DDRV.DLL exported function reference

NOTE: This document is the complete function reference for DDRV.DLL and the DDRV Print Preview system. The examples are given in the VB3 format. If you are using another language such as Delphi (Pascal) or C there are a few things that you should know about VB. VB cannot process pointers to strings - anywhere you see a ByVal x$ - you should pass a pointer to a null-terminated string. Integers are indicated by the percent (%) symbol - x%, JustShow%, etc. are all 2-byte integers. Long integers (4-bytes) are indicated by the & symbol. Single precision floats (4-byte IEEE) are proceeded by a bang (!). Strings are indicated with a $ - remember these are PChars or String pointers. Subs in VB are the same as Procedures in pascal and void functions in C. With this info, it should be fairly easy to read the syntax of this reference and use it with respect to a language other than VB. Please read the file **HOWTO.WRI** for a discussion and examples of the uses of the functions defined below.

Here is a list of all the functions exported from ddrv.dll
with VB declarations and descriptions.

---

rvInit

```
Declare Function rvInit% Lib "ddrv.dll" ( ByVal hParent%, ByVal Title$,
                                          ByVal View%,
                                          ByVal UnitOfMeas%,
                                          ByVal Options%)
```

This function is used to initialize the print previewer. You must call it before any others !!

hParent%	The handle to your application (e.g. Form1.hWnd)
Title$	A string indicating the title of the report.
View%	Use constants - RV_JUSTPRINT, RV_SHOWFIRST, or RV_CREATETEMP - Print the document, view the document or just create the document, respectively.
UnitofMeas%	Determines if your page coordinates are given in inches or cm. Use constants - RV_CM or RV_INCH
Options%	Sets the options for the printed page and screen. This is a "bitwise" parameter. Use constants - RV_PORTRAIT, RV_LANDSCAPE, RV_ZOOM100, RV_ZOOM75, RVZOOM50, RV_ZOOMFIT, RV_MAGNIFY, RV_PRINTSELECT, RV_HIDEDEFPRINTER, RV_SAVETEMPFILE.

Note about Options%: This is bitwise so you would pass parameters like this:
RV_PORTRAIT or RV_ZOOMFIT or RV_MAGNIFY

RV_PORTRAIT / RV_LANDSCAPE initialize the page orientation.
RV_ZOOMxxx sets the initial zoom setting of the viewer - 100%, 75%, 50% or FIT.
RV_MAGNIFY will enable the magnifying glass zoom cursor - the left mouse button zooms in and the right mouse button zooms out.
RV_PRINTSELECT - when this bit is set, the viewer will prompt the user for a page range, collating sequence for multiple copies, and allows the user to select an alternate printer. The common dialog box is utilized. If this bit is not set, the viewer print the whole document to the default printer.
RV_HIDEDEFPRINTER - Set this bit to hide the button that lets you change the default printer.
RV_SAVETEMPFILE - Tells the viewer not to delete the temp file created by ddrv.dll.

---

rvViewerPos

```
Declare Sub rvViewerPos Lib "ddrv.dll"  ( ByVal x%, ByVal y%,
                                          ByVal w%, ByVal h%,
                                        ByVal ShowControl%)
```

This function is used to set the initial position of the print preview screen.

x%, y%	These define the coordinate of the upper-left hand corner of the screen. These units should be passed as pixels. If you wish to initialize the viewer in a maximized or minimized state, you can pass the constants RV_MINIMIZE or RV_MAXIMIZE in the x% parameter. If you do this, ddrvexe.exe will ignore the values of y%, w%, and h%.
w%, h%	The width and height of the screen in pixels.
ShowControl%	This can be either RV_CTLBOX (show the form's control box) or RV_NOCTLBOX (don't show the form's control box)

If you don't call this function, default screen size and positions will be used and the control box will not be visible.

---

rvButtonCaption

```
Declare Sub rvButtonCaption Lib "ddrv.dll"   (  ByVal Which%,
                                                ByVal
TheCaption$)
```

Should the programmer wish to change the wording of the toolbar hints from their defaults (for instance to have multi-language support - thanks Thomas Kirstein in Germany <g>), this function will allow the programmer to change the hint wording.

