CUA Controls package for VBX environments, Stingsoft

(for example Visual Basic, Visual C++, Borland C++, PowerBuilder, SQL-Windows, dBase, Delphi)

What is CUA/Controls?

CUA Controls contains 22 different components in VBX-format for usage in your application. Some components are enhanced versions of standard components that are part of Visual Basic Professional, others are completely new without parallels in VB.

The most popular and requested components of todays modern applications has been integrated in this package. Among these are:

tool bars like MS Word, complete with tool tips
tabbed dialogs like in MS Excel 5.0 and MS Word 6.0. These are very common in Windows 95 (Chicago).
note books (popular in OS/2),
vertical/horizontal/circular progress indicators,
status bars with built in time/ date/ keyboard status support,
tool boxes,
command buttons with pictures,
spin buttons,
sliders,
knobs,
formatted input,
list boxes and combo boxes with pictures
thermometers.

The purpose is to enable rapid development of modern user interfaces, for prototypes as well as for commercial applications. All components are well tested and simple to use.

Most components may be attached to Visual Basic:s data control, which eases rapid and efficient development of data base applications. As an extra aid, CUA Controls has advanced but easy-to-use support for input validation.

There are built in options for three dimensional effects, however very limited. A lot of other VBX packages (such as for example "VB Tools") have a lot of properties for customization of 3D looks. The reasons for the somewhat limited variety of "cosmetics" in CUA Controls concerning 3D are that performance and efficiency where given a high priority when this package was designed. The more properties put into a control, the more resource consuming - and slower - it gets. Parts of CUA Controls are assembly language for the same reasons; efficiency.

CUA Controls can change appearance like a chameleon. It can take on the looks of a number of popular environments. Besides OS/2 and CUA, which are default, Windows 3, Windows 3/3D, Borland BWCC and Windows 95 (Chicago) can be emulated. Instead of controlling the individual appearance (OS/2, Windows 3 osv) of every single control, CUA Controls as a whole is configured via a call to the subroutine ConfigCUA. This way, the look of the user interface can be changed as a whole using one single call. Using this functionality you can begin to develop applications using the Windows 95 look now! It also makes it very easy to create applications where the designer (or the user perhaps) can decide what look to use.

Remarks

This help file is distributed with the CUA Controls package, a product from Stingsoft.

Installing CUA Controls

Install CUA Controls using the installation utility (SETUP.EXE).

1. Insert the distribution disk in station A (or B).
2. Choose **File/Run** from the menu of File Manager or Program Manager.
3. Type **a:\setup** (or **b:\setup**).
4. Follow the instructions on your screen.

The installation program will copy **cuactls.vbx**, **cuactls.hlp** and **cuarun10.dll** to the Windows system catalog (C:\WINDOWS\SYSTEM). All other VBX:es are copied to the directory named at installation (C:\CUACTLS). During installation a subdirectory named 'SAMPLE' is created, and the demonstration project is copied there.

Using CUA/Controls

This section describes how to use CUA/Controls with Visual Basic.

In Visual Basic VBX components are used on a project basis. When you have included a VBX in your project and saved it, the component will be shown in the tool box whenever you reopen the project.

To include a VBX in a project, first open your project. Then choose menu item 'Add File' from the 'File' menu. Depending on which VBX component you want to use in your app, choose among the following files:

cuactls.vbx	All 22 components.
cuabook.vbx	Note book (OS/2 style).
cuacheck.vbx	Check button.
cuaclock.vbx	Analogue clock.
cuacmd.vbx	Command button.
cuacombo.vbx	Combo box.
cuaedit.vbx	Text box.
cuaframe.vbx	Group frame.
cuagrid.vbx	Table window.
cuaicon.vbx	Icon with adherent text.
cualabel.vbx	Lable.
cualamp.vbx	Lamps, "traffic lights".
cualist.vbx	List box.
cuameter.vbx	Progress indicator.
cuaradio.vbx	Radio button.
cuarotor.vbx	Knob.
cuaslide.vbx	Slider.
cuaspin.vbx	Spin button.
cuastat.vbx	Status bar.
cuatabs.vbx	Tabbed dialog.
cuatermo.vbx	Thermometer.
cuatolbn.vbx	Tool button.
cuatolbx.vbx	Tool box.
cuatoltp.vbx	Tool tips.

The icon/icons for the chosen component will appear in Visual Basic:s tool box. Usable constants and declarations for CUA/Controls are supplied in the file **cuactls.bas**, suitable to be included in your project.

When the VBX components has been loaded and shown in VB:s tool box, they are ready to be used just like any other standard component.

Using help

The help file is an extensive reference containing information on CUA/Controls. The texts follow the clear structure and grouping of subjects that has made Visual Basic:s help texts so easy to use.

Here are four ways to request help on CUA/Controls:

1. Double click the icon for CUA/Controls help in program manager. This takes you to the table of contents in help for CUA/Controls.
2. Select a component in the tool box and press F1. This takes you directly to the section about the selected component.
3. Select a property   in the property window and press F1. This will show you help on the selected property directly.
4. When the "Procedure" box in Visual Basic:s code window has focus, you will be presented help on the currently selected event.

Buying CUA/Controls

Unregistered evaluation copies of CUA/Controls are fully functional, but will present a "nag window" presenting it as an evaluation copy each time it is used. When CUA/Controls has been licensed, this "nag window" will no longer appear.

If you find CUA/Controls attractive, you can license it and receive a complete version that you can distribute with your programs. The complete version includes a printed handbook, as well as each component as a separate VBX, so that only necessary components need to be loaded.

Important

This section contains tips on how to use CUA/Controls.

To accomplish three-dimesional effects, CUA/Controls sometimes draw outside the components own client area. This makes the component somewhat larger, and the window may need some adjustment before everything looks good. Additionally a 3d-component leaves garbage if it is moved (since it draws outside it's own area).

Some components, like for instance "CUARotor" needs the form's "Clipcontrols" set to False to function correctly.

Most components has a standard size, which is used every time a new is placed on a form. This size is used only when you double click the component in Visual Basic:s tool box, not using common drag and drop placement. Try this, it saves time and effort!

A simple program

...

CUACheck Control

See also Properties Events Methods

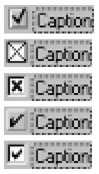
Description

A check box displays an X when selected; the X disappears when the check box is cleared. Use this control to give the user a True/False or Yes/No option. You can use check boxes in groups to display multiple choices from which the user can select one or more. The functionality is the same as the standard *CheckBox*, with some additions. *KeyDown* and all messages concerning drag and drop are passed on to the owning to ease implementation of keyboard support and drag and drop.

This is a bound control, it easily connects to the database engine.

Example

OS/2 CUA Windows 3 Windows 3D Borland BWCC Windows 95



File name

CUACHECK.VBX

Remarks

Check boxes and option buttons function similarly but with an important difference: any number of check boxes on a form can be selected at the same time. In contrast, only one option button in a group can be selected. To display text next to the check box, set the Caption property. Use the Value property to determine the state of the boxselected, cleared, or unavailable.

