vsView 2.0

VideoSoft Custom Control Library

To learn how to use help, press F1

Introduction
Find out about installation, product support, licensing, registration, and other VideoSoft products.

vsInForm
Customize forms and controls, monitor the clipboard, drag and drop files from the File Manager.

vsPrinter
Print files, text, graphics, and tables with automatic word wrap, headers and footers, multiple columns, and previewing.

vsViewPort
Fit more controls on your windows with virtual scrollable areas.

vsDraw
Create complex images, view them on the screen, copy them to the clipboard, and print them.

Introduction

Welcome to **vsView** 2.0, a VideoSoft Custom Control Library. VideoSoft custom controls are innovative, flexible, and powerful. If you like them, make sure you check out our other award-winning products, VSVBX and VSFLEX.

Our distribution policy is almost as innovative as the controls. We want every Visual Basic programmer to get a copy of **vsView** and try it for as long as they want. Those who like the product and find it useful (almost everybody, we hope) can buy a license for a reasonable price. The only restriction is that unlicensed copies of **vsView** display a VideoSoft banner whenever they are loaded, to remind developers to license the product.

We hope you'll like **vsView**. If you have suggestions and ideas for new features or new controls, call us or write.

VideoSoft
2625 Alcatraz Avenue, Suite 271
Berkeley, CA 94705
(510) 704-8200 (phone)
(510) 843-0174 (fax)

Control Summary

Icon	Object	Name	Description
	**vsInForm**	InForm	A control that you can drop into any container to customize its title bar, frame, resizing behavior, and frame buttons. InForm also allows you to monitor the clipboard, drag and drop files from File manager, and more.
	**vsPrinter**	Printer	A much improved printer object with word wrap, headers and footers, multi-column printing, graphics, and multi-page Print Preview capability.
	**vsViewPort**	ViewPort	A control that gives you a scrollable virtual area so you can fit more controls in your windows. Great for implementing Print Preview and programs that look like the Program Manager.
	**vsDraw**	Draw	A versatile drawing control that lets you create complex images, view them on the screen, copy them to the clipboard, or print them. Great for technical drawings, maps, and diagrams.

Installation

To install the OCX version of **vsView**, use the INSTALL utility provided on the distribution diskette.

To install the VBX version of vsView, just copy the following files to your WINDOWS\SYSTEM directory:

VSVIEW.VBX This file contains the controls. To use vsView from Visual Basic, you must include this file in your project.

VSVIEW.LIC This is the vsView license file. If vsView cannot find this file when it starts running, it displays a VideoSoft banner and waits for the user to click Ok, unless the project was compiled on a machine that had vsView.LIC installed.

VSVIEW.HLP This file contains the vsView on-line help.

Distribution

vsView is royalty-free. You may include copies of the VBX and HLP files with as many copies of as many applications you ship.

You cannot distribute the license file VSVIEW.LIC. And you don't have to: as long as you have the license file installed on your machine, vsView will stamp every application you compile so the banner will not appear when your users run the applications.

If you work with other developers, you may be interested in VideoSoft's site licenses. Call us for details.

If you haven't yet registered your copy of VSVIEW and would like to do it now, click **HERE** to get a Registration Form.

Product Support

Product support for VSVIEW is available to licensed users through the following channels:

Compuserve	CIS 74774,420 or join our forum by typing "**GO VIDEOSOFT**"
Mail	VideoSoft 2625 Alcatraz Avenue, Suite 271 Berkeley, California 94705
Phone	(510) 704-8200
Fax	(510) 843-0174

Before calling for technical support, please make sure you know what version of VSVIEW you are using. The version number appears in the About box that pops up when you double-click the About property in any of the VSVIEW controls.

Also, please make sure you check the last section of the manual, "Hints and Troubleshooting". It contains answers to the most common questions people ask our technical support staff. Maybe you can find your answer there.

vsInForm Reference

Description The **vsInForm** control allows you to customize the non-client area of its parent form or control. With **vsInForm**, you can add a chiseled look to your window borders and caption, add custom buttons, and control window resizing. And because **vsInForm** works with any container control, you can add these effects not only to forms, but also to **Picture Boxes**, **Frames**, **Elastics**, **IndexTabs**, and so on.

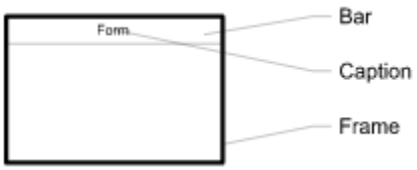
vsInForm also allows you to monitor the clipboard, implement file drag-and-drop, and retrieve information about the environment.

File Name VSVIEW.VBX for the VBX version,
VSVIEW32.OCX for the 32-bit OCX version, or
VSVIEW16.OCX for the 16-bit OCX version

Object Type vsInForm

Note Before you can use a **vsInForm** control in your application, you must add VSVIEW to your project (see the Visual Basic manual for details). To automatically include VSVIEW in new projects, put it in an AUTOLOAD file. When distributing your application, you should install the vsView files in the user's Microsoft Windows SYSTEM subdirectory.

Remarks **vsInForm** has many properties that are used to customize its parent's non-client area (title bar and window frame.) These properties are divided in three groups: **Bar*** properties control the title bar, **Cap*** properties control the caption inside the title bar, and **Frame*** properties control the window frame. The picture below illustrates what parts of the window these names refer to:



The **CustomFrame** property determines whether the **vsInForm** control should or should not take over managing the non-client area of the **vsInForm**'s parent. If you set this property to **True** and the parent is a form, the form must not have a menu. Remember also to set the form's **ControlBox** property to **False**, or users will be able to use alt-space to show the system menu.

When using the **vsInForm** control, keep in mind that one of Windows's great virtues is that it promotes a standard interface across applications. Generally, it is not a good idea to change this standard. There are many exceptions to this rule, though: Visual Basic itself is a good example.

VB's floating toolbox and palette windows have skinny title bars, to save room. VB's main window can be resized horizontally, but not vertically. With vsInForm, your applications can have those features too.

vsInForm Summary

Properties (default: Caption)

(About)	* AcceptFiles	* BarColor
* BarColorInactive	* BarHeight	* BarStyle
* ButtonsLeft	* ButtonsRight	* CapAlign
* CapColor	* CapColorInactive	* CapMultiLine
* CapStyle	Caption	* ClipMon
* CustomFrame	* FileName	FontBold
FontItalic	FontName	FontSize
FontStrike	FontUnder	* FrameButtons
* FrameColor	* FrameColorInactive	* FrameCorners
* FrameSizing	* FrameStyle	* FrameWidth
* FreeGDI	* FreeMemory	* FreeSystem
* FreeUser	hWnd	Index
Left	* MaxHeight	* MaxWidth
* MinHeight	* MinWidth	Name
* NumFiles	* OnTop	Parent
* PictLeft0	* PictLeft1	* PictLeft2
* PictRight0	* PictRight1	* PictRight2
Tag	Top	* Version

Events

* ClickCaption	* ClickLButton	* ClickRButton
* DblClickLButton	* DblClickRButton	* DblClickCaption
* DropFile	* Move	* NewClipboardData
* Resize		

AcceptFiles Property

Description Sets or returns a value that determines whether the user should be able to drop files from the File Manager into the **vsInForm**'s parent window..

Syntax [*form.*]*vsInForm*.**AcceptFiles** [= {**True**|**False**}]

Remarks If this property is set to **True**, the **DropFile** event is fired when the user drops a file into the **vsInForm**'s parent.

This property is not available in the OCX edition of the control.

Default Value **False**

Data Type Integer (Boolean)

BarColor, BarColorInactive Properties

Description These are the colors used to paint the caption bar of the **vsInForm**'s parent. At run time, **vsInForm** chooses whether to use **BarColor** or **BarColorInactive** automatically, based on whether the **vsInForm**'s parent is active or not.

Syntax [*form*.]*vsInForm*.**BarColor** [= *color&*]

Remarks These properties are used only if the **CustomFrame** property is set to **True**.

Setting either of these properties to zero causes the system default colors to be used. If you want the caption to be black, set these properties to one (if you simply pick black from the color box, VB will pick color zero and the control will use the default system colors).

Default Value Zero (use system colors)

Data Type Long (Color)

BarHeight Property

Description Sets or returns the height of the caption bar of the **vsInForm**'s parent, in pixels.

Syntax [*form.*]*vsInForm*.**BarHeight** [= *height%*]

Remarks This property is used only if the **CustomFrame** property is set to **True.**

Setting this property to zero causes the system default caption height to be used. If you want a window with no caption, set the **BarStyle** property to *0 - None*.

Default Value Zero (use system value)

Data Type Integer

BarStyle Property

Description Sets or returns a value that determines the appearance of the caption bar of the **vsInForm**'s parent.

Syntax [*form*.]*vsInForm*.**BarStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - None
1 - Classic
2 - No Border
3 - Raised
4 - Inset

This property is used only if the **CustomFrame** property is set to **True**.

Default Value 1 - Classic

Data Type Integer (Enumerated)

ButtonsLeft, ButtonsRight Properties

Description These properties determine how many buttons should be displayed on the left and right-hand side of the **vsInForm** parent's caption bar. The buttons are similar to the standard Windows maximize and minimize buttons, except you can customize the pictures in the buttons and attach custom code to clicks and double clicks.

Syntax [*form*.]*vsInForm*.**ButtonsLeft** [= *setting%*]

Remarks The number of buttons ranges from 0 to 3 on each side.

These properties are used only if the **CustomFrame** property is set to **True**.

Default Value Zero

Data Type Integer

CapAlign Property

Description Sets and returns the alignment of the text in the **vsInForm** parent's caption bar.

Syntax [*form.*]*vsInForm*.**CapAlign** [= *setting%*]

Remarks Valid settings for this property are:

0 - Left Top
1 - Left Center
2 - Left Bottom
3 - Center Top
4 - Center Center
5 - Center Bottom
6 - Right Top
7 - Right Center
8 - Right Bottom

This property is used only if the **CustomFrame** property is set to **True**.

Default Value 4 - Center Center

Data Type Integer (Enumerated)

CapColor, CapColorInactive Properties

Description These are the colors used to paint the text in the caption bar of the **vsInForm**'s parent. At run time, InForm chooses whether to use **CapColor** or **CapColorInactive** automatically, based on whether the **vsInForm**'s parent is active or not.

Syntax [*form*.]*vsInForm*.**CapColor** [= *color&*]

Remarks These properties are used only if the **CustomFrame** property is set to **True**.

Setting either of these properties to zero causes the system default colors to be used. If you want the caption to be black, set these properties to one (if you simply pick black from the color box, VB will pick color zero and the control will use the default system colors).

Default Value Zero (use system colors)

Data Type Color (Long)

CapMultiLine Property

Description Sets or returns a value that determines whether the text in the **vsInForm** parent's caption bar should wrap, allowing for multi-line captions.

Syntax [*form*.]*vsInForm*.**CapMultiLine** [= {**True|False**}]

Remarks This property is used only if the **CustomFrame** property is set to **True**.

Default Value **False**

Data Type Integer (Boolean)

CapStyle Property

Description Sets or returns a value that determines the appearance of the text in the caption bar of the **vsInForm**'s parent.

Syntax [*form*.]*vsInForm*.**CapStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Flat
1 - Raised
2 - Inset
3 - Raised Light
4 - Inset Light

For best effects with 3D captions, set the **CapColor** and **CapColorInactive** properties to dark gray.

This property is used only if the **CustomFrame** property is set to **True**.

Default Value 0 - Flat

Data Type Integer (Enumerated)

ClickCaption Event

Description Fired after the caption of the **vsInForm**'s parent is clicked.

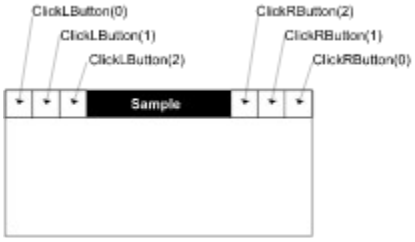
Syntax **Sub** *vsInForm*_**ClickCaption ()**

ClickLButton, ClickRButton Events

Description Fired after one of the custom caption buttons is clicked.

Syntax **Sub** *vsInForm*_**ClickLButton (***Button%***)**

Remarks Buttons are numbered from the outside towards the center of the form, as shown in the diagram below:



ClipMon Property

Description Determines whether the **vsInForm** should monitor Clipboard activity.

Syntax [*form.*]*vsInForm*.**ClipMon** [= {**True**|**False**}]

Remarks If this property is set to **True**, the **vsInForm** control fires the **NewClipboardData** event whenever the contents of the Clipboard change.

Default Value **False**

Data Type Integer (Boolean)

CustomFrame Property

Description Determines whether the **vsInForm** should take over painting and managing the non-client area of its parent.

Syntax [*form*.]*vsInForm*.**CustomFrame** [= {**True**|**False**}]

Remarks If this property is set to **True**, the appearance of the **vsInForm**'s parent and its caption are controlled through the other **vsInForm** properties.

Default Value **False**

Data Type Integer (Boolean)

DblClickCaption Event

Description Fired after the caption of the **vsInForm**'s parent is double-clicked.

Syntax **Sub** *vsInForm*_**DblClickCaption ()**

Remarks Normally, you should trap this event to implement the Windows standard behavior: flip the window state between maximized and normal.

DblClickLButton, DblClickRButton Events

Description Fired after one of the custom caption buttons is double-clicked.

Syntax **Sub** *vsInForm*_**DblClickLButton (***Button%***)**

Remarks Buttons are numbered from the outside towards the center of the form. See diagram in the description of the **ClickLButton** event.

DropFile Event

Description Fired when the user drops one or more files from another application – such as the Program Manager or the File Manager – into the **vsInForm**'s parent.

Syntax **Sub** *vsInForm*_**DropFile ()**

Remarks To enable this feature, you must set the **vsInForm**'s **AcceptFiles** property to **True**.

To find out which files were dropped into the control, use the **vsInForm**'s **NumFiles** and **FileName** properties, as shown below.

```
Sub vsInForm_DropFile ()
  Dim i%
  For i = 0 to vsInform.NumFiles
  ProcessFile vsInForm.FileName(i)
  Next
End Sub
```

FileName Property

Description Returns the names of files dropped into the **vsInForm**'s parent.

Syntax *filename$* = [*form.*]*vsInForm*.**FileName(***index%***)**

Remarks This property is read-only and should only be used while handling the **vsInForm**'s **DropFile** event. The number of files dropped can be determined by reading the **NumFiles** property.

See the description of the **DropFile** event for an example.

Data Type String Array

FrameButtons Property

Description Sets or returns a value that affects the look of the buttons on the **vsInForm**'s custom frame. If set to **True**, a thin black border is drawn around the buttons. If set to **False**, no border is drawn and the buttons appear "immersed" in the caption bar.

Syntax [*form*.]*vsInForm*.**FrameButtons** [= {**True**|**False**}]

Remarks This property has no effect if the **CustomFrame** property is set to **False** or if the custom frame has no buttons.

