See also

AutoToggle Property
FieldType Property Array
FieldWidth Property Array
KeyOffText, KeyOnText Properties
Message Property Array
Toggle Event

See also

AutoToggle Property
FieldType Property Array
FieldWidth Property Array
KeyOffText, KeyOnText Properties
Message Property Array
Numlock, ScrollLock, Capslock Properties

See also

FieldType Property Array
FieldWidth Property Array
KeyOffText, KeyOnText Properties
Message Property Array
Numlock, ScrollLock, Capslock Properties
Toggle Event

See also

FieldWidth Property Array

Message Property Array

See also

FieldWidth Property Array

See also

FieldType Property Array

FieldWidth Property Array

See also

FieldWidth Property Array
FloodColor Property
FloodField Property
FloodPercent Property
FloodShowPct Property
ForeColor Property Array
Message Property Array

See also

FieldWidth Property Array
FloodColor Property
FloodField Property
FloodInvertText Property
FloodPercent Property
ForeColor Property Array
Message Property Array

See also

FieldWidth Property Array
FloodColor Property
FloodField Property
FloodInvertText Property
FloodShowPct Property
ForeColor Property Array
Message Property Array

See also

FloodColor Property
FloodInvertText Property
FloodPercent Property
FloodShowPct Property
ForeColor Property Array
Message Property Array

See also

FloodField Property
FloodInvertText Property
FloodPercent Property
FloodShowPct Property
ForeColor Property Array
Message Property Array

See also

FieldWidth Property Array

See also

Alignment Property Array
ForeColor Property Array
Message Property Array

See also

Alignment Property Array
FieldWidth Property Array
ForeColor Property Array

See also

FieldWidth Property Array
RedrawField Property
Message Property Array

See also

FieldType Property Array

SpaceAfter Property Array

See also

FieldType Property Array
FieldWidth Property Array
Message Property Array

See also

AutoToggle Property
FieldWidth Property Array
KeyOffText, KeyOnText Properties
Message Property Array
Toggle Event

See also

FieldWidth Property Array

See also

AutoToggle Property
FieldType Property Array
Numlock, ScrollLock, Capslock Properties
Toggle Event

Toggle Example, StatusBar Control

Copy

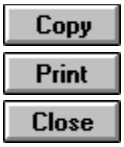
Print

Close

```
Const FIELD_NORMAL = 0
Const FIELD_CAPSLOCK = 1
Const FIELD_NUMLOCK = 2
Const FIELD_SCROLLOCK = 3
Const FIELD_CLOCK = 4

Sub Form_Load ()

    Const ALIGN_LEFT = 0
    Const ALIGN_RIGHT = 1
    Const ALIGN_CENTER = 2

    Const OFF = 0
    Const ON  = 1

    ' Align the statusbar to the bottom of the window
    ' and set the height
    StatusBar1.Align = 2
    StatusBar1.Height = 330

    ' Set messages for the toggle-keys
    StatusBar1.CapsLockOnText = "CAPS"
    StatusBar1.CapsLockOffText = "caps"
    StatusBar1.NumLockOnText = "NUM"
    StatusBar1.NumLockOffText = "num"
    StatusBar1.ScrollLockOnText = "SCROLL"
    StatusBar1.ScrollLockOffText = "scroll"

    ' The statusbar initializes three fields by setting their
    ' FieldWidth property to a value greater than zero.
    StatusBar1.FieldWidth(0) = 400
    StatusBar1.FieldWidth(1) = 40
    StatusBar1.FieldWidth(2) = 40
    StatusBar1.FieldWidth(3) = 40

    ' The spacing between the fields is set
    StatusBar1.SpaceAfter(0) = 4
    StatusBar1.SpaceAfter(1) = 2
    StatusBar1.SpaceAfter(2) = 2
    ' SpaceAfter(3) isn't set since this is the last field
    ' and it's SpaceAfter property isn't used.

    StatusBar1.FieldType(0) = FIELD_NORMAL    ' Plain text (DEFAULT)
    StatusBar1.FieldType(1) = FIELD_NUMLOCK   ' Num-lock key status
    StatusBar1.FieldType(2) = FIELD_CAPSLOCK  ' Caps-lock key status
    StatusBar1.FieldType(3) = FIELD_SCROLLOCK ' Scroll-lock key status

    ' Allow the user to double-click a toggle-field to toggle the key
    StatusBar1.AutoToggle = True
```

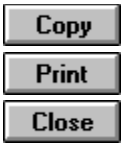
```
    ' Turn num-lock on and turn caps- and scroll-lock off.
    StatusBar1.NumLock = ON
    StatusBar1.CapsLock = OFF
    StatusBar1.ScrollLock = OFF

    ' The alignment for the last three fields is set to ALIGN_CENTER
    StatusBar1.Alignment(1) = ALIGN_CENTER    ' Center
    StatusBar1.Alignment(2) = ALIGN_CENTER    ' Center
    StatusBar1.Alignment(3) = ALIGN_CENTER    ' Center

    ' Set a message for the first field (the other three
    ' messages are ignored anyhow
    StatusBar1.Message(0) = "Demonstration of the StatusBar control"

    ' Sets the FontBold property to False to get better readable texts
    StatusBar1.FontBold = False
End Sub

Sub StatusBar1_Toggle(FieldType As Integer, KeyState As Integer)
Dim sMsg As String
    Select Case FieldType
          Case FIELD_NUMLOCK: sMsg = "Numlock"
          Case FIELD_CAPSLOCK: sMsg = "Capslock"
          Case FIELD_SCROLLOCK: sMsg = "Scroll-lock"
    End Select
    sMsg = sMsg + " is turned "
    If KeyState Then sMsg = sMsg + "on."  Else sMsg = sMsg + "off."
    MsgBox sMsg
End Sub
```

Progress Indicator Example, StatusBar Control




```
Sub Form_Load ()

    ' Only display one field of the statusbar control
    StatusBar1.FieldWidth(0)=100
    ' Set the field to automatically expand
    StatusBar1.ExpandField = 0
    ' Set the field to show a percentage
    StatusBar1.FloodField = 0
    ' Set the field to invert the text not under the filled area
    StatusBar1.FloodInvertText = True
    ' Don't display the percentage
    StatusBar1.FloodShowPct = False

    ' Set a message for the percentage field
    StatusBar1.Message(0) = "Busy creating database..."
    StatusBar1.Align = 2
    StatusBar1.FontBold = False
    StatusBar1.Height = 400
End Sub

Sub Form_Click ()
    StatusBar1.FloodPercent = StatusBar1.FloodPercent + 5
End Sub
```

