the clipboard and erase it.

Example

For use with TPDBMultiImage

```
procedure TForm1.Cut1Click(Sender: TObject);
begin
 PDBMultiImage1.CutToClipboard
end;
```

+ **SHIFT DELETE and CRTL X does the same.**

For use with TPDBMultiMedia

```
procedure TForm1.Cut1Click(Sender: TObject);
begin
 PDBMultiMedia1.CutToClipboard
end;
```

+ **SHIFT DELETE and CRTL X does the same.**

For use with TPMultiImage

```
procedure TForm1.Cut1Click(Sender: TObject);
begin
 PMultiImage1.CutToClipboard
end;
```

+ **SHIFT DELETE and CRTL X does the same.**

For use with TPMultiMedia

```
procedure TForm1.Cut1Click(Sender: TObject);
begin
 PMultiMedia1.CutToClipboard
end;
```

+ **SHIFT DELETE and CRTL X does the same.**

PasteFromClipboard (Procedure)

Value

None

Purpose

Paste an image from the clipboard into the MultiImage

Example

For use with TPDBMultiImage

```
procedure TForm1. Paste1Click(Sender: TObject);
begin
 PDBMultiImage1.PasteFromClipboard;
end;
```

- **SHIFT INSERT and CRTL V does the same**

For use with TPDBMultiMedia

```
procedure TForm1. Paste1Click(Sender: TObject);
begin
 PDBMultiMedia1.PasteFromClipboard;
end;
```

- **SHIFT INSERT and CRTL V does the same**

For use with TPMultiImage

```
procedure TForm1. Paste1Click(Sender: TObject);
begin
 PMultiImage1.PasteFromClipboard;
end;
```

- **SHIFT INSERT and CRTL V does the same**

For use with TPMultiMedia

```
procedure TForm1. Paste1Click(Sender: TObject);
begin
 PMultiMedia1.PasteFromClipboard;
end;
```

- **SHIFT INSERT and CRTL V does the same**

DLL Image Call Back Procedures

The callback procedure is generated by the DLL and has 3 main goals:

1. To show a progress bar to the user;

2. To process windows messages to give other windows programs the chance to do what they have to do; and

3. To inform the DLL that either everything is OK or to cancel the operation.

It's up to you, the application developer to process the application's message loop. You can do this by adding APPLICATION.PROCESSMESSAGES in the callback procedure.   The DLL expects the following type of callback function to be registered:

+ **Changed in version 2.2 from a procedure to a function. Changed in version 2.2.1 from to use a C calling convention.**

TCallBackFunction = function (I: Integer): cdecl Integer;

TCallBackFunction = function (I: Integer): cdecl Integer;

Value

You need to pass a 1 if O.K. or a 0 if you want to cancel

Returns

A value between 1 and 100 which identifies the progress of the image being loaded.

Example and Remarks

There are two things you *MUST* do to add a callback to your app:

1: You need to declare a function of the type above with the EXPORT and cdecl clause:

```
Function ImageLibCallBack(i: integer): integer; cdecl;
export;
begin
 if Application.Terminated then
  Result:=0
 else begin
 Application.ProcessMessages;
 Form1.Gauge1.Progress:=i;
 Result:=1;
 end;
end;
```

2: You need to register the callback to the VCL. The best place to do that is in the FormCreate function:

For use with TPDBMultiImage

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 TPDBMultiImageCallBack:= ImageLibCallBack;
end;
```

For use with TPDBMultiMedia

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 TPDBMultiMediaCallBack:= ImageLibCallBack;
end;
```

For use with TPMultiImage

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 TPMultiImageCallBack:= ImageLibCallBack;
end;
```

For use with TPMultiMedia

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 TPMultiMediaCallBack:= ImageLibCallBack;
end;
```

Image Information

GetInfoAndType: String (Function)

Value

None

Purpose

GetInfoAndType is a very fast function which retrieves image information without actually loading the complete image.

Returns

The function GetInfoAndType returns the extension format of the file stored in the BlobField. In addition, GetInfoAndType will store the following information:

For all filetypes:

Bfiletype	: String;	Return: BMP, CMS, GIF, ICO, JPEG, PCX, PNG, SCM, TIF, and WMF
Bwidth	: Integer;	Return: Width of the image
BHeight	: Integer;	Return: Height of the image
BSize	: Longint	Return: File size in bytes
Bcompression	: String;	Return: Compression method For BMP, GIF, JPEG, PCX, PNG, and TIF only (ICO, CMS, SCM, and WMF will return 0)
Bbitspixel	: Integer;	Return: Bits per Pixel
Bplanes	: Integer;	Return: Planes
Bnumcolors	: Integer;	Return: Number of colors

Example

For use with TPDBMultiImage

```
procedure TForm1.DataSource1DataChange(Sender: TObject;
Field: TField);
begin
 If not PDBMultiImage1.autodisplay then
PDBMultiImage1.GetInfoAndType;
 Edit1.text:='This BLOB image is a
'+TPDBMultiImage1.BFiletype;
 Edit2.text:=IntToStr(PDBMultiImage1.Bwidth);
 Edit3.text:=IntToStr(PDBMultiImage1.BHeight);
 Edit4.text:=IntToStr(PDBMultiImage1.Bbitspixel);
 Edit5.text:=IntToStr(PDBMultiImage1.Bplanes);
 Edit6.text:=IntToStr(PDBMultiImage1.Bnumcolors);
 Edit7.text:=PDBMultiImage1.Bcompression;
 Edit8.text:=IntToStr(PDBMultiImage1.BSize);
end;
```

+ **GetInfoAndType is called automatically by the VCL during an Image load (if autodisplay is true). If no Image is displayed or autodisplay is false you can call this function manually.**

For use with TPDBMultiMedia

```
procedure TForm1.DataSource1DataChange(Sender: TObject;
Field: TField);
begin
 If not PDBMultiImage1.autodisplay then
PDBMultiImage1.GetInfoAndType;
 Edit1.text:='This BLOB image is a
'+TPDBMultiMedia1.BFiletype;
 Edit2.text:=IntToStr(PDBMultiMedia1.Bwidth);
 Edit3.text:=IntToStr(PDBMultiMedia1.BHeight);
 Edit4.text:=IntToStr(PDBMultiMedia1.Bbitspixel);
 Edit5.text:=IntToStr(PDBMultiMedia1.Bplanes);
 Edit6.text:=IntToStr(PDBMultiMedia1.Bnumcolors);
 Edit7.text:=PDBMultiMedia1.Bcompression;
 Edit8.text:=IntToStr(PDBMultiMedia1.BSize);
end;
```

+ **GetInfoAndType is called automatically by the VCL during an Image load (if autodisplay is true). If no Image is displayed or autodisplay is false you can call this function manually.**

Image Manipulation

Image Manipulation properties and procedures allow the user to change the image from its original orientation and configuration.

Canvas (Property)

See Delphi Documentation

FlipImage (Procedure)

Purpose

Will flip the image from left to right (Mirror it)

Example

For use with TPDBMultiImage

```
procedure TTIToolBar.FlipImageClick(Sender: TObject);
begin
  PDBMultiImage.FlipImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio image are set.**

For use with TPDBMultiMedia

```
procedure TTIToolBar.FlipImageClick(Sender: TObject);
begin
  PDBMultiMedia.FlipImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio image are set.**

For use with TPMultiImage

```
procedure TTIToolBar.FlipImageClick(Sender: TObject);
begin
  PMultiImage.FlipImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio image are set.**

For use with TPMultiMedia

```
procedure TTIToolBar.FlipImageClick(Sender: TObject);
begin
  PMultiMedia.FlipImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio image are set.**

ResetImage (Procedure)

Purpose

Resets the Image to its original size and orientation

Example

For use with TPDBMultiImage

```
procedure TTIToolBar.ResetImageClick(Sender: TObject);
begin
   PDBMultiImage.ResetImage;
end;
```

For use with TPDBMultiMedia

```
procedure TTIToolBar.ResetImageClick(Sender: TObject);
begin
   PDBMultiMedia.ResetImage;
end;
```

For use with TPMultiImage

```
procedure TTIToolBar.ResetImageClick(Sender: TObject);
begin
   PMultiImage.ResetImage;
end;
```

For use with TPMultiMedia

```
procedure TTIToolBar.ResetImageClick(Sender: TObject);
begin
   PMultiMedia.ResetImage;
end;
```

RotateImage (Procedure)

Purpose

Will rotate the image 180o (Upside down)

Example

For use with TPDBMultiImage

```
procedure TTIToolBar.RotateImageClick(Sender: TObject);
begin
  PDBMultiImage.RotateImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPDBMultiMedia

```
procedure TTIToolBar.RotateImageClick(Sender: TObject);
begin
  PDBMultiMedia.RotateImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPMultiImage

```
procedure TTIToolBar.RotateImageClick(Sender: TObject);
begin
  PMultiImage.RotateImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPMultiMedia

```
procedure TTIToolBar.RotateImageClick(Sender: TObject);
begin
  PMultiMedia.RotateImage;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

StretchRatio (Property)

Value

True or False

Purpose

Fits the image to the window but maintains the aspect ratio

Example

For use with TPDBMultiImage

```
procedure TViewImageForm.SstretchRatioOnOff(Sender: TObject);
begin
   PDBMultiImage1.Stretchratio:=CheckBox1.Checked;
End.
```

For use with TPDBMultiMedia

```
procedure TViewImageForm.SstretchRatioOnOff(Sender: TObject);
begin
   PDBMultiMedia1.Stretchratio:=CheckBox1.Checked;
End.
```

For use with TPMultiImage

```
procedure TViewImageForm.SstretchRatioOnOff(Sender: TObject);
begin
   PMultiImage1.Stretchratio:=CheckBox1.Checked;
End.
```

For use with TPMultiMedia

```
procedure TViewImageForm.SstretchRatioOnOff(Sender: TObject);
begin
   PMultiMedia1.Stretchratio:=CheckBox1.Checked;
End.
```

TransformImage(Rect : TRect) (Procedure)

Purpose

This call zooms, flips and rotates

Example

This part is the same for all

```
procedure TViewImageForm.MultiImage1MouseDown(Sender:
TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
    {The begin points of zooming/flipping or rotating}
     R.Left:=X;
     R.Top:=Y;
end;
```

For use with TPDBMultiImage

```
procedure TViewImageForm.MultiImage1MouseUp(Sender: TObject;
Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    if Button = mbRight then begin
     {The end points of zooming/flipping or rotating}
     R.Right:=X ;
     R.Bottom:=Y;
     {This call Zooms, flips and rotate}
     PDBMultiImage1.TransformImage(R);
    end;
end;
```

+ **TransformImage must be used with the ImageLib Palette and Stretch or StretchRatio to function properly.**

For use with TPDBMultiMedia

```
procedure TViewImageForm.MultiImage1MouseUp(Sender: TObject;
Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    if Button = mbRight then begin
     {The end points of zooming/flipping or rotating}
     R.Right:=X ;
     R.Bottom:=Y;
     {This call Zooms, flips and rotate}
     PDBMultiMedia1.TransformImage(R);
    end;
end;
```

+ **TransformImage must be used with the ImageLib Palette and Stretch or StretchRatio to function properly.**

For use with TPMultiImage

```
 procedure TViewImageForm.MultiImage1MouseUp(Sender: TObject;
Button: TMouseButton;
   Shift: TShiftState; X, Y: Integer);
begin
    if Button = mbRight then begin
     {The end points of zooming/flipping or rotating}
     R.Right:=X ;
     R.Bottom:=Y;
     {This call Zooms, flips and rotate}
     PMultiImage1.TransformImage(R);
    end;
end;
```

+ **TransformImage must be used with the ImageLib Palette and Stretch or StretchRatio to function properly.**

For use with TPMultiMedia

```
procedure TViewImageForm.MultiImage1MouseUp(Sender: TObject;
Button: TMouseButton;
   Shift: TShiftState; X, Y: Integer);
begin
    if Button = mbRight then begin
     {The end points of zooming/flipping or rotating}
     R.Right:=X ;
     R.Bottom:=Y;
     {This call Zooms, flips and rotate}
     PMultiMedia1.TransformImage(R);
    end;
end;
```

+ **TransformImage must be used with the ImageLib Palette and Stretch or StretchRatio to function properly.**

ZoomIn (Procedure)

Purpose

To increase the size of an Image such that fine details are easier to see.

Example

For use with TPDBMultiImage

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PDBMultiImage.ZoomIn;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPDBMultiMedia

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PDBMultiMedia.ZoomIn;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPMultiImage

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PMultiImage.ZoomIn;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPMultiMedia

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PMultiMedia.ZoomIn;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

ZoomOut (Procedure)

Purpose

To decrease the size of the Image such that more of the image may be displayed in a smaller space.

Example

For use with TPDBMultiImage

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PDBMultiImage.ZoomOut;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPDBMultiMedia

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PDBMultiMedia.ZoomOut;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPMultiImage

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PMultiImage.ZoomOut;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

For use with TPMultiMedia

```
procedure TTIToolBar.ZoomInImageClick(Sender: TObject);
begin
  PMultiMedia.ZoomOut;
end;
```

+ **Will only work when ImageLib Palette and stretch/stretchratio are set.**

Printing MultiImage Images

TPDBMultiImage, TPDBMultiMedia, TPMultiImage, and TPMultiMedia have full printing support to print BMP, GIF, ICO, JPG, PCX, PNG, TIF and WMF. It does this with one procedure call.

PrintMultiImage(X, Y, pWidth, pHeight: Integer) (Procedure)

PrintMultiImage(X, Y, pWidth, pHeight: Integer) (Procedure)

Value

X	The Left position of the image on the paper in pixels
Y	The Top position of the image on the paper in pixels
pWidth	The Right position of the image on the paper in pixels
pHeight	The Bottom position of the image on the paper in pixels

Purpose

PrintMultiImage will Stretch the image on the Printer Canvas and print it.

Example

For use with TPDBMultiImage

```
procedure TForm1.Print1Click(Sender: TObject);
begin
 if PrintDialog1.execute then
  PDBMultiImage1.PrintMultiImage(0, 0, 0, 0);
end;
```

+ **Icons can't be stretched and will be printed in their original size. If pWidth and/or pHeight are 0, then the image will be printed in its original size.**

For use with TPDBMultiMedia

```
procedure TForm1.Print1Click(Sender: TObject);
begin
 if PrintDialog1.execute then
  PDBMultiMedia1.PrintMultiImage(0, 0, 0, 0);
end;
```

+ **Icons can't be stretched and will be printed in their original size. If pWidth and/or pHeight are 0, then the image will be printed in its original size.**

For use with TPMultiImage

```
procedure TForm1.Print1Click(Sender: TObject);
begin
 if PrintDialog1.execute then
  PMultiImage1.PrintMultiImage(0, 0, 0, 0);
end;
```

+ **Icons can't be stretched and will be printed in their original size. If pWidth and/or pHeight are 0, then the image will be printed in its original size.**

For use with TPMultiMedia

```
procedure TForm1.Print1Click(Sender: TObject);
begin
 if PrintDialog1.execute then
  PMultiMedia1.PrintMultiImage(0, 0, 0, 0);
end;
```

+ Icons can't be stretched and will be printed in their original size. If pWidth and/or pHeight are 0, then the image will be printed in its original size.

