

Using ActiveX Slider Control

1. Overview

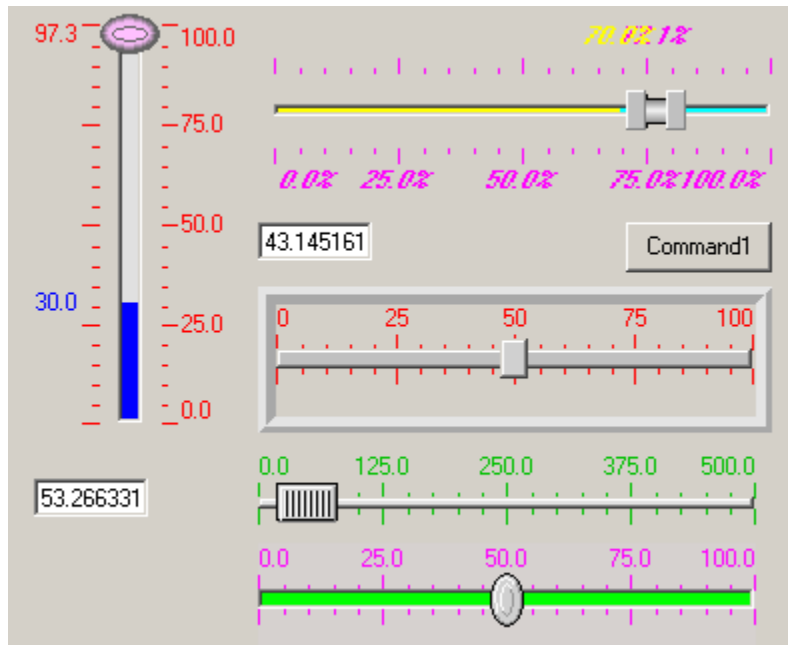
The slider is a versatile input/output instrument graphic component which has been widely used in the industrial applications. In general, the slider is used to graphically (visually) adjust or setup the values of different control variables in the SCADA systems or Human-Machine-Interface (HMI) systems. Of course, it can be applied in other fields (like Game Simulators, scientific analytic applications) which need to setup options or adjust control values.

