ReSize OCX Custom Control

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

File Name
ReSize16.OCX for 16-bit Visual Basic™
ReSize32.OCX for 32-bit Visual Basic™

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.

ReSize is also available as a .VBX and is sold separately.

For the most up to date information on ReSize, check out the Larcom and Young web site at **http://www.lyoung.com**-

Registering ReSize

Registered users of ReSize receive the current version of ReSize, version updates, support via Email (70555.312@compuserve.com), the ability to operate the control outside of the VB development environment, and the right to redistribute the runtime only version of ReSize royalty free.

ReSize OCX is priced at $49.00 U.S.   This price includes both the 16 and 32 bit OCX's. It does NOT include the VBX version of ReSize.

You may purchase ReSize directly from Component Source™ by calling

United Kingdom	**+44 (0) 118 958 1111**
France	**0800 90 92 62**
Germany	**0130 86 07 06**
Italy	**1677 90046**
Sweden	**020 794 989**
The Netherlands	**06 022 8832**

Detailed instructions for purchasing software from Component Source™ are included on the Component Source™ CD.

Known Limitations

The ReSize OCX is intended to be used with Visual Basic™ 4.0 and 5.0.   It is not designed for use with C/C++ or Delphi, Power Builder or any other product other than Visual Basic™.   ReSize OCX (like all OCXs) is not compatible with versions of Visual Basic™ prior to 4.0.   You must use the 16-bit version of ReSize (ReSize16.OCX) with the 16-bit version of Visual Basic™ and the 32-bit version of ReSize (ReSize32.OCX) with the 32-bit version of Visual Basic™.

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).

The ReSize OCX   has been tested with the standard controls that ship with Visual Basic™ 4.0 Professional and Enterprise Editions, but it may or may not work with any given 3rd party controls. Some grid and tab controls will not work properly with ReSize.   Limited testing has been done with Visual Basic 5.0.

ReSize can be placed inside OCX controls created with VB 5.0 and VB 5.0 CCE.   ReSize will perform properly if these OCXs are run from Visual Basic programs, but it will not (at this time) perform correctly if these OCXs are run from Access, Internet Explorer, or Delphi etc.

The SSTab control that ships with Visual Basic is now supported by ReSize OCX.   ReSize does not correctly handle SSTab controls placed inside OCX controls created with VB 5.0 or VB 5.0 CCE. The control will work, but proportions will be slightly distorted.   We expect to have this working correctly in a future release.

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.

For best results, you should use true type fonts with ReSize.

Properties

About	Displays an about box.
AutoCenterFormOnLoad	Causes form to be centered each time it is loaded.
Enabled	Enables or disables the functioning of the ReSize control.
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.
Index	The standard index property. (*Generally you should **not** place ReSize in a control array*.)
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.
IgnoreControl	IgnoreControl is always passed in as FALSE. If you set it to TRUE in the PreResize event, this particular control will be ignored by ReSize. It will not be moved or sized in any way.
CurrentControl	The actual object that ReSize is about to size and position. You can use this parameter to reference various properties of the object. Ex.: NewHeight = CurrentControl.Height

Methods

ReSize has only one method.

CenterForm

The CenterForm method causes ReSize to center the form that contains it. The form will be centered in the middle of the physical screen if it is an SDI form and in the middle of its MDI container if it is an MDI form.

Please note that there is also a property called AutoCenterFormOnLoad that performs a similar function automatically each time a form is loaded.

If you have a form that you wish to remain centered at all times, you must re-execute this method after a user manually moves or sizes the form or after your VB code sizes the form.

About Property

Clicking the about property from the property box in the Visual Basic™ development environment 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 between Runtime and Registered. ReSize will display Registered in the version property any time it detects the ReSizOCX.LIC development license file.

AutoCenterFormOnLoad

The AutoCenterFormOnLoad property determines if ReSize should automatically center a form on the screen each time the form is loaded. Please note that there is also a method called CenterForm. Calling this method will center a form any time during the life of the form.

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 2.5

Made ReSize slightly more efficient .
ReSize will no longer take any action if windows fires a WM_SIZE message, but the form size does not change. This eliminates unneeded processing and in some cases reduces screen flicker.