Default Value **True**

Data Type Integer (Boolean)

FrameColor, FrameColorInactive Properties

Description These are the colors used to paint the frame of the **vsInForm**'s parent. At run time, the **vsInForm** control chooses whether to use **FrameColor** or **FrameColorInactive** automatically, based on whether the **vsInForm**'s parent is active or not.

Syntax [*form*.]*vsInForm*.**FrameColor** [= *color&*]

Remarks These properties are used only if the **CustomFrame** property is set to **True**.

Setting either of these properties to zero causes the system default colors to be used. If you want the frame to be black, set these properties to one (if you simply pick black from the color box, VB will pick color zero and the control will use the default system colors).

Default Value Zero (use system colors)

Data Type Color (Long)

FrameCorners Property

Description Sets or returns a value that determines whether the **vsInForm** should mark the resizing corners on its parent's frame.

Syntax [*form*.]*vsInForm*.**FrameCorners** [= {**True**|**False**}]

Remarks This property is used only if the **CustomFrame** property is set to **True**.

This property only affects the window appearance, not its resizing behavior.

Default Value **True**

Data Type Integer (Boolean)

FrameSizing Property

Description Sets or returns a value that determines whether the **vsInForm** should allow its parent to be resized in both directions, in one direction, or not at all.

Syntax [*form*.]*vsInForm*.**FrameSizing** [= *setting%*]

Remarks Valid settings for this property are:

0 - None Move freely, but no resizing
1 - Horizontal Move freely, resize width only
2 - Vertical Move freely, resize height only
3 - Both Move freely, resize freely
4 - Locked No moving, no resizing

This property is used only if the **CustomFrame** property is set to **True**.

Default Value 3 - Both

Data Type Integer (Enumerated)

FrameStyle Property

Description Sets or returns a value that determines the appearance of the resizing frame around the **vsInForm**'s parent.

Syntax [*form*.]*vsInForm*.**FrameStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - None
1 - Classic
2 - Outside
3 - Raised Form
4 - Raised Frame

This property is used only if the **CustomFrame** property is set to **True**.

Default Value 1 - Classic

Data Type Integer (Enumerated)

FrameWidth Property

Description Sets or returns the width of the frame around the **vsInForm**'s parent, in pixels.

Syntax [*form.*]*vsInForm*.**FrameWidth** [= *width%*]

Remarks This property is used only if the **CustomFrame** property is set to **True**.

Setting this property to zero causes the system default frame width to be used. If you want a window with no frame, set the **FrameStyle** property to *0 - None*.

Default Value Zero (use system value)

Data Type Integer

FreeGDI, FreeSystem, FreeUser Properties

Description Return the percentage of free Windows GDI, system, and user resources.

Syntax
GDI% = [*form.*]*vsInForm*.**FreeGDI**
Sys% = [*form.*]*vsInForm*.**FreeSystem**
Usr% = [*form.*]*vsInForm*.**FreeUser**

Remarks These properties are not available in the OCX edition of the control.

Data Type Integer

FreeMemory Property

Description Return amount of free memory, in bytes.

Syntax *Mem&* = [*form*.]*vsInForm*.**FreeMemory**

Remarks This property is not available in the OCX edition of the control.

Data Type Long

MaxHeight, MinHeight Properties

Description Set or return limits to the height of the **vsInForm**'s parent, in Twips.

Syntax [*form.*]*vsInForm*.**MaxHeight** [= *maxhei&*]
[*form.*]*vsInForm*.**MinHeight** [= *minhei&*]

Remarks These properties are useful for windows that only show information along one direction, such as "roll-up" toolbars or VB's main window.

These properties are used only if the **CustomFrame** property is set to **True**.

Default Value Zero

Data Type Long (Twips)

MaxWidth, MinWidth Properties

Description Set or return limits to the width of the **vsInForm**'s parent, in Twips.

Syntax [*form.*]*vsInForm*.**MaxWidth** [= *maxwid&*]
[*form.*]*vsInForm*.**MinWidth** [= *minwid&*]

Remarks These properties are useful for windows that only show information along one direction, such as "roll-up" toolbars or VB's main window.

These properties are used only if the **CustomFrame** property is set to **True**.

Default Value Zero

Data Type Long (Twips)

Move Event

Description Fired after the **vsInForm**'s parent has been moved.

Syntax **Sub** *vsInForm*_**Move ()**

Remarks To enable this feature, you must set the **vsInForm**'s **CustomFrame** property to **True**.

NewClipboardData Event

Description Fired whenever new data is copied to the Windows Clipboard. You can then use VB's Clipboard object to examine, retrieve, or change the clipboard contents.

Syntax **Sub** *vsInForm*_**NewClipboardData ()**

Remarks To enable this feature, you must set the InForm's **ClipMon** property to **True**.

NumFiles Property

Description Returns the number of files dropped into the **vsInForm**'s parent.

Syntax *numfiles%* = [*form*.]*vsInForm*.**NumFiles**

Remarks This property is read-only and should only be used while handling the **vsInForm**'s **DropFile** event. The names of the files dropped can be determined by reading the **vsInForm**'s **FileName** property.

See the description of the **DropFile** event for an example.

Data Type Integer

OnTop Property

Description Sets or returns a value that deterines whether the **vsInForm**'s parent to remain on top of other windows, even when it is not active.

Syntax [*form*.]*vsInForm*.**OnTop** [= {**True**|**False**}]

Remarks This is especially useful for floating toolbars and other small windows. A good way to implement **OnTop** functionality is to use one of the custom caption buttons to turn it on and off (see the example on page 10).

Default Value **False**

Data Type Integer (Boolean)

PictLeft*, PictRight* Properties

Description Set or return the pictures that appear in the **vsInForm** parent's custom caption bar buttons. There are 6 properties in total, left and right ranging from 0 to 2.

Syntax [*form*.]*vsInForm*.**PictLeft0** [= *picture*]
[*form*.]*vsInForm*.**PictRight0** [= *picture*]

Remarks The **vsInForm** draws the bevels automatically. All you have to supply is a small bitmap to be drawn on the button. For best results, make sure the background of the bitmaps is light gray, so it blends with the button background.

Data Type Picture

Resize Event

Description Fired after the **vsInForm**'s parent has been resized.

Syntax **Sub** *vsInForm*_**Move** **()**

Remarks There is no need for this event if the **vsInForm**'s parent is a form, since forms have their own **Resize** event. However, you need this event to trap resizing of other controls.

To enable this feature, you must set the **CustomFrame** property to **True**.

Version Property

Description Returns the version of the **vsInForm** control currently loaded in memory.

Syntax *CheckVer%* = [*form*.]*vsInForm*.**Version**

Remarks You may want to check this value at the **Form_Load** event, to make sure the version that is executing is at least as current as the version used to develop your application.

The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.

Data Type Integer

vsPrinter Reference

Description The VideoSoft **vsPrinter** control allows you to create printed output quickly and easily. It supports multiple columns, text wrapping, headers and footers, tables, and graphics. Best of all, it lets you do Print Previews just by setting a single property.

File Name VSVIEW.VBX for the VBX version,
VSVIEW32.OCX for the 32-bit OCX version, or
VSVIEW16.OCX for the 16-bit OCX version

Object Type vsPrinter

Note Before you can use a **vsPrinter** control in your application, you must add VSVIEW to your project (see the Visual Basic manual for details). To automatically include VSVIEW in new projects, put it in an AUTOLOAD file. When distributing your application, you should install the vsView files in the user's Microsoft Windows SYSTEM subdirectory.

Remarks The **vsPrinter** control has a fixed coordinate system. It uses Twips as units, with the origin located at the top left of the page. To specify measurements in inches or points, remember that one inch is equivalent to 1440 Twips, and that one point is equivalent to 20 Twips.

For convenience, two measurements are expressed in units other than Twips. Font sizes are expressed in points, and line spacing is expressed specified as a percentage of the current font height.

When used in print preview mode, there are two ways to control the scale:

If you set the **Zoom** property to a value greater than zero, it is used as the percentage of the actual page size. Scroll bars are automatically added to the control if they are needed.

If you set the **Zoom** property to zero, the scale is adjusted so that a whole page always fits the control. In this case, you must adjust the control size in code, and possibly place the control in a **vsViewPort** to allow scrolling.

vsPrinter Summary

Properties (default: Paragraph)

(About)	* AbortCaption	* AbortTextButton
* AbortTextDevice	* AbortTextPage	* AbortWindow
* Action	BackColor	BorderStyle
* BrushColor	* BrushStyle	* CalcParagraph
* CalcTable	* CalcText	* Collate
* ColorMode	* Columns	* Copies
* CurrentColumn	* CurrentLine	* CurrentPage
* CurrentX	* CurrentY	* Device
* Devices	* DPI	* Draw
* Driver	* Duplex	* EmptyColor
Enabled	* Error	* FileName
FontBold	FontItalic	FontName
FontSize	FontUnderline	* Footer
* hDC	* HdrColor	* HdrFontBold
* HdrFontItalic	* HdrFontName	* HdrFontSize
* HdrFontUnderline	* Header	Height
HelpContextID	* IndentFirst	* IndentLeft
* IndentRight	* IndentTab	Index
Left	* LineSpacing	* MarginBottom
* MarginLeft	* MarginRight	* MarginTop
* Measure	* MousePage	MousePointer
* MouseScroll	* MouseZoom	Name
* NDevices	* NPorts	* Orientation
* PageBorder	* PageHeight	* PageWidth
* PaperBin	* PaperHeight	* PaperSize
* PaperWidth	* Paragraph	Parent
* PenColor	* PenStyle	* PenWidth
* PhysicalPage	* Picture	* Polygon
* Polyline	* Port	* Ports
* Preview	* PreviewMode	* PreviewPage
* PrintQuality	* Scale	* SpaceAfter
* SpaceBefore	* Table	* TableBorder
* TableSep	Tag	* Text
* TextAlign	* TextAngle	* TextColor
* TextHei	* TextHeight	* TextRTF
* TextWid	* TextWidth	Top
* TrueType	* TwipsPerPixelX	* TwipsPerPixelY
* Version	Visible	Width
* X1	* X2	* Y1
* Y2	* Zoom	

Events

Click	DblClick	DragDrop
DragOver	* EndDoc	* EndPage
* Error	KeyDown	KeyPress
KeyUp	* LoadingDoc	* SavingDoc
MouseDown	MouseMove	MouseUp
* NewColumn	* NewLine	* NewPage
* NewTableCell	* ResetDC	* StartDoc

Methods

* AddTable	* DrawCircle	* DrawEllipse
* DrawLine	* DrawRectangle	* EndDoc
* KillDoc	* LoadDoc	Move
* NewColumn	* NewPage	* PrintDoc
Refresh	* SaveDoc	SetFocus
* StartDoc	ZOrder	

AbortCaption Property

Description Sets the caption for the default Abort dialog.

Syntax [*form.*]*vsPrinter.***AbortCaption** [= *caption$*]

Remarks See the **AbortWindow** property for details on the default Abort dialog.

Default Value “Printing...”

Data Type String

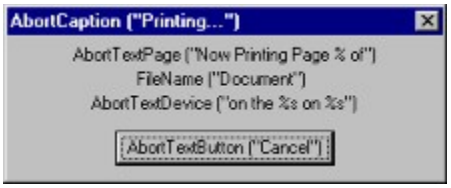
AbortTextButton, AbortTextDevice, AbortTextPage Properties

Description These properties determine the text that is displayed in the **vsPrinter**'s built-in abort dialog box.

Syntax [*form!*]*vsPrinter.***AbortTextButton**[= *string*]
[*form!*]*vsPrinter.***AbortTextDevice**[= *string*]
[*form!*]*vsPrinter.***AbortTextPage**[= *string*]

Remarks Use these properties to customize or internationalize **vsPrinter**'s built-in abort dialog.

The picture below shows where each of the strings is displayed: At each position, the picture shows the corrsponding property name and the default string.



The "%d" in the **AbortTextPage** string gets replaced with the current page number. The "%s" in the **AbortTextDevice** string get replaced with the current device and port names.
These properties only work when the **AbortDialog** property is set to **True**.

Default Value **AbortTextButton** = "Cancel"
AbortTextDevice = "on the %s on %s"
AbortTextPage = "Now Printing Page %d of"

Data Type String

AbortWindow Property

Description Sets or returns a value that determines whether the **vsPrinter** should automatically display an Abort dialog while it prints.

Syntax [*form*.]*vsPrinter*.**AbortWindow** [= {**True**|**False**}]

Remarks The automatic Abort dialog shows the name of the document being printed (from the **FileName** property), the output device, port, and the current page. If the user presses the Abort button, printing stops and an **Error** event is fired.

You may modify the text in the automatic Abort dialog by setting the contents of the **AbortCaption**, **FileName**, **AbortTextPage**, **AbortTextDevice**, and **AbortTextButton** properties. These properties are especially useful if you are writing applications in a language other than English.

Set this property to **False** if you want to display a custom Abort dialog.

Default Value True

Data Type Integer (Boolean)

Action Property

Description Method-style property that allows you to control the printer.

Syntax [*form.*]*vsPrinter.***Action** = *action%*

Remarks Valid settings for this property are:

0 - None
1 - Print File
2 - Choose Printer & Print File
3 - StartDoc
4 - NewPage
5 - NewCol
6 - EndDoc
7 - KillDoc
8 - Print Page
9 - Choose Printer & Print Page
10 - Copy Page
11 - Print All Pages
12 - Choose Printer & Print All Pages

The *Print File* action prints an entire text file using the current settings for font, margins, header and footer, and number of columns. The file name must be specified in advance by setting the **FileName** property. For example:

```
vsPrinter.FileName = "C:\AUTOEXEC.BAT"
vsPrinter.Action = 1    ' print the AUTOEXEC.BAT file
```

The *Choose Printer & Print File* action is similar to the *Print File* action, except it displays a dialog box that allows the user to choose the printer to be used.

The *StartDoc* and *EndDoc* actions are used to create and end print jobs. All other printing commands should be called between these actions. The example below illustrates this:

```
vsPrinter.Action = 3          ' start document
  If vsPrinter.Error Then     ' always check for errors
    Exit Sub
  EndIf
  vsPrinter = "VideoSoft"     ' print some text
  vsPrinter = " 2625 Alcatraz Avenue, Suite 271"
  vsPrinter = " Berkeley, CA 94705"
  vsPrinter = " (510) 547-7295 (phone)"
  vsPrinter = " (510) 547-1084 (fax)"
vsPrinter.Action = 6          ' end document
```

The *NewPage* and *NewCol* actions are used to force page and column breaks. If you use *NewCol* at the last column on a page, the current page is ejected and a fresh one is started.

The *KillDoc* action cancels the current print job. This action should be used when you provide your own custom Abort dialog.

The *Print Page* action prints the current preview page to the current printer. The *Choose Printer & Print Page* action shows a printer selection dialog box, then prints the current preview page to the selected printer. If nothing has been printed to the current preview page, this action does nothing.

The *Copy Page* action copies the current page to the clipboard. If nothing has been printed to the current preview page, this action does nothing.

