



Using Counter.Net Control

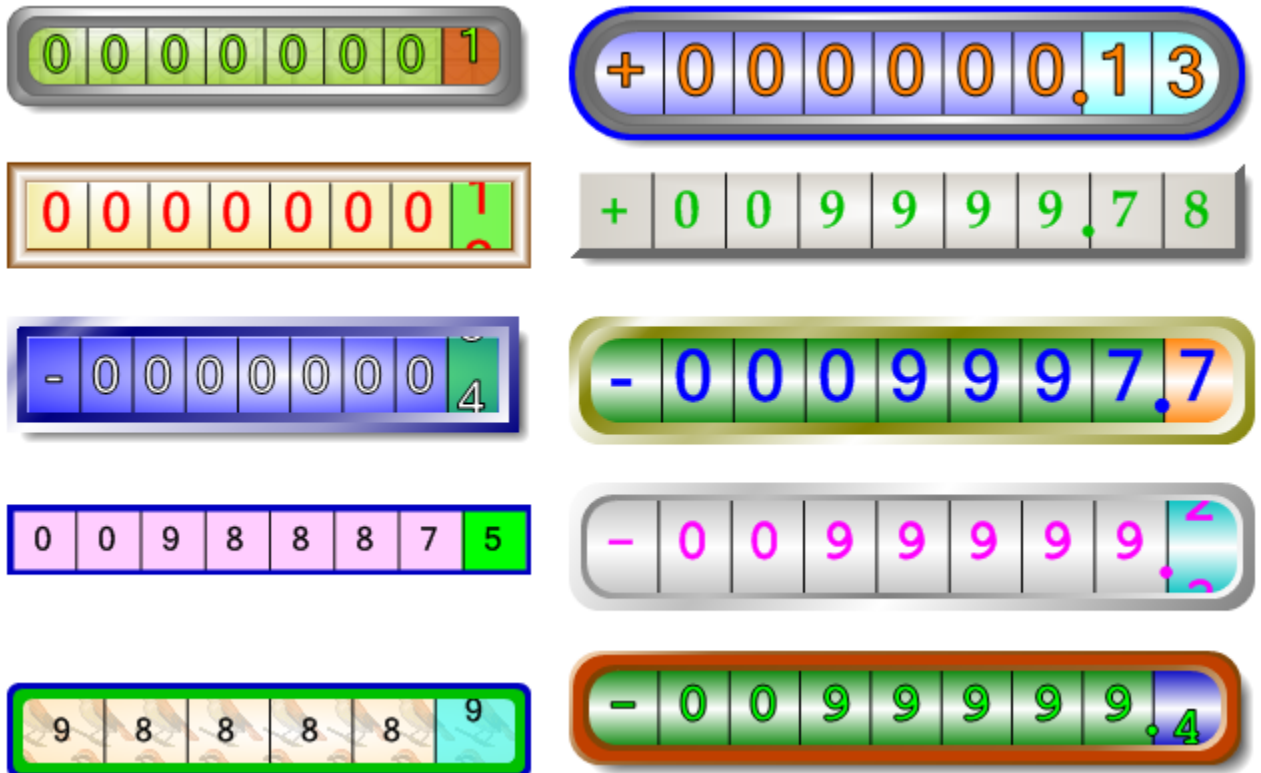
1. Overview

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 .Net custom control can be used in many .Net Windows form applications such as Factory Instrumentation Readouts, Automobile trip odometer, Aircraft digital display system and other numeric display cases.

This .Net 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 in the .Net Window Form Applications, the following picture shows some styles which can be configured by setting up these properties.

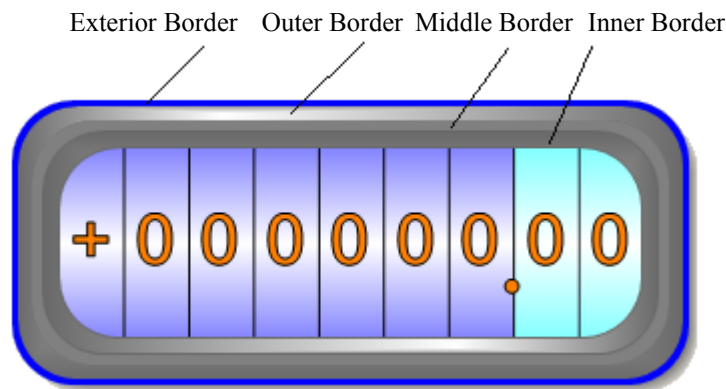




Properties:

Border Properties:

To make the control more intuitive and vivid look-and-feel, a group of border properties are provided for the designers to configure the control's border style. There are four border layers, i.e., Exterior Border layer, Outer Border layer, Middle Border layer and Inner Border layer,



We can configure different color and layer length for each layer. Especially for Outer-Border and Inner-Border, the light color and the dark color can be configured to generate different gradient 3D-Border feeling.

