

Using ActiveX Switch Control

1. Overview

The switch is an instrument graphic component that is a TRUE/FALSE, YES/NO, or ON/OFF toggle switch, and it has been widely used in the industrial applications. The switch is a multi-purpose and highly customizable graphic component that can be designed for a toggle switch interface. In general, the switch is used to graphically (visually) change the binary value of different switch variables in the SCADA systems or Human-Machine-Interface (HMI) systems. A Switch ActiveX Control can help people to configure its layout to look like toggle switches on all kinds of electronic and industrial equipment. For other multi-option switches, the Switch Slider ActiveX Control and Switch Knob ActiveX Control can deal with these kinds of functions.

Switch ActiveX Control is designed for BOOL variables such as toggle options. By using this ActiveX controls, User Interfaces can be greatly enhanced with an outstanding look. Many powerful properties are provided for people to configure the borders, face, and other parts of the switch, then they can design their own look-and-feel. Mouse operation is the basic feature for user to control the switch, meanwhile the interface methods are provided for developers to programmatically to control the switch.

2. Interfaces



The designers can use the interface properties and methods to implement many types of the switches, the above picture shows some styles which can be configured.



Properties:

Property *b3DText* As Boolean

b3DText specifies whether the text of switch is shown as 3D or not.

Property *bCircleStatus* As Boolean

bCircleStatus determines whether the status indicator is shown in a circle shape or a rectangle shape when *bShowStatus* is TRUE.

Property *bForeGradientToggle* As Boolean

This property only takes effect when *SwitchType*= *GROOVE_TOGGLE*. If *bForeGradientToggle*=*TRUE*, the groove face is drawn gradiently.



bForeGradientToggle=FALSE



bForeGradientToggle=TRUE

Property *BkColor* As OLE_COLOR

BkColor specifies the color of the switch background 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 *BorderLen* As Integer

BorderLen specifies the width of the border. The following is *BorderLen*=10.



Property *bShowStatus* As Boolean

bShowStatus specifies whether the status indicator is visible or not.

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 *ButtonHeight* As Integer

ButtonHeight specifies the height of the Button. The following figures show different *ButtonHeights*.



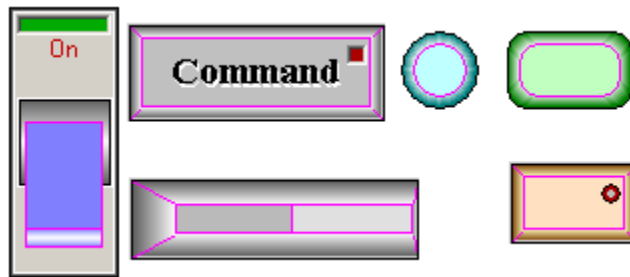
ButtonHeight=2 *ButtonHeight=8* *ButtonHeight=12*
But this property does not apply to *SWITCH_TOGGLE* and *USER_IMAGE*.

Property *bVerticalToggle* As Boolean

This property specifies whether the toggle switch is drawn vertically or horizontally. And it only applies to *GROOVE_TOGGLE* and *WEDGE_TOGGLE*.

Property *EdgeColor* As OLE_COLOR

EdgeColor specifies the line color of the edge of the face. Suppose *EdgeColor* is Pink,



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 *ForeColor* As OLE_COLOR

ForeColor defines the face color of switches. **But if *SwitchType* = *GROOVE_TOGGLE* and *bForeGradientToggle*=*TRUE*, the face will be drawn in gradient mode from *SideDarkColor* to *SideLightColor*.**

Property *GradientRate* As Single

GradientRate defines the gradient degree of drawing.

Property *hWnd* As Long

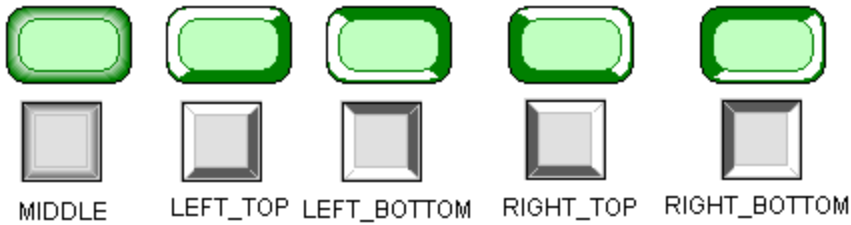
Retrieves the window handler of the control. It's a read-only property.

Property *Light3D* As LightDirection

Light3D defines the light source direction.



Members of 'LightDirection'



Property ImageStyle As ImagePaintStyle

Specifies the render style of the picture when drawing the picture in the plot area. There are three definitions,

IPS_ORIGINALSIZE =0, // the picture is drawn at its original size on the center of plot
// area.

IPS_TEXTURE=1, // the picture is drawn at its original size on the plot area from the
//left-/top point, if the size picture is smaller than the size of the plot area,
//more pictures are drawn to make a texture background which is //made
//of multiple copies of the picture.