The *Print All Pages* and *Choose Printer & Print All Pages* actions send all preview pages directly to the printer, so you don't have to call your printing routine twice. This is normally faster than printing the normal way, but the quality of some graphics may degrade when printed directly from the preview pages. Naturally, these actions only work if there are preview pages available.

The **Action** property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each action.

The vsPrinter control has new methods that replace some of the actions described above. See the **StartDoc**, **EndDoc**, **KillDoc**, **NewPage**, **NewColumn**, and **PrintDoc** methods.

Data Type Integer (Enumerated)

AddTable Method

Description This method is used to print tables. **vsPrinter** takes care of all the justification, word wrapping, and text flow. This method offers more flexibility than the **Table** property.

Syntax [*form!*]*vsPrinter.***AddTable** *format$, header$, body$*
[, *headershade&, bodyshade&, append%*]

Remarks The table string is divided into rows and columns by special characters defined through the **TableSep** property. By default, rows are separated by semi-colons (";") and columns by column pipes ("|"), as shown in the example below.

The *format$* string contains formatting information and is not printed. The formatting information consists of sequences of formatting characters (one sequence for each column) followed by a column width specified in Twips. For example, the following string would format a table with four center-aligned, two-inch wide columns:

```
s$ = "^+2880|^+2880|^+2880|^+2880"
```

Valid formatting characters are the following:

<	Align Left	**+**	Center Align (vertically)
>	Align Right	**_**	Bottom Align
^	Align Center	**~**	No Wrapping
=	Justify Text	**!**	Vertical Border

The *header$* string contains information to be printed in the first row of the table and whenever there is a page break while printing the table. If you are appending to an existing table and would like the header to appear at the page breaks, but not in the beginning of the table, set the *append%* parameter to **True**.

The *body$* string contains the main text in the table. By default, rows are separated by semi-colons (";") and columns by column pipes ("|"), but you may choose other characters through the **TableSep** property.

The *headershade&* and *bodyshade&* parameters specify colors to be used for shading the cells in the header and in the body of the table.

The **TableBorder** and **Pen*** properties determine the appearance of lines between the table entries

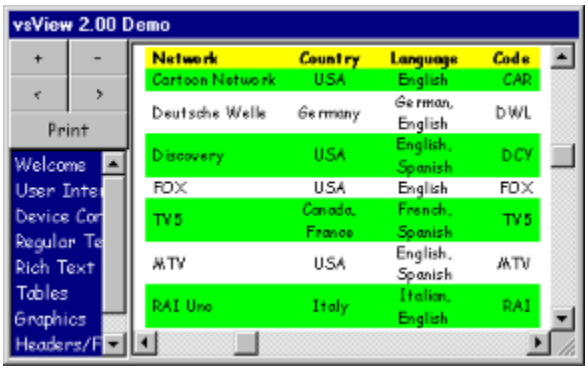
You can customize the appearance of individual table cells by trapping the **NewTableCell** event.

Tables are printed at the current vertical position on the page, specified by the **CurrentY** property. Tables are aligned on the page like paragraphs: left, center or right, depending on the setting of the **TextAlign** property.

The example below shows a table printed with the following code:

```
Dim hclr&, bclr&  ' yellow header, green/white body
hclr = RGB(256, 256, 0)
bclr = RGB(0, 256, 0)

Dim f$  ' format string (widths in Twips)
f = "+1440|+^1000|+^1000|+>700;"

Dim h$  ' header strings (repeat across page breaks)
h = "Network|Country|Language|Code;"

Redim b$(7)  ' data
b(1) = "Cartoon Network|USA|English|CAR;"
```

```
b(2) = "Deutsche Welle|Germany|German, English|DWL;"
b(3) = "Discovery|USA|English, Spanish|DCY;"
b(4) = "FOX|USA|English|FOX;"
b(5) = "TV5|Canada, France|French, Spanish|TV5;"
b(6) = "MTV|USA|English, Spanish|MTV;"
b(7) = "RAI Uno|Italy|Italian, English|RAI;"

' show header and green first row
vp.AddTable f, h, b1, hclr, bclr

' append row pairs (white, green)
Dim i%
For i = 2 To 6 Step 2
    vp.AddTable f, h, b(i), hclr, 0, True
    vp.AddTable f, h, b(i + 1), hclr, bclr, True
Next
```



BrushColor Property

Description Sets or returns the color to be used when filling objects.

Syntax [*form.*]*vsPrinter*.**BrushColor** [= *color&*]

Remarks This property affects the drawing of rectangles, ellipses, and polygons.

For details, see the **Draw** and **Polygon** properties and the **DrawRectangle**, **DrawCircle**, and **DrawEllipse** methods.

Default Value Black

Data Type Long (Color)

BrushStyle Property

Description Sets or returns the style to be used when filling objects.

Syntax [*form.*]*vsPrinter.***BrushStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Solid
1 - Transparent
2 - Horizontal Line
3 - Vertical Line
4 - Upward Diagonal
5 - Downward Diagonal
6 - Cross
7 - Diagonal Cross

This property affects the drawing of rectangles, ellipses, and polygons.

For details, see the **Draw** and **Polygon** properties and the **DrawRectangle**, **DrawCircle**, and **DrawEllipse** methods.

Default Value 0 - Solid

Data Type Integer (Enumerated)

CalcParagraph Property

Description Sets a paragraph string to be measured, taking into account the current font and the page setup.

Syntax [*form.*]*vsPrinter.***CalcParagraph** = *paragraphstring$*

Remarks The width and height of the paragraph are returned in the **TextWidth** and **TextHeight** properties. The bounding box for the paragraph is returned in the **X1**, **Y1**, **X2**, and **Y2** properties.

You can use this property to keep paragraphs together on a page or to draw shaded paragraphs, as the code below shows:

```
Sub ShadedNoBreakParagraph (s$)

  '*** measure the paragraph
  vp.CalcParagraph = s

  '*** if it won't fit, skip a page and measure again
  If vp.Y2 > vp.PageHeight - vp.MarginBottom Then
    vp.Text = Chr(12) '*** page break
    vp.CalcParagraph = s
  End If

  '*** draw shaded box (x1, y1, x2, y2 already set)
  vp.Draw = 2

  '*** draw text
  vp = s
End Sub
```

Data Type String

CalcTable Property

Description Sets a table string to be measured, taking into account the current font and the page setup.

Syntax [*form.*]*vsPrinter*.**CalcTable** = *tablestring$*

Remarks The width and height of the table are returned in the **TextWidth** and **TextHeight** properties. The bounding box for the table is returned in the **X1**, **Y1**, **X2**, and **Y2** properties.

The table string is specially formatted. For details, see the **Table** property or the **AddTable** method.

You can use this property to keep tables together on a page or to draw shaded rows. For an example, see the **CalcParagraph** property.

Data Type String

CalcText Property

Description Sets a text string to be measured, taking into account the current font and the page setup.

Syntax [*form.*]*vsPrinter*.**CalcText** = *textstring$*

Remarks The width and height of the text are returned in the **TextWidth** and **TextHeight** properties. The bounding box for the text is returned in the **X1**, **Y1**, **X2**, and **Y2** properties.

You can use this property to keep text together on a page or to draw shaded text. For an example, see the **CalcParagraph** property.

Data Type String

Collate Property

Description Specifies whether collation should be used when printing multiple copies.

Syntax [*form!*]*vsPrinter.***Collate**[= *setting%*]

Remarks Valid settings for this property are:

0 - COLLATE_FALSE: Do not collate when printing multiple copies.
1 - COLLATE_TRUE: Collate when printing multiple copies.

Using *1 - COLLATE_TRUE* provides faster, more efficient output for collation, since the data is sent to the device driver just once, no matter how many copies are required. The printer is told to simply print the page again.

This property relies on the printer driver, and may or may not be available depending on the capabilities of the printer. After setting any this property, always check the **vsPrinter**'s **Error** property. If it is set to *7 - ER_DEVICEINCAPABLE*, then the printer does not support it and you should take appropriate action.

This property is only available in the 32-bit OCX edition of the control.

Data Type Integer

ColorMode Property

Description Returns or sets a value that determines whether a color printer prints output in color or monochrome. Not available at design time.

Syntax [*form!*]*vsPrinter.***ColorMode**[= *setting%*]

Remarks Valid settings for this property are:

1 - Monochrome: Print output in monochrome
2 - Color: Print output in color.

The default value depends on the printer driver and the current printer settings. Monochrome printers ignore this property.

This property relies on the printer driver, and may or may not be available depending on the capabilities of the printer. After setting any this property, always check the **vsPrinter**'s **Error** property. If it is set to *7 - ER_DEVICEINCAPABLE*, then the printer does not support it and you should take appropriate action.

Data Type Integer (Enumerated)

Columns Property

Description Sets or returns the number of columns to be used when printing text.

Syntax [*form.*]*vsPrinter.***Columns** [= *numcolumns%*]

Remarks The setting of the **Columns** property affects the flow of text as you send it to the printer. The **vsPrinter** control takes care of word wrapping, column, and page feeds automatically for you.

Default Value 1

Data Type Integer

Copies Property

Description Returns or sets a value that determines the number of copies to be printed. Not available at design time.

Syntax [*form!*]*vsPrinter.***Copies**[= *number%*]

Remarks *Number* is a numeric expression that specifies the number of copies to print. This value must be an integer.

Multiple copies may or may not be collated, depending on the printer driver and on the setting of the **Collate** property. Multiple copies of the entire document or multiple copies of each page may be printed. For printers that don't support collating, set **Copies** = 1, and then use a loop in code to print multiple copies of the entire document.

This property relies on the printer driver, and may or may not be available depending on the capabilities of the printer. After setting any this property, always check the **vsPrinter**'s **Error** property. If it is set to *7 - ER_DEVICEINCAPABLE*, then the printer does not support it and you should take appropriate action.

Data Type Integer

CurrentColumn, CurrentLine, CurrentPage Properties

Description Return the current page, column and line being printed by the **vsPrinter** control.

Syntax *page%* = [*form.*]*vsPrinter*.**CurrentPage**

Remarks These properties are read-only.

Data Type Integer

CurrentX, CurrentY Properties

Description Set or return the position of the cursor, where text will be printed on the page. Coordinates are in Twips, measured from the top left of the page.

Syntax [*form.*]*vsPrinter.***CurrentX** [= *newposx&*]
[*form.*]*vsPrinter.***CurrentY** [= *newposy&*]

Remarks As text is printed, these properties change to reflect the new cursor position. You may modify these properties directly if you want to set the cursor at a specific position.

These properties are similar to the standard **CurrentX** and **CurrentY** properties available in VB's **PictureBox** control and **Printer** object.

Data Type Long (Twips)

Device Property

Description Sets or returns the name of the default Windows printer.

Syntax [*form.*]*vsPrinter*.**Device** [= *devname$*]

Remarks You can use this property to provide user feedback – so they know which printer is being used – or to set the default Windows printer.

You cannot change this property while a job is printing. Trying to do so will fire an **Error** event.

You can only set this property to a valid device name. To obtain a list of valid device names, read the **Devices** array property. Trying to set this property to an invalid device name will fire an **Error** event.

Changing this property will affect all Windows applications. Therefore, you may want to save the current printer before changing it and restore it later, as shown in the example below:

```
' ** this example lists all printing devices installed

' save current device and port to restore later
dev$ = vsPrinter.Device
prt$ = vsPrinter.Port

' loop through devices
For i = 0 to vsPrinter.NDevices - 1
  vsPrinter.Device = vsPrinter.Devices(i)

  ' print all ports this device is connected to
  For j = 0 to vsPrinter.NPorts
    Debug.Print vsPrinter.Device; " on "; vsPrinter.Ports(j)
  Next
Next

' restore original settings
vsPrinter.Device = dev$
vsPrinter.Port = prt$
```

Changing this property automatically updates all other printer-related properties (such as **Driver**, **Port**, and **DPI**).

Data Type String

Devices Property

Description Returns a list of the printing devices available.

Syntax *devname$* = [*form*.]*vsPrinter*.**Devices(***devindex%***)**

Remarks To find out how many devices are available, read the **NDevices** property.

For an example of how to use this property, see the description of the **Device** property.

Data Type String Array

DPI Property

Description Returns the resolution of the current printer, expressed as dots per inch.

Syntax *Res%* = [*form.*]*vsPrinter*.**DPI**

Remarks This property is read-only.

Data Type Integer

Draw Property

Description Action-type property for drawing lines, rectangles, and ellipses.

Syntax [*form.*]*vsPrinter*.**Draw** = *setting%*

Remarks Valid settings for this property are:

0 - Nothing
1 - Line
2 - Rectangle
3 - Ellipse

All objects are drawn extending between the (x1, y2) and (x2, y2) points defined by the **X1**, **Y1**, **X2**, and **Y2** properties.

Objects are drawn with the current pen and filled with the current brush. Pen and brush attributes are defined with the **Pen*** and **Brush*** properties.

This property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each object.

See also the **DrawLine**, **DrawRectangle**, **DrawCircle**, and **DrawEllipse** methods.

Data Type Integer (Enumerated)

DrawCircle Method

Description Draws a circle, arc, or pie slice.

Syntax [*form!*]*vsPrinter.***DrawCircle** *x1!, y1!, radius!* [, *startAngle!, endAngle!*]

Remarks This method uses the following parameters:

x1!, y1! Required. Single-precision values indicating the coordinates for the center point of the circle, arc, or pie slice. Units are Twips, measured from the top-left of the page.

radius! Required. Single-precision value indicating the radius of the circle, arc, or pie slice. Units are Twips.

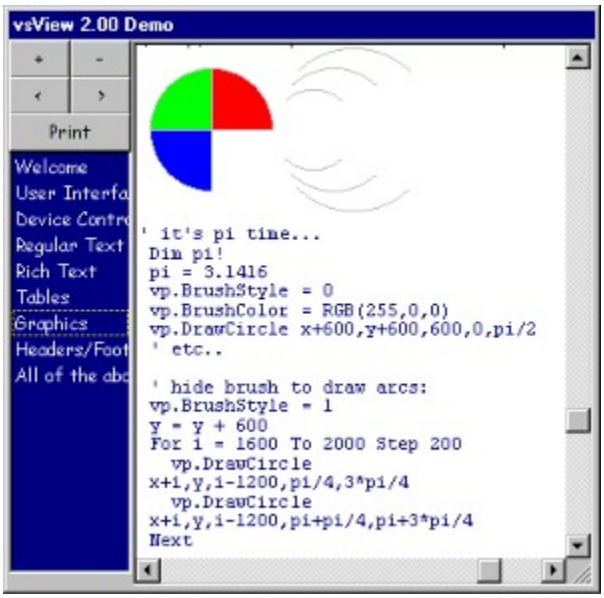
startAngle!, endAngle! Optional. Single-precision values. When an arc or a pie slice is drawn, these values specify the beginning and end positions of the arc, in radians. If these parameters are not supplied, a circle is drawn.

To fill the circle, set the **BrushColor** and **BrushStyle** properties.

The width, color, and style of the line used to draw the circle, arc, or pie slice depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

To create an arc, set the **BrushStyle** property to *1 - Transparent* before calling the **DrawCircle** method. To create a pie slice, set the **BrushStyle** property to any other value. (The pie slice is really just a filled arc.)

