



## Using ActiveX Temperature Control

### 1. Overview

The Temperature ActiveX control allows users to show the temperature in a visual meter. This control can be used in heats control and temperature monitor system or other monitoring systems. This ActiveX product is very powerful for designers to configure different kinds of temperature meters. There are a rich group of the properties which enable people to design any temperature meter 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 temperature meters, the following picture shows some styles which can be configured.

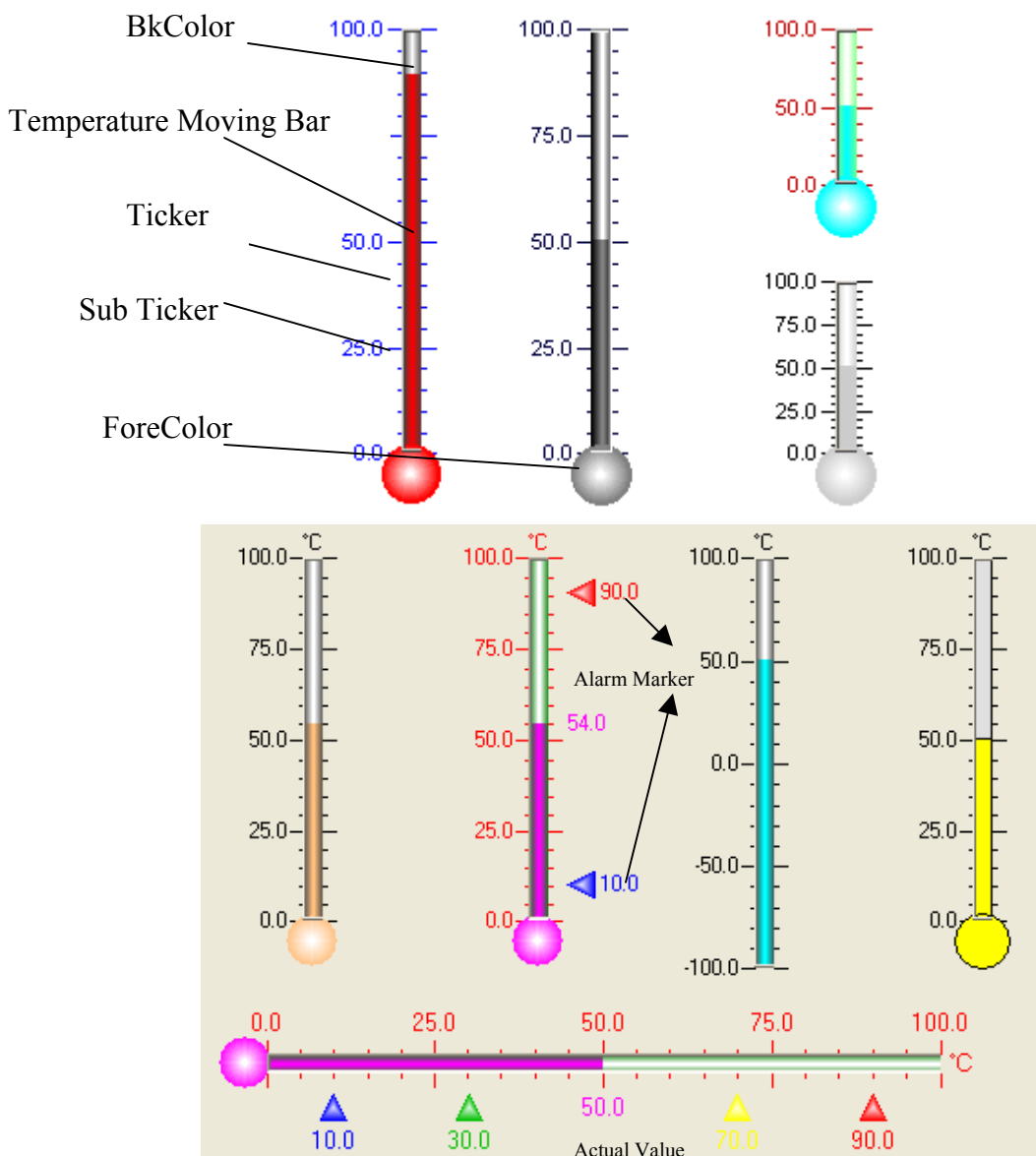


Figure 1



Property AlarmHighColor As OLE\_COLOR

Property AlarmMax As Double

If the temperature is higher than the *AlarmMax* with *bAlarm=TRUE*, then the color of the temperature moving bar will change to color specified by *AlarmHighColor*.

Property AlarmLowColor As OLE\_COLOR

Property AlarmMin As Double

If the temperature is lower than the *AlarmMin* with *bAlarm=TRUE*, then the color of the temperature moving bar will change to color specified by *AlarmLowColor*.

Property bAlarm As Boolean

*bAlarm* is used to determine whether to change the temperature color when the temperature exceeds the range (*AlarmMin*, *AlarmMax*). Meanwhile it will trigger the Event *TemperatureHighAlarm* and *TemperatureLowAlarm* if the alarm status changes. Otherwise the control will ignore all alarm properties.

Property ShowAlarmMarker As Boolean

Specifies whether to show the alarm markers or not (including the high alarm marker and the low alarm marker) which are used to mark the high alarm temperature setpoint (*AlarmMax*) and the low alarm temperature setpoint (*AlarmMin*) in the temperature meter bar.

Property ShowAlarmValue As Boolean

Specifies whether to show the alarm values or not (including the high alarm setpoint (*AlarmMax*) and the low alarm setpoint (*AlarmMin*)).