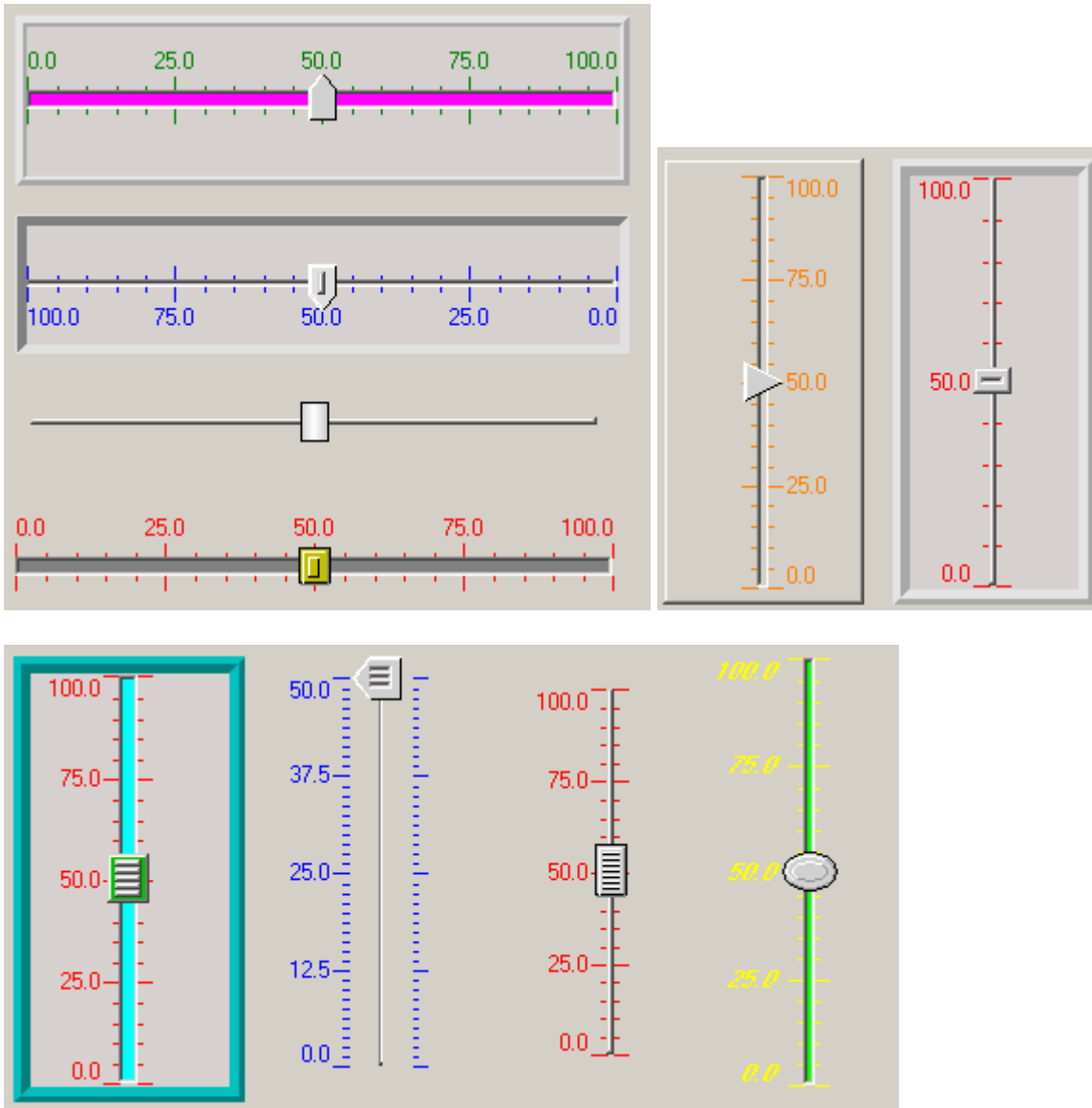
There are two types of ActiveX slider controls, i.e., the Slider ActiveX Control and the Switch Slider ActiveX Control, where the Slider ActiveX Control is designed for continuous variables such as pressure control setpoints or velocity control setpoints, and the Switch Slider ActiveX Control is designed for discrete event variables such as multiple options. Both controls take almost same look-and-feel. By using these ActiveX controls, User Interfaces can be greatly enhanced with an outstanding look. Many powerful properties are provided for people to configure the borders, scale, marker, indicator and other parts of the sliders, then they can design their own look-and-feel. Mouse operations are the basic feature for user to control the slider, meanwhile the interface methods are provided for developers to programmatically to control the sliders.

The sliders can be as well used as gauges of indicators to display the values or statuses.

2. Slider ActiveX Control

Interfaces



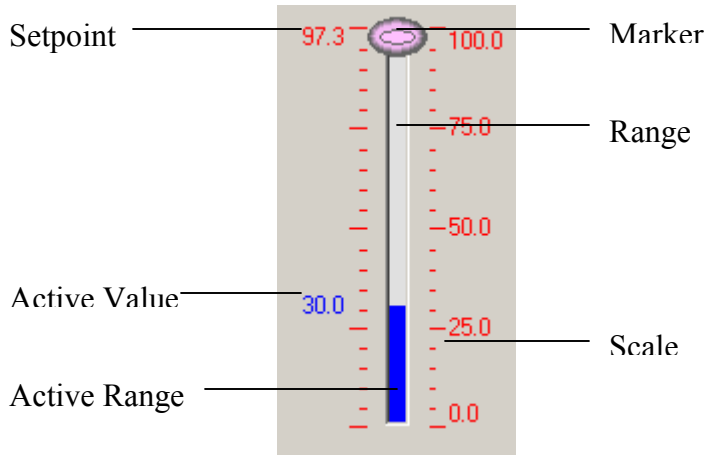


The designers can use the interface properties and methods to implement many types of the sliders as their request, the above pictures show some styles which can be configured.

Properties:

Property *ActualRangeVisible* As Boolean

ActualRangeVisible specifies whether the range (from Min value to actual value) in the Range field is filled with color of *RangeActualColor* or not.



Property *BkColor* As *OLE_COLOR*

BkColor specifies the background color of the control if *BackStyle*=*BS_Opaque*.

Property *BackStyle* As *BackStyleType*

If *BackStyle*=*BS_Opaque*, *BkColor* is used to render its background; If *BackStyle*=*BS_Transparent*, the background color of the container of the control is used to render the control background.

