

## Introduction

Support - EMail [Info@tmssoftware.com](mailto:Info@tmssoftware.com)

Ver 1.0

### **Welcome and thank you for using TMS Instrumentation Workshop!**

TMS Instrumentation Workshop contains a large number of components, objects and routines for Borland Delphi with full source code. This library is designed for Borland Delphi 4/5/6 and CBuilder 4/5.

TMS Instrumentation Workshop is the ultimate "must have" package when you are looking for professional instrumentation and digital components. Our complete suite of 70+ components includes scopes, leds, meters, sliders, buttons, switches, banners, animation components and more.

Delphi and C++Builder are trademarks of Borland Corp.

**[What's New](#)**

**[VCL Overview](#)**

**[Ordering Information](#)**

**[Our License Agreement](#)**

**[How to contact us](#)**

## What's New

[see also](#)

Ver 1.0

What's new in version 1.0

- First release as TMS Instrumentation Workshop

## Help Install

Support - EMail [Info@tmssoftware.com](mailto:Info@tmssoftware.com)  
[see also](#)

### Installation

In order to integrate this help documentation into the Delphi IDE you need to follow the steps described below.

### Installing this help file

#### All supported versions

To add your help file use the OpenHelp utility located in \bin\oh.exe (or accessed using Help|Customize in the IDE). You will find information in the OpenHelp.hlp file about using OpenHelp, including adding your help file to the Help System.

#### No OpenHelp utility? (standard versions only)

First copy the helpfiles to the help folder. Then you can add the helpfiles manually by editing the following files in the help directory: DELPHIx.ohc, DELPHIx.ohi and DELPHIx.ohl where x stands for version number of your product. The entries you need to make are very straight forward. After changing the files remove all .gid files from the help folder so Delphi will recreate it's index.

Using BCB look for BCBx.ohc, BCBx.ohi and BCBx.ohl where x stands for version number of your product.

## IDE Install

Support - EMail [info@tmssoftware.com](mailto:info@tmssoftware.com)  
[see also](#)

### Installation

To be able to use the components they have to be added in the IDE.

### Adding the components

#### Delphi 4

1. Run Delphi 4 and select the FILE / OPEN menu item
2. Open the file ..\Delphi4\Vcwd4.dpk and click Install.
3. Choose Environment options from the Tools menu.
4. Add the folder of vcl to the library path.

#### Delphi 5

1. Run Delphi 5 and select the FILE / OPEN menu item
2. Open the file ..\Delphi5\Vcwd5.dpk and click Install.
3. Choose Environment options from the Tools menu.
4. Add the folder of vcl to the library path.

#### Delphi 6

1. Run Delphi 6 and select the FILE / OPEN menu item
2. Open the file ..\Delphi6\Vcwd6.dpk and click Install.
3. Choose Environment options from the Tools menu.
4. Add the folder of vcl to the library path.

#### BCB 4

1. Run BCB 4 and select the FILE / OPEN menu item
2. Open the file ..\Bcb4\Vcwb4.bpk and click Install.
3. Choose Environment options from the Tools menu.
4. Add the folder of vcl to the library path.

#### BCB 5

1. Run BCB 5 and select the FILE / OPEN menu item
2. Open the file ..\Bcb5\Vcwb5.bpk and click Install.
3. Choose Environment options from the Tools menu.
4. Add the folder of vcl to the library path.

The TMS Instrumentation Workshop pages will appear at the end of the current palette divided in 4 tabs

## TVrAnalogClock

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrAnalogClock is an analog clock in LCD style.

### Unit

vranalog

### Description

TVrAnalogClock is an analog clock in LCD style. TVrAnalogClock is derived from TVrGraphicControl and uses the system internal clock to indicate the current system time.

## **TVrAnalogClock.Active**

[TVrAnalogClock](#) [see also](#)

Determines if the clock is enabled or disabled.

**property** Active: Boolean;

### **Description**

Set Active to True in order to show the current system time. Although Active can be set in the designer, it is only activated during runtime sessions.

## **TVrAnalogClock.AlarmTime**

[TVrAnalogClock](#) [see also](#)

Contains the date and time on which to trigger the alarm.

**property** AlarmTime: TDateTime;

### **Description**

The property AlarmTime contains the date and time on which to trigger the alarm. Make sure EnableAlarm is set to True. When the alarm date and time is reached the OnAlarm event is called.  
Runtime only property.

## **TVrAnalogClock.EnableAlarm**

[TVrAnalogClock](#) [see also](#)

Used to switch to alarm mode.

**property** EnableAlarm: Boolean;

### **Description**

Set EnableAlarm to true in order to switch to alarm mode. The clock component will check the date and time contained in [AlarmTime](#) in order to signal an alarm.



## **TVrAnalogClock.Glyph**

[TVrAnalogClock](#) [see also](#)

Describes the background of the clock.

**property** Glyph: TBitmap;

### **Description**

Glyph is a bitmap image type which can be used as a background image. The painting method uses stretchpaint to fill the components canvas. If Glyph is not used the normal background color is used to fill the client area.

## **TVrAnalogClock.HandsColor**

[TVrAnalogClock](#) [see also](#)

Determines the color of the minute and hour hands of the clock.

**property** HandsColor: TColor;

### **Description**

The HandsColor property determines the color of the minute and hour hands of the clock.

## **TVrAnalogClock.HourMarks**

[TVrAnalogClock](#) [see also](#)

Determines the whether or not seconds will be displayed.

**property** HourMarks: Boolean;

### **Description**

The ShowSeconds property determines the whether or not seconds will be displayed.

## **TvrAnalogClock.Hours**

[TvrAnalogClock](#) [see also](#)

Returns the current system time.

**property** Hours: Word;

### **Description**

Hours returns the clock's actual time and is only accessible during runtime sessions (Readonly property). When the clock is active, hours contains the current system time otherwise it will contain the value from the point the clock was last activate.

## **TVrAnalogClock.Minutes**

[TVrAnalogClock](#) [see also](#)

Returns the current system time.

**property** Minutes: Word;

### **Description**

Minutes returns the clock's actual time and is only accessible during runtime sessions (Readonly property). When the clock is active, minutes contains the current system time otherwise it will have the value from the point the clock was last active.

## **TVrAnalogClock.OnAlarm**

[TVrAnalogClock](#) [see also](#)

Is called when EnableAlarm is set to true and AlarmTime is reached.

**property** OnAlarm: TNotifyEvent;

### **Description**

AlarmTime consists out of a date and time value. OnAlarm is called when EnableAlarm is set to true and AlarmTime is reached.

## **TVrAnalogClock.OnHoursChanged**

[TVrAnalogClock](#) [see also](#)

This event will occur when the hours property changed.

```
type TVrHoursChangeEvent = procedure(Sender: TObject; Hours: Word) of object;  
property OnHoursChanged: TVrHoursChangeEvent;
```

### **Description**

This event will occur when the hours property changed.

## **TVrAnalogClock.OnMinutesChanged**

[TVrAnalogClock](#) [see also](#)

This event will occur when the minutes property changed.

```
type TVrMinutesChangeEvent = procedure(Sender: TObject; Minutes: Word) of  
object;  
property OnMinutesChanged: TVrMinutesChangeEvent;
```

### **Description**

This event will occur when the minutes property changed.



## **TVrAnalogClock.OnSecondsChanged**

[TVrAnalogClock](#) [see also](#)

This event will occur when the seconds property changed.

```
type TVrSecondsChangeEvent = procedure(Sender: TObject; Seconds: Word) of  
object;  
property OnSecondsChanged: TVrSecondsChangeEvent;
```

### **Description**

This event will occur when the seconds property changed.

## **TVrAnalogClock.SecHandColor**

[TVrAnalogClock](#) [see also](#)

Determines the color of the second hand of the clock.

**property** SecHandColor: TColor;

### **Description**

The SecHandColor property determines the color of the second hand of the clock.

## **TVrAnalogClock.Seconds**

[TVrAnalogClock](#) [see also](#)

Returns the current system time.

**property** Seconds: Word;

### **Description**

Seconds returns the clock's actual time and is only accessible during runtime sessions (Readonly property). When the clock is active, seconds contains the current system time otherwise it will have the value from the point the clock was last active.

## **TVrAnalogClock.SecondsIndicator**

[TVrAnalogClock](#) [see also](#)

Hides the seconds indicator.

**property** SecondsIndicator: Boolean;

### **Description**

Set SecondsIndicator to False in order to hide the seconds indicator.

## **TVrAnalogClock.Threaded**

[TVrAnalogClock](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrAnalogClock.TickColor**

[TVrAnalogClock](#) [see also](#)

Defines the color of the hour markings.

**property** TickColor: TColor;

### **Description**

TickColor defines the color of the hour markings.

## **TVrAnalogClock.TickOutline**

[TVrAnalogClock](#) [see also](#)

Defines the outline color of the hour markings.

**property** TickOutline: TColor;

### **Description**

TickColor defines the outline color of the hour markings.

## **TVrAnalogClock.TickWidth**

[TVrAnalogClock](#) [see also](#)

Defines the size of the hour markings.

**property** TickWidth: Integer;

### **Description**

TickWidth defines the size of the hour markings.



## **TVrAnalogClock.Transparent**

[TVrAnalogClock](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## TVrAngularMeter

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrAngularMeter represents a rounded analog meter device.

### Unit

vringularmeter

### Description

TVrAngularMeter represents a rounded analog meter device derived from TVrGraphicControl.

TVrAngularMeter owns many properties for customization.

## **TvrAngularMeter.Angle**

[TvrAngularMeter](#) [see also](#)

Describes the starting angle of the scale and labels inside the controls boundaries.

**property** Angle: Integer;

### **Description**

Angle describes the starting angle of the scale and labels inside the controls boundaries.

## **TVrAngularMeter.AngleOffset**

[TVrAngularMeter](#) [see also](#)

Describes the size/length of the scale starting from angle.

**type** TVrDegrees = 0..360;

**property** AngleOffset: TVrDegrees;

### **Description**

AngleOffset describes the size/length of the scale starting from angle.

## **TVrAngularMeter.CenterDotColor**

[TVrAngularMeter](#) [see also](#)

Describes the fill color of the inner circle.

**property** CenterDotColor: TColor;

### **Description**

CenterDotColor describes the fill color of the inner circle.

## **TVrAngularMeter.CenterDotWidth**

[TVrAngularMeter](#) [see also](#)

Describes the size of the inner circle.

**property** CenterDotWidth: Integer;

### **Description**

CenterDotWidth describes the size of the inner circle.

## **TVrAngularMeter.ColorZone1**

[TVrAngularMeter](#) [see also](#)

Defines the color used to paint the lower area of the scale.

**property** ColorZone1: TColor;

### **Description**

The scale used by TVrAngularMeter is build out of three areas, High, Medium and Low. ZoneColor1 defines the color used to paint the lower area of the scale.

## **TVrAngularMeter.ColorZone2**

[TVrAngularMeter](#) [see also](#)

Defines the color used to paint the medium area of the scale.

**property** ColorZone2: TColor;

### **Description**

The scale used by TVrAngularMeter is build out of three areas, High, Medium and Low. ColorZone2 defines the color used to paint the medium area of the scale.



## **TVrAngularMeter.ColorZone3**

[TVrAngularMeter](#) [see also](#)

Defines the color used to paint the high area of the scale.

**property** ColorZone3: TColor;

### **Description**

The scale used by TVrAngularMeter is build out of three areas, High, Medium and Low. Color3 defines the color used to paint the high area of the scale.

## **TVrAngularMeter.Decimals**

[TVrAngularMeter](#) [see also](#)

Describes the number of decimals to display within the scale.

**type** TVrWordInt = 0..MaxInt;

**property** Decimals: TVrWordInt;

### **Description**

Describes the number of decimals to display within the scale.

## **TvrAngularMeter.Labels**

[TvrAngularMeter](#) [see also](#)

Defines the number of text labels displayed along the scale of the meter control.

**property** Labels: Integer;

### **Description**

Labels defines the number of text labels displayed along the scale of the meter control. All labels are automatically positioned.

## **TVrAngularMeter.LabelsFont**

[TVrAngularMeter](#) [see also](#)

Defines the font attributes for the text labels of the scale.

**property** LabelsFont: TFont;

### **Description**

LabelFont defines the font attributes for the text labels of the scale.

## **TVrAngularMeter.LabelsOffset**

[TVrAngularMeter](#) [see also](#)

Describes the space between the scale and the text labels.

**property** LabelOffset: Integer;

### **Description**

LabelOffset describes the space between the scale and the text labels.

## **TvrAngularMeter.MaxValue**

[TvrAngularMeter](#) [see also](#)

Use MaxValue to set a upper limit to the value that can be represented using the AngularMeter.

**property** MaxValue: Double;

### **Description**

Use MaxValue to set a upper limit to the value that can be represented using the AngularMeter.

## **TvrAngularMeter.MinValue**

[TvrAngularMeter](#) [see also](#)

Use MinValue to set a lower limit to the value that can be represented using the AngularMeter.

**property** MinValue: Double;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the AngularMeter.

## **TvrAngularMeter.NeedleBaseWidth**

[TvrAngularMeter](#) [see also](#)

Defines the size of the needle base.

**property** NeedleBaseWidth: Integer;

### **Description**

NeedleBaseWidth defines the size of the needle base.



## **TvrAngularMeter.NeedleColor**

[TvrAngularMeter](#) [see also](#)

Used to change the color of the needle.

**property** NeedleColor: TColor;

### **Description**

NeedleColor is used to change the color attribute of the needle. The needle points to the current position within the scale.

## **TVrAngularMeter.NeedleLength**

[TVrAngularMeter](#) [see also](#)

Defines the length or radius of the needle.

**property** NeedleLength: Integer;

### **Description**

NeedleLength defines the length or radius of the needle.

## **TVrAngularMeter.OnChange**

[TVrAngularMeter](#) [see also](#)

OnChange event occurs when you assign another value to the position property.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange event occurs when you assign another value to the position property.

## **TVrAngularMeter.Percent1**

[TVrAngularMeter](#) [see also](#)

Describes the size of the scale used for the lower segment color.

**property** Percent1: TVrPercentInt;

### **Description**

Percent1 and Percent2 make up the levels in TVrAngularGauge. Percent1 describes the percentage of the scale which is filled with the lower segment ColorZone1 color. Percent1 and Percent2 together can never exceed 100%.

## **TVrAngularMeter.Percent2**

[TVrAngularMeter](#) [see also](#)

Describes the size of the scale used for the medium segment color.

**property** Percent2: TVrPercentInt;

### **Description**

Percent1 and Percent2 make up the levels in TVrAngularMeter. Percent2 describes the percentage of the scale which is filled with the medium segment ColorZone2 colors. Percent1 and Percent2 together can never exceed 100%.

## **TvrAngularMeter.Position**

[TvrAngularMeter](#) [see also](#)

Points to the current state.

**property** Position: Double;

### **Description**

Position points to the current state. The position can never exceed the values defined by the MinValue and MaxValue properties.

## **TVrAngularMeter.Radius**

[TVrAngularMeter](#) [see also](#)

Used to define a custom size of the scale image starting from the left/top corner.

**property** Radius: Integer;

### **Description**

Radius is used to define a custom size of the scale image starting from the left/top corner. When setting Radius to -1 the radius is automatically calculated (and centered) by the control.

## **TVrAngularMeter.Spacing**

[TVrAngularMeter](#) [see also](#)

Describes the empty area around the meter image.

**property** Spacing: Integer;

### **Description**

Spacing describes the empty area around the meter image.



## **TVrAngularMeter.Ticks**

[TVrAngularMeter](#) [see also](#)

Defines the number of ticks of the scale.

```
type TVrWordInt = 0..MaxInt;
```

```
property Ticks: TVrWordInt;
```

### **Description**

The scale used by TVrAngularMeter is build out of three areas, High, Medium and Low. Ticks defines the number of ticks of the scale.

## **TVrAngularMeter.TicksColor**

[TVrAngularMeter](#) [see also](#)

Describes the color attribute of the scale ticks.

**property** TicksColor: TColor;

### **Description**

TicksColor describes the color attribute of the scale ticks.

## TVrAngularMeter.TicksEnlarge

[TVrAngularMeter](#) [see also](#)

Used to paint the scale of TVrAngularMeter

**property** TicksEnlarge: TVrWordInt;

### Description

TicksEnlarge is used to paint the scale of TVrAngularMeter. Each single dot within the scale can be upsized, defined by the TicksMax property. If TicksEnlarge is set to Zero all dots are painted with a TicksMin size, in pixels.

## **TVrAngularMeter.TicksMax**

[TVrAngularMeter](#) [see also](#)

Describes the size in pixels of each enlarged scale position.

**property** TicksMax: Integer;

### **Description**

TicksMax describes the size in pixels of each enlarged scale position.

## **TVrAngularMeter.TicksMin**

[TVrAngularMeter](#) [see also](#)

Describes the default size in pixels of each scale position.

**property** TicksMin: Integer;

### **Description**

TicksMin describes the default size in pixels of each scale position.

## **TVrAngularMeter.Transparent**

[TVrAngularMeter](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrAniButton**

[properties](#)[methods](#)

[events](#)

[see also](#)

Uses a filmstrip like TBitmap object for it's button shape and behaviour.

### **Unit**

VrButtons

### **Description**

TVrAniButton is derived from TVrGraphicControl and uses a filmstrip like TBitmap object for it's button shape and behaviour. Use TVrAniButton to put a Windows push button on a form. Users choose button controls to initiate actions.

## **TVrAniButton.AutoSize**

[TVrAniButton](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit the size of the graphical image.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit the size of the graphical image. The size of the image is calculated by dividing the width of the image with ImageCount.



## **TVrAniButton.BitmapIndex**

[TVrAniButton](#) [see also](#)

Indicates which graphical image to display contained in a TVrBitmapList component.

**property** BitmapIndex: Integer;

### **Description**

BitmapIndex determines which graphical image to display contained in a TVrBitmapList component. No image is displayed when the value defined by BitmapIndex is invalid.

## **TVrAniButton.BitmapList**

[TVrAniButton](#) [see also](#)

TVrBitmapList contains a external list of bitmaps.

**property** BitmapList: TVrBitmapList;

### **Description**

TVrBitmapList contains a external list of bitmaps. Therefore multiple components can use the same images which reduces the resource usage by the application.

## **TVrAniButton.Down**

[TVrAniButton](#) [see also](#)

Specifies whether the button is selected (down) or unselected (up).

**property** Down: Boolean;

### **Description**

Read Down to determine whether the button is selected. Can only be used when SwitchStyle is set to True.