' Click Example, StatusBar Control




```
Sub Form_Load ()
Dim I As Integer

    WindowState = 2
    StatusBar1.Align = 2
    StatusBar1.FontBold = False
    StatusBar1.Height = 400

    For I = 0 To 7
          StatusBar1.FieldWidth(I) = 50
          StatusBar1.Message(I) = "Field " & Format$(I)
    Next I

    StatusBar1.FieldWidth(8) = 200
    StatusBar1.Message(8) = "Click on any field"
End Sub

Sub StatusBar1_Click(Field As Integer)
Dim sMsg As String
    If Field<>8 Then
          sMsg = "Clicked on field " & Format$(Field)
          StatusBar1.ForeColor(8) = QBColor(0)
    Else
          sMsg = "Hey, you clicked on me!"
          StatusBar1.ForeColor(8) = RGB(0,0,255)
    End If
    StatusBar1.Message(8) = sMsg
End Sub
```

StatusBar Initialize Example, StatusBar Control

Copy

Print

Close

```
Sub Form_Load ()

    Const FIELD_NORMAL = 0
    Const FIELD_CAPSLOCK = 1
    Const FIELD_NUMLOCK = 2
    Const FIELD_SCROLLOCK = 3
    Const FIELD_CLOCK = 4

    Const ALIGN_LEFT = 0
    Const ALIGN_RIGHT = 1
    Const ALIGN_CENTER = 2

    Const CLOCK_HHMMSS = 0
    Const CLOCK_HHMM = 1
    Const CLOCK_HHMMSSAM = 2
    Const CLOCK_HHMMAM = 3

    Const RAISED = 1
    Const HEAVY_RAISED = 2
    Const INSET = 3
    Const HEAVY_INSET = 4

    ' Align the statusbar to the bottom of the window
    ' and set the height
    StatusBar1.Align = 2
    StatusBar1.Height = 330

    ' Make the statusbar display a lightly inset font
    StatusBar1.Font3D = INSET

    ' The statusbar initializes three fields by setting their
    ' FieldWidth property to a value greater than zero.
    StatusBar1.FieldWidth(0) = 400
    StatusBar1.FieldWidth(1) = 33
    StatusBar1.FieldWidth(2) = 70

    ' The first field is set to AutoExpand
    ' Since field 0 is the expand field, the specified width
    ' will be treated as the minimum width for this field"
    StatusBar1.ExpandField = 0

    ' All fields receive their own Forecolor
    StatusBar1.ForeColor(0) = QBColor(12)     ' Red
    StatusBar1.ForeColor(1) = QBColor(0)      ' Black
    StatusBar1.ForeColor(2) = RGB(0,0,255)    ' Bright Blue

    ' The spacing between the fields is set
    StatusBar1.SpaceAfter(0) = 4
    StatusBar1.SpaceAfter(1) = 2
```

```
    ' SpaceAfter(2) isn't set since this is the last field
    ' and it's SpaceAfter property isn't used.

    StatusBar1.FieldType(0) = FIELD_NORMAL    ' Plain text (DEFAULT)
    StatusBar1.FieldType(1) = FIELD_NUMLOCK   ' Num-lock key status
    StatusBar1.FieldType(2) = FIELD_CLOCK     ' Clock

    'Set the type of clock to use (12 hour clock with seconds)
    StatusBar1.ClockFormat = CLOCK_HHMMSSAM

    ' The alignment for the last two field is set to ALIGN_CENTER
    StatusBar1.Alignment(1) = ALIGN_CENTER    ' Center
    StatusBar1.Alignment(2) = ALIGN_CENTER    ' Center

    ' Set a message for the first field (the other two
    ' messages are ignored anyhow
    StatusBar1.Message(0) = "Demonstration of the StatusBar control"

    ' Sets the FontBold property to False to get better readable texts
    StatusBar1.FontBold = False
End Sub
```

Toggle Event, StatusBar Control

see also example

Description

Occurs when the user presses one of the Caps-lock, Num-lock or Scroll-lock keys on the keyboard or double-click a field width FieldType 1, 2 or 3 and the AutoToggle property has been set to True.

Syntax

Sub *StatusBar1*_**Toggle** (*Index* **As Integer**, *FieldType* **As Integer,** *KeyState* **As Integer**)

Remarks

The argument *Index* uniquely identifies a control if it is in a control array. The *FieldType* argument specifies the special key that was toggled. It has one of the following three values:

Value	Short name	Description
1	FIELD_CAPSLOCK	Caps-lock has been toggled
2	FIELD_NUMLOCK	Num-lock has been toggled
3	FIELD_SCROLLOCK	Scroll-lock has been toggled

The *KeyState* argument is a boolean argument and specifies whether the toggled key was toggled on or off.

The example shows you a possible way to react to the Toggle event. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

Click Event, StatusBar Control

see also example

Description

Occurs when the user presses and then releases the left-mouse button over a field of the StatusBar control. You can not trigger the Click event for the StatusBar event in code.

Syntax

Sub *StatusBar1*_**Click** (*Index* **As Integer**, *Field* **As Integer**)

Remarks

The argument *Index* uniquely identifies a control if it is in a control array. The *Field* argument specifies the number of the text-field that was clicked on. You can use this number to take some action whenever the user clicks on a field.

The example shows you a possible way to react to the Click event. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

Font3D Property, StatusBar Control

see also example

Description

Specifies how that StatusBar control displays the text in the text-fields.

Usage

[*form!*]*StatusBar1*.**Font3D** [= *setting%*]

Remarks

The StatusBar can display fonts in five different ways. The Font3D property takes one of the following values to specify the way the Statusbar displays the font in a text-field.

Value	Short name	Description
0	DEFAULT	(Default) Simply displays the selected font.
1	RAISED	Displays the font slightly raised.
2	HEAVY_RAISED	Displays a heavy raised font.
3	INSET	Displays the font slightly inset.
4	HEAVY_INSET	Display a heavy inset font.

Data Type

Integer (Enumerated)

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

ClockFormat Property, StatusBar Control

see also example

Description

Specifies the time-format for text-fields with FieldType 3 (FIELD_CLOCK).

Usage

[*form!*]*StatusBar1*.**ClockFormat** [= *setting%*]

Remarks

The StatusBar can display four different clocks. The ClockFormat property takes one of the following values to specify the time-format of a text-field with FieldType set to 3.

Value	Short name	Description
0	CLOCK_HHMMSS	(Default) Display a 24-hour clock with seconds.
1	CLOCK_HHMM	Displays a 24-hour clock without the seconds.
2	CLOCK_HHMMSSAM	Displays a 12-hour clock with seconds prefixed by AM or PM.
3	CLOCK_HHMMAM	Displays a 12-hour clock without the seconds prefixed by AM or PM.

Data Type

Integer (Enumerated)

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

NumLock, CapsLock and ScrollLock Properties, StatusBar Control

see also example

Description

Sets or returns the current state of the num-lock, caps-lock or scroll-lock key.

Usage

[*form!*]*StatusBar1*.**NumLock** [= *setting%*]

[*form!*]*StatusBar1*.**CapsLock** [= *setting%*]

[*form!*]*StatusBar1*.**ScrollLock** [= *setting%*]

Remarks

You can use the NumLock, CapsLock and ScrollLock properties to set or get the state of the num-lock, caps-lock keys.