*Added the **CurrentControl** parameter to the PreReSize event .*
The PreReSize event now includes the CurrentControl parameter. CurrentControl is of type Object and represents a pointer to the control that ReSize is currently sizing and positioning.

Fixed handling of SSTab control .
Some controls would disappear from SSTabs. The temporary workaround was to place a panel or picture box down on the surface of the tab. The SSTab is now handled correctly and this workaround is no longer necessary.

Fixed a problem with entering debug mode in VB 5.0 .
ReSize could crash when entering debug mode from the development environment while in VB 5.0.

Fixed handling of minimized forms in design mode .
Forms that were minimized as opposed to hidden while in design mode would paint with incorrect initial proportions if form dimensions were being altered in the Form_Load event.

Made the PreResize event more tolerant of errors in VB code .
VB code within the PreResize event could cause a program to hang or crash when the development environment tried to enter debug mode as a result of detecting a run-time error.

ReSize OCX can run at the same time as ReSize VBX .
Corrected an naming conflict that could prevent an .EXE with ReSize OCX from running at the same time as an .EXE with ReSize VBX.

Fixed a version conflict with a previous beta version of ReSize .
A limited number of users with 1.7 beta version of ReSize experienced problems when upgrading to a higher version of ReSize. This problem was corrected.

Made Larcom and Young version numbering compatible with more install utilities .
In the future all patch, beta, and other intermediate releases will be versioned with the format: <major release>.<minor release>. Letters will not be used (ex: 1.7d) for minor releases.

Made several corrections to the ReSize documentation.

For more information see Release Notes for version 2.00

Release Notes for version 2.00

ReSize now lets you design your application in ANY resolution. Previously you had to design for the lowest common denominator (usually 640 X 480).

ReSize now lets you design with your form maximized.

You no longer have to write any code at all to synchronize ReSize to an MDI child form.

ReSize now supports changing the size of a form in the Form_Load event.

The PreResize event now has an argument called IgnoreControl. Setting IgnoreControl to TRUE in the PreResize event causes ReSize to ignore the control. In other words, the control will not be moved or resized in any way.

Modified ReSize to handle the line property correctly in VB5 CCE.

Corrected a problem which could cause controls to size improperly after a form was minimized and then restored.

Fixed several problems in ReSize that could GPF or hang an application if an END statement was executed in the Form_Load event of the control. The same could also happen if a developer set a break point in the Form_Load event and then terminated execution of the program.

Added code to allow ReSize to properly handle fonts for controls that are dynamically added at run time.

In previous versions of ReSize, you could not properly set the font size of controls from the PreResize event of the ReSize control. This has now been corrected.

Corrected missing quotation marks in the on-line documentation.

Corrected PsL phone number in the online documentation.

ReSize now has the correct version embedded in the resource portion of the .OCX file.

Fixed a bug in ReSize that could cause ReSize to not resize one or two controls on a form.

For more information see Release Notes for version 1.60

Release Notes for version 1.60

Added a CenterForm method.   Executing this method causes the form containing the ReSize control to be centered on the physical screen if it is an SDI form and centered in its MDI container if it is an MDI form.

Added AutoCenterFormOnLoad property.   This property causes the form to be centered each time it is loaded.

SSTabs are now supported in forms of any scale mode.

SSTabs worked correctly in the 32 bit version of ReSize OCX 1.50, but not in the 16 bit version.   This has now been corrected.

ReSize was changed to internally set font sizes before it sets left, right, width or height properties.   In some cases setting a controls properties in the opposite order will give the control an improper height setting.

Previously the 32 bit version of ReSize OCX caused combo boxes to operate incorrectly.   They would not display a drop down list and they would not fire change or click events.   This problem has been corrected in the present version of ReSize OCX.

ReSize would not work properly with beta versions of NT 4.0 in the past.   The background of the form containing the ReSize control would not paint.   This has been corrected.

A problem was fixed that could cause ReSize to register itself incorrectly.

For more information see Release Notes for beta version 1.50

Release Notes for version 1.50

1.50 is the first production (non-beta) release of the ReSize OCX. Earlier versions (1.00, 1.10, 1.20, and 1.2x) were all beta releases.