Text on Image

ImageLib supports placing text on BMP, GIF, ICO, JPG, PCX, PNG, TIF, and WMF image formats.

Font (Property)
Text (Property)
TextLeft (Property)
TextRotate (Property)
TextShadow (Property)
TextShadowColor (Property)
TextTop (Property)
TextTransParent (Property)
Text Example

Font (Property)

See Delphi Documentation

Text (Property)

Value

String

Purpose

To choose the text desired on top of the image

Example

For use with TPDBMultiImage
```
    PDBMultiImage1.Text:=Edit10.Text;
```

For use with TPDBMultiMedia
```
    PDBMultiMedia1.Text:=Edit10.Text;
```

For use with TPMultiImage
```
    PMultiImage1.Text:=Edit10.Text;
```

For use with TPMultiMedia
```
    PMultiMedia1.Text:=Edit10.Text;
```

TextLeft (Property)

Value

Integer

Purpose

Left Position from text on top of image

Example

For use with TPDBMultiImage

```
PDBMultiImage1.TextLeft := 1;
```

For use with TPDBMultiMedia

```
PDBMultiMedia1.TextLeft := 1;
```

For use with TPMultiImage

```
PMultiImage1.TextLeft := 1;
```

For use with TPMultiMedia

```
PMultiMedia1.TextLeft := 1;
```

TextShadow (Property)

Value

True or False

Purpose

Give text on top of Image a shadow effect

Example

For use with TPDBMultiImage
```
    PDBMultiImage1.TextShadow := True;
```

For use with TPDBMultiMedia
```
    PDBMultiMedia1.TextShadow := True;
```

For use with TPMultiImage
```
    PMultiImage1.TextShadow := True;
```

For use with TPMultiMedia
```
    PMultiMedia1.TextShadow := True;
```

TextShadowColor (Property)

Value

TColor

Purpose

Shadow color from text on top of Image

Example

For use with TPDBMultiImage

```
PDBMultiImage1.TextShadowColor := ‘clBlack’;
```

For use with TPDBMultiMedia

```
PDBMultiMedia1.TextShadowColor := ‘clBlack’;
```

For use with TPMultiImage

```
PMultiImage1.TextShadowColor := ‘clBlack’;
```

For use with TPMultiMedia

```
PMultiMedia1.TextShadowColor := ‘clBlack’;
```

TextRotate (Property)

Value

Integer

Purpose

Angle in degrees to rotate the text on top of image

Example

For use with TPDBMultiImage

```
PDBMultiImage1.TextRotate:=0;
```

For use with TPDBMultiMedia

```
PDBMultiMedia1.TextRotate:=0;
```

For use with TPMultiImage

```
PMultiImage1.TextRotate:=0;
```

For use with TPMultiMedia

```
PMultiMedia1.TextRotate:=0;
```

TextTop (Property)

Value

Integer

Purpose

Top Position from text on top of image

Example

For use with TPDBMultiImage
```
    PDBMultiImage1.TextTop:=MultiImage1.ClientRect.Top+20;
```

For use with TPDBMultiMedia
```
    PDBMultiMedia1.TextTop:=MultiImage1.ClientRect.Top+20;
```

For use with TPMultiImage
```
    PMultiImage1.TextTop:=MultiImage1.ClientRect.Top+20;
```

For use with TPMultiMedia
```
    PMultiMedia1.TextTop:=MultiImage1.ClientRect.Top+20;
```

TextTransParent (Property)

Value

True or False

Purpose

Background of text on top of image Transparent or not

Example

For use with TPDBMultiImage
```
PDBMultiImage1.TextTransParent := True;
```

For use with TPDBMultiMedia
```
PDBMultiMedia1.TextTransParent := True;
```

For use with TPMultiImage
```
PMultiImage1.TextTransParent := True;
```

For use with TPMultiMedia
```
PMultiMedia1.TextTransParent := True;
```

Text Example

For use with TPDBMultiImage

```
procedure TViewImageForm.BitBtn7Click(Sender: TObject);
begin
  PDBMultiImage1.Text:=Edit10.Text;
  PDBMultiImage1.TextShadow:=True;
  PDBMultiImage1.TextLeft:=1;
 PDBMultiImage1.TextTop:=MultiImage1.ClientRect.Top+20;
  PDBMultiImage1.TextRotate:=0;
  PDBMultiImage1.TextTransParent:=True;
end;
```

For use with TPDBMultiMedia

```
procedure TViewImageForm.BitBtn7Click(Sender: TObject);
begin
  PDBMultiMedia1.Text:=Edit10.Text;
  PDBMultiMedia1.TextShadow:=True;
  PDBMultiMedia1.TextLeft:=1;
 PDBMultiMedia1.TextTop:=MultiImage1.ClientRect.Top+20;
  PDBMultiMedia1.TextRotate:=0;
  PDBMultiMedia1.TextTransParent:=True;
end;
```

For use with TPMultiImage

```
procedure TViewImageForm.BitBtn7Click(Sender: TObject);
begin
  PMultiImage1.Text:=Edit10.Text;
  PMultiImage1.TextShadow:=True;
  PMultiImage1.TextLeft:=1;
 PMultiImage1.TextTop:=MultiImage1.ClientRect.Top+20;
  PMultiImage1.TextRotate:=0;
  PMultiImage1.TextTransParent:=True;
end;
```

For use with TPMultiMedia

```
procedure TViewImageForm.BitBtn7Click(Sender: TObject);
begin
  PMultiMedia1.Text:=Edit10.Text;
  PMultiMedia1.TextShadow:=True;
  PMultiMedia1.TextLeft:=1;
 PMultiMedia1.TextTop:=MultiImage1.ClientRect.Top+20;
  PMultiMedia1.TextRotate:=0;
  PMultiMedia1.TextTransParent:=True;
end;
```

Image Format Specific Properties and Procedures (Database)

BMP Database Update and File Write

SaveToFileAsBMP(FN:TFilename) (Procedure)
UpdateAsBMP (Property)

SaveToFileAsBMP(FN:TFilename) (Procedure)

Value

Filename of the file to which it is being saved

Purpose

Save the displayed image to a BMP file

Example

For use with TPDBMultiImage

```
procedure TForm1.BitButton8Click(Sender: TObject);
begin
PDBMultiImage1.ImageWriteRes := Color256;
if SaveDialog2.execute then
   PDBMultiImage1.SaveToFileAsBMP(SaveDialog2.FileName); end;
```

For use with TPDBMultiMedia

```
procedure TForm1.BitButton8Click(Sender: TObject);
begin
PDBMultiMedia1.ImageWriteRes := Color256;
if SaveDialog2.execute then
   PDBMultiMedia1.SaveToFileAsBMP(SaveDialog2.FileName); end;
```

UpdateAsBMP (Property)

Value

True or False

Purpose

To store a new image or to update the displayed image.   If True then the BLOB image will be updated to a BMP.

Example

For use with TPDBMultiImage

```
procedure TForm1.UpdateAsBMP(Sender: TObject);
begin
 PDBMultiImage1.UpdateAsBMP:=True;
 PDBMultiImage1.PastefromClipboard;
 Table1.Post;
end;
```

+ Image must be displayed

For use with TPDBMultiMedia

```
procedure TForm1.UpdateAsBMP(Sender: TObject);
begin
   PDBMultiMedia1.UpdateAsBMP:=True;
   PDBMultiMedia1.PastefromClipboard;
   Table1.Post;
end;
```

+ Image must be displayed

CMS Database Update and File Write

Read information below before proceeding to the topics listed.

CreateCreditMessage (Function)
FreeMsg (Procedure)
NewCreditMessage (Procedure)

Credit messages are TPDBMultiImages created by the VCL on the fly.
The following are stored in the BLOB:

MsgFont	: TFont;	the message's font
MsgSpeed	: Integer;	the scrolling speed 1 is fast 10 is slow
MsgBkgrnd	: TColor;	the background color
CreditBoxList	: TStringList;	the credit messages in a stringlist

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason for this is that an application can have only one Application.OnIdle event. This event then needs to be shared by other events which may need a trigger. Note that other VCLs may use a Trigger. Make sure that their OnIdle procedure does not destroy the MultiImage trigger. In your application you need to add a procedure to the private clauses called, Trigger (See example below)

Example

```
type
 TForm1 = class(TForm)
private
  Procedure Trigger(Sender: TObject; Var Done: Boolean);
public
```

In the form create you will assign Trigger to the onIdle event.

```
procedure Form1.FormCreate(Sender: TObject);
begin
  Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

For use with PDBMultiImage

```
Procedure TForm1.Trigger(Sender: TObject; Var Done: Boolean);
begin
  PDBMultiImage1.Trigger;
end;
```

For use with PDBMultiMedia

```
    Procedure TForm1.Trigger(Sender: TObject; Var Done: Boolean);
    begin
      PDBMultiMedia1.Trigger;
end;
```

CreateCreditMessage (Function)

Purpose

CreateCreditMessage will open the Message editor. The user can create his/her own Credit message and save this message to a file with a CMS extension as the default.

Return

True or False

Example

For use with TPDBMultiImage

```
procedure TBtnBottomDlg.BitBtn7Click(Sender: TObject);
begin
Table1.Append;
 If PDBMultiImage1.CreateCreditMessage then
    Table1.Post
 else
    Table1.Cancel;
end;
```

+ **To save current BLOB message to a file use SaveToFile..**

For use with TPDBMultiMedia

```
procedure TBtnBottomDlg.BitBtn7Click(Sender: TObject);
begin
Table1.Append;
 If PDBMultiMedia1.CreateCreditMessage then
    Table1.Post
 else
    Table1.Cancel;
end;
```

+ **To save current BLOB message to a file use SaveToFile..**

FreeMsg (Procedure)

Value

None

Purpose

Removes the current message and then assigns the picture to Null

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
 PDBMultiImage1.FreeMsg;
end;
```

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
 PDBMultiMedia1.FreeMsg;
end;
```

For use with TPMultiImage

```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
 PMultiImage1.FreeMsg;
end;
```

For use with TPDBMultiImage

```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
 PMultiMedia1.FreeMsg;
end;
```

NewCreditMessage (Procedure)

Value

None

Purpose

To initiate a new message.   This shows messages created on the fly

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  PDBMultiImage1.FreeMsg;
  PDBMultiImage1.CreditBoxList.Clear;
  PDBMultiImage1.CreditBoxList.Add(' ImageLib');
  PDBMultiImage1.CreditBoxList.Add(' Another fine product
  of');
  PDBMultiImage1.CreditBoxList.Add(' SKYLINE TOOLS');
  PDBMultiImage1.CreditBoxList.Add(' Programming: Kevin
  Adams');
  PDBMultiImage1.CreditBoxList.Add(' Programming: Jan
  Dekkers');
  PDBMultiImage1.CreditBoxList.Add(' Artwork & PR: Jillian
  Pinsker');     PDBMultiImage1.MsgFont.Name:='Arial';
  PDBMultiImage1.MsgFont.Size:=-40;
  PDBMultiImage1.MsgFont.Style:=[fsitalic, fsbold];
  PDBMultiImage1.MsgFont.Color:=clWhite;
  PDBMultiImage1.MsgColor:=clNavy;
  PDBMultiImage1.MsgSpeed:=1;
  PDBMultiImage1.NewCreditMessage;
end;
```

GIF Database Update and File Write

SaveToFileAsGIF(FN: TFilename) (Procedure)
UpdateAsGIF (Property)

SaveToFileAsGIF(FN: TFilename) (Procedure)

Value

The filename of the image being saved as a GIF.

Purpose

To save the Image displayed as a GIF file.

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiImage1.ImageWriteRes:= Color16;
 If SaveDialog2.Execute then
    PDBMultiImage1. SaveToFileAsGIF(SaveDialog2.Filename);
end;
```

+ Image must be displayed.

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiMedia1.ImageWriteRes:= Color16;
 If SaveDialog2.Execute then
    PDBMultiMedia1. SaveToFileAsGIF(SaveDialog2.Filename);
end;
```

+ Image must be displayed.

UpdateAsGIF (Property)

Value

True or False

Purpose

To store a new image or to update the displayed image.   If True then the BLOB image will be updated to a GIF.

Example

For use with TPDBMultiImage

```
procedure TForm1.UpdateAsGIF(Sender: TObject);
begin
   PDBMultiImage1.UpdateAsGIF:=True;
   PDBMultiImage1.PastefromClipboard;
   Table1.Post;
end;
```

+ Image must be displayed

For use with TPDBMultiMedia

```
procedure TForm1.UpdateAsGIF(Sender: TObject);
begin
   PDBMultiMedia1.UpdateAsGIF:=True;
   PDBMultiMedia1.PastefromClipboard;
   Table1.Post;
end;
```

+ Image must be displayed

JPG Database Update and File Write

JPegSaveQuality (Property)

Value

0...100

Purpose

0 is poor and 100 excellent. We normally use 25 to have a reasonable quality with a 1/10 saving in size.

Example

For use with TPDBMultiImage
```
PDBMultiImage1.JPegSaveQuality:=25;
```

For use with TPDBMultiMedia
```
PDBMultiMedia1.JPegSaveQuality:=25;
```

For use with TPMultiImage
```
PMultiImage1.JPegSaveQuality:=25;
```

For use with TPMultiMedia
```
PMultiMedia1.JPegSaveQuality:=25;
```

JPegSaveSmooth (Property)

Value

0...100

Purpose

0 is no smoothing and 100 is full smoothing. Because of the lossy compression of JPegs, an image might be too hard, smoothing can give it a better look.

Example

For use with TPDBMultiImage

```
PDBMultiImage1.JPegSaveSmooth:=5;
```

For use with TPDBMultiMedia

```
PDBMultiMedia1.JPegSaveSmooth:=5;
```

For use with TPMultiImage

```
PMultiImage1.JPegSaveSmooth:=5;
```

For use with TPMultiMedia

```
PMultiMedia1.JPegSaveSmooth:=5;
```

SaveToFileAsJpg(FN: TFilename) (Procedure)

Value

The filename for the image being saved

Purpose

To saves the image displayed as a JPG file.

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
  PDBMultiImage1.JPegSaveQuality:=25;
  PDBMultiImage1.JPegSaveSmooth:=5;
  If SaveDialog2.Execute then
  PDBMultiImage1.SaveToFileAsJpeg(SaveDialog2.Filename);
end;
```

+ Image must be displayed

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
  PDBMultiMedia1.JPegSaveQuality:=25;
  PDBMultiMedia1.JPegSaveSmooth:=5;
  If SaveDialog2.Execute then
  PDBMultiMedia1.SaveToFileAsJpeg(SaveDialog2.Filename);
end;
```

+ Image must be displayed

UpdateAsJPG (Property)

Value

True or False

Purpose

To store a new image or to update the displayed image.   If True then the BLOB image will be updated to a JPG.