The NumLock, CapsLock and ScrollLock properties are only available at run-time, since they are of no use in the design time environment (its much quicker to press one of the keys than to set it explicitly through the Properties-window). These properties always reflect the current state of the toggle keys, so if you set the NumLock property to one (ON) and a user presses the Num-Lock key after that, getting the NumLock property will return zero (OFF). There are two possible settings for the NumLock, CapsLock and ScrollLock properties:

Value	Short Name	Meaning
0	OFF	The state of num--lock, caps-lock or scroll-lock is (turned) off.
1	ON	The state of num--lock, caps-lock or scroll-lock is (turned) on.

Data Type

Integer (Enumerated)

The example demonstrates the use of the NumLock, CapsLock and ScrollLock properties. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

AutoToggle Property, StatusBar Control

see also example

Description

Specifies if the user can toggle the special keys num-lock, caps-lock and scroll-lock by double clicking on a field with FieldType 1, 2 or 3.

Usage

[*form!*]*StatusBar1*.**AutoToggle** [= *setting%*]

Remarks

The AutoToggle property only has effect when you have defined fields with FieldType 1, 2 or 3. There are two possible settings for the AutoToggle property:

AutoToggle	Result
True	Toggles num-lock, caps-lock or scroll-lock when a user double-clicks on such a field.
False	Does not toggle num-lock, caps-lock or scroll-lock on double-clicks.

Data Type

Integer (Boolean)

The example demonstrates the use of the AutoToggle property. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

FloodInvertText Property, StatusBar Control

see also example

Description

Specifies whether or not to change the color of the text according to the progress of the percentage in the text of the text-field specified by the the FloodField property.

Usage

[*form!*]*StatusBar1*.**FloodInvertText** [= *setting%*]

Remarks

The FloodInvertText property affects the text that is displayed in the text-field specified by the FloodField property. There are two possible settings for the FloodInvertText property:

FloodInvertText	Color of text-field
True	The text-color of the text displayed in the text-field is different for the text on the progress indicator bar and the space next to the bar. The text-color in the space right next to the bar is set to the same color as the color specified in the FloodColor property. The text on the bar is drawn with the color specified in the ForeColor property array for that field.
False	The entire text is displayed in the color specified in the ForeColor property array.

Data Type

Integer (Boolean)

The example demonstrates the use of the FloodInvertText property. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

FloodShowPct Property, StatusBar Control

see also example

Description

Specifies whether or not to include the percentage in the text of the text-field specified by the the FloodField property.

Usage

[*form!*]*StatusBar1*.**FloodShowPct** [= *setting%*]

Remarks

The FloodShowPct property affects the text that is displayed in the text-field specified by the FloodField property. Another thing that affects the text is the Message property for that text-field. There are four possible combinations:

FloodShowPct	Message	Contents of text-field
True	some text	The text specified in the Message property-array followed by a colon, followed by the percentage (taken from the FloodPercent property).
True	empty	Only the percentage.
False	some text	The text specified in the Message property-array.
False	empty	empty

Data Type

Integer (Boolean)

The example demonstrates the use of the FloodShowPct property. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

FloodPercent Property, StatusBar Control

see also example

Description

Specifies the percentage of the text-field specified by the FloodField property to be filled.

Usage

[*form!*]*StatusBar1*.**FloodPercent** [= *setting%*]

Remarks

The FloodPercent property sets or retrieves the percentage of the text-field that will be filled with the color specified by the FloodColor property. If the FloodPercent property is set to a value smaller than zero or greater than one hundred it's value is automatically adjusted. This setting will have no effect is the FloodField property is set to -1 or the FloodField property is set to a field with a FieldWidth of zero.

Data Type

Integer

The example demonstrates the use of the FloodPercent property. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

FloodField Property, StatusBar Control

see also example

Description

Specifies which field is to be used as a progress indicator.

Usage

[*form!*]*StatusBar1*.**FloodField** [= *setting%*]

Remarks

The FloodField property allows you to use one field of the StatusBar control as a progress indicator. The message for this field (specified in the Message property array) will be placed before the percentage specified in the FloodPercent property. A colon is automatically inserted between the message and the percentage. If no message is specified for the text-field the colon is omitted. The FloodShowPct property specifies whether or not to append the actual percentage to the message specified for the FloodField.

If the FloodField property is set to a field that has a FieldWidth of zero, the FloodField is not displayed.

The field is filled with the color specified by the FloodColor property.

Data Type

Integer

The example demonstrates the use of the FloodPercent property. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

FloodColor Property, StatusBar Control

see also example

Description

Sets or retrieves the color used to paint the area inside a text-field when the statusbar is used as a percentage indicator (only when the FloodField property is setting is other than -1).

Usage

[*form!*]*StatusBar1*.**FloodColor** [= *color&*]

Remarks

The FloodColor property has the same range of settings as standard Visual Basic color settings.

Use this property with FloodField and FloodPercent to cause the StatusBar to display a colored percentage bar indicating the degree of completion of a task.

At design time you can set this property by entering a hexadecimal value in the Settings box or by clicking the three dots that appear at the right of the Settings box. Clicking this button displays a dialog box that allows you to select a FloodColor setting from the Visual Basic Color Palette.

Note

The FloodColor property defaults to bright blue: RGB (0, 0, 255). The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower three bytes, from least to most significant, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF).

Data Type

Long

The example demonstrates the use of the FloodPercent property. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

SpaceAfter Property Array, StatusBar Control

see also example

Description

Sets or retrieves the spacing between the text-fields StatusBar control.

Usage

[*form!*]*StatusBar1*.**SpaceAfter(I)** [= *setting%*]

Remarks

The SpaceAfter property array allows you to define the spacing between the text-fields in the StatusBar control. It specifies how much space (in pixels) to leave behind a text-field before the next text-field is shown. The index in the SpaceAfter array is the same index as in the FieldWidth and Message arrays. So if you set SpaceAfter(0) to 4, you define a spacing of 4 pixels after the first field. The default setting is 3 pixels, but you can set it to any value you like.

Data Type

Integer

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

RedrawField Property, StatusBar Control

see also

Description

Forces the StatusBar to redisplay one or all text-fields.

Usage

[*form!*]*StatusBar1*.**RedrawField** [= *setting%*]

Remarks

If you specify an alternate Alignment and ForeColor for a text-field, the field is not updated automatically. This is done on purpose to avoid flickering if you want to display a new message in a different color.

If you only want to change the forecolor or alignment of a field without changing the message, use the RedrawField property to redisplay the message with the new settings. Specifying -1 for the RedrawField property forces all messages to be redrawn. Of course, you can also force the StatusBar control to redraw all fields by using the Refresh method or by assigning the Message property of a field to itself (StatusBar1.Message(0) = StatusBar1.Message(0)), but using the RedrawField property is more efficient.

This property is not available at design time and write-only at run-time.

Data Type

Integer

Message Property Array, StatusBar Control

see also example

Description

Sets or retrieves the text to be displayed in one of the text-fields of the StatusBar control.