## **TVrAniButton.ImageCount**

[TVrAniButton](#) [see also](#)

Indicates how many images the bitmap contains.

**property** ImageCount: Integer;

### **Description**

ImageCount Indicates how many images the bitmap contains.

## **TVrAniButton.Interval**

[TVrAniButton](#) [see also](#)

Determines the animation speed of the button image.

**property** Interval: Integer;

### **Description**

Use Interval to increase or decrease the animation speed of the button image.

## **TVrAniButton.SwitchStyle**

[TVrAniButton](#) [see also](#)

Use the button as a switch.

**property** SwitchStyle: Boolean;

### **Description**

Set SwitchStyle to True in order to use the button as a switch with a down state. If SwitchStyle is false the button will function as a normal push button.

## **TVrAniButton.Threaded**

[TVrAniButton](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrAniButton.Transparent**

[TVrAniButton](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.



## **TVrAnimate**

[properties](#)[methods](#)

[events](#)

[see also](#)

The TVrAnimate component displays simple animations on a form.

### **Unit**

vanimate

### **Description**

The TVrAnimate component displays simple animations on a form. The animation that appears is determined by the bitmap property. The bitmap consists of a series of two or more Frames arranged in filmstrip fashion. Filmstrip frames may be sequenced automatically at fixed time intervals or may be changed under program control. When sequenced at fixed time intervals, the Loop property determines if sequencing continues indefinitely.

## **TVrAnimate.Active**

[TVrAnimate](#) [see also](#)

Controls whether the component is active.

**property** Active: Boolean;

### **Description**

Set Active to True in order to play the loaded film strip.

## **TVrAnimate.AutoSize**

[TVrAnimate](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit the size of the graphical image.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit the size of the graphical image.

## **TVrAnimate.Bitmap**

[TVrAnimate](#) [see also](#)

Determines the animation that appears on the TVrAnimate control.

**property** Bitmap: TBitmap;

### **Description**

The bitmap property determines the animation that appears on the TVrAnimate control. The bitmap is an arranged series of two or more horizontal frames in a filmstrip fashion.

## **TVrAnimate.CurrentFrame**

[TVrAnimate](#) [see also](#)

Determines which filmstrip frame will be displayed.

**property** CurrentFrame: Integer;

### **Description**

The Frame property indicates or determines which filmstrip frame will be displayed.

CurrentFrame values are constrained to be within 0 and FrameCount-1, so incrementing or decrementing CurrentFrame past the limits will result in wrap-around.

## **TVrAnimate.FrameCols**

[TVrAnimate](#) [see also](#)

Support for very large image strips by using multiple rows and columns.

**property** FrameCols: TVrColInt;

### **Description**

The width and height of a bitmap are restricted by Windows. This way VrAnimate can support very large image strips by using multiple rows and columns in each strip.

## **TVrAnimate.FrameRows**

[TVrAnimate](#) [see also](#)

Support for very large image strips by using multiple rows and columns.

**property** FrameRows: TVrRowInt;

### **Description**

The width and height of a bitmap are restricted by Windows. This way VrAnimate can support very large image strips by using multiple rows and columns in each strip.

## **TVrAnimate.Interval**

[TVrAnimate](#) [see also](#)

Specifies the time (in milliseconds) between frame changes.

**property** Interval: Integer;

### **Description**

The Interval property specifies the time (in milliseconds) between frame changes when the filmstrip is played. Set the Active property to false if you plan to program frame changes using the CurrentFrame (Runtime only) property.



## **TVrAnimate.Loop**

[TVrAnimate](#) [see also](#)

Determines if the filmstrip will play continuously or terminate.

**property** Loop: Boolean;

### **Description**

The Loop property determines if the filmstrip will play continuously or terminate when the last frame is displayed.

## **TVrAnimate.OnNotify**

[TVrAnimate](#) [see also](#)

Called when the last frame within the film strip is displayed.

**property** OnNotify: TNotifyEvent;

### **Description**

OnNotify is called when the last frame within the film strip is displayed. OnNotify only applies when the Loop property is set to false.

## **TVrAnimate.Stretch**

[TVrAnimate](#) [see also](#)

Indicates whether the image should be changed so that it exactly fits the bounds of the image control.

**property** Stretch: Boolean;

### **Description**

Set Stretch to True to cause the image to assume the size and shape of the image control. When the image control resizes, the image resizes also. Stretch resizes the height and width of the image independently. Thus, unlike a simple change in magnification, stretch can distort the image if the image control is not the same shape as the image.

To resize the control to the image rather than resizing the image to the control, use the AutoSize property instead.

## **TVrAnimate.Threaded**

[TVrAnimate](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrAnimate.Transparent**

[TVrAnimate](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrArrow**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrArrow is an arrow shaped led control.

### **Unit**

vrrarrow

### **Description**

TVrArrow is an arrow shaped led control. Like any ordinary led it can be turned on or off. It can also be used as a button type control. Just assign an event to the OnClick event. When clicked the control only generates the OnClick event it doesn't visually changes shape or look. TVrArrow is derived from TVrGraphicControl.

## **TVrArrow.Active**

[TVrArrow](#) [see also](#)

Controls whether the component is active.

**property** Active: Boolean;

### **Description**

Set Active to True in order to activate the led control.

## **TVrArrow.Direction**

[TVrArrow](#) *see also*

Describes the direction of the arrow.

```
type TVrArrowDirection = (pdLeft, pdRight, pdUp, pdDown);  
property Direction: TVrArrowDirection;
```



## **TVrArrow.Palette**

[TVrArrow](#) *see also*

Defines the color attributes for TVrArrow.

**property** Palette: TVrPalette;

### **Description**

Use Palette to define the color attributes for TVrArrow.

## **TVrArrow.TrackMouse**

[TVrArrow](#) *see also*

Determines if mouse movement is detected.

**property** TrackMouse: Boolean;

### **Description**

When TrackMouse is set to True the led is automatically activated when the mouse is moved within the controls boundaries.

## **TVrArrow.Transparent**

[TVrArrow](#) *see also*

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrBanner**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrBanner is a component that scrolls a bitmap from one side to another.

### **Unit**

vrbanner

### **Description**

TVrBanner is a component that scrolls a bitmap from one side to another. You can define scrolling speed and direction. When a bitmap is loaded it's converted into a LCD image which is being displayed.

## **TVrBanner.AutoScroll**

[TVrBanner](#) [see also](#)

Enables or disables scrolling.

**property** AutoScroll: Boolean;

### **Description**

Set AutoScroll to True in order to start the scrolling of the bitmap image. AutoScroll is only activated during runtime sessions.

## **TVrBanner.Bevel**

[TVrBanner](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrBanner.Bitmap**

[TVrBanner](#) [see also](#)

Image which is used to create the LCD display.

**property** Bitmap: TBitmap;

### **Description**

Use the bitmap property to define the image which is converted to a LCD graphical image. The converted bitmap will then be displayed within it's raster.

## **TVrBanner.Direction**

[TVrBanner](#) [see also](#)

Describes the bitmap scroll direction.

```
type TVrBannerDirection = (bdRightToLeft, bdLeftToRight, bdTopToBottom,  
bdBottomToTop);
```

```
property Direction: TVrBannerDirection;
```

### **Description**

Scrolling can occur from Right-to-Left, Left-to-Right, Top-to-Bottom and Bottom-to-Top.



## **TVrBanner.OnScrollDone**

[TVrBanner](#) [see also](#)

Event which is triggered after each scroll loop.

**property** OnScrollDone: TNotifyEvent;

### **Description**

When the bitmap is scrolled outside the view of the display it will call the OnScrollDone event. This also makes it possible to display several bitmaps without pause or breaks between each scroll.

## **TVrBanner.PixelColor**

[TVrBanner](#) [see also](#)

Defines the color of each LCD segment.

**property** PixelColor: TColor;

### **Description**

Use PixelColor to define the color of each LCD segment or pixel group. PixelColor is used when [PixelMode](#) is set to pmCustom.

## **TVrBanner.PixelMode**

[TVrBanner](#) [see also](#)

Defines the way VrBanner retrieves the pixelcolor to use.

```
type TVrBannerPixelMode = (pmAuto, pmCustom);
```

```
property PixelMode: TVrBannerPixelMode;
```

### **Description**

When PixelMode is set to pmCustom, VrBanner will use the color defined by [PixelColor](#) to draw it's raster. When PixelMode is set to pmAuto, VrBanner will try to retrieve the background color of the loaded bitmap. It will then look at the [0,0] coordinates of the bitmap, at the upper left corner, and will use that color as the active raster/lcd color. To use this feature make sure that the first pixel at [0,0] of the bitmap always contains the active LCD color.

## **TVrBanner.PixelSize**

[TVrBanner](#) [see also](#)

The size in pixels of each LCD segment.

**property** PixelSize: Integer;

### **Description**

Each segment is painted as a small square (1x1, 2x2). PixelSize defines the size in pixels of each lcd segment.

## **TVrBanner.Spacing**

[TVrBanner](#)

[see also](#)

Spacing determines the number of pixels between each LCD pixel.

**property** Spacing: Integer;

### **Description**

Spacing determines the number of pixels between each LCD pixel. The visible area defined by spacing is filled with the normal background color of the control.

## **TVrBanner.Threaded**

[TVrBanner](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrBanner.TimeInterval**

[TVrBanner](#) [see also](#)

Determines the scrolling speed of the image.

**property** TimeInterval: Integer;

### **Description**

Use TimeInterval to increase or decrease the scrolling speed of the bitmap image. Only applies when the AutoScroll property is set to true.

## **TVrBevel**

[properties](#)[methods](#)

[events](#)

[see also](#)

Defines the attributes of the border which is painted around the client area.

### **Unit**

vrclasses

### **Description**

Use TVrBevel to create beveled boxes or frames. The bevel can appear raised or lowered. The bevel object is derived from TPersistent and contains several separate properties. These properties make the bevel shape on the components canvas.



## **TvrBevel.BorderColor**

[TvrBevel](#)

[see also](#)

**property** BorderColor: TColor;

## **TvrBevel.BorderWidth**

[TvrBevel](#)

[see also](#)

**property** Borderwidth: Integer;

## **TvrBevel.InnerColor**

[TvrBevel](#)

[see also](#)

**property** InnerColor: TColor;

## **TvrBevel.InnerHighlight**

[TvrBevel](#)

[see also](#)

**property** InnerHighlight: TColor;

## **TVrBevel.InnerOutline**

[TVrBevel](#)

[see also](#)

```
type TVrBevelOutlineStyle = (osOuter, osInner, osNone);  
property InnerOutline: TVrBevelOutlineStyle;
```

## **TvrBevel.InnerShadow**

[TvrBevel](#)

[see also](#)

**property** InnerShadow: TColor;

## **TVrBevel.InnerSpace**

[TVrBevel](#)

[see also](#)

```
type TVrBevelSpace = 0..MaxInt;  
property InnerSpace: TVrBevelSpace;
```

## **TVrBevel.InnerStyle**

[TVrBevel](#)

[see also](#)

```
type TVrBevelStyle = (bsNone, bsLowered, bsRaised);  
property InnerStyle: TVrBevelStyle;
```



## **TVrBevel.InnerWidth**

[TVrBevel](#)

[see also](#)

```
type TVrBevelWidth = 1..MaxInt;  
property InnerWidth: TVrBevelWidth;
```

## **TvrBevel.OuterColor**

[TvrBevel](#)

[see also](#)

**property** OuterColor: TColor;

## **TvrBevel.OuterHighlight**

[TvrBevel](#)

[see also](#)

**property** OuterHighlight: TColor;

## **TvrBevel.OuterOutline**

[TvrBevel](#)

[see also](#)

```
type TvrBevelOutlineStyle = (osOuter, osInner, osNone);  
property OuterOutline: TvrBevelOutlineStyle;
```

## **TvrBevel.OuterShadow**

[TvrBevel](#)

[see also](#)

**property** OuterShadow: TColor;

## **TVrBevel.OuterSpace**

[TVrBevel](#)

[see also](#)

```
type TVrBevelSpace = 0..MaxInt;  
property OuterSpace: TVrBevelSpace;
```

## **TVrBevel.OuterStyle**

[TVrBevel](#)

[see also](#)

```
type TVrBevelStyle = (bsNone, bsLowered, bsRaised);  
property OuterStyle: TVrBevelStyle;
```

## **TVrBevel.OuterWidth**

[TVrBevel](#)

[see also](#)

```
type TVrBevelWidth = 1..MaxInt;  
property OuterWidth: TVrBevelWidth;
```



## **TvrBevel.Visible**

[TvrBevel](#)

[see also](#)

**property** Visible: Boolean;

### **Description**

Setting visible to false means no bevel is used.

## **TVrBitmapButton**

[properties](#)[methods](#)   [events](#)   [see also](#)

Uses a TBitmap object for it's button shape.

### **Unit**

vrdesign

### **Description**

TVrBitmapButton is derived from TGraphicControl and uses a TBitmap object for it's button shape. Use TVrBitmapButton to put a standard Windows push button on a form. TVrBitmapButton introduces several properties to control its behavior. Users choose button controls to initiate actions.

## **TVrBitmapButton.AutoSize**

[TVrBitmapButton](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit the size of the graphical image.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit the size of the graphical image.

## **TVrBitmapButton.Glyph**

[TVrBitmapButton](#) [see also](#)

Contains the graphical button image.

**property** Glyph: TBitmap;

### **Description**

Glyph defines the actual button image. It can contain up to 4 separate images, each indicating a different state. See also [NumGlyphs](#).

## **TVrBitmapButton.HIndent**

[TVrBitmapButton](#) [see also](#)

Describes the horizontal offset when the button is pressed.

**property** HIndent: Integer;

### **Description**

HIndent describes the horizontal offset when the button is pressed. When AutoSize is true the width of the control will be increased with the value defined by HIndent.

## TVrBitmapButton.NumGlyphs

[TVrBitmapButton](#) [see also](#)

Describes the number of button images.

**type** TVrNumGlyphs = 1..4;

**property** NumGlyphs: Integer;

### Description

NumGlyphs defines the number of button images contained in the Glyph property. Each button image represents a certain state. At least 1 button image must be defined.

Value	Meaning
1	Normal
2	Mouse is moved over the button
3	Button is pressed
4	Button is disabled

## **TVrBitmapButton.Transparent**

[TVrBitmapButton](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrBitmapButton.TransparentMode**

[TVrBitmapButton](#) [see also](#)

Defines the way VrBitmapButton retrieves the pixelcolor used to make the graphical image transparent.

```
type TVrTransparentMode = (tmPixel, tmColor);
```

```
property TransparentMode: TVrTransparentMode;
```

### **Description**

When TransparentMode is set to tmColor, VrBitmapButton will use the color defined by the Color property to make the graphical image transparent. When TransparentMode is set to tmPixel, VrBitmapButton will try to retrieve the background color of the loaded bitmap. It will then look at the [0,0] coordinates of the bitmap, at the upper left corner, and will use that color to make the graphical image transparent.



## **TVrBitmapButton.VIndent**

[TVrBitmapButton](#) [see also](#)

Describes the vertical offset when the button is pressed.

**property** VIndent: Integer;

### **Description**

VIndent describes the vertical offset when the button is pressed. When AutoSize is true the height of the control will be increased with the value defined by VIndent.

## TVrBitmapCheckBox

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrBitmapCheckBox represents a Windows check box.

### Unit

vrdesign

### Description

A TVrBitmapCheckBox component presents an option for the user. The user can check the box to select the option, or uncheck it to deselect the option. The images of the checked, unchecked and grayed states are user defined. TVrBitmapCheckBox is derived from TVrCustomControl.

## **TVrBitmapCheckBox.AllowGrayed**

[TVrBitmapCheckBox](#)

[see also](#)

Determines whether check box can be in a "grayed" state.

**property** AllowGrayed: Boolean;

### **Description**

If AllowGrayed is set to True, the check box has three possible states: checked, unchecked, and grayed.

If AllowGrayed is set to False, the check box has only two possible states: checked and unchecked.

## **TVrBitmapCheckBox.BitmapList**

[TVrBitmapCheckBox](#)

[see also](#)

TVrBitmapList contains a external list of bitmaps.

**property** BitmapList: TVrBitmapList;

### **Description**

TVrBitmapList contains a external list of bitmaps. Therefore multiple components can use the same images which reduces the resource usage by the application.

## **TVrBitmapCheckBox.Checked**

[TVrBitmapCheckBox](#)

[see also](#)

Specifies whether the check box appears checked.

**property** Checked: Boolean;

### **Description**

Use Checked to determine whether the check box is checked or unchecked. Checked is True when the check box has a checked state, False if it is unchecked or grayed.

## **TVrBitmapCheckBox.DisabledGlyphIndex**

[TVrBitmapCheckBox](#)

[see also](#)

Points to the glyph index (Disabled images) contained in the BitmapList.

**property** DisabledGlyphIndex: Integer;

### **Description**

DisabledGlyphIndex is used in combination with the BitmapList property. DisabledGlyphIndex points to the glyph index (Disabled images) contained in the BitmapList. Each glyph can contain multiple images. See also [NumGlyphs](#).

## **TVrBitmapCheckBox.EnabledGlyphIndex**

[TVrBitmapCheckBox](#)

[see also](#)

Points to the glyph index contained in the BitmapList.

**property** EnabledGlyphIndex: Integer;

### **Description**

EnabledGlyphIndex is used in combination with the BitmapList property. EnabledGlyphIndex points to the glyph index (enabled images) contained in the BitmapList. Each glyph can contain multiple images. See also [NumGlyphs](#).

## **TVrBitmapCheckBox.FocusColor**

[TVrBitmapCheckBox](#)

[see also](#)

Color for the outline when the checkbox becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the checkbox receives focus, a small rectangle is painted around the button with the color defined by FocusColor.



## **TVrBitmapCheckBox.FocusOffset**

[TVrBitmapCheckBox](#)

[see also](#)

Describes the overall offset from the controls boundaries

**property** FocusOffset: Integer;

### **Description**

FocusOffset describes the overall offset from the controls boundaries from which the focus rectangle is painted. Setting FocusOffset less than zero means no focus rectangle is painted.

## **TVrBitmapCheckBox.Font3D**

[TVrBitmapCheckBox](#) [see also](#)

Defines the attributes of the text.

```
type TVrFont3D = class(TVrPersistent)  
property Font3D: TVrFont3D;
```

## TvrBitmapCheckBox.Layout

[TvrBitmapCheckBox](#)

[see also](#)

Determines where the image or text appears on the button.