This custom control is distributed with the CUA/Controls package.
Distribution Note When you create and distribute applications that use the CUACheck control you should install the file CUACHECK.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUARadio

Properties

Alignment	BackColor	Caption
DataChanged	DataField	DataSource
DragIcon	DragMode	Enabled
Font3D	FontBold	FontItalic
FontName	FontSize	FontStrikethru
FontUnderline	ForeColor	Height
HelpContextID	hWnd	Index
Left	MousePointer	Name
Parent	TabIndex	TabStop
Tag	Top	Value
Visible	Width	

Events

Click	DragDrop	DragOver
GotFocus	KeyDown	KeyPress
KeyUp	LostFocus	

Methods

Drag	Move
Refresh	SetFocus
ZOrder	

CUAComboControl

See also Properties Events Methods

Description

A combo box combines the features of a text box and a list box. Use a combo box to give the user the choice of typing in the text box portion or selecting an item from the list portion of the control.
It is compatible with the standard *ComboBox*, with some additions. It can connect with VB:s database engine unlike the original. The list and the text box can be connected to two different sources. The property ListDataSource can be used to automatically fill the list with lines from the database. It is also more visually appealing, no white space can be seen between the box and the button.

A ComboBox is used to let the user select one of several options. Either it can only be selected (such combo boxes have no spacing between the box and button), or the user can type text in the text box. The main advantage with a ComboBox is that it consumes little screen space.

Filling the list

Use the property ListDataSource to fill the list with options from a database. ListDataSource shall in that case name the data control which supplies the lines from the database.

If you do not have many options, the easiest thing is to enter the alternatives in ListDataSource behind a leading equal sign separated with semicolon.(;) (for example "=Apple;Banana;Lemon").

If the list contain multiple columns; for example a code followed by text, you can use the property ListColBound to specify which column should be connected to the text box.

Connecting the combo box to a field in the database

Use the properties DataSource and DataField to connect the combo to a field in a database.

Pictures

The property ListPicture is an array of pictures, one for every line in the combo box. This can not be set in design mode, it has to be done from code.

```
' Add a new line
cboFont.AddItem "Courier New"
' Use 'NewIndex' to assign a picture to the line,
' picTT is a PictureBox. LoadPicture may also be used.
cboFont.ListPicture(cboFont.NewIndex) = picTT
```

File name

CUACOMBO.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note When you create and distribute applications that use the CUACombo control you should install the file CUACOMBO.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAList

Properties

BackColor	BorderStyle	DataChanged
DataField	DataSource	DragIcon
DragMode	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	HelpContextID	hWnd
Index	ItemData	Left
List	ListCount	ListDataSource
ListIndex	ListPicture	MousePointer
Name	NewIndex	Parent
Pattern	SelLength	SelStart
SelText	Sorted	Style
TabIndex	TabStop	Tag
Top	Visible	Width

Events

Change	Click	DblClick
DragDrop	DragOver	DropDown
GotFocus	KeyDown	KeyPress
KeyUp	LostFocus	

Methods

AddItem	Drag	Move
Refresh	RemoveItem	SetFocus
ZOrder		

CUACommand Control

See also   Properties   Events   Methods

Description

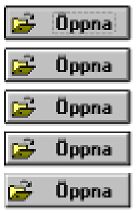
A command button is chosen by the user to begin, interrupt, or end a process.   When chosen, a command button appears pushed in, and is sometimes called a "push button."

To display text on a command button, set its Caption property.   A user can always choose a command button by clicking it.   To allow the user to choose it by pressing Enter, set the Default property to True.   To allow the user to choose the button by pressing Esc, set the button's Cancel property to True.

Corresponds to the standard button, but can take on the looks of different popular UI:s and may also contain Pictures. There are 40 standard pictures build in and available through the property StandardButton. Use the property Picture if you want any other picture.

Example

OS/2 CUA   Windows 3   Windows 3D   Borland BWCC   Windows 95



File name

CUACMD.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note   When you create and distribute applications that use the CUACommand control you should install the file CUACMD.VBX in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above.   Properties and events that apply only to this control, or require special consideration when used with it, are underlined.   They are documented in this help file.   See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUALabel

Properties

BackColor	Cancel	Caption
Default	DragIcon	DragMode
Enabled	Font3D	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	Height
HelpContextID	hWnd	Index
Left	MousePointer	Name
Parent	Picture	PictureAlign
StandardButton	TabIndex	TabStop
Tag	Top	Value
Visible	Width	

Events

Click	DragDrop	DragOver
GotFocus	KeyDown	KeyPress
KeyUp	LostFocus	MouseDown
MouseMove	MouseUp	

Methods

Drag	Move
Refresh	SetFocus
ZOrder	

CUAEdit Control

See also Properties Events Methods

Description

A text box control, sometimes called an "edit field" or "edit control," can display information entered at design time, entered by the user, or assigned to the control in code at run time.

To display multiple lines of text in a text box, set the MultiLine property to True. If a multiline text box does not have a horizontal scroll bar, text in the box wraps automatically even when the text box is resized. To customize the scroll bar combination on a text box, set the ScrollBars property.A text box can also act as a destination link in a DDE conversation.

This is a database component, it can automatically connect to the database engine. Added functionality consist of: messages concerning drag and drop are passed to the owning container to ease drag and drop programming. One often requested property missing from the standard TextBox is ReadOnly.

Readonly fields

ReadOnly fields are automatically shown without frame in white forms, or sunken with gray background in gray forms. This conforms to to standard praxis for graphical interfaces, and clearly signals -- to the user -- whether the field is editable or not.

The textbox still works with Windows clipboard in ReadOnly mode. It can be changed during program execution, for example to reflect a user's authority level. If the property Enabled is set False, it will not be possible to tab to it.

Validation

CUAEdit has extensive support for validation. Allowed data type can be specified via Type (text, integer, float, date, time). If that is not sufficient, input can be controlled on character level via an input Mask. For example, the mask "### ##" is suitable for a Swedish ZIP code. Validation can also be performed when the focus is about to leave the field (Validate). If the content is non-appetizing to *Validate*, CUA/Controls will flash the field with a red background color which is restored to normal at the next keypress.

File name

CUAEDIT.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note When you create and distribute applications that use the CUAEdit control you should install the file CUAEDIT.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUALabel

Properties

BackColor	BorderStyle	DataChanged
DataField	DataSource	DragIcon
DragMode	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	HelpContextID	HideSelection
hWnd	Index	Left
LinkItem	LinkMode	LinkTimeout
LinkTopic	LowerCase	Mask
MaxLength	MousePointer	MultiLine
Name	OEMConvert	PasswordChar
Pattern	ReadOnly	ScrollBars
SelLength	SelStart	SelText
TabIndex	TabStop	Tag
Text	Top	Type
UpperCase	Visible	Width

Events

Change	DragDrop	DragOver
GotFocus	KeyDown	KeyPress
KeyUp	LinkClose	LinkError
LinkNotify	LinkOpen	LostFocus
MouseDown	MouseMove	MouseUp
Validate		

Methods

Drag	LinkExecute	LinkPoke
LinkRequest	Move	Refresh
SetFocus	ZOrder	

CUAFrame Control

See also Properties Events Methods

Description

A frame provides an identifiable grouping for controls. You can also use a frame to subdivide a form functionallyfor example, to separate a group of option buttons from other groups of option buttons.

To group controls, first draw the frame and then draw the controls inside the frame. This enables you to move the frame and controls together. If you draw a control outside the frame and then try to move it inside, the control will be on top of the frame and you'll have to move the frame and controls separately. To select multiple controls in a frame, hold down the Ctrl key while using the mouse to draw a box around the controls.

File name

CUAFRAME.VBX

Remarks

This custom control is distributed with the CUA/Controls package

Distribution Note When you create and distribute applications that use the CUAFrame control you should install the file CUAFRAME.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUALabel

Properties

BackColor	BorderStyle	Caption
ClipControls	DragIcon	DragMode
Enabled	Font3D	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	HelpContextID	hWnd
Index	Left	MousePointer
Name	Pattern	TabIndex
Tag	Top	Visible
Width		

Events

DragDrop	DragOver

Methods

Drag	Move
Refresh	ZOrder

CUAGrid Control

See also   Properties   Events   Methods

Description

Not documented yet.

