Known Limitations

ReSize is intended to be used with Visual Basic™ 3.0. It is not designed for use with C/C++ or Delphi, Power Builder or any other product other than Visual Basic™ 3.0. ReSize 3.60 is for the most part compatible with Visual Basic™ 4.0 (16-bit) with a few known exceptions. ReSize will not work properly with the tab control that ships with VB 4.0. ReSize.VBX (like all .VBXs) will not run at all with the 32 bit version of Visual Basic™ 4.0. We are working on a ReSize .OCX that will be able to run in 32 bits. We hope to have beta copies of the .OCX available in late Feburary or early March of 1996.

The ReSize custom control is able to resize most controls with top, left, height and width properties. ReSize can not operate on text that has been printed on a form or in a picture box. The ReSize control may actually end up clearing this text as a result of a resize operation.

In some cases there is a limit to the extent that ReSize can shrink a font. This is a Windows restriction, not one imposed by ReSize. (For more information see resizing fonts with the ReSize custom control).

You need write a small amount of code to set and clear the enabled property when using ReSize with MDI child forms (see Resizing MDI child forms under features and tips).

ReSize has been tested with the standard controls that ship with Visual Basic™ 3.0 Professional Edition, but it may or may not work with any given 3rd party controls. Some grid and tab controls will not work properly with ReSize.

Resizing fonts with the ReSize custom control

The ReSize custom control will resize the fonts of all of the controls on a form as well as resizing the controls themselves.

Be aware that some fonts will not shrink beyond a certain point and that some fonts will substitute Small Fonts when they shrink beyond a certain point. In the case of fonts that substitute Small Fonts, ReSize is smart enough to see that the original font is restored to each control when the control again assumes a size that is large enough to support the originally specified font.

While most controls are able to change size in pixel increments, fonts are scaled according to point size. Fonts do not always scale in perfect proportion to the controls that contain them.

Properties

About	Displays an about box.
Enabled	Enables or disables the functioning of the ReSize control.
Index	The standard index property. (*Generally you should **not** place ReSize in a control array*.)
FormMinHeight	The minimum height that the form can be resized to at runtime.
FormMinWidth	The minimum width that the form can be resized to at runtime.
Left	Stores the position of control.
Name	Contains the name of the ReSize control.
Parent	The standard parent property. Returns the form that contains this ReSize control. (*Available only at runtime.*)
Tag	User property.
Top	Stores the position of the control.
Version	Contains version information for the control.

Events

ReSize has only one event.

PreResize

This event fires after ReSize determines the new size and position of each control, but before it actually makes the change to the size and position. You may alter what ReSize is about to do by altering the NewTop, NewLeft, NewHeight, NewWidth, or NewFontSize properties while processing this event (see resizing scroll bars and data controls under features and tips for pointers on keeping the width of scroll bars from changing).

Parameters:

ControlName	The name of the control about to be resized.
ControlType	The type of control about to be resized. (i.e. CommandButton)
NewTop	The new value that the controls top property will be set to.
NewLeft	The new value that the controls left property will be set to.
NewHeight	The new value that the controls height property will be set to.
NewWidth	The new value that the controls width property will be set to.
NewFontSize	The new value that the controls font size property will be set to.

Methods

The ReSize control has no methods. It takes action automatically when the form that contains it is resized.

About Property

Clicking the about property from the property box in the Visual Basic™ development environment or setting the about property will bring up the ReSize about box.

Version Property

The version property contains the numeric version of the ReSize control.   This property also contains the word Shareware, Runtime, or Registered depending on which version of the control you are running.

There is really no difference be Runtime and Registered. ReSize will display Registered in the version property any time it detects the ReSize.LIC development license file.

FormMinWidth Property

When the FormMinWidth property is set to a value greater than zero the parent form of the ReSize control will not shrink to a width beyond this minimum.   Use this property to limit how small you let a user resize a form.

FormMinHeight Property

When the FormMinHeight property is set to a value greater than zero the parent form of the ReSize control will not shrink to a height beyond this minimum.   Use this property to limit how small you let a user resize a form.

Using ReSize with MDI parent forms

ReSize now supports MDI parent forms. The ReSize control cannot be placed directly on an MDI parent form. It should be placed on a picture box, 3D pannel (or other 3rd party control) that can be placed directly on an MDI form.