Which%	A constant indicating which button you want to change the hint for. RV_BTN_EXIT, RV_BTN_PRINT, etc. - see constants below.
TheCaption$	The string containing the new hint.

e.g.

if you wish to make the exit button hint say "Good Bye", you would pass a call like this:
rvButtonCaption RV_BTN_EXIT, "Good Bye"

---

rvButtons

```
Declare Sub rvButtons Lib "ddrv.dll" ( ByVal w%, ByVal h%,
                            ByVal GraphicFile$)
```

Programmers sometimes have a special look for their programs - graphic style, etc. To help accomodate different tastes, I allow the programmer to change the bitmaps on the buttons, by sending the previewer one bitmap containing all of the needed button bitmaps.

w%	The width of the bitmap on the button (1/7 of the actual width of the bitmap)
h%	The height of the bitmap
GraphicFile$	The name of the file containing the button bitmaps

The bitmap must contain the bitmaps for all of the buttons lined up next to each other, so the width of your bitmap sent is 7 times the width of a single bitmap on a button. There are only 5 bit-buttons on the preview screen, the extra 2 are needed for the disabled state of the Prev and Next buttons. When you draw a bitmap the images should be aligned as follows:

1st bitmap	Goes on the "Exit" button
2nd bitmap	Goes on the "Prev" button in the enabled state
3rd bitmap	Goes on the "Prev" button in the disabled state
4th bitmap	Goes on the "Next" button in the enabled state
5th bitmap	Goes on the "Next" button in the disabled state
6th bitmap	Goes on the "Print" button
7th bitmap	Goes on the "Choose Printer" button

There is a sample alternate button bitmap - alt_tool.bmp - included with the sample files. If you don't understand what I tried to explain above, open this bitmap up in PaintBrush - a picture is worth a 1000000000 words <g>. Note also that the panel that contains the buttons and the zoom combo box will resize it's height to fit the new button size. There is no error checking in place now to test for bitmaps that are too big, so don't go crazy using HUGE bmps or you may get some strange results.

---

rvWhatIsTempFile

```
Declare Function rvWhatIsTempFile Lib "ddrv.dll" (ByVal FileName$)
                                                  as Integer
```

This function fills the FileName$ character buffer with the name of the temp file that ddrv.dll is currently creating. The number of characters that were copied into the buffer is returned by the function. Remember that FileName$ must be pre-dimensioned by either pre-allocating the space for the string (Dim FileName as String * 129) or by dynamically allocating the space (FileName$ = Space$(129). I suggest allowing for up to 129 characters. If the space is not properly allocated, you will get an ugly GPF.

The main use for this function is to be able to call the document up later (pass the RV_CREATETEMP constant in the ViewOrPrint% parameter of rvInit). See HOWTO.WRI for information on how to call up an existing document file.

---

rvGetHandle

```
Declare Function rvGetHandle Lib "ddrv.dll" ()as Integer
```

This function takes no parameters and returns the handle of the last viewer created. It will return an unpredicatable integer if the viewer has not been created, and a zero if the viewer has been created and destroyed. It is suggested that if the programmer is going to allow a user to have more than one viewer open at a time, that they store the handles of viewers in an array for future reference. This function should be called shortly after rvEndDoc to ensure a correct return.

---

rvViewerActive

```
Declare Function rvViewerActive Lib "ddrv.dll" (hWnd%) as Integer
```

This function takes a window handle and returns a zero if the window handle is valid, ie. if the window exists. The parameter should be obtained with rvGetHandle.

---

rvGetCloseHandle

```
Declare Function rvGetCloseHandle Lib "ddrv.dll" ()as Integer
```

This function takes no parameters and returns the handle of the close button in the last viewer created.It will return zero if the viewer has been destroyed. It is suggested that if the programmer is going to allow a user to have more than one viewer open at a time, that they store the close handles of viewers in an array for future reference. This function must be called shortly after rvEndDoc to ensure that it is accurate.

---

rvKillViewer

```
Declare Sub rvKillViewer Lib "ddrv.dll" (hWndClose%)
```

This routine takes the close handle - retrieved with rvGetCloseHandle, and uses it to close the viewer referenced by the close handle. Please, please note that the handle passed here is **not** the handle of the viewer window, it is the handle of the close button on the viewer window. It can most easily be retrieved with rvGetCloseHandle.