**type** TvrImageTextLayout = (ImageLeft, ImageRight, ImageTop, ImageBottom);

**property** Layout: TvrImageTextLayout;

### Description

Value	Meaning
ImageLeft	The image or caption appears near the left side of the button.
ImageRight	The image or caption appears near the right side of the button.
ImageTop	The image or caption appears near the top of the button.
ImageBottom	The image or caption appears near the bottom of the button.

## **TVrBitmapCheckBox.Margin**

[TVrBitmapCheckBox](#)

[see also](#)

Use Margin to specify the indentation of the image or the text specified by the Caption property.

**property** Margin: Integer;

### **Description**

Use Margin to specify the indentation of the image or the text specified by the Caption property. The edges that Margin separates depends on the Layout property. If Layout is ImageLeft, the margin appears between the left edge of the image or caption and the left edge of the control. If Layout is ImageRight, the margin separates the right edges. If Layout is ImageTop, the margin separates the top edges, and if Layout is ImageBottom, the margin separates the bottom edges.

If Margin is -1 then the image or text are centered on the button.

## TVrBitmapCheckBox.NumGlyphs

[TVrBitmapCheckBox](#) [see also](#)

Describes the number of button images.

**type** TVrCheckBoxGlyphs = 1..3;

**property** NumGlyphs: TVrCheckBoxGlyphs;

### Description

NumGlyphs defines the number of button images contained in the Enabled and Disabled glyph images. Each button image represents a certain state. Atleast 1 button image must be defined.

Value	Meaning
-------	---------

---

1	Unchecked
2	Checked
3	Grayed

## **TVrBitmapCheckBox.OnChange**

[TVrBitmapCheckBox](#)

[see also](#)

OnChange is called as soon as the control changes state.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called as soon as the control changes state (Checked property).

## **TVrBitmapCheckBox.Spacing**

[TVrBitmapCheckBox](#)

[see also](#)

Defines the space between the Image and Caption text.

**property** Spacing: Integer;

### **Description**

Defines the space between the Image and Caption text.

## **TVrBitmapCheckBox.State**

[TVrBitmapCheckBox](#) [see also](#)

Indicates the current state of the checkbox.

**type** TVrCheckBoxState = (vcbUnchecked, vcbChecked, vcbGrayed);

**property** State: TVrCheckBoxState;

### **Description**

State indicates the current state of the checkbox.



## **TVrBitmapCheckBox.TextureIndex**

[TVrBitmapCheckBox](#)

[see also](#)

Describes the image used to fill the background of the control.

**property** TextureIndex: Integer;

### **Description**

TextureIndex is used in combination of the BitmapList property. TextureIndex describes the image used to fill the background of the control.

## **TVrBitmapCheckBox.TextureStyle**

[TVrBitmapCheckBox](#)

[see also](#)

Describes the way the background is filled.

**type** TVrCheckBoxTexture = (cbtTile, cbtStretch);

**property** TextureStyle: TVrCheckBoxTexture;

### **Description**

TextureStyle describes the way the background is filled.

## **TVrBitmapCheckBox.Toggle**

[TVrBitmapCheckBox](#)

[see also](#)

Use Toggle to change the state of the checkbox.

**procedure** Toggle;

### **Description**

Use Toggle to change the state of the checkbox. It will cycle through the unchecked, checked and grayed states.

## **TVrBitmapCheckBox.TransparentColor**

[TVrBitmapCheckBox](#)

[see also](#)

Is used to make the Glyphs, contained in the BitmapList, transparent.

**property** TransparentColor: TColor;

### **Description**

TransparentColor is used to make the Glyphs, contained in the BitmapList, transparent. The color is masked out from the background which creates a transparent effect.

## TVrBitmapDial

[properties](#)[methods](#)

[events](#)

[see also](#)

A slider like control.

### Unit

vrbitmapdial

### Description

TVrBitmapDial is a slider like control. Instead of using a single predefined image and scale it uses a film-strip like TBitmap object for it's button shape and behaviour. TVrBitmapDial is derived from TVrGraphicControl.

## **TVrBitmapDial.AutoSize**

[TVrBitmapDial](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit the size of the graphical image.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit the size of the graphical image. The size of the image is calculated by dividing the width of the image with ImageCount.

## **TVrBitmapDial.BitmapIndex**

[TVrBitmapDial](#) [see also](#)

Indicates which graphical image to display contained in a TVrBitmapList component.

**property** BitmapIndex: Integer;

### **Description**

BitmapIndex determines which graphical image to display contained in a TVrBitmapList component. No image is displayed when the value defined by BitmapIndex is invalid.

## **TVrBitmapDial.BitmapList**

[TVrBitmapDial](#) [see also](#)

TVrBitmapList contains a external list of bitmaps.

**property** BitmapList: TVrBitmapList;

### **Description**

TVrBitmapList contains a external list of bitmaps. Therefore multiple components can use the same images which reduces the resource usage by the application.



## **TVrBitmapDial.HideCursor**

[TVrBitmapDial](#) [see also](#)

Hides the cursor while operation the dial.

**property** HideCursor: Boolean;

### **Description**

Use HideCursor in order to hide the cursor while operation the dial.

## **TVrBitmapDial.ImageCount**

[TVrBitmapDial](#) [see also](#)

Indicates how many images the bitmap contains.

**property** ImageCount: Integer;

### **Description**

ImageCount Indicates how many images the bitmap contains.

## **TVrBitmapDial.Increment**

[TVrBitmapDial](#) [see also](#)

Increase or decrease the position with the value defined by Increment.

**property** Increment: Integer;

### **Description**

Increment is used when Style is "dsButtonClick" . The position is increased or decreased with value defined by Increment as soon as a mousebutton is pressed.

## **TvrBitmapDial.MaxValue**

[TvrBitmapDial](#) [see also](#)

Use MaxValue to set a upper limit to the value that can be represented using the dial.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set a upper limit to the value that can be represented using the dial.

## **TvrBitmapDial.MinValue**

[TvrBitmapDial](#) [see also](#)

Use MinValue to set a lower limit to the value that can be represented using the dial.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the dial.

## **TVrBitmapDial.OnChange**

[TVrBitmapDial](#) [see also](#)

OnChange event occurs when you assign another value to the position property.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange event occurs when you assign another value to the position property.

## **TvrBitmapDial.Position**

[TvrBitmapDial](#) [see also](#)

Points to the current state.

**property** Position: Integer;

### **Description**

Position points to the current state. The position can never exceed the values defined by the MinValue and MaxValue properties.

## **TVrBitmapDial.Style**

[TVrBitmapDial](#) [see also](#)

```
type TDialStyle = (dsLeftRightMove, dsTopBottomMove, dsButtonClick);  
property Style: TDialStyle;
```

### **Description**

#### **dsLeftRightMove**

Press the left mouse button and slide from left to right.

#### **dsTopBottomMove**

Press the left mouse button and slide from top to bottom.

#### **dsButtonClick**

Press the left mouse button and the right mouse button the change the position.



## **TVrBitmapDial.Transparent**

[TVrBitmapDial](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrBitmapImage**

[properties](#)[methods](#) [events](#) [see also](#)

Used to display a graphical image on a form.

### **Unit**

vrdesign

### **Description**

Use TBitmapImage to display a graphical image on a form. TBitmapImage introduces several properties to determine how the image is displayed within the boundaries of the TBitmapImage object.

TBitmapImage needs a TVrBitmapList component which is used as a buffer for the graphical image to display.

## **TVrBitmapImage.AutoSize**

[TVrBitmapImage](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit the size of the graphical image.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit the size of the graphical image.

## **TVrBitmapImage.BitmapIndex**

[TVrBitmapImage](#) [see also](#)

Indicates which graphical image to display contained in a TVrBitmapList component.

**property** BitmapIndex: Integer;

### **Description**

BitmapIndex determines which graphical image to display contained in a TVrBitmapList component. No image is displayed when the value defined by BitmapIndex is invalid.

## **TVrBitmapImage.BitmapList**

[TVrBitmapImage](#) [see also](#)

Contains the graphical images

**property** BitmapList: TVrBitmapList;

### **Description**

A BitmapList is separate buffer component which contains a list of unformatted TBitmap objects. These graphical images are used by TVrBitmapImage in combination with the BitmapIndex property.

## **TVrBitmapImage.Center**

[TVrBitmapImage](#) [see also](#)

Indicates whether the image is centered in the image control.

**property** Center: Boolean;

### **Description**

When the image does not fit perfectly within the image control, use Center to specify how the image is positioned. When Center is True, the image is centered in the control. When Center is False, the upper left corner of the image is positioned at the upper left corner of the control.

Note: Center has no effect if the AutoSize property is True or if the Stretch property is True.

## **TVrBitmapImage.Stretch**

[TVrBitmapImage](#) [see also](#)

Indicates whether the image should be changed so that it exactly fits the bounds of the image control.

**property** Stretch: Boolean;

### **Description**

Set Stretch to True to cause the image to assume the size and shape of the image control. When the image control resizes, the image resizes also. Stretch resizes the height and width of the image independently. Thus, unlike a simple change in magnification, stretch can distort the image if the image control is not the same shape as the image.

To resize the control to the image rather than resizing the image to the control, use the AutoSize property instead.

## **TVrBitmapImage.Transparent**

[TVrBitmapImage](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.



## **TVrBitmapList**

[properties](#)[methods](#)

[events](#)

[see also](#)

Non visual TVrBitmaps container component.

### **Unit**

vrsystem

### **Description**

TVrBitmapList is a wrapper component which contains a list of bitmaps. In compare to a normal imagelist, bitmaps are stored without any formatting. No resizing will take place when adding a bitmap to the list.

## **TVrBitmapList.Bitmaps**

[TVrBitmapList](#) [see also](#)

Contains the individual bitmaps in the list component.

**property** Bitmaps: TVrBitmaps;

### **Description**

Bitmaps is a TVrBitmaps object.

## **TVrBitmapList.GetBitmap**

[TVrBitmapList](#) [see also](#)

Retrieves a bitmap from the list of bitmaps.

```
function GetBitmap(Index: Integer): TBitmap;
```

### **Description**

GetBitmap retrieves a bitmap from the list of bitmaps. If Index is invalid then a nil pointer is returned.

## **TVrBitmapList.OnChange**

[TVrBitmapList](#) [see also](#)

Occurs immediately after the list of bitmaps changes.

**property** OnChange: TNotifyEvent;

### **Description**

Write an OnChange event handler to respond to changes in the list of bitmaps.

Whenever bitmaps in the list are added, deleted, moved, or modified, the following events take place:

- 1 The bitmaps are added, deleted, moved, or modified.
- 2 Finally, an OnChange event occurs.

## **TVrBitmapRadioButton**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrBitmapRadioButton is a wrapper for a Windows type radio button.

### **Unit**

vrdesign

### **Description**

Use TVrBitmapRadioButton to add a radio button to a form. Radio buttons present a set of mutually exclusive options to the user—that is, only one radio button in a set can be selected at a time. When the user selects a radio button, the previously selected radio button becomes unselected. Radio buttons are frequently grouped in a radio group box (TRadioGroup). Add the group box to the form first, then get the radio buttons from the Component palette and put them into the group box.

By default, all radio buttons that are directly contained in the same windowed control container, such as a TPanel, are grouped. For example, two radio buttons on a form can be checked at the same time only if they are contained in separate containers, such as two different group boxes.

## **TVrBitmapRadioButton.BitmapList**

[TVrBitmapRadioButton](#) [see also](#)

TVrBitmapList contains a external list of bitmaps.

**property** BitmapList: TVrBitmapList;

### **Description**

TVrBitmapList contains a external list of bitmaps. Therefore multiple components can use the same images which reduces the resource usage by the application.

## **TVrBitmapRadioButton.Checked**

[TVrBitmapRadioButton](#) [see also](#)

Determines whether the option represented by the radio button is selected.

**property** Checked: Boolean;

### **Description**

Read Checked to determine whether the radio button is selected. Set Checked to True to select the radio button and deselect all other radio buttons in the same container. Set Checked to False to deselect the radio button, leaving no radio button in the group selected.

## **TVrBitmapRadioButton.DisabledGlyphIndex**

[TVrBitmapRadioButton](#) [see also](#)

Points to the glyph index (Disabled images) contained in the BitmapList.

**property** DisabledGlyphIndex: Integer;

### **Description**

DisabledGlyphIndex is used in combination with the BitmapList property. DisabledGlyphIndex points to the glyph index (Disabled images) contained in the BitmapList. Each glyph can contain multiple images. See also [NumGlyphs](#).



## **TVrBitmapRadioButton.EnabledGlyphIndex**

[TVrBitmapRadioButton](#) [see also](#)

Points to the glyph index contained in the BitmapList.

**property** EnabledGlyphIndex: Integer;

### **Description**

EnabledGlyphIndex is used in combination with the BitmapList property. EnabledGlyphIndex points to the glyph index (enabled images) contained in the BitmapList. Each glyph can contain multiple images. See also [NumGlyphs](#).

## **TVrBitmapRadioButton.FocusColor**

[TVrBitmapRadioButton](#) [see also](#)

Color for the outline when the RadioButton becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the RadioButton receives focus, a small rectangle is painted around the button with the color defined by FocusColor.

## **TVrBitmapRadioButton.FocusOffset**

[TVrBitmapRadioButton](#) [see also](#)

Describes the overall offset from the controls boundaries

**property** FocusOffset: Integer;

### **Description**

FocusOffset describes the overall offset from the controls boundaries from which the focus rectangle is painted. Setting FocusOffset less than zero means no focus rectangle is painted.

## **TVrBitmapRadioButton.Font3D**

[TVrBitmapRadioButton](#) [see also](#)

Defines the attributes of the text.

**type** TVrFont3D = class(TVrPersistent)

**property** Font3D: TVrFont3D;

## TvrBitmapRadioButton.Layout

[TvrBitmapRadioButton](#) [see also](#)

Determines where the image or text appears on the button.

**type** TvrImageTextLayout = (ImageLeft, ImageRight, ImageTop, ImageBottom);

**property** Layout: TvrImageTextLayout;

### Description

Value	Meaning
ImageLeft	The image or caption appears near the left side of the button.
ImageRight	The image or caption appears near the right side of the button.
ImageTop	The image or caption appears near the top of the button.
ImageBottom	The image or caption appears near the bottom of the button.

## **TvrBitmapRadioButton.Margin**

[TvrBitmapRadioButton](#) [see also](#)

Use Margin to specify the indentation of the image or the text specified by the Caption property.

**property** Margin: Integer;

### **Description**

Use Margin to specify the indentation of the image or the text specified by the Caption property. The edges that Margin separates depends on the Layout property. If Layout is ImageLeft, the margin appears between the left edge of the image or caption and the left edge of the control. If Layout is ImageRight, the margin separates the right edges. If Layout is ImageTop, the margin separates the top edges, and if Layout is ImageBottom, the margin separates the bottom edges.

If Margin is -1 then the image or text are centered on the button.

## TVrBitmapRadioButton.NumGlyphs

[TVrBitmapRadioButton](#) [see also](#)

Describes the number of button images.

**type** TVrRadioButtonGlyphs = 1..2;

**property** NumGlyphs: TVrRadioButtonGlyphs;

### Description

NumGlyphs defines the number of button images contained in the Enabled and Disabled glyph images. Each button image represents a certain state. Atleast 1 button image must be defined.

Value	Meaning
-------	---------

---

1	Unchecked
---	-----------

2	Checked
---	---------

## **TvrBitmapRadioButton.OnChange**

[TvrBitmapRadioButton](#) [see also](#)

OnChange is called as soon as the control changes state.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called as soon as the control changes state (Checked property).



## **TVrBitmapRadioButton.Spacing**

[TVrBitmapRadioButton](#) [see also](#)

Defines the space between the Image and Caption text.

**property** Spacing: Integer;

### **Description**

Defines the space between the Image and Caption text.

## **TVrBitmapRadioButton.TextureIndex**

[TVrBitmapRadioButton](#) [see also](#)

Describes the image used to fill the background of the control.

**property** TextureIndex: Integer;

### **Description**

TextureIndex is used in combination of the BitmapList property. TextureIndex describes the image used to fill the background of the control.

## **TVrBitmapRadioButton.TextureStyle**

[TVrBitmapRadioButton](#) [see also](#)

Describes the way the background is filled.

**type** TVrRadioButtonTexture = (rbtTile, rbtStretch);

**property** TextureStyle: TVrRadioButtonTexture;

### **Description**

TextureStyle describes the way the background is filled.

## **TVrBitmapRadioButton.TransparentColor**

[TVrBitmapRadioButton](#) [see also](#)

Is used to make the Glyphs, contained in the BitmapList, transparent.

**property** TransparentColor: TColor;

### **Description**

TransparentColor is used to make the Glyphs, contained in the BitmapList, transparent. The color is masked out from the background which creates a transparent effect.

## TVrBitmaps

[properties](#)[methods](#)

[events](#)

[see also](#)

Defines a list of TBitmap objects.

### Unit

vrclasses

### Description

Use TVrBitmaps to maintain a list of TBitmap objects. In compare to the standard imagelist the images are added without any formatting. TVrBitmaps is derived from TVrPersistent.

## **TVrBitmaps.Add**

[TVrBitmaps](#) [see also](#)

Inserts a new bitmap to the end of a list.

```
function Add(Value: TBitmap): Integer;
```

### **Description**

Call Add to insert a new bitmap at the end of the Items array. Add returns the index of the new bitmap, where the first item in the list has an index of 0. Add allocates more memory if the Items array already uses up the Capacity of the list object. Add increases the value of Count to reflect the addition of a new bitmap.

Note: Add always inserts the bitmap at the end of the Items array, even if the Items array contains nil pointers.

## **TvrBitmaps.Bitmaps**

[TvrBitmaps](#) [see also](#)

Lists the object references.

**property** Bitmaps[Index: Integer]: TBitmap;

### **Description**

Use Bitmaps to obtain a pointer to a specific bitmap in the array. The Index parameter indicates the index of the object, where 0 is the index of the first object, 1 is the index of the second object, and so on. Set Items to change the reference at a specific location.

Use bitmaps with the Count property to iterate through all of the objects in the list.

## **TVrBitmaps.Clear**

[TVrBitmaps](#) [see also](#)

Deletes all bitmaps from the list.

**procedure** Clear;

### **Description**

Call Clear to empty the bitmap list and set the Count to 0. Clear also frees the memory used to store the Items array and sets the Capacity to 0.



## **TVrBitmaps.Count**

[TVrBitmaps](#) [see also](#)

Indicates the number of entries in the list that are in use.