File name

CUAGRID.VBX

Remarks

This custom control is distributed with the CUA/Controls package.

Distribution Note   When you create and distribute applications that use the CUAGrid control you should install the file CUAGRID.VBX in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above.   Properties and events that apply only to this control, or require special consideration when used with it, are underlined.   They are documented in this help file.   See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAListBox

Properties

BackColor	DragIcon	
DragMode	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	hWnd	Index
Left	MousePointer	Name
Parent	TabIndex	Tag
Top	Visible	Width

Events

Change

Methods

AddItem	Drag	Move
Refresh	RemoveItem	ZOrder

CUAIcon Control

See also Properties Events Methods

Description

This component is designed for the occations where you need a draggable icon with a descriptive text that moves with the picture.

Two properties are more important than the other:
*	Icon to choose picture
*	Caption to set the text.

File name

CUAICON.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note   When you create and distribute applications that use the CUAIcon control you should install the file CUAICON.VBX in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above.   Properties and events that apply only to this control, or require special consideration when used with it, are underlined.   They are documented in this help file.   See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAMeter

Properties

BackColor	BorderStyle	DragIcon
DragMode	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	hWnd	Index
Left	MousePointer	Name
Parent	TabIndex	Tag
Top	Visible	Width

Events

Change

Methods

Drag	Move
Refresh	ZOrder

CUALabel Control

See also Properties Events Methods

Description

A label is a graphical control you can use to display text that the user can't change directly.

You can write code that changes a label in response to events at run time.   For example, if your application takes a few minutes to commit a change, you could display a processing-status message in a label.   You can also use a label to identify a control, such as a text box, that does not have its own Caption property.   Set the AutoSize and WordWrap properties if you want the label to properly display variable-length lines or varying
numbers of lines.   A label can also act as a destination link in a DDE conversation.   To do this, set the LinkTopic property to establish a link, set the LinkItem property to specify an item for the conversation, and set the LinkMode property to activate the link.   When these have been set, Visual Basic attempts to initiate the conversation and displays a message if it is unable to do so.

All messages concerning drag and drop are passed on to the owning container to ease development of drag and drop applications. You can also use three-dimensional effects with CUALabel, the text can appear sunken or raised (besides normal).

File name

CUALABEL.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note   When you create and distribute applications that use the CUALabel control you should install the file CUALABEL.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above.   Properties and events that apply only to this control, or require special consideration when used with it, are underlined.   They are documented in this help file.   See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAEdit

Properties

Alignment	AutoSize	BackColor
BackStyle	BorderStyle	Caption
DataField	DataSource	DragIcon
DragMode	Enabled	Font3D
FontBold	FontItalic	FontName
FontSize	FontStrikethru	FontUnderline
ForeColor	Height	Index
Left	LinkItem	LinkMode
LinkTimeout	LinkTopic	MousePointer
Name	Pattern	TabIndex
Tag	Top	Visible
Width	WordWrap	

Events

Change	Click	DblClick
DragDrop	DragOver	LinkClose
LinkError	LinkNotify	LinkOpen
MouseDown	MouseMove	MouseUp

Methods

Drag	LinkExecute	LinkPoke
LinkRequest	Move	Refresh
ZOrder		

CUALamp Control

See also Properties Events Methods

Description

The lamp control can be used as an alternative to a meter; to indicate work in progress. It can take on four different styles, accessed through the Style property. Each style supports four values: No light, Green, Yellow and Red.

File name

CUALAMP.VBX

Remarks

This custom control is distributed with the CUA/Controls package.

Distribution Note When you create and distribute applications that use the CUALamp control you should install the file CUALAMP.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAMeter

Properties

BackColor	DragIcon	DragMode
Enabled	FontBold	FontItalic
FontName	FontSize	FontStrikethru
FontUnderline	ForeColor	Height
hWnd	Index	Left
MousePointer	Name	Parent
Style	TabIndex	Tag
Top	Value	Visible
Width		

Events

Change

Methods

Drag	Move
Refresh	ZOrder

CUAList Control

See also Properties Events Methods

Description

This control is compatible with VB:s *ListBox*, with some additions. For example; each line may contain a picture. It can also be bound to a database.

Filling the list box

Use the property ListDataSource to fill the list with alternatives from a database. ListDataSource should contain the name of the data control that supplies the lines from the database.

You may also type the alternatives directly into ListDataSource, if you enter an equal sign first and separate the fields with a semicolon ("=Car;Lorry;Train").

Even columns using proportional fonts

When the standard font "MS Sans Serif" -- a proportional typeface -- is used with data in multiple columns, the result is often less than satisfying.
To cure this; begin by separeting the columns with a tab character (Chr$(9)), not spaces. Then use the property ColWidth to control exactly how wide each column should be. Use only numbers, and separate each number with a comma or semicolon.

```
' Set column widths
lstList.ColWidth = "5,15,25"
' Add a new line to the list
lstList.AddItem "1234" & Chr$(9) & "Name" & Chr$(9) & "Address"
```

Connecting to a database

The properties DataSource and DataField are used to connect the listbox to a field in the database.

Pictures

The property ListPicture is an array of pictures, one for every line in the list. This cannot be set in design mode, it must be done from code.

```
' Add a new line
lstFont.AddItem "Courier New"
' Use 'NewIndex' to assign the new line a picture,
' picTT is a PictureBox. LoadPicture can also be used.
lstFont.ListPicture(lstFont.NewIndex) = picTT
```

File name

CUALIST.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note When you create and distribute applications that use the CUAList control you should install the file CUALIST.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUACombo

Properties

BackColor	BorderStyle	Columns
ColWidth	DataChanged	DataField
DataSource	DragIcon	DragMode
Enabled	FontBold	FontItalic
FontName	FontSize	FontStrikethru
FontUnderline	ForeColor	Height
HelpContextID	hWnd	Index
ItemData	Left	List
ListCount	ListDataSource	ListIndex
ListPicture	MousePointer	MultiSelect
Name	NewIndex	Parent
Pattern	Selected	Sorted
TabIndex	TabStop	Tag
Text	Top	TopIndex
Visible	Width	

Events

Click	DblClick	DragDrop
DragOver	GotFocus	KeyDown
KeyPress	KeyUp	LostFocus
MouseDown	MouseMove	MouseUp

Methods

AddItem	Drag	Move
Refresh	RemoveItem	SetFocus
ZOrder		

CUAMeter Control

See also Properties Events Methods

Description

A horizontal or vertical gauge that is often used to give feedback about some process that will take some time to complete. The official recommendation is that a progress indicator should be used if something takes more than five seconds to complete. Otherwise the hour glass is a viable alternative for processes taking more than two seconds to finish. Remember these are rough estimates and times vary significantly depending on how fast the computer is.

Use the property Orientation to set up a horizontal or vertical meter.

The properties Min, Max and Value determines how the meter will be drawn.

To display the percentage rate, set the Percent property to True.

If you want a segmented look to the meter, set the Segmented value to True.

If you want a 3D look to the meter, experiment with the Meter3D and Pattern properties.

The following code snippet will drive the meter from it's min value to max:

```
For I% = mtr1.Min To mtr1.Max
   mtr1.Value = I%
Next
```

File name

CUAMETER.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note When you create and distribute applications that use the CUAMeter control you should install the file CUAMETER.VBX in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUASlide
CUARotor

Properties

BackColor	BorderStyle	DragIcon
DragMode	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	hWnd	Index
Left	Max	Meter3D
Min	MousePointer	Name
Orientation	Parent	Pattern
Percent	Segmented	TabIndex
Tag	Top	Value
Visible	Width	

Events

Change

Methods

Drag	Move
Refresh	ZOrder

CUANoteBook Control

See also Properties Events Methods

Description

The intention is to give the impression of a notebook with multiple pages that can be seen one at a time. This reduces screen clutter, operations that otherwise require multiple windows can be simplified to one single notebook.