For example, the picture below shows a simple pie chart and some arcs as well as the code used to to draw them (**vp** is a **vsPrinter** control):



DrawEllipse Method

Description Draws an ellipse, elliptical arc, or elliptical pie slice.

Syntax [*form!*]*vsPrinter.***DrawEllipse** *x1!, y1!, x2!, y2!* [, *startAngle!, endAngle!*]

Remarks This method uses the following parameters:

x1!, y1!, x2!, y2! Required. Single-precision values indicating the coordinates for the corner points of the rectangle that would enclose the ellipse, elliptical arc, or pie slice. Units are Twips, measured from the top-left of the page.

startAngle!, endAngle! Optional. Single-precision values. When an elliptical arc or a pie slice is drawn, these values specify the beginning and end positions of the arc, in radians. If these parameters are not supplied, an ellipse is drawn.

To fill the circle, set the **BrushColor** and **BrushStyle** properties.

The width, color, and style of the line used to draw the ellipse, elliptical arc, or pie slice depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

To create an arc, set the **BrushStyle** property to *1 - Transparent* before calling the **DrawCircle** method. To create a pie slice, set the **BrushStyle** property to any other value. (The pie slice is really just a filled arc.)

DrawLine Method

Description Draws a line segment.

Syntax [*form!*]*vsPrinter.***DrawLine** *x1!, y1!* [*, x2!, y2!*]

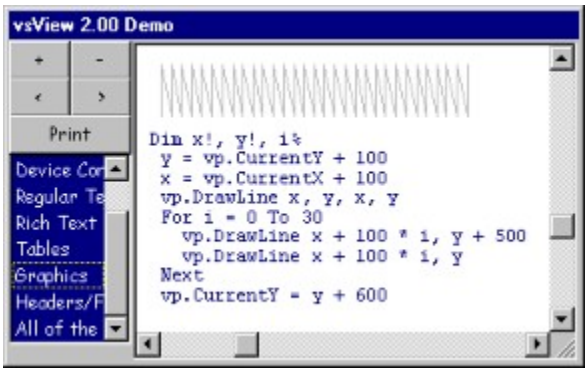
Remarks This method uses the following parameters:

x1!, y1! Required. Single-precision values indicating the coordinates for the start of the line segment, in Twips.

x2!, y2! Optional. Single-precision values indicating the coordinates for the end of the line segment, in Twips. If omitted, the line is drawn from the current point to the specified *x1, y1* coordinates.

The width, color, and style of the line used to draw the rectangle depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

For example, the picture below shows a zig-zag line and the code used to draw it (**vp** is a **vsPrinter** control):



See also the **Polyline** and **Polygon** properties.

DrawRectangle Method

Description Draws a rectangle or a rounded rectangle.

Syntax [*form!*]*vsPrinter.***DrawRectangle** *x1!, y1!, x2!, y2!* [, *radiusX!, radiusY!*]

Remarks This method uses the following parameters:

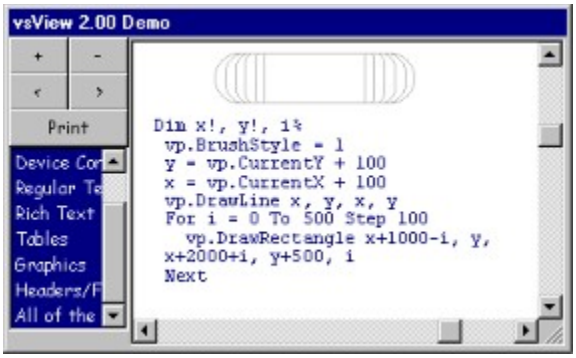
x1!, y1!, x2!, y2! Required. Single-precision values indicating the coordinates for the corner points of the rectangle. Units are Twips, measured from the top-left of the page.

radiusX!, radiusY! Optional. Single-precision values. These values specify the radii to be used for the rectangle's rounded corners, in the horizontal and vertical directions. If they are omitted, a simple rectangle is drawn.

To fill the rectangle, set the **BrushColor** and **BrushStyle** properties.

The width, color, and style of the line used to draw the rectangle depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

For example, the picture below shows several rectangles, progressively more rounded, and the code used to draw it (**vp** is a **vsPrinter** control):



Driver Property

Description Returns the name of the current printer driver.

Syntax *drivername$* = [*form.*]*vsPrinter*.**Driver**

Remarks This property is read-only.

Data Type String

Duplex Property

Description Returns or sets a value that determines whether a page is printed on both sides (if the printer supports this feature). Not available at design time.

Syntax [*form!*]*vsPrinter.***Duplex**[= *setting%*]

Remarks Valid settings for this property are:

1 - Simplex: Single-sided printing with the current orientation setting.
2 - Horizontal: Double-sided printing using a horizontal page turn.
3 - Vertical: Double-sided printing using a vertical page turn.

With horizontal duplex printing, the top of both sides of the page are at the same end of the sheet. With vertical duplex printing, the bottom of one page is at the same end of the sheet as the top of the next page.

This property relies on the printer driver, and may or may not be available depending on the capabilities of the printer. After setting any this property, always check the **vsPrinter**'s **Error** property. If it is set to *7 - ER_DEVICEINCAPABLE*, then the printer does not support it and you should take appropriate action.

Data Type Integer (Enumerated)

EmptyColor Property

Description Sets or returns the color to be used to paint the empty area between the page image and the control borders.

Syntax [*form!*]*vsPrinter.***EmptyColor**[= *color*]

Remarks This property is used only when the **Zoom** property is set to a non-zero value.

See also the **Zoom** property.

Default Value &H8000000C& (Dark Gray)

Data Type Long (Color)

EndDoc Event

Description Fired after a document finishes printing.

Syntax **Sub** *vsPrinter*_**EndDoc ()**

EndPage Event

Description Fired before a page is ejected.

Syntax **Sub** *vsPrinter*_**EndPage ()**

Error Event

Description Fired when an error is detected while printing. The type of error can be determined by reading the **Error** property.

Syntax **Sub** *vsPrinter*_**Error ()**

Error Property

Description Returns a value that indicates what type of printer error has occurred, if any.

Syntax *errcode%* = [*form.*]*vsPrinter*.**Error**

Remarks You should always check this property after starting a new print job and also while printing. The error code is reset whenever a new document is started.

The error codes are:

0 - No Error
1 - Can't Open File
2 - Line Too Long
3 - Can't Access Printer
4 - Can't Start Job
5 - User Aborted
6 - Already Printing
7 - Device Incapable

The *Can't Open File* error occurs when you try to print a file using **Action** = 1 and the file specified by the **FileName** property cannot be opened.

The *Line Too Long* error occurs when a text file is being printed but a line is too long to fit in memory. The line gets truncated.

The *Can't Access Printer* and *Can't Start Job* errors occur when the printer is not available.

The *User Aborted* error occurs when the users clicks the Abort button while a document is printing.

The *Already Printing* error occurs when you try to start a new document while a document is being printed.

The *Device Incapable* error occurs when you specify a printer setting that is not supported by the printer driver.

Default Value 0 - No Error

Data Type Integer

FileName Property

Description Sets or returns the name of the text file being printed with **Action** = 1.

Syntax [*form.*]*vsPrinter*.**FileName** [= *filename$*]

Remarks This property also sets the name of the current job, a string that appears in the default Abort dialog and on the Print Manager list.

Default Value “” (Empty String)

Data Type String

Footer Property

Description Sets or returns the text of the footer printed at the bottom of every page.

Syntax [*form.*]*vsPrinter*.**Footer** [= *footer$*]

Remarks The footer is composed of three sections, separated by pipe characters (“|”). The first section is left-justified, the second is centered, and the third is right-justified.

You may also include a page number field by embedding a “%d” code in the string. Do not use any percent signs in footers except for the page code.

For example, the following footer would print the file name and page number on the left and right corners of every page:

```
vsPrinter.Footer = vsPrinter.FileName + "||Page %d"
```

The footer is printed using the font defined by the **HdrFont*** properties.

Default Value “” (Empty String)

Data Type String

hDC Property

Description This property returns the Windows hDC (handle to device context) being used by the **vsPrinter** control.

Syntax *hdc%* = [*form.*]*vsPrinter.***hDC**

Remarks This property is useful if you wish to call Windows API functions to draw to the printer.

The value of the **hDC** property corresponds to a printer or metafile DC, depending on whether the control is drawing to the printer or to the screen (in print preview mode).

If there's no open document, the **hDC** property returns zero.

You should only use this property if you are familiar with the Windows GDI calls. If you run into problems using this property, our technical support staff will not be able to support you.

Data Type Integer

HdrColor Property

Description Sets or returns the color of the font used to draw the document headers and footers.

Syntax [*form.*]*vsPrinter*.**HdrColor** [= *color&*]

Default Value 0 (Black)

Data Type Long (Color)

HdrFont* Properties

Description These properties are similar to the default **Font*** properties, except they define the font used for printing the headers and footers, while the standard properties define the font used for printing regular text.

Syntax

[*form.*]*vsPrinter.***HdrFontName** [= *fontname$*]
[*form.*]*vsPrinter.***HdrFontSize** [= *size%*]
[*form.*]*vsPrinter.***HdrFontBold** [= {**True**|**False**}]
[*form.*]*vsPrinter.***HdrFontItalic** [= {**True**|**False**}]
[*form.*]*vsPrinter.***HdrFontStrikethru** [= {**True**|**False**}]
[*form.*]*vsPrinter.***HdrFontUnderline** [= {**True**|**False**}]

Header Property

Description Sets or returns the text of the header printed at the top of every page.

Syntax [*form.*]*vsPrinter*.**Header** [= *header$*]

Remarks The header is composed of three sections, separated by pipe characters (“|”). The first section is left-justified, the second is centered, and the third is right-justified.

You may also include a page number field by embedding a “%d” code in the string. Do not use any percent signs in headers except for the page code.

For example, the following header would print the file name and page number on the top left and right corners of every page:

```
vsPrinter.Header = vsPrinter.FileName + "||Page %d"
```

The header is printed using the font defined by the **HdrFont*** properties.

Default Value “” (Empty String)

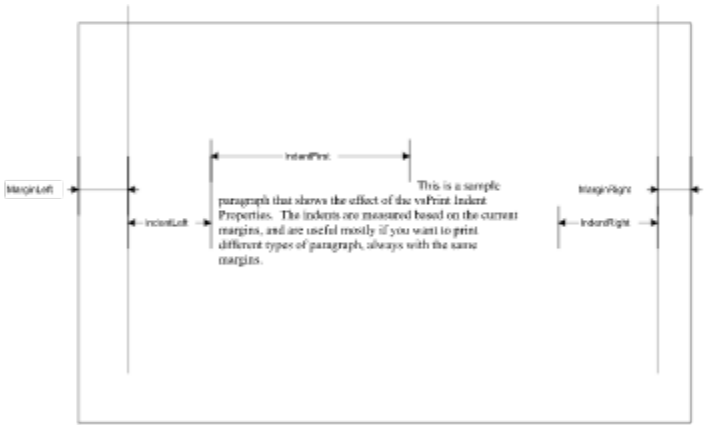
Data Type String

IndentLeft, IndentRight, IndentFirst Properties

Description Set or return the indentation to be used when printing paragraphs, measured in Twips.

Syntax [*form.*]*vsPrinter*.**IndentLeft** [= *indent&*]

Remarks The indents are measured from the current margins, as illustrated below:



Default Value 0

Data Type Long (Twips)

IndentTab Property

Description Sets or returns the size of the tab stops used for printing text, in Twips.

Syntax [*form.*]*vsPrinter*.**IndentTab** [= *indenttab&*]

Remarks Whenever a tab character is detected in the text, the **vsPrinter** control advances the print position to the next multiple of the **IndentTab** value. If the next position is beyond the end of the current line, then the output advances to the next line.

Default Value 770 Twips (1/2”)

Data Type Long (Twips)

LineSpacing Property

Description Sets or returns the amount of space to leave between lines of text.

Syntax [*form.*]*vsPrinter*.**LineSpacing** [= *linespacing%*]

Remarks Line spacing is expressed as a percentage of the current font height.

For example, the following table summarizes common settings for the **LineSpacing** property:

Setting	Effect
100	Single line spacing
150	1.5 line spacing
200	Double line spacing
50	Half line spacing.

Default Value 100 (single line spacing)

Data Type Integer

LoadDoc Method

Description Loads a document saved with the **SaveDoc** method.

Syntax [*form!*]*vsPrinter.***LoadDoc** *filename$*

Remarks The **SaveDoc** and **LoadDoc** methods are useful for loading documents that take a long time to generate and do not change often, or to distribute documents that you create so that others can view and print them.

The *filename$* parameters specifies the name of the file to be loaded. The file must exist and must have been saved with the **SaveDoc** method, or an error will occur.

LoadingDoc, Events

Description These events get fired after each page is saved to or read from disk while the **SaveDoc** and **LoadDoc** methods are executed.

Syntax **Sub** *vsPrinter*_**SavingDoc**(*Page%, Of%, Cancel%*)
Sub *vsPrinter*_**LoadingDoc**(*Page%, Of%, Cancel%*)

Remarks The arguments for these methods are:

Page% The page being saved or loaded.

Of% The total number of pages being saved or loaded.

Cancel% Set this parameter to **True** if you want to cancel the saving/loading operation.

Use these events to provide user feedback and optionally abort lengthy saving/loading operations. See also the **SaveDoc** and **LoadDoc** methods.

MarginBottom, MarginTop, MarginLeft, MarginRight Properties

Description Set or return the distance between the edge of the page and the printed text,in Twips.

Syntax
[*form.*]*vsPrinter*.**MarginBottom** [= *margin&*]
[*form.*]*vsPrinter*.**MarginTop** [= *margin&*]
[*form.*]*vsPrinter*.**MarginLeft** [= *margin&*]
[*form.*]*vsPrinter*.**MarginRight** [= *margin&*]

Default Value 770 Twips (1/2”)

Data Type Long (Twips)

Measure Property

Description Sets a text string to be measured, taking into account the current font and the page setup.

Syntax [*form.*]*vsPrinter*.**Measure** [= *text$*]

Remarks To measure the text, assign it to this property. The width and height of the text are returned in the **TextWid** and **TextHei** properties.

See also the **CalcText**, **CalcParagraph**, and **CalcTable** properties.

Data Type String

MousePage, MouseScroll, MouseZoom Properties

Description These properties determine whether the vsPrinter should automatically handle the mouse to provide scrolling, zooming, and paging of the current preview document.

Syntax [*form!*]*vsPrinter.***MousePage**[= {**True** | **False**}]
[*form!*]*vsPrinter.***MouseScroll**[= {**True** | **False**}]
[*form!*]*vsPrinter.***MouseZoom**[= {**True** | **False**}]

Remarks These properties only work when the **Zoom** property is set to a non-zero value.

If **MouseScroll** is set to **True**, then the user may scroll around the preview page by clicking and dragging the mouse around.

If **MouseZoom** is set to **True**, then the user may zoom in or out by double-clicking the mouse with the left or right mouse buttons.

