



Using ActiveX Digit LED Control

1. Overview

The Digit LED allows users to create realistic alpha-numeric display using the seven-segment LED standard. This control can be used in many applications such as Factory Instrumentation Readouts, Digital Clock, Calculator, Countdown Clock, Digital Counter and other numeric displays.

This ActiveX product is very powerful for designers to configure different kinds of numeric displays which present visual realistic effect. There are a rich group of the properties which enable people to design many kinds of numeric displays to meet their requirements. All these properties will be addressed in the following sections.

2. Interfaces

The designers can use the interface properties and methods to implement many kinds of numeric display, the following picture shows some styles which can be configured.

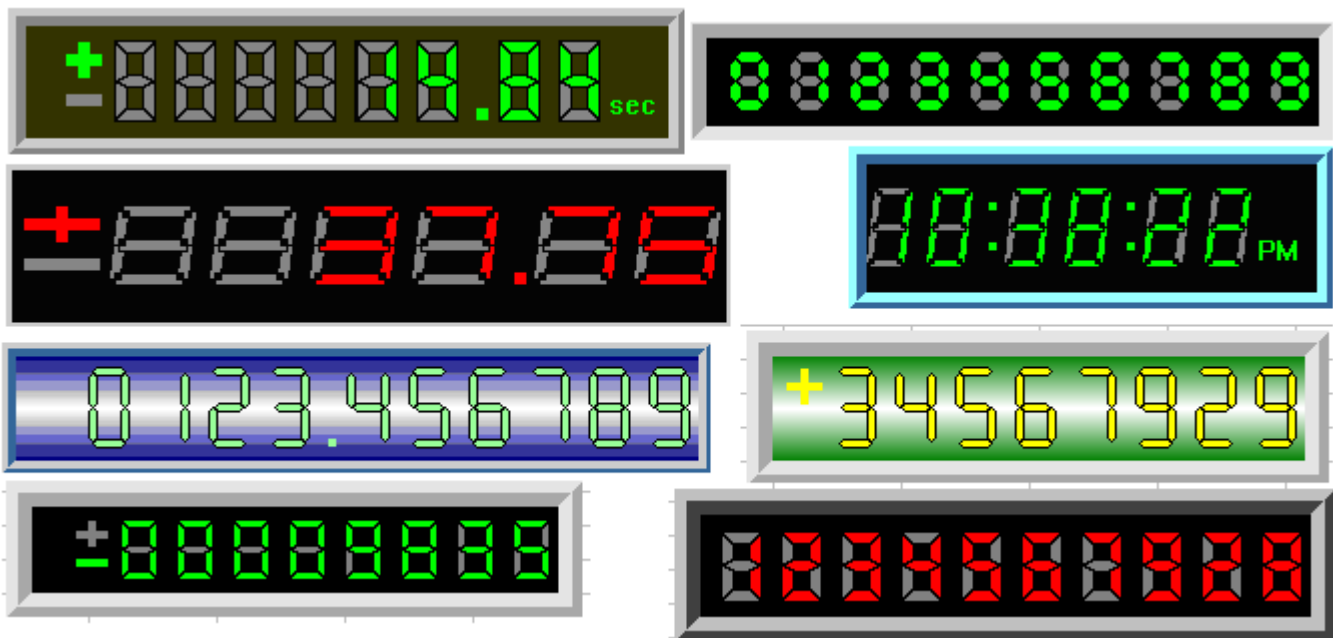


Figure 1

Properties

Property *bFlashing* As Boolean

bFlashing specifies whether the LEDs flash or not. *TRUE* means flashing, *FALSE* means no flashing.

Property *bGradient* As Boolean

bGradient = *TRUE* means to draw the background in a gradient manner, otherwise in normal manner (In Figure 1, the LED panels of the third row are in gradient mode, others are in normal mode).

Property *bInclinedShow* As Boolean

bInclinedShow = *TRUE*, the LEDs will be drawn in an small oblique angle (See the LEDs of the second row in the Figure 1); *bInclinedShow* = *FALSE*, the LEDs will be drawn right vertically .



Property *bkColor* As *OLE_COLOR*

bkColor specifies the background color of the LED panel if “*BackStyle* = *BS_Opaque*”.

