# Imagelib 2.2

Borland Pascal and Delphi Users' Guide



Kevin Adams (CIS) 74742,1444

Jan Dekkers (CIS) 72130,353

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**

## What is ImageLib DLL/VCL?

The ImageLib VCL'S\DLL is an inexpensive way to add Jpeg, Gif, SCM and Pcx to your applications. Yes, there are image libraries supporting many more formats than ImageLib, but those libraries are more expensive and add more overhead to your applications. In addition it adds DBMultiImage and DBMultiMedia to store and display JPEG, BMP, GIF, SCM, PCX , AVI, MOV, MID, WAV and RMI multimedia files in/from a TBlobField. For international developers: Strings are displayed in the DLL as a resource file thereby enabling the translation into foreign languages.

ImageLib is a enhanced Timage and TDBImage VCL/DLL with the following added features:

* Enables the reading and writing of JPEG images to/from a file or a Tblobfield;
* Jpeg 4, 8 and 24 bit dithering;
* Jpeg 0 to 100% save quality;
* Jpeg 0 to 100% smoothing;
* **Enables the reading and writing of Scrolling messages images to/from a file or a Tblobfield (New format in 2.2);**
* Enables the reading of GIF images from a file or a TBlobfield;
* Enables the reading of PCX images from a file or a TBlobfield;
* Enables the reading and writing of BMP images to/from a file or a TBlobfield;
* Enables the reading and writing of AVI images to/from TBlobfield;
* Enables the reading and writing of MOV images to/from TBlobfield;
* Enables the reading and writing of WAV images to/from TBlobfield;
* Enables the reading and writing of RMI images to/from TBlobfield;
* Enables the reading and writing of MID images to/from TBlobfield;
* Enables the reading and writing of ICO images to/from a file(Delphi inherited);
* Enables the reading and writing of WMF images to/from a file (Delphi inherited);
* **TMultiImage CUT/COPY and Paste to/from the clipboard (New in 2.2);**
* **All Multi VCL's have full Print Support with 1 line of code (New in 2.2);**
* **Internal scrolling message editor (New in 2.2);**
* DLL Callback function, to show a progress bar and to process Messages;
* No code necessary (VCL) to display all image formats from a TBlobfield;
* Loads/Saves all Tblobfield images to/from file;
* Converts all Tblobfield images to Jpeg/Bmp file;
* Pastes images from Clipboard and stores as a Jpeg/Bmp file/Blob;
* Retrieves File/Blob info without actually opening the file; and
* **Foreign error strings. DLL strings are stored in the DLL resource**
* **Full VCL source code provided without extra charge**

## Installation Instructions

BACKUP YOUR \DELPHI\BIN\COMPLIB.DCL  Better safe than ;-(

Copy the IMGLIB22.dll to a directory on your path or to the windows\system directory. IMGLIB22.DLL IS A DISTRIBUTABLE FILE and need to be included with your application.

If you have installed an earlier version of TMultiImage, you must remove the old TMultiImage component. Execute Delphi. In Delphi select Options\Install components (select reg_image or reg_im20) and remove.  Press OK. Delete reg_image.pas and/or reg_im20.pas from your system.

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

**WHEN UNREGISTERED**
MULTIREG.PAS, MULITIREG.DCR, TMULTI.PAS, TDBMULTI.PAS, DLL22LIN.DCU SETSRMSG.DFM and SETSRMSG.DCU

**WHEN REGISTERED**
MULTIREG.PAS, MULITIREG.DCR, TMULTI.PAS, TDBMULTI.PAS, DLL22LIN.PAS SETSRMSG.DFM and SETSRMSG.PAS.

Execute Delphi. In Delphi select Options\Install components\ Add and browse your 3rd party added VCL's directory. Select **MULTIREG** and press the OK button.

After the library is rebuilt, you will notice 4 new icons on your Delphi toolbar under images called:

**MultiImage,**
**DBMultiImage,**
**DBMultiMedia,**
**DBMediaPlayer.**

**Troubleshooting:**
The Delphi Library searchpath is very short (We assume 256 characters) The more VCL components you add the larger your searchpath. Should you get a message **MULTIREG.PAS or MULTIREG.DCU** not found than your path is being truncated. The solution is to copy several 3rd party VCL's into one directory and delete the freed directories from your searchpath.