Property *BorderDarkColor* As *OLE_COLOR*

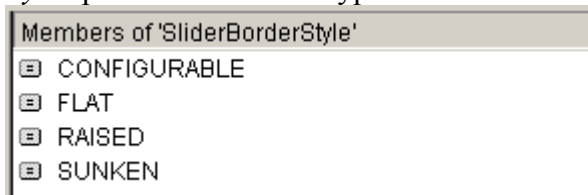
This property is used to configure the color of the border side which is hidden from the light. It is used in conjunction with *BorderLightColor* to create the 3D-Look border.

Property *BorderLightColor* As *OLE_COLOR*

This property is used to configure the color of the border side which faces the light. It is used in conjunction with *BorderDarkColor* to create the 3D-Look border.

Property *BorderStyle* As *SliderBorderStyle*

BorderStyle specifies the border type. There are four options in the *SliderBorderStyle*,



- FLAT* : No any border;
- RAISED*: The raised border is drawn;
- SUNKEN*: The sunken border is drawn;
- CONFIGURABLE*: The border is drawn using *BorderDarkColor*, *BorderLightColor*, *InnerBorderLen*, *OuterBorderLen* four properties.

Property *Enabled* As *Boolean*

Specifies whether the window of the ActiveX control is enabled or disabled. If disabled, there is no window message or event triggered by the control.

Property Font As IFontDisp

Font defines the standard windows font property.

Property hWnd As Long

Retrieves the window handler of the control. It's a read-only property.

Property InnerBorderLen As Integer

This property is used to specify the length of the inner border.

Property InverseDirection As Boolean

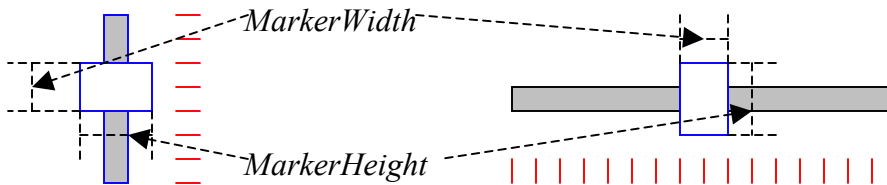
InverseDirection specifies the Min—Max direction. If *InverseDirection*=*FALSE*, Min-Max direction is from left to right or from bottom to top; otherwise Min-Max direction is from right to left or from top to bottom.

Property MarkerColor As OLE_COLOR

MarkerColor specifies the color of the Marker.

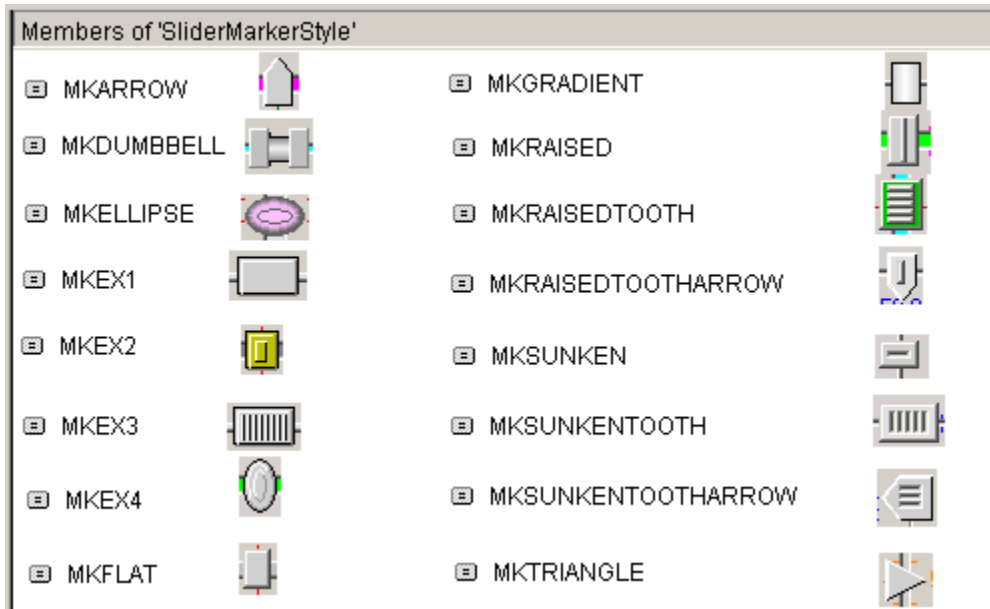
Property MarkerHeight As Integer

MarkerHeight specifies the height of the Marker.