If **MousePage** is set to **True**, then the user may switch pages back and forth by shift double-clicking the mouse with the left or right mouse buttons.

Data Type Integer (Boolean)

Default Value **True** (for all three properties)

NDevices Property

Description Returns the number of printing devices currently installed.

Syntax *devices%* = [*form.*]*vsPrinter*.**NDevices**

Remarks To access the name of a specific device, use the **Devices**() array property.

For an example of how to use this property, see the description of the **Device** property.

Data Type Integer

NewColumn, NewPage, NewLine Events

Description Fired after each column, page, or line break.

Syntax
Sub *vsPrinter*_**NewColumn** ()
Sub *vsPrinter*_**NewPage** ()
Sub *vsPrinter*_**NewLine** ()

Remarks When the **NewPage** event is fired, the page is still blank. You can trap this event to print custom headers and footers. You can even create watermarks, gray or transparent text and graphics that appear behind the document text.

NewColumn, NewPage Methods

Description These methods are used to force page and column breaks.

Syntax [*form!*]*vsPrinter.***NewPage**
[*form!*]*vsPrinter.***NewColumn**

Remarks Calling these methods is equivalent to setting the **Action** property to 4 and 5, respectively. The method syntax is cleaner and easier to remember, however, so it is recommended that you use the methods instead of the **Action** property.

The **NewColumn** method is useful for multi-column documents. If your document has a single column, its effect is identical to that of the **NewPage** method.

NewTableCell Event

Description Fired before each cell is printed while creating a table.

Syntax **Sub** *vsPrinter*_**NewTableCell (***Row%, Column%, Cell$***)**

Remarks This event allows you to customize the format of individual table cells.

All parameters are read-only. The *Row%* and *Column%* parameters identify the cell about to be printed, and the *Cell$* parameter contains the text that will be printed.

For example, the following code customizes the third column of a table. It prints negative numbers in bold red and positive numbers in bold green:

```
Sub VSPrint_NewTableCell (Row%, Column%, Cell$)
  ' ** we're only customizing column 3:
  If Column = 3 Then
    VSPrinter1.FontBold = True
    If Val (Cell$) < 0 Then
      VSPrinter1.TextColor = RGB (255, 0, 0)
    Else
      VSPrinter1.TextColor = RGB (0, 255, 0)
  ' ** restore regular formatting
  Else
    VSPrinter1.FontBold = False
    VSPrinter1.TextColor = 0
  End If
End Sub
```

NPorts Property

Description Returns the number of ports to which the default Windows printer is connected.

Syntax *nports%* = [*form.*]*vsPrinter*.**NPorts**

Remarks This property will return 1 unless you have identical printers connected to different ports.

For example, if the default Windows printer is an HP LaserJet IV connected to LPT1: and there's another HP LaserJet IV connected to LPT2:, then **NPorts** would return 2, and the **Ports** property array would contain the strings "LPT1:" and "LPT2:".

For an example of how to use this property, see the description of the **Device** property.

Data Type Integer

Orientation Property

Description Sets or returns the current paper orientation setting.

Syntax [*form.*]*vsPrinter.***Orientation** [**=** *orient%*]

Remarks Valid settings for this property are:

0 - Portrait
1 - Landscape

Most popular printers support portrait and landscape orientations, but some do not. Therefore, after changing the paper orientation through code, you may want to check and make sure the change actually took place. You can do this simply by reading back the **Orientation** value, as show below:

```
vsPrinter.Orientation = 1   ' ** try landscape
If vsPrinter.Orientation <> 1 Then  ' ** check if it worked
  MsgBox "Sorry, unable to switch to landscape"
End If
```

Data Type Integer (Enumerated)

PageBorder Property

Description Sets or returns a value that determines the type of border to be drawn around each page.

Syntax [*form.*]*vsPrinter*.**PageBorder** [= *setting%*]

Remarks Valid settings for this property are:

0 - None
1 - Bottom
2 - Top
3 - Top & Bottom
4 - Box
5 - Columns
6 - Columns, Top, & Bottom
7 - All
8 - Top & Columns
9 - Bottom & Columns

The effect of some these settings is illustrated below:

PageBorder = 3

PageBorder = 4

PageBorder = 5

PageBorder = 7

The border is drawn using the pen defined by the **Pen*** properties.

Default Value 3 (Top & Bottom)

Data Type Integer (Enumerated)

PageHeight, PageWidth Properties

Description Return the size of a page to be printed on the current printer, in Twips.

Syntax *pagehei%* = [*form.*]*vsPrinter*.**PageHeight**

Remarks These values are useful when drawing graphics on the page and also when implementing print preview. For example, to preview a printed page at 60% of its actual size, one could use the following code:

```
vsPrinter.Width = vsPrinter.PageWidth * 0.6
vsPrinter.Height = vsPrinter.PageHeight * 0.6
```

The above code would resize the **vsPrinter** control to 60% of the size of a printed page, along with its contents.

Data Type Long (Twips)

PaperBin Property

Description Returns or sets a value indicating the default paper bin on the printer from which paper is fed when printing. Not available at design time.

Syntax [*form!*]*vsPrinter.***PaperBin**[= *setting%*]

Remarks Valid settings for this property are:

1 - Upper: Use paper from the upper bin.
2 - Lower: Use paper from the lower bin.
3 - Middle: Use paper from the middle bin.
4 - Manual: Wait for manual insertion of each sheet of paper.
5 - Envelope: Use envelopes from the envelope feeder.
6 - EnvManual: Use envelopes from feeder, but wait for manual insertion.
7 - Auto: (Default) Use paper from the current default bin.
8 - Tractor: Use paper fed from the tractor feeder.
9 - SmallFmt: Use paper from the small paper feeder.
10 - LargeFmt: Use paper from the large paper bin.
11 - LargeCapacity: Use paper from the large capacity feeder.
12 - Cassette: Use paper from the attached cassette cartridge.

These constants are listed in the Visual Basic (VB) object library in the Object Browser with the “vbPRBN” prefix.

Not all of the bin options are available on every printer. Check the printer documentation for more specific descriptions of these options.

Data Type Integer (Enumerated)

PaperHeight, PaperWidth Properties

Description Sets or returns the physical size of the paper set up for the printing device; not available at design time. If set at run time, this value is used instead of the setting of the **PaperSize** property.

Syntax [*form!*]*vsPrinter.***PaperHeight**[= *value*]
[*form!*]*vsPrinter.***PaperWidth**[= *value*]

Remarks If you set the **PaperHeight** and **PaperWidth** properties for a printer driver that doesn't allow these properties to be set, no error occurs and the size of the paper remains as it was. If you set **PaperHeight** and **PaperWidth** for a printer driver that allows only certain values to be specified, no error occurs and the property is set to whatever the driver allows. For example, you could set Height to 150 and the driver would set it to 144.

Data Type Long (Twips)

PaperSize Property

Description Returns or sets a value indicating the paper size for the current printer. Not available at design time.

Syntax [*form!*]*vsPrinter.***PaperSize**[= *setting%*]

Remarks Valid settings for this property are:

1 - Letter: Letter, 8½ x 11 in.
2 - LetterSmall: Letter Small, 8½ x 11 in.
3 - Tabloid: Tabloid, 11 x 17 in.
4 - Ledger: Ledger, 17 x 11 in.
5 - Legal: Legal, 8½ x 14 in.
6 - Statement: Statement, 5½ x 8½ in.
7 - Executive: Executive, 7½ x 10½ in.
8 - A3: A3, 297 x 420 mm
9 - A4: A4, 210 x 297 mm
10 - A4Small: A4 Small, 210 x 297 mm
11 - A5: A5, 148 x 210 mm
12 - B4: B4, 250 x 354 mm
13 - B5: B5, 182 x 257 mm
14 - Folio: Folio, 8½ x 13 in.
15 - Quarto: Quarto, 215 x 275 mm
16 - 10x14: 10 x 14 in.
17 - 11x17: 11 x 17 in.
18 - Note: Note, 8½ x 11 in.
19 - Env9: Envelope #9, 37/8 x 87/8 in.
20 - Env10: Envelope #10, 41/8 x 9½ in.
21 - Env11: Envelope #11, 4½ x 10 3/8 in.
22 - Env12: Envelope #12, 4½ x 11 in.
23 - Env14: Envelope #14, 5 x 11½ in.
24 - CSheet: C size sheet
25 - DSheet: D size sheet
26 - ESheet: E size sheet
27 - EnvDL: Envelope DL, 110 x 220 mm
28 - EnvC3: Envelope C3, 324 x 458 mm
29 - EnvC4: Envelope C4, 229 x 324 mm
30 - EnvC5: Envelope C5, 162 x 229 mm
31 - EnvC6: Envelope C6, 114 x 162 mm
32 - EnvC65: Envelope C65, 114 x 229 mm
33 - EnvB4: Envelope B4, 250 x 353 mm
34 - EnvB5: Envelope B5, 176 x 250 mm
35 - EnvB6: Envelope B6, 176 x 125 mm
36 - EnvItaly: Envelope, 110 x 230 mm
37 - EnvMonarch: Envelope Monarch, 37/8 x 7½ in.
38 - EnvPersonal: Envelope, 35/8 x 6½ in.
39 - FanfoldUS: U.S. Standard Fanfold, 147/8 x 11 in.
40 - FanfoldStdGerman: German Standard Fanfold, 8½ x 12 in.
41 - FanfoldLglGerman: German Legal Fanfold, 8½ x 13 in.
256 - User: User-defined

These constants are listed in the Visual Basic (VB) object library in the Object Browser with the "vbPRPS" prefix.

Setting the **PaperHeight** or **PaperWidth** properties automatically sets **PaperSize** to *256 - User.*

Data Type Integer (Enumerated)

Paragraph Property

Description Sets a paragraph to be printed. The **vsPrinter** control takes care of justification, word wrapping, and page breaks.

Syntax [*form*.]*vsPrinter*[.**Paragraph**] = *text$*

Remarks After printing a string with this property, the **vsPrinter** control will automatically skip a line and get ready for the next paragraph. If you want to print a paragraph in pieces, use the **Text** property instead.

Data Type String

PenColor Property

Description Sets or returns the color of the pen used to draw all graphics and page borders.

Syntax [*form.*]*vsPrinter*.**PenColor** [= *color&*]

Remarks This property does not affect printed text. To change text color, use the **TextColor** and **HdrColor** properties.

Default Value 0 (Black)

Data Type Long (Color)

PenStyle Property

Description Sets or returns the style of the pen used to draw all graphics and page borders.

Syntax [*form.*]*vsPrinter*.**PenStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Solid
1 - Dash
2 - Dot
3 - Dash-Dot
4 - Dash-Dot-Dot
5 - Transparent
6 - Inside Solid

Note that non-solid styles only work when the **PenWidth** property is set to zero. This is a GDI limitation.

Default Value 0 -Solid

Data Type Integer (Enumerated)

PenWidth Property

Description Sets or returns the width of the pen used to draw all graphics and page borders, in Twips.

Syntax [*form.*]*vsPrinter*.**PenWidth** [= *width%*]

Remarks Setting **PenWidth** to zero causes the thinnest possible pen to be used.

Default Value 0

Data Type Integer (Twips)

PhysicalPage Property

Description Sets or returns a value that determines whether the logical page used by the **vsPrinter** control should correspond to the entire physical page or only to its printable area.

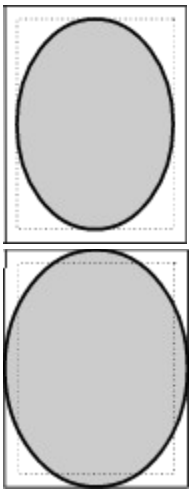
Syntax [*form.*]*vsPrinter.***PhysicalPage** [= {**True | False**}]

Remarks Most printers have a “logical” paper size that corresponds to the printer’s printable area and a “physical” paper size that corresponds to the actual page size. The physical paper size is always a little larger than the logical paper size.

If the **PhysicalPage** property is set to **True**, the **vsPrinter** control draws and prints to the physical page. If the **PhysicalPage** property is set to **False**, the **vsPrinter** control draws and prints to the logical page. This affects the page dimensions returned by the **PageWidth** and **PageHeight** properties, and it also offsets the origin of the **vsPrinter**’s coordinate system.

The pictures below show the effect of the **PhysicalPage** property when the following code is executed. Note how the edges of the oval are not printed when **PhysicalPage** is set to **True**.

```
vsPrinter.Action = 3
vsPrinter.X1 = 0
vsPrinter.Y1 = 0
vsPrinter.X2 = vsPrinter.PageWidth
vsPrinter.Y2 = vsPrinter.PageHeight
vsPrinter.Draw = 3 ' oval
vsPrinter.Action = 6
```



PhysicalPage = False **PhysicalPage = True**

Normally, this property should be set to **False**, to ensure that all output sent to the printer will actually be reproduced on the page, and not truncated if it’s too close to the edges of the page.
In a few special situations – when printing labels or filling pre-printed forms, for example – you may want to set this property to **True**, so you can print at exact positions with respect to the physical page.

Default Value False

Data Type Integer (Boolean)

Picture Property

Description Sets a picture to be drawn on the page.

Syntax [*form.*]*vsPrinter*.**Picture** = *picture*

Remarks The **vsPrinter** control can handle bitmaps, icons, and metafiles in print and preview mode.

The position of the picture is determined by the **X1**, **Y1**, **X2**, and **Y2** properties, and the picture is stretched, if necessary, in order to fill the entire rectangle.

If you need to determine the size of the picture (in order to scale it propertionally, for example), set **X1** = **X2**, then set **Picture** = *picture*. Because the target dimensions are invalid (zero width), the **vsPrinter** control will measure the picture and will set the **X2** and **Y2** properties to the appropriate values, in Twips.

Note that previewing large pictures requires a lot of memory. If you are going to print long documents with lots of large pictures, you may want to provide a “draft preview mode” which displays picture holders instead of actual pictures, as shown in the code below:

```
vp.Action = 3         ' start document
vp.x1 = vp.MarginLeft + 1440 ' 1 inch from left
vp.y1 = vp.MarginTop + 1440  ' 1 inch from top
vp.x2 = vp.x1 + 2880         ' 2 inches wide
vp.y2 = vp.y1 + 2880         ' 2 inches tall
If DraftPreview Then
  vp.Draw = 2         ' picture holder
Else
  vp.Picture = Picture1             ' picture
End If
vp.Action = 6         ' end document
```

Data Type Picture

Polyline Property

Description Sets a value that plots a line composed of many points.

Syntax [*form.*]*vsPrinter*.**Polyline** = *points$*

Remarks The *points$* string contains a sequence of X, Y coordinates, in Twips, separated by spaces. Only the integer part of the coordinates is used. For example, the following code would draw a sine wave one inch tall across the page:

```
s = ""
For i = 0 to 12 * PI Step PI/4
  x = vsPrinter.PageWidth * i / (12 * PI)
  y = vsPrinter.PageHeight / 2 + 770 * Sin(i)
  s = s + Str(Int(x)) + Str(Int(y))
Next
```

You can create complex lines by repeatedly setting the **Draw** property to *Line* and updating **X1**, **Y1**, **X2**, and **Y2**, but the **Polyline** is usually a more convenient and efficient way of obtaining the same results.