When used with an MDI parent form the ReSize will resize all controls that are associated with the MDI parent form. It will not attempt to resize MDI child forms contained in the MDI parent. If you wish to do this you must write a small amount of code and drive it from the resize event of the MDI parent. You may place a ReSize control in each MDI child form. This will cause the child forms controls to be resized each time the MDI child is resized. See Resizing MDI child forms for more information about placing ReSize controls on MDI child forms.

Release Notes for version 3.60

Paint is fast again:
ReSize 3.50 repainted forms slowly (relatively speaking) after a resize operation in order to be more compatible with VB 4.0. ReSize 3.60 is able to paint at a speed comparable to previous (pre-3.50) versions of ReSize and still paint correctly in VB 4.0. In some cases it may be necessary to set the Clip Controls property to False to get an accurate repaint.

Fixed out of memory error:
Loading and unloading many forms (or the same form again and again) could previously have caused Visual Basic™ to run out of memory. Forms with many controls encountered this problem sooner than forms with fewer controls. This has now been corrected.

Fixed a memory leak:
ReSize 3.50 contained a memory leak that could sometimes cause the control to behave in a random fashion. This is now corrected.

ReSize now terminates subclassing properly:
Subclassing is now terminated more gracefully when a form is unloaded. This prevents the control from conflicting with the subclassing of some other 3rd party controls.

Context sensitive help now works for FormMinWidth and FormMinHeight properties:
Previously these properties did not respond properly when the F1 key was pressed to invoke context sensitive help in design mode.

See also: Release Notes for version 3.50

Release Notes for version 2.00

The line control is now properly resized:
The line control that ships with Visual Basic™ is now properly handled by ReSize. The line control is a special case, because it does not contain the standard left, top, height, and width properties. The line control will now resize properly even when you set the ScaleWidth and ScaleHeight properties of a form to custom values.

Support for the MicroHelp tab control:
ReSize will now work correctly with the MicroHelp tab control that ships with VBTools 4.

Fonts are now limited at 1.9 point size:
Some fonts appear to have trouble when ReSize sets their point size to less than 1.9, but only in certain circumstances. These fonts include Arial, Arial Rounded Bold, Britanic Bold, and Times Roman New. These are fonts that switch to Small Fonts when sized below about 6 points. Since 1.9 point seems as small as anyone would ever want to set a font, we have elected to limit fonts from being set smaller than this. Doing so corrects the problem. We choose not to try to predict which fonts will have the problem and under what circumstances. We think our solution is faster more compact and perhaps more reliable than predicting exactly when the problem will occur.

Minimizing a form is handled better:
When a form is minimized, ReSize no longer alters the size of the controls on the form. This makes applications run a little faster and avoids some problems that previously could cause ReSize to generate a stack overflow.

A memory leak in ReSize 1.50 has been corrected:
ReSize 1.50 could cause Visual Basic™ to display an out of memory error. This was due to a memory leak in the 1.50 version of ReSize. This has now been corrected.

.FRX files are now included in distributed files:
.FRX files that make up part of the VBSAMPLE program were omitted in the past. This has now been corrected.

See also: Release Notes for version 1.50

Release Notes for version 3.50

ReSize now supports MDI parent forms.
MDI forms may now include a ReSize control. The control must be placed inside a picture box or panel because it is not possible to paste a ReSize (or any control) directly to an MDI parent form.

ReSize now redraws slower, but correctly in all cases.
Previous versions of ReSize did not always properly execute a redraw for controls inside of various containers. This was a serious problem particularly in Visual Basic™ 4.0.
This problem has now been corrected (at the expense of some redraw speed). See Repaint problems with Frames.

Fixed a bug that could cause an MDI application to crash.
When an MDI parent terminated before its child forms and any of the child form contained a ReSize control an application could crash. Internal memory structures are now released when the ReSize control is destroyed as opposed to when a form is destroyed. Subclassing of the MDI child form also comes to an end when the ReSize control is destroyed as opposed to when the MDI child form is destroyed. This eliminates the possibility of trying to access the ReSize control after it has been destroyed just prior to the form being destroyed.

Fixed a bug that could cause an MDI child form to maximize improperly.
The internal message handling code for the WM_MINMAXINFO message now calls the default form procedure after setting information in the MINMAXINFO structure. Returning a 0L as documented in the SDK only works for non-MDI child forms.

