



Using ActiveX Counter Control

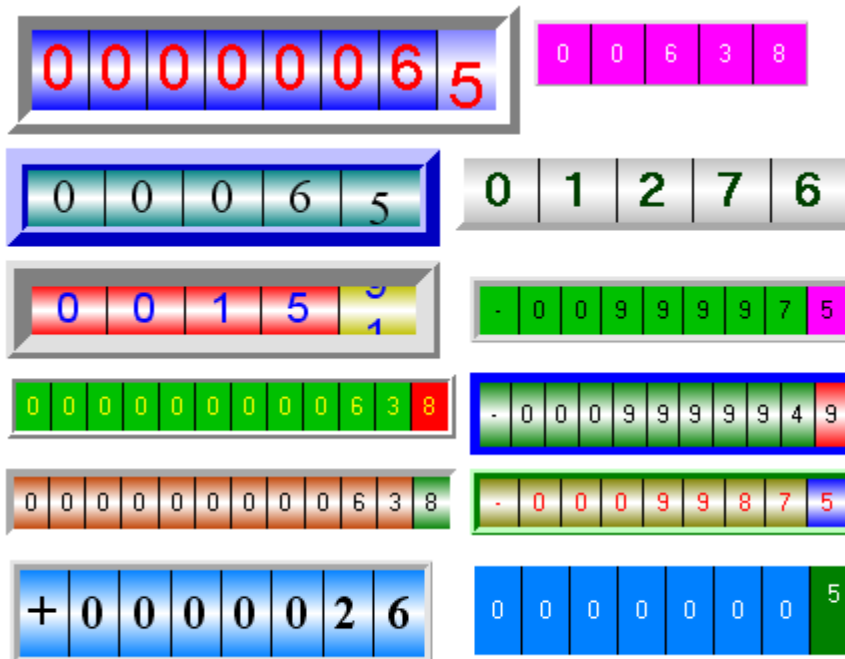
1. Overview

There are so many cases we need to use counters to record all kinds of numeric data. The odometer is a common example of the numeric display, which we can see in any automobile or aircraft system. This ActiveX control can be used in many applications such as Factory Instrumentation Readouts, Automobile trip odometer, Aircraft digital display system and other numeric display cases. It can even be used in the Web pages which shows how many people have visited these pages or be used as Hit counters.

This ActiveX control is very powerful for designers to configure different kinds of odometers which has the ability to roll numbers for realistic odometer-like visual effect. The fonts size, text color, border style, Inner border color and length, outer border color and length, number of digits, gradient display or not, highlight change or not, and other features can be configured by the designers at design time or run time. All these properties will be addressed in the following sections.

2. Interfaces

The designers can use the properties and methods provided by the control to implement many kinds of the counters or odometers, the following picture shows some styles which can be configured by setting up these properties.

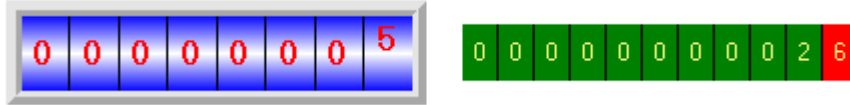




Properties:

Property *bGradient* As Boolean

If *bGradient* is set to *true*, then the background color of digital numbers will be shown in gradient style, otherwise, it will be shown in normal style. In the following figure, *bGradient=TRUE* in the left counter and *bGradient=FALSE* in the right counter



Property *BkColor* As OLE_COLOR

This property is used to configure the background color of digital numbers if “*BackStyle* = *BS_Opaque*”. In the above figure, *BkColor* is Blue in the left counter and *BkColor* is green in the right counter.

Property *BackStyle* As BackStyleType

If *BackStyle=BS_Opaque*, then the background is rendered using *BkColor*; If *BackStyle=BS_Transparent*, then the background is rendered using the ambient background color of the container of the control.

Property *BorderDarkColor* As OLE_COLOR

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

Property *DigitNum* As Integer

This property is used to configure how many digit numbers to be shown in the counter.

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

Defines the standard font property to display the scale texts and the value.

Property *HighlightColor* As OLE_COLOR

In some application cases, people like to highlight the most right digital number because of its frequent change. This property allows designers to highlight the frequent change digit using this color. This property should be used in conjunction with *NoHighlight*



property. (Only when *NoHighlight* is false, the *HighlightColor* will shown as back ground color of most right digit).

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 *NoHighlight As Boolean*

The default value of this property is *true*, when this property is set to *false*, then the *Highlightcolor* is used to mark the most right digit.

Property *OuterBorderLen As Integer*

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

Property *Precision As Integer*

Defines how many decimal digits.

Property *ShowSign As Boolean*

The counter value can be positive (+) or negative (-). This property specifies whether to show the sign of the number or not. If the value is negative, the (-) sign is always displayed.

Property *TextColor As OLE_COLOR*

This property is used to specify the color of all the numeric digits.

Methods & Functions:

Function *GetValue() As Long*

Retrieve the counter value.

Sub *Reset(InitValue As Long)*

This function is used to reset the numeric digital value to the *InitValue*. By the way, when we use *UpdateValue* to update the numeric digital value, if the value exceeds the maximum digit number, it will be automatically reset to 0.

Sub *UpdateValue(dblDelta As double)*

This method is used to update the numeric digits. In the counter implementation, internally there is variable which records the current counter value, if this method is called, the counter will add the *dblDelta* to that variable, and then update the numeric display. If *dblDelta* has more decimal precision than "*Precision*", then the counter will roll the last right digit from current number to next number (0→1→2 ...9→0, or 9→8→7...→1→0→9). The "Rolling" of value display can be seen if *dblDelta* has more decimal precision than "*Precision*".



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.