Usage

[*form!*]*StatusBar1*.**Message(I)** [= *message$*]

Remarks

Setting a new message for a text-field automatically redisplays the message, so be sure to set other properties, like Alignment or ForeColor, before you change the message. If the message is to big to fit in the field, the message is clipped.

Only text-fields with a FieldWidth bigger than zero are displayed. Fields with zero-length are not displayed regardless of the message you specify for them

Data Type

String

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

LeftMargin, RightMargin Properties, StatusBar Control

Description

Set or retrieve the left of right-margin for the StatusBar control.

Usage

[*form!*]*StatusBar1*.**LeftMargin** [= *setting%*]

[*form!*]*StatusBar1*.**RightMargin** [= *setting%*]

Remarks

The LeftMargin property specifies how much room to leave before displaying the first text-field, the RightMargin property specifies the space to reserve after the last text-field.

The Left- and Right margin are always expressed in pixels, regardless of the ScaleMode of the StatusBar's parent.

Data Type

Integer

ForeColor Property Array, StatusBar Control

see also example

Description

Sets or retrieves the field text-color of one of the text-fields of the StatusBar control.

Usage

[*form!*]*StatusBar1*.**ForeColor(I)** [= *color&*]

Remarks

Visual Basic uses the Microsoft Windows environment RGB scheme for colors. Each property has the following ranges of settings:

Range of settings	Description
Normal RGB colors	Colors by using the RGB or QBColor functions in code.
System default colors	Colors specified with system color constants from CONSTANT.TXT, a Visual Basic file that specifies system defaults. The Windows environment substitutes the user's choices as specified in the user's Control Panel settings.

For the StatusBar control the default settings at design time are:

ForeColor(I) set to the WINDOW_TEXT system color as specified in CONSTANT.TXT.

Setting the ForeColor property does not affect messages already displayed to avoid flickering if you want to display a new message in a different color. To force the current message to be redisplayed in the new color set the RedrawField property to the number of the field to redisplay.

The valid range for a normal RGB color is 0 to 16,777,215 (&HFFFFFF). The high byte of a number in this range equals 0; the lower three bytes, from least to most significant byte, determine the amount of red, green, and blue, respectively. The red, green, and blue components are each represented by a number between 0 and 255 (&HFF). If the high byte is not 0, Visual Basic uses the system colors, as defined in the user's Control Panel and enumerated in CONSTANT.TXT.

To display text in the Windows environment, the text-colors must be solid. If the text-colors you've selected are not displayed, the selected color may be dithered that is, comprised of up to three different-colored pixels. If you choose a dithered color for the text, the nearest solid color will be substituted.

Data Type

Long

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

FieldWidth Property Array, StatusBar Control

see also example

Description

Set or retrieves the field width of one of the text-fields of the StatusBar control.

Usage

[*form!*]*StatusBar1*.**FieldWidth(I)** [= *setting%*]

Remarks

The FieldWidth of a field is always expressed in pixels. A field that has a field-width of zero is not displayed, nor is it's SpaceAfter property used in calculating the relative distances between the text-fields.

Data Type

Integer

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

Alignment Property Array, StatusBar Control

see also example

Description

Specifies the alignment for a text-field of the StatusBar control.

Usage

[*form!*]*StatusBar1*.**Alignment(I)** = *Alignment%*

Remarks

The StatusBar can align each text-field in three ways. The Alignment property array takes one of the following values to specify the alignment of a text-field.

Value	Short name	Description
0	LEFT	(Default) Aligns the text at the left-edge of the text-field.
1	RIGHT	Aligns the text at the right-edge of the text-field.
2	CENTER	Centers the text in the text-field.

The index in the Alignment property array is the same index in the Message property array.

Data Type

Integer (Enumerated)

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

FieldType Property Array, StatusBar Control

see also example

Description

Sets or retrieves the field-type for the text-field with the same index.

Usage

[*form!*]*StatusBar1*.**FieldType(I)** [= *FieldType%*]

Remarks

The StatusBar supports five different field-types. The FieldType property array takes one of the following values to specify the type of a text-field.

Value	Short name	Description
0	FIELD_NORMAL	(Default) The text-field displays the programmer defined text.
1	FIELD_CAPSLOCK	Displays the current status of the CapsLock key. If the CapsLock key is toggled on, the text from the property CapsLockOffText is displayed, otherwise the CapsLockOnText is displayed.
2	FIELD_NUMLOCK	Displays the current status of the NumLock key.
3	FIELD_SCROLLOCK	Displays the current status of the ScrollLock key.
4	FIELD_TIME	Displays a digital 24-hour clock

The index in the FieldType property array is the same index in the Message and FieldWidth property arrays.

Data Type

Integer (Enumerated)

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

ExpandField Property, StatusBar Control

see also example

Description

The ExpandField property specifies which field is stretched to the size of the StatusBar control.

Usage

[*form!*]*StatusBar1*.**ExpandField** [= *setting%*]

Remarks

The effect of the ExpandField property is that all other field are positioned first and that after that the remaining space is filled by the text-field specified by the ExpandField property (starting at the correct position of course). If you don't want a text-field to be stretched, set the ExpandField property to a text-field with FieldWidth set to zero. The FieldWidth property for the text-field that the ExpandField points to, will be regarded as the minimum width for that text-field. If the StatusBar control isn't wide enough to display all fields, the fields will extent off the StatusBar control.

If you don't want any field stretched to the full size of your window, set the value of the ExpandField property to -1.

Data Type

Integer

The example shows a Form_Load that initializes three text-fields on a StatusBar control. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

CapsLockOffText, CapsLockOnText, NumLockOffText, NumLockOnText, ScrollLockOffText, ScrollLockOnText Properties, StatusBar Control

see also

Description

Specify the text to show for a text-field of type 2 (Caps-lock), 3 (Num-lock) or 4 (Scroll-lock).

Usage

[*form!*]*StatusBar1*.***Key*OnText** [= *setting$*]

[*form!*]*StatusBar1*.***Key*OffText** [= *setting$*]

Remarks

The *Key*OnText is shown when the specified key is toggled on, *Key*OffText is shown when the key is toggled off. You can specify the type of a text-field by setting the FieldType property.

Data Type

String

The example shows you a how these properties are used. To use this example create a form with one StatusBar control and paste the code into the Declarations section of your form.

Events

The StatusBar control supports the following events:

Click DragDrop DragOver Toggle

Methods

The StatusBar control supports the following methods:

Drag	Move	Refresh	ZOrder

Properties

All the properties that apply to the Status Bar control are listed in the following table. All properties that are marked with an asterisk (*) are only available at run-time.