Fixed a bug related which could cause VB to crash when ReSize was run from an .EXE and the VB development environment at the same time.
The subclass procedure of ReSize was incorrectly calling the next procedure in the chain due to an error in retrieving the previous subclass address from ReSizes internal memory structures. This eventually caused ReSize to return a null pointer when calling the VBGetControl VBAPI function with the GC_CONTAINER parameter. The null pointer then caused ReSize to perform an illegal memory reference and crash.

See also: Release Notes for version 3.00

Release Notes for version 3.00

The license file can now be in \windows\system.
Registered versions of ReSize can now locate the license file in the \Windows\System directory as well as in the \Windows directory and the current working directory.

A bug was fixed that is related to running multiple program instances containing ReSize.
When two separate program instances (could be the same or different programs) were run and the first instance that was started was not the last instance to terminate, ReSize would GPF. This was due to the internal handling of global memory and has now been corrected.

ReSize now includes the standard index property.
This was added for completeness, but you still should never place more than one ReSize control on a single form.

ReSize now includes an Enabled property.
Setting the Enabled property to false turns ReSize off. This is particularly useful when you are working with MDI child forms. See resizing MDI child forms for more information about working with MDI child forms.

ReSize now includes the standard Parent property.

A PreResize event was added to ReSize.
This event fires after ReSize determines the new size and position of each control, but before it actually makes the change to the size and position.

Lines now correctly resize after a form has been resized to a zero twip height.
In the past, setting a form to a height of zero could cause a GPF or cause lines to be in the wrong position when the form again assumed a non-zero height. Whenever a form is resized to a height of 0 or minimized, controls on the form are not altered. Since you cant see the controls on a 0 height form or a minimized form, the fact that the controls have not been resized should not matter.

ReSize now includes the FormMinHeight and FormMinWidth properties.
These properties allow you to prevent the parent form of ReSize from shrinking below a predetermined minimum.

See also: Release Notes for version 2.00

Release Notes for version 1.50

Controls can now be moved at runtime:
ReSize version 1.50 is smart enough to keep tract of controls that are moved around and sized by VB code at runtime.   Version 1.00 would simply move these controls back to their original relative positions and sizes each time the form was resized.

Fonts can now be changed at runtime:
ReSize version 1.50 now allows you to change all font properties (i.e. fontname, fontsize, etc.) at runtime.   Version 1.00 would set changed fonts back to there design time property values each time the form was resized.

Control now contains a version stamp:
ReSize now contains version information that can be read by ver.dll.   This is helpful when you include ReSize in projects that are redistributed with the Visual Basic™ setup wizard and other installation tools.

License bug fixed:
The registered version of ReSize 1.00 would not load into the development environment unless the ReSize.LIC file was in the Visual Basic™ working directory.   The documentation directed registered developers to place the ReSize.LIC file in the \windows directory.   This has been corrected and ReSize will now function correctly in the development environment if the ReSize.LIC file is in either the Visual Basic™ working directory or the \windows directory.

Text not staying bold bug:
ReSize 1.00 would unbold text in some cases when controls were resized to very small sizes and then made large again.   This is now corrected.

Features and Tips

Resizing MDI child forms

ReSize now includes the Enabled property that makes it relatively easy to ReSize MDI child forms. These forms must be handled slightly differently than other forms. The initial size of an MDI child form is not under program control. This is usually corrected by setting the size of the form in the form load event of the MDI child form. To work properly, a ReSize control placed in an MDI child form should be disabled at design time. After the form has been sized in the form load event is the correct time to enable ReSize.

```
Sub Form_Load ()

        MDIChildForm1.Width = 3000
        MDIChildForm1.Height = 5000

        MDIChildForm1.ReSize1.Enabled = True

End Sub
```

Resizing scroll bars and data controls

It is frequently desirable to resize data controls in such a way that the height of the control remains constant. This is also true of horizontal scroll bars. Vertical scroll bars are similar, but it is desirable to keep the width of vertical scroll bars constant. ReSize now has the ability to do this. In the PreResize event of the resize control place the following code:

Sub ReSize1_PreResize (ControlName As String, ControlType As String, NewTop As Long, NewLeft As Long, NewHeight As Long, NewWidth As Long, NewFontSize As Single)

```
If ControlName = Data1 then
      NewHeight = Data1.NewHeight
End If
```

End Sub

You may also wish to reposition a data control or scroll bar slightly to correct for the height or width remaining constant. This is best done in the resize event of the associated form (or other container).