**property** Count: Integer;

### **Description**

Read Count to determine the number of bitmaps in the Items array.

## **TVrBitmaps.Delete**

[TVrBitmaps](#) [see also](#)

Removes the bitmap at the position given by the Index parameter.

**procedure** Delete(Index: Integer);

### **Description**

Call Delete to remove the bitmap at a specific position from the list. The index is zero-based, so the first item has an Index value of 0, the second item has an Index value of 1, and so on. Calling Delete moves up all items in the Items array that follow the deleted item, and reduces the Count.

## **TvrBitmaps.Exchange**

[TvrBitmaps](#) [see also](#)

Swaps the position of two items in the Items array.

**procedure** Exchange(Index1, Index2: Integer);

### **Description**

Call Exchange to swap the positions of the items at positions Index1 and Index2 of the Items array. The indexes are zero-based, so the first item in the list has an index value of 0, the second item has an index value of 1, and so on.

## **TVrBitmaps.IndexOf**

[TVrBitmaps](#) [see also](#)

Returns the index of the first entry in the Items array with a specified value.

```
function IndexOf(Item: TBitmap): Integer;
```

### **Description**

Call IndexOf to get the index for a pointer in the Items array. Specify the pointer as the Item parameter. The first item in the array has index 0, the second item has index 1, and so on. If an item is not in the list, IndexOf returns -1. If a pointer appears more than once in the array, IndexOf returns the index of the first appearance.

## **TVrBitmaps.Insert**

[TVrBitmaps](#) [see also](#)

Adds an object to the Items array at the position specified by Index.

**procedure** Insert(Index: Integer; Item: TBitmap);

### **Description**

Call Insert to add Item to the middle of the Items array. The Index parameter is a zero-based index, so the first position in the array has an index of 0. Insert adds the item at the indicated position, shifting the item that previously occupied that position, and all subsequent items, up. Insert expands the Capacity of the list if necessary, and increases the Count property.

## **TVrBitmaps.LoadFromFile**

[TVrBitmaps](#) [see also](#)

Reads the file specified in FileName and loads the data into the TVrBitmaps object.

**procedure** LoadFromFile(const FileName: string);

### **Description**

Use LoadFromFile to read a list of bitmaps from disk. If the file is not recognized an exception is raised.

Note: The file must be created first with WriteToFile.

## **TVrBitmaps.LoadFromStream**

[TVrBitmaps](#) [see also](#)

Loads a list of bitmaps from a stream.

**procedure** LoadFromStream(Stream: TStream);

### **Description**

Use LoadFromStream to read a list of bitmaps from a stream.

## **TVrBitmaps.Move**

[TVrBitmaps](#) [see also](#)

Changes the position of an item in the Items array.

**procedure** Move(CurIndex, NewIndex: Integer);

### **Description**

Call Move to move the item at the position CurIndex so that it occupies the position NewIndex. CurIndex and NewIndex are zero-base indexes into the Items array.



## **TvrBitmaps.SaveToFile**

[TvrBitmaps](#) [see also](#)

Writes all the bitmaps to disk.

**procedure** SaveToFile(const FileName: string);

### **Description**

Use SaveToFile to save all the bitmaps to one single file specified in FileName.

## **TVrBitmaps.SaveToStream**

[TVrBitmaps](#) [see also](#)

Saves a list of bitmaps to a stream.

**procedure** SaveToStream(Stream: TStream);

### **Description**

Use SaveToStream to write all bitmaps to a stream.

## **TVrBlotter**

[properties](#)[methods](#)

[events](#)

[see also](#)

It provides methods to help manage the placement of child controls embedded inside the blotter component.

### **Unit**

vrblotter

### **Description**

TVrBlotter is a container control and is derived from TVrCustomControl. It provides methods to help manage the placement of child controls embedded inside the blotter component.

## **TVrBlotter.BackImage**

[TVrBlottersee also](#)

BackImage is used to fill the background of the control.

**property** BackImage: TBitmap;

### **Description**

BackImage is used to fill the background of the control. Using special background images ensures that there are no gaps visible between each image transition.

## **TVrBlotter.Bevel**

[TVrBlottersee also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## TVrBorder

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrBorder represents a beveled outline.

### Unit

vrborder

### Description

TVrBorder represents a beveled outline. Use TVrBorder to create beveled boxes, frames, or lines. The bevel can appear raised or lowered.

## **TVrBorder.HighlightColor**

[TVrBorder](#) [see also](#)

HighlightColor is one of the two colors which make the 3d effect.

**property** HighlightColor: TColor;

### **Description**

HighlightColor is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: [ShadowColor](#) and HighlightColor.

## **TVrBorder.ShadowColor**

[TVrBorder](#) [see also](#)

ShadowColor is one of the two colors which make the 3d effect.

**property** ShadowColor: TColor;

### **Description**

ShadowColor is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ShadowColor and HighlightColor.



## TVrBorder.Shape

[TVrBorder](#) [see also](#)

Determines the shape of the bevel.

```
type TVrBorderShape = (bsBox, bsFrame, bsTopLine, bsBottomLine, bsLeftLine,  
bsRightLine, bsSpacer);
```

```
property Shape: TVrBorderShape
```

### Description

Determines the shape of the bevel. Set Shape to specify whether the bevel appears as a line, box, frame, or space. These are the possible values of Shape:

Value	Meaning
bsBox	The entire client area appears raised or lowered, depending on the value of Style.
bsFrame	The client area is outlined by a raised or lowered frame.
bsTopLine	The bevel displays a line at the top of the client area.
bsBottomLine	The bevel displays a line at the bottom of the client area.
bsLeftLine	The bevel displays a line at the left side of the client area.
bsRightLine	The bevel displays a line at the right side of the client area.
bsSpacer	The bevel is an empty space.

## TVrBorder.Style

[TVrBorder](#) [see also](#)

Set Style to indicate whether the bevel should create a raised or a lowered effect.

```
type TVrBorderStyle = (bsLowered, bsRaised);
```

```
property Style: TVrBorderStyle;
```

### Description

Set Style to indicate whether the bevel should create a raised or a lowered effect. When the Shape property is bsBox, the entire client area appears raised or lowered. For all other values of Shape, the bevel displays a raised or lowered line along the edge or edges of the client area. These are the possible values of Style:

<b>Value</b>	<b>Meaning</b>
bsLowered	The bevel is lowered.
bsRaised	The bevel is raised.

## TVrCalendar

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrCalendar is used to select a number or graphical image from one of the cells within the control.

### Unit

vrcalendar

### Description

TVrCalendar is used to select a number or graphical image from one of the visible cells within the control. A cell becomes active when the user selects it with a OnClick event or when the visible property has been manually set within the Items property. TVrCalendar is derived from TVrGraphicControl.

## **TvCalendar.Alignment**

[TvCalendar](#) [see also](#)

Describes the location of the numbers within each cell.

**type** TVrGridAlignment = (gaUpperLeft, gaUpperRight, gaBottomLeft, gaBottomRight, gaCenter);  
**property** Alignment: TVrGridAlignment;

## **TVrCalendar.Bevel**

[TVrCalendar](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrCalendar.Columns**

[TVrCalendar](#) [see also](#)

Describes the number of visible columns.

```
type TVrColInt = 1..MaxInt;  
property Columns: TVrColInt;
```

### **Description**

Number of visible horizontal cells.

## **TVrCalendar.Count**

[TVrCalendar](#) [see also](#)

Count returns the number of visible cells within the control.

**property** Count: Integer;

### **Description**

Count returns the number of visible cells within the control (Columns \* Rows).

```
procedure TForm1.VrCalendar1Click(Sender: TObject);
var
  I: Integer;
begin
  with VrCalendar1 do
    for I := 0 to Count - 1 do
      Items[I].Active := True;
end;
```

## **TVrCalendar.Digits**

[TVrCalendar](#) [see also](#)

Digits describes the number of digits displayed in each cell.

**property** Digits: Integer;

### **Description**

Digits describes the number of digits used to displayed in each cell. Digits specifies the minimum number width. If the resulting number is shorter than the minimum number width, it is padded with zero's to increase the number width.



## TVrCalendar.DrawStyle

[TVrCalendar](#) [see also](#)

Implements custom drawing.

```
type TVrDrawStyle = (dsOwnerDraw, dsNormal);
```

```
property DrawStyle: TVrDrawStyle;
```

### Description

Set DrawStyle to dsOwnerDraw to override any default painting behavior in [OnDraw](#).

## **TvrCalendar.FirstIndex**

[TvrCalendar](#) [see also](#)

FirstIndex is the first number which is displayed within the grid.

**property** FirstIndex: Integer;

### **Description**

FirstIndex is the first number which is displayed within the grid. The last number in the grid depends on the number of cells.

## TVrCalendar.Grid

[TVrCalendar](#) [see also](#)

Defines the attributes of the grid.

```
typeTVrGridStyle = (gsRaised, gsLowered, gsSingle);  
typeTVrCalendarGrid = class(TPersistent)  
property Grid: TVrCalendarGrid;
```

### Description

Use Grid to customize the way the grid is painted.

Value	Meaning
Style, TVrGridStyle	gsRaised, gsLowered, gsSingle
Color, TColor	Color of the Grid, only applies when style is set to gsSingle .
Highlight3D, TColor	Color of the highlighted parts of the grid.
Shadow3D, TColor	Color of the shadow parts of the grid.
Width, Integer	Size of the grid lines.

## **TVrCalendar.ItemIndex**

[TVrCalendar](#) [see also](#)

Will return the actual cell being selected by the user after an OnClick event.

**property** ItemIndex: Integer;

### **Description**

This property will return the actual cell selected by the user after an OnClick event. Remember that the internal structure is zero based. When the user clicks cell 1 this property would return 0. If no valid cells are selected FirstIndex returns -1.

```
procedure TForm1.VrCalendar1Click(Sender: TObject);
begin
  with VrCalendar1 do
    if ItemIndex <> -1 then
      Items[ItemIndex].Active := True;
end;
```

## TVrCalendar.Items

[TVrCalendar](#) [see also](#)

Each cell within TVrCalendar has it's own properties.

```
type TVrCalendarItem = class(TVrCollectionItem);  
property Items[Index: Integer]: TVrCalendarItem;
```

### Description

Each cell within TVrCalendar has it's own properties. This is a runtime only property which can be used as follows:

```
Items[0].Active := true
```

This would activate the first cell. Valid ranges can be determined by examining the count property. In order to select the last cell:

```
Items[Count-1].Active := true
```

### property Active

- This property uses the palette property setting to visual change state.

### property Caption

- Text to display in the cell.

### property Visible

- Determines if the cell is visible and painted.

## **TVrCalendar.NextStep**

[TVrCalendar](#) [see also](#)

NextStep is an increment value from FirstIndex.

**property** NextStep: Integer;

### **Description**

NextStep is an increment value for FirstIndex.

## TVrCalendar.OnDraw

[TVrCalendar](#) [see also](#)

Implements custom painting within TVrCalendar.

**type** TOwnerDrawEvent = **procedure** (Sender: TObject; Canvas: TCanvas; Rect: TRect; Index: Integer; State: Boolean) of **object**;  
**property** OnDraw: TOwnerDrawEvent;

### Description

Write code in an OnDraw handler to draw to the Calendar's canvas before its items are painted.

```
procedure TForm1.VrCalendar2Draw(Sender: TObject; Canvas: TCanvas;  
  Rect: TRect; Index: Integer; State: Boolean);  
var  
  X, Y: Integer;  
begin  
  Canvas.Brush.Color := clBlack;  
  Canvas.FillRect(Rect);  
  X := Rect.Left + (WidthOf(Rect) - ImageList1.Width) div 2;  
  Y := Rect.Top + (HeightOf(Rect) - ImageList1.Height) div 2;  
  ImageList1.Draw(Canvas, X, Y, ord(State));  
end;
```

## TvrCalendar.Options

[TvrCalendar](#) [see also](#)

Specifies various display and behavioral properties of the grid.

```
type TvrCalendarOption = (coActiveClick, coMouseClip, coTrackMouse);  
type TvrCalendarOptions = set of TvrCalendarOption;  
property Options: TvrCalendarOptions;
```

### Description

<b>Value</b>	<b>Meaning</b>
coActiveClick	The selected/highlighted cell is activated on a mouse click.
coMouseClip boundaries.	On a mousebutton down event the cursor cannot leave the controls
coTrackMouse	Each cell is activated as the mouse is moved over it.



## **TVrCalendar.Orientation**

[TVrCalendar](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrCalendar.Palette**

[TVrCalendar](#) [see also](#)

Defines the color attributes for TVrCalendar.

### **Declaration**

property Palette: TVrPalette;

## **TVrCalendar.Reset**

[TVrCalendar](#) [see also](#)

Reset method which sets all active cells to false.

**procedure** Reset;

### **Description**

Call reset to set all active cells to false.

## **TVrCalendar.Rows**

[TVrCalendar](#) [see also](#)

Describes the number of visible rows within the controls boundaries.

### **Declaration**

```
type TVrRowInt = 1..MaxInt;  
property Rows: TVrRowInt;
```

### **Description**

Number of visible vertical rows.

## **TVrCheckLed**

[properties](#)[methods](#)

[events](#)

[see also](#)

A CheckLed component presents an option for the user.

### **Unit**

vrchecked

### **Description**

TVrCheckLed is derived from TVrCustomImageControl. A CheckLed component presents an option for the user. The user can check the box to select the option, or uncheck it to deselect the option. It behaves just like the standard windows checkboxes.

## **TVrCheckLed.Checked**

[TVrCheckLed](#) [see also](#)

Use Checked to toggle the state during runtime.

**property** Checked: Boolean;

### **Description**

TVrCheckLed knows two states, checked or not checked. When the control is checked this property is True. Use Checked also to toggle the state during runtime.

## **TvrCheckLed.CheckHeight**

[TvrCheckLed](#) [see also](#)

Use CheckHeight to increase or decrease the image boundaries.

**property** CheckHeight: Integer;

### **Description**

TvrCheckLed uses a button shaped image to change state. This occurs when a user clicks the image with a mouse or presses the space bar on the keyboard. Use CheckHeight to increase or decrease the image boundaries.

## **TVrCheckLed.CheckStyle**

[TVrCheckLed](#) [see also](#)

**type** TVrCheckLedStyle = (csRadioButton, csCheckbox);

**property** CheckStyle: TVrCheckLedStyle;



## **TVrCheckLed.CheckWidth**

[TVrCheckLed](#) [see also](#)

Use CheckWidth to increase or decrease the image boundaries.

**property** CheckWidth: Integer;

### **Description**

TVrCheckLed uses a button shaped image to change state. This occurs when a user clicks the image with a mouse or presses the space bar on the keyboard. Use CheckWidth to increase or decrease the image boundaries.

## TVrCheckLed.Layout

[TVrCheckLed](#) [see also](#)

Determines where the image or text appears on the button.

```
type TVrImageTextLayout = (ImageLeft, ImageRight, ImageTop, ImageBottom);
```

```
property Layout: TVrImageTextLayout;
```

### Description

Value	Meaning
ImageLeft	The image or caption appears near the left side of the button.
ImageRight	The image or caption appears near the right side of the button.
ImageTop	The image or caption appears near the top of the button.
ImageBottom	The image or caption appears near the bottom of the button.

## **TVrCheckLed.Margin**

[TVrCheckLed](#) [see also](#)

Use Margin to specify the indentation of the image or the text specified by the Caption property.

**property** Margin: Integer;

### **Description**

Use Margin to specify the indentation of the image or the text specified by the Caption property. The edges that Margin separates depends on the Layout property. If Layout is ImageLeft, the margin appears between the left edge of the image or caption and the left edge of the control. If Layout is ImageRight, the margin separates the right edges. If Layout is ImageTop, the margin separates the top edges, and if Layout is ImageBottom, the margin separates the bottom edges.

If Margin is -1 then the image or text are centered on the button.

## **TVrCheckLed.OnChange**

[TVrCheckLed](#) [see also](#)

OnChange is called as soon as the control changes state.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called as soon as the control changes state (Checked property).

## **TVrCheckLed.Palette**

[TVrCheckLed](#) [see also](#)

Defines the color attributes for TVrCheckLed.

**property** Palette: TVrPalette;

## **TVrCheckLed.Spacing**

[TVrCheckLed](#) [see also](#)

Defines the space between the Image and Caption text.

**property** Spacing: Integer;

### **Description**

Defines the space between the Image and Caption text.

## TVrClock

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrClock is an easy to use timer display in LCD style.

### Unit

vrlcd

### Description

TVrClock is an easy to use timer display in LCD style. TVrClock is derived from TVrNum. Like TVrNum you can change the style, colors and spacing of the display. Three clock types are supported RealTime, Elapsed, Custom.

## **TVrClock.Active**

[TVrClock](#) [see also](#)

Enable or disable the timer feature.

**property** Active: Boolean;

### **Description**

Enable or disable the timer feature. This property will be ignored with ctCustom clock type set. Although blink can be set in the designer, it is only activated during runtime sessions.



## **TVrClock.AutoSize**

[TVrClock](#)

[see also](#)

Determines if the control automatically changes it's boundaries to fit all digits.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit all digits. AutoSize only applies when Align is set to alNone.

## **TVrClock.Blink**

[TVrClock](#) [see also](#)

Makes the seperators blink.

**property** Blink: Boolean;

### **Description**

Set Blink to True to make the time seperators blink after each elapsed second. Does not apply if ClockType is ctCustom.

## TVrClock.ClockType

[TVrClock](#)

[see also](#)

Determines the type of clock to use.

```
type TVrClockType = (ctRealTime, ctElapsed, ctCustom);
```

```
property ClockType: TVrClockType;
```

### Description

<b>Value</b>	<b>Meaning</b>
ctRealtime	Returns the actual system time
ctElapsed	Returns the expired time since the start of the clock
ctCustom	User defined, for manual assigning values to the hour, minute and second properties.

## **TVrClock.Hours**

[TVrClock](#) [see also](#)

Describes the value for the hours segments.

```
type TVrHoursInt = 0..23;  
property Hours: TVrHoursInt;
```

### **Description**

Valid values for Hours: 0..23

This value can only be used with a "ctCustom" Clock type.

## **TVrClock.Hours24**

[TVrClock](#) [see also](#)

Handle the display of time in AM/PM format.

**property** Hours24: Boolean;

### **Description**

Set Hours24 to false in order to handle the display of time in AM/PM format, instead of a 24-hour military display.