Align	Alignment *	AutoToggle	CapsLock *
CapsLockOffText	CapsLockOnText	ClockFormat	Enabled
ExpandField	FieldType *	FieldWidth *	FloodColor
FloodField	FloodInvertText	FloodPercent	Font3D
FontBold	FontItalic	FontName	FontSize
FontStrikethru	FontUnderline	ForeColor *	Height
hWnd	Index	Left	LeftMargin
Message *	Name	NumLock *	NumLockOffText
NumLockOnText	Parent	RedrawField *	RightMargin
ScrollLock *	ScrollLockOffText	ScrollLockOnText	SpaceAfter *
Tag	Top	Visible	Width

Note the Alignment, CapsLock, FieldType, FieldWidth, ForeColor, NumLock, RedrawField and ScrollLock properties are only available at run-time. Name is the default property for the StatusBar control.

The StatusBar Custom Control

Properties Methods Events

Description

The Microsoft Visual Basic programming system for Windows comes with a large set of 3D controls. Unfortunately, a 3D status-bar, as used in almost every MS-Windows application is lacking. Therefore the the Status Bar Custom Control was designed. This control allows you to create a very versatile status bar for all your applications.

File Name

TOOLBARS.VBX

Object Type

StatusBar

Toolbox Icon

Remarks

The StatusBar allows an application to display a status-bar at the bottom of a form. The statusbar has 20 fully configurable text-fields. Also, the statusbar provides standard fields like the Num-lock, Caps-lock and Scroll-lock toggles and a little clock. The StatusBar control can also be used as a progress indicator.

Usage

To use the StatusBar, perform the following steps:

1. Add the toolbar custom control to your project. The StatusBar and the ButtonBar icons will appear in the Visual Basic tool-palette.

2. In the Form_Load event procedure for the form, set the FieldWidth and FieldType properties for the number of fields you want to display.

3. Specify a Message for each field with FieldType 0.

4. Eventually add code to respond to a Click on a certain field in the StatusBar.

Distribution Note When you create and distribute applications that use the StatusBar control, you should install the file TOOLBARS.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory.

Copyright Notice

The Toolbar Custom Controls, Version 1.1

Description

The Microsoft Visual Basic programming system for Windows comes with a large set of 3D controls. Unfortunately, a 3D status-bar and a 3D tool-bar, as used in almost every MS-Windows application are lacking. Therefore, the Toolbar custom controls (TOOLBARS.VBX) were developed. This VBX contains two custom controls: the Status Bar Custom Control and the Button Bar custom control. These control allows you to create very versatile status- and buttonbars for all your applications.

File Name

TOOLBARS.VBX

Object Types

StatusBar, ButtonBar

Toolbox Icons


StatusBar control


ButtonBar control

Remarks

The StatusBar allows an application to display a status-bar at the bottom of a form. The statusbar has 20 fully configurable text-fields. Also, the statusbar provides standard fields like the Num-lock, Caps-lock and Scroll-lock toggles and a little clock. The StatusBar control can also be used as a progress indicator.

The ButtonBar allows an application to display a button-bat at the top of a form. The button-bar has 20 fully configurable buttons. The only thing you have to do is specify the bitmap for the up-position of each button. The ButtonBar control than calculates the different bitmaps for the down and the two disabled states (up and down) of the button. It is also possible to connect the button-bar to the status-bar and specify messages for each button to be shown when the button is selected.

Distribution Note When you create and distribute applications that use one of the ToolBar controls, you should install the file TOOLBARS.VBX in the customer's Microsoft Windows \SYSTEM subdirectory.

See also

ButtonGroup Property Array
ButtonState Property Array
ButtonType Property Array

See also

ButtonMessage Property Array
hWndStatusBar Property
StatusBar Custom Control

See also

ButtonMessage Property Array
StatusField Property
StatusBar Custom Control

See also

ButtonEnabled Property Array
ButtonState Property Array
Picture Property Array

See also

ButtonEnabled Property Array
ButtonState Property Array
PictureDisabled Property Array

See also

ButtonGroup Property Array
ButtonState Property Array
ButtonType Property Array

See also

Picture Property Array

See also

hWndStatusBar Property
StatusField Property
StatusBar Custom Control

See also

ButtonState Property Array
ButtonType Property Array
GroupAllowAllUp Property Array

See also

ButtonGroup Property Array
ButtonType Property Array
GroupAllowAllUp Property Array

See also

ButtonGroup Property Array
ButtonState Property Array
GroupAllowAllUp Property Array

See also

Picture Property Array

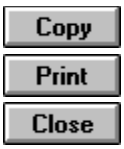
PictureDisabled Property Array

Usage with the StatusBar Control, ButtonBar Control

Copy
Print
Close

```
Sub Form_Load ()
Dim I As Integer

Const BUTTON_2STATE = 1

WindowState = 2
ButtonBar1.Align = 1
ButtonBar1.Height = 400
StatusBar1.Align = 2
StatusBar1.Height = 400

' Specify the pictures for the first four buttons
ButtonBar1.Picture(0)=LoadPicture("c:\vb\bitmaps\toolbar3\tbl-up.bmp")
ButtonBar1.Picture(1)=LoadPicture("c:\vb\bitmaps\toolbar3\tbc-up.bmp")
ButtonBar1.Picture(2)=LoadPicture("c:\vb\bitmaps\toolbar3\tbr-up.bmp")
ButtonBar1.Picture(3)=LoadPicture("c:\vb\bitmaps\toolbar3\tbd-up.bmp")

' Make all buttons members of group 0
ButtonBar1.ButtonGroup(0) = 0
ButtonBar1.ButtonGroup(1) = 0
ButtonBar1.ButtonGroup(2) = 0
ButtonBar1.ButtonGroup(3) = 0
' Allow all buttons in the group to be up.
ButtonBar1.GroupAllowAllUp(0) = True

' Set all buttons to type 1 (2-state button)
For I = 0 To 3
    ButtonBar1.ButtonType(I) = BUTTON_2STATE
Next I

' Specify messages to display in the statusbar
ButtonBar1.ButtonMessage(0) = "Set a left align tab"
ButtonBar1.ButtonMessage(1) = "Set a centered tab"
ButtonBar1.ButtonMessage(2) = "Set a right align tab"
ButtonBar1.ButtonMessage(3) = "Set a decimal tab"

' Specify the statusbar and the field to display the messages
ButtonBar1.hWndStatusBar = StatusBar1.hWnd
ButtonBar1.StatusField = 0

' Set a few properties of the statusbar control
StatusBar1.FieldWidth(0) = 100
StatusBar1.Message(0) = "Static message for field 0"

End Sub
```