Example

For use with TPDBMultiImage

```
procedure TForm1.UpdateAsJPG(Sender: TObject);
begin
   PDBMultiImage1.UpdateAsJPG:=True;
   PDBMultiImage1.PastefromClipboard;
   Table1.Post;
end;
```

+ Image must be displayed

For use with TPDBMultiMedia

```
procedure TForm1.UpdateAsJPG(Sender: TObject);
begin
   PDBMultiMedia1.UpdateAsJPG:=True;
   PDBMultiMedia1.PastefromClipboard;
   Table1.Post;
end;
```

+ Image must be displayed

PCX Database Update and File Write

SaveToFileAsPCX(FN: TFilename) (Procedure)

Value

The filename of the image being saved as a PCX.

Purpose

To save the Image displayed as a PCX file.

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiImage1.ImageWriteRes:= lColorTrue;
 If SaveDialog2.Execute then
 PDBMultiImage1.SaveToFileAsPCX(SaveDialog2.Filename);
end;
```

+ **Image must be displayed**

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiMedia1.ImageWriteRes:= lColorTrue;
 If SaveDialog2.Execute then
 PDBMultiMedia1.SaveToFileAsPCX(SaveDialog2.Filename);
end;
```

+ **Image must be displayed**

UpdateAsPCX (Property)

Value

True or False

Purpose

To store a new image or to update the displayed image.   If True then the BLOB image will be updated to a PCX.

Example

For use with TPDBMultiImage

```
procedure TForm1.UpdateAsPCX (Sender: TObject);
begin
 PDBMultiImage1.UpdateAsPCX:=True;
 PDBMultiImage1.PastefromClipboard;
 Table1.Post;
end;
```

+ Image must be displayed

For use with TPDBMultiMedia

```
procedure TForm1.UpdateAsPCX (Sender: TObject);
begin
 PDBMultiMedia1.UpdateAsPCX:=True;
 PDBMultiMedia1.PastefromClipboard;
 Table1.Post;
end;
```

+ Image must be displayed

PNG Database Update and File Write

PNGInterLaced (Property)

Value

True or False

Purpose

Save the PNG Interlaced.   CompuServe's new format to replace GIF.

Example

For use with TPDBMultiImage
```
    PDBMultiImage1.PNGInterLaced := True;
```

For use with TPDBMultiMedia
```
    PDBMultiMedia1.PNGInterLaced := True;
```

For use with TPMultiImage
```
    PMultiImage1.PNGInterLaced := True;
```

For use with TPMultiMedia
```
    PMultiMedia1.PNGInterLaced := True;
```

SaveToFileAsPNG(FN: TFilename) (Procedure)

Value

The filename of the image being saved as PNG.

Purpose

To save the Image displayed as a PNG file.

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiImage1.ImageWriteRes:= sColor256;
 If SaveDialog2.Execute then
     PDBMultiImage1. SaveToFileAsPNG(SaveDialog2.Filename);
end;
```

+ Image must be displayed.

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiMedia1.ImageWriteRes:= sColor256;
 If SaveDialog2.Execute then
     PDBMultiMedia1. SaveToFileAsPNG(SaveDialog2.Filename);
end;
```

+ Image must be displayed.

UpdateAsPNG (Property)

Value

True or False

Purpose

To store a new image or to update the displayed image.   If True then the BLOB image will be updated to a PNG.

Example

For use with TPDBMultiImage

```
procedure TForm1.UpdateAsPNG (Sender: TObject);
begin
 PDBMultiImage1.UpdateAsPNG:=True;
 PDBMultiImage1.PastefromClipboard;
 Table1.Post;
end;
```

+ Image must be displayed

For use with TPDBMultiMedia

```
procedure TForm1.UpdateAsPNG (Sender: TObject);
begin
 PDBMultiMedia1.UpdateAsPNG:=True;
 PDBMultiMedia1.PastefromClipboard;
 Table1.Post;
end;
```

+ Image must be displayed

Rich Text Format (RTF)

Rich Text Format (RTF) support is only provided in the 32-bit version of ImageLib. ImageLib can store and read back RTF BLOBs from any of the databases supported by Delphi.

LoadFromFile(FN: TFilename) (Procedure)

Value

The filename of the image to be read

Purpose

To read a file that contains any image format supported by TPDBMultiImage.   This is the only way to read a RTF file into TPDBMultiImage.

Example

```
PDBMultiImage1.LoadfromFile := ‘C:\RTF\EXAMPLE.RTF’;
```

+ RTF support available in the 32 bit version only.

RichTBarLeft (Property)

Value

Integer

Purpose

Left position of the RichTools toolbar

Example

```
PDBMulitImage1.RichTBarLeft := 10;
```

+ RTF support available in the 32 bit version only.

RichTBarTop (Property)

Value

Integer

Purpose

Top position of the RichTools toolbar

Example

```
PDBMulitImage1.RichTBarTop := 10;
```

+ RTF support available in the 32 bit version only.

RichTools (Property)

Value

True or False

Purpose

Toggles the RichTools toolbar between on and off.   The toolbar will only pop up when a RTF is active and the RichTools property is set to True.

Example

```
PDBMulitImage1.RichTools := True;
```

+ RTF support available in the 32 bit version only.

SaveToFileAsRTF (Procedure)

Value

The filename of the RTF being saved.

Purpose

To save the RTF to a file.

Example

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 If SaveDialog2.Execute then
      PDBMultiImage1. SaveToFileAsRTF(SaveDialog2.Filename);
end;
```

+ **RTF must be displayed. RTF support available in the 32 bit version only.**

SCM Database Update and File Write

Read information below before proceeding to the topics listed.

Scrolling messages are TPDBMultiImages created by the VCL on the fly. The average BLOB of a scrolling message is only 200 bytes. The following are stored in the BLOB:

MsgText	: String;	The message text
MsgFont	: Tfont;	The message font
MsgColor	: Tcolor;	Background color
MsgSpeed	: Integer;	Scrolling Speed

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason is an application can have only one Application.OnIdle event. This event then needs to be shared with other events which may need a trigger. Note that other VCLs may also need a Trigger. Make sure that their OnIdle procedure does not destroy the MultiImage trigger. In your application you need to add a procedure to the private clauses called, Trigger (See example Below):

Example

```
type
 TForm1 = class(TForm)
private
 Procedure Trigger(Sender: TObject; Var Done: Boolean);
public
end;
```

In the FormCreate you will assign Trigger to the onIdle event.

```
procedure Form1.FormCreate(Sender: TObject);
begin
 Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

```
 Procedure TForm1.Trigger(Sender: TObject; Var Done:
Boolean);
begin
 DBMultiImage1.Trigger;
end;
```

CreateMessage (Function)

Value

None

Purpose

CreateMessage will open the Message editor. The user can create his/her own scrolling message and store it in the BlobField.

Returns

True if successful otherwise false

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn13Click(Sender: TObject);
begin
 Table1.Append;
 If PDBMultiImage1.CreateMessage then
      Table1.Post
 else
 Table1.Cancel;
end;
```

+ To save current BLOB message to a file use SaveToFile.

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn13Click(Sender: TObject);
begin
 Table1.Append;
 If PDBMultiMedia1.CreateMessage then
      Table1.Post
 else
 Table1.Cancel;
end;
```

+ To save current BLOB message to a file use SaveToFile.

NewMessage (Procedure)

Value

None

Purpose

Initiate a new messages and show messages created on the fly

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 PDBMultiImage1.MsgText:='ImageLib 3.0 A great tool ‘;
  PDBMultiImage1.MsgFont.Name:='Arial';
  PDBMultiImage1.MsgFont.Size:=-40;
  PDBMultiImage1.MsgFont.Style:=[fsitalic,fsbold];
  PDBMultiImage1.MsgFont.Color:=clWhite;
  PDBMultiImage1.MsgColor:=clNavy;
  PDBMultiImage1.MsgSpeed:=1;
  PDBMultiImage1.NewMessage;
end;
```

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
 PDBMultiMedia1.MsgText:='ImageLib 3.0 A great tool ‘;
  PDBMultiMedia1.MsgFont.Name:='Arial';
  PDBMultiMedia1.MsgFont.Size:=-40;
  PDBMultiMedia1.MsgFont.Style:=[fsitalic,fsbold];
  PDBMultiMedia1.MsgFont.Color:=clWhite;
  PDBMultiMedia1.MsgColor:=clNavy;
  PDBMultiMedia1.MsgSpeed:=1;
  PDBMultiMedia1.NewMessage;
end;
```

Trigger (Procedure)

See Information CMS or SCM Image formats for additional about establishing the trigger for initial use.

Value

None

Purpose

Trigger message movements

Example

For use with TPDBMultiImage

```
Procedure TForm1.Trigger(Sender: TObject; Var Done: Boolean);
begin
 PDBMultiImage1.Trigger;
end;
```

For use with TPDBMultiMedia

```
Procedure TForm1.Trigger(Sender: TObject; Var Done: Boolean);
begin
 PDBMultiMedia1.Trigger;
end;
```

For use with TPMultiImage

```
Procedure TForm1.Trigger(Sender: TObject; Var Done: Boolean);
begin
 PMultiImage1.Trigger;
end;
```

For use with TPMultiMedia

```
Procedure TForm1.Trigger(Sender: TObject; Var Done: Boolean);
begin
 PMultiMedia1.Trigger;
end;
```

TIF Database Update and File Write

SaveToFileAsTIF(FN: TFilename) (Procedure)

Value

The filename of the image being saved as TIFF.

Purpose

To save the Image displayed as a TIFF file.

Example

For use with TPDBMultiImage

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiImage1.ImageWriteRes:= sColor256;
 If SaveDialog2.Execute then
     PDBMultiImage1. SaveToFileAsTIF(SaveDialog2.Filename);
end;
```

+ Image must be displayed.

For use with TPDBMultiMedia

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
 PDBMultiMedia1.ImageWriteRes:= sColor256;
 If SaveDialog2.Execute then
     PDBMultiMedia1. SaveToFileAsTIF(SaveDialog2.Filename);
end;
```

+ Image must be displayed.

UpdateAsTIF (Property)

Value

True or False

Purpose

To store a new image or to update the displayed image. If True then the BLOB image will be updated to a TIF.

Example

For use with TPDBMultiImage

```
procedure TForm1.UpdateAsTIF (Sender: TObject);
begin
 PDBMultiImage1.UpdateAsTIF:=True;
 PDBMultiImage1.PastefromClipboard;
 Table1.Post;
end;
```

+ Image must be displayed

For use with TPDBMultiMedia

```
procedure TForm1.UpdateAsTIF (Sender: TObject);
begin
 PDBMultiMedia1.UpdateAsTIF:=True;
 PDBMultiMedia1.PastefromClipboard;
 Table1.Post;
end;
```

+ Image must be displayed

TPDBMultiMedia (Component)

TPDBMultiMedia has all the same properties and functions as TPDBMultiImage. However, in addition to the storing and displaying of BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, TIF, and WMF (ICO and WMF are read only) from a TBlobField, it also stores and plays AVI, MOV, MID, WAV and RMI multimedia files. TPDBMediaPlayer is derived from Delphi's MediaPlayer and has the same functions and properties. When using the TPDBMediaPlayer you do not need to assign anything to TPDBMediaPlayer directly, TPDBMultiMedia will take care of it. TPDBMULTIMEDIA will automatically enable/disable the playback of:

AVI:	If video for windows isn't installed;
MOV:	If quicktime for windows isn't installed;
WAV:	If no sound support is installed;
RMI:	If no midi playback drivers are installed;
MID:	If no midi playback drivers are installed.

Thus your program will not crash if no sound card is installed or Video for Windows is not present. Again, all the properties from TPDBMultiImage are there and we added the following:

Sample project

MMBLOB.dpr

Properties for use with TPDBMultiMedia
Properties Common to all TPDBMultiMedia and TPDBMultiImage Components

Properties for use with TPDBMultiMedia

GetMultiMediaExtensions (Function)
PathForTempFile (Property)
TempMov (Property)
TempAVI (Property)
TempWAV (Property)

GetMultiMediaExtensions (Function)

Value

None

Purpose

This function will return all multimedia extensions from the computer running your application and those supported by TPDBMultiMedia in the filter format used by the filedialog.

Return

String - MultiMedia extensions in filter format

Example

```
procedure TBtnBottomDlg.BitBtn1Click(Sender: TObject);
begin
 OpenDialog1.filter:=PDBMultiMedia1.GetMultiMediaExtensions;
 if OpenDialog1.Execute then begin
  Table1.Append;
  PDBMultiMedia1.LoadfromFile(OpenDialog1.FileName);
  Table1.Post;
 end;
end;
```

+ **Run the example file MMBLOB.DPR. You will notice that the Append MM dialogbox contains all the Multimedia supported by the VCL and your PC.**

PathForTempFile (Property)

Value

PathName

Purpose

TPDBMULTIMEDIA saves its AVI, MID, MOV, RMI, and WAV BLOBs to a temporary file before it is played and then deletes the temporary file. The reason is that most multimedia BLOBs are too large to be played from memory. Your application might be distributed and executed from a CD. In order to write a temporary file you need to supply a directory and drive.

Example

```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin
   PDBMultiMedia1.PathForTempFile:='C:\TEMP';
end;
```

+ **BMP, CMS, GIF, JPG, PCX, PNG, SCM TIF and WMF BLOBs are not written to a temporary file but expanded directly into memory. If directory or drive does not exist it defaults to C:\.**

TempMov (Property)

Value

Filename

Default

$$$.MOV

Purpose

TDBMULTIMEDIA saves its MOV BLOBs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example

```
DBMULTIMEDIA1.TempMov:='$TEMP$.MOV';
```

Remark

+ **Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.**

TempAVI (Property)

Value

Filename

Default

$$$.AVI

Purpose

TDBMULTIMEDIA saves its AVI BLOBs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example

```
DBMULTIMEDIA1.TempAvi:='$TEMP$.AVI';
```

Remark

+ **Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.**

TempWAV (Property)

Value

Filename

Default

$$$.WAV

Purpose

TDBMULTIMEDIA saves its WAV BLOBs first to a temporary file before it is played and then deletes the temporary file. This property holds the name of the temporary file.