## **TVrClock.Minutes**

[TVrClock](#) [see also](#)

Describes the value for the minutes segments.

```
type TVrMinutesInt = 0..59;  
property Minutes: TVrMinutesInt;
```

### **Description**

Valid values for Minutes: 0..59

These value can only be used with a "ctCustom" Clock type.

## **TVrClock.OnHoursChanged**

[TVrClock](#)

[see also](#)

This event will occur when the hours property changed.

```
type TVrHoursChangeEvent = procedure(Sender: TObject; Hours: Word) of object;  
property OnHoursChanged: TVrHoursChangeEvent;
```

### **Description**

This event will occur when the hours property changed.

## **TVrClock.OnMinutesChanged**

[TVrClock](#) [see also](#)

This event will occur when the minutes property changed.

```
type TVrMinutesChangeEvent = procedure(Sender: TObject; Minutes: Word) of  
object;  
property OnMinutesChanged: TVrMinutesChangeEvent;
```

### **Description**

This event will occur when the minutes property changed.



## **TVrClock.OnSecondsChanged**

[TVrClock](#) [see also](#)

This event will occur when the seconds property changed.

```
type TVrSecondsChangeEvent = procedure(Sender: TObject; Seconds: Word) of  
object;  
property OnSecondsChanged: TVrSecondsChangeEvent;
```

### **Description**

This event will occur when the seconds property changed.

## **TVrClock.Palette**

[TVrClock](#) [see also](#)

Defines the color attributes for TVrClock.

**property** Palette: TVrPalette;

## **TVrClock.Seconds**

[TVrClock](#) [see also](#)

Describes the value for the seconds segments.

```
type TVrSecondsInt = 0..59;  
property Seconds: TVrSecondsInt;
```

### **Description**

Valid values for Seconds: 0..59

These value can only be used with a "ctCustom" Clock type

## **TVrClock.ShowSeconds**

[TVrClock](#)

[see also](#)

Show or hide the "Seconds" segments.

**property** ShowSeconds: Boolean;

### **Description**

Show or hide the "Seconds" segments.

## **TVrClock.ShowTimeZone**

[TVrClock](#) [see also](#)

Determines if the AM/PM indicator is visible.

**property** ShowTimeZone: Boolean;

### **Description**

ShowTimeZone determines if the AM/PM indicator is visible. Applies only when clocktype is set to ctRealTime.

## **TVrClock.Style**

[TVrClock](#) [see also](#)

Describes the size of the digits.

**type** TVrNumStyle = (ns13x24, ns11x20, ns7x13, ns12x17, ns5x7);

**property** Style: TVrNumStyle;

## **TVrClock.Threaded**

[TVrClock](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrClock.Transparent**

[TVrClock](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.



## TVrCompass

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrCompass as a basic needle component.

### Unit

vrcompass

### Description

Use TVrCompass as a basic needle component. TVrCompass is designed to be used in combination with a graphical image of a meter or gauge. TVrCompass is derived from TGraphicControl.

## **TVrCompass.AutoSize**

[TVrCompass](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit the size of the graphical image.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit the size of the graphical image defined by BackImage.

## **TVrCompass.BackImage**

[TVrCompass](#)

[see also](#)

Used to fill the background of the control.

**property** BackImage: TBitmap;

### **Description**

BackImage is used to fill the background of the control. This could be a graphical image representing a meter or gauge.

## **TVrCompass.CircleColor**

[TVrCompass](#)

[see also](#)

Describes the fill color of the inner circle.

**property** CircleColor: TColor;

### **Description**

CircleColor describes the fill color of the inner circle.

## **TVrCompass.CircleOutlineColor**

[TVrCompass](#)

[see also](#)

Describes the color of the border painted around the inner circle.

**property** CircleOutlineColor: TColor;

### **Description**

CircleOutlineWidth describes the color of the border painted around the inner circle.

## **TVrCompass.CircleOutlineWidth**

[TVrCompass](#)

[see also](#)

Describes the thickness of the border painted around the inner circle.

**property** CircleOutlineWidth: Integer;

### **Description**

CircleOutlineWidth describes the thickness of the border painted around the inner circle.

## **TvrCompass.CircleWidth**

[TvrCompass](#)

[see also](#)

Defines the radius of the inner circle.

**property** CircleWidth: Integer;

### **Description**

CircleWidth defines the radius of the inner circle which with needle creates the compass image.

## **TvrCompass.Heading**

[TvrCompass](#)

[see also](#)

Defines the angle in which the needle is painted.

**property** Heading: Integer;

### **Description**

Heading defines the angle in which the needle is painted. Valid ranges are 0..360 degrees.



## **TVrCompass.NeedleColor**

[TVrCompass](#)

[see also](#)

Defines the color of the compass needle.

**property** NeedleColor: TColor;

### **Description**

NeedleWidth defines the color of the compass needle.

## **TvrCompass.NeedleLength**

[TvrCompass](#)

[see also](#)

Defines the length or radius of the compass needle.

**property** NeedleLength: Integer;

### **Description**

NeedleLength defines the length or radius of the compass needle.

## **TVrCompass.NeedleTransparent**

[TVrCompass](#)

[see also](#)

Creates a transparent effect for the needle.

**property** NeedleTransparent: Boolean;

### **Description**

Set NeedleTransparent to true to create a transparent effect for the needle.

## **TvrCompass.NeedleWidth**

[TvrCompass](#)

[see also](#)

Defines the thickness of the compass needle.

**property** NeedleWidth: Integer;

### **Description**

NeedleWidth defines the thickness of the compass needle.

## **TVrCompass.OnChange**

[TVrCompass](#) [see also](#)

Is called when the Heading property changed.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called when the Heading property changed.

## **TVrCompass.Transparent**

[TVrCompass](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TvrCopyFile**

[properties](#)[methods](#)

[events](#)

[see also](#)

VrCopyFile is used to copy a single file to a new destination on disk.

### **Unit**

vrsystem

### **Description**

VrCopyFile is a system tool used to copy a single file to a new destination on disk. This component includes some of the needed windows api calls and therefore simplifies the copying process by providing an interface on design level.

## **TvrCopyFile.AfterCopy**

[TvrCopyFile](#) [see also](#)

AfterCopy is called as soon as the specified source file is copied.

**property** AfterCopy: TNotifyEvent;

### **Description**

AfterCopy is called as soon as the specified source file is copied to the new destination file.



## TVrCopyFile.BeforeOpen

[TVrCopyFile](#) [see also](#)

After calling Execute in order to start the copying process, the file information of the source file is retrieved from disk.

```
type TVrOpenEvent = procedure(Sender: TObject; Size: Integer; Date, Time: TDateTime) of object;  
property BeforeOpen: TVrOpenEvent;
```

### Description

After calling Execute in order to start the copying process, the file information of the source file is retrieved from disk. The filesize and timestamp are passed as parameters.

## **TVrCopyFile.BeforeOverwrite**

[TVrCopyFile](#) [see also](#)

BeforeOverwrite is called if Overwrite (mode property) is set to omEvent and the file defined by DestFile already exists.

```
type TVrOverwriteEvent = procedure(Sender: TObject; var Overwrite: Boolean) of  
object;  
property BeforeOverwrite: TVrOverwriteEvent;
```

### **Description**

BeforeOverwrite is called if Overwrite (mode property) is set to omEvent and the file defined by DestFile already exists. Set the var parameter Overwrite to false in order to cancel the copying process.

## **TVrCopyFile.BufferSize**

[TVrCopyFile](#) [see also](#)

BufferSize determines the size of the internal databuffer.

**property** BufferSize: TVrMaxInt;

### **Description**

BufferSize determines the size of the internal databuffer used during the copying process. Increasing BufferSize will result in faster copying. Advised is to increase BufferSize with blocks of 1024k.

## **TVrCopyFile.CopyDateTime**

[TVrCopyFile](#) [see also](#)

If CopyDateTime is set to True, the date and timestamp of the original source file is also applied to the newly created DestFile.

**property** CopyDateTime: Boolean;

### **Description**

If CopyDateTime is set to True, the date and timestamp of the original source file is also applied to the newly created DestFile.

## **TvrCopyFile.DestFile**

[TvrCopyFile](#) [see also](#)

DestFile describes the destination file on disk or network.

**property** DestFile: string;

### **Description**

DestFile describes the destination file on disk or network. DestFile must include a full path including a valid filename.

## **TVrCopyFile.Execute**

[TVrCopyFile](#) [see also](#)

Calling execute will start the actual copying process.

**procedure** Execute;

### **Description**

Calling execute will start the actual copying process.

## **TvrCopyFile.OnProgress**

[TvrCopyFile](#) [see also](#)

OnProgress is called during copying of the source file.

```
type TvrProgressEvent = procedure(Sender: TObject; BytesCopied: Integer; var  
Cancel: Boolean) of object;  
property OnProgress: TvrProgressEvent;
```

### **Description**

OnProgress is called during copying of the source file. BytesCopied will contain the number of bytes already copied. Cancel can be set to false in order to cancel the copying process.

## **TvrCopyFile.Overwrite**

[TvrCopyFile](#) [see also](#)

If overwrite is set to omEvent and the file defined by DestFile already exists, the BeforeOverwrite event is triggered.

```
type TVrOverwriteMode = (omAlways, omEvent);  
property Overwrite: TVrOverwriteMode;
```

### **Description**

If overwrite is set to omEvent and the file defined by DestFile already exists, the BeforeOverwrite event is triggered. If set to omAlways, the file defined by DestFile is always overwritten without any notification.



## **TvrCopyFile.SourceFile**

[TvrCopyFile](#) [see also](#)

SourceFile describes the source file on disk or network.

**property** SourceFile: string;

### **Description**

SourceFile describes the source file on disk or network. SourceFile must include a full path including a valid existing filename.

## **TVrCopyFile.Terminate**

[TVrCopyFile](#) [see also](#)

Call Terminate in order to cancel the copying process.

**procedure** Terminate;

### **Description**

Call Terminate in order to cancel the copying process.

## **TVrCounter**

[properties](#)[methods](#)

[events](#)

[see also](#)

Graphical control to represent any number of 1 to 10 digits.

### **Unit**

vrdesign

### **Description**

TVrCounter is derived from TGraphicControl. There are several properties available to customize the control. TVrCounter needs a graphical image to display values. This graphical image is a TBitmap type with a fixed format with digits from zero to nine: "0123456789".

## **TVrCounter.AutoSize**

[TVrCounter](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit all digits.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit all digits.

## **TVrCounter.Bitmap**

[TVrCounter](#) [see also](#)

Graphical image which contains the numeric values to display.

**property** Bitmap: TBitmap;

### **Description**

TVrCounter needs a graphical image to display values. This graphical image is a TBitmap type with a fixed format with digits ranging from zero to nine: "0123456789".

## **TVrCounter.Digits**

[TVrCounter](#) [see also](#)

Used to increase or decrease the number of visible digits.

```
type TVrCounterDigits = 1..10;  
property Digits: TVrCounterDigits;
```

### **Description**

Use digits to increase or decrease the number of visible digits. When AutoSize is true the digits property is also used to calculate the new size of the control.

## **TVrCounter.OnChange**

[TVrCounter](#) [see also](#)

Is called when the Value property changed.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called when the Value property changed.

## **TVrCounter.Spacing**

[TVrCounter](#) [see also](#)

Defines the space between each single digit.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single digit.



## **TvrCounter.Stretch**

[TvrCounter](#) [see also](#)

Indicates whether the image should be changed so that it exactly fits the bounds of the image control.

**property** Stretch: Boolean;

### **Description**

Set Stretch to True to cause the image to assume the size and shape of the image control. When the image control resizes, the image resizes also. Stretch resizes the height and width of the image independently. Thus, unlike a simple change in magnification, stretch can distort the image if the image control is not the same shape as the image.

To resize the control to the image rather than resizing the image to the control, use the AutoSize property instead.

## **TvrCounter.Transparent**

[TvrCounter](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrCounter.Value**

[TVrCounter](#) [see also](#)

Contains the actual value which is displayed.

```
type TVrCounterValue = 0..MaxInt;
```

```
property Value: TVrCounterValue;
```

### **Description**

The value property contains the actual value which is displayed. Values can range from zero to MaxInt.

## **TVrDemoButton**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrDemoButton is a push button control with additional features.

### **Unit**

vrbuttons

### **Description**

TVrDemoButton is a 3d button control with a additional features. Use TVrDemoButton to put a Windows push button on a form. TVrDemoButton introduces several properties to control its behavior. Users choose button controls to initiate actions.

## **TVrDemoButton.BevelWidth**

[TVrDemoButton](#)

[see also](#)

Describes the size of the 3d border of the button.

```
type TVrByteInt = 0..255;  
property BevelWidth: TVrByteInt;
```

### **Description**

Describes the size of the 3d border of the button.

## **TVrDemoButton.Bitmap**

[TVrDemoButton](#)

[see also](#)

Defines the texture or background of the button.

**property** Bitmap: TBitmap;

### **Description**

The buttonface of TDemoButton will be filled (tiled) with the image defined by Bitmap.

## **TVrDemoButton.DisabledTextColor**

[TVrDemoButton](#)

[see also](#)

Describes the color for disabled text.

**property** DisabledTextColor: TColor;

### **Description**

DisabledTextColor describes the color for disabled text. DisabledTextColor overrides the existing font color when enabled is set to false.

## **TVrDemoButton.Flat**

[TVrDemoButton](#)

[see also](#)

Support for flat styled buttontypes.

**property** Flat: Boolean;

### **Description**

Button now also supports flat styled buttontypes.



## **TVrDemoButton.FocusColor**

[TVrDemoButton](#)

[see also](#)

Color for the outline when the DemoButton becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the DemoButton receives focus, a small rectangle is painted around the button with the color defined by FocusColor.

## **TVrDemoButton.Font3D**

[TVrDemoButton](#) [see also](#)

Defines the attributes of the text.

```
type TVrFont3D = class(TVrPersistent)
property Font3D: TVrFont3D;
```

## **TVrDemoButton.FontEnter**

[TVrDemoButton](#)

[see also](#)

Specifies the active text font of the control.

**property** FontEnter: TFont;

### **Description**

FontEnter becomes the active text font as soon as the mouse is moved within the controls boundaries.

## **TVrDemoButton.FontLeave**

[TVrDemoButton](#)

[see also](#)

Specifies the active text font of the control.

**property** FontLeave: TFont;

### **Description**

FontLeave becomes the active text font as soon as the mouse is moved outside the controls boundaries.

## **TvrDemoButton.HighlightColor**

[TvrDemoButton](#)

[see also](#)

HighlightColor is one of the two colors which make the 3d effect.

**property** HighlightColor: TColor;

### **Description**

HighlightColor is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: [ShadowColor](#) and HighlightColor.

## **TVrDemoButton.OutlineColor**

[TVrDemoButton](#)

[see also](#)

Describes the attributes of a small rectangle painted around the button image.

**property** OutlineColor: TColor;

### **Description**

OutlineColor defines the color which is used to paint a small outline rectangle around the button image.

## **TVrDemoButton.OutlineWidth**

[TVrDemoButton](#)

[see also](#)

Describes the size in pixels of the outline which is painted as a small rectangle.

**property** OutlineWidth: Integer;

### **Description**

OutlineWidth describes the size in pixels of the outline which is painted as a small rectangle.

## **TVrDemoButton.ShadowColor**

[TVrDemoButton](#)

[see also](#)

ShadowColor is one of the two colors which make the 3d effect.

**property** ShadowColor: TColor;

### **Description**

ShadowColor is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ShadowColor and [HighlightColor](#).



## **TVrDemoButton.TextAlignment**

[TVrDemoButton](#)

[see also](#)

Controls the placement of the text within the image control.

**type** TVrTextAlignment = (vtaLeft, vtaCenter, vtaRight, vtaTopLeft, vtaTop, vtaTopRight, vtaBottomLeft, vtaBottom, vtaBottomRight);

**property** TextAlignment: TVrTextAlignment;

### **Description**

Set TextAlignment to specify how the text is aligned within the ClientRect of the image control.

## **TVrDeskTop**

[properties](#)[methods](#)

[events](#)

[see also](#)

Use VrDeskTop to create custom backgrounds.

### **Unit**

vrdesktop

### **Description**

Use TVrDeskTop to create custom backgrounds. After assigning a bitmap to the Glyph property the background will be tiled (Wallpaper). TVrDeskTop is derived from TVrGraphicControl.

## **TVrDeskTop.FormDrag**

[TVrDeskTop](#) [see also](#)

Move the owner form by dragging the desktop control.

**property** FormDrag: Boolean;

### **Description**

Set FormDrag to true in order to move the owner form by dragging the desktop control.

## **TVrDeskTop.Glyph**

[TVrDeskTop](#) [see also](#)

Glyph is used to fill the background of the controls client area.

**property** Glyph: TBitmap;

### **Description**

Glyph is used to fill the background of the controls client area.

## TVrDigit

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrDigit represents a single digit with a decimal point.

### Unit

vrdigit

### Description

TVrDigit represents a single digit with a decimal point. In compare to the TVrNum component the digits are fully scalable. Resizing the control boundaries will make the digit appear to be larger and thicker.

## **TVrDigit.ActiveOnly**

[TVrDigit](#) [see also](#)

Determines if only active segments are visible.

**property** ActiveOnly: Boolean;

### **Description**

When ActiveOnly is true only active segments are displayed.

## **TVrDigit.OnChange**

[TVrDigit](#)

[see also](#)

OnChange is called when the value property is changed.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called when the Value property is changed.

## **TVrDigit.OutlineColor**

[TVrDigit](#) [see also](#)

Outlinecolor defines the bordercolor of each segment.

**property** OutlineColor: TColor;

### **Description**

Outlinecolor defines the bordercolor of each segment.



## **TVrDigit.Palette**

[TVrDigit](#) [see also](#)

Defines the color attributes for TVrDigit.

**property** Palette: TVrPalette;

## **TVrDigit.Transparent**

[TVrDigit](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TvrDigit.Value**

[TvrDigit](#) [see also](#)

Defines the digit displayed.

**property** Value: Integer;

### **Description**

Value defines the digit displayed in the controls boundaries. Use [ValueBinary](#) in order to activate the decimal point.

## **TVrDigit.ValueBinary**

[TVrDigit](#) [see also](#)

Binary representation of the Value property.

**property** ValueBinary: Byte;

### **Description**

Binary representation of the Value property. With ValueBinary you can control each separate led segment including the decimal point.

## **TVrDirScan**

[properties](#)[methods](#)

[events](#)

[see also](#)

Is used to locate files on local or network drives.