Details

The CUA notebook is a flexible control that requires a small amount of coding to implement the pages in the control. When you first place a notebook on a form, it is empty. To add pages to the dialog, place one CUAStatus control per page within the notebook. Organize the pages in a control array with a name such as "stsPage". Then set the property TabCount to the number of pages. Captions of the individual tabs are set through the **TabCaption** property, and the color of individual tabs may be set with **TabColor**.

The notebook can be implemented in one of two ways:

1. Manually, by handling the event TurnPage as the user clicks a tab.
   This is the most flexible way, but also designwise more complicated (and it also requires you to monitor the "Resize" event to adjust the size of the pages).

```
Sub bok1_TurnPage (ToPage As Integer)
   ' User wants to turn page, first hide the current page ...
   stsPage(bok1.Tab).Visible = False
   ' ... then show the requested page
   stsPage(ToPage).Visible = True
End Sub
```

2. Automatically, by giving page one the name **page1**, page two **page2** etc. This way, pages will be automatically positioned, sized and shown/hidden and you will be able to switch between pages in design mode using the right mouse button.

File name

CUABOOK.VBX

Remarks

This custom control is distributed with the CUA/Controls package
Distribution Note When you create and distribute applications that use the CUABook control you should install the file CUABOOK.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

Tabbed dialog

Properties

BackColor	Caption	ClipControls
DragIcon	DragMode	Enabled
Font3D	FontBold	FontItalic
FontName	FontSize	FontStrikethru
FontUnderline	Height	HelpContextID
Index	Left	MousePointer
Name	TabCountTab	
Pattern	Spiral	Stacked
TabIndex	TabStop	Tag
Top	Visible	Width

Events

Click	DragDrop	DragOver
MouseDown	MouseMove	MouseUp
TurnPage		

Methods

Refresh	Move	Drag
SetFocus	ZOrder	

CUARadio Control

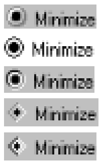
See also Properties Events Methods

Description

An option button displays an option that can be turned on or off.

Usually, option buttons are used as part of an option group to display multiple choices from which the user can select only one. You can group option buttons by drawing them inside a frame, a picture box, or directly on a form. To group option buttons in a frame or picture box, draw the frame or picture box first, and then draw the option buttons inside. All option buttons within a frame or picture box are treated as a group. Option buttons on a form are also treated as a separate group from any option buttons in a frame or picture box. While option buttons and check boxes may appear to function similarly, there is an important difference: when a user selects an option button, the other option buttons in the same group are automatically cancelled. In contrast, any number of check boxes can be selected.

OS/2 CUA Windows 3 Windows 3D Borland BWCC Windows 95



File name

CUARADIO.VBX

Remarks

This custom control is distributed with the CUA/Controls package.

Distribution Note When you create and distribute applications that use the CUARadio control you should install the file CUARADIO.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUACheck

Properties

Alignment	BackColor	Caption
DragIcon	DragMode	Enabled
Font3D	FontBold	FontItalic
FontName	FontSize	FontStrikethru
FontUnderline	ForeColor	Height
HelpContextID	hWnd	Index
Left	MousePointer	Name
Parent	TabIndex	TabStop
Tag	Top	Value
Visible	Width	

Events

Click	DragDrop	DragOver
GotFocus	KeyDown	KeyPress
KeyUp	LostFocus	

Methods

Drag	Move
Refresh	SetFocus
ZOrder	

CUARotor Control

See also   Properties   Events   Methods

Description

A rotating knob which behaves like it's real life counterpart. A "round slider".

File name

CUAROTOR.VBX

Remarks

This custom control is distributed with the CUA/Controls package.

Distribution Note   When you create and distribute applications that use the CUARotor control you should install the file CUAROTOR.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above.   Properties and events that apply only to this control, or require special consideration when used with it, are underlined.   They are documented in this help file.   See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUASlider
CUAMeter

Properties

DragIcon	DragMode	Enabled
FontBold	FontItalic	FontName
FontSize	FontStrikethru	FontUnderline
ForeColor	HelpContextID	hWnd
Index	Left	Max
Min	MousePointer	Name
Parent	TabIndex	TabStop
Tag	Top	Value
Visible		

Events

Change

Methods

Drag	Move
Refresh	ZOrder

CUASlide Control

See also Properties Events Methods

Description

This control allows the user to create a slider on a form.

File name

CUASLIDE.VBX

Remarks

This custom control is distributed with the CUA/Controls package

Distribution Note When you create and distribute applications that use the CUASlide control you should install the file CUASLIDE.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

Meter
Rotor

Properties

BackColor	Min	Visible
DragIcon	MousePointer	
DragMode	Name	
Enabled	Orientation	
ForeColor	TabIndex	
Height	TabStop	
HelpContextID	Tag	
Index	Top	
Left	Value	
Max	Width	

Events

Change

Methods

Refresh	Move
Drag	ZOrder

CUASpin Control

See also Properties Events Methods

Description

Basically there are two ways to manage the CUASpin control:

For simple numeric values, the best way is to use the 'automatic' mode. In this case, you must give a value for the Buddy property. With this mode, if the user give an invalid value in the text control, the spin doesn' t perform any action.

The second way is to handle CUASpin notifications message and do it yourself. In this case, you must give values for Min, Max and you must update the Value property each time the text control is updated.

File name

CUASPIN.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note When you create and distribute applications that use the CUASpin control you should install the file CUASPIN.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAEdit

Properties

Buddy	DataChanged	DataField
DataSource	DragIcon	DragMode
Enabled	Height	HelpContextID
hWnd	Index	Left
Max	Min	MousePointer
Name	Orientation	Parent
TabIndex	Tag	Top
Value	Visible	Width
Wrap		

Events

Change	DragDrop	DragOver

Methods

Drag	Move
Refresh	ZOrder

CUAStatus Control

See also Properties Events Methods

Description

A status bar is common in Windows applications, used to give the user information and short help texts. The CUAStatus provides for six common status fields through it's StandardStatus property.

File name

CUASTAT.VBX

Remarks

This custom control is distributed with the CUA/Controls package.

Distribution Note When you create and distribute applications that use the CUAStatus control you should install the file CUASTAT.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAMeter

Properties

BackColor	BorderStyle	DragIcon
DragMode	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	hWnd	Index
Left	MousePointer	Name
Parent	StandardStatus	
TabIndex	Tag	Top
Visible	Width	

Events

Change

Methods

Drag	Move
Refresh	ZOrder

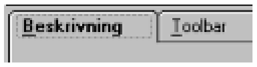
CUATabs Control

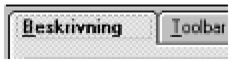
See also Properties Events Methods

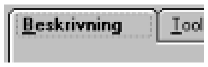
Description

The intention is to give the impression of a window with multiple pages that can be seen one at a time. This reduces screen clutter, operations that otherwise require multiple windows can be simplified to one single "tabbed dialog". Depending on configuration (see ConfigCUA), this control take on these different looks:

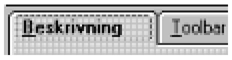
OS/2 CUA Windows 3 Windows 3D

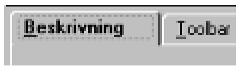






Borland BWCC Windows 4





Details

The CUA tabbed dialog is a flexible control that requires a small amount of coding to implement the pages in the control. When you first place a tabbed dialog on a form, it is empty. To add pages to the dialog, place one CUAStatus control per page within the tabbed dialog. Organize the pages in a control array with a name such as "stsPage". Then set the property TabCount to the number of pages. Captions of the individual tabs are set through the **TabCaption** property, and the color of individual tabs may be set with **TabColor**.