Example

```
DBMULTIMEDIA1.TempWav:='$TEMP$.WAV';
```

+ **Since the Delphi MultiMediaPlayer is extension sensitive the extension can't be changed.**

Properties Common to all TPDBMultiMedia and TPDBMultiImage Components

The following is a list of properties that are available to MultiMedia and MultiImage Components. These properties are identical to those covered in the section for TPDBMultiImage. (SEE Chapter on TPDBMultiImage for additional information)

Canvas (Property)
CopyToClipboard (Procedure)
CreateCreditMessage (Function)
CreateMessage (Function)
CutToClipboard (Procedure)
DataField (Property)
DataSource (Property)
FlipImage (Procedure)
Font (Property)
FreeMsg (Procedure)
GetInfoAndType (Function)
GLOBALPALETTE (Variable)
ImageDither (Property)
ImageLibPalette (Property)
ImageReadRes (Property)
ImageWriteRes (Property)
JPegSaveQuality (Property)
JPegSaveSmooth (Property)
LoadFromFile (Procedure)
MakeThumbNail (Procedure)
NewCreditMessage (Procedure)
NewMessage (Procedure)
PasteFromClipboard (Procedure)
PrintMultiImage (Procedure)
PNGInterLaced (Property)
ResetImage (Procedure)
RichTBarLeft (Property)
RichTools (Property)
RichTBarTop (Property)
RotateImage (Procedure)
SaveToFile (Procedure)
SaveToFileAsBMP (Procedure)
SaveToFileAsGIF (Procedure)
SaveToFileAsJpg (Procedure)

SaveToFileAsPCX (Procedure)
SaveToFileAsPNG (Procedure)
SaveToFileAsRTF (Procedure)
SaveToFileAsTIF (Procedure)
StretchRatio (Property)
TCallBackFunction (Function)
Text (Property)
TextLeft (Property)
TextRotate (Property)
TextShadow (Property)
TextShadowColor (Property)
TextTop (Property)
TextTransParent (Property)
TransformImage (Procedure)
Trigger (Procedure)
UpdateAsBMP (Property)
UpdateAsGIF (Property)
UpdateAsJPG (Property)
UpdateAsPCX (Property)
UpdateAsPNG (Property)
UpdateAsTIF (Property)
VideoToPicture (Procedure)
ZoomIn (Procedure)
ZoomOut (Procedure)

Value

Varies (SEE Chapter on TPDBMultiImage for additional information)

Purpose

Varies (SEE Chapter on TPDBMultiImage for additional information)

Example

```
TPDBMultiMedia1.PropertyName := Varies (SEE Chapter on
TPDBMultiImage for additional information)
```

TPMultiImage

Displays and stores BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, and WMF (ICO and WMF are read only) to/from a file. TPMultiImage is a data-aware VCL. TPMultiImage is derived from TCustomControl and has the same properties as Delphi’s TImage with the following additions.

Sample Projects

• im_cvrt.dpr	Converting images (example)
• scrollim.dpr	Scrolling messages (example)
• simple.dpr	A few lines of code (example)
• viewph.dpr	Extensive example

MultiImage has the same properties as Delphi's TImage with the following additions:

Reading and displaying images for all Image formats

Common Image Properties and Procedures
Image Format Specific Properties and Procedures

Common Image Properties and Procedures (File)

General Image Properties and Procedures (File)

DefSaveFileName (Property)
GLOBALPALETTE (Variable)
ImageDither (Property)
ImageLibPalette (Property)
ImageReadRes (Property)
ImageWriteRes (Property)
MakeThumbNail (Procedure)
Resource File, Loading Images From
VideoToPicture(HWND); (Procedure)

DefSaveFileName (Property)

Value

Filename of the BMP, GIF, PCX, PNG, JPG, and TIF which needs to be saved.

Purpose

To store a filename before the file is actually saved.   You can use this as a filename scratchpad.

Example

For use with TPMultiImage

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then begin
  PMultiImage1.JPegSaveQuality:=25;
  PMultiImage1.JPegSaveSmooth:=5;
  PMultiImage1.DefSaveFileName:=SaveDialog1.FileName;
  PMultiImage1.SaveAsJpg('');
 end;
end;
```

+ Changed from JPGSaveFileName in version 2.0

For use with TPMultiMedia

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then begin
  PMultiMedia1.JPegSaveQuality:=25;
  PMultiMedia1.JPegSaveSmooth:=5;
  PMultiMedia1.DefSaveFileName:=SaveDialog1.FileName;
  PMultiMedia1.SaveAsJpg('');
 end;
end;
```

+ Changed from JPGSaveFileName in version 2.0

Variable GLOBALPALETTE : HPalette (Variable)

Value

Hpallette: The Palette to be used

Purpose

To set a global palatte for all images

Example

For use with TPDBMultiImage

```
Variable GLOBALPALETTE : HPalette;
GLOBALPALETTE:=0;
PDBMultiImage1.ImageName:='Beet.jpg';
```

+ This image will use its own palette.

```
Variable GLOBALPALETTE : HPalette;
GLOBALPALETTE:=MultiImage1.Picture.Bitmap.Palette;
PDBMultiImage2.ImageName:='clown.jpg';
```

+ This image will use the palette of PDBMultiImage1

```
Variable GLOBALPALETTE : HPalette;
GLOBALPALETTE:=0;
PDBMultiImage3.ImageName:='earth.bmp';
```

+ This image will use its own palette.

For use with TPDBMultiMedia

```
Variable GLOBALPALETTE : HPalette;
GLOBALPALETTE:=0;
PDBMultiMedia1.ImageName:='Beet.jpg';
```

+ This image will use its own palette.

```
Variable GLOBALPALETTE : HPalette;
GLOBALPALETTE:=MultiImage1.Picture.Bitmap.Palette;
PDBMultiMedia2.ImageName:='clown.jpg';
```

+ This image will use the palette of PDBMultiMedia1

```
Variable GLOBALPALETTE : HPalette;
```

```
    GLOBALPALETTE:=0;
    PDBMultiMedia3.ImageName:='earth.bmp';
```

+ This image will use its own palette.

For use with TPMultiImage

```
    Variable GLOBALPALETTE : HPalette;
    GLOBALPALETTE:=0;
    PMultiImage1.ImageName:='Beet.jpg';
```

+ This image will use its own palette.

```
    Variable GLOBALPALETTE : HPalette;
    GLOBALPALETTE:=MultiImage1.Picture.Bitmap.Palette;
    PMultiImage2.ImageName:='clown.jpg';
```

+ This image will use the palette of PMultiImage1

```
    Variable GLOBALPALETTE : HPalette;
    GLOBALPALETTE:=0;
    PMultiImage3.ImageName:='earth.bmp';
```

+ This image will use its own palette.

For use with TPMultiMedia

```
    Variable GLOBALPALETTE : HPalette;
    GLOBALPALETTE:=0;
    PMultiMedia1.ImageName:='Beet.jpg';
```

+ This image will use its own palette.

```
    Variable GLOBALPALETTE : HPalette;
    GLOBALPALETTE:=MultiImage1.Picture.Bitmap.Palette;
    PMultiMedia2.ImageName:='clown.jpg';
```

+ This image will use the palette of PMultiMedia1

```
    Variable GLOBALPALETTE : HPalette;
    GLOBALPALETTE:=0;
    PMultiMedia3.ImageName:='earth.bmp';
```

+ This image will use its own palette.

Resource File, Loading Images From

ImageLib can load images from an application resource file. This is done by adding the extension .RES to the filename. Please study the example code below for more information on loading images from resource files.

Purpose

To Load images from an application resource file

Example

```
procedure TViewImageForm.BitBtn5Click(Sender: TObject);
        {Load a BMP image from any executable or dll. Note that the dll or
exe needs to be
         loaded in order to find its module handle}
begin

        {Indicate in which program the BMP Lives. For instance:
         MultiImage1.ResProgName:=' ' means in this executable (RES
        (Resource file))}
        {Dont enter a path name};

        MultiImage1.ResProgName:=''; {this executable linked in
        BLIT8.RES}

        {Name of the resource. Note that the .res indicate to the vcl that it
        is a resource BMP
         but your resource BMP is called FRIDGE.   Incase the resource is a
        number put e.g:
         MultiImage1.ImageName:='98255.RES';}

        MultiImage1.ImageName:='FRIDGE.RES';

end;

procedure TViewImageForm.BitBtn6Click(Sender: TObject);
begin

        {Indicate in which program the BMP Lives. For instance:
         MultiImage1.ResProgName:=' ' means in this executable (RES
        (Resource file))}
        {Dont enter a path name};

        MultiImage1.ResProgName:='DELPHI.EXE';    {Bmp is in
        delphi.exe}
        {Name of the resource. Note that the .res indicate to the vcl that it
        is a resource BMP but your resource BMP is called athena. (Delphi
        logo) Incase the resource is a number put e.g:
        MultiImage1.ImageName:='98255.RES';}

        MultiImage1.ImageName:='ATHENA.RES';

    end;
```

Hint: See UIMAGE.PAS for an example.

Image Information (File)

GetInfoAndType(filename: TFilename): Boolean (Function)

Value

Filename of the image

Purpose

GetInfoAndType is a very fast function which retrieves image information without actually loading the complete image.

Returns

True, if successful, otherwise False. GetInfoAndType will store the following information:

For all filetypes:

Bfiletype	: String;	Return: BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, WMF
Bwidth	: Integer;	Return: Width of the image
BHeight	: Integer;	Return: Height of the image
BSize	: Longint,	Return: File size in bytes
Bcompression	: String;	Return: Compression method

For BMP, GIF, JPEG, PNG, PCX only (ICO, SCM, CMS, WMF will return o)

Bbitspixel	: Integer;	Return: Bits per Pixel
Bplanes	: Integer;	Return: Planes
Bnumcolors	: Integer;	Return: Number of colors

+ **GetInfoAndType is called automatically by the VCL during an Image load. If no Image is displayed you can call this function manually.**

Example

For use with TPMultiImage

```
procedure TForm1.DisPlayInfo(filename: TFilename);
begin
 if GetInfoAndType(filename) then begin
 Edit1.Text:=IntToStr(PMultiImage1.Bwidth);
 Edit2.Text:=IntToStr(PMultiImage1.BHeight);
 Edit3.Text:=IntToStr(PMultiImage1.Bbitspixel);
 Edit4.Text:=IntToStr(PMultiImage1.Bplanes);
 Edit5.Text:=IntToStr(PMultiImage1.Bnumcolors);
 Edit6.Text:=PMultiImage1.BFileType;
 Edit7.Text:=PMultiImage1.Bcompression;
```

```
     Edit8.Text:=IntToStr(PMultiImage1.BSize)+ bytes';
     end;
    end;
```

For use with TPMultiMedia

```
    procedure TForm1.DisPlayInfo(filename: TFilename);
    begin
     if GetInfoAndType(filename) then begin
     Edit1.Text:=IntToStr(PMultiMedia1.Bwidth);
     Edit2.Text:=IntToStr(PMultiMedia1.BHeight);
     Edit3.Text:=IntToStr(PMultiMedia1.Bbitspixel);
     Edit4.Text:=IntToStr(PMultiMedia1.Bplanes);
     Edit5.Text:=IntToStr(PMultiMedia1.Bnumcolors);
     Edit6.Text:=PMultiMedia1.BFileType;
     Edit7.Text:=PMultiMedia1.Bcompression;
     Edit8.Text:=IntToStr(PMultiMedia1.BSize)+ bytes';
     end;
    end;
```

Image Format Specific Properties and Procedures (File)

BMP File Read and Write

Read From File Examples

To read/display a BMP image you can use either ImageLib or Delphi

Using the Delphi way

This example uses two picture components. When the form first appears, two bitmaps are loaded into the picture components and stretched to fit the size of the components. To try this code, substitute names of bitmaps you have available. The following code will load BMP, WMF and ICO Images

For use with TPMultiImage

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 PMultiImage1.Stretch:= True;
 PMultiImage2.Stretch:= True;
 PMultiImage 1.Picture.LoadFromFile('BITMAP1.BMP');
 PMultiImage 2.Picture.LoadFromFile('BITMAP2.BMP');
end;
```

For use with TPMultiMedia

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 PMultiMedia1.Stretch:= True;
 PMultiMedia2.Stretch:= True;
 PMultiMedia1.Picture.LoadFromFile('BITMAP1.BMP');
 PMultiMedia2.Picture.LoadFromFile('BITMAP2.BMP');
end;
```

Using the ImageLib way

This example uses two picture components. When the form first appears, two bitmaps are loaded into the picture components and stretched to fit the size of the components. To try this code, substitute names of bitmaps you have available. The following code will load BMP, GIF, ICO, JPG, PCX, SCM and WMF Images.

For use with TPMultiImage

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 PMultiImage1.Stretch:= True;
 PMultiImage2.Stretch:= True;
 PMultiImage 1.ImageName:='BITMAP1.BMP';
 PMultiImage 2.ImageName:='BITMAP2.BMP';
end;
```

For use with TPMultiMedia

```
procedure TForm1.FormCreate(Sender: TObject);
```

```
    begin
     PMultiMedia1.Stretch:= True;
     PMultiMedia2.Stretch:= True;
     PMultiMedia1.ImageName:='BITMAP1.BMP';
     PMultiMedia2.ImageName:='BITMAP2.BMP';
end;
```

Write to File Examples

To Save a BMP image you can use either ImageLib or Delphi

Using the Delphi way

This example uses two picture components.

For use with TPMultiImage

```
    begin
      PMultiImage1.Picture.SaveToFile('BITMAP1.BMP');
      PMultiImage2.Picture.SaveToFile('BITMAP2.BMP');
    end;
```

For use with TPMultiMedia

```
    begin
      PMultiMedia1.Picture.SaveToFile('BITMAP1.BMP');
      PMultiMedia2.Picture.SaveToFile('BITMAP2.BMP');
    end;
```

Read/Write a BPM the ImageLib way

See the following procedure/property description.

SaveAsBMP(FN:TFilename) (Procedure)

SaveAsBMP(FN:TFilename) (Procedure)

Value

Filename of the file to which it is being saved

Purpose

Save the displayed image to a BMP file

Example

For use with TPMultiImage

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then begin
 PMultiImage1.DefSaveFileName:=SaveDialog1.FileName;
 PMultiImage1.SaveAsBMP('');
 end;
end;
```

or

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then
 PMultiImage1.SaveAsBMP(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

For use with TPMultiMedia

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then begin
 PMultiMedia1.DefSaveFileName:=SaveDialog1.FileName;
 PMultiMedia1.SaveAsBMP('');
 end;
end;
```

or

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then
 PMultiMedia1.SaveAsBMP(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

CMS File Read and Write

Read information below before proceeding to the topics listed.

Credit messages are TPMultiImages created by the VCL on the fly. The average filesize of a Credit message (CMS) is only 200 bytes. The maximum size is 64Kb. Stored in the CMS file are:

MsgFont :	TFont;	the message's font
MsgSpeed :	Integer;	the scrolling speed ;1 is fast 10 is slow
MsgColor :	TColor;	the background color
CreditBoxList :	TStringList;	the credit messages in a stringlist

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason for this is that an application can have only one Application.OnIdle event. This event then needs to be shared by other events which may need a trigger. Note that other VCLs could also use a Trigger. Make sure that their OnIdle procedure does not destroy the MultiImage trigger. In your application, you need to add a procedure to the private clauses called, trigger (See example below).

Example

```
type
 TForm1 = class(TForm)
procedure FormCreate(Sender: TObject);
private
  Procedure Trigger(Sender: TObject; Var Done: Boolean);
public
end;
```

In the form create you will assign Trigger to the onIdle event.