If Complib can not find IMGLIB22.DLL you will notice that all Icons are gone from your delphi toolbar and you get a message COMPLIB.DCL not found. No Panic, Just copy IMGLIB22.DLL to a directory on your path or to the windows\system directory.

**Installation Instructions for the Examples**

In delphi select Open\Project and open one of the projects in the newly created directory. Select rebuild. Run the program.

**IMPORTANT: IF YOU INSTALLED THE OLD MULTIIMAGE or DBMULTIIMAGE**

What to do with your existing programs using the old MultiImage VCL:

Incase of OLD MULTIIMAGE:

Change the uses clause of your programs from REG_IMAG or REG_IM20 to

**TMULTI, which is the replacement for** REG_IMAG or REG_IM20

Incase of OLD TDBMULTIIMAGE:

Change the uses clause of your programs from REG_IM20 to TDBMULTI.

**TDBMULTI, which is the replacement for** REG_IMAG or REG_IM20

**(Only for update from version 1.0 to version 2.0)**
When you startup your existing programs using the MultiImage VCL you might notice a complain (Property JPegSaveSmooh doesn't exist or Property JPegSaveFileName doesn't exists)

**Property JPegSaveSmooh is renamed to JPegSaveSmooTh** (watch the T)
To fix this, Load the FORM (the *.DFM) file complaining about this and replace JPegSaveSmooh with JPegSaveSmooTh (add the T)

**Property JPegSaveFileName is renamed to DefSaveFileName.**
To fix this, Load the FORM (the *.DFM) file complaining about this and replace **JPegSaveFileName** with **DefSaveFileName**

**New added Visual Components**

The new VCL objects added to your toolbar are called

**MultiImage,**
**DBMultiImage,**
**DBMultiMedia**
**DBMediaPlayer.**

**TMULTIIMAGE: JPEG, BMP, GIF, WMF, ICO and PCX.**

**Sample projects,**

| | |
|---|---|
| im_cvrt.dpr | Converting images example |
| scrollim.dpr | Scrolling messages example |
| simple.dpr | 2 lines of code example |
| viewph.dpr | Very extensive example |

MutlImage is derived from TCustomControl while Timage is derived TGraphicsControl.

However, it has the same properties as Delphi's TImage with the following additions:

## Reading and displaying images for all Image formats

### property ImageName

**Value**
Filename of the image which needs to be displayed.

**Purpose**
All images are loaded with one single line of code.

**Example**
MultiImage1.Imagename:=C:\ CLOWN.JPG';

## JPEG File read and write

### property JPegSaveQuality

**Value**
0...100

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

**Example**
MultiImage1.JPegSaveQuality:=25;

### property JPegSaveSmooth

**Value** 0...100

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

**Example**
MultiImage1.JPegSaveSmooth:=5;

### procedure SaveAsJpg(FN : TFilename);

**Value**
Filename of the file saved to

**Purpose**
Save the displayed image to a jpeg file.

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

**Example**

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
  if SaveDialog1.execute then begin
        MultiImage1.JPegSaveSmooth:=5;
        MultiImage1.JPegSaveQuality:=25;
        MultiImage1.SaveAsJpg(SaveDialog1.FileName);
 end;
end;
```

**property DefSaveFileName**
(Changed from JPGSaveFileName in version 2.0)

**Value**
Filename of the BMP or JPG which need to be saved.

**Purpose**
It can come in handy to store a filename long before the file is actually
saved. You can use this as a filename scratchpad.

**Example**

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
if SaveDialog1.execute then begin
  MultiImage1.JPegSaveQuality:=25;
  MultiImage1.JPegSaveSmooth:=5;
  MultiImage1.DefSaveFileName:=SaveDialog1.FileName;
  MultiImage1.SaveAsJpg('');
end;
end;
```