Property WarningHighColor As OLE\_COLOR

Property WarningMax As Double

If the temperature is higher than the *WarningMax* with *bWarning=TRUE* and *bAlarm=FALSE*, or the temperature is higher than the *WarningMax* and less than *AlarmMax* with *bWarning=TRUE* and *bAlarm=TRUE*, then the color of the temperature moving bar will change to color specified by *WarningHighColor*.

Property WarningLowColor As OLE\_COLOR

Property WarningMin As Double

If the temperature is lower than the *WarningMin* with *bWarning=TRUE* and *bAlarm=FALSE*, or the temperature is less than the *WarningMin* and higher than *AlarmMin* with *bWarning=TRUE* and *bAlarm=TRUE*, then the color of the temperature moving bar will change to color specified by *WarningLowColor*.

Property bWarning As Boolean

*bWarning* is used to determine whether to change the temperature color when the temperature exceeds the range (*WarningMin*, *WarningMax*). Meanwhile it will trigger the Event *TemperatureHighWarning* and *TemperatureLowWarning* if the warning status changes. Otherwise the control will ignore all warning properties.



Property ShowWarningMarker As Boolean

Specifies whether to show the warning markers or not (including the high warning marker and the low warning marker) which are used to mark the high warning temperature setpoint (*WarningMax*) and the low warning temperature setpoint (*WarningMin*) in the temperature meter bar.

Property ShowWarningValue As Boolean

Specifies whether to show the warning values or not (including the high warning setpoint (*WarningMax*) and the low warning setpoint (*WarningMin*)).

Property BkColor As OLE\_COLOR

*BkColor* specifies the color of the temperature meter bar.

Property CntlBkColor As OLE\_COLOR

*CntlBkColor* specifies the background color of the control if “*BackStyle* = *BS\_Opaque*”.

Property BackStyle As BackStyleType

If *BackStyle*=*BS\_Opaque*, then the background is rendered using *CntlBkColor*; If *BackStyle*=*BS\_Transparent*, then the background is rendered using the ambient background color of the container of the control.

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 text, legend text and the value.

Property ForeColor As OLE\_COLOR

*ForeColor* Specifies the color of the moving part (including the bottom bulk and temperature moving bar).

Property Gradient As Boolean

Specifies whether to draw the meter bar and the markers in gradient mode or not.

Property hWnd As Long

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

Property MaxVal As Double

*MaxVal* specifies the maximum value of the temperature scale.

Property MinVal As Double

*MinVal* specifies the minimum value of the temperature scale.

Property NoBulb As Boolean

Specifies whether to draw the bulb of the temperature bar.



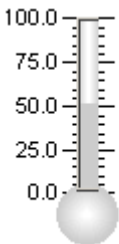
## Property ShowActualValue As Boolean

Specifies whether the actual temperature value is displayed or not.

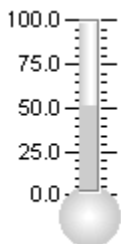
## Property ShowUnit As Boolean

Specifies whether to display the temperature unit or not.

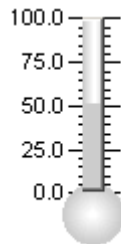
## Property StickStyle As Long



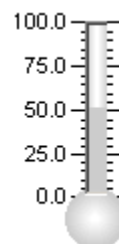
*StickStyle=0*



*StickSyle=1*



*StickSyle=2*



*StickStyle>=3*

*StickStyle* specifies four kinds of the temperature stick style.

## 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* is used to specifies the scale text color and the scale ticker color.

## Property Unit As String

Specifies the unit of the temperature.

## Property Value As Double

Set or retrieve the temperature value.

## Property Vertical As Boolean

Specifies whether the temperature meter bar is drawn vertically or horizontally.

## Method

### Sub UpdateValue(newVal As Double)

*UpdateValue* is used to update the value of the temperature to the new value specified by *newVal*.

## Events

### Event TemperatureHighAlarm(bActive As Boolean)

When *bAlarm=TRUE*, the temperature changes from the value which is lower than *AlarmMax* to the value which is higher than (or equal to) *AlarmMax*, then the event is triggered with *bActive=TRUE*; or the temperature changes from the value which is higher than (or equal to) *AlarmMax* to the value which is lower than *AlarmMax*, then the event is triggered with *bActive=FALSE*.

### Event TemperatureLowAlarm(bActive As Boolean)



When *bAlarm=TRUE*, the temperature changes from the value which is higher than *AlarmMin* to the value which is lower than (or equal to) *AlarmMin*, then the event is triggered with *bActive=TRUE*; or the temperature changes from the value which is lower than (or equal to) *AlarmMin* to the value which is higher than *AlarmMin*, then the event is triggered with *bActive=FALSE*.

### Event TemperatureHighWarning(*bActive As Boolean*)

When *bWarning=TRUE*, the temperature changes from the value which is lower than *WarningMax* to the value which is higher than (or equal to) *WarningMax*, then the event is triggered with *bActive=TRUE*; or the temperature changes from the value which is higher than (or equal to) *WarningMax* to the value which is lower than *WarningMax*, then the event is triggered with *bActive=FALSE*.

### Event TemperatureLowWarning(*bActive As Boolean*)

When *bWarning=TRUE*, the temperature changes from the value which is higher than *WarningMin* to the value which is lower than (or equal to) *WarningMin*, then the event is triggered with *bActive=TRUE*; or the temperature changes from the value which is lower than (or equal to) *WarningMin* to the value which is higher than *WarningMin*, then the event is triggered with *bActive=FALSE*.

### 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.