Property MarkerStyle As SliderMarkerStyle

MarkerStyle specifies the shape of the Marker.



Property MarkerWidth As Integer

MarkerHeight specifies the width of the Marker.

Property MaxVal As Double

MaxVal specifies the maximum value of the scale.

Property MinVal As Double

MinVal specifies the minimum value of the scale.

Property OuterBorderLen As Integer

This property is used to specify the length of the outer border.

Property PercentageScale As Boolean

This property specifies whether the values are displayed in the form of Percentage(xxx%) or not.

Property RangeActualColor As OLE_COLOR

RangeActualColor specifies the color of the actual range.

Property RangeBkColor As OLE_COLOR

RangeBkColor specifies the color of the slider range.

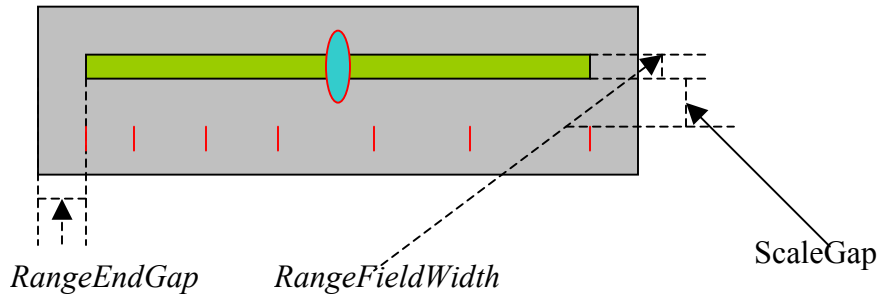
Property RangeBorderStyle As SliderBorderStyle

RangeBorderStyle specifies the border style of the range. But *CONFIGURABLE* is not supported for this property, hence if *CONFIGURABLE* is chosen, it has the same effect of *FLAT*.



Property RangeEndGap As Integer

RangeEndGap specifies the gap length between the border of the slider and border of the Slider Range in the Min-Max direction.



Property RangeFieldWidth As Integer

RangeFieldWidth specifies the width of the range.

Property ScaleGap As Integer

ScaleGap specifies the gap length between the scale and the border of the range

Property ScalePrecision As Integer

ScalePrecision specifies the decimal digit number. (“5.000” implies *ScalePrecision*=3)

Property ScaleSideChanged As Boolean

ScaleSideChanged=*TRUE* means the scale is drawn in the bottom side or right side. Otherwise the top side or the left side.

Property ScaleVisible As Boolean

ScaleVisible specifies whether the scale is visible or not.

Property ShowActualValue As Boolean

ShowActualValue specifies whether the actual value is displayed or not.

Property ShowSetpointValue As Boolean

ShowSetpointValue specifies whether the setpoint value is displayed or not.

Property TickerNum As Integer

Property SubTickerNum As Integer

TickerNum and *SubTickerNum* are used to define the scale ticker number and sub scale ticker number.

Property TextColor As OLE_COLOR

TextColor specifies the color of the scale.

Property Vertical As Boolean

Vertical specifies the slider is vertical or horizontal. If *Vertical* =*TRUE*, it’s vertical, otherwise it’s horizontal.

Functions or Methods:

Function GetSetpoint() As Double

Retrieve the setpoint of the slider.

Sub UpdateActualValue(Val As Double)

Update the actual value of the slider.

Sub UpdateSetpoint(Val As Double)

Update the setpoint of the slider.

Events:

Event Click()

When the chart is clicked, the event is triggered to the container. Fired when the control captures the mouse, any **BUTTONUP** (left, middle, or right) message is received, and the button is released over the control. The **MouseDown** and **MouseUp** events occur before this event.

Event DblClick()

When the chart is double clicked, the event is triggered to the container. Similar to **Click** but fired when a **BUTTONDBLCLK** message is received.