ReSize OCX can now correctly move and resize controls placed inside a Visual Basic 4.0 SSTab control. In this version of ReSize the form containing the SSTab control must have the ScaleMode property set to 1 - Twip for resize to work properly. In the next release of ReSize all ScaleModes (including user defined) will be supported.

A problem was fixed that sometimes caused ReSize to record the incorrect initial sizes and positions of controls.

Forms that were scaled to small increments were not being handled correctly. Control properties are now stored inside the ReSize control as real values (as opposed to integers). This corrects the problem.

For more information see Release Notes for beta version 1.20

Release Notes for beta version 1.20

The ability to move controls at runtime was added to the ReSize OCX. Controls on a form can now be relocated at runtime under program control. This behavior is now compatible with the VBX version of ReSize.

For more information see Release Notes for beta version 1.10

Release Notes for beta version 1.10

The file MFC40.DLL was added to the distribution.   The copy of MFC40.DLL that ships with Visual Basic 4.0 is not current enough to allow the 32 bit version of ReSize to register properly.

A problem in the ReSize licensing code which expressed itself in Windows 3.x when running the 16 bit version of ReSize was corrected.

For more information see Release Notes for beta version 1.00

Release Notes for beta version 1.00

The DBList box that ships with VB 4.0 does not operate properly with ReSize.   This is probably due to a bug in the list box control.   We are presently investigating this.

The DBGrid control that ships with VB 4.0 sometimes fails to repaint properly on the screen after a form has been resized.   The present work around for this is to force a repaint of the form containing the DBGrid control (Form1.Refresh in the Resize event of the form).

This beta version of the ReSize OCX will not work properly with the tab control that ships with VB 4.0. We hope to have ReSize working with the tab control at some point in the not too distant future.

Features and Tips

SSTabs inside VB created OCXs

ReSize does not correctly handle SSTab controls placed inside OCX controls when those OCX controls are created with VB 5.0. The size and placement of controls may be slightly inaccurate. We expect to have this working correctly in a future release of ReSize.

VB 5.0 Created OCXs

ReSize has been modified to work with VB 5.0. In addition to placing a ReSize control on a form, you may now place a ReSize control inside an OCX that you create with VB 5.0. ReSize will proportionally move and size all other controls contained inside the OCX. When an OCX created by VB 5.0 is placed into a VB program, the ReSize that is within the OCX will only size and move controls that are part of the OCX.

If the OCX has been set up to act as a container for other controls, it will not resize controls that are contained but not compiled into the OCX. You should include a second ReSize control on the form containing the OCX to make all controls on the form (including the OCX and any controls contained by it) move and size properly.

Resizing MDI child forms

In previous versions of ReSize you had to write a few lines of special synchronizing code to accommodate MDI child forms.   ReSize OCX is now capable of handling MDI child forms without the need for any additional code.   You dont need to do anything special for MDI child forms.

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, IgnoreControl As Boolean)

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

End Sub

Any time you wish to ReSize to handle some, but not all of a controls properties you should follow a similar procedure. You can make ReSize completely ignore a control by setting the IgnoreControl property to TRUE in the PreResize event.

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.

Converting existing projects containing ReSize.VBX

Visual Basic™ will automatically convert existing projects that contain ReSize.VBX. To take advantage of this feature of Visual Basic™ you need to place a line of the format:

VBXFileName=(<OLEControlGUID>)#<version>#<lcid>;OCXFileName

in the VB.INI file.

Specifically the line(s) corresponding to ReSize16.OCX and ReSize32.OCX are:

```
[VBX Conversions16]
Resize.vbx={A964BDA3-3E93-11CF-9A0F-9E6261DACD1C}#2.5#0;c:\resizedir\ReSize16.OCX

[VBX Conversions32]
Resize.vbx={A964BDA3-3E93-11CF-9A0F-9E6261DACD1C}#2.5#0;c:\resizedir\ReSize32.OCX
```

These lines should be placed in the VBX Conversions16 and VBX Conversions32 sections of the VB.INI file. Alter the value of resizedir in the lines above to reflect the location of the ReSize OCX files on your system. All other information should be entered exactly as shown.