**property JPegDither**

**Value**
0 : No dithering 24 bit
1 : One Pass No dither
2 : One Pass dither
3 : Two Pass No dither
4 : Two Pass dither

**Purpose**
To set dithering methods for various VGA displays.
16 color display best JpegDither is 2
256 color display best JpegDither is 4
True color display best JpegDither is 0

**Example**
```
procedure TForm1.OpenFileClick(Sender: TObject);
begin
 if OpenDialog1.execute then begin
          MultiImage1.JPegDither:=4;
          MultiImage1.JPegResolution:=8;
          MultiImage1.imagename:=OpenDialog1.filename;
  end;
end;
```

**property JPegResolution**

**Value**

| | |
|---|---|
| 4 | (16 colors) |
| 8 | (256 colors) |
| 24 | (16 Million colors) |

**Purpose**
To set resolution for various VGA displays.

**Example**
```
procedure TForm1.OpenFileClick(Sender: TObject);
begin
 if OpenDialog1.execute then begin
          MultiImage1.JPegDither:=4;
          MultiImage1.JPegResolution:=8;
          MultiImage1.imagename:=OpenDialog1.filename;
  end;
end;
```

## BMP File read and write

**to read/display a BMP image you can use either Imagelib or delphi**

**Example 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.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 MultiImage1.Stretch := True;
 MultiImage2.Stretch := True;
 MultiImage 1.Picture.LoadFromFile('BITMAP1.BMP');
 MultiImage 2.Picture.LoadFromFile('BITMAP2.BMP');
end;
```

**Example 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.

```
procedure TForm1.FormCreate(Sender: TObject);
begin
 MultiImage1.Stretch := True;
 MultiImage2.Stretch := True;
 MultiImage 1.ImageName:='BITMAP1.BMP';
 MultiImage 2.ImageName:='BITMAP2.BMP';
end;
```

**<u>To Save a BMP image you can use either Imagelib or delphi</u>**

**Example using the delphi way.**

This example uses two picture components.

```
begin
        MultiImage1.Picture.SaveToFile('BITMAP1.BMP');
        MultiImage2.Picture.SaveToFile('BITMAP2.BMP');
end;
```

**Saving BMP's the Imagelib way.**

**procedure SaveAsBMP(FN : TFilename);**

**Value**
Filename of the file saved to

**Purpose**
Save the displayed image to a bmp file.

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

**Example**

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
{Open save dialog}
if SaveDialog1.execute then begin
  MultiImage1.DefSaveFileName:=SaveDialog1.FileName;
  MultiImage1.SaveAsBMP('');
end;
end;
```

**Or**

```
procedure TForm1.SaveButtonClick(Sender: TObject);
begin
{Open save dialog}
if SaveDialog1.execute then begin
  MultiImage1.SaveAsBMP(SaveDialog1.FileName);
end;
end;
```

## Scrolling Messages  File read and write

**Overview**

Scrolling messages are TMultiImages created by the VCL on the fly. An average filesize of an Scrolling message (SCM) is only 200 bytes. Stored in the SCM file are:

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

The VCL does NOT have a moving engine by its self. You “the programmer” must trigger the movements. The reason is that an application can have only one Application.OnIdle event.This event needs then be subdivided to other events which may need one. Note that other VCL’s could also use a Trigger. Make sure that their OnIdle proc. don’t destroy MultiImage’s trigger.

In your application you need to add a procedure to the private clauses called for instance Trigger:

```
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
        MultiImage3.Trigger;
        MultiImage2.Trigger;
        MultiImage1.Trigger;
end;
```

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

**Procedure Trigger;**

**Value**
none

**Purpose**
Trigger the scrolling message movements.

**Example**

```
Procedure TForm1.Trigger(Sender : TObject; Var Done : Boolean);
begin
        MultiImage1.Trigger;
```

```
end;
```

**procedure CreateMessage(MessagePath : String;  AutoLoad : Boolean);**

**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 their own scrolling message and save this message to a file witha SCM extension as default.