Event KeyDown(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYDOWN** or **WM_KEYDOWN** message is received. *nChar* specifies the virtual key code of the given key. For a list of standard virtual key codes, see *Winuser.h*; *nRepCnt* specifies the repeat count, that is, the number of times the keystroke is repeated as a result of the user holding down the key; *nFlags* specifies the scan code, key-transition code, previous key state, and context code. For details, please refer to MSDN.

Event KeyUp(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYUP** or **WM_KEYUP** message is received. Please refer to *KeyDown* event.

Event MouseDown(x As Long, y As Long)

Fired if any **BUTTONDOWN** (left, middle, or right) is received. The mouse is captured immediately before this event is fired. *x* and *y* represent the corresponding coordinate values in the control, the origin is at the left-top point of the control.

Event MouseMove(x As Long, y As Long)

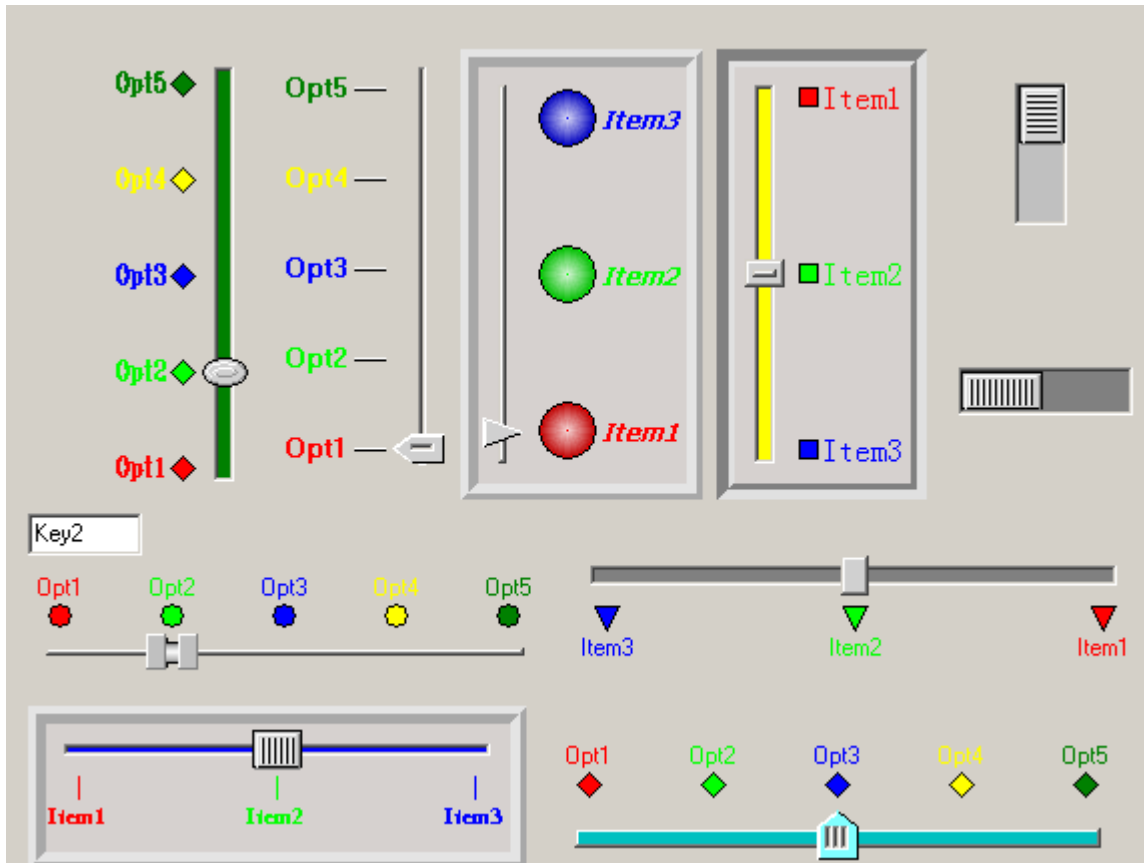
Fired when a **WM_MOUSEMOVE** message is received.

Event MouseUp(x As Long, y As Long)

Fired if any **BUTTONUP** (left, middle, or right) is received. The mouse capture is released before this event is fired.

3. Switch Slider ActiveX Control

Interfaces



The designers can use the interface properties and methods to implement many types of the switch sliders, the above pictures show some styles which can be configured.