The tabbed dialog can be implemented in one of two ways:

1. Manually, by handling the event TurnPage as the user clicks a tab.
   This is the most flexible way, but also designwise more complicated (and it also requires you to monitor the "Resize" event to adjust the size of the pages).

```
Sub tab1_TurnPage (ToPage As Integer)
   ' User wants to turn page, first hide the current page ...
   stsPage(tab1.Tab).Visible = False
   ' ... then show the requested page
   stsPage(ToPage).Visible = True
End Sub
```

2. Automatically, by giving page one the name **page1**, page two **page2** etc. This way, pages will be automatically positioned, sized and shown/hidden and you will be able to switch between pages in design mode using the right mouse button.

File name

CUATABS.VBX

Remarks

This custom control is distributed with the CUA/Controls package
Distribution Note When you create and distribute applications that use the CUATabs control you should install the file CUATABS.VBX in the customer's Microsoft Windows \

SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

NoteBook

Properties

ActiveTabBold	BackColor	ClipControls
DragIcon	DragMode	Enabled
Font3D	FontBold	FontItalic
FontName	FontSize	FontStrikethru
FontUnderline	Height	HelpContextID
Index	Left	MousePointer
Name	Spiral	Stacked
Tab	TabCaption	TabColor
TabCount	TabGroup	TabIndex
TabStop	TabWidth	Tag
Top	Visible	Width

Events

Click	DragDrop	DragOver
MouseDown	MouseMove	MouseUp
TurnPage		

Methods

Refresh	Move	Drag
SetFocus	ZOrder	

CUATermo Control

See also Properties Events Methods

Description

This is special meter control in the form of a thermometer. It differs from the meter in that it's value can fluctuate up and down, and that it has a warning area.

File name

CUATERMO.VBX

Remarks

This custom control is distributed with the CUA/Controls package.

Distribution Note When you create and distribute applications that use the CUATermo control you should install the file CUATERMO.VBX in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAMeter
CUASlide

Properties

BackColor	DragIcon	DragMode
Enabled	FontBold	FontItalic
FontName	FontSize	FontStrikethru
FontUnderline	ForeColor	Height
High	hWnd	Index
Left	Max	Min
MousePointer	Low	Name
Parent	TabIndex	Tag
Top	Value	Visible
WarnHigh	WarnLow	Width

Events

Change

Methods

Drag	Move
Refresh	ZOrder

CUAToolBox Control

See also Properties Events Methods

Description

A toolbox consist of a "topmost" floating window populated with buttons. Each button usually represents a different "tool" with a picture, and only one "tool" can be selected at a time.

File name

CUATOLBX.VBX

Remarks

This custom control is distributed with the CUA/Controls package.

Distribution Note When you create and distribute applications that use the CUAToolBox control you should install the file CUATOLBX.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAToolButton

Properties

BackColor	DragIcon	
DragMode	Enabled	FontBold
FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor
Height	hWnd	Index
Left	MousePointer	Name
Parent		
TabIndex	Tag	Top
Visible	Width	

Events

Change

Methods

AddItem	Drag	Move
Refresh	RemoveItem	ZOrder

CUAToolButton Control

See also Properties Events Methods

Description

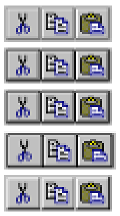
Toolbars are very popular in todays modern Windows applications. The idea is to collect the most common commands for quick access. This is faster and more efficient than menues, since the menu has to be pulled down. On the downside, a toolbar consume some screen space. It is common practise to let the user decide for himself whether to use the toolbar or not.

A toolbar is populated with tool buttons, which this component supplies.There are 40 standard pictures built in through the property StandardButton, which is both preactical and resource preserving. If you are not satisfied with any of the standard pictures, you can design and use your own with the property Picture, or the properties ButtonSource=Custom with CustomButton and CustomCount set to indicate which picture should be used.

A button can be either a command button or a toggle button. Command buttons do not stay down, while toggle buttons stay up or down for each click with the mouse.

You only have to use one single image per button. CUA/Controls will derive different looks (normal, attribute, pressed, disabled) from this single image. If you want to draw your own images, you should probably use the bitmap size 16*15 pixels (if property StandardSize is True) or a different size of your choice if StandardSize is set False.

OS/2 CUA Windows 3 Windows 3D Borland BWCC Windows 95



Do this to implement a tool bar in your own application:

1. Place a **CUAStatus** in the window, to be used as the toolbar.
2. Set **CUAStatus.Align=Top** for standard appearance.
3. Place several **CUAToolButton** on the toolbar, one for each command.
4. Use **StandardButton** if any of the 40 standard images is suitable.
5. Enter code to handle **Click** event when the user clicks the button.

File name

CUATOLBN.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note When you create and distribute applications that use the CUAToolButton control you should install the file CUATOLBN.VBX in the customer's Microsoft Windows \SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above. Properties and events that apply only to this control, or require special consideration when used with it, are underlined. They are documented in this help file. See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUACommand

Properties

AutoSize	ButtonSource	ButtonType
CustomButton	CustomCount	DragIcon
DragMode	Enabled	Height
hWnd	Index	Left
MousePointer	Name	Parent
Picture	StandardButton	TabIndex
Tag	Top	Value
Visible	Width	

Events

Click	DragDrop	DragOver
MouseDown	MouseMove	MouseUp

Methods

Drag	Move
Refresh	ZOrder

CUAToolTips Control

See also Properties Events Methods

Description

Tool tips are the small, yellow windows that can be seen in modern applications, supplying brief help texts when the cursor rests over a tool button. This can be done on any control, but is especially useful with tool buttons, since they are som small that it is often hard to interpret the pictures if you are not used to the application.

You only have to use one **CUAToolTips** per window, it can be used to show help on every component that resides in that window. During execution, the event NeedText is activated whent **CUAToolTips** needs the text. Enter your own code in this event handler to return your desired tool tips help text.

Do this to implement tool tips:

1. Place a **CUAToolTips** in the window.
2. Enter code in the event **NeedText** to supply the texts.

File name

CUATOLTP.VBX

Remarks

This custom control is distributed with the CUA/Controls package.
Distribution Note   When you create and distribute applications that use the CUAToolTips control you should install the file CUATOLTP.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory. The Visual Basic Setup Kit included with the Professional VB product provides tools to help you write setup programs that install your applications correctly.

All of the properties, events, and methods for this control are listed above.   Properties and events that apply only to this control, or require special consideration when used with it, are underlined.   They are documented in this help file.   See the Visual Basic *Language Reference* or online Help for documentation of the remaining properties, events, and methods.

See Also

CUAToolButton

Properties

BackColor	Caption	Enabled
FontBold	FontItalic	FontName
FontSize	FontStrikethru	FontUnderline
Index	Left	Name
Parent	TabIndex	Tag
Top	Visible	Width

Events

NeedText

Methods

Move	Refresh

ActiveTabBold Property

See also

Applies To

Tabbed Dialog

Description

Decides if the current tab's text is shown in bold face.

Usage

[*form*.]*control*.**ActiveTabBold**[= *setting* %]

Remarks

By default, ActiveTabBold is set to False.