```
procedure Form1.FormCreate(Sender: TObject);
begin
  Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

```
Procedure Form1.Trigger(Sender: TObject; Var Done:
Boolean);
begin
  PMultiImage3.Trigger;
  PMultiImage2.Trigger;
  PMultiImage1.Trigger;
end;
```

+ For an extensive example load the project Scrollim.dpr

For use with TPMultiMedia

```
Procedure Form1.Trigger(Sender: TObject; Var Done:
Boolean);
begin
  PMultiMedia3.Trigger;
  PMultiMedia2.Trigger;
  PMultiMedia1.Trigger;
end;
```

+ For an extensive example load the project Scrollim.dpr

CreateCreditMessage(MessagePath:String;AutoLoad:boolean) (Procedure)

Value

MessagePath	The initial path displayed in the save dialog.
AutoLoad	True or False. If true, message is displayed after saving it.

Purpose

CreateCreditMessage will open the Message editor. The user can create his own Credit message and save this message to a file with a CMS extension as default.

Example

For use with TPMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
   PMultiImage1.CreateCreditMessage
   (ExtractFilePath(Application.Exename), True);
end;
```

For use with TPMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
   PMultiMedia1.CreateCreditMessage
   (ExtractFilePath(Application.Exename), True);
end;
```

ImageName (Property)

Value

Filename of the image which needs to be displayed.

Purpose

BMP, CMS, GIF, ICO, JPG, PCX, PNG, SCM, TIF, and WMF images are loaded with one single line of code.

Example

For use with TPMultiImage

```
PMultiImage1.ImageName:='C:\ CLOWN.BMP';
PMultiImage1.ImageName:='C:\ CLOWN.CMS';
PMultiImage1.ImageName:='C:\ CLOWN.GIF';
PMultiImage1.ImageName:='C:\ CLOWN.ICO';
PMultiImage1.ImageName:='C:\ CLOWN.JPG';
PMultiImage1.ImageName:='C:\ CLOWN.PCX';
PMultiImage1.ImageName:='C:\ CLOWN.PNG';
PMultiImage1.ImageName:='C:\ CLOWN.SCM';
PMultiImage1.ImageName:='C:\ CLOWN.TIF';
```

or

```
PMultiImage1.ImageName:='C:\ CLOWN.WMF';
```

For use with TPMultiMedia

```
PMultiMedia1.ImageName:='C:\ CLOWN.BMP';
PMultiMedia1.ImageName:='C:\ CLOWN.CMS';
PMultiMedia1.ImageName:='C:\ CLOWN.GIF';
PMultiMedia1.ImageName:='C:\ CLOWN.ICO';
PMultiMedia1.ImageName:='C:\ CLOWN.JPG';
PMultiMedia1.ImageName:='C:\ CLOWN.PCX';
PMultiMedia1.ImageName:='C:\ CLOWN.PNG';
PMultiMedia1.ImageName:='C:\ CLOWN.SCM';
PMultiMedia1.ImageName:='C:\ CLOWN.TIF';
```

or

```
PMultiMedia1.ImageName:='C:\ CLOWN.WMF';
```

NewCreditMessage (Procedure)

Value

None

Purpose

Initiate a new message. Ideal to show messages created on the fly.

Example

For use with TPMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  PMultiImage1.FreeMsg;
  PMultiImage1.CreditBoxList.Clear;
  PMultiImage1.CreditBoxList.Add(' ImageLib');
  PMultiImage1.CreditBoxList.Add(' Another fine product
  of');
  PMultiImage1.CreditBoxList.Add(' SKYLINE TOOLS');
  PMultiImage1.CreditBoxList.Add(' Programming: Kevin
  Adams');
  PMultiImage1.CreditBoxList.Add(' Programming: Jan
  Dekkers');
  PMultiImage1.CreditBoxList.Add(' Artwork & PR: Jillian
  Pinsker');  PMultiImage1.MsgFont.Name:='Arial';
  PMultiImage1.MsgFont.Size:=-40;
  PMultiImage1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiImage1.MsgFont.Color:=clWhite;
  PMultiImage1.MsgColor:=clNavy;
  PMultiImage1.MsgSpeed:=1;
  PMultiImage1.NewCreditMessage;
end;
```

For use with TPMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  PMultiMedia1.FreeMsg;
  PMultiMedia1.CreditBoxList.Clear;
  PMultiMedia1.CreditBoxList.Add(' ImageLib');
  PMultiMedia1.CreditBoxList.Add(' Another fine product
  of');
  PMultiMedia1.CreditBoxList.Add(' SKYLINE TOOLS');
  PMultiMedia1.CreditBoxList.Add(' Programming: Kevin
  Adams');
  PMultiMedia1.CreditBoxList.Add(' Programming: Jan
  Dekkers');
  PMultiMedia1.CreditBoxList.Add(' Artwork & PR: Jillian
  Pinsker');
  PMultiMedia1.MsgFont.Name:='Arial';
  PMultiMedia1.MsgFont.Size:=-40;
  PMultiMedia1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiMedia1.MsgFont.Color:=clWhite;
  PMultiMedia1.MsgColor:=clNavy;
  PMultiMedia1.MsgSpeed:=1;
  PMultiMedia1.NewCreditMessage;
end;
```

SaveCurrentCreditMessage(MessageName: TFileName) (Procedure)

Value

MessageName	The filename to which the message is being saved.

Purpose

Save the message with values of: (These are the values of the current message being displayed).

PMultiImage1.CreditBoxList	: TStringList;	The credit messages in a stringlist
PMultiImage1.MsgFont	: Tfont;	The message font
PMultiImage1.MsgColor	: Tcolor;	Background color
PMultiImage1.MsgSpeed	: Integer;	Scrolling Speed

Example

For use with TPMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  PMultiImage1.FreeMsg;
  PMultiImage1.CreditBoxList.Clear;
  PMultiImage1.CreditBoxList.Add(' ImageLib');
  PMultiImage1.CreditBoxList.Add(' Another fine product
  of');
  PMultiImage1.CreditBoxList.Add(' SKYLINE TOOLS');
  PMultiImage1.CreditBoxList.Add(' Programming: Kevin
  Adams');
  PMultiImage1.CreditBoxList.Add(' Programming: Jan
  Dekkers');
  PMultiImage1.CreditBoxList.Add(' Artwork & PR: Jillian
  Pinsker');  PMultiImage1.MsgFont.Name:='Arial';
  PMultiImage1.MsgFont.Size:=-40;
  PMultiImage1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiImage1.MsgFont.Color:=clWhite;
  PMultiImage1.MsgColor:=clNavy;
  PMultiImage1.MsgSpeed:=1;
if SaveDialog1.Execute then
  PMultiImage1.SaveCurrentCreditMessage(SaveDialog1.File
  Name);
end;
```

+ **MessageFont.Name, MessageFont.Size, MessageFont.Style and MessageFont.Color could also be defined using a font dialog box.**

For use with TPMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
```

```
  PMultiMedia1.FreeMsg;
  PMultiMedia1.CreditBoxList.Clear;
  PMultiMedia1.CreditBoxList.Add(' ImageLib');
  PMultiMedia1.CreditBoxList.Add(' Another fine product
  of');
  PMultiMedia1.CreditBoxList.Add(' SKYLINE TOOLS');
  PMultiMedia1.CreditBoxList.Add(' Programming: Kevin
  Adams');
  PMultiMedia1.CreditBoxList.Add(' Programming: Jan
  Dekkers');
  PMultiMedia1.CreditBoxList.Add(' Artwork & PR: Jillian
  Pinsker');
  PMultiMedia1.MsgFont.Name:='Arial';
  PMultiMedia1.MsgFont.Size:=-40;
  PMultiMedia1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiMedia1.MsgFont.Color:=clWhite;
  PMultiMedia1.MsgColor:=clNavy;
  PMultiMedia1.MsgSpeed:=1;
if SaveDialog1.Execute then
  PMultiMedia1.SaveCurrentCreditMessage(SaveDialog1.File
  Name);
end;
```

+ MsgFont.Name, MsgFont.Size, MsgFont.Style and MsgFont.Color could also be defined using a font dialog box.

GIF File Read and Write

+ **Gif uses LZW compression which is patented by Unisys. On CompuServe use GO PICS to obtain information about the Unisys patents. In order to use the ImageLib's GIF read and write, you need to buy a license from Unisys. By using ImageLib’s GIF Read and Write features you acknowledge that SkyLine has notified you about the LZW patent and do not hold SkyLine liable for any legal action.**

ImageName (Property)
SaveAsGIF(FN:TFilename) (Procedure)

SaveAsGIF(FN:TFilename) (Procedure)

Value

Filename of the file to which it is being saved

Purpose

Save the displayed image to a GIF file

Example

For use with TPMultiImage

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 PMultiImage1.SaveAsGIF(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

For use with TPMultiMedia

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 PMultiMedia1.SaveAsGIF(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

ICO File Read

See BMP file read and write in this chapter

JPG File Read and Write

SaveAsJpg(FN: TFilename) (Procedure)

Value

Filename of the file to which it is being saved

Purpose

Save the displayed image to a Jpeg file

Example

For use with TPMultiImage

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then begin
  PMultiImage1.JPegSaveSmooth:=5;
  PMultiImage1.JPegSaveQuality:=25;
  PMultiImage1.SaveAsJpg(SaveDialog1.FileName);
 end;
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

For use with TPMultiMedia

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then begin
  PMultiMedia1.JPegSaveSmooth:=5;
  PMultiMedia1.JPegSaveQuality:=25;
  PMultiMedia1.SaveAsJpg(SaveDialog1.FileName);
 end;
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

PCX File Read and Write

SaveAsPCX(FN: TFilename) (Procedure)

Value

Filename of the file being saved

Purpose

Save the displayed image to a PCX file

Example

For use with TPMultiImage

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 MultiImage1.SaveAsPCX(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

For use with TPMultiMedia

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 MultiMedia1.SaveAsPCX(SaveDialog1.FileName);
end;
```

+ An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName

PNG File Read and Write

SaveAsPNG(FN: TFilename) (Procedure)

Value

Filename of the file being saved

Purpose

Save the displayed image to a PNG file

Example

For use with TPMultiImage

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 PMultiImage1.SaveAsPNG(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

For use with TPMultiMedia

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 PMultiMedia.SaveAsPNG(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

SCM File Read and Write

Read information below before proceeding to the topics listed.

CreateMessage (Procedure)
FreeMsg (Procedure)
ImageName (Property)
NewMessage (Procedure)
SaveCurrentMessage (Procedure)
Trigger (Procedure)

Scrolling messages are TPMultiImages created by the VCL on the fly. The average file size of a Scrolling message (SCM) is only 200 bytes. Stored in the SCM file are:

MsgText	: String;	The message text.
MsgFont	: Tfont;	The message font.
MsgColor	: Tcolor;	Background color.
MsgSpeed	: Integer;	Scrolling Speed.

The VCL does NOT have its own moving engine. You "the programmer" must trigger the movements. The reason for this is that an application can have only one Application.OnIdle event. This event then needs to be subdivided to other events which may need an Idle event. Note that other VCLs could also use a Trigger. Make sure that their OnIdle procedure does not destroy the MultiImage trigger. In your application you need to add a procedure to the private clauses called, Trigger (See example below).

Example

```
type
 TForm1 = class(TForm)
procedure FormCreate(Sender: TObject);
private
  Procedure Trigger(Sender: TObject; Var Done: Boolean);
public
end;
```

In the form create you will assign Trigger to the onIdle event.

procedure Form1.FormCreate(Sender: TObject);

```
begin
  Application.OnIdle:=Trigger;
end;
```

The procedure trigger will then trigger the VCL:

For use with TPMultiImage

```
Procedure Form1.Trigger(Sender: TObject; Var Done:
Boolean);
begin
  PMultiImage3.Trigger;
  PMultiImage2.Trigger;
  PMultiImage1.Trigger;
end;
```

+ For an extensive example load the project Scrollim.dpr.

For use with TPMultiMedia

```
Procedure Form1.Trigger(Sender: TObject; Var Done:
Boolean);
begin
  PMultiMedia3.Trigger;
  PMultiMedia2.Trigger;
  PMultiMedia1.Trigger;
end;
```

+ For an extensive example load the project Scrollim.dpr.

CreateMessage(MessagePath:String;AutoLoad: Boolean) (Procedure)

Value

MessagePath	The initial path displayed in the save dialog.
AutoLoad	True or False. If true, message is displayed after saving it.

Purpose

CreateMessage will open the Message editor. The user can create his own scrolling message and save this message to a file with an SCM extension (default).

Example

For use with TPMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
PMultiImage1.CreateMessage(ExtractFilePath(Application.Ex
ename), True);
end;
```

For use with TPMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
PMultiMedia1.CreateMessage(ExtractFilePath(Application.Ex
ename), True);
end;
```

NewMessage (Procedure)

Value

None

Purpose

Initiate a new message.   This shows messages created on the fly.

Example

For use with TPMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  PMultiImage1.MsgText:='ImageLib 3.0 A great tool ';
  PMultiImage1.MsgFont.Name:='Arial';
  PMultiImage1.MsgFont.Size:=-40;
  PMultiImage1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiImage1.MsgFont.Color:=clWhite;
  PMultiImage1.MsgColor:=clNavy;
  PMultiImage1.MsgSpeed:=1;
  PMultiImage1.NewMessage;
end;
```

For use with TPMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
  PMultiMedia1.MsgText:='ImageLib 3.0 A great tool ';
  PMultiMedia1.MsgFont.Name:='Arial';
  PMultiMedia1.MsgFont.Size:=-40;
  PMultiMedia1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiMedia1.MsgFont.Color:=clWhite;
  PMultiMedia1.MsgColor:=clNavy;
  PMultiMedia1.MsgSpeed:=1;
  PMultiMedia1.NewMessage;
end;
```

SaveCurrentMessage(MessageName: TFileName) (Procedure)

Value

The filename of the message is being saved

Purpose

Saves the message with values for: (These are the values of the current message being displayed).

PMultiImage1.MsgText	: String;	The Message text.
PMultiImage1.MsgFont	: Tfont;	The message font
PMultiImage1.MsgColor	: Tcolor;	Background color
PMultiImage1.MsgSpeed	: Integer;	Scrolling Speed

Example

For use with TPMultiImage

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
MultiImage1.MsgText:='ImageLib A great tool ‘;
  PMultiImage1.MsgFont.Name:='Arial';
  PMultiImage1.MsgFont.Size:=-40;
  PMultiImage1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiImage1.MsgFont.Color:=clWhite;
  PMultiImage1.MsgColor:=clNavy;
  PMultiImage1.MsgSpeed:=1;
  if SaveDialog1.Execute then
  PMultiImage1.SaveCurrentMessage(SaveDialog1.FileName);
end;
```