Click Example, ButtonBar Control




```
Sub Form_Load ()
Const BUTTON_NORMAL = 0
Const BUTTON_2STATE = 1

 WindowState = 2
 ButtonBar1.Align = 1
 ButtonBar1.Height = 400

 ButtonBar1.Picture(0)=LoadPicture("c:\vb\bitmaps\toolbar\lft-up.bmp")
 ButtonBar1.Picture(1)=LoadPicture("c:\vb\bitmaps\toolbar\rt-up.bmp")
 ButtonBar1.Picture(2)=LoadPicture("c:\vb\bitmaps\toolbar\cnt-up.bmp")
 ButtonBar1.PictureDisabled(0)=LoadPicture("c:\vb\bitmaps\toolbar\lft-
dis.bmp")
 ButtonBar1.PictureDisabled(1)=LoadPicture("c:\vb\bitmaps\toolbar\rt-
dis.bmp")
 ButtonBar1.PictureDisabled(2)=LoadPicture("c:\vb\bitmaps\toolbar\cnt-
dis.bmp")
 ButtonBar1.ButtonType(0) = BUTTON_2STATE
 ButtonBar1.ButtonType(1) = BUTTON_2STATE
 ButtonBar1.ButtonType(2) = BUTTON_2STATE
 ButtonBar1.ButtonGroup(0) = 0
 ButtonBar1.ButtonGroup(1) = 0
 ButtonBar1.ButtonGroup(2) = 0

 ButtonBar1.ButtonEnabled(1)=False

 ButtonBar1.ButtonState(0) = True
 ButtonBar1.GroupAllowAllUp(0) = False
End Sub

Sub ButtonBar1_Click(Button As Integer, Group As Integer, State As Integer)
    Label1.Alignment = Button
    Select Case Button
          Case 0: Label1.Caption = "Text is now left-aligned"
          Case 1: 'Should never get here since the centered button is
disabled.
                  Label1.Caption = "Text is now right-aligned"
          Case 2: Label1.Caption = "Text is now centered"
    End Select
End Sub
```

ButtonBar Initialize Example, ButtonBar Control

Copy
Print
Close

```
Sub Form_Load ()
Dim I As Integer
Const BUTTON_NORMAL = 0
Const BUTTON_2STATE = 1

Const RAISED = 0
Const DEPRESSED = -1

' Align the button to the top of the window
' and set the height
ButtonBar1.Align = 1
ButtonBar1.Height = 400

' Set the first four button-pictures.
ButtonBar1.Picture(0)=LoadPicture("c:\vb\bitmaps\toolbar3\tbl-up.bmp")
ButtonBar1.Picture(1)=LoadPicture("c:\vb\bitmaps\toolbar3\tbc-up.bmp")
ButtonBar1.Picture(2)=LoadPicture("c:\vb\bitmaps\toolbar3\tbr-up.bmp")
ButtonBar1.Picture(3)=LoadPicture("c:\vb\bitmaps\toolbar3\tbd-up.bmp")
' Specify five pixels room after button #3
ButtonBar1.SpaceAfter(3) = 5
' Make the first four buttons a member of group 0
ButtonBar1.ButtonGroup(0) = 0
ButtonBar1.ButtonGroup(1) = 0
ButtonBar1.ButtonGroup(2) = 0
ButtonBar1.ButtonGroup(3) = 0
' Specify that in group 0, all buttons may be raised
ButtonBar1.GroupAllowAllUp(0) = True

' Create four extra buttons
ButtonBar1.Picture(4)=LoadPicture("c:\vb\bitmaps\toolbar3\lft-up.bmp")
ButtonBar1.Picture(5)=LoadPicture("c:\vb\bitmaps\toolbar3\cnt-up.bmp")
ButtonBar1.Picture(6)=LoadPicture("c:\vb\bitmaps\toolbar3\rt-up.bmp")
ButtonBar1.Picture(7)=LoadPicture("c:\vb\bitmaps\toolbar3\jst-up.bmp")
' Leave 5 pixels space after button #7
ButtonBar1.SpaceAfter(7) = 5
' Make the second four buttons a member of group 1
ButtonBar1.ButtonGroup(4) = 1
ButtonBar1.ButtonGroup(5) = 1
ButtonBar1.ButtonGroup(6) = 1
ButtonBar1.ButtonGroup(7) = 1
' Specify that at least one button of the group must be depressed
ButtonBar1.GroupAllowAllUp(1) = False

' Make three extra buttons
ButtonBar1.Picture(8)=LoadPicture("c:\vb\bitmaps\toolbar3\bld-up.bmp")
ButtonBar1.Picture(9)=LoadPicture("c:\vb\bitmaps\toolbar3\itl-up.bmp")
ButtonBar1.Picture(10)=LoadPicture("c:\vb\bitmaps\toolbar3\ulin-up.bmp")

' Leave 5 pixels room after button 10
```

```
ButtonBar1.SpaceAfter(10) = 5

' Mark buttons 0-10 as 2-state buttons
For I = 0 To 10
    ButtonBar1.ButtonType(I) = BUTTON_2STATE
Next I

' Press button 4 down
ButtonBar1.ButtonState(4) = True

' Create the last button
ButtonBar1.Picture(11)=LoadPicture("c:\vb\bitmaps\toolbar3\hlp-up.bmp")
End Sub
```

Click Event, ButtonBar Control

see also example

Description

Occurs when the user presses and then releases the left-mouse button over a button of the ButtonBar control. You can not trigger the Click event for the ButtonBar event in code.

Syntax

Sub *ButtonBar1*_**Click** (*Index* **As Integer**, *Button* **As Integer,** *Group* **As Integer,** *State* **As Integer**)

Remarks

The argument *Index* uniquely identifies a control if it is in a control array. The *Button* argument specifies the number of the button-field that was clicked on. You can use this number to take some action whenever the user clicks on a button. The *Group* argument specifies the group the clicked button is a member of. If the *Group* argument is -1, the button is not a member of a group. The State argument specifies the state of the button. This argument is always True (-1) if the ButtonType of the button is 0 (BUTTON_NORMAL). If the ButtonType of the clicked button is 1 (BUTTON_2STATE) then the *State* argument specifies whether the button is up (State = False) or down (State = True).

The example shows you a possible way to react to the Click event. To use this example create a form with one ButtonBar control and a Label control. Then, paste the code into the Declarations section of your form.

OutlineChildren Property, ButtonBar Control

Description

Apllies a 3D look to controls placed on the ButtonBar.

Usage

[*form!*]*ButtonBar1*.**OutlineChildren** [= *setting%*]

Remarks

The ButtonBar can automatically give all the controls you place on it a 3D look by drawing a raised or inset outline. The OutlineChildren property takes one of the following values to specify the way the ButtonBar applies the 3D effect.

Value	Short name	Description
0	NONE	(Default) Does not draw an outline around children.
1	RAISED	Displays a raised border around each control placed on the ButtonBar.
2	INSET	Displays an inset border around each controls placed on the ButtonBar.

The 3D effect is useful when you want to put more than just buttons on the ButtonBar (like a font-selection combo-box). These control can appear raised or inset just by setting the OutlineChildren property.

The OutlineChildren property only has effect on controls that have a window-handle. Therefore, graphical controls (like labels) do not receive an outline.

Warning

Since the ButtonBar has to keep track of all children placed on it and their position, the maximum number of windowed child controls (that is; controls with a window-handle) on the ButtonBar is limited to 20.

Data Type

Integer (Enumerated)

StatusField Property, ButtonBar Control

see also example

Description

Specifies the text-field of the status-bar that is to display the messages specified for the buttons.