Data Type

Boolean

Alignment Property

See also

Applies To

CUACheck, CUARadio

Description

Controls if the text is shown to the left or right of the radio button / check box.

Usage

[*form*.]*control*.**Alignment**[= *setting* %]

Setting

The BorderStyle Property settings are:

Setting	Description
0	(Default) Right.
1	Left.

Remarks

By default, Alignment is set to 0 (Right).

Data Type

Integer (Enumerated)

AutoSize Property

See also

Applies To

Label, ToolButton

Description

Determines if the ToolButton should take on a standard size.

Usage

[*form*.]*control*.**AutoSize**[= *setting* %]

Setting

The AutoSize Property settings are:

Setting	Description
0	False, You may size the button freely.
1	Standard size 26x25 pixels.

Remarks

By default, Alignment   is set to 1 (True).

Data Type

Integer (Enumerated)

BorderStyle Property

See also

Applies To

CUACombo, CUAFrame, CUALabel, CUAList, CUAStatus

Description

Determines what kind of border the control has.

Usage

[*form*.]*control*.**BorderStyle**[= *setting* %]

Setting

The BorderStyle Property settings are:

Setting	Description
0	No frame.
1	(Default) Single border.
2	Shadowed border.
3	Sunken.
4	Raised.
5	Dip.
6	Bump.
7	Sunken (thin).
8	Raised (thin).

Data Type

Integer (Enumerated)

Buddy Property

See also

Applies To

Spin Button

Description

Determines if the spin button has an automated connection to a related text box. If Buddy is set to the name of a numeric text box on the same form, the spin button will automatically adjust the contents of the associated text box.

Usage

[*form*.]*control*.**Buddy**[= *name$*]

Setting

Data Type

Integer (Enumerated)

ButtonSource Property

See also

Applies To

ToolButton

Description

Determines if the graphics in the button are from the standard or a custom source.

Usage

[*form*.]*control*.**ButtonSource**[= *setting* %]

Setting

The ButtonSource Property settings are:

Setting	Description
0	Standard
1	Custom

Data Type

Integer (Enumerated)

ButtonType Property

See also

Applies To

ToolButton

Description

Determines if the ToolButton is a Command button or a Toggle button.

Usage

[*form*.]*control*.**ButtonType**[= *setting* %]

Setting

The ButtonType Property settings are:

Setting	Description
0	Command
1	Toggle (stays down)

Data Type

Integer (Enumerated)

Cancel Property

See also

Applies To

Command button

Description

Determines whether a command button is the Cancel button on a form.

Usage

[*form*.]*control*.**Cancel**[= *boolean*]

Setting

The Cancel property settings are:

Setting	Description
True	The command button is the Cancel button.
False	(Default) The command button isn't the Cancel button.

Remarks

Use the Cancel property to give a user the option of canceling uncommitted changes and returning the form to its previous state.

Only one command button on a form can be the Cancel button. When the Cancel property is set to True for one command button, it is automatically set to False for all other command buttons on the form. When a command button's Cancel property is True and the form is the active form, the user can choose the command button by clicking it, pressing the Esc key, or pressing Enter when the button has the focus.

Tip For a form that supports irreversible operations, such as deletions, it's a good idea to make the Cancel button the default button. To do this, set both the Cancel property and the Default property to True.

Data Type

Boolean

ColWidth Property

See also

Applies To

List Box

Description

Determines width of each column in a list box. Columns are separated by tab characters (Ascii code 9). Enter each column width expressed in number of characters, separated with a comma.

Usage

[*form*.]*control*.**ColWidth**[= *setting* $]

Remarks

If this property is left empty, columns will be seven characters wide. If you supply only one number, all columns will be given that width. If you supply several numbers separated by commas you can control the width of each and every column. The average character width as reported by Windows API GetTextMetrics will be used to determine column widths for proportional fonts.

Data Type

String

CustomButton Property

See also

Applies To

ToolButton

Description

Index into custom bitmap given by the property Picture.

Usage

[*form*.]*control*.**CustomButton**[= *setting* %]

Remarks

Indexes start at zero, and go up to CustomCount - 1.

Data Type

Integer (Enumerated)

CustomCount Property

See also

Applies To

ToolButton

Description

Determines how many graphic slices there are in the Picture property.

Usage

[*form*.]*control*.**CustomCount**[= *setting* %]

Data Type

Integer (Enumerated)

Default Property

See also

Applies To

Command button

Description

Determines whether a command button is the Default button on a form.

Usage

[*form*.]*control*.**Default**[= *boolean*]

Setting

The Default property settings are:

Setting	Description
True	The command button is the default button. Only one button on a form can be the default button. When Default is set to True for one button, it is automatically set to False for all other buttons on the form. When the button's Default property is True and its parent form is active, the user can choose the button (invoking its Click event) by pressing the Enter key. Any other control with the focus does not receive a keyboard event (KeyDown, KeyPress, or KeyUp) for the Enter key unless the user has moved the focus to another command button on the same form. In this case, pressing Enter chooses that button instead of the default button.
False	(Default) The command button is not the default button.

Remarks

For a form or dialog box that supports an irreversible action such as a delete operation, make the Cancel button the default button by setting its Default property to True.

Data Type

Boolean

Font3D Property

See also

Applies To

Command, Frame, Label

Description

Affects three-dimensional effects in a component's text.

Usage

[*form*.]*control*.**Font3D**[= *setting* %]

Setting

The Font3D Property settings are:

Setting	**Description**
0	(Default) Normal.
1	Sunken.
2	Raised.

Data Type

Integer (Enumerated)

High Property

See also

Applies To

Termometer

Description

Indicates the highest value reached for the termometer.

Usage

[*form*.]*control*.**High**[= *setting* %]

Remarks

This property is read only.

Data Type

Integer

ListColBound Property

See also

Applies To

Combo box

Description

Determines which of the columns - in a multi column list - that represents the value of the control.

Usage

[*form*.]*control*.**ListColBound**[= *integer%*]

Remarks

Multiple columns can be shown in a list; for example a code in one column and an explanation in another. ListColBound determines which of the columns that is used in the text portion of the combo box.

Data Type

Integer

ListDataSource Property

See also

Applies To

Combo box, List box

Description

Used for automatic filling of the list.

Usage

[*form*.]*control*.**ListDataSource**[= *string$*]

Remarks

If items shall be fetched from a database, this field shall contain the name of the supplying data control.
Can also contain the values directly; in that case the field is started with an equal sign followed by the values separated with semicolon (t ex =Apples;Bananas;Oranges).

Data Type

String

ListPicture Property

See also

Applies To

Combo box, List box, Tool box

Description

Determines which graphic to be displayed on a selected line in the control.

Usage

[*form*.]*control*.**ListPicture(Index)**[= *picture*]

Setting

The picture property settings are:

Setting	Description
(none)	(Default) No picture.
(Bitmap, icon, metafile)	Specifies a graphic. You can load the graphic from the Properties window at design time. At run time, you can also set this property using the LoadPicture function on a bitmap, icon, or metafile.

Remarks

At design time, you can transfer a graphic with the Clipboard using the Copy, Cut, and Paste commands on the Edit menu. At run time, you can use Clipboard methods such as GetData, SetData, and GetFormat with the non-text Clipboard formats CF_BITMAP, CF_METAFILE, and CF_DIB, as defined in CONSTANT.TXT, a Visual Basic file that specifies system defaults.
When setting the Picture property at design time, the graphic is saved and loaded with the form. If you create an executable file, the file contains the image. When you load a graphic at run time, the graphic is not saved with the application. Use the SavePicture statement to save a graphic from a form or picture box into a file.