Properties:

Property BkColor As OLE_COLOR

BkColor specifies the background color of the control if *BackStyle*=*BS_Opaque*.

Property BackStyle As BackStyleType

If *BackStyle*=*BS_Opaque*, *BkColor* is used to render its background; If *BackStyle*=*BS_Transparent*, the background color of the container of the control is used to render the control background.

Property BorderDarkColor As OLE_COLOR

This property is used to configure the color of the border side which is hidden from the light. It is used in conjunction with *BorderLightColor* to create the 3D-Look border.



Property *BorderLightColor* As *OLE_COLOR*

This property is used to configure the color of the border side which faces the light. It is used in conjunction with *BorderDarkColor* to create the 3D-Look border.

Property *BorderStyle* As *SliderBorderStyle*

BorderStyle specifies the border type. Please refer to the same property description of the Slider Control in Section 2.

Property *Enabled* As *Boolean*

Specifies whether the window of the ActiveX control is enabled or disabled. If disabled, there is no window message or event triggered by the control.

Property *Font* As *IfontDisp*

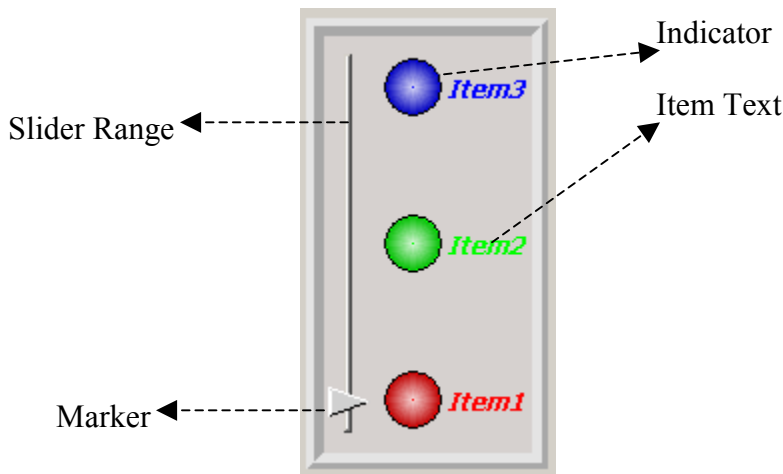
Font defines the standard windows font property.

Property *hWnd* As *Long*

Retrieves the window handler of the control. It's a read-only property.

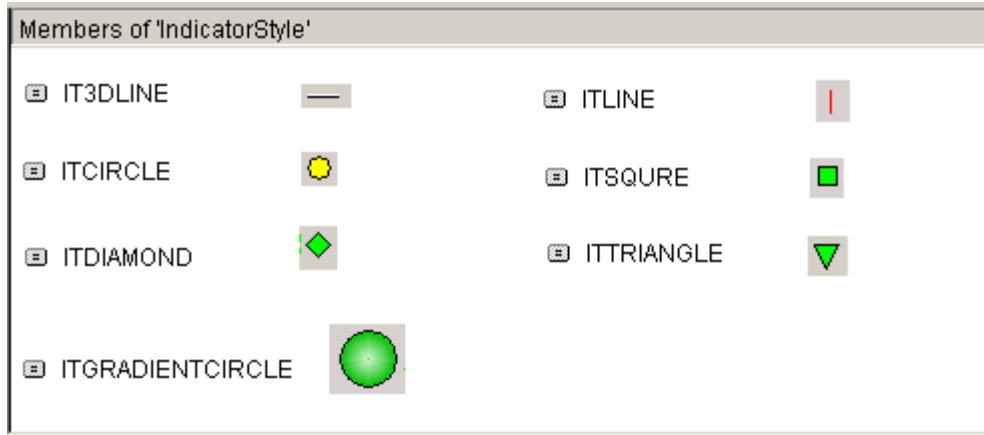
Property *IndicatorSize* As *Integer*

IndicatorSize specifies the size of the indicator. Indicator is used to represent the each individual options.



Property *IndicatorType* As *IndicatorStyle*

IndicatorType specifies the style of the indicator. The options are listed as follows,



Property *InnerBorderLen* As Integer

This property is used to specify the length of the inner border.