Sub Form_Resize ()

```
VScroll1.Left = Form1.Width - VScroll1.Width
```

End Sub

Retrofitting existing applications

Retrofitting existing applications is as easy as placing a ReSize control on each form you wish to give resizing capabilities.

Repaint problems with Frames

This version of ReSize uses a new method to repaint forms after a resize operation. If you experience any problems where the form you are resizing is not fully repainted, try setting the Clip Controls property of that form to False. In our tests this has corrected the problem in all cases.

Moving controls around on the form

If you move controls around on a form under control of your program, ReSize is smart enough to detect this.   The controls will be positioned and sized in accordance with the new positions assigned by your Visual Basic™   program.

Using ReSize with grid controls

ReSize is able to resize the outer dimensions of most grid controls, but it does not resize each individual cell of the grid. In some cases, the cells of a grid may appear to be changing in size due to changes in the font size which ReSize DOES make to the grid control as it is resized.

There are many grid controls available and they all handle sizing of cells a little bit differently. At least at this point, we have not attempted to make ReSize recognize each brand of grid control and handle its resizing requirements as a special case.

It is fairly easy to combine ReSize with some of your own Visual Basic™ code to correctly resize the cells of a grid control. The following code handles resizing of cells in the grid control that ships with Visual Basic™. You should be able to write similar code for other grid controls on the market.

Create a form with a ReSize control and a grid control. Set the grid to 5 columns and 9 rows. Place the following code in the form resize event. In practice you would probably use a for loop.

```
Sub Form_Resize ()
    Grid1.ColWidth(0) = Grid1.Width / 5
    Grid1.ColWidth(1) = Grid1.Width / 5
    Grid1.ColWidth(2) = Grid1.Width / 5
    Grid1.ColWidth(3) = Grid1.Width / 5
    Grid1.ColWidth(4) = Grid1.Width / 5
    Grid1.RowHeight(0) = Grid1.Height / 9
    Grid1.RowHeight(1) = Grid1.Height / 9
    Grid1.RowHeight(2) = Grid1.Height / 9
    Grid1.RowHeight(3) = Grid1.Height / 9
    Grid1.RowHeight(4) = Grid1.Height / 9
    Grid1.RowHeight(5) = Grid1.Height / 9
    Grid1.RowHeight(6) = Grid1.Height / 9
    Grid1.RowHeight(7) = Grid1.Height / 9
    Grid1.RowHeight(8) = Grid1.Height / 9
End Sub
```

Please note that you can divide by numbers other than 9 and 5 to get varied effects. Note also that you don't have to divide each row or column by the same number.

The above code forces cell sizes to be relative to the total size of the grid control which ReSize is manipulating. This works because ReSize changes the sizes and locations of the controls on the form before the form resize event fires.

ReSize Custom Control

Properties	Events	Methods	Known Limitations
Register	Release notes	Features & Tips	

Description
A ReSize control resizes all other controls on the same form each time the form changes its size.

File Name
RESIZE.VBX

Object Type
ReSize

Remarks
Place a ReSize control anywhere on a form and it will cause the other controls on the form to resize along with the form. This saves you from writing code. ReSize is so simple to use that you dont need to set even a single property to make it work. ReSize is also smart enough to resize fonts on most controls. (See resizing fonts with the ReSize custom control for more information).

You must place a ReSize control on each form that you want to affect. You should not place two ReSize controls on a single form.

The shareware version of the ReSize control will only operate in the development environment. To find out how to get a registered version of ReSize, see the register help topic.

Registering ReSize

When you register your copy of ReSize, you will receive the most recent version of ReSize. The registered version of ReSize will run in both the runtime and development Visual Basic™ environments. The shareware version of ReSize runs only in the development environment. The nag screen is not present in the registered version. Registered users of ReSize may distribute the ReSize runtime for free.

Cost: $29.00 U.S.

ReSize may be registered with Component Source™ in the UK. Instructions for registering with Component Source are detailed elsewhere on this CD.