---

rvFloatViwer

```
Declare Sub rvFloatViewer Lib "ddrv.dll" (hWnd%, YesOrNo%)
```

This sub takes a window handle, and a 0/non-0 value as it's parameters. The call forces that window to float on top of all other windows on the desktop if the value of YesOrNo% is non-zero. It returns the window to a normal state if YesOrNo% = 0. Typically this function should be used by passing the value returned by rvGetHandle. It may, however, be used to float any window whose handle is known. e.g. rvFloatViewer(Form1.hWnd, 1) will force the VB form called Form1 to float above all others.

---

rvNewPage

```
Declare Sub rvNewPage Lib "ddrv.dll" ()
```

This procedure tells ddrv.dll to start a new page.

---

rvEndDoc

```
Declare Sub rvEndDoc Lib "ddrv.dll" ()
```

This procedure tells ddrv.dll to end the document. No more calls can be made to the dll after this is called, unless the viewer is initialized again by calling rvInit.

---

rvText

```
Declare Sub rvText Lib "ddrv.dll" (ByVal x!, ByVal y!, ByVal align%,
                                   ByVal TheText$)
```

This procedure prints text out on the screen at the location x, y using the alignment align%.

x!	The left-most postion of the text in inches.
y!	The top position of the text in inches.
align%	Use the constants - RV_LEFT, RV_CENTER, RV_DECIMAL, and RV_RIGHT to align the text at the x,y postition indicated.
TheText$	A null-terminated string that will be sent to the previewer.

e.g.

```
TheText$ = "This is a test"
rvText 4.25, 1, RV_CENTER, TheText$
```

This tells ddrv.dll to print TheText$ on the screen centered at 4.25 inches over and 1 inch down.

---

rvAngleText

```
Declare Sub rvAngleText Lib "ddrv.dll" (x!, y!, angle%, TheText$)
```

This subroutine writes text to the screen at an angle given in degrees - angle%.

x!	The x coordinate to start printing at
y!	The y coordinate to start printing at
angle%	The angle to print the text at in degrees. 0 degrees prints normal 180 prints upside-down, 90 prints straight up. The degrees are measured counter-clockwise from a horizon of 0 degrees.
TheText$	A null-terminated string that will be sent to the previewer.

---

rvParagraph

```
Declare Sub rvParagraph Lib "ddrv.dll" (x1!, y1!, x2!, y2!, TheText$)
```

This subroutine writes text to the viewer and wordwraps it to fit into the bounding rectangle defined by the corner points (x1!, y1!) and (x2!, y2!).

x1!, y1!	These coordinates mark the upper-left hand corner of the bounding rectangle.
x2!, y2!	These coordinates mark the lower-right hand corner of the bounding rectangle.
TheText$	A null-terminated string that will be sent to the previewer.

NOTE: This function will not print any text that falls below the bottom of the rectangle, nor will it tell you how much text was actually printed. I suggest using this call to print small paragraphs, or to print a paragraph that will for sure fit in the bounding rectangle.

---

rvGetEditBoxLine

```
Declare Sub rvGetEditBoxLine Lib "ddrv.dll" (ByVal hWnd%, Which&,
                                             ByVal TheBuffer$)
```

hWnd%	The handle of the edit box
Which&	Either 0 if you wish to know how many lines are in the edit box, or an integer value representing the line you wish to retrieve. If you pass a 0, Which& is changed to contain the number of lines in the edit box after the call is made. If you pass > 0, Which& will return with the number of characters in the line and TheBuffer$ will return with the contents of the line. The fact that this is not declared with the ByVal keyword is not a mistake, it needs to pass the address of the long integer. Non-VB users should pass this parameter as a pointer to a 4-byte integer.
TheBuffer$	A pre-allocated string space which will be filled with the contents of a line in the edit box. Be careful to pre-dimension the space for this variable.

This function can be used to read through the lines in an edit box whose handle is passed as a parameter. If the Which& parameter is passed as 0, the routine will fill Which& with the number of lines in the edit box when it returns. If Which& is >0, it will contain the number of characters in the line asked for. See the file HOWTO.WRI for more information on this function and it's potential uses to programmers needing to print paragraphs of text to a preview screen.

---

rvRect

```
Declare Sub rvRect Lib "ddrv.dll" (ByVal x1!, ByVal y1!, ByVal x2!,
                                   ByVal y2!, ByVal p!, ByVal s%)
```

This procedure draws a rectangle from (x1, y1) to (x2, y2) with a border thickness of p! points, where x1, y1, x2, and y2 are print-out positions in inches. s% indicates a style for the fill of the rectangle (either RV_GREYBACK for a grey background or RV_CLEARBACK for a white background.