### **Unit**

vrsystem

### **Description**

TVrDirScan is a non visible component and it is used to locate files on local or network drives. Files to locate are described by a filemask which can contain wildcards.

## TVrDirScan.Attributes

[TVrDirScan](#) [see also](#)

Scan for certain files.

```
type TFileAttribute = (fatArchive, fatReadOnly, fatHidden, fatSystem,  
fatDirectory);
```

```
property Attributes: TFileAttributes;
```

### Description

Attributes property to scan for certain files. TVrDirscan can now also locate folders.

## **TVrDirScan.Cancel**

[TVrDirScan](#) [see also](#)

Used to terminate the scanning process.

**procedure** Cancel;

### **Description**

Use cancel to terminate the scanning process.

## **TVrDirScan.Execute**

[TVrDirScan](#) [see also](#)

Used to start the scanning process.

**procedure** Execute;

### **Description**

Use Execute to start the scanning process. For each file found the OnLocate event is triggered.



## **TVrDirScan.Mask**

[TVrDirScan](#) [see also](#)

Describes the files to look for.

**property** Mask: string;

### **Description**

Mask describes the files to look for. Mask can contain any valid filename or wildcard. Wildcards like \*.\* (all files) or \*.exe (only .exe files).

## **TVrDirScan.OnLocate**

[TVrDirScan](#) [see also](#)

Is called for each file found during the scanning process.

```
type TVrLocateEvent = procedure(Sender: TObject; Path: string; SearchRec: TSearchRec; var Cancel: Boolean) of object;  
property OnLocate: TVrLocateEvent;
```

### **Description**

The OnLocate event is called for each file found during the scanning process. Set Cancel to True in order to terminate the scanning process. Path contains the current folder on a local or network drive, while SearchRec contains the actual file information.

## **TVrDirScan.OnNotify**

[TVrDirScan](#) [see also](#)

Is called when there are no more files found or when the scanning process is cancelled.

**property** OnNotify: TNotifyEvent;

### **Description**

OnNotify is called when there are no more files found or when the scanning process is terminated with "cancel".

## TVrDirScan.OnPathChange

[TVrDirScan](#) [see also](#)

Called when a new folder is about to be scanned.

```
type TVrPathChange = procedure(Sender: TObject; const Path: string) of object;  
property OnPathChange: TVrPathChange;
```

### Description

OnPathChange is called when a new folder is about to be scanned.

## **TVrDirScan.Path**

[TVrDirScan](#) [see also](#)

Describes the starting folder or network location to scan for files.

**property** Path: string;

### **Description**

Path describes the starting folder or network location to scan for files. Path can contain any valid drive mapping to a local or network drive including sub folders.

## **TVrDirScan.Recursive**

[TVrDirScan](#) [see also](#)

Scan the location defined by the Path property and all it's sub folders.

**property** Recursive: Boolean;

### **Description**

Set Recursive to True in order to scan the location defined by the Path property and all it's sub folders. If Recursive is set to false only the folder defined by Path is scanned.

## **TVrDisplay**

[properties](#)[methods](#)

[events](#)

[see also](#)

a panel like control.

### **Unit**

vrdisplay

### **Description**

TVrDisplay is derived from TVrCustomControl. It is a LCD panel styled control which can contain it's own set of controls. Second, it contains a shadow and seperate bevel which make up the 3d effect.

## **TVrDisplay.Bevel**

[TVrDisplay](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;



## **TVrDisplay.ShadowColor1**

[TVrDisplay](#) [see also](#)

Defines the source color used to generate the shadow image.

**property** ShadowColor1: TColor;

### **Description**

ShadowColor1 defines the source color used to generate the shadow image.

## **TVrDisplay.ShadowColor2**

[TVrDisplay](#) [see also](#)

Defines the target color used to generate the shadow image.

**property** ShadowColor2: TColor;

### **Description**

ShadowColor2 defines the target color used to generate the shadow image.

## TVrDisplay.ShadowLayout

[TVrDisplay](#) [see also](#)

Defines the position of the shadow.

```
type TVrShadowLayout = (soTopLeft, soTopRight, soBottomLeft, soBottomRight);  
property ShadowLayout: TVrShadowLayout;
```

## **TVrDisplay.ShadowWidth**

[TVrDisplay](#) [see also](#)

Defines the width of the 3d shadow.

**property** ShadowWidth: Integer;

### **Description**

ShadowWidth defines the width of the 3d shadow.

## **TVrFont3D**

[properties](#)[methods](#)

[events](#)

[see also](#)

Defines the attributes of the text which is painted in the client area.

### **Unit**

vrclasses

### **Description**

Use TVrFont3D to create beveled text. The text can appear raised or lowered. The VrFont3D object is derived from TPersistent and contains several separate properties.

## **TVrFont3D.HighlightColor**

[TVrFont3D](#)

[see also](#)

s one of the two colors which make the 3d effect.

**property** HighlightColor: TColor;

### **Description**

ColorHighlight is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: [ShadowColor](#) and HighlightColor.

## **TVrFont3D.HighlightDepth**

[TVrFont3D](#)

[see also](#)

Describes the offset from the original text position

**property** HighlightDepth: Integer;

### **Description**

HighlightDepth describes the offset from the original text position and is used to paint the highlighted text area.

## **TVrFont3D.ShadowColor**

[TVrFont3D](#)

[see also](#)

Is one of the two colors which make the 3d effect.

**property** ShadowColor: TColor;

### **Description**

ColorShadow is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ShadowColor and HighlightColor.



## **TVrFont3D.ShadowDepth**

[TVrFont3D](#)

[see also](#)

Describes the offset from the original text position

**property** ShadowDepth: Integer;

### **Description**

ShadowDepth describes the offset from the original text position and is used to paint the shadow text area.

## **TVrFont3D.Style**

[TVrFont3D](#)

[see also](#)

Describes the type of effect which is used to paint the text.

```
type TVrFont3DStyle = (f3dNone, f3dRaised, f3dSunken, f3dShadow);  
property Style: TVrFont3DStyle;
```

### **Description**

Describes the type of effect which is used to paint the text.

## **TVrFormShape**

[properties](#)[methods](#)

[events](#)

[see also](#)

used to give a standard windows form a new shape.

### **Unit**

vrformshape

### **Description**

TVrFormShape is derived from TGraphicControl and is used to give a standard windows form a new shape. This new shape is defined by a graphical bitmap image. By default each image is rendered during design-time and stored in the .dfm But replacing the mask during runtime is also allowed.

This component can only be used when directly placed on a form!

## **TVrFormShape.Mask**

[TVrFormShape](#) [see also](#)

Contains the actual bitmap which defines the new form shape.

**property** Mask: TBitmap;

### **Description**

Mask contains the actual bitmap which defines the new form shape. Use [MaskColor](#) to make parts of the image transparent. When clicked on a form the control is aligned within it's parent boundaries.

## **TVrFormShape.MaskColor**

[TVrFormShape](#)

[see also](#)

Defines the transparent area's of the bitmap image defined by Mask.

**property** MaskColor: TColor;

### **Description**

MaskColor defines the transparent area's of the bitmap image defined by Mask. Applies only when a valid bitmap is loaded into the Mask bitmap.

## TVrGauge

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrGauge is an easy to use control in LCD style.

### Unit

vrgauge

### Description

TVrGauge is an easy to use control in LCD style. TVrGauge is derived from TVrGraphicControl. With the properties minvalue, maxvalue and position you can monitor processes like cd-playing time, communications, file transfers etc.

## **TVrGauge.ActiveClick**

[TVrGauge](#) [see also](#)

Clicking inside the controls boundaries updates the position.

**property** ActiveClick: Boolean;

Clicking inside the controls boundaries updates the position.

## **TVrGauge.Bevel**

[TVrGauge](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;



## **TVrGauge.MaxValue**

[TVrGauge](#) [see also](#)

Use MaxValue to set a upper limit to the value that can be represented using the gauge.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set a upper limit to the value that can be represented using the gauge. The highlighted area indicates the current position in a range between MinValue and MaxValue.

## **TvrGauge.MinValue**

[TvrGauge](#) [see also](#)

Use MinValue to set a lower limit to the value that can be represented using the gauge.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the gauge. The highlighted area indicates the current position in a range between MinValue and MaxValue.

## **TVrGauge.OnChange**

[TVrGauge](#) [see also](#)

OnChange event occurs when you assign another value to the position property.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange event occurs when you assign another value to the position property.

## **TVrGauge.OnMaxValue**

[TVrGauge](#) [see also](#)

Occurs when Position reaches the value defined by MaxValue.

**property** OnMaxValue: TNotifyEvent;

### **Description**

OnMaxValue occurs when Position reaches the value defined by MaxValue.

## **TvrGauge.OnMinValue**

[TvrGauge](#) [see also](#)

Occurs when Position reaches the value defined by MinValue.

**property** OnMinValue: TNotifyEvent;

### **Description**

OnMinValue occurs when Position reaches the value defined by MinValue.

## **TVrGauge.Orientation**

[TVrGauge](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrGauge.Palette**

[TVrGauge](#) [see also](#)

Defines the color attributes for TVrGauge.

**property** Palette: TVrPalette;

## **TvrGauge.PercentDone**

[TvrGauge](#) [see also](#)

Returns the current position calculated against the min and max property values.

**property** PercentDone: Integer;

### **Description**

PercentDone returns the current position calculated against the min and max property values. This is a runtime only property.



## **TVrGauge.Position**

[TVrGauge](#) [see also](#)

Points to the current state or progress made.

**property** Position: Integer;

### **Description**

Position points to the current state or progress made. The position can never exceed the values defined by the MinValue and MaxValue properties.

## **TVrGauge.SolidFill**

[TVrGauge](#) [see also](#)

Determines if each single step is filled with a dither pattern.

**property** SolidFill: Boolean;

### **Description**

When SolidFill is false each single step is filled with a dither pattern.

## **TVrGauge.Spacing**

[TVrGauge](#) [see also](#)

Increase or decrease the space between each single step.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single step within the image with Spacing.

## **TVrGauge.Step**

[TVrGauge](#) [see also](#)

Specifies the amount that Position increases when the StepIt method is called.

**property** Step: Integer;

### **Description**

Specifies the amount that Position increases when the StepIt method is called.

Set Step to specify the granularity of the gauge. Step should reflect the size of each step in the process tracked by the gauge, in the logical units used by the MaxValue and MinValue properties.

When a gauge is created, MinValue and MaxValue represent percentages, where MinValue is 0 (0% complete) and MaxValue is 100 (100% complete). If these values are not changed, Step is the percentage of the process completed before the user is provided with additional visual feedback.

When the StepIt method is called, the value of Position increases by Step.

## **TVrGauge.StepBy**

[TVrGauge](#) [see also](#)

Advances the Position of the level bar by a specified amount.

**procedure** StepBy(Delta: Integer);

### **Description**

Advances the Position of the level bar by a specified amount. Call StepBy to increase the value of Position by the value of the Delta parameter. To advance Position by a default amount that represents a single step in the process, use the StepIt method.

## **TVrGauge.StepIt**

[TVrGauge](#) [see also](#)

Call the StepIt method to increase the value of Position by the value of the Step property.

**procedure** StepIt;

### **Description**

Advances Position by the amount specified in the Step property.

Call the StepIt method to increase the value of Position by the value of the Step property. If Step represents the size of one logical step in the process tracked by the gauge, call Step after each logical step is completed.

## TVrGauge.Style

[TVrGauge](#) [see also](#)

Determines the direction of the gauge.

**type** TVrProgressStyle = (psBottomLeft, psTopRight);

**property** Style: TVrProgressStyle;

### Description

Position points to the current state or progress made. With psBottomLeft progress is displayed from Bottom-to-Top or from Left-to-Right (default). Use psTopRight to change the starting point of the display.

## **TVrGauge.TickHeight**

[TVrGauge](#) [see also](#)

Defines the thickness or resolution of each single step.

**property** TickHeight: Integer;

### **Description**

TVrGauge is build out of steps. Tickheight defines the thickness or resolution of each single step.



## **TVrGradient**

[properties](#)[methods](#)

[events](#)

[see also](#)

Used for creating multi colored backgrounds on forms or panels.

### **Unit**

vrgradient

### **Description**

TVrGradient is a component for creating multi colored backgrounds on forms or panels.

System requirements: atleast 256 color mode.

## **TVrGradient.ColorWidth**

[TVrGradient](#) [see also](#)

Describes the size of each color transition.

**property** ColorWidth: Integer;

### **Description**

ColorWidth can be used to speed up the painting process. Assigning a small value will make the gradient look smoother but will take a longer time to refresh on screen.

## **TVrGradient.Direction**

[TVrGradient](#)    [see also](#)

**type** TVrGradDirection = (gdUpDown, gdUpDownHalf, gdLeftRight, gdLeftRightHalf, gdChord1, gdChord2);

**property** Direction: TVrGradDirection;

## **TVrGradient.EndColor**

[TVrGradient](#) [see also](#)

Is one of the two colors used to create the gradient fill.

**property** EndColor: TColor;

### **Description**

The EndColor property is one of the two colors used to create the gradient fill. The gradient fill is composed of two colors: [StartColor](#) and EndColor.

## **TVrGradient.FormDrag**

[TVrGradient](#) [see also](#)

Move the owner form by dragging the gradient control.

**property** FormDrag: Boolean;

### **Description**

Set FormDrag to true in order to move the owner form by dragging the gradient control.

## **TVrGradient.StartColor**

[TVrGradient](#) [see also](#)

Is one of the two colors used to create the gradient fill.

**property** StartColor: TColor;

### **Description**

The StartColor property is one of the two colors used to create the gradient fill. The gradient fill is composed of two colors: StartColor and [EndColor](#).

## **TVrGradient.SwapColors**

[TVrGradient](#) [see also](#)

Move the owner form by dragging the gradient control.

**property** SwapColors: Boolean;

### **Description**

Set swapcolors to true in order to exchange the start and end colors during painting. This way there is no need to change the assigned colors manually.

## **TVrHotImage**

[properties](#)[methods](#)

[events](#)

[see also](#)

A web link styled component.

### **Unit**

VrHyperCtrls

### **Description**

A web link styled component. Moving the mouse within the controls boundaries makes the color attributes or image to change. This way the user can press the control in order to make a selection. TVrHotImage is more like a normal graphics control without real internet transport facilities.



## **TVrHotImage.ColorEnter**

[TVrHotImage](#) [see also](#)

Specifies the active background color of the control.

**property** ColorEnter: TColor;

### **Description**

ColorEnter becomes the active background color as soon as the mouse is moved within the controls boundaries.

## **TVrHotImage.ColorLeave**

[TVrHotImage](#) [see also](#)

Specifies the active background color of the control.

**property** ColorLeave: TColor;

### **Description**

ColorLeave becomes the active background color as soon as the mouse is moved outside the controls boundaries.

## TVrHotImage.DrawStyle

[TVrHotImage](#) [see also](#)

Indicates whether the image should be changed so that it exactly fits the bounds of the image control.

**type** TVrHotImageDrawStyle = (dsCenter, dsStretch);

**property** DrawStyle: TVrHotImageDrawStyle

### Description

Set DrawStyle to dsStretch to cause the image to assume the size and shape of the image control. When the image control resizes, the image resizes also. Stretch resizes the height and width of the image independently. Thus, unlike a simple change in magnification, stretch can distort the image if the image control is not the same shape as the image.

## **TVrHotImage.HotRect**

[TVrHotImage](#) [see also](#)

Defines if a rectangle is shown as soon as the mouse enters the controls boundaries.

**property** HotRect: TVrHotRect;

## **TVrHotImage.ImageEnter**

[TVrHotImage](#) [see also](#)

Specifies the visible graphical image of the control.

**property** ImageEnter: TBitmap;

### **Description**

ImageEnter becomes the active or visible graphical image as soon as the mouse is moved within the controls boundaries.

## **TVrHotImage.ImageLeave**

[TVrHotImage](#) [see also](#)

Specifies the visible graphical image of the control.

**property** ImageLeave: TBitmap;

### **Description**

ImageLeave becomes the active or visible graphical image as soon as the mouse is moved outside the controls boundaries.

## **TVrHotImage.OnFontChanged**

Is called when the font has changed.

**property** OnFontChanged;

### **Description**

OnFontChanged is called when the font has changed.

## **TVrHotImage.OnMouseEnter**

[TVrHotImage](#) [see also](#)

Occurs when the mouse is moved within the controls boundaries.

**property** OnMouseEnter: TNotifyEvent;

### **Description**

Use the OnMouseEnter event handler to cause any special processing to occur when the mouse is moved within the controls boundaries.



## **TVrHotImage.OnMouseLeave**

[TVrHotImage](#) [see also](#)

Occurs when the mouse is moved outside the controls boundaries.

**property** OnMouseLeave: TNotifyEvent;

### **Description**

Use the OnMouseLeave event handler to cause any special processing to occur when the mouse is moved outside the controls boundaries.

## **TVrHotImage.TextAlignment**

[TVrHotImage](#) [see also](#)

Controls the placement of the text within the image control.

```
type TVrTextAlignment = (vtaLeft, vtaCenter, vtaRight, vtaTopLeft, vtaTop, vtaTopRight, vtaBottomLeft, vtaBottom, vtaBottomRight);
```

```
property TextAlignment: TVrTextAlignment;
```

### **Description**

Set TextAlignment to specify how the text is aligned within the ClientRect of the image control.

## **TVrHotRect**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrHotRect provides some additional output styles.

### **Unit**

vrhotimage

### **Description**

TVrHotRect provides some additional output styles. If Visible is set to True a rectangle is painted around the client area of the control.

Note: Only used when the mouse is moved inside the controls boundaries.

## **TVrHotRect.Color**

[TVrHotRect](#) [see also](#)

Determines the color of the rectangle.

**property** Color: TColor;

## **TVrHotRect.Visible**

[TVrHotRect](#) [see also](#)

Determines if the HotRect attributes are used.

**property** Visible: Boolean;

## **TVrHotRect.Width**

[TVrHotRect](#) [see also](#)

Determines the width of the rectangle.

**property** Width: Integer;

## **TVrHyperButton**

[properties](#)[methods](#)

[events](#)

[see also](#)

A web link styled component.

### **Unit**

vrbuttons

### **Description**

A web link styled component. Moving the mouse within the controls boundaries makes the color attributes and outline of the button to change. This way the user can press the control in order to make a selection. TVrHyperButton is more like a normal graphics control without real internet transport facilities.

## **TVrHyperButton.BorderColor**

[TVrHyperButton](#) [see also](#)

Describes the color attribute of the button outline in it's lowered state.