Data Type

Integer

Low Property

See also

Applies To

Termometer

Description

Determines the lowest value reached for the termometer.

Usage

[*form*.]*control*.**Low**[= *setting* %]

Remarks

This value is read only.

Data Type

Integer

LowerCase Property

See also

Applies To

Edit box

Description

Determines if characters are forced to lower case.

Usage

[*form*.]*control*.**LowerCase**[= *picture*]

Data Type

Boolean

Mask Property

See also

Applies To

Edit box

Description

Controls individual characters entered into an edit box. The following standard, predefined input masks are available at design time.

Mask	Description
Null String	(Default) No mask. Acts like a standard text box.
(###) ###-####	Standard North American Phone number.
(###) ###-#### Ext(#####)	Standard North American Phone number with extension.
###-##-####	US Social Security Number.
##-???-##	US Medium date. Example: 20-May-92
##-##-##	US Short date. Example: 05-20-92
##:## ??	Medium time. Example: 05:36 AM
##:##	Short time. Example: 17:36
### ##	Swedish Zip Code.

Setting

The mask property characters are:

Mask character	Description
#	Digit placeholder.
.	Decimal placeholder
,	Thousands separator
:	Time separator
/	Date separator
\	Treat the next character in the mask as a literal. This allows you to include the '#', '&', 'A' and '?' characters in the mask.
&	Character placeholder. Valid values for this placeholder are ANSI characters in the following ranges: 32-126 and 128-255.
A	Alphanumeric character placeholder. For example a-z, A-Z, 0-9.
?	Letter placeholder, for example: a-z, A-Z.
Literal	All other symbols are displayed as literals; that is, as themselves.

Usage

[*form*.]*control*.**Mask**[= *string* $]

Data Type

String

Max Property

See also

Applies To

Meter, Slider, Spin, Termometer

Description

Contains the maximum value allowed for the control.

Usage

[*form*.]*control*.**Max**[= *setting* %]

Remarks

The Max Property setting cannot be lower than the Min Property.
By default, Max is set to 100.

Data Type

Integer

Meter3D Property

See also

Applies To

Meter

Description

Determines if 3D effects should be given the filled meter.

Usage

[*form*.]*control*.**Meter3D**[= *setting* %]

Remarks

By default, Max is set to False.

Data Type

Boolean

Min Property

See also

Applies To

Meter, Slider, Spin, Termometer

Description

Contains the minimum value allowed for the control.

Usage

[*form*.]*control*.**Min**[= *setting* %]

Setting

The Min property is not allowed to exceed the value of the Max property.
By default, Min is set to 0.

Data Type

Integer

OEMConvert Property

See also

Applies To

Edit box

Description

Determines if characters are converted between DOS/Windows character sets.

Usage

[*form*.]*control*.**OEMConvert**[= *boolean*]

Data Type

Boolean

Orientation Property

See also

Applies To

Meter, Slider, Spin

Description

Sets a horizontal or vertical orientation for the control.

Usage

[*form*.]*control*.**Orientation**[= *setting* %]

Setting

The Orientation Property settings are:

Setting	Description
0	Horizontal.
1	Vertical.

Data Type

Integer (Enumerated)

Pattern Property

See also

Applies To

Combo, Frame, Label, List

Description

Determines the pattern used to fill the control.

Usage

[*form*.]*control*.**Pattern**[= *setting%*]

Settings

The Pattern Property settings are:

Setting	Description
0	No pattern.
1	Borland "chiseled steel".
2	25% filling.
3	50% filling.
4	75% filling.

Data Type

Integer

Percent Property

See also

Applies To

Meter

Description

Determines if a percentage text should be shown in the meter's centre.

Usage

[*form*.]*control*.**Percent**[= *boolean*]

Data Type

Boolean

Picture Property

See also

Applies To

Command, ToolButton

Description

Determines a graphic to be displayed in the control.

Usage

[*form*.]*control*.**Picture**[= *picture*]

Setting

The picture property settings are:

Setting	Description
(none)	(Default) No picture.
(Bitmap, icon, metafile)	Specifies a graphic. You can load the graphic from the Properties window at design time. At run time, you can also set this property using the LoadPicture function on a bitmap, icon, or metafile.

Remarks

At design time, you can transfer a graphic with the Clipboard using the Copy, Cut, and Paste commands on the Edit menu. At run time, you can use Clipboard methods such as GetData, SetData, and GetFormat with the non-text Clipboard formats CF_BITMAP, CF_METAFILE, and CF_DIB, as defined in CONSTANT.TXT, a Visual Basic file that specifies system defaults.
When setting the Picture property at design time, the graphic is saved and loaded with the form. If you create an executable file, the file contains the image. When you load a graphic at run time, the graphic is not saved with the application. Use the SavePicture statement to save a graphic from a form or picture box into a file.

Data Type

Integer

PictureAlign Property

See also

Applies To

Command

Description

Determines where a graphic is to be displayed, in relation to button text.

Usage

[*form*.]*control*.**PictureAlign**[= *setting* %]

Setting

The PictureAlign property settings are:

Setting	Description
0	(Default) Vertically centered to the left of button text.
1	Horizontally centered above the button text.

Data Type

Integer (Enumerated)

ReadOnly Property

See also

Applies To

Edit box

Description

Determines if editing is allowed in the control.

Usage

[*form*.]*control*.**ReadOnly**[= *boolean*]

Remarks

If ReadOnly is True, the control will automatically signal this to the user by changing it's appearance. It will be sunken with a gray background in gray windows, else it will have no visual frame.

Data Type

Boolean

Segmented Property

See also

Applies To

Meter

Description

Determines if the filled part of the meter should be segmented.

Usage

[*form*.]*control*.**Segmented**[= *boolean*]

Data Type

Boolean

Spiral Property

See also

Applies To

NoteBook, Tabs

Description

Determines if the control should have a spiral binder, like a real life notebook .

Usage

[*form*.]*control*.**Spiral**[= *boolean*]

Remarks

By default, Spiral is set to False.

Data Type

Boolean

Stacked Property

See also

Applies To

NoteBook, Tabs

Description

Determines if the control border has the look of several stacked papers.

Usage

[*form*.]*control*.**Stacked**[= *boolean*]

Remarks

By default, Stacked is set to False for Tabs but True for a NoteBook.

Data Type

Boolean

StandardButton Property

See also

Applies To

Command, ToolButton

Description

There are 40 standard pictures built into CUA/Controls. This property determines which of the standard graphics that should be shown in the control. Additional bitmaps can be found in the \CUACTLS\BMP subdirectory.

Usage

[*form*.]*control*.**StandardButton**[= *setting* %]

Setting	Description
0	(Default) Blank.
1	OK
2	Cancel
3	Help
4	New
5	Open
6	Save
7	Print
8	Exit
9	Cut
10	Copy
11	Paste
12	Undo
13	Context help
14	First
15	Previous
16	Next
17	Last
18	New Record
19	Ascending
20	Descending
21	Find
22	Find Next
23	Replace
24	Zoom
25	Look
26	Mail
27	Palette
28	Stop
29	Back
30	Drop
31	Index

32	Book
33	Lock
34	Margin Text
35	Left Text
36	Centered Text
37	Right Text
38	Bold
39	Italic
40	Underline

Data Type

Integer (Enumerated)

StandardStatus Property

See also

Applies To

Status Bar

Description

There are 6 status items built into CUA/Controls. This property determines which of the standard items should be shown in the control.