Property *DAS BorderStyle BorderShape*

Specify which border shape (Rectangle or Round Rectangle) is used to draw the border of control. There are two definitions, i.e., *BS_Rect* defines Rectangle Shape and *BS_RoundRect* defines Round Rectangle Shape (The size of the round corners is defined by the property *RoundRadius*).

Property *int BorderExteriorLength*

Specify the border length of the Exterior Border, this layer is drawn using the color defined by the property *BorderExteriorColor*. The default length of this layer is zero.

Property *Color BorderExteriorColor*

Specify the color to render the exterior border layer.

Property *int OuterBorderLength*

Specify the border length of the outer border, this layer is drawn using the colors defined by the property *OuterBorderDarkColor* and *OuterBorderLightColor*. This layer can be rendered in different gradient modes which is defined by the property *BorderGradientType*.

Property *Color OuterBorderDarkColor*



Specify the dark color of the outer border that is hidden from the light.

Property Color OuterBorderLightColor

Specify the light color of the outer border that faces the light.

Property int InnerBorderLength

Specify the border length of the inner border, this layer is drawn using the colors defined by the property *InnerBorderDarkColor* and *InnerBorderLightColor*.

This layer can be rendered in different gradient modes which is defined by the property *BorderGradientType*.

Property Color InnerBorderDarkColor

Specify the dark color of the inner border that is hidden from the light.

Property Color InnerBorderLightColor

Specify the light color of the inner border that faces the light.

Property int MiddleBorderLength

Specify the length of the middle border layer, this layer is rendered by property *MiddleBorderColor*.

Property Color MiddleBorderColor

Specify the color to render the middle border layer.

Property int RoundRadius

Specify the round corner size of the round-rectangle shape.

Border Gradient Properties:

There are six approaches defined in *DAS_BorderGradientStyle* to render the control border, i.e.,



BGS_None



BGS_Flat



BGS_Ring



BGS_Linear



BGS_Linear2



BGS_Path

BGS_None ---- Ignore the Outer Layer, Middle Layer and Inner Layer, only the Exterior layer is rendered.

BGS_Flat ---- All layers are rendered. The left side and top side of the Outer Layer are rendered using *OuterBorderLightColor*, the right side and bottom side of the Outer Layer are rendered using



OuterBorderDarkColor; The left side and top side of the Inner Layer are rendered using *InnerBorderDarkColor*, the right side and bottom side of the Inner Layer are rendered using *InnerBorderLightColor*;

- BGS_Ring ---- All layers are rendered. The outer layer is rendered in gradient mode at radial direction from *OuterBorderDarkColor* at the outer most periphery of the outer layer to *OuterBorderLightColor* at the inner most periphery of the outer layer. The inner layer is rendered in gradient mode at radial direction from *InnerBorderLightColor* at the outer most periphery of the inner layer to *InnerBorderDarkColor* at the inner most periphery of the inner layer.
- BGS_Linear --- All layers are rendered. The outer layer is rendered in gradient mode at the direction defined by property *BorderGradientAngle* from *OuterBorderLightColor* at the one end (defined by property *BorderGradientLightPos1* and *BorderGradientLightPos2*) to *OuterBorderDarkColor* at the other end of the outer layer. The inner layer is rendered in gradient mode at the direction defined by property *BorderGradientAngle* from *InnerBorderDarkColor* at the one end (defined by property *BorderGradientLightPos1* and *BorderGradientLightPos2*) of the inner layer to *InnerBorderLightColor* at the other end of the inner layer.
- BGS_Linear2 --- Similar to BGS_Linear. The difference is the definitions of property *BorderGradientLightPos1* and *BorderGradientLightPos2* of inner layer. The inner layer is rendered in gradient mode at the direction defined by property *BorderGradientAngle* from *InnerBorderDarkColor* at the one end of the inner layer to *InnerBorderLightColor* at the other end (defined by property *BorderGradientLightPos1* and *BorderGradientLightPos2*) of the inner layer.
- BGS_Path ---- All layers are rendered. The outer layer is rendered in gradient mode at the radial direction from *OuterBorderDarkColor* at the outer most periphery to *OuterBorderLightColor* at the center point that is in the outer layer at the direction defined by property *BorderGradientAngle*.. The inner layer is rendered in gradient mode at the radial direction from *InnerBorderDarkColor* at the outer most periphery to *InnerBorderLightColor* at the center point that is in the outer layer at the direction defined by $(BorderGradientAngle + 180^\circ)$.