+ **MsgFont.Name, MsgFont.Size, MsgFont.Style and MsgFont.Color could also be defined using a Fontdialog box.**

Example

```
PMultiImage1. MsgFont:= FontDialog1.Font;
```

For use with TPMultiMedia

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
MultiImage1.MsgText:='ImageLib A great tool ‘;
  PMultiMedia1.MsgFont.Name:='Arial';
  PMultiMedia1.MsgFont.Size:=-40;
  PMultiMedia1.MsgFont.Style:=[fsitalic, fsbold];
  PMultiMedia1.MsgFont.Color:=clWhite;
  PMultiMedia1.MsgColor:=clNavy;
  PMultiMedia1.MsgSpeed:=1;
  if SaveDialog1.Execute then
  PMultiMedia1.SaveCurrentMsg(SaveDialog1.FileName)
end;
```

+ **MsgFont.Name, MsgFont.Size, MsgFont.Style and MsgFont.Color could also be defined using a Fontdialog box.**

Example

```
PMultiMedia1. MsgFont:= FontDialog1.Font;
```

TIF File Read and Write

SaveAsTif(FN: TFilename) (Procedure)

Value

Name of file being saved

Purpose

Save the displayed image to a TIF file

Example

For use with TPMultiImage

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 PMultiImage1.SaveAsTif(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

For use with TPMultiMedia

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
 if SaveDialog1.execute then
 PMultiMedia1.SaveAsTif(SaveDialog1.FileName);
end;
```

+ **An active image needs to be displayed on the form. If no filename is passed it will use the DefSaveFileName**

TifSaveCompress (Property)

Values

CCITT, NONE, LZW, or PACKBITS

CCITT	Compression Method
NONE	No compression
LZW	Compression Method
PACKBITS	Compression Method

+ **LZW compression is patented by Unisys. On CompuServe use GO PICS to obtain information about the Unisys patent. By using LZW compression you acknowledge that SkyLine has notified you about the LZW patent and does not hold SkyLine liable for any legal actions.**

Purpose

Compression methods to save a TIFF Image

Example

For use with TPMultiImage

```
procedure TViewImageForm.TiffComboChange(Sender: TObject);
begin
{Set the compression method to save TIFFs}
if TiffCombo.Text ='NONE' then
 PMultiImage1.TifSaveCompress:=sNONE;
if TiffCombo.Text ='CCITT' then
 PMultiImage1.TifSaveCompress:=sCCITT;
if TiffCombo.Text ='LZW' then
 PMultiImage1.TifSaveCompress:=sLZW;
if TiffCombo.Text ='PACKBITS' then
 PMultiImage1.TifSaveCompress:=sPACKBITS;
end;
```

+ **DLL95V1 must be included in the clauses of the unit calling this function**

For use with TPMultiMedia

```
procedure TViewImageForm.TiffComboChange(Sender: TObject);
begin
{Set the compression method to save TIFFs}
if TiffCombo.Text ='NONE' then
 PMultiMedia1.TifSaveCompress:=sNONE;
if TiffCombo.Text ='CCITT' then
 PMultiMedia1.TifSaveCompress:=sCCITT;
if TiffCombo.Text ='LZW' then
 PMultiMedia1.TifSaveCompress:=sLZW;
```

```
if TiffCombo.Text ='PACKBITS' then
 PMultiMedia1.TifSaveCompress:=sPACKBITS;
end;
```

+ **DLL95V1 must be included in the clauses of the unit calling this function**

WMF File Read

See BMP Read and Write in this chapter

TPMultiMedia (Component)

TPMultiMedia has all the same properties and functions as TPMultiImage (SEE Chapter on TPMultiImage). However, in addition to the storing and displaying of BMP, CMS, GIF, ICO, JPEG, PCX, PNG, SCM, TIFF, and WMF (ICO and WMF are read only) from a file; TPMultiMedia also stores and plays AVI, MOV, MID, WAV, and RMI multimedia files. When using the TMIMediaPlayer, it is not necessary to assign anything to the TMIMediaPlayer directly, TPMultiMedia will take care of it. TPMULTIMEDIA will automatically enable/disable the playback of:

AVI:	If video for windows isn't installed;
MOV:	If quicktime for windows isn't installed;
WAV:	If no sound support is installed;
RMI:	If no midi playback drivers are installed;
MID:	If no midi playback drivers are installed;

Thus your program will not crash if no sound card is installed or Video for Windows is not present. Again, all the properties from TPMultiImage are there and we added the following:

+ **The Delphi MultiMediaPlayer is extension sensitive, thus extensions can’t be changed.**

Properties for use with TPMultiMedia
Properties Common to all TPMultiMedia and TPMultiImage Components

Properties for use with TPMultiMedia

GetMultiMediaExtensions: String (Function)

GetMultiMediaExtensions: String (Function)

Value

None

Purpose

This function will return all multimedia extensions from the computer running your application and those supported by TPMultiMedia in the filter format used by the filedialog.

Example

```
procedure TBtnBottomDlg.BitBtn1Click(Sender: TObject);
begin
OpenDialog1.filter:=TPMultiMedia1.GetMultiMediaExtensions;
 if OpenDialog1.Execute then begin
  Table1.Append;
 TPMultiMedia1.LoadfromFile(OpenDialog1.FileName);
 Table1.Post;
 end;
end;
```

Properties Common to all TPMultiMedia and TPMultiImage Components

The following is a list of properties that are available to MultiMedia and MultiImage Components. These properties are identical to those covered in the section for TPMultiImage. (SEE Chapter on TPMultiImage for additional information)

Canvas (Property)
CopyToClipboard (Procedure)
CreateCreditMessage (Procedure)
CreateMessage (Procedure)
CutToClipboard (Procedure)
DefSaveFileName (Property)
FlipImage (Procedure)
Font (Property)
FreeMessage (Procedure)
FreeMsg (Procedure)
GetInfoAndType (Function)
GLOBALPALETTE (Variable)
ImageDither (Property)
ImageLibPalette (Property)
ImageName (Property)
ImageReadRes (Property)
ImageWriteRes (Property)
JPegSaveQuality (Property)
JPegSaveSmooth (Property)
MakeThumbNail (Procedure)
NewCreditMessage (Procedure)
NewMessage (Procedure)
PasteFromClipboard (Procedure)
PrintMultiImage (Procedure)
PNGInterLaced (Property)
SaveAsPCX(Procedure)
ResetImage (Procedure)
RotateImage (Procedure)
SaveAsBMP (Procedure)
SaveAsGIF (Procedure)
SaveAsJpg (Procedure)
SaveAsPNG (Procedure)
SaveAsTif (Procedure)
SaveCurrentCreditMessage (Procedure)

Value

Varies (SEE Chapter on TPMultiImage for additional information)

Purpose

Varies (SEE Chapter on TPMultiImage for additional information)

Example

```
TPMultiMedia1.PropertyName := Varies (SEE Chapter on
TPMultiImage for additional information)
```

Twain Support for Delphi

The following are functions and procedures for use with Twain Compliant Devices.

HasTwain (Hwind : HWnd) : Boolean (function)
ScanImage (Hwind : HWnd) (Procedure)
SelectScanner (Hwind : HWnd) (Procedure)

HasTwain (Hwind : HWnd) : Boolean (function)

Purpose

This function will check the windows environment to see if Twain is available. The function looks for the "TWAIN.DLL" file and the source manager. If an error occurs, ensure the Twain software was correctly installed for use with your Twain compliant device.

Value

HWnd:	Windows Handle

Return

Boolean:	True or False

Example

```
Var
AppCanScan:boolean
initialization
    AppCanScan:=HasTwain(Handle) ;
end.
```

+ This call can take some time thus it should only be called once on application startup. Set the value equal to a boolean so the application can refer to that value to enable/disable the scan buttons.

ScanImage (Hwind : HWnd) (Procedure)

Purpose

This procedure is used to acquire an image from a twain compliant device. This call will start the selected twain source (if no Source has been selected the default is used). The source manager for the twain device will allow the user to control that device.

Value

HWnd:	Windows Handle

Example

```
procedure TViewImageForm.Scan1Click(Sender: TObject);
begin
    {Scan an image using a twain scanner}
    MultiImage1.ScanImage(Handle);
end;
```

SelectScanner (Hwind : HWnd) (Procedure)

Purpose

This procedure starts the Twain source selection window. The user can choose the source manager they wish to use.   Selectsource is for systems that have more that one Twain Source loaded.   Selecting "Cancel" in the source selection window will not cause an error.

Value

HWnd:	Windows Handle

Example

```
procedure TViewImageForm.SelectScanner1Click(Sender:
TObject);
begin
    {Select a twain scanner source}
    MultiImage1.SelectScanner(Handle);
end;
```

C Programming and the ImageLib DLL

ImageLib is an inexpensive way to add BMP, GIF, JPEG, PCX, PNG and TIFF graphic formats to your C applications. The ImageLib DLL supports the reading and writing of images from memory or file and supports the use of an optional callback function. The callback can provide a progress display of read and write functions. In addition, read functions can be canceled in progress.

The DLL also provides functions to retrieve information about an image in memory or a file without reading the whole image. The Image Information functions return the type of image, compression, width, height, bits per pixel, number of planes, and number of colors. The memory functions of the ImageLib DLL are specifically designed to support database BLOB operations. All calls return error codes and the DLL will optional display error messages. The error codes refer to error text strings located in a string table resource inside the DLL.

The ImageLib DLL supports Device Dependent Bitmaps(DDB) or Device Independent Bitmaps(DIB) in the reading and writing of images. The DLL contains a sophisticated color quantization engine that can be used when reading or writing images. When reading an image, settings can be used to ensure the resolution you specify is used and is independent of the input image. If the developer wants all images to be passed back as 256 color 8 bit dithered images then all bitmaps passed back will be 8 bit whether they where originally 24 bit or 4 bit. The color quantizer is designed to produce the best image possible at the desired resolution. When writing an image, the developer may specify the resolution of the image to be written (resolution must be valid for image type).

The ImageLib DLL is Twain compliant and can be used with Twain compliant devices such as scanners, Snappy (TM), and other digital cameras. The DLL includes a SelectSource call to select a Twain Source and an AquireImage call to invoke the vendor's Twain Source Manager. ImageLib's Twain will work with 16 bit and 32 bit Twain Sources.

+ ImageLib includes examples for Borland C++ 4.0x and for Microsoft VC++ 1.5x. To find these examples go to the directory where you installed ImageLib then select the appropriate subdirectory. Note! The install program has the option not to install these subdirectories. If you cannot find them, run the install program again. The Borland examples are OWL examples and are located in the Borland (bcc) subdirectory. The Microsoft Visual C++ demo is an MFC example is located in the (mvc) subdirectory.

Essential Information (C Apps.)
Error Strings and Foreign Languages (C Apps.)
Image Information (C Apps.)
Image Manipulation (C Apps.)

Essential Information (C apps.)

The information in this section is essential to ensure proper use of the DLL. Please refer to this section before seeking technical support.

DLL Initialization (C Apps.)

Before any DLL calls are made the DLL must be properly initialized with the passcode.   Remember, if the DLL is unloaded and then reloaded later, the DLL must be initialized again.

InitDll (Procedure)

Purpose

To intialize the DLL

Parameters

hwnd:	application window handle
passcode:	the passcode string is "yk127e".

Syntax

```
InitDll (HWND hwnd, const char *passcode);
```

+ Your application must make this call for the DLL to work.   If the DLL is not unloaded, this call is required only once.

.h Files and .lib Files (C Apps.)

Include the .h files from our example subdirectory for your compiler. To save time, use the .lib files from these examples in your project.

+ ImageLib includes examples for Borland C++ 4.0x and for Microsoft VC++ 1.5x. To find these examples go to the directory where you installed ImageLib then select the appropriate subdirectory. Note! The install program has the option not to install these subdirectories. If you cannot find them, run the install program again. The Borland examples are OWL examples and are located in the Borland (bcc) subdirectory. The Microsoft Visual C++ demo is an MFC example is located in the (mvc) subdirectory.

Color Quantizer Explained (C Apps.)

The color quantizer is used to reduce an image to a lower bit depth while preserving as much quality as possible. The color quantizer will analyze the input image and produce an optimized color palette using the maximum number of colors allowed for the output bit depth. The color quantizer will then reduce the input image by mapping the input image pixels to the output image pixels through the optimized color palette. This can be done with dithering or without dithering. In most cases dithering produces better results. Input images that have a bit depth of 8 or higher can be reduced. Input images of 1 bit per pixel will not be reduced. The reduction options are as follows:

24 bit 16.7 million color to 8 bit 256 color
24 bit 16.7 million color to 4 bit 16 color
24 bit 16.7 million color to 4 bit 16 VGA colors (for VGA systems)
24 bit 16.7 million color to 1 bit 2 color
8 bit 256 color to 4 bit 16 color
8 bit 256 color to 4 bit 16 VGA colors
8 bit 256 color to 1 bit 2 color
4 bit 16 color to 1 bit 2 color

+ If the 16 color output is selected, an optimized color palette based on the input image is used. This means that with VGA modes the 16 color output may not look good because the optimized 16 color palette does not match the windows system palette. To overcome this problem the VGA 16 colors option should be used for VGA systems.

All read calls support the passing in of a handle to a windows logical color palette. If a windows logical color palette handle is passed, the DLL will use that color palette and color reduce the image to match the input color palette. When passing in a color palette, make sure the requested output resolution matches the number of colors in the color palette.

Parameter Key for DLL Calls (C Apps.)

In order to save space, a single parameter key is proveded. Please refer to this section when using the DLL calls.

bitspixel Short integer pointer that indicates the pixel depth or bits per pixel of the image

compression Short integer value to determine the compression method

1:	No compression
2:	CCITT compression (1 bit; not supported)
5:	LZW compression
32773:	PackBits compression

+ This compression parameter is for use with TIFF read and write functions.

compression Character pointer for the type of compression used for the image or other useful information. For JPEG and PNG images this variable will indicate a RGB or Grayscale colorspace type for the image.