The line is drawn with the current pen, defined by the **Pen*** properties.

Data Type String

Polygon Property

Description Sets a value that plots an arbitrary filled polygon. It is similar to the **Polyline** property except that it produces a closed figure.

Syntax [*form*.]*vsPrinter*.**Polygon** = *points$*

Remarks For details on the syntax of the *points$* parameter, see the description of the **Polyline** property.

Data Type String

Port Property

Description Sets or returns the name of the port to which the default Windows printer is connected.

Syntax [*form.*]*vsPrinter*.**Port** [**=** *portname$*]

Remarks You can only set this property to a valid port name. To obtain a list of valid port names, read the **Ports** array property. Trying to set this property to an invalid port name will fire an **Error** event.

Changing this property will affect all Windows applications. Therefore, you may want to save the current port before changing it and restore it later.

Data Type String

Ports Property

Description Returns a list of the ports to which the default Windows printer is connected.

Syntax *portname$* = [*form.*]*vsPrinter*.**Ports(***portindex%***)**

Remarks To find out how many ports are connected to the default Windows printer, read the **NPorts** property.

For an example of how to use this property, see the description of the **Device** property.

Data Type String Array

Preview Property

Description Sets or returns a value that determines whether **vsPrint** control output should be sent to the printer or to the screen.

Syntax [*form.*]*vsPrinter.***Preview** [= {**True**|**False**}]

Remarks If the **Preview** property is set to **False**, the **vsPrinter** control sends its output to the printer. Unless the **AbortWindow** property is set to **False**, a default Abort dialog is created and shown to allow the user to abort the print job.

If the **Preview** property is set to **True**, the **vsPrinter** control saves all output in memory and allows you to display it, one page at a time, on the **vsPrinter** control itself. You can then switch pages by changing the **PreviewPage** property and scale the output by changing the **Zoom** property.

This allows you to use the same code to create printed output and to perform print previews. The code below illustrates how it is done:

```
' This routine sends output to the printer
Sub PrintButton_Click ()
  vsPrinter.Preview = False
  DoPrinting
End Sub

' This routines sends output to the vsPrinter control
Sub Preview_Click ()
  vsPrinter.Preview = True
  vsPrinter.PreviewPage = 1
  DoPrinting
End Sub

' This routine switches preview page
Sub PreviewPage_Click (Index as Integer)
  Select Case Index
    Case 0 ' first page
      vsPrinter.PreviewPage = 1
    Case 3 ' lastpage
      vsPrinter.PreviewPage = vsPrinter.CurrentPage
    Case 1 ' next page
      vsPrinter.PreviewPage = vsPrinter.PreviewPage + 1
    Case 2 ' previous page
      vsPrinter.PreviewPage = vsPrinter.PreviewPage - 1
  End Select
End Sub
```

In preview mode, you can change the zoom factor by setting the **Zoom** property to a non-zero value (if you set the **Zoom** property to azero, the **vsPrinter** control stretches the page image to fill the control).

The maximum number of pages in a preview document is 1000.

Default Value False

Data Type Integer (Boolean)

PreviewMode Property

Description Sets or returns a value that determines whether the preview image should match the printer colors, the screen colors, or force it to monochrome.

Syntax [*form.*]*vsPrinter.***PreviewMode** [= *mode%*]

Remarks Valid settings for this property are:

0 - Screen Compatible (usually color)
1 - Printer Compatible (usually black and white)
2 - Monochrome (always black and white)

This property is useful when you are going to print documents that consist mostly of black and white text. In this case, you can save a lot of memory and speed up program execution by setting the **PreviewMode** property to *2 - Monochrome.*

Default Value 0 - Screen Compatible

Data Type Integer (Enumerated)

PreviewPage Property

Description Sets or returns the number of the page being previewed.

Syntax [*form.*]*vsPrinter*.**PreviewPage** [= *page%*]

Remarks Valid page numbers start at one and go up to the last page printed. The number of the last page can be determined by reading the **CurrentPage** property.

Default Value 1

Data Type Integer

PrintDoc Method

Description This method prints the preview document on the printer.

Syntax [*form!*]*vsPrinter.***PrintDoc** [*choose%, startpage%*, *endpage%*]

Remarks Calling this method is equivalent to setting the **Action** property to 11 or 12, except that the method allows you to specify a print range. The method syntax is cleaner and easier to remember, however, so it is recommended that you use the methods instead of the **Action** property.

The optional *choose%* parameter determines whether a print selection dialog shold be displayed to the user before printing. The default value for this parameter is **False**.

The optional *startpage%* and *endpage%* parameters determine the range of pages to be printed. The default values for these parameters are 1 and **CurrentPage** (which corresponds to the entire preview document).

There are two ways to create printed output with the **vsPrinter** object: you may set the **Preview** property to **True**, generate the entire document, and then print it with this method, or you may set the **Preview** property to **False** and generate the document directly on the printer. In general, the second option is slower but yields better quality when your document includes graphics.

PrintQuality Property

Description Returns or sets a value indicating the printer resolution. Not available at design time.

Syntax [*form!*]*vsPrinter.***PrintQuality**[= *setting%*]

Remarks Valid settings for this property are:

1 - Draft: Draft resolution
2 - Low: Low resolution
3 - Medium: Medium resolution
4 - High: High resolution

In addition to the predefined negative values, you can also set value to a positive dots per inch (dpi) value, such as 300.

These constants are listed in the Visual Basic (VB) object library in the Object Browser with the “vbPRPQ” prefix.

The default value depends on the printer driver and the current settings of the printer. The effect of these settings varies among printers and printer drivers. On some printers, some or all of the settings may produce the same result.

This property relies on the printer driver, and may or may not be available depending on the capabilities of the printer. After setting any this property, always check the **vsPrinter**’s **Error** property. If it is set to *7 - ER_DEVICEINCAPABLE*, then the printer does not support it and you should take appropriate action.

Data Type Integer (Enumerated)

ResetDC Event

Description Fired after a page is ejected and before the next page is started.

Syntax **Sub** *vsPrinter*_**ResetDC ()**

Remarks This event is provided for a few users who need to call the Windows API function ResetDC. For details on the ResetDC function and its uses, see the Windows SDK documentation.

This event is fired between pages, so you cannot print anything while responding to it. Instead, add printing code to the **NewPage** and **EndPage** events.

SaveDoc Method

Description Saves the entire preview document to a disk file, optionally compressing it.

Syntax [*form!*]*vsPrinter.***SaveDoc** *filename$* [*, compress%, firstpage%, lastpage%*]

Remarks The **SaveDoc** and **LoadDoc** methods are useful for loading documents that take a long time to generate and do not change often, or to distribute documents that you create so that others can view and print them.

The *filename$* parameter specifies the name of the file that should be saved. There is no default extension, so you must supply your own. If the file already exists, it is deleted before the saving process starts.

The optional *compress%* parameter specifies whether the file should be compressed as it is saved. Compressing the file typically reduces its size to about 25% of the uncompressed size, but saving takes about twice as long. The default value for this parameter is **True**.

The optional *firstpage%* and *lastpage%* parameters allow you to select a range of pages to save. The default values for these parameters are from 1 to **CurrentPage**, which saves the entire document.

After each page gets saved, the **vsPrinter** control fires the **SavingDoc** event, so you can cancel the process or provide user feedback while saving long documents.

Scale Property

Description Specifies the factor by which the printed output is to be scaled.

Syntax [*form!*]*vsPrinter.***Scale**[= *value%*]

Remarks The apparent page size is scaled from the physical page size by a factor of **Scale**/100. For example, a letter-sized page with a **Scale** value of 50 would contain as much data as a page of 17- by 22-inches because the output text and graphics would be half their original height and width.

Note that this property does not affect the measurements or coordinate system used to produce images on a page. It only works at the printer driver level.

This property relies on the printer driver, and may or may not be available depending on the capabilities of the printer. After setting any this property, always check the **vsPrinter**'s **Error** property. If it is set to *7 - ER_DEVICEINCAPABLE*, then the printer does not support it and you should take appropriate action.

Data Type Integer

SpaceAfter SpaceBefore Properties

Description Set or return the amount of vertical spacing before and after each paragraph, in Twips.

Syntax [*form.*]*vsPrinter*.**SpaceAfter** [= *space&*]
[*form.*]*vsPrinter*.**SpaceBefore** [= *space&*]

Remarks The specified vertical spacing is added before and after each paragraph printed with the **Paragraph** property. The spacing is also added before and after the contents of each table cell.

Default Value 0

Data Type Long (Twips)

StartDoc Event

Description Fired before a document starts printing.

Syntax **Sub** *vsPrinter*_**StartDoc ()**

StartDoc, EndDoc, KillDoc Methods

Description These methods control printer jobs. All output sent to the vsPrinter control must be enclosed between **StartDoc**/**EndDoc** or **StartDoc**/**KillDoc** calls.

Syntax
[*form!*]*vsPrinter.***StartDoc**
[*form!*]*vsPrinter.***EndDoc**
[*form!*]*vsPrinter.***KillDoc**

Remarks Calling these methods is equivalent to setting the **Action** property to 3, 6, and 7, respectively. The method syntax is cleaner and easier to remember, however, so it is recommended that you use the methods instead of the **Action** property.

StartDoc initiates a print job. If there is a job already in progress, vsPrinter fires the internal error *6 - Already Printing.*

EndDoc terminates a print operation sent to the Printer object, releasing the document to the print device or to the preview window.

KillDoc immediately terminates the current print job. If the operating system's Print Manager is handling the print job (the Print Manager is running and has background printing enabled), **KillDoc** deletes the current print job and the printer receives nothing. If Print Manager isn't handling the print job (background printing isn't enabled), some or all of the data may be sent to the printer before **KillDoc** can take effect. In this case, the printer driver resets the printer when possible and terminates the print job.

Table Property

Description Sets a table to be printed. The **vsPrinter** control takes care of all the justification, word wrapping, and page breaks.

Syntax [*form.*]*vsPrinter.***Table** = *table$*

Remarks The table string is divided into rows and columns by special characters defined through the **TableSep** property. By default, rows are separated by semi-colons (";") and columns by column pipes ("|"), as shown in the example below.

The first row contains formatting information only and is not printed. The formatting information consists of sequences of formatting characters (one sequence for each column) followed by a column width specified in Twips. For example, the following string would format a table with four center-aligned, two-inch wide columns:

```
s$ = "^+2880|^+2880|^+2880|^+2880"
```

Valid formatting characters are the following:

<	Align Left	**+**	Center Align (vertically)
>	Align Right	**_**	Bottom Align
^	Align Center	**~**	No Wrapping
=	Justify Text	**!**	Vertical Border

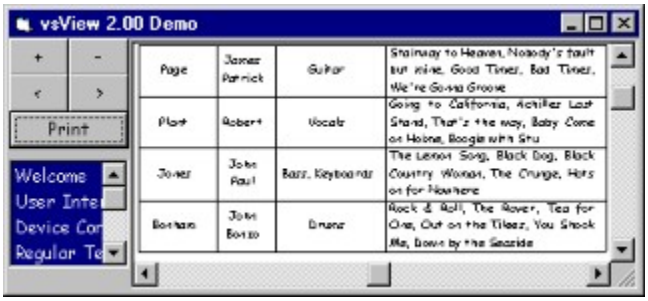
The **TableBorder** and **Pen*** properties determine the appearance of lines between the table entries

You can customize the appearance of individual table cells by trapping the **NewTableCell** event.

Tables are printed at the current vertical position on the page, specified by the **CurrentY** property. Tables are aligned on the page like paragraphs: left, center or right, depending on the setting of the **TextAlign** property.

The example below shows a table printed with the following code:

```
f$ = "+^1440|+^1050|+^2000|+=4000;"
s$ = "Page|James Patrick|Guitar|Stairway to Heaven,..."
s$ = s$ & "Plant|Robert|Vocals|Going to California,..."
s$ = s$ & "Jones|John Paul|Bass, Keyboards|The Lemon..."
s$ = s$ & "Bonham|John Bonzo|Drums|Rock & Roll, The..."
vp.Table = f & s
```



See also the new **AddTable** method, which offers additional functionality.

Data Type String

TableBorder Property

Description Sets or returns the type of border to be used when drawing tables.

Syntax [*form.*]*vsPrinter*.**TableBorder** [= *setting%*]

Remarks Valid settings for this property are:

0 - None
1 - Bottom
2 - Top
3 - Top & Bottom
4 - Box
5 - Columns
6 - Columns, Top, & Bottom
7 - All
8 - Box Rows
9 - Box Columns

This property is analogous to the **PageBorder** property, except it applies to tables instead of pages.

Default Value 7 - All

Data Type Integer (Enumerated)

TableSep Property

Description This property is used to specify the characters to be used as table row and column separators. It is used in conjunction with the **Table** property and **AddTable** method.

Syntax [*form*.]*vsPrinter*.**TableSep** = *tablesep$*

Remarks TheTableSep property must hold a two-character string. The first character is used as the column separator and the second as the row separator. The characters must be different.

If the string is less than two characters long, or if the characters are equal, the default table separator string is used (“|;”).

Default Value “|;”

Data Type String

Text Property

Description Sets text to be printed. The **vsPrinter** control takes care of all the justification, word wrapping, and page breaks.

Syntax [*form.*]*vsPrinter.***Text** = *text$*

Remarks This property is used to print pieces of paragraphs. This is useful if you want to change fonts or colors halfway through a left-aligned paragraph. For example, the code below prints VB files and uses bold red type for function names:

```
Sub PrintVBFile (FileName as String)
  Dim ln$, subname$, subargs$
  Open FileName For Input As #1
  vsPrinter.StartDoc
  While Not EOF (1)
    Line Input #1, ln
    If Left(ln, 3) = "Sub" Then ' print function header
      ln = Mid(ln, 4)
      subname = Left(ln, Instr (ln, " "))
      subargs = Mid(ln, Instr (ln, " "))
      vsPrinter.Text = "Sub "
      vsPrinter.FontBold = True
      vsPrinter.TextColor = RGB (255, 0, 0)
      vsPrinter.Text = subname
      vsPrinter.FontBold = False
      vsPrinter.TextColor = 0
      vsPrinter.Text = subargs
    Else ' print regular paragraph
      vsPrinter = ln
    End If
  Wend
  vsPrinter.EndDoc
End Sub
```

If you want to print an entire paragraph, use the **Paragraph** property instead.

Note that the text string may have embedded line breaks. For example, the code

```
vsPrinter.text = "1: Hello" & Chr$(13) & "World (2 lines)"
vsPrinter.text = "2: Hello "
vsPrinter.text = "World (1 line)"
```

would print

```
1: Hello
World (2 lines)
2: Hello World (1 line)
```

Data Type String

TextAlign Property

Description Sets or returns a value that determines the alignment of printed paragraphs and tables.