**Example**

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
        MultiImage1.CreateMessage(ExtractFilePath(Application.Exename), True);
end;
```

**procedure SaveCurrentMessage(MessageName : TFileName);**

**Value**

| | |
|---|---|
| MessageName | The filename the message is being saved to. |

**Purpose**

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

| | | |
|---|---|---|
| MultiImage1.MsgText | : String; | The message text. |
| MultiImage1.MsgFont | : Tfont; | The message font |
| MultiImage1.MsgBkGrnd | : Tcolor; | Background color |
| MultiImage1.MsgSpeed | : Integer; | Scrolling Speed |

**Example**

```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
        MultiImage1.MsgText:='ImageLib 2.2  A great tool to create a superb application';
        MultiImage1.MsgFont.Name:='Arial';
        MultiImage1.MsgFont.Size:=-40;
        MultiImage1.MsgFont.Style:=[fsitalic, fsbold];
        MultiImage1.MsgFont.Color:=clWhite;
        MultiImage1.MsgBkGrnd:=clNavy;
        MultiImage1.MsgSpeed:=1;
        if SaveDialog1.Execute then
          MultiImage1.SaveCurrentMessage(SaveDialog1.FileName);
end;
```

**Remark**
MsgFont.Name, MsgFont.Size, MsgFont.Style and MsgFont.Color could also be defined using a fontdialog box e.g. MultiImage1.MsgFont:= FontDialog1.Font;

**procedure NewMessage;**
**Value**
None

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

**Example**
```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
        MultiImage1.MsgText:='ImageLib 2.2  A great tool to create a superb application';
        MultiImage1.MsgFont.Name:='Arial';
        MultiImage1.MsgFont.Size:=-40;
        MultiImage1.MsgFont.Style:=[fsitalic, fsbold];
        MultiImage1.MsgFont.Color:=clWhite;
        MultiImage1.MsgBkGrnd:=clNavy;
        MultiImage1.MsgSpeed:=1;
        MultiImage1.NewMessage;
end;
```

**Procedure FreeMsg;**
**Value**
None

**Purpose**
Dispose the current message and assign then Picture to Nil

**Example**
```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
        MultiImage1.FreeMsg;
end;
```

## CLIPBOARD

**procedure CopyToClipboard;**
**Value** None

**Purpose**
Copy the current displayed image to the clipboard

**Example**
```
procedure TForm1.Copy1Click(Sender: TObject);
begin
        MultiImage1.CopyToClipboard;
```

end;

**procedure CutToClipboard;**
**Value** None

**Purpose**
Copy the current displayed image to the clipboard and erases it.

**Example**
```
procedure TForm1.Cut1Click(Sender: TObject);
begin
        MultiImage1.CutToClipboard
end;
```

**procedure PasteFromClipboard;**
**Value** None

**Purpose**
Paste an image from the clipboard into the MultiImage.

**Example**
```
procedure TForm1. Paste1Click(Sender: TObject);
begin
        MultiImage1.PasteFromClipboard;
end;
```

## Printing MultiImage Images

TmultiImage has full printing support to print JPEG, GIF, BMP, PCX, WMF and ICO. It does it with one procedure call.

**procedure PrintMultiImage(X, Y, pWidth, pHeight: Integer);**

**Value**

| | |
|---|---|
| X | The **Left** position of the image on the paper |
| Y | The **Top** position of the image on the paper |
| pWidth | The **Right** position of the image on the paper |
| pHeight | The **Bottom** position of the image on the paper |

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

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

**Example**
```
procedure TForm1.Print1Click(Sender: TObject);
begin
 if PrintDialog1.execute then
        MultiImage1.PrintMultiImage(0, 0, 0, 0);
end;
```

## Image Information

**Function GetInfoAndType(filename : TFilename) : Boolean;**
**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 successfull otherwise False. GetInfoAndType will store the following information:
For all filetypes:

| | | | |
|---|---|---|---|
| Bfiletype | : | String; | **Return:** JPEG, BMP, GIF, PCX, ICO, WMF, SCM |
| 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 JPEG, BMP, GIF, PCX only (ICO, WMF, SCM will return 0)

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

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

**Example**
```
procedure TForm1.DisPlayInfo(filename : TFilename);
begin
  if GetInfoAndType(filename) then begin
```