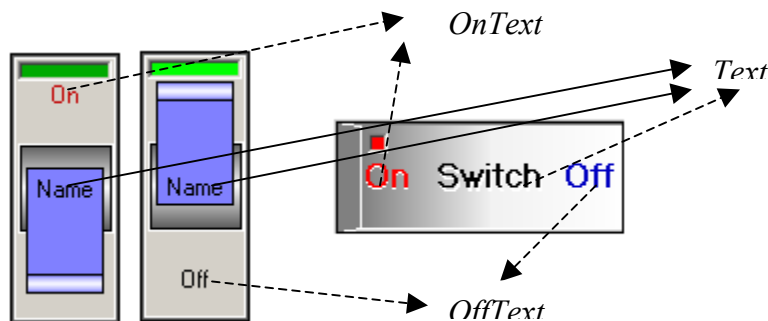
IPS_AUTOSIZE=2 //the picture will be first resized to the size of the plot area, and then
//drawn on the plot area.

Property OffPicture As IPictureDisp

OffPicture defines the image drawn on the face of the control if *Value* (See *Value* property) is *FALSE* and *SwitchType*=*USER_IMAGE*.

Property OffText As String

OffText defines the OFF text on the OFF side of the switch. This property only applies to *GROOVE_TOGGLE*, *WEDGE_TOGGLE* and *SWITCH_TOGGLE*.



Property OffTextColor As OLE_COLOR

OffTextColor defines the color of the *OffText*.

Property OnPicture As IPictureDisp

OnPicture defines the image drawn on the face of the control if control if *Value* (See *Value* property) is *TRUE* and *SwitchType*=*USER_IMAGE*.



Property OnText As String

OnText defines the ON text on the ON side of the switch. This property only applies to *GROOVE_TOGGLE*, *WEDGE_TOGGLE* and *SWITCH_TOGGLE*.

Property OnTextColor As OLE_COLOR

OnTextColor defines the color of the *OnText*.

Property RoundHeight As Integer

RoundHeight defines the Height of the corner ellipses of the Round Rectangle if *SwitchType* is *ROUND_RECTANGLE*.

Property RoundWidth As Integer

RoundWidth defines the width of the corner ellipses of the Round Rectangle if *SwitchType* is *ROUND_RECTANGLE*.

Property SideDarkColor As OLE_COLOR

SideDarkColor defines the color of the switch sides that are hidden from the light.

Property SideLightColor As OLE_COLOR

SideLightColor defines the color of the switch sides that are faced to the light.

Property Status As Boolean

Status defines the switch status, i.e., ON or OFF, TRUE or FALSE, OPEN or CLOSE.

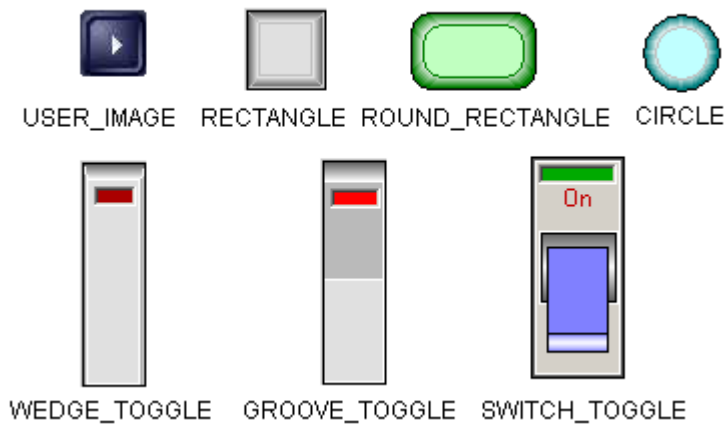
Property StatusColor As OLE_COLOR

If the status is shown on the control (*bShowStatus=TRUE*), *StatusColor* specifies the color of the status indicator.

Property SwitchType As SwitchShape

SwitchType defines the shape and the style of the switch.

Members of 'SwitchShape'



Property Text As String

Text defines the caption of the switch.

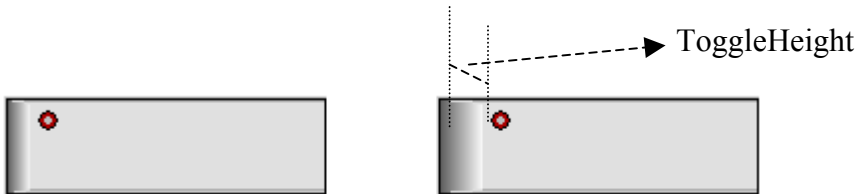


Property TextColor As OLE_COLOR

TextColor defines the color of the Text

Property ToggleHeight As Integer

ToggleHeight defines the height of the toggle change.



Property Value As Boolean

Value defines the switch position, TRUE means the switch is at ON position, FALSE means the switch is at OFF position. *Status* is the actual status, and *Value* is setpoint.

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.

Event ValueChange(newValue As Boolean)

When the switch position is changed, then the event is triggered with new position value *newValue* (*TRUE* means the switch is at ON position, *FALSE* means the switch is at OFF position).

Event StatusChange(newStatus As Boolean)

When the switch status is changed, then the event is triggered with new status *newStatus*.