**property** BorderColor: TColor;

### **Description**

BorderColor describes the color attribute of the button outline in it's lowered state.



## **TVrHyperButton.BorderHighlight**

[TVrHyperButton](#) [see also](#)

Describes the highlighted color attribute of the button outline.

**property** BorderHighlight: TColor;

### **Description**

BorderHighlight describes the highlighted color attribute of the button outline in it's raised state (mouse is moved over the button).

## **TVrHyperButton.BorderShadow**

[TVrHyperButton](#) [see also](#)

Describes the shadow color attribute of the button outline.

**property** BorderShadow: TColor;

### **Description**

BorderShadow describes the shadow color attribute of the button outline in it's raised state (mouse is moved over the button).

## **TVrHyperButton.ColorEnter**

[TVrHyperButton](#) [see also](#)

Specifies the active background color of the control.

**property** ColorEnter: TColor;

### **Description**

ColorEnter becomes the active background color as soon as the mouse is moved within the controls boundaries.

## **TVrHyperButton.ColorLeave**

[TVrHyperButton](#) [see also](#)

Specifies the active background color of the control.

**property** ColorLeave: TColor;

### **Description**

ColorLeave becomes the active background color as soon as the mouse is moved outside the controls boundaries.

## **TVrHyperButton.DisabledAnimate**

[TVrHyperButton](#) [see also](#)

Disable all animations when the mouse is moved over the button.

**property** DisabledAnimate: Boolean;

### **Description**

Set DisabledAnimate to false to disable all animations when the mouse is moved over the button. This only applies when Enabled is set to false.

## **TVrHyperButton.DisabledText**

[TVrHyperButton](#) [see also](#)

Describes the color for disabled text.

**property** DisabledText: TColor;

### **Description**

DisabledText describes the color for disabled text. DisabledText overrides the existing font color when enabled is set to false.

## **TVrHyperButton.Glyph**

[TVrHyperButton](#) [see also](#)

Describes a separate Glyph image.

**property** Glyph: TBitmap;

### **Description**

TVrHyperButton can have a separate Glyph image. The glyph is painted transparent by using the glyphs own transparent color.

## TVrHyperButton.Layout

[TVrHyperButton](#) [see also](#)

Determines where the image or text appears on the button.

**type** TVrImageTextLayout = (ImageLeft, ImageRight, ImageTop, ImageBottom);

**property** Layout: TVrImageTextLayout;

### Description

Value	Meaning
ImageLeft	The image or caption appears near the left side of the button.
ImageRight	The image or caption appears near the right side of the button.
ImageTop	The image or caption appears near the top of the button.
ImageBottom	The image or caption appears near the bottom of the button.



## **TVrHyperButton.Margin**

[TVrHyperButton](#) [see also](#)

Use Margin to specify the indentation of the image or the text specified by the Caption property.

**property** Margin: Integer;

### **Description**

Use Margin to specify the indentation of the image or the text specified by the Caption property. The edges that Margin separates depends on the Layout property. If Layout is ImageLeft, the margin appears between the left edge of the image or caption and the left edge of the control. If Layout is ImageRight, the margin separates the right edges. If Layout is ImageTop, the margin separates the top edges, and if Layout is ImageBottom, the margin separates the bottom edges.

If Margin is -1 then the image or text are centered on the button.

## **TVrHyperButton.OnMouseEnter**

[TVrHyperButton](#) [see also](#)

Occurs when the mouse is moved within the controls boundaries.

**property** OnMouseEnter: TNotifyEvent;

### **Description**

Use the OnMouseEnter event handler to cause any special processing to occur when the mouse is moved within the controls boundaries.

## **TVrHyperButton.OnMouseLeave**

[TVrHyperButton](#) [see also](#)

Occurs when the mouse is moved outside the controls boundaries.

**property** OnMouseLeave: TNotifyEvent;

### **Description**

Use the OnMouseLeave event handler to cause any special processing to occur when the mouse is moved outside the controls boundaries.

## **TVrHyperButton.Spacing**

[TVrHyperButton](#) [see also](#)

Defines the space between the Image and Caption text.

**property** Spacing: Integer;

### **Description**

Defines the space between the Image and Caption text.

## **TVrHyperButton.Transparent**

[TVrHyperButton](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrHyperLink**

[properties](#)[methods](#)

[events](#)

[see also](#)

A web link styled component.

### **Unit**

VrHyperCtrls

### **Description**

A web link styled component. Moving the mouse within the controls boundaries makes the text or background attributes to change. This way the user can press the control in order to make a selection. TVrHyperLink is more like a normal label without internet transport facilities.

## **TVrHyperLink.Alignment**

[TVrHyperLink](#) [see also](#)

Controls the horizontal placement of the text within the HyperLink.

**property** Alignment: TAlignment;

### **Description**

Controls the horizontal placement of the text within the control. Set Alignment to specify how the text of the control is aligned within the ClientRect of the control.

## **TVrHyperLink.ColorEnter**

[TVrHyperLink](#) [see also](#)

Specifies the active background color of the control.

**property** ColorEnter: TColor;

### **Description**

ColorEnter becomes the active background color as soon as the mouse is moved within the controls boundaries.



## **TVrHyperLink.ColorLeave**

[TVrHyperLink](#) [see also](#)

Specifies the active background color of the control.

**property** ColorLeave: TColor;

### **Description**

ColorLeave becomes the active background color as soon as the mouse is moved outside the controls boundaries.

## **TVrHyperLink.FontEnter**

[TVrHyperLink](#) [see also](#)

Specifies the active text font of the control.

**property** FontEnter: TFont;

### **Description**

FontEnter becomes the active text font as soon as the mouse is moved within the controls boundaries.

## **TVrHyperLink.FontLeave**

[TVrHyperLink](#) [see also](#)

Specifies the active text font of the control.

**property** FontLeave: TFont;

### **Description**

FontLeave becomes the active text font as soon as the mouse is moved outside the controls boundaries.

## **TVrHyperLink.OnMouseEnter**

[TVrHyperLink](#) [see also](#)

Occurs when the mouse is moved within the controls boundaries.

**property** OnMouseEnter: TNotifyEvent;

### **Description**

Use the OnMouseEnter event handler to cause any special processing to occur when the mouse is moved within the controls boundaries.

## **TVrHyperLink.OnMouseLeave**

[TVrHyperLink](#) [see also](#)

Occurs when the mouse is moved outside the controls boundaries.

**property** OnMouseLeave: TNotifyEvent;

### **Description**

Use the OnMouseLeave event handler to cause any special processing to occur when the mouse is moved outside the controls boundaries.

## TVrHyperLink.TextOutline

[TVrHyperLink](#) [see also](#)

Is used to draw a outline pattern around the text.

```
type TVrTextOutline = class(TPersistent);  
property TextOutline: TVrTextOutline;
```

### Description

TVrTextOutline is derived from TPersistent and contains the properties which can be set through the designer in order to customize the output of the text.

## **TVrHyperLink.Transparent**

[TVrHyperLink](#) [see also](#)

Specifies whether controls that sit below the hyperlink on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is used to add text to a graphic, set Transparent to True so that the control does not stand out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the Hyperlink is not obscuring a complicated image, performance can be improved by setting the background color of the control to match the object beneath it and setting Transparent to False.

## **TVrHyperLink.WordWrap**

[TVrHyperLink](#) [see also](#)

Specifies whether the label text wraps when it is too long for the width of the label.

**property** WordWrap: Boolean;

### **Description**

Set WordWrap to True to allow the label to display multiple line of text. When WordWrap is True, text that is too wide for the label control wraps at the right margin and continues in additional lines.

Set WordWrap to False to limit the label to a single line. When WordWrap is False, text that is too wide for the label appears truncated.



## TVrImageLed

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrImageLed uses images to represent a state change.

### Unit

vrimageled

### Description

TVrImageLed is a standard led control, derived from TVrGraphicControl, which uses images to represent a state change. The image's used are not scalable. They are always automatically centered within the controls boundaries on a resize event.

## TVrImageLed.Active

[TVrImageLed](#) [see also](#)

Active describes the state of the control.

**property** Active: Boolean;

### Description

Set Active to True in order to activate the led control. When active the image appears highlighted depending on the palette property settings.

## **TVrImageLed.Blink**

[TVrImageLed](#) [see also](#)

Determines if the control will automatically change state.

**property** Blink: Boolean;

### **Description**

When Blink is enabled the control will automatically change state. The speed of each state change is defined by the TimeInterval property. Although blink can be set in the designer, it is only activated during runtime sessions.

## TVrImageLed.ImageType

[TVrImageLed](#) [see also](#)

Describes the type of image used.

```
type TVrImageType = (itSound, itCD, itPlug, itMike, itPlayMedia,  
itSpeaker,itNote, itPlayBack, itFrequency, itRecord, itRewind, itReplay);  
property ImageType: TVrImageType;
```

### Description

Each type represents a different graphical image which is used to create the led.

## TVImageLed.Inverted

[TVImageLed](#) [see also](#)

The background is highlighted in order to represent a state change.

**property** Inverted: Boolean;

### Description

When enabled the background is highlighted in order to represent a state change, instead of the image itself. Default set to false.

## **TVrImageLed.OnChange**

[TVrImageLed](#) [see also](#)

Event which is called whenever the Active property value changes.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called whenever the Active property value changes.

## TVrImageLed.Palette

[TVrImageLed](#) [see also](#)

Defines the color attributes for TVrImageLed.

**property** Palette: TVrPalette;

## TVrImageLed.TimeInterval

[TVrImageLed](#) [see also](#)

Determines the animation speed

**property** TimeInterval: Integer;

### Description

TimeInterval is used to define the blink speed. Decrease TimeInterval to speed up the animation.



## **TVrImageLed.Transparent**

[TVrImageLed](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## TVrIndicator

[properties](#)[methods](#)

[events](#)

[see also](#)

An indicator is a horizontal row of leds with three different areas.

### Unit

vrscanner

### Description

TVrIndicator is derived from a TVrLedGroup collection which is derived from TVrGraphicControl. With the properties min, max and position you can monitor processes like cd-playing time, communications, file transfers etc. An indicator is a horizontal row of leds with three different areas.

## **TVrIndicator.ColorWidth**

[TVrIndicator](#) [see also](#)

Describes the size of each color transition.

**property** ColorWidth: Integer;

### **Description**

ColorWidth can be used to speed up the painting process. Assigning a small value will make the gradient look smoother but will take a longer time to refresh on screen. Only applies when PlainColors is set to false.

## **TVrIndicator.LedStyle1**

[TVrIndicator](#) [see also](#)

**type** TVrLedStyle = class;

**property** LedStyle1: TVrLedStyle;

### **Description**

LedStyle1 contains several color attributes which make up the led image.

## TVrIndicator.LedStyle2

[TVrIndicator](#) [see also](#)

**type** TVrLedStyle = class;

**property** LedStyle2: TVrLedStyle;

### Description

LedStyle2 contains several color attributes which make up the led image.

## TVrIndicator.LedStyle3

[TVrIndicator](#) [see also](#)

**type** TVrLedStyle = class;

**property** LedStyle3: TVrLedStyle;

### Description

LedStyle3 contains several color attributes which make up the led image.

## **TVrIndicator.MaxValue**

[TVrIndicator](#) [see also](#)

Used to set a upper limit to the value that can be represented.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set a upper limit to the value that can be represented using the indicator. The highlighted area indicates the current position in a range between MinValue and MaxValue.

## **TVrIndicator.MinValue**

[TVrIndicator](#) [see also](#)

Used to set a lower limit to the value that can be represented.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the indicator. The highlighted area indicates the current position in a range between MinValue and MaxValue.



## TVrIndicator.OnChange

[TVrIndicator](#) [see also](#)

Occurs when the position value changes.

**property** OnChange: TNotifyEvent;

### Description

OnChange occurs when the position value changes.

## TVrIndicator.Percent1

[TVrIndicator](#) [see also](#)

Describes the number of leds used for the lower segment.

**property** Percent1: TVrPercentInt;

### Description

Percent1 and Percent2 make up the levels in TVrIndicator. Percent1 describes the number of leds used for the lower segment, the percentage of the bar which is filled with the lower segment color. Percent1 and Percent2 together can never exceed 100%.

## **TVrIndicator.Percent2**

[TVrIndicator](#) [see also](#)

Describes the number of leds used for the medium segment.

**property** Percent2: TVrPercentInt;

### **Description**

Percent1 and Percent2 make up the levels in TVrIndicator. Percent2 describes the number of leds which are filled with the medium segment colors. Percent1 and Percent2 together can never exceed 100%.

## **TVrIndicator.PercentDone**

[TVrIndicator](#) [see also](#)

Defines the current position in the range between min and max.

**property** PercentDone: Integer;

### **Description**

Percentage of the position value calculated in the range defined by the min and max properties. This is a runtime and read-only property.

## TVIndicator.Position

[TVIndicator](#) [see also](#)

Describes the current position or progress made.

**property** Position: Integer;

### Description

The position or progress made. This value can never exceed the value defined by the min and max properties. Position is used to calculate the number of active leds.

## **TVrIndicator.Spacing**

[TVrIndicator](#) [see also](#)

Increases or decreases the space between each individual led.

**property** Spacing: Integer;

### **Description**

Increases or decreases the space between each individual led.

## TVIndicator.Step

[TVIndicator](#) [see also](#)

Specifies the amount that Position increases when the StepIt method is called.

**property** Step: Integer;

### Description

Specifies the amount that Position increases when the StepIt method is called.

Set Step to specify the granularity of the indicator bar. Step should reflect the size of each step in the process tracked by the indicator bar, in the logical units used by the MaxValue and MinValue properties.

When a indicator control is created, MinValue and MaxValue represent percentages, where MinValue is 0 (0% complete) and MaxValue is 100 (100% complete). If these values are not changed, Step is the percentage of the process completed before the user is provided with additional visual feedback.

When the StepIt method is called, the value of Position increases by Step.

## **TVrIndicator.StepBy**

[TVrIndicator](#) [see also](#)

Advances the Position of the indicator by a specified amount.

**procedure** StepBy(Delta: Integer);

### **Description**

Advances the Position of the indicator by a specified amount. Call StepBy to increase the value of Position by the value of the Delta parameter. To advance Position by a default amount that represents a single step in the process, use the StepIt method.



## **TVIndicator.StepIt**

[TVIndicator](#) [see also](#)

Call the StepIt method to increase the value of Position by the value of the Step property.

```
procedure StepIt;
```

### **Description**

Advances Position by the amount specified in the Step property.

Call the StepIt method to increase the value of Position by the value of the Step property. If Step represents the size of one logical step in the process tracked by the indicator bar, call Step after each logical step is completed.

## **TVrIndicator.Transparent**

[TVrIndicator](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Transparent specifies whether the background of the image obscures objects below the image object. Set Transparent to True to allow objects behind the object to show through the background of the control. Set Transparent to False to make the background opaque.

## TVrJoypad

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrJoypad is a collection of arrows.

### Unit

vrjoypad

### Description

TVrJoypad is a collection of arrows, each pointing to a different direction, north, west, south, east.

TVrJoypad is derived from TVrGraphicControl. Each arrow can be switched on or off depending on the Direction property.

## TVrJoypad.Direction

[TVrJoypad](#)

[see also](#)

Determines the active or highlighted arrow.

```
type TVrJoypadDirection = (jdUp, jdDown, jdLeft, jdRight);
```

```
type TVrJoypadDirections = set of TVrJoypadDirection;
```

```
property Direction: TVrJoypadDirections;
```

### Description

Direction determines the active or highlighted arrows.

## **TVrJoypad.Palette**

[TVrJoypad](#)

[see also](#)

Defines the color attributes for TVrJoyPad.

**property** Palette: TVrPalette;

## TVrJoypad.Spacing

[TVrJoypad](#)

[see also](#)

Spacing defines the space between each arrow.

**property** Spacing: Integer;

### Description

Spacing defines the horizontal and vertical space between each arrow.

## TVrJoyPad.Transparent

[TVrJoypad](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### Description

Transparent specifies whether the background of the image obscures objects below the image object. Set Transparent to True to allow objects behind the object to show through the background of the control. Set Transparent to False to make the background opaque.

## TVrJoypad.VisibleArrows

[TVrJoypad](#)

[see also](#)

Hide or show one of the four arrows.

```
type TVrVisibleArrow = (vaUp, vaDown, vaLeft, vaRight);
```

```
type TVrVisibleArrows = set of TVrVisibleArrow;
```

```
property VisibleArrows: TVrVisibleArrows;
```

### Description

Hide or show one of the four arrows.



## **TVrKeyStatus**

[properties](#)[methods](#)

[events](#)

[see also](#)

Non visual key handler component.

### **Unit**

vrsystem

### **Description**

TVrKeyStatus handles all keyboard events for Num Lock, Caps Lock and Scroll Lock.

## TVrKeyStatus.Keys

[TVrKeyStatus](#) [see also](#)

Describes the active keys.

```
type TVrKeyStateType = (ksNUM, ksCAPS, ksSCROLL);
```

```
type TVrKeyStateTypes = set of TVrKeyStateType;
```

```
property Keys: TVrKeyStateTypes
```

### Description

Setting one of the three different key types will activate the keys. When MonitorEvents is set to True this property will change as soon as a keyboard event occurs.

## **TVrKeyStatus.MonitorEvents**

[TVrKeyStatus](#) [see also](#)

Enables or disabled monitoring of keyboard events.

**property** MonitorEvents: Boolean;

### **Description**

When MonitorEvents is set to True it will install a keyboard handler to see if a keyboard event occurs.

## **TVrKeyStatus.OnChange**

[TVrKeyStatus](#) [see also](#)

Occurs immediately after a keyboard event.

**property** OnChange: TNotifyEvent;

### **Description**

When MonitorEvents is set to True it will install a keyboard handler to see if a keyboard event occurs. If the state of Num Lock, Caps Lock or Scroll Lock is changed it will call OnChange.

## **TVrLabel**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrLabel is a nonwindowed control that displays 3d text on a form.

### **Unit**

vrlabel

### **Description**

TVrLabel is a nonwindowed control that displays 3d text on a form. TVrLabel can be used as a normal label component but it contains additional features in order to create some special effects. TVrLabel is derived from TVrGraphicControl.

## **TVrLabel.Alignment**

[TVrLabel](#)

[see also](#)

Controls the horizontal placement of the text within the label.

**property** Alignment: TAlignment;

### **Description**

Controls the horizontal placement of the text within the label. Set Alignment to specify how the text of the label is aligned within the ClientRect of the label control.