```
    Edit1.Text:=IntToStr(MultiImage1.Bwidth);
    Edit2.Text:=IntToStr(MultiImage1.BHeight);
    Edit3.Text:=IntToStr(MultiImage1.Bbitspixel);
    Edit4.Text:=IntToStr(MultiImage1.Bplanes);
    Edit5.Text:=IntToStr(MultiImage1.Bnumcolors);
    Edit6.Text:=MultiImage1.BFileType;
    Edit7.Text:=MultiImage1.Bcompression;
    Edit8.Text:=IntToStr(MultiImage1.BSize)+ bytes';
  end else begin
    Edit1.Text:='';
    Edit2.Text:='';
    Edit3.Text:='';
    Edit4.Text:='';
    Edit5.Text:='';
    Edit6.Text:='';
    Edit7.Text:='';
    Edit8.Text:='';
  end;
end;
```

## DLL Image CallBack Procedure

(Changed in version 2.2 from a procedure to a function.)

**Overview**

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.
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 messageloop. You can do this by adding APPLICATION.PROCESSMESSAGES in the callback procedure.

The Dll expects a callback function being registered of the following type:

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

**Value**
You need to pass a **1** if ok or a **0** if you want to cancel

**Returns**
a value between 1 and 100 which is the progress of the image being loaded.

**Remarks and Example**
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** clause:

```
Function ImageLibCallBack(i : integer) : integer; 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:

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

## TDBMULTIIMAGE: Sample project Blob.dpr

**Displays and stores JPEG, BMP, GIF, SCM and PCX from/to a TBLOBField.**

TDBMutltimage is the data-aware VCL version of TMultiImage. DBMutltimage is derived from TCustomControl. It has the same properties as Delphi's TDBImage with the following additions:

**property JPegSaveQuality**

**Value**
0...100

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

**Example**
```
DBMultiImage1.JPegSaveQuality:=25;
```

**property JPegSaveSmooth**

**Value**
0...100

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

**Example**
```
DBMultiImage1.JPegSaveSmooth:=5;
```

**procedure SaveToFileAsJpg(FN : TFilename);**
**Value**
The filename of the jpeg being saved to

**Purpose**

To saves the Image displayed as a Jpeg file.

**Remark**
Image must be displayed

**Example**
```
procedure TForm1.BitBtn8Click(Sender: TObject);

begin
        DBMultiImage1.JPegSaveQuality:=25;
        DBMultiImage1.JPegSaveSmooth:=5;
        If SaveDialog2.Execute then
                DBMultiImage1.SaveToFileAsJpeg(SaveDialog2.Filename);
end;
```

**property JPegDither**

**Value**
0 : No dithering 24 bit
1 : One Pass No dither
2 : One Pass dither
3 : Two Pass No dither
4 : Two Pass dither

**Purpose**
To set dithering methods for various VGA displays.
16 color display best JpegDither is 2
256 color display best JpegDither is 4
True color display best JpegDither is 0

**Example**
```
procedure TForm1.RefreshClick (Sender: TObject);
begin
        DBMultiImage1.JPegDither:=4;
        DBMultiImage1.JPegResolution:=8;
        DBMultiImage1.Refresh;
end;
```

**property JPegResolution**

**Value**
4 (16 colors)
8 (256 colors)
24 (16 Million colors)

**Purpose**
To set resolution for various VGA displays.

**Example**
```
procedure TForm1.RefreshClick (Sender: TObject);
```

```
begin
        DBMultiImage1.JPegDither:=4;
        DBMultiImage1.JPegResolution:=8;
        DBMultiImage1.Refresh;
end;
```

**procedure SaveToFileAsBMP(FN : TFilename);**

**Value**
The filename of the bmp being saved to

**Purpose**
To saves the Image displayed as a bmp file.

**Remark**
Image must be displayed

**Example**

```
procedure TForm1.BitBtn8Click(Sender: TObject);
begin
        If SaveDialog2.Execute then
                DBMultiImage1. SaveToFileAsBMP(SaveDialog2.Filename);
end;
```