Possible Values for BMP Images:
`"NONE", "RLE 8-bit" or "RLE 4-bit"`

Possible Values for GIF Images:
`"LZW" or "LZW, Interlaced"`

Possible Values for JPG Images
`"RGB" or "GRAYSCALE"`

Possible Values for PCX Images
`"RLE"`

Possible Values for PNG Images
`"Palette", "RGB", "GRAYSCALE", "RGB A",`
`"GRAY A", or "UNKNOWN"`

Possible Values for TIFF Images
`Tiff values have two parts seperated by a space. The first part is from column one and the second part is from column two.`

Column One	**Column Two**
CIE	CCITT
CMYK	GRP3FAX
GRAYSCALE	GRP4FAX
PALETTE	JPEG
RGB	LZW

```
TRANS MASK                NONE
UNKNOWN                   PACKBITS
YCbCr                     UNKNOWN

Examples: "CIE LZW" or "RGB LZW"
```

+ This compression parameter is for use with file and stream info functions.

dither Short Integer used to turn dithering on and off
- 0: No dithering
- 1: Dither

errormode Short integer used to turn error messages on and off
- 0: Do not show error messages in DLL
- 1: Display error messages in DLL

+ All of the DLL calls will return a 1 for success or a negative number to indicate a particular error. All of the error codes have text string equivalents located in a string table resource inside of the DLL. The first example imgview illustrates how to access the strings. If you want to control the error messages, use 0 so that the DLL will not automatically display error messages.

filename A contant character pointer to a string containing the name and path of output file

filetype A character pointer will contain the type of image contained in the file.

hddb Unsigned integer for Device Dependent Bitmap (DDB) handle

+ This is not a pointer

hdib Unsigned integer for Device Independent Bitmap (DIB) handle

+ This is not a pointer

height Short integer pointer for the height of the image in pixels

hideUI Short integer (this option not currently used)

hpal Unsigned integer for a logical palette handle

+ The DLL checks the input palette handle. If the value is NOT 0, the DLL will attempt to use that color palette for the output image if color reduction is performed. Otherwise, the DLL will create an optimum color palette. This is not a pointer.

hwind A handle to an application window

inbuffer A void pointer to a memory location that contains the input image to read or the memory location where the output image will be written. The memory location should be globally allocated.

interlaced Short integer to turn interlacing on and off

1:	write an interlaced type PNG image
0:	write a non-interlaced type PNG image

lzwpasswd Constant character pointer for the password to use LZW compression. LZW compression is patented by Unisys. In order to obtain the password you must provide SkyLine Tools with a copy of your license from Unisys. If you do not have a license from Unisys assign the pointer to NULL. When the password is assigned to NULL images compressed with LZW will be unuseable. For information on obtaining a license contact SkyLine Tools at (818) 766-3900 or Unisys at (215) 986-4411.

numcolors Short integer pointer for the number of palette entries used by the image. Will be 0 for RGB or true color images.

planes Short integer pointer for the number of bit planes in the image

pf Pointer to the callback function defined as:

```
short pf (short);
```

+ If no callback function is defined, pass NULL for pf. If a callback function is used, that function will be called periodically with an integer input value between 0 and 100. This value represents how much of the current operation has been completed (in percent). The application's callback function must return a short value indicating status. A return value equal to 1 continues the DLL function; and a return value equal to 0 cancels the DLL function. If the DLL function is canceled, it returns a valid partial bitmap and palette handle. The read function will not indicate an error has occured if processing is canceled. For write functions, the DLL will ignore any requests to cancel.

quality Short integer between 0 and 100 that controls the quality of the image to be stored.

+ 0 is poor and 100 is excellent. We normally use 75 to have reasonable quality with 1/10 savings in size.

resolution Short integer for choosing the resolution

1:	2 bit	(2 colors)
0:	4 bit VGA palette	(16 colors)
4:	4 bit	(16 colors)
8:	8 bit	(256 colors)
24:	24 bit	(16 Million colors)

scale	Short integer for choosing the scale (Jpeg Only) 1: 1/1 normal size 2: 1/2 size 4: 1/4 size 8: 1/8 size
size	This is a long integer containing the number of bytes in the buffer of the image to be read or indicates on return how much of the memory location was actually used to store the output image.
smooth	Short integer between 0 and 100 that controls the amount of smoothing performed on the image

+ 0 is no smoothing and 100 is full smoothing

stripsize	Short integer indicating the number of strips
width	Short integer pointer for the width of the image in pixels

Error Strings and Foreign Languages (C Apps.)

Each function returns an integer thet refers to a particular error. For a list of these errors use your resource editor on the ImageLib DLL. To change the error messages displayed from the ImageLib DLL, use your resource editor to modify the error strings. The error strings can be changed or converted into the language of your choosing. In addition, the code example below can be used to extract strings from the string table resource located within Imagelib.

```
HINSTANCE moduleinst;
short msgcode; // the error code to retrieve
char buffer[MSG_LENGTH];//MSG_LENGTH should be about 80
// Get the module handle for the DLL
if ((moduleinst = GetModuleHandle(SKYDLL)) != NULL)
   {
    // Call the LoadString api to copy error message
    // into buffer
    LoadString(moduleinst, msgcode, buffer, MSG_LENGTH);
    /* get message string */
    /* Create the message */
   }
```

Image Information (C Apps.)

The ImageLib DLL supports functions that provide image information without reading the entire image.

fileinfo (function)
streaminfo (function)

fileinfo (function) (C Apps.)

Purpose

This function takes the filename of an image and returns information about the image.   This function works with BMP, GIF, JPG, PCX, PNG and TIFF images.   It will identify the image type regardless of the   file extension.   All parameters are pointer to variables that will be filled by the function.

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short fileinfo(const char * filename, char * filetype, short
               * width, short * height, short * bitspixel,
               short * planes, short * numcolors, char *
               compression, short errormode);
```

streaminfo (function) (C Apps.)

Purpose

This function identifies the type of image in the buffer. The entire image should be in memory when trying to use this function. While the function may work with an incomplete image, it is not certified to do as such.

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short streaminfo(void * inbuffer, long size, char *
                 filetype, short * width, short * height,
                 short * bitspixel, short * planes, short *
                 numcolors, char * compression, short
                 errorcode);
```

Image Manipulation (C Apps.)

This section contains tools that allow the user to rotate and flip images.   Using combinations of two our more of these functions will give the developer even more way to manipulate images.

FlipDdb (function) (C Apps.)

Purpose

Flips a device dependent bitmap image along the y-axis (mirror)

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short FlipDdb(short resolution, unsigned int * hddb,
              unsigned int *hpal, short errormode);
```

+ The resolution parameter must indicate the current pixel depth of your windows resolution.

FlipDib (function) (C Apps.)

Purpose

Flips a device independent bitmap image along the y-axis (mirror)

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short FlipDib(unsigned int * hdib,   short errormode);
```

+ These functions are independent of the actual pixel depth.

RotateDdb90 (function) (C Apps.)

Purpose

Rotates a device dependent bitmap image clockwise 90 degrees

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short RotateDdb90(short resolution, unsigned int * hddb,
                  unsigned int *hpal, short errormode);
```

+ The resolution parameter must indicate the current pixel depth of your windows resolution.

RotateDdb180 (function) (C Apps.)

Purpose

Rotates a device dependent bitmap image clockwise 180 degrees

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short RotateDdb180(short resolution, unsigned int * hddb,
                   unsigned int *hpal,short errormode);
```

+ The resolution parameter must indicate the current pixel depth of your windows resolution.

RotateDib90 (function) (C Apps.)

Purpose

Rotates a device independent bitmap image clockwise 90 degrees

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short RotateDib90(unsigned int * hdib, short errormode);
```

+ These functions are independent of the actual pixel depth.

RotateDib180 (function) (C Apps.)

Purpose

Rotates a device independent bitmap image clockwise 180 degrees

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short RotateDib180(unsigned int * hdib, short errormode);
```

+ These functions are independent of the actual pixel depth.

TWAIN Support (C Apps.)

The ImageLib DLL supports Twain Compliant Devices.   The functions to control such devices are listed below:

aquiredibimage (function)
aquireimage (function)
selectsource (function)
twainavailable (function)

aquiredibimage (function) (C Apps.)

Purpose

This function is used to aquire an image from a twain compliant device. The image returned will have a device independent bitmap (DDB). This call will start the selected twain source (if no Source has been selected the default is used). The source manager for the twain device will allow the user to control that device. Once the user has finished with the twain device a handle to the BITMAP and PALETTE indicate where the image is stored.

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short aquiredibimage(HWND hwind, short resolution, short
                     dither, short hideUI, unsigned int *
                     hddb, unsigned int hpal, short (*pf)
                     (short), short errormode);
```

aquireimage (function) (C Apps.)

Purpose

This function is used to aquire an image from a twain compliant device. The image returned will have a device dependent bitmap (DDB). This call will start the selected twain source (if no Source has been selected the default is used). The source manager for the twain device will allow the user to control that device. Once the user has finished with the twain device a handle to the BITMAP and PALETTE indicate where the image is stored.

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short aquireimage(HWND hwind, short resolution, short
                  dither, short hideUI, unsigned int *
                  hddb, unsigned int * hpal, short (*pf)
                  (short), short errormode);
```

selectsource (function) (C Apps.)

Purpose

This call starts the Twain source selection window. The user can choose the source manager they wish to use. Selectsource is for systems that have more that one Twain Source loaded. Selecting "Cancel" in the source selection window is not an error.

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short selectsource(HWND hwind, short errormode);
```

twainavailable (function) (C Apps.)

Purpose

This call will check the windows environment to see if Twain is available.   The function looks for the "TWAIN.DLL" file and the source manager.   A one indicates Twain is available and a negative number indicates an errorcode.   If a negative number is returned, ensure the Twain software was correctly installed for use with your Twain complient device.

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short twainavailable(HWND hwind, short errormode);
```

BMP Image Format (C Apps.)

The ImageLib DLL supports BMP read and write functions.   These functions can be used with files or memory streams.

BMP Read Functions (C Apps.)

The ImageLib DLL read functions for BMP can be used with files or memory streams.

rdbmpfiledib (function) (C Apps.)

Purpose

To read a BMP Image from a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdbmpfiledib(const char *filename, short resolution,
                   short dither, unsigned int * hdib, unsigned
                   int hpal, short (*pf)(short), short
                   errormode);
```

+ The input BMP image must contain the BITMAPFILEHEADER part of a BMP at the front before the BITMAPINFOHEADER.   The read BMP functions do support RLE type BMP files.

rdbmpstreamdib (function) (C Apps.)

Purpose

To read a BMP Image from memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdbmpstreamdib(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hdib, unsigned int hpal, short(*pf)(short),
                short errormode);
```

+ The input BMP image must contain the BITMAPFILEHEADER part of a BMP at the front before the BITMAPINFOHEADER.   The read BMP functions do support RLE type BMP files.

readbmpfile (function) (C Apps.)

Purpose

To read a BMP Image from a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readbmpfile(const char *filename, short resolution,
                  short dither, unsigned int * hddb, unsigned
                  int * hpal, short (*pf)(short), short
                  errormode);
```

+ The input BMP image must contain the BITMAPFILEHEADER part of a BMP at the front before the BITMAPINFOHEADER. The read BMP functions do support RLE type BMP files.

readbmpstream (function) (C Apps.)

Purpose

To read a BMP Image from memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readbmpstream(void * inbuffer, long size, short
                    resolution, short dither, unsigned int *
                    hddb, unsigned int * hpal, short(*pf)
                    (short), short errormode);
```

+ The input BMP image must contain the BITMAPFILEHEADER part of a BMP at the front before the BITMAPINFOHEADER. The read BMP functions do support RLE type BMP files.

BMP Write Functions (C Apps.)

The ImageLib DLL read functions for BMP can be used with files or memory streams.

wrbmpfiledib (function)
wrbmpstreamdib (function)
writebmpfile (function)
writebmpstream (function)

wrbmpfiledib (function) (C Apps.)

Purpose

To write a BMP Image to a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrbmpfiledib(const char * filename,   short
                   resolution, unsigned int hdib, short
                   (*pf)(short), short errormode);
```

wrbmpstreamdib (function) (C Apps.)

Purpose

To write a BMP Image to memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrbmpstreamdib(void * inbuffer, long * size,   short
                     resolution, unsigned int hdib, short
                     (*pf)(short), short errormode);
```

writebmpfile (function) (C Apps.)

Purpose

To write a BMP Image to a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writebmpfile(const char * filename,   short
                resolution, unsigned int hddb, unsigned int
                hpal, short (*pf)(short), short errormode);
```

writebmpstream (function) (C Apps.)

Purpose

To write a BMP Image to memory using a device dependent bitmap

Parameters

See: Parameter Key for DLL Calls

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writebmpstream(void * inbuffer, long * size,   short
                resolution, short password, unsigned int
                hddb, unsigned int hpal, short (*pf)
                (short), short errormode);
```

GIF Image Format (C Apps.)

The ImageLib DLL supports the GIF 87a standard.   The GIF read and write functions can be used with files or memory streams

GIF Read Functions
GIF Write Functions

GIF Read Functions (C Apps.)

rdgiffiledib (function) (C Apps.)

Purpose

To read a GIF Image from a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdgiffiledib(const char *filename, short resolution,
                   short dither, unsigned int * hdib, unsigned
                   int hpal, short (*pf)(short), short
                   errormode, const char * lzwpasswd);
```

rdgifstreamdib (function) (C Apps.)

Purpose

To read a GIF Image from memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdgifstreamdib(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hdib, unsigned int hpal, short(*pf)(short),
                short errormode, const char * lzwpasswd);
```

readgiffile (function) (C Apps.)

Purpose

To read a GIF Image from a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readgiffile(const char *filename, short resolution,
                  short dither, unsigned int * hddb, unsigned
                  int * hpal, short (*pf)(short), short
                  errormode, const char * lzwpasswd);
```

readgifstream (function) (C Apps.)

Purpose

To read a GIF Image from memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readgifstream(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hddb, unsigned int * hpal, short(*pf)
                (short), short errormode, const char *
                lzwpasswd);
```

GIF Write Functions (C Apps.)

wrgiffiledib (function) (C Apps.)

Purpose

To write a GIF Image to a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrgiffiledib(const char * filename,   short
                resolution, unsigned int hdib, short (*pf)
                (short), short errormode, const char *
                lzwpasswd);
```

wrgifstreamdib (function) (C Apps.)

Purpose

To write a GIF Image to memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrgifstreamdib(void * inbuffer, long * size,   short
                    resolution, unsigned int hdib, short (*pf)
                    (short), short errormode, const char *
                    lzwpasswd);
```

writegiffile (function) (C Apps.)

Purpose

To write a GIF Image to a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writegiffile(const char * filename,   short
                resolution, unsigned int hddb, unsigned int
                hpal, short (*pf)(short), short errormode,
                const char * lzwpasswd);
```

writegifstream (function) (C Apps.)

Purpose

To write a GIF Image to memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writegifstream(void * inbuffer, long * size, short
                resolution, unsigned int hddb, unsigned int
                hpal, short (*pf)(short), short errormode,
                const char * lzwpasswd);
```

JPG Image Format (C Apps.)

The ImageLib DLL supports JPEG read and write functions.   These functions can be used with files or memory streams.

JPG Read Functions
JPG Write Functions

JPG Read Functions (C Apps.)

rdjpgfiledib (function) (C Apps.)