Usage

[*form!*]*ButtonBar1*.**StatusField** [= *setting%*]

Remarks

The StatusField property sets or retrieves the text-field of the status-bar that is to display the messages you specify in the ButtonMessage property array. If the text-field has a FieldWidth of zero, the messages are not displayed.

The messages are displayed in the statusbar specified by the hWndStatusBar property.

Data Type

Integer

The example shows a Form_Load that initializes some buttons with messages on a ButtonBar control. To use this example create a form with one ButtonBar and one StatusBar control and paste the code into the Declarations section of your form.

hWndStatusBar Property, ButtonBar Control

see also example

Description

Specifies the window handle of the status-bar that is to display the messages specified for the buttons.

Usage

[*form!*]*ButtonBar1*.**hWndStatusBar** [= [*form!*].*StatusBar1*.**hWnd**]

Remarks

The hWndStatusBar property sets or retrieves the window handle for the status-bar that is to display the messages you specify in the ButtonMessage property array. The window-handle is checked to see if it is really the window handle of a statusbar control. If it isn't, a runtime error is generated.

The messages are displayed in the statusbar in the text-field specified by the StatusField property. This property defaults to 0.

Data Type

Integer

The example shows a Form_Load that initializes some buttons with messages on a ButtonBar control. To use this example create a form with one ButtonBar and one StatusBar control and paste the code into the Declarations section of your form.

PictureDisabled Property Array, ButtonBar Control

see also example

Description

Defines the picture to use for showing a disabled button on the ButtonBar.

Usage

[*form!*]*ButtonBar1*.**PictureDisabled(I)** [= *picture*]

The Picture property settings are:

Setting	Description
(none)	(Default) No picture.
(bitmap)	You can set this property using the LoadPicture function on a bitmap.

Remarks

If you don't like the default look of the disabled pictures (all black pixels transformed to gray), you can set the PictureDisabled property to another bitmap that has a different disabled look. Like the Picture property array, the picture is assumed to be a bitmap that is already formatted as a button.

When the PictureDisabled property of the button is set, the ButtonBar uses this picture to calculate the pictures for the disabled up and down states of the button.

The ButtonBar control makes some assumptions about the format of the picture you specify for the PictureDisabled property in order to be able to create the different pictures needed for the other states of the button. All pictures in the TOOLBAR3 directory that come with Visual Basic have a correct layout. You can use or modify these pictures as needed.

The following table specifies the operations that the ButtonBar will perform on a bitmap for the different ButtonStates and ButtonEnabled properties of the button:

ButtonState	ButtonEnabled	Description
UP (0)	True	Value from the Picture property unchanged.
UP	False	Value from the PictureDisabled property unchanged.
DOWN (-1)	True	Value from the Picture property shifted one pixel to the right and below. The dark-gray bevel is removed and the white bevel is replaced with a dark gray bevel.
DOWN	False	Value from the PictureDisabled property modified shifted on pixel to the right and below, just like with the ButtonEnabled property set to True.

Data Type

Integer

The example shows a Form_Load that initializes some buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

Hint Since you have to set a picture for each button, you can use the ButtonBar control

perfectly in conjunction with the PictureClip control that comes with Visual Basic. The easiest way to do this is to create a large bitmap that contains two rows of bitmaps; one with the normal button pictures and the other one with the disabled pictures. This allows you to load all the bitmaps needed for the ButtonBar in a single for-loop.

Picture Property Array, ButtonBar Control

see also example

Description

Defines the picture to use for showing a button on the ButtonBar.

Usage

[*form!*]*ButtonBar1*.**Picture(I)** [= *picture*]

The Picture property settings are:

Setting	Description
(none)	(Default) No picture.
(bitmap)	You can set this property using the LoadPicture function on a bitmap.

Remarks

The Picture property array is the most important property for the ButtonBar. It specifies the picture to be used for a button on the ButtonBar. The picture is assumed to be a bitmap that is already formatted as a button.

If there is no picture in the PictureDisabled property array for the button, the ButtonBar uses this picture to calculate the pictures for the down and disabled states of the button.

The ButtonBar control makes some assumptions about the format of the picture you specify for the Picture property in order to be able to create the different pictures needed for the other states of the button. All pictures in the TOOLBAR3 directory that come with Visual Basic have a correct layout. You can use or modify these pictures as needed.

The following table specifies the operations that the ButtonBar will perform on a bitmap for the different ButtonStates and ButtonEnabled properties of the button if there is no picture specified for the disabled state:

ButtonState	ButtonEnabled	Description
UP (0)	True	Value from the Picture property unchanged.
UP	False	Value from the Picture property with all black pixels transformed to dark-gray.
DOWN (-1)	True	Value from the Picture property shifted one pixel to the right and below. The dark-gray bevel is removed and the white bevel is replaced with a dark gray bevel.
DOWN	False	Value from the Picture property modified as with the ButtonEnabled property set to True, but with all black pixels transformed to dark-gray.

Data Type

Integer

The example shows a Form_Load that initializes some buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

Hint Since you have to set a picture for each button, you can use the ButtonBar control

perfectly in conjunction with the PictureClip control that comes with Visual Basic. The easiest way to do this is to create a large bitmap that contains two rows of bitmaps; one with the normal button pictures and the other one with the disabled pictures. This allows you to load all the bitmaps needed for the ButtonBar in a single for-loop.

GroupAllowAllUp Property Array, ButtonBar Control

see also example

Description

Sets or retrieves the ability of buttons in a group to be all in the up-state.

Usage

[*form!*]*ButtonBar1*.**GroupAllowAllUp(I)** [= *setting%*]

Remarks

The GroupAllowAllUp property settings are:

Setting	Description
True	(Default) Allows all buttons in the group to be raised.
False	At least one button in the group must be depressed.

The GroupAllowAllUp property specifies if the user can deselect all buttons in a group, or that at least one button must be in the down position. Normally, when buttons are selected in a group, there is always one button in the down state. Clicking this button will leave it in the down position. When you specify True for the GroupAllowAllUp property, the user can deselect all buttons, by clicking the button that is down.

Data Type

Integer (Boolean)

The example shows a Form_Load that initializes some buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

SpaceAfter Property Array, ButtonBar Control

see also example

Description

Sets or retrieves the spacing between the buttons on a ButtonBar control.

Usage

[*form!*]*ButtonBar1*.**SpaceAfter(I)** [= *setting%*]

Remarks

The SpaceAfter property array allows you to define the spacing between the buttons in the ButtonBar control. It specifies how much space (in pixels) to leave behind a button before the next button is shown. The index in the SpaceAfter array is the same index as in the Picture array. So, if you set SpaceAfter(0) to 4, you define a spacing of 4 pixels after the first button. The default setting is 0 pixels, but you can set it to any value you like.

Data Type

Integer

The example shows a Form_Load that initializes some buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

ButtonMessage Property Array, ButtonBar Control

see also example

Description

Sets or retrieves the text to be displayed in one of the text-fields of a StatusBar control if the button is selected.