Syntax [*form.*]*vsPrinter*.**TextAlign** = *setting%*

Remarks Valid settings for this property are:

0 - Left Top
1 - Center Top
2 - Right Top
3 - Left Bottom
4 - Center Bottom
5 - Right Bottom
6 - Left Baseline
7 - Center Baseline
8 - Right Baseline
9 - Justified Top
10 - Justified Bottom
11 - Justified Baseline

Embedded tab characters (**Chr**(9)) only expand properly when text is left-aligned (settings 0, 3, or 6).

Default Value 0 - Left

Data Type Integer (Enumerated)

TextAngle Property

Description Sets or returns the angle, in tenths of degrees, between the base line of a character and the horizontal.

Syntax [*form*.]*vsPrint*.**TextAngle** [= *angle%*]

Remarks The angle is measured in the counterclockwise direction.

Default Value 0

Data Type Integer

TextColor Property

Description Sets or returns the color used to print regular text.

Syntax [*form*.]*vsPrinter*.**TextColor** [= *color&*]

Remarks Use the **HdrColor** to define the color used to print header and footer text.

Default Value 0 (Black)

Data Type Long (Color)

TextHei, TextWid Properties

Description Return text measurements, in Twips. These properties work in conjunction with the **Measure**, **CalcText**, **CalcParagraph**, and **CalcTable** properties.

Syntax *hei&* = [*form.*]*vsPrinter*.**TextHei**
wid& = [*form.*]*vsPrinter*.**TextWid**

Remarks To measure text, start by selecting the appropriate font through the **Font*** properties. Then, assign the text to be measured to the **Measure**, **CalcText**, **CalcParagraph**, or **CalcTable** properties. Finally, read the text width and height, in Twips, from the **TextHei** and **TextWid** properties.

These properties are useful for doing things such as drawing boxes around strings or placing text at specific positions on the page. For example, the code below draws a string at the current cursor position with a box around it:

```
Sub BoxString (s As String)
  vp.Measure = s
  vp.X1 = vp.CurrentX
  vp.Y1 = vp.CurrentY
  vp.X2 = vp.CurrentX + vp.TextWid
  vp.Y2 = vp.CurrentY + vp.TextHei
  vp.Text = s                ' draw text
  vp.Action = 2              ' draw box around it
End Sub
```

Data Type Long (Twips)

TextHeight, TextWidth Properties

Description Returns the height or width of a text string as it would be printed in the current font. Units are Twips.

Syntax [*form!*]*vsPrinter.***TextHeight**
[*form!*]*vsPrinter.***TextWidth**

Remarks The value returned does not take into account any text wrapping. For that, see the **CalcText**, **CalcPara**, and **CalcTable** properties.

Data Type Long

TextRTF Property

Description This property is used to print RTF (rich text format) text.

Syntax [*form!*]*vsPrinter.***TextRTF** = *string*

Remarks RTF text may contain special formatting codes, allowing you to compose and print complex text. You may obtain RTF text from RTF files (such as those saved by Microsoft Word) or simply copy strings from the **RichEdit** control.

If you know the RTF codes you need want, you may create RTF text yourself. The most complicated part of the RTF string is usually the header. If you don't feel like creating one, don't: **vsPrinter** will realize it's missing and will create a header for you on-the-fly. For example, the following line of code would create a paragraph with some embedded bold and italic characters:

```
s$ = "This is bold: {\b Hello} "
s$ = s$ & "and this is italics: {\i Hello}"
vp.TextRTF = s$
```

To find out more about RTF and RTF codes, check out the RTF topic in VB's on-line help.

This property is only available in the 32-bit OCX edition of the control.

Data Type String

TrueType Property

Description Specifies how TrueType fonts should be printed.

Syntax [*form!*]*vsPrinter.***TrueType**[= *setting%*]

Remarks Valid settings for this property are:

1 - BITMAP: Prints TrueType fonts as graphics.
2 - DOWNLOAD: Downloads TrueType fonts as soft fonts.
3 - SUBDEV: Substitute device fonts for TrueType fonts.

The default value for this property depends on the type of printer you have. Generally, dot-matrix printers will use the *BITMAP*, Hewlett-Packard printers will use *DOWNLOAD*, and PostScript printers will use *SUBDEV* by default.

This property relies on the printer driver, and may or may not be available depending on the capabilities of the printer. After setting any this property, always check the **vsPrinter**'s **Error** property. If it is set to *7 - ER_DEVICEINCAPABLE*, then the printer does not support it and you should take appropriate action.

Data Type Integer (Enumerated)

TwipsPerPixelX, TwipsPerPixelY Properties

Description Return the number of twips per pixel for an object measured horizontally (**TwipsPerPixelX**) or vertically (**TwipsPerPixelY**).

Syntax [*form!*]*vsPrinter.***TwipsPerPixelX**
[*form!*]*vsPrinter.***TwipsPerPixelY**

Remarks Windows API routines generally require measurements in pixels. You can use these properties to convert measurements quickly.

Data Type Long

Version Property

Description Returns the version of the **vsPrinter** control currently loaded in memory.

Syntax *CheckVer%* = [*form*.]*vsPrint*.**Version**

Remarks You may want to check this value at the **Form_Load** event, to make sure the version that is executing is at least as current as the version used to develop your application.

The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.

Data Type Integer

X1, X2, Y1, Y2 Properties

Description Set or return the boundaries of a rectangle used to define the size of graphics drawn with the **Draw** and **Picture** properties, or measurements made by setting the **CalcParagraph**, **CalcText**, and **CalcTable** properties.

Syntax

[*form.*]*vsPrinter.***X1** [= *value&*]
[*form.*]*vsPrinter.***Y1** [= *value&*]
[*form.*]*vsPrinter.***X2** [= *value&*]
[*form.*]*vsPrinter.***X2** [= *value&*]

Remarks For an example of how these properties are used, see the description of the **Picture** property.

Data Type Long (Twips)

Zoom Property

Description Sets or returns the size of a preview page as compared to a printed page, in percentage.

Syntax [*form!*]*vsPrinter*.**Zoom** [= *value%*]

Remarks If you set this property to 100%, then the preview page appears in actual size. If you set it to 50%, the page appears in half its actual size, and so on.

If the control is smaller than the page image, scroll bars are automatically added. If the control is larger than the page image, the image is centered and the empty area around it is filled with the color specified by the **EmptyColor** property.

You can add automatic mouse support for scrolling the page, zooming in or out, and even switching pages simply by setting the **MouseScroll**, **MouseZoom**, and **MousePage** properties.

The picture below illustrates the effect of changing the **Zoom** property:

Zoom = 150% **Zoom** = 100%

Zoom = 50% **Zoom** = 25%

If you set the **Zoom** property to zero, then the page image is stretched to fill the entire control. In this case, you

should normally set the control size to match the aspect ratio of a page. You can do this by reading the **PageWidth** and **PageHeight** properties.

Data Type Integer

Default Value 0

vsDraw Reference

Description The VideoSoft **vsDraw** control lets you create detailed, scaleable, resolution-independent drawings. The drawings can be shown on your forms, printed, or copied to the Windows clipboard, where they become available for pasting into other Windows applications such as Microsoft Word.

After the drawing is created, you can change the drawing extents to provide distortion and zoom.

Use the **vsDraw** control to create maps, charts, diagrams, or whatever graphics you need.

File Name VSVIEW.VBX for the VBX version,
VSVIEW32.OCX for the 32-bit OCX version, or
VSVIEW16.OCX for the 16-bit OCX version

Object Type vsDraw

Note Before you can use a **vsDraw** control in your application, you must add VSVIEW to your project (see the Visual Basic manual for details). To automatically include VSVIEW in new projects, put it in an AUTOLOAD file. When distributing your application, you should install the vsView files in the user's Microsoft Windows SYSTEM subdirectory.

Remarks When you send drawing commands to the **vsDraw** control, they are stored in a list, but nothing is actually drawn on the screen or printer until you use the **Action** property or the **Show** method to render the drawing. This is done this way to improve efficiency.

The **vsDraw** control is based on Windows metafile technology, which has limited support for text manipulation. This makes it less than ideal for text-intensive applications. If you need to handle a lot of text, use the **vsPrinter** control instead.

vsDraw Summary

Properties (default: Action)

(About)	* Action	BackColor
* BackStyle	BorderStyle	* BrushColor
* BrushStyle	DragIcon	DragMode
* Draw	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrike	FontUnder	Height
HelpContextID	Index	Left
MousePointer	Name	Parent
* PenColor	* PenStyle	* PenWidth
* Picture	* Polygon	* Polyline
* ScaleHeight	* ScaleLeft	* ScaleTop
* ScaleWidth	Tag	* TextAlign
* TextAngle	* TextColor	* TextOut
Top	Visible	* Version
Width	* X1	* X2
* Y1	* Y2	

Events

Click	DblClick	DragDrop
DragOver	MouseDown	MouseMove
MouseUp		

Methods

* Clear	* DrawCircle	* DrawEllipse
* DrawLine	* DrawRectangle	Move
Refresh	SetFocus	* Show
ZOrder		

Action Property

Description Method-style property that allows you to specify actions to be taken by the **vsDraw** control.

Syntax [*form*.]*vsDraw*.**Action** = *action%*

Remarks Valid settings for this property are:

0 - None
1 - Clear
2 - Draw
3 - Print
4 - Choose Printer & Print
5 - Copy

The *Clear* action erases everything in the **vsDraw** control.

The *Draw* action renders the current drawing on the control. Note that you can create an entire drawing, but nothing will appear on the screen until you set the **Action** property to 2 or use the **Show** method. This is done to improve rendering speed.

The *Print* action renders the drawing on the printer. The drawing is automatically scaled to fill as much of the page as possible, while preserving the aspect ratio of the drawing on the screen.

The *Choose Printer & Print* action is similar to the *Print* action, except it displays a dialog box that allows the user to choose the printer to be used.

The *Copy* action copies the current drawing to the Windows clipboard, where it becomes available to be pasted into other applications.

This property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each action.

Data Type Integer (Enumerated)

BackStyle Property

Description Sets or returns a value that determines whether text drawn on the **vsDraw** control is transparent or opaque.

Syntax [*form*.]*vsDraw*.**BackStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Transparent
1 - Opaque

Default Value 0 - Transparent

Data Type Integer (Enumerated)

BrushColor Property

Description Sets or returns the color to be used when filling objects.

Syntax [*form.*]*vsDraw*.**BrushColor** [= *color&*]

Remarks This property affects the drawing of rectangles, ellipses, and polygons.

For details, see the **Draw** and **Polygon** properties and the **DrawRectangle**, **DrawCircle**, and **DrawEllipse** methods.

Default Value Black

Data Type Long (Color)

BrushStyle Property

Description Sets or returns the style to be used when filling objects.

Syntax [*form.*]*vsPrinter.***BrushStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Solid
1 - Transparent
2 - Horizontal Line
3 - Vertical Line
4 - Upward Diagonal
5 - Downward Diagonal
6 - Cross
7 - Diagonal Cross

This property affects the drawing of rectangles, ellipses, and polygons.

For details, see the **Draw** and **Polygon** properties and the **DrawRectangle**, **DrawCircle**, and **DrawEllipse** methods.

Default Value 0 - Solid

Data Type Integer (Enumerated)

Clear Method

Description Erases everything in the **vsDraw** control.

Syntax [*form!*]*vsDraw*.**Clear**

Remarks Calling this method is equivalent to setting the **Action** property to 1. The method syntax is cleaner and easier to remember, however, so it is recommended that you use the method instead of the **Action** property.

Draw Property

Description Action-type property for drawing lines, rectangles, and ellipses.

Syntax [*form.*]*vsDraw*.**Draw** = *setting%*

Remarks Valid settings for this property are:

0 - Nothing
1 - Line
2 - Rectangle
3 - Ellipse

All objects are drawn extending between the (x1, y2) and (x2, y2) points defined by the **X1**, **Y1**, **X2**, and **Y2** properties.

Objects are drawn with the current pen and filled with the current brush. Pen and brush attributes are defined with the **Pen*** and **Brush*** properties.

This property only acts at run-time, but the above list is also shown at design-time, on the properties window. This makes it easy for you to remember the code for each object.

See also the **DrawLine**, **DrawRectangle**, **DrawCircle**, and **DrawEllipse** methods.

Data Type Integer (Enumerated)

DrawCircle Method

Description Draws a circle, arc, or pie slice.

Syntax [*form!*]*vsDraw.***DrawCircle** *x1!, y1!, radius!* [, *startAngle!, endAngle!*]

Remarks This method uses the following parameters:

x1!, y1! Required. Single-precision values indicating the coordinates for the center point of the circle, arc, or pie slice.

radius! Required. Single-precision value indicating the radius of the circle, arc, or pie slice.

startAngle!, endAngle! Optional. Single-precision values. When an arc or a pie slice is drawn, these values specify the beginning and end positions of the arc, in radians. If these parameters are not supplied, a circle is drawn.

To fill the circle, set the **BrushColor** and **BrushStyle** properties.

The width, color, and style of the line used to draw the circle, arc, or pie slice depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

DrawEllipse Method

Description Draws an ellipse, elliptical arc, or elliptical pie slice.

Syntax [*form!*]*vsDraw.***DrawEllipse** *x1!, y1!, x2!, y2!* [, *startAngle!, endAngle!*]

Remarks This method uses the following parameters:

x1!, y1!, x2!, y2! Required. Single-precision values indicating the coordinates for the corner points of the rectangle that would enclose the ellipse, elliptical arc, or pie slice.

startAngle!, endAngle! Optional. Single-precision values. When an elliptical arc or a pie slice is drawn, these values specify the beginning and end positions of the arc, in radians. If these parameters are not supplied, an ellipse is drawn.

To fill the circle, set the **BrushColor** and **BrushStyle** properties.

The width, color, and style of the line used to draw the ellipse, elliptical arc, or pie slice depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

DrawLine Method

Description Draws a line segment.

Syntax [*form!*]*vsPrinter.***DrawLine** *x1!, y1!* [*, x2!, y2!*]

Remarks This method uses the following parameters:

x1!, y1! Required. Single-precision values indicating the coordinates for the start of the line segment, in Twips.

x2!, y2! Optional. Single-precision values indicating the coordinates for the end of the line segment. If omitted, the line is drawn from the current point to the specified *x1, y1* coordinates.

The width, color, and style of the line used to draw the rectangle depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

See also the **Polyline** and **Polygon** properties.

DrawRectangle Method

Description Draws a rectangle or a rounded rectangle.

Syntax [*form!*]*vsDraw.***DrawRectangle** *x1!, y1!, x2!, y2!* [, *radiusX!, radiusY!*]

Remarks This method uses the following parameters:

x1!, y1!, x2!, y2! Required. Single-precision values indicating the coordinates for the corner points of the rectangle.

radiusX!, radiusY! Optional. Single-precision values. These values specify the radii to be used for the rectangle's rounded corners, in the horizontal and vertical directions. If they are omitted, a simple rectangle is drawn.

To fill the rectangle, set the **BrushColor** and **BrushStyle** properties.

The width, color, and style of the line used to draw the rectangle depend on the setting of the **PenWidth**, **PenColor**, and **PenStyle** properties.