Purpose

To read a JPeg Image from a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdjpgfiledib (const char *filename, short resolution,
                  short scale, short dither, unsigned int
                  *hdib, unsigned int hpal, short (*pf)
                  (short), short errormode);
```

rdjpgstreamdib (function) (C Apps.)

Purpose

To read a JPeg Image from memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdjpgstreamdib (void * inbuffer, long size, short
                    resolution, short scale, short dither,
                    unsigned int * hdib, unsigned int hpal,
                    short (*pf) (short), short errormode);
```

readjpgfile (function) (C Apps.)

Purpose

To read a JPeg Image from a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readjpgfile (const char *filename, short resolution,
                  short scale, short dither, unsigned int *
                  hddb, unsigned int * hpal, short (*pf)
                  (short), short errormode);
```

readjpgstream (function) (C Apps.)

Purpose

To read a JPeg Image from memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readjpgstream (void * inbuffer, long size, short
                  resolution, short scale, short dither,
                  unsigned int * hddb, unsigned int * hpal,
                  short (*pf) (short), short errormode);
```

JPG Write Functions (C Apps.)

writejpegfile (function) (C Apps.)

Purpose

To write a JPeg Image to a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writejpegfile (const char * filename, short quality,
                  short smooth, short resolution, unsigned
                  int hddb, unsigned int hpal, short (*pf)
                  (short), short errormode);
```

writejpegstream (function) (C Apps.)

Purpose

To write a JPeg Image to memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writejpegstream (void * inbuffer, long * size,   short
                     quality, short smooth, short
                     resolution, unsigned int hddb, unsigned
                     int hpal, short (*pf)(short), short
                     errormode);
```

wrjpegfiledib (function) (C Apps.)

Purpose

To write a JPeg Image to a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrjpegfiledib(const char * filename,   short quality,
                  short smooth, short resolution, unsigned
                  int hdib, short (*pf)(short), short
                  errormode);
```

wrjpegstreamdib (function) (C Apps.)

Purpose

To write a JPeg Image to memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrjpegstreamdib(void * inbuffer, long * size,   short
                        quality, short smooth, short resolution,
                        unsigned int hdib, short (*pf)(short),
                        short errormode);
```

PCX Image Format (C Apps.)

The ImageLib DLL supports PCX read and write functions.   These functions can be used with files or memory streams.

PCX Read Functions
PCX Write Functions

PCX Read Functions (C Apps.)

rdpcxfiledib (function) (C Apps.)

Purpose

To read a PCX Image from a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdpcxfiledib(const char *filename, short resolution,
               short dither, unsigned int * hdib, unsigned
               int hpal, short (*pf)(short), short
               errormode);
```

rdpcxstreamdib (function) (C Apps.)

Purpose

To read a PCX Image from memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdpcxstreamdib(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hdib, unsigned int hpal, short(*pf)(short),
                short errormode);
```

readpcxfile (function) (C Apps.)

Purpose

To read a PCX Image from a file using a device dependent bitmap

Parameters

See: Parameter Key for DLL Calls

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readpcxfile(const char *filename, short resolution,
                short dither, unsigned int * hddb, unsigned
                int * hpal, short (*pf)(short),short
                errormode);
```

readpcxstream (function) (C Apps.)

Purpose

To read a PCX Image from memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readpcxstream(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hddb, unsigned int * hpal, short(*pf)
                (short), short errormode);
```

PCX Write Functions (C Apps.)

writepcxfile (function) (C Apps.)

Purpose

To write a PCX Image to a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writepcxfile(const char * filename,   short
                resolution, unsigned int hddb, unsigned int
                hpal, short (*pf)(short),short errormode);
```

writepcxstream (function) (C Apps.)

Purpose

To write a PCX Image to memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writepcxstream(void * inbuffer, long * size,   short
                resolution, unsigned int hddb, unsigned int
                hpal,short (*pf)(short), short errormode);
```

wrpcxfiledib (function) (C Apps.)

Purpose

To write a PCX Image to a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrpcxfiledib(const char * filename,   short
                 resolution, unsigned int hdib, short (*pf)
                 (short), short errormode);
```

wrpcxstreamdib (function) (C Apps.)

Purpose

To write a PCX Image to memory using a device in dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrpcxstreamdib(void * inbuffer, long * size,   short
                resolution, unsigned int hdib, short (*pf)
                (short), short errormode);
```

PNG Image Format (C Apps.)

The ImageLib DLL supports PNG read and write functions. These functions can be used with files or memory streams.

PNG Read Functions
PNG Write Functions

PNG Read Functions (C Apps.)

rdpngfiledib (function) (C Apps.)

Purpose

To read a PNG Image from a file using a device independent bitmap

Parameters

See: Parameter Key for DLL Calls

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdpngfiledib(const char *filename, short resolution,
                   short dither, unsigned int * hdib, unsigned
                   int hpal, short (*pf)(short), short
                   errormode);
```

rdpngstreamdib (function) (C Apps.)

Purpose

To read a PNG Image from memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdpngstreamdib(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hdib, unsigned int hpal, short(*pf)(short),
                short errormode);
```

readpngfile (function) (C Apps.)

Purpose

To read a PNG Image from a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readpngfile(const char *filename, short resolution,
                short dither, unsigned int * hddb, unsigned
                int * hpal, short (*pf)(short), short
                errormode);
```

readpngstream (function) (C Apps.)

Purpose

To read a PNG Image from memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readpngstream(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hddb, unsigned int * hpal, short(*pf)
                (short), short errormode);
```

PNG Write Functions (C Apps.)

writepngfile (function) (C Apps.)

Purpose

To write a PNG Image to a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writepngfile(const char * filename,   short
                resolution, short interlaced, unsigned int
                hddb, unsigned int hpal, short (*pf)
                (short), short errormode);
```

writepngstream (function) (C Apps.)

Purpose

To write a PNG Image to memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writepngstream(void * inbuffer, long * size,   short
                    resolution, short interlaced, unsigned int
                    hddb, unsigned int hpal, short (*pf)
                    (short), short errormode);
```

wrpngfiledib (function) (C Apps.)

Purpose

To write a PNG Image to a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrpngfiledib(const char * filename,   short
                resolution, short interlaced, unsigned int
                hdib, short (*pf)(short), short errormode);
```

wrpngstreamdib (function) (C Apps.)

Purpose

To write a PNG Image to memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrpngstreamdib(void * inbuffer, long * size,   short
                resolution, short interlaced, unsigned int
                hdib, short (*pf)(short), short errormode);
```

TIFF Image Format (C Apps.)

The ImageLib DLL meets the TIFF baseline specifications and will support TIFF images from 1 to 24 bits.   ImageLib will read and write TIFF images of the following types:

No Compression
Packbits
LZW
CCITT Group 3 (1 bit only)

TIFF Read Funcations
TIFF Write Functions

TIFF Read Funcations (C Apps.)

rdtiffiledib (function) (C Apps.)

Purpose

To read a TIFF Image from a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdtiffiledib(const char *filename, short resolution,
                   short dither, unsigned int * hdib, unsigned
                   int hpal, short (*pf)(short), short
                   errormode, const char * lzwpasswd);
```

rdtifstreamdib (function) (C Apps.)

Purpose

To read a TIFF Image from memory using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short rdtifstreamdib(void * inbuffer, long size, short
               resolution, short dither, unsigned int *
               hdib, unsigned int hpal, short(*pf)(short),
               short errormode, const char * lzwpasswd);
```

readtiffile (function) (C Apps.)

Purpose

To read a TIFF Image from a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readtiffile(const char *filename, short resolution,
               short dither, unsigned int * hddb, unsigned
               int * hpal, short (*pf)(short), short
               errormode, const char * lzwpasswd);
```

readtifstream (function) (C Apps.)

Purpose

To read a TIFF Image from memory using a device dependent bitmap

Parameters

See: Parameter Key for DLL Calls

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short readtifstream(void * inbuffer, long size, short
                resolution, short dither, unsigned int *
                hddb, unsigned int * hpal, short(*pf)(short),
                short errormode, const char * lzwpasswd);
```

TIFF Write Functions (C Apps.)

writetiffile (function) (C Apps.)

Purpose

To write a TIFF Image to a file using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writetiffile(const char * filename, short resolution,
                   short compression, short stripsize,
                   unsigned int hddb, unsigned int hpal, short
                   (*pf)(short), short errormode, const char *
                   lzwpasswd);
```

writetifstream (function) (C Apps.)

Purpose

To write a TIFF Image to memory using a device dependent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short writetifstream(void * inbuffer, long * size, short
                resolution, short compression, short
                stripsize, unsigned int hddb, unsigned int
                hpal, short (*pf)(short), short errormode,
                const char * lzwpasswd);
```

wrtiffiledib (function) (C Apps.)

Purpose

To write a TIFF Image to a file using a device independent bitmap

Parameters

See: **Parameter Key for DLL Calls**

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrtiffiledib(const char * filename,   short
                resolution, short compression, short
                stripsize, unsigned int hdib, short (*pf)
                (short), short errormode, const char *
                lzwpasswd);
```

wrtifstreamdib (function) (C Apps.)

Purpose

To write a TIFF Image to memory using a device independent bitmap

Parameters

See: [Parameter Key for DLL Calls]()

Return

Integer	Returns a one for successful completion, negative integers indicate errors

Syntax

```
short wrtifstreamdib(void * inbuffer, long * size,   short
                resolution, short compression, short
                stripsize, unsigned int hdib, short (*pf)
                (short), short errormode, const char *
                lzwpasswd);
```

Visual Basic Programming and the ImageLib DLL

ImageLib is an inexpensive way to add BMP, GIF, JPEG, PCX, PNG and TIFF graphic formats to your Visual Basic (VB) applications. The ImageLib DLL supports the reading and writing of images from memory or file. The DLL supports the use of an optional callback function. The callback can provide a progress display of read and write functions. In addition, read functions can be canceled in progress.

The DLL also provides functions to retrieve information about an image in memory or a file without reading the whole image. The Image Information functions return the type of image, compression, width, height, bits per pixel, number of planes, and number of colors. The memory functions of the ImageLib DLL are specifically designed to support database BLOB operations. All calls return error codes and the DLL will optional display error messages. The error codes refer to error text strings located in a string table resource inside the DLL.

The ImageLib DLL supports Device Dependent Bitmaps(DDB) or Device Independent Bitmaps(DIB) in the reading and writing of images. The DLL contains a sophisticated color quantization engine that can be used when reading or writing images. When reading an image, settings can be used to ensure the resolution you specify is used and is independent of the input image. If the developer wants all images to be passed back as 256 color 8 bit dithered images then all bitmaps passed back will be 8 bit whether they where originally 24 bit or 4 bit. The color quantizer is designed to produce the best image possible at the desired resolution. When writing an image, the developer may specify the resolution of the image to be written (resolution must be valid for image type).

The ImageLib DLL is Twain compliant and can be used with Twain compliant devices such as scanners. The DLL includes a SelectSource call to select a Twain Source and an AquireImage call to invoke the vender's Twain Source Manager. Our Twain will work with 16 bit and 32 bit Twain Sources.

+ ImageLib includes examples for Microsoft VB. To find these examples go to the directory where you installed ImageLib then select the appropriate subdirectory. Note! The install program has the option not to install these subdirectories. If you cannot find them, run the install program again. The Microsoft Visual Basic demo is located in the (VB) subdirectory.

Essential Information

Essential Information (Visual Basic)

The information in this section is essential to ensure proper use of the DLL. Please refer to this section before seeking technical support.

Visual Basic Example Files

The Visual Basic example files provide the function declarations, syntax, and other relevant information required to use the ImageLib DLL with your VB applications. The example program will show you how to read and write each of the file formats supported by the ImageLib DLL. Provided with the example program is a ".BAS" module file, which will show the declarations for the ImageLib functions in Visual Basic. Feel free to use this to link the DLL with your projects. Aside from the Visual Basic components, the only other file needed is the ImageLib DLL. Below is a list of the example project files:

IMGLIB.MAK	Visual Basic project file
IMGLIB.BAS	Module file for the ImageLib DLL functions
MAINFORM.FRM	The main project form

Function Calls (Visual Basic)

For a detailed description of each function in the ImageLib DLL, please refer to the section titled "C Programming with the ImageLib DLL." This section describes the function parameters and return values.

TThumbPreview (Component)

ImageLib supports the use of thumbnail images with the TThumbPreview component. Thumbnails are miniature copies of larger image files. The TThumbPreview component uses a thumbnail manager to display multiple thumbnails, create new thumbnails, and remove old thumbnails. Double clicking one of the images on the thumbnail manager will display that image.

AutoLoad (Property)
DataFileDir (Property)
DataFileName (Property)
Filename (Hidden Property)
PreviewDir (Property)

AutoLoad (Property)

Value

True or False

Purpose

To automatically load the thumbnail previews when the thumbnail manager is opened, set AutoLoad to true.

Example

```
ThumbPreview1.AutoLoad := True;
```

DataFileDir (Property)

Value

The name of the directory that contains the thumbnail datafile

Purpose

To provide the name of the directory that contains the thumbnail datafile. The thumbnail datafile stores the thumbnail names and the corresponding file names of each image.

Example

```
ThumbPreview1.DataFileDir := ‘C\Thumbs’;
```

DataFileName (Property)

Value

The name of the thumbnail datafile

Purpose

To provide the name of the thumbnail datafile. The thumbnail datafile stores the thumbnail names and the corresponding file names of each image.

Example

```
ThumbPreview1.DataFileName := ‘thumbs.dat’;
```

Filename for use with TThumbPreview (Hidden Property)

Value

The filename of the image to be opened

Purpose

To pass the filename for the image that corresponds to the thumbnail selected. That file will be opened and displayed in your image component.

Example

```
procedure TForm1.Button1Click(Sender: TObject);
begin
if ThumbPreview1.Execute then
    PMultiImage1.Imagename := ThumbPreview1.Filename;
end;
```

PreviewDir (Property)

Value

The name of the directory that contains the thumbnail files (.THB)

Purpose

To provide the name of the directory that contains the thumbnail files.

Example

```
ThumbPreview1.PreviewDir := 'C\Thumbs';
```

Distributing the ImageLib DLLs

Applications created using ImageLib Components or the ImageLib DLLs will require the ImageLib DLLs at runtime.   To accommodate this requirement the ImageLib DLLs must be distributed with your applications.   Any use or distribution of the ImageLib DLLs must be consistent with the Software License Agreement at the beginning of this manual.

ImageLib uses two DLLs to handle its image and graphics features.   The SKY16V3C.DLL is for use with your 16-Bit applications.   This DLL is included with ImageLib 3.1 and ImageLib Combo.   The SKY32V3C.DLL is for use with your 32-Bit applications.   This DLL is included with ImageLib 95 and ImageLib Combo. When the ImageLib installation program was run, these DLLs were installed in both the /windows/system directory and the directory chosen for component installation.