Usage

[*form*.]*control*.**StandardStatus**[= *setting* %]

Setting

The type property may have one of the following settings:

Setting	Description
0	(Default) Blank.
1	Ins
2	Caps
3	Num
4	Scroll
5	Date
6	Time

Data Type

Integer (Enumerated)

Style Property

See also

Applies To

Lamp

Description

Four different sizes of lamps may be chosen.

Usage

[*form*.]*control*.**Style**[= *setting* %]

Setting

The Style Property settings are:

Setting	Description
0	Small lamp.
1	Standard lamp.
2	Small traffic light.
3	Traffic light.

Tab Property

See also

Applies To

NoteBook, Tabs

Description

Contains the current page number in a notebook or tabbed dialog. Read/write, i e the current page number can be examined, as well as set - which will generate a TurnPage event.

Usage

[*form*.]*control*.**Tab**[= *setting* %]

Remarks

This value is zero-based, by default it is set to 0 (the first page).

Data Type

Integer

TabCaption Property

See also

Applies To

NoteBook, Tabs

Description

Contains the caption of the current tab.

Usage

[*form*.]*control*.**TabCaption**[= *setting$*]

Remarks

By default, TabCount is set to "Tab " + the tab number.

Data Type

String

TabColor Property

See also

Applies To

NoteBook, Tabs

Description

Contains the background color that should be used to paint the current tab.

Usage

[*form*.]*control*.**TabColor**[= *setting* &]

Remarks

By default, TabColor is set to 0 (black), which means it will use standard colors.

Data Type

Long

TabCount Property

See also

Applies To

NoteBook, Tabs

Description

Contains the number of pages in a notebook or tabbed dialog.

Usage

[*form*.]*control*.**TabCount**[= *setting* %]

Remarks

By default, TabCount is set to 2.

Data Type

Integer

TabGroup Property

See also

Applies To

Tabbed Dialog

Description

Determines where a new line of tabs starts. If True, a new line of tabs starts from here.

Usage

[*form*.]*control*.**TabGroup**[= *setting* %]

Remarks

By default, TabGroup is set to False.

Data Type

Integer

TabWidth Property

See also

Applies To

Tabbed Dialog

Description

Contains the width of the current tab.

Usage

[*form*.]*control*.**TabWidth**[= *setting* %]

Remarks

If set to zero, it will be automatically sized.

Data Type

Integer

Type Property

See also

Applies To

Edit Box

Description

Determines what kind of data is allowed in the Edit Box.

Usage

[*form*.]*control*.**Type**[= *setting* %]

Setting

The type property may have one of the following settings:

Setting	Description
0	(Default) Character.
1	Integer
2	Float.
3	Date.

Data Type

Integer (Enumerated)

UpperCase Property

See also

Applies To

Edit box

Description

Determines if characters are forced to upper case.

Usage

[*form*.]*control*.**UpperCase**[= *picture*]

Data Type

Boolean

Value Property

See also

Applies To

Check, Lamp, Meter, Radio, Rotor, Slider, Spin, Termometer, ToolButton

Description

Contains the current value of the control.

Usage

[*form*.]*control*.**Value**[= *setting* %]

Setting

This property has somewhat different meanings depending on the control it applies to.

Check Box and Attributed ToolButton:

Setting	Description
0	Unchecked.
1	Checked
2	Indeterminate.

Radio Button and Command ToolButton:

Setting	Description
0	Unchecked.
1	Checked

Meter, Rotor, Slider, Spin: Value is always in the range specified of the Min and Max property.

Data Type

Integer (Enumerated for Check, Radio and ToolButton)

WarnHigh Property

See also

Applies To

Termometer

Description

Determines where the warning area ends in the termometer.

Usage

[*form*.]*control*.**WarnHigh**[= *setting* %]

Remarks

By default, WarnHigh is set to 0.

Data Type

Integer

WarnLow Property

See also

Applies To

Termometer

Description

Determines where the warning area starts in the termometer.

Usage

[*form*.]*control*.**WarnLow**[= *setting* %]

Remarks

By default, WarnLow is set to 0.

Data Type

Integer

Wrap Property

See also

Applies To

Spin

Description

Indicates if the spinner, when reaching it's limits, should wrap around.

Usage

[*form*.]*control*.**Wrap**[= *boolean*]

Setting

By default, Wrap is set to False.

Data Type

Boolean

Change Event

See also Example

Description

Occurs when the value of the control changes; either as a result of the user manipulating the control, or if changed from inside program code.

Syntax

Sub *ctlname*_**Change** ()

Remarks

The new value is already set when this event occurs.

NeedText Event

See also   Example

Description

Occurs when the mouse cursor is over some window, and the ToolTips control needs the descriptive text for that window.

Syntax

Sub *ctlname*_**NeedText** (*hWnd* **As Integer,** *ToolText* **As String**)

Remarks

The NeedText event uses the following arguments:

Argument	Description
hWnd	The handle of the window the mouse cursor is over.
ToolText	The short tip text returned by the event handler. If an empty text is returned, the ToolTips window will be hidden.

TurnPage Event

See also Example

Description

Occurs either as a result of the user selecting one of the tabs, or if the property Tab is set from program code. It is the responsibility of this event handler to implement the turning of pages in a notebook.

Syntax

Sub *ctlname*_**TurnPage** (*ToPage* **As Integer**)

Remarks

The TurnPage event uses the following arguments:

Argument	**Description**
ToPage	The number of the page that should be made current.

Validate Event

See also   Example

Description

Occurs when the focus shifts a away from the control.

Syntax

Sub *ctlname*_**Validate** (*Cancel* **As Integer**)

Remarks

The Validate event uses the following arguments:

Argument	Description
Cancel	Set True inside Validate to prevent the focus leaving the control.

ConfigCUA Sub

Description

Use this subroutine to determine which of the supported looks to use.

Syntax

Sub ConfigCUA (**ByVal** *Look* **As Integer, ByVal** *Redraw* **As Integer**)

Remarks

The ConfigCUA subroutine uses the following arguments:

Argument	Description
Look	Determines which look will be used.
Redraw	Boolean, determines if all windows should be redrawn or not.

The Look argument can take on the following values (found in **cuactls.bas**):

Value	Description
32759	OS/2
32760	Windows 3.x - 3D
32761	Windows 95 (Chicago)
32762	Borland BWCC
32763	Windows 3.x

Change example

To try this example, paste the code into the Declarations section of a form that contains a CUAEdit control named txt1. Then press F5 and enter something in the text box.

```
Sub txt1_Change ()
   Beep
End Sub
```

NeedText example

To try this example, paste the code into the Declarations section of a form that contains a CUAToolTips and four CUAToolButtons named tbn1. Then press F5 and move the cursor.

```
Sub tip1_NeedText (hWnd As Integer, ToolText As String)
   Select Case hWnd
      Case tbn1(0).hWnd
         ToolText = "Ny"
      Case tbn1(1).hWnd
         ToolText = "Öppna"
      Case tbn1(2).hWnd
         ToolText = "Spara"
      Case tbn1(3).hWnd
         ToolText = "Skriv ut"
   End Select
End Sub
```

TurnPage example

To try this example, paste the code into the Declarations section of a form that contains a CUATab control named tab1. Then press F5 and click the tabs.

```
Sub tab1_TurnPage (ToPage As Integer)
   picPage(tab1.Tab).Visible = False
   picPage(ToPage).Visible = True
End Sub
```

Validate example

To try this example, paste the code into the Declarations section of a form that contains a CUAEdit control named txt1. Then press F5 and enter something in the text box.

```
Sub txt1_Validate (Cancel As Integer)
   Dim Number%

   Number = Val(txt1.Text)
   Cancel = (Number < 1 Or Number > 1000)
End Sub
```