Property *DAS BorderGradientStyle BorderGradientType*

Specify which border gradient type is applied to render the border of the control.

Property *int BorderGradientAngle*

Specify the gradient angle for *BGS_Linear*, *BGS_Linear2* and *BGS_Path*.

Property *float BorderGradientRate*

Specify the gradient focus size. It should be within 0.0 to 1.0.

Property *float BorderGradientLightPos1*

Specify the gradient light position 1 at the gradient angle. It should be within 0.0 to 1.0. This property just takes effect for *BGS_Linear* and *BGS_Linear2*.

Property *float BorderGradientLightPos2*

Specify the another gradient light position at the gradient angle. It should be within 0.0 to 1.0. If it is negative, the second gradient light position is ignored. This property just takes effect for *BGS_Linear* and *BGS_Linear2*.

Property *float BorderLightIntermediateBrightness*

Specify the color at the middle point between the light position 1 and the light position2. It should be within 0.0 to 1.0. If it's 0, the dark color is applied at middle point, if it's 1.0, the light color is used at the middle point. This property just takes effect for *BGS_Linear* and *BGS_Linear2*.

Background Properties:

In general, the background of the control can be rendered in gradient mode or normal mode. And the background image (defined by *BackgroundImage*) can be rendered as well using the standard image rendering methods (defined by *BackgroundImageLayout*), like "Tile", "Zoom", "Stretch", "Center" and "None", please refer to the standard background image rendering in MSDN. If *BackgroundImage* is set, the image is first rendered, and then the back color with transparency (defined by *BkTransparency*) is rendered. There are four background gradient types are defined as follows (*DAS_BkGradientStyle*),



BKGS_Linear



BKGS_Polygon



BKGS_Sphere



BKGS_Shine

The gradient direction for *BKGS_Linear* and *BKGS_Shine* is defined by the property *BkGradientAngle*.

Property *Color BackColor*



Specify the background color (digit cell background) of the control.

Property Image BackGroundImage

Specify the background image of the control. For details, refer to MSDN.

Property ImageLayout BackGroundImageLayout

Specify the background image layout style. For details, refer to MSDN.

Property float BkTransparency

Specify the back color rendering transparency rate. If it's 0.0, there should be no transparency, so if BackGroundImage is set, but BkTransparency is zero, the image is invisible, if it's 1.0, the image is fully visible, and the gradient back color rendering is not invisible.

Property float BkGradientRate

Specify the gradient focus size, it should be between 0.0 and 1.0.

Property bool BkGradient

Specify whether the background is rendered in gradient mode or normal mode.

Property int BkGradientAngle

Specify the background gradient direction angle. (Only for *BKGS_Linear* and *BKGS_Shine*)

Property Color BkGradientColor

Specify the background gradient color.

Property DAS BkGradientStyle BkGradientType

Specify the background gradient type.

Property float BkShinePosition

Specify the background gradient shine (focus) position (for *BKGS_Shine*), it should be between 0.0 and 1.0. In conjunction with *BkGradientAngle*, the gradient shine position can be fully adjusted in the background rendering area.

Shadow Properties:

Shadow display is a basic feature of the Dragonfly .Net controls.

Property bool ControlShadow

Specify whether to show the shadow or not.

Property int ShadowDepth

Specify the depth of the shadow.

Property Color ShadowColor

Specify the color of the shadow.



Property float ShadowRate

Specify the emission rate of the shadow.

General Properties:

Property Color ForeColor

Specify the color to render the digit numbers and the sign.

Property Font Font

Specify the standard font to render the digit numbers and the sign.

Property int DigitNumber

Specify how many digits are supported for the counter.

Property bool HighLight

Specify whether the last digit or the decimal part is highlighted or not using color *HighLightColor*.

Property Color HighLightColor

Specify the color to highlight the last digit or the decimal part when *HighLight* is true.

Property bool ShowSign

Specify whether to show the number sign (+/-) or not

Property int Precision

Specify the decimal digit number. Default is zero.

Property double Value

Specify the displayed value of the counter.

Property bool TextOutline

Specify whether the digits and sign symbol are outlined or not.

Property bool Rolling

Specify whether the last digit rolls or not when the valve has more precision than the precision the counter provides.

Functions & Methods:

DeltaUpdate(double fDelta)

The valve change delta can be added to the counter.

The designers can use the this method or *Value* property to update the counter display.