Property *InverseDirection* As Boolean

InverseDirection specifies the item layout (Item1 ... Item N) direction. If *InverseDirection* = *FALSE*, the direction is from left to right or from bottom to top; otherwise the direction is from right to left or from top to bottom.

Property *MarkerColor* As OLE_COLOR

MarkerColor specifies the color of the Marker.

Property *MarkerHeight* As Integer

MarkerHeight specifies the height of the Marker. Please refer to the same property description of the Slider Control in Section 2.

Property *MarkerStyle* As SliderMarkerStyle

MarkerStyle specifies the shape of the Marker. Please refer to the same property description of the Slider Control in Section 2.

Property *MarkerWidth* As Integer

MarkerWidth specifies the width of the Marker. Please refer to the same property description of the Slider Control in Section 2.

Property *OuterBorderLen* As Integer

This property is used to specify the length of the outer border.

Property *RangeBkColor* As OLE_COLOR

RangeBkColor specifies the color of the slider range.

Property *RangeBorderStyle* As SliderBorderStyle

RangeBorderStyle specifies the border style of the range. But *CONFIGURABLE* is not supported for this property, hence if *CONFIGURABLE* is chosen, it has the same effect of *FLAT*.

Property RangeEndGap As Integer

RangeEndGap specifies the gap length between the border of the slider and border of the Slider Range in the item layout direction. Please refer to the same property description of the Slider Control in Section 2.

Property RangeFieldWidth As Integer

RangeFieldWidth specifies the width of the range. Please refer to the same property description of the Slider Control in Section 2.

Property ScaleGap As Integer

ScaleGap specifies the gap length between the scale and the border of the range. Please refer to the same property description of the Slider Control in Section 2.

Property ScaleSideChanged As Boolean

ScaleSideChanged=TRUE means the scale is drawn in the bottom side or right side. Otherwise the top side or the left side.

Property ScaleVisible As Boolean

ScaleVisible specifies whether the scale is visible or not.

Property Vertical As Boolean

Vertical specifies the slider is vertical or horizontal. If *Vertical =TRUE*, it's vertical, otherwise it's horizontal.

Functions or Methods:

Sub AddItem(strKey As String, strName As String, color As OLE_COLOR)

Add one option item to the switch slider, where *strKey* is the key to mark the item, *strName* is item text, and *color* is color of item indicator and item text.

Function GetSetpoint() As String

Retrieve the chosen item of the slider. The return is the key of the option items.

Sub RemoveItem(strKey As String)

Remove the option item specified by *strKey* from the option item collection.

Sub UpdateSetpoint(strKey As String)

Change the chosen option item to the item specified by *strKey* of the slider.

Events:

Event Click()

When the chart is clicked, the event is triggered to the container. Fired when the control captures the mouse, any **BUTTONUP** (left, middle, or right) message is received, and the button is released over the control. The **MouseDown** and **MouseUp** events occur before this event.

Event DbClick()

When the chart is double clicked, the event is triggered to the container. Similar to Click but fired when a **BUTTONDBLCLK** message is received.

Event KeyDown(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYDOWN** or **WM_KEYDOWN** message is received. *nChar* specifies the virtual key code of the given key. For a list of standard virtual key codes, see Winuser.h ; *nRepCnt* specifies the repeat count, that is, the number of times the keystroke is repeated as a result of the user holding down the key; *nFlags* specifies the scan code, key-transition code, previous key state, and context code. For details, please refer to MSDN.

Event KeyUp(nChar As Long, nRepCnt As Long, nFlags As Long)

Fired when a **WM_SYSKEYUP** or **WM_KEYUP** message is received. Please refer to *KeyDown* event.

Event MouseDown(x As Long, y As Long)

Fired if any **BUTTONDOWN** (left, middle, or right) is received. The mouse is captured immediately before this event is fired. *x* and *y* represent the corresponding coordinate values in the control, the origin is at the left-top point of the control.

Event MouseMove(x As Long, y As Long)

Fired when a **WM_MOUSEMOVE** message is received.

Event MouseUp(x As Long, y As Long)

Fired if any **BUTTONUP** (left, middle, or right) is received. The mouse capture is released before this event is fired.

Event ValueChange(strKey As String)

Fired if the setpoint value of the switch slider has been changed by calling *UpdateSetpoint* or moving the mouse to change the setpoint value.