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 *bNoDigitFrame* As *Boolean*

bNoDigitalFrame=*TRUE* means to show the digit LED frames; *bNoDigitalFrame*=*FALSE* means no digit LED frame. (In Figure 1, *bNoDigitalFrame*=*FALSE* in the LED controls of the third row, *bNoDigitalFrame*=*TRUE* in others.)

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 *bShowAllZero* As *Boolean*

This property is used to specify whether to show all the zero digits or not. (See Figure 2, In the top LED panel, *bShowAllZero*=*TRUE*; In the bottom LED panel, *bShowAllZero*=*FALSE*)

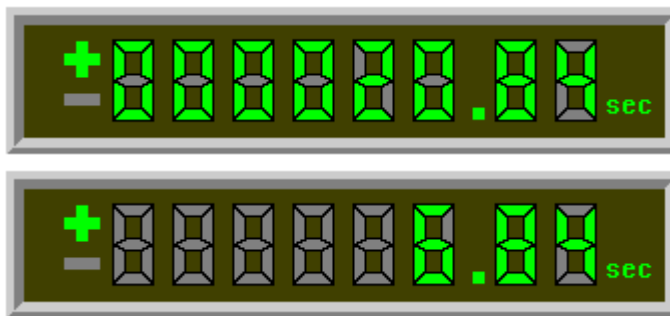


Figure 2

Property *bShowUnit* As *Boolean*

bShowUnit=*TRUE* means to show unit (See Figure 2, unit is sec), *bShowUnit*=*FALSE* means not to show unit.

Property *bSignShow* As *Boolean*

Sometimes the value may be negative or positive, so *bSignShow* is used to determine to show the sign or not. *bSignShow* does not take effect if *bTimeStyle*=*TRUE*.

Property *bTimeStyle* As *Boolean*

If we want use the LED control as a digital time clock, we can set *bTimeStyle*=*TRUE*.



Figure 3

Property *DecimalPrecision As Integer*

Sometimes we like to show the double or float value which should have decimal part. *DecimalPrecision* is used to determine how many decimal digits to be displayed. If the *DecimalPrecision* is greater than zero, then the decimal point will be displayed.

Property *digitColor As OLE_COLOR*

digitColor specifies the color of the digit.

Property *digitFontWidth As Integer*

digitFontWidth is used to specify the width of the digit LED segment. (In Figure 3, *digitFontWidth*=4 in the left control, and *digitFontWidth*=2 in the right control)

Property *digitFrameColor As OLE_COLOR*

digitFrameColor specifies the color of the frames of digit LED segment if *bNoDigitFrame*=*FALSE*.

Property *digitHeight As Integer*

Property *digitWidth As Integer*

digitHeight and *digitWidth* are used to specify the height and the width of a LED digit.

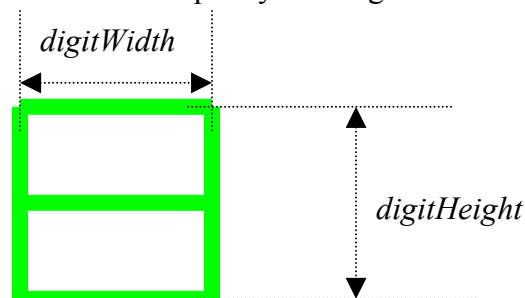


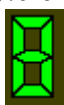
Figure 4

Property *digitNumber As Integer*

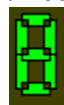
DigitNumber specifies how many digits which will be displayed.

Property *DigitType As LEDStyle*

LEDStyle enumerates three types of LED segments, i.e.,



Ladder = 0



TShape = 1



Rhomb = 2

DigitType is used to specify which LED segment style the LED control will take.



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 FlashingDuration As Long

FlashingDuration defines the flashing period if *bFlashing=TRUE*

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 OuterBorderLen As Integer

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

Property unit As String

This property defines the unit of digit value.

Property unitLength As Integer

This property specifies the length the unit takes up.

Property Value As Double

Set or retrieve the displayed value.

Methods

Sub UpdateTime(hour As Integer, minite As Integer, second As Integer, b24HourStyle As Boolean)

UpdateTime is used update the shown time, where *hour*, *minute* and *second* specify the corresponding shown hours, minutes and seconds according to "hh:mm:ss" format. *b24HourStyle* is used to determine the display takes 24 hour style or 12 hour+AM/PM style.

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.

EventKeyUp(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.

EventMouseDown(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.

EventMouseMove(x As Long, y As Long)

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

EventMouseUp(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.