PenColor Property

Description Sets or returns the color of the pen used to draw all graphics.

Syntax [*form.*]*vsDraw*.**PenColor** [= *color&*]

Remarks This property does not affect text. To change text color, use the **TextColor** property.

Default Value 0 (Black)

Data Type Long (Color)

PenStyle Property

Description Sets or returns the style of the pen used to draw all graphics.

Syntax [*form.*]*vsDraw*.**PenStyle** [= *setting%*]

Remarks Valid settings for this property are:

0 - Solid
1 - Dash
2 - Dot
3 - Dash-Dot
4 - Dash-Dot-Dot
5 - Transparent
6 - Inside Solid

Note that non-solid styles only work when the **PenWidth** property is set to zero. This is a GDI limitation.

Default Value 0 -Solid

Data Type Integer (Enumerated)

PenWidth Property

Description Sets or returns the width of the pen used to draw all graphics.

Syntax [*form.*]*vsDraw*.**PenWidth** [= *width%*]

Remarks Setting **PenWidth** to zero causes the thinnest possible pen to be used.

Default Value 0

Data Type Integer

Picture Property

Description Returns a picture with the contents of the **vsDraw** control.

Syntax [*form.*]*vsPrint*.**Picture** = [*form.*]*vsDraw*.**Picture**

Remarks This property is mainly useful when you want to copy the contents of a **vsDraw** control directly into a **vsPrinter** control or to the clipboard.

Data Type Picture (Metafile)

Polyline Property

Description Sets a value that plots a line composed of many points.

Syntax [*form.*]*vsDraw*.**Polyline** = *points$*

Remarks The *points$* string contains a sequence of X, Y coordinates separated by spaces. Only the integer part of the coordinates is used.

You can create complex lines by repeatedly setting the **Draw** property to *Line* and updating **X1**, **Y1**, **X2**, and **Y2**, but the **Polyline** is usually a more convenient and efficient way of obtaining the same results.

The line is drawn with the current pen, defined by the **Pen*** properties.

Data Type String

Polygon Property

Description Sets a value that plots an arbitrary filled polygon. It is similar to the **Polyline** property except that it produces a closed figure.

Syntax [*form*.]*vsDraw*.**Polygon** = *points$*

Remarks For details on the syntax of the *points$* parameter, see the description of the **Polyline** property.

Data Type String

ScaleHeight, ScaleWidth Properties

Description Set or return the extents of the coordinate system used by the **vsDraw** control.

Syntax [*form.*]*vsDraw*.**ScaleHeight** = [*hei%*]
[*form.*]*vsDraw*.**ScaleWidth** = [*wid%*]

Remarks All drawing on the **vsDraw** control is done in an arbitrary coordinate system, determined by the **Scale*** properties.

For example, the following code draws a rectangle that fills the entire **vsDraw** control, regardless of its physical dimensions:

```
vsDraw.X1 = 0
vsDraw.Y1 = 0
vsDraw.X2 = 1000
vsDraw.Y2 = 1000
vsDraw.Draw = 2   ' Draw = Rectangle
vsDraw.Show
```

You can stretch drawings, zoom, or scroll by modifying the coordinate system and calling the **Show** method.

Default Value 1000

Data Type Integer

ScaleLeft, ScaleTop Properties

Description Set or return the logical origin of the coordinate system used by the **vsDraw** control.

Syntax [*form.*]*vsDraw*.**ScaleLeft** = *left%*
[*form.*]*vsDraw*.**ScaleTop** = *top%*

Remarks All drawing on the **vsDraw** control is done in an arbitrary coordinate system, determined by the **Scale*** properties.

For example, the following code sets the (0,0) point of the logical coordinate system to the center of the **vsDraw** control, regardless of its physical dimensions:

```
vsDraw.ScaleLeft = -vsDraw.ScaleWidth / 2
vsDraw.ScaleTop = -vsDraw.ScaleHeight / 2
```

You can stretch drawings, zoom, or scroll by modifying the coordinate system and calling the **Show** method.

Default Value 0

Data Type Integer

Show Method

Description Renders the current drawing on the **vsDraw** control. Note that you can create an entire drawing, but nothing will appear on the screen until you use this method. This is done to improve rendering speed.

Syntax [*form!*]*vsDraw*.**Show**

Remarks Calling this method is equivalent to setting the **Action** property to 2. The method syntax is cleaner and easier to remember, however, so it is recommended that you use the method instead of the **Action** property.

TextAlign Property

Description Sets or returns the alignment to be used when drawing text.

Syntax [*form.*]*vsDraw*.**TextAlign** = *setting%*

Remarks Valid settings for this property are:

0 - Left
1 - Center
2 - Right

Default Value 0 - Left

Data Type Integer (Enumerated)

TextAngle Property

Description Setsor returns the angle, in tenths of degrees, between the base line of a character and the horizontal.

Syntax [*form*.]*vsDraw*.**TextAngle** [= *angle%*]

Remarks The angle is measured in the counterclockwise direction when the Y direction is down and in a clockwise direction when the Y direction is up.

Default Value 0

Data Type Integer

TextColor Property

Description Sets or returns the color used to print text.

Syntax [*form.*]*vsDraw*.**TextColor** [= *color&*]

Default Value 0 (Black)

Data Type Long (Color)

TextOut Property

Description Sets a string to be drawn at the current cursor position, determined by the **X1** and **Y1** properties.

Syntax [*form.*]*vsDraw*.**TextOut** = *text$*

Remarks The cursor position is not updated after printing.

The string assigned to **TextOut** may have embedded carriage-returns (**Chr**(13)) characters, which cause line breaks.

Data Type String

Version Property

Description Returns the version of the **vsDraw** control currently loaded in memory.

Syntax *CheckVer%* = [*form*.]*vsDraw*.**Version**

Remarks You may want to check this value at the **Form_Load** event, to make sure the version that is executing is at least as current as the version used to develop your application.

The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.

Data Type Integer

X1, X2, Y1, Y2 Properties

Description Set or return the boundaries of a rectangle used to define the position and size of objects drawn with the **Draw** and **TextOut** properties.

Syntax
[*form.*]*vsDraw*.**X1** [= *value%*]
[*form.*]*vsDraw*.**Y1** [= *value%*]
[*form.*]*vsDraw*.**X2** [= *value%*]
[*form.*]*vsDraw*.**X2** [= *value%*]

Remarks These properties define a rectangle. The coordinates are arbitrary, determined by the **Scale*** properties.

Data Type Integer

vsViewPort Reference

Description The VideoSoft **vsViewPort** is a scrollable container control. With **vsViewPort**, you no longer have to write tedious code to synchronize scroll bars and picture boxes. You decide how big your window should be, then let the **vsViewPort** scroll your controls for you.

File Name VSVIEW.VBX for the VBX version,
VSVIEW32.OCX for the 32-bit OCX version, or
VSVIEW16.OCX for the 16-bit OCX version

Object Type vsViewPort

Note Before you can use a **vsViewPort** control in your application, you must add VSVIEW to your project (see the Visual Basic manual for details). To automatically include VSVIEW in new projects, put it in an AUTOLOAD file. When distributing your application, you should install the vsView files in the user's Microsoft Windows SYSTEM subdirectory.

Remarks The **vsViewPort** automatically scrolls all its windowed child controls, but it does not scroll graphical controls such as **Labels** and **ImageBoxes**.

You can use the **vsViewPort** with the **vsPrinter** control to implement print preview in your programs.

vsViewPort Summary

Properties (default: AutoScroll)

(About)	* AutoScroll	BackColor
BorderStyle	Height	HelpContextID
hWnd	Index	* LargeChangeHorz
* LargeChangeVert	Left	Name
Parent	* SmallChangeHorz	* SmallChangeVert
Tag	Top	* Track
* Version	* VirtualHeight	* VirtualLeft
* VirtualTop	* VirtualWidth	Width

Events

Click	DblClick	DragDrop
DragOver	KeyDown	KeyPress
KeyUp	MouseDown	MouseMove
MouseUp	* Scroll	

AutoScroll Property

Description Sets or returns a value that determines whether **vsViewPort** should automatically scroll its contents when the user clicks the scroll bars.

Syntax [*form*.]*vsViewPort*.**AutoScroll** [= {**True**|**False**}]

Remarks This property should be set to **True** in most applications, so that scrolling is done automatically by the **vsViewPort** control.

If you wish to process scrolling yourself, set **AutoScroll** to **False** and respond to the **Scroll** event.

Default Value True

Data Type Integer (Boolean)

LargeChangeHorz, LargeChangeVert Properties

Description Set or return the amount of change to the **VirtualLeft** and **VirtualTop** properties when the user clicks the area between the scroll box and scroll arrow, in Twips.

Syntax [*form*.]*vsViewPort*.**LargeChangeHorz** [= *value&*]
[*form*.]*vsViewPort*.**LargeChangeVert** [= *value&*]

Default Value 300 Twips

Data Type Long (Twips)

Scroll Event

Description Fired whenever the values of **VirtualLeft** and **VirtualTop** change.

Syntax **Sub** *vsViewPort*_**Scroll ()**

Remarks If **AutoScroll** is set to **True**, the **Scroll** event is fired after the actual scrolling takes place. If **AutoScroll** is set to **False**, the **Scroll** event is fired anyway.

SmallChangeHorz, SmallChangeVert Properties

Description Set or return the amount of change to the **VirtualLeft** and **VirtualTop** properties when the user clicks the scroll arrow, in Twips.

Syntax [*form*.]*vsViewPort*.**SmallChangeHorz** [= *setting&*]
[*form*.]*vsViewPort*.**SmallChangeVert** [= *setting&*]

Default Value 30 Twips

Data Type Long (Twips)

Track Property

Description Sets or returns a value that determines whether the **vsViewPort** control should perform scrolling while the user moves the scroll boxes.

Syntax [*form*.]*vsViewPort*.**Track** [= {**True**|**False**}]

Remarks Setting **Track** to **True** provides more user feedback during scrolling, but it can slow down your application and cause some flicker. If in doubt, try it both ways and see which works best for you.

Default Value **False**

Data Type Integer (Boolean)

VirtualHeight, VirtualWidth Properties

Description Set or return extent of the area that can be seen by scrolling, regardless of the actual size of the control. Units are Twips.

Syntax [*form.*]*vsViewPort*.**VirtualHeight** [= *value&*]
[*form.*]*vsViewPort*.**VirtualWidth** [= *value&*]

Remarks If the **VirtualHeight** is smaller than the actual height of the control, the **vsViewPort** automatically hides the vertical scroll bar. The same applies to **VirtualWidth** and the horizontal scroll bar.

See also the description of the **VirtualLeft** and **VirtualTop** properties.

Default Value 0 Twips

Data Type Long (Twips)

Version Property

Description Returns the version of the **vsViewPort** control currently loaded in memory.

Syntax *CheckVer%* = [*form*.]*vsViewPort*.**Version**

Remarks You may want to check this value at the **Form_Load** event, to make sure the version that is executing is at least as current as the version used to develop your application.

The version number is a three digit integer where the first digit represents the major version number and the last two represent the minor version number. For example, version 3.5 would return 350.

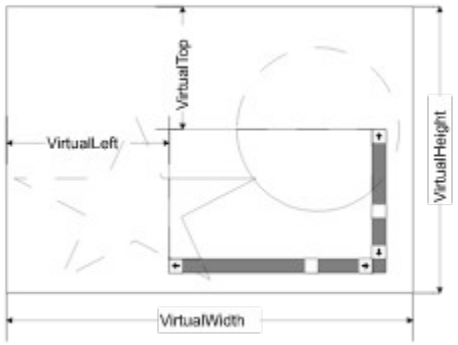
Data Type Integer

VirtualLeft, VirtualTop Properties

Description Set or return the coordinates of the top left point of the scrollable area, in Twips.

Syntax [*form.*]*vsViewPort*.**VirtualLeft** [= *value&*]
[*form.*]*vsViewPort*.**VirtualTop** [= *value&*]

Remarks The diagram below illustrates the relationship between **VirtualLeft, VirtualTop, VirtualWidth**, and **VirtualHeight**:



Changes to these properties cause **ViewPort** to scroll its contents and to fire the **Scroll** event.

Default Value 0 Twips

Data Type Long (Twips)

VideoSoft Products

To get an order form, click **HERE**.

vsOCX/vsVBX

A set of three custom controls for interface design and text parsing.

Icon	Name	Object	Description
	Elastic	vsElastic	Smart containers that resize themselves and their child controls, automatically create labels and 3-D frames for its child controls, and can also be used as progress indicators and labels.
	IndexTab	vsIndexTab	Allows you to group controls by subject, using the familiar notebook metaphor that has become a Windows standard.
	Awk	vsAwk	Parsing engine named and patterned after the popular Unix utility, plus a powerful expression evaluator.

vsView

A set of four custom controls for creating, viewing, and printing text and graphics.

Icon	Name	Object	Description
	InForm	vsInForm	A control that you can drop into any container to customize its title bar, frame, resizing behavior, and frame buttons. InForm also allows you to monitor the clipboard, drag and drop files from File manager, and more.
	Printer	vsPrinter	A much improved printer object with word wrap, headers and footers, multi-column printing, graphics, and multi-page Print Preview capability.
	ViewPort	vsViewPort	A control that gives you a scrollable virtual area so you can fit more controls in your windows. Great for implementing Print Preview and programs that look like the Program Manager.
	Draw	vsDraw	A versatile drawing control that lets you create complex images, view them on the screen, copy them to the clipboard, or print them. Great for technical drawings, maps, and diagrams.

vsFlex

A set of two custom controls for analyzing, formatting, and displaying information.

Icon	Name	Object	Description
	FlexArray	vsFlexArray	A new way to display and operate on tabular data. FlexArray gives you total flexibility to display, sort, merge, and format tables containing strings and pictures.
	FlexString	vsFlexString	A powerful regular expression engine. With FlexString, you can find and replace patterns in strings. Use it to provide regular expression search-and-replace capabilities or to parse input strings.

Order Form

(You may print this form by selecting the File|Print command).

TO: VideoSoft
2625 Alcatraz Avenue, Suite 271
Berkeley, California 94705

To order by phone, call
(800) 547-7295 (from within the US)
(510) 704-8200 (from anywhere)
(510) 843-0174 (fax)

Please register my copy of the following VideoSoft products. I am enclosing a check or money order for the amount of:

OCX Version (includes VBX)		
VSOCX.OCX	Single developer	US$ 99.00
	Additional developers	__ x 99.00
VSVIEW.OCX	Single developer	149.00
	Additional developers	__ x 149.00
VSFLEX.OCX	Single developer	149.00
	Additional developers	__ x 149.00
VBX Version		
VSVBX.VBX	Single developer	US$ 45.00
	Additional developers	__ x 45.00
VSVIEW.VBX	Single developer	99.00
	Additional developers	__ x 99.00
VSFLEX.VBX	Single developer	99.00
	Additional developers	__ x 99.00

Note: Call us for details on site licenses and volume discounts.

Name:

Company:

Street:

City, State, ZIP:

Country:

Phone:

Where did you hear about the VideoSoft products?