**procedure SaveToFile(filename : TFilename);**

**Value**
The filename of the file being saved to

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

**Example**

```
procedure TForm1.BitBtn2Click(Sender: TObject);
var temp : string;
begin
 temp:=DBMultiImage1.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 = '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
   DBMultiImage1.SaveToFile(SaveDialog1.FileName);
end;
```

## Image Information

**Function GetInfoAndType(filename : TFilename) : STRING**
**Value**
Filename of the image

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

**Returns**
Extension format of the file stored in the blobfield. GetInfoAndType will store the following information:

For all filetypes:

| | | |
|---|---|---|
| Bfiletype | : String; | **Return:** JPEG, BMP, GIF, PCX, ICO, WMF, SCM |
| 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 JPEG, BMP, GIF, PCX only (ICO, WMF, SCM will return 0)

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

**Remark**
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.

**Example**

```
procedure TForm1.DataSource1DataChange(Sender: TObject; Field: TField);
begin
 If not DBMultiImage1.autodisplay then DBMultiImage1.GetInfoAndType;

 Edit1.text:='This blob image is a '+TDBMultiImage1.BFiletype;
 Edit2.text:=IntToStr(DBMultiImage1.Bwidth);
 Edit3.text:=IntToStr(DBMultiImage1.BHeight);
 Edit4.text:=IntToStr(DBMultiImage1.Bbitspixel);
 Edit5.text:=IntToStr(DBMultiImage1.Bplanes);
 Edit6.text:=IntToStr(DBMultiImage1.Bnumcolors);
 Edit7.text:=TDBMultiImage1.Bcompression;
```

```
 Edit8.text:=IntToStr(DBMultiImage1.BSize);
end;
```

**property UpDateBlobAsJpeg : boolean**

**Value**
True or  False

**Purpose**
To store the image displayed either as a JPEG or as a BMP If True then the Blob Image will be updated to a Jpeg Blob. If False then the Blob Image will be updated to a BMP Blob.

**Remark**
Image must be displayed

**Example**

```
procedure TForm1.UpdateAsJpeg(Sender: TObject);
begin
  DBMultiImage1.UpdateBlobAsJpeg:=True;
  DBMultiImage1.PastefromClipboard;
  Table1.Post;
end;

procedure TForm1.UpdateAsBMP(Sender: TObject);
begin
  DBMultiImage1.UpdateBlobAsJpeg:=False;
  DBMultiImage1.PastefromClipboard;
  Table1.Post;
end;
```

## Printing DBMultiImage Images

TDBmultiImage has full printing support to print JPEG, GIF, BMP, PCX,. It does it with one procedure call.

**procedure PrintMultiImage(X, Y, pWidth, pHeight: Integer);**

**Value**

| | |
|---|---|
| X | The **Left** position of the image on the paper |
| Y | The **Top** position of the image on the paper |
| pWidth | The **Right** position of the image on the paper |
| pHeight | The **Bottom** position of the image on the paper |

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

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

**Example**

```
procedure TForm1.Print1Click(Sender: TObject);
begin
 if PrintDialog1.execute then
        DBMultiImage1.PrintMultiImage(0, 0, 0, 0);
end;
```

## CLIPBOARD

**procedure CopyToClipboard;**
**Value** None

**Purpose**
Copy the current displayed image to the clipboard

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

**Example**
```
procedure TForm1.Copy1Click(Sender: TObject);
begin
        DBMultiImage1.CopyToClipboard;
end;
```

**procedure CutToClipboard;**
**Value** None

**Purpose**
Copy the current displayed image to the clipboard and erases it.

**Remark**
SHIFT DELETE and CRTL X does the same

**Example**
```
procedure TForm1.Cut1Click(Sender: TObject);
begin
        DBMultiImage1.CutToClipboard
end;
```

**procedure PasteFromClipboard;**
**Value** None

**Purpose**
Paste an image from the clipboard into the MultiImage.

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

**Example**
```
procedure TForm1. Paste1Click(Sender: TObject);
begin
        DBMultiImage1.PasteFromClipboard;
end;
```

## Scrolling TBobField Messages

**Overview**

Scrolling messages are TDBMultiImages created by the VCL on the fly. An average blob of an Scrolling message is only 200 bytes. Stored in the blob are:

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

The VCL does NOT have a moving engine by its self. You “the programmer” must trigger the movements. The reason is that an application can have only one Application.OnIdle event.This event needs then be subdived to other events which may need one. Note that other VCL’s could also use a Trigger. Make sure that their OnIdle proc. don’t destroy MultiImage’s trigger.