---

rvGraphic

```
Declare Sub rvGraphic Lib "ddrv.dll" ( ByVal FileName$, ByVal x1!,
ByVal                                  y1!, ByVal x2!, ByVal y2!)
```

This procedure loads in a bitmap given it's file name (FileName$). It will stretch the bitmap to fit in the rectangle defined by (x1!, y1!) and (x2!, y2!) in inches. If the bitmap file cannot be found an error message will appear on the screen.

---

rvFontName

```
Declare Sub rvFontName Lib "ddrv.dll" (ByVal FontName$)
```

This procedure changes the current font to be FontName$

e.g. rvFontName "Arial"
will change the font name to be Arial.

---

rvFontSize

```
Declare Sub rvFontSize Lib "ddrv.dll" (ByVal FontSize%)
```

This procedure changes the current font size to be FontSize% points.

e.g. rvFontSize 10
will change the current font size to be 10 points.

---

rvFontStyle

```
Declare Sub rvFontStyle Lib "ddrv.dll" (ByVal FontStyle%)
```

This procedure will change the style of the current font. The FontStyle% parameter is a bitwise representation of the style. Use the constants defined below.

e.g. rvFontStyle RV_FONTBOLD or RV_FONTITALIC
will change the font style to be bold and italic.

e.g. rvFontStyle RV_FONTNORMAL
will change the font style to be normal text.

Constants

used by rvFontStyle

```
Global Const RV_FONTNORMAL = 0
Global Const RV_FONTBOLD = 1
Global Const RV_FONTITALIC = 2
Global Const RV_FONTUNDERLINE = 4
Global Const RV_FONTSTRIKEOUT = 8
```

used by rvText for alignment

```
Global Const RV_LEFT = 0
Global Const RV_CENTER = 1
Global Const RV_RIGHT = 2
Global Const RV_DECIMAL = 3
```

used by rvRect for background style

```
Global Const RV_CLEARBACK = 0
Global Const RV_GREYBACK = 2
```

used by rvInit for setting the units of measure

```
Global Const RV_INCH = 0
Global Const RV_CM = 1
```

used by rvInit for print control

```
Global Const RV_JUSTPRINT = 0
Global Const RV_VIEWFIRST = 1
Global Const RV_CREATETEMP = 2
```

used bit-wise in rvInit in the options parameter these options are also passed on the command line to ddrvexe if you are calling up an existing document (see HOWTO.WRI)

```
Global Const RV_PORTRAIT = 0
Global Const RV_LANDSCAPE = 1
Global Const RV_MAGNIFY = 2
Global Const RV_PRINTSELECT = 4
Global Const RV_ZOOM100 = 0
Global Const RV_ZOOM75 = 8
Global Const RV_ZOOM50 = 16
Global Const RV_ZOOMFIT = 32
Global Const RV_HIDEDEFPRINTER = 64
Global Const RV_SAVETEMPFILE = 128
```

used by rvViewPos in the X% parameter to control initial window state

```
Global Const RV_MAXIMIZE = -1
Global Const RV_MINIMIZE = -2
```

used by rvViewPos to show/hide the window's control box

```
Global Const RV_NOCTLBOX = 0
Global Const RV_CTLBOX = 1
```

used by rvButtonCaption in the Which% parameter

```
Global Const RV_BTN_EXIT = 0
Global Const RV_BTN_PREV = 1
Global Const RV_BTN_NEXT= 2
Global Const RV_BTN_PRINT = 3
Global Const RV_BTN_PRINTER = 4
Global Const RV_CBO_ZOOM = 5
```

Special Programming Note

Each time you start a new page by either initializing the viewer (this starts the first page) or by calling rvNewPage, you should immediately set the font information for that page. Call rvFontName, rvFontSize, rvFontStyle right after each call to rvNewPage or rvInit. If you don't do this you will get unpredictable font styles on that page if the user moves backwards a page in the document, prints a page range, or views the first page for the first time.

Also, be aware that ddrv.dll will allow you to print to the edge of the page (at 0" and at 8.5" or even off of the page at 100"). Printers have defined a certain printable area - on the laser jet it is about 8" by 10.6". Any text printed in the margins will not be printed on paper. It is up to the programmer to all for unprintable margins. I usually don't print within .35" of any edge of the page.

Make sure you read the document HOWTO.WRI for more examples of how to use the features of the DDRV Print Preview system.