Usage

[*form!*]*ButtonBar1*.**ButtonMessage(I)** [= *message$*]

Remarks

It is possible to make a connection between the StatusBar control and the ButtonBar control, by filling in the StatusField and hWndStatusBar properties.

After that you can specify a message for each button you have defined, by filling in the appropriate entry in the ButtonMessage property array. The message will be displayed in the specified field of the StatusBar control whenever the button is selected and will be removed from that field when the button is no longer selected.

Data Type

String

The example shows a Form_Load that initializes some buttons with messages on a ButtonBar control. To use this example create a form with one ButtonBar and one StatusBar control and paste the code into the Declarations section of your form.

ButtonGroup Property Array, ButtonBar Control

see also example

Description

Sets or retrieves the group a button on the ButtonBar control is a member of.

Usage

[*form!*]*ButtonBar1*.**ButtonGroup(I)** [= *setting%*]

Remarks

You can group 2-state buttons in a group, by setting the ButtonGroup property of all the buttons you want to group to the same number.

When buttons are selected into a group, they behave like toggles; when one of the buttons in a group is depressed, all other buttons are raised. Once a button is depressed it cannot be raised again by clicking on it a second time, unless you alter the GroupAllowAllUp property for the group the button is a member of.

To remove a button from a group, set the ButtonGroup property of that button to -1.

Data Type

Integer

The example shows a Form_Load that initializes a couple of buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

LeftMargin Property, ButtonBar Control

Description

Set or retrieve the left-margin for the ButtonBar control.

Usage

[*form!*]*StatusBar1*.**LeftMargin** [= *setting%*]

Remarks

The LeftMargin property specifies how much room to leave before displaying the first button

The left margin is always expressed in pixels, regardless of the ScaleMode of the ButtonBar's parent.

Data Type

Integer

ButtonState Property Array, ButtonBar Control

see also example

Description

Specifies the state for a button of the ButtonBar control.

Usage

[*form!*]*StatusBar1*.**ButtonState(I)** = *State%*

Remarks

Each 2-state button on the ButtonBar can have the state RAISED (0) or DEPRESSED (-1). The ButtonState property array takes one of the following values to specify the state of a text-field.

Value	Short name	Description
0	RAISED	(Default) The button is raised.
1	DEPRESSED	The button is depressed.

It is not possible to change the state of a normal button (ButtonState = 0) on the ButtonBar. Using the ButtonState property allows the programmer to reset a 2-state button to it's up-state even if the GroupAllowAllUp property of the group is set to False.

Data Type

Integer (Enumerated)

The example shows a Form_Load that initializes some buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

ButtonType Property Array, ButtonBar Control

see also example

Description

Sets or retrieves the button-type for the button with the same index.

Usage

[*form!*]*ButtonBar1*.**ButtonType(I)** [= *ButtonType%*]

Remarks

The ButtonBar supports two different button-types. The ButtonType property array takes one of the following values to specify the type of a button.

Value	Short name	Description
0	BUTTON_NORMAL	(Default) The button reacts like a normal command-button.
1	BUTTON_2STATE	The button is a 2-state button. After the first click, the button remains depressed. After the second click the button returns to it's normal state. This behavior can be different if the A 2-state button is member of a group (by setting the ButtonGroup property). If the button is part of a group, it is raised if another member of the group is depressed. If the GroupAllowAllUp property of the group the button is a member of is set to False, the button remains depressed if you click on it a second time.

The index in the ButtonType property array is the same index in the all the other button-related property arrays (i.e. ButtonState, ButtonGroup, ButtonEnabled, Picture).

Data Type

Integer (Enumerated)

The example shows a Form_Load that initializes a couple of buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

ButtonEnabled Property Array, ButtonBar Control

see also example

Description

Sets or retrieves the enabled-state for the button with the same index.

Usage

[*form!*]*ButtonBar1*.**ButtonEnabled(I)** [= *setting%*]

Remarks

The ButtonEnabled property settings are:

Setting	Description
True	(Default) Allows the button to respond to events.
False	Prevents the object from responding to events.

This property allows buttons to be enabled or disabled at run time. For example, you can disable objects that don't apply to the current state of the application. When a button is disabled, the picture, specified by the Picture property, will be changed so that it looks disabled (all black pixels will be transformed to dark-gray) if no picture for the disabled state is specified.

Data Type

Integer (Boolean)

The example shows a Form_Load that initializes a couple of buttons on a ButtonBar control. To use this example create a form with one ButtonBar control and paste the code into the Declarations section of your form.

Events

The ButtonBar control supports the following events:

Click DragDrop DragOver

Methods

The ButtonBar control supports the following methods:

Drag	Move	Refresh	ZOrder

Properties

All the properties that apply to the ButtonBar control are listed in the following table. All properties that are marked with an asterisk (*) are only available at run-time.

Align	ButtonEnabled *	ButtonGroup *	ButtonMessage *
ButtonState *	ButtonType *	Enabled	GroupAllowAllUp *
Height	hWndStatusBar *	Index	Left
LeftMargin	Name	OutlineChildren	Picture *
PictureDisabled *	SpaceAfter *	StatusField *	Top
Visible	Width		

Note the ButtonEnabled, ButtonGroup, ButtonMessage, ButtonState, ButtonType, GroupAllowAllUp, hWndStatusBar, Picture, PictureDisabled, SpaceAfter and StatusField properties are only available at run-time. Name is the default property for the ButtonBar control.

The ButtonBar Custom Control

Properties Methods Events

Description

The Microsoft Visual Basic programming system for Windows comes with a large set of 3D controls. Unfortunately, a 3D status-bar, as used in almost every MS-Windows application is lacking. Therefore the ButtonBar Custom Control was designed. This control allows you to create a very versatile button bar for all your applications.

File Name

TOOLBARS.VBX

Object Type

ButtonBar

Toolbox Icon

Remarks

The ButtonBar allows an application to display a button-bat at the top of a form. The button-bar has 20 fully configurable buttons. To get the ButtonBar working, the only thing you have to do is specify a bitmap for the up-position of each button. The ButtonBar control automatically creates the different bitmaps for the down and the two disabled states (up and down) of the button. It is also possible to connect the button-bar to the status-bar and specify messages for each button to be shown when the button is selected.

Usage

To use the ButtonBar, perform the following steps:

1. Add the Toolbars custom control to your project. The StatusBar and the ButtonBar icons will appear in the Visual Basic tool-palette.

2. In the Form_Load event procedure for the form, set the Picture (and eventually the PictureDisabled) and ButtonType properties for the number of buttons you want to display.

3. Add code to respond to a Click on a certain button in the ButtonBar.

Warning

The total number of ButtonBar controls on one system is limited to 20.

Distribution Note When you create and distribute applications that use the ButtonBar control, you should install the file TOOLBARS.VBX in the customer's Microsoft Windows \ SYSTEM subdirectory.