In your application you need to add a procedure to the private clauses called e.g. Trigger:

```
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:

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

**Procedure Trigger;**
**Value** none

**Purpose**
Trigger the scrolling message movements.

**Example**

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

**Function CreateMessage : boolean;**
**Value** none

**Purpose**
CreateMessage will open the Message editor. The user can create their own scrolling message and store these in the blobfield.

**Returns**
True if successfull otherwise false

**Example**
```
procedure TForm1.BitBtn13Click(Sender: TObject);
begin
        Table1.Append;
        If DBMultiImage1.CreateMessage then
                Table1.Post
      else
                Table1.Cancel;
end;
```

**Note: To save current blob message to a file use SaveToFile.**

**procedure NewMessage;**
**Value** None

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

**Example**
```
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
        DBMultiImage1.MsgText:='ImageLib 2.2  A great tool to create a superb application';
        DBMultiImage1.MsgFont.Name:='Arial';
        DBMultiImage1.MsgFont.Size:=-40;
        DBMultiImage1.MsgFont.Style:=[fsitalic, fsbold];
        DBMultiImage1.MsgFont.Color:=clWhite;
        DBMultiImage1.MsgBkGrnd:=clNavy;
        DBMultiImage1.MsgSpeed:=1;
        DBMultiImage1.NewMessage;
end;
```

**Procedure FreeMsg;**
**Value** None

**Purpose**
Dispose the current message and assign then Picture to Nil

**Example**
```
procedure TForm1.BitBtn5Click(Sender: TObject);
begin
        DBMultiImage1.FreeMsg;
end;
```

## TDBMULTIMEDIA and TDBMEDIAPLAYER: Sample project: MMBLOB.dpr

**Overview**
DBMultiMedia has all the same properties and functions as DBMultiImage. However, besides the storing and displaying of JPEG, BMP, GIF, SCM and PCX from a TBLOBField it does also store and play AVI, MOV, MID, WAV and RMI multimedia files.

DBMediaPlayer is a derived Delphi MediaPlayer and has exact all the same functions and properties. Using the DBMediaPlayer you don't need to assign anything to DBMediaPlayer directly, DBMultiMedia will take care of it.

**TDBMULTIMEDIA 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 soundsupport is installed;
RMI: If no midi playback drivers are installed;
MID: If no midi playback drivers are installed.

Thus you don't need to be afraid that your program chrashes when no soundcard is installed or Video for windows isn't present.

Again, all the properties from DBMultiImage are there and we added the following:

**function GetMultiMediaExtensions : String;**
**Value** none

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

**Remark**
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.

**Example**
```
procedure TBtnBottomDlg.BitBtn1Click(Sender: TObject);
begin
  OpenDialog1.filter:=DBMultiMedia1.GetMultiMediaExtensions;
        if OpenDialog1.Execute then begin
         Table1.Append;
           DBMultiMedia1.LoadfromFile(OpenDialog1.FileName);
           Table1.Post;
        end;
end;
```

**property PathForTempFile : string**
**Value**
PathName

**Purpose**
TDBMULTIMEDIA saves its AVI, MOV, WAV, MID and RMI blobs first to a temporary file before it is being played and then deletes the temporary file. The reason is that average multimedia blobs are too large in size to be played from memory. Your application might be distributed and executed from a CD. In order to write a temporary file you need the supply a directory and drive.

**Remark**
JPG, PCX, GIF and BMP Blobs are Not written to a temporary file but expanded directly into memory. If directory or drive doesn't exists it defaults to C:\

**Example**
```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin

 DBMultiMedia1.PathForTempFile:='C:\TEMP';

end;
```

**property AutoPlayMultiMedia : Boolean;**
**Value**
True or False

**Purpose**
If AutoPlayMultiMedia and AutoDisplay are True, the control automatically displays new data when the underlying BLOB field changes (such as when moving to a new record).If AutoPlayMultiMedia and AutoDisplay are False, the control clears whenever the underlying BLOB field changes. To display the data, the user can double-click on the control or select it and press Enter.

**Example**
```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin
 DBMultiMedia1.AutoPlayMultiMedia:=true;
end;
```

**property AutoRePlayMultiMedia : Boolean**
**Value**
True or False

**Purpose**
If AutoDisplay and AutoPlayMultiMedia are true then the multimedia is replayed automatically;

**Example**
```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin
 DBMultiMedia1.AutoRePlayMultiMedia:=true;
end;
```

**property AutoHideMediaPlayer : Boolean;**
**Value**
True or False

**Purpose**
If the blobfield doesn't contains multimedia it will hide the attached MediaPlayer automatically.

**Example**
```
procedure TBtnBottomDlg.FormCreate(Sender: TObject);
begin
 DBMultiMedia1.AutoHideMediaPlayer:=true;
end;
```

**property MediaPlayer:**
**Value**
DbMediaPlayer

**Purpose**
The ImageLib comes with its own DBmediaplayer directly derived from Tmediaplayer. You need to drop one on your form and set the property MediaPlayer to for instance: DBmediaplayer1.

**Remark**
There is no need to attach a filename to DbMediaPlayer. AutoOpen must be false since DBMultiMedia will take care of opening and closing the DbMediaPlayer.

**Example**
```
procedure TForm1.FormCreate(Sender: TObject);
begin
        DBMultiMedia1.MediaPlayer:=DBMediaPlayer1;
end;
```

**DBMediaPlayer1.Display  and DisplayRect.**

**Remark**
In order to display the video in the exact retangle of your DBMultiMedia you'll need to supply a display and rect to the DBMediaPlayer.

**Example**
```
procedure TBtnBottomDlg.DataSource1DataChange(Sender: TObject;
 Field: TField);
begin
        DBMediaPlayer1.DisplayRect:=Rect(0,0,DBMultiMedia1.Width,DBMultiMedia1.Height);
        DBMediaPlayer1.Display:=DBMultiMedia1;
end;
```

**PASCAL AND DELPHI DLL Calls and Scrolling messages File/Stream calls**

You might never have a need to make calls directly to the DLL. But incase you have a need for it we listed all the pascal interface call with the DLL. You can find all the calls in DLL22LIN.INT.

**Incase you find the supplied VCL's usefull we would like you to register. How to register:**

On CompuServe GO SWREG.  The SWREG ID = 6791.

The registration fee using SWREG is $69.-.

To register by mail send a check or moneyorder of $65.- to :

Jan Dekkers
11956 Riverside Drive, 206
North Hollywood CA 91607

**You will receive the DLL22LIN.PAS, SetSrMsg.PAS and a password to access the DLL. This will eliminate the shareware messages.**

---

I just would like to say a few words about Turbopower. I've used Turbopowers' products for over 4 years now and am very impressed with their state of the art development libraries. Their technical support is the best I have ever experienced. They provide a good example for Kevin and me of how to do business and how to treat customers.

**Turbopower's products:**

Async Professional,
B-Tree Filer,
Object Professional,
TSRs and more,
Turbo Analyst,
Turbo Professional,
Data Entry Workshop,
Win/Sys Library, and their latest great Delphi product,
Orpheus.

on CompuServe, Go PCVENB to download their free trial libraries.

**Contacting TurboPower Sales**

Telephone : 800-333-4160 (sales in the U.S. & Canada)
719-260-9136 (international sales)
719-260-7151 (fax)
CompuServe : 76004,2611
Internet : 76004.2611@compuserve.com
Postal mail : TurboPower Software
P.O. Box 49009
Colorado Springs, CO 80949-9009

---

Gif and Tiff uses LZW compression which is patented by Unisys. On CompuServe GO PICS to obtain information about the Unisys patents. This work "jpeg file i/o" is based in part on the Independent JPEG Group