## **TVrLabel.Angle**

[TVrLabel](#) [see also](#)

Determines the angle of the text.

**type** TVrTextAngle = 0..359;

**property** Angle: Integer;

### **Description**

With Angle the text is rotated 0..359 degrees. Use 0 for plain horizontal text.

## **TVrLabel.AutoSize**

[TVrLabel](#) [see also](#)

Used to make the label adjust its size automatically so the client area accommodates the height and width of the text.

**property** AutoSize: Boolean;

### **Description**

Use AutoSize to make the label adjust its size automatically so the client area accommodates the height and width of the text. When AutoSize is False, the label is fixed in size. When AutoSize is True, the size of the label is readjusted whenever its text changes. The size of the label is also readjusted when the Font property changes.



## **TVrLabel.Bitmap**

[TVrLabel](#) [see also](#)

Paints a graphical image on the label text.

**property** Bitmap: TBitmap;

### **Description**

When a bitmap is assigned it is painted on top of the Label text in order to create some additional effects.

## **TVrLabel.ColorHighlight**

[TVrLabel](#)

[see also](#)

Is one of the two colors which make the 3d effect.

**property** ColorHighlight: TColor;

### **Description**

ColorHighlight is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: [ColorShadow](#) and ColorHighlight.

## **TVrLabel.ColorShadow**

[TVrLabel](#) [see also](#)

Is one of the two colors which make the 3d effect.

**property** ColorShadow: TColor;

### **Description**

ColorShadow is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ColorShadow and [ColorHighlight](#).

## TVrLabel.Layout

[TVrLabel](#) [see also](#)

Specifies the vertical placement of the text within the label.

```
type TTextLayout = (tlTop, tlCenter, tlBottom);
```

```
property Layout: TTextLayout;
```

### Description

Specifies the vertical placement of the text within the label. Set Layout to specify how the text of the label is placed within the ClientRect of the label control. Layout can be one of the following values:

<b>Value</b>	<b>Meaning</b>
tlTop	The text appears at the top of the label.
tlCenter	The text is vertically centered in the label.
tlBottom	The text appears along the bottom of the label.

## **TVrLabel.ShadowDepth**

[TVrLabel](#)

[see also](#)

Describes the offset from the original text position

**property** ShadowDepth: Integer;

### **Description**

ShadowDepth describes the offset from the original text position and is used to paint the shadow text area. ShadowDepth only applies when the [Style](#) property is set to IsShadow.

## TVrLabel.Style

[TVrLabel](#) [see also](#)

Describes the type of effect which is used to paint the text.

**type** TVrLabelStyle = (lsNone, lsRaised, lsLowered, lsShadow);

**property** Style: TVrLabelStyle;

### Description

Style describes the type of effect which is used to paint the text.

## **TVrLabel.Transparent**

[TVrLabel](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Specifies whether controls that sit below the label on a form can be seen through the label. Set Transparent to True to prevent the label from obscuring other controls on the form. For example, if the label is used to add text to a graphic, set Transparent to True so that the label does not stand out as a separate object.

Writing text so that the background is transparent is slower than writing text when Transparent is False. If the label is not obscuring a complicated image, performance can be improved by setting the background color of the label to match the object beneath it and setting Transparent to False.

## TVrLed

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrLed is a control which represents a small light bulb.

### Unit

vrleds

### Description

TVrLed is a control which represents a small light bulb. Use the active property to turn it on/off. The user can select three kinds of led styles: rounded or rectangle, large rect. TVrLed is derived from TCustomVrLed, which is a TVrGraphicControl. The image's used aren't scalable. They are always automatically centered within the controls boundaries after a resize event.



## **TVrLed.Active**

[TVrLed](#) [see also](#)

Used to toggle the state of the control.

**property** Active: Boolean;

### **Description**

Use the active property to toggle the state of the control (true/false).

## TVrLed.Glyphs

[TVrLed](#) [see also](#)

Glyphs is used to implement a custom led control.

**property** Glyphs: TBitmap;

### Description

Glyphs is used to implement a custom led control. Glyphs must contain at least two images for the OFF state and the ON state. To make the led images transparent set the TVrLed Color property to the bitmap transparent color. Make sure LedType is set to LtCustom.

## TVrLed.Layout

[TVrLed](#) [see also](#)

Determines where the image or text appears on the button.

**type** TVrImageTextLayout = (ImageLeft, ImageRight, ImageTop, ImageBottom);

**property** Layout: TVrImageTextLayout;

### Description

Value	Meaning
ImageLeft	The image or caption appears near the left side of the button.
ImageRight	The image or caption appears near the right side of the button.
ImageTop	The image or caption appears near the top of the button.
ImageBottom	The image or caption appears near the bottom of the button.

## **TVrLed.LedType**

[TVrLed](#) [see also](#)

Determines the type of graphical image to represent the led.

```
type TVrLedType = (ltRounded, ltRectangle, ltLargeRect, ltCustom);
```

```
property LedType: TVrLedType;
```

### **Description**

LedType determines the type of graphical image to represent the led.

## **TVrLed.Margin**

[TVrLed](#) [see also](#)

Use Margin to specify the indentation of the image or the text specified by the Caption property.

**property** Margin: Integer;

### **Description**

Use Margin to specify the indentation of the image or the text specified by the Caption property. The edges that Margin separates depends on the Layout property. If Layout is ImageLeft, the margin appears between the left edge of the image or caption and the left edge of the control. If Layout is ImageRight, the margin separates the right edges. If Layout is ImageTop, the margin separates the top edges, and if Layout is ImageBottom, the margin separates the bottom edges.

If Margin is -1 then the image or text are centered on the button.

## **TVrLed.OnChange**

[TVrLed](#) [see also](#)

This event is called when the state of the control has changed.

**property** OnChange: TNotifyEvent;

### **Description**

This event is called when the state of the control has changed. See the Active property for more information.

## **TVrLed.Palette**

[TVrLed](#) [see also](#)

Defines the color attributes for TVrLed.

**property** Palette: TVrPalette;

## **TVrLed.Spacing**

[TVrLed](#) [see also](#)

Defines the space between the Image and Caption text.

**property** Spacing: Integer;

### **Description**

Spacing defines the space between the Image and Caption text.



## **TVrLed.Transparent**

[TVrLed](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Transparent specifies whether the background of the image obscures objects below the image object. Set Transparent to True to allow objects behind the object to show through the background of the control. Set Transparent to False to make the background opaque.

## TVrLevelBar

[properties](#)[methods](#)

[events](#)

[see also](#)

Use TVrLevelBar to add a progress bar to a form.

### Unit

vrlevelbar

### Description

Use TVrLevelBar to add a progress bar to a form. Progress bars provide users with visual feedback about the progress of a procedure within an application. As the procedure progresses, the rectangular progress bar gradually fills from left to right with the defined palette colors.

## **TVrLevelBar.Bevel**

[TVrLevelBar](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrLevelBar.MaxValue**

[TVrLevelBar](#) [see also](#)

Used to set an upper limit to the value that can be represented using the LevelBar.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set an upper limit to the value that can be represented using the LevelBar. The position property can never exceed this value. A highlighted area indicates the current position in a range between MinValue and MaxValue.

## **TVrLevelBar.MinValue**

[TVrLevelBar](#) [see also](#)

Used to set a lower limit to the value that can be represented using the LevelBar.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the LevelBar. The [position](#) property can never be smaller than this value. A highlighted area indicates the current position in a range between MinValue and MaxValue.

## **TVrLevelBar.OnChange**

[TVrLevelBar](#) [see also](#)

Occurs when the position property is changed.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange occurs when the position property is changed.

## **TVrLevelBar.OnMaxValue**

[TVrLevelBar](#) [see also](#)

Occurs when Position reaches the value defined by MaxValue.

**property** OnMaxValue: TNotifyEvent;

### **Description**

OnMaxValue occurs when Position reaches the value defined by MaxValue.

## **TVrLevelBar.OnMinValue**

[TVrLevelBar](#) [see also](#)

Occurs when Position reaches the value defined by MinValue.

**property** OnMinValue: TNotifyEvent;

### **Description**

OnMinValue occurs when Position reaches the value defined by MinValue.



## **TVrLevelBar.Orientation**

[TVrLevelBar](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrLevelBar.Palette1**

[TVrLevelBar](#) [see also](#)

Color attributes for the lower bar area.

**property** Palette1: TVrPalette;

### **Description**

Used to fill part of the bar described by Percent1.

## **TVrLevelBar.Palette2**

[TVrLevelBar](#) [see also](#)

Color attributes for the medium bar area.

**property** Palette2: TVrPalette;

### **Description**

Used to fill part of the bar described by Percent2.

## **TVrLevelBar.Palette3**

[TVrLevelBar](#) [see also](#)

Color attributes for the higher bar area.

**property** Palette3: TVrPalette;

### **Description**

The Percent1 and Percent2 properties describe the lower and medium bar areas. The rest is filled with Palette3.

## TVrLevelBar.Percent1

[TVrLevelBar](#) [see also](#)

Describes the size of the bar used for the lower segment palette.

**property** Percent1: Integer;

### Description

Percent1 and Percent2 make up the levels in TVrLevelBar. Percent1 describes the percentage of the bar which is filled with the lower segment palette1 colors. Percent1 and Percent2 together can never exceed 100%.

## TVrLevelBar.Percent2

[TVrLevelBar](#) [see also](#)

Describes the size of the bar used for the medium segment palette.

**property** Percent2: Integer;

### Description

Percent1 and Percent2 make up the levels in TVrLevelBar. Percent2 describes the percentage of the bar which is filled with the medium segment palette2 colors. Percent1 and Percent2 together can never exceed 100%.

## **TVrLevelBar.Position**

[TVrLevelBar](#) [see also](#)

Points to the current state or progress made.

**property** Position: Integer;

### **Description**

Position points to the current state or progress made. The position can never exceed the values defined by the MinValue and MaxValue properties. A highlighted area indicates the current position between MinValue and MaxValue.

## **TVrLevelBar.Spacing**

[TVrLevelBar](#) [see also](#)

Defines the space between each single segment within the image with Spacing.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single step within the image with Spacing.



## **TVrLevelBar.Step**

[TVrLevelBar](#) [see also](#)

Specifies the amount that Position increases when the StepIt method is called.

**property** Step: Integer;

### **Description**

Specifies the amount that Position increases when the StepIt method is called.

Set Step to specify the granularity of the level bar. Step should reflect the size of each step in the process tracked by the level bar, in the logical units used by the MaxValue and MinValue properties.

When a level bar is created, MinValue and MaxValue represent percentages, where MinValue is 0 (0% complete) and MaxValue is 100 (100% complete). If these values are not changed, Step is the percentage of the process completed before the user is provided with additional visual feedback.

When the StepIt method is called, the value of Position increases by Step.

## **TVrLevelBar.StepBy**

[TVrLevelBar](#) [see also](#)

Advances the Position of the level bar by a specified amount.

**procedure** StepBy(Delta: Integer);

### **Description**

Advances the Position of the level bar by a specified amount. Call StepBy to increase the value of Position by the value of the Delta parameter. To advance Position by a default amount that represents a single step in the process, use the StepIt method.

## **TVrLevelBar.StepIt**

[TVrLevelBar](#) [see also](#)

Call the StepIt method to increase the value of Position by the value of the Step property.

**procedure** StepIt;

### **Description**

Advances Position by the amount specified in the Step property.

Call the StepIt method to increase the value of Position by the value of the Step property. If Step represents the size of one logical step in the process tracked by the level bar, call Step after each logical step is completed.

## **TVrLevelBar.Style**

[TVrLevelBar](#) [see also](#)

Determines the direction of the slider.

```
type TVrProgressStyle = (psBottomLeft, psTopRight);
```

```
property Style: TVrProgressStyle;
```

### **Description**

Position points to the current state or progress made. With psBottomLeft progress is displayed from Bottom-to-Top or from Left-to-Right (default). Use style to change the starting point of the display.

## **TVrLevelBar.TickHeight**

[TVrLevelBar](#) [see also](#)

Defines the height of each bar segment.

**property** TickHeight: Integer;

### **Description**

Each bar consists out of bar segments. Tickheight defines the height of these ticks.

## TVrLights

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrLights is a row of leds in the colors green/yellow/red.

### Unit

vrlights

### Description

TVrLights is a row of leds in the colors green/yellow/red. Each led can be activated or deactivated with the LedState property. It also introduces two new led styles, ItGlassRounded and ItGlassRect. TVrLights is derived from TGraphicControl.

## TVrLights.LedState

[TVrLights](#)

[see also](#)

Used to turn each led on or off.

```
type TVrLightsState = (lsGreen, lsYellow, lsRed);  
property LedState: TVrLightsStates;
```

### Description

With LedState you can turn each led on or off. While in designing mode just click on the +LedState property to configure each led.

Runtime example:

```
LedState := LedState + [lsRed];
```

This will turn the red colored led on.

## **TVrLights.LedsVisible**

[TVrLights](#)

[see also](#)

Determines if a led is visible.

**type** TVrLightsState = (lsGreen, lsYellow, lsRed);

**type** TVrLightsStates = set of TVrLightsState;

**property** LedsVisible: TVrLightsStates;

### **Description**

Hide or show a led.



## **TVrLights.LedType**

[TVrLights](#)

[see also](#)

Determines the type of graphical image to represent the led.

```
type TVrLightsType = (ltGlassRounded, ltGlassRect, ltGlassSquare,  
ltGlassDiamond);
```

```
property LedType: TVrLightsType;
```

### **Description**

LedType determines the type of graphical image to represent the led.

## **TVrLights.OnChange**

[TVrLights](#)

[see also](#)

This event is called when one of the leds changes it's state.

**property** OnChange: TNotifyEvent;

### **Description**

This event is called whenever one of the leds changes it's state.

## **TVrLights.Order**

[TVrLights](#) [see also](#)

Determines the order of each led.

```
type TVrLightsOrder = (loGreenToRed, loRedToGreen);  
property Order: TVrLightsOrder;
```

### **Description**

Order defines the order of each led: Red-to-Green or Green-to-Red

## **TVrLights.Orientation**

[TVrLights](#)

[see also](#)

Specifies whether the control is horizontal or vertical.

```
type TVrOrientation = (voVertical, voHorizontal);  
property Orientation: TVrOrientation;
```

## **TVrLights.Spacing**

[TVrLights](#)

[see also](#)

Defines the space between each single led.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single led.

## **TVrLights.Transparent**

[TVrLights](#)

[see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Transparent specifies whether the background of the image obscures objects below the image object. Set Transparent to True to allow objects behind the object to show through the background of the control. Set Transparent to False to make the background opaque.

## **TVrMatrix**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrMatrix is a LCD control to display text.

### **Unit**

vrmatrix

### **Description**

TVrMatrix is a dot - matrix control to display text. It is a nonwindowed control and is derived from TVrGraphicControl . TVrMatrix can be used as a normal text display component but it contains additional features in order to create some special effects.

## TVrMatrix.Alignment

[TVrMatrix](#) [see also](#)

Controls the horizontal placement of the text within the matrix.

**type** TVrAlignment = (vaLeftJustify, vaRightJustify, vaCenter);

**property** Alignment: TVrAlignment;

### Description

TVrAlignment is the type of the Alignment property. The following table lists the values of the TVrAlignment type:

Value	Meaning
vaLeftJustify	Align text to the left side of the control
vaCenter	Center text horizontally in the control
vaRightJustify	Align text to the right side of the control



## **TvrMatrix.AutoScroll**

[TvrMatrix](#)

[see also](#)

Enables or disables scrolling.

**property** AutoScroll: Boolean;

### **Description**

Set AutoScroll to True in order to start the scrolling of the text. AutoScroll is only activated during runtime sessions. The animation speed is defined by the [TimeInterval](#) property.

## **TVrMatrix.Bevel**

[TVrMatrix](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrMatrix.Leds**

[TVrMatrix](#) [see also](#)

Defines the number of matrix leds

**property** Leds: Integer;

### **Description**

Defines the number of matrix leds to be painted within the control. All text longer than the number of leds automatically gets truncated.

## **TVrMatrix.LedStyle**

[TVrMatrix](#)

[see also](#)

Determines the size of each led segment.

```
type TVrMatrixLedStyle = (1s9x13, 1s14x20, 1s19x27);
```

```
property LedStyle: TVrMatrixLedStyle;
```

### **Description**

LedStyle determines the size of each led segment where the numbers in the styles represent the horizontal and vertical size of each segment in pixels.

## **TVrMatrix.LedsVisible**

[TVrMatrix](#)

[see also](#)

Determines if the background led segments are visible.

**property** LedsVisible: Boolean;

### **Description**

Set LedsVisible to false in order to hide the led segments (dot matrix).

## **TVrMatrix.OnScrollDone**

[TVrMatrix](#) [see also](#)

Event which is triggered after each scroll loop.

**property** OnScrollDone: TNotifyEvent;

### **Description**

When the text is scrolled outside the view of the display it will call the OnScrollDone event. This also makes it possible to display several text strings without pause or breaks between each scroll loop.

## **TVrMatrix.Orientation**

[TVrMatrix](#)

[see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## TVrMatrix.Palette

[TVrMatrix](#) [see also](#)

Defines the color attributes for TVrMatrix.

**property** Palette: TVrPalette;

### Multi-color character support:

Each color code is made out of a percentage character and a combination of digits: for example "%CHello" will display all characters in Yellow. Combinations are also possible: %4H%0ello. This will only display the "H" in a different color.

% + Code:

1 = clBlack	A = clRed
2 = clMaroon	B = clLime
3 = clGreen	C = clYellow
4 = clOlive	D = clBlue
5 = clNavy	E = clFuchsia
6 = clPurple	F = clAqua
7 = clTeal	G = clWhite
8 = clGray	
9 = clSilver	

**Note:** in order to reset to the default palette color use "%0", or to display a % character use %%.



## TVrMatrix.ScrollDirection

[TVrMatrix](#)

[see also](#)

Defines the scroll direction of the text.

```
type TVrMatrixScrollDirection = (msdRightToLeft, msdLeftToRight);
```

```
property ScrollDirection: TVrMatrixScrollDirection;
```

### Description

ScrollDirection defines the scroll direction of the text. Text can scroll from Right-to-Left or from Left-to-Right. Only applies when [AutoScroll](#) is set to True.

## **TVrMatrix.ScrollText**

Will move the text in the direction defined by ScrollDirection.

**procedure** ScrollText;

### **Description**

Calling ScrollText will move the text in the direction defined by ScrollDirection. Instead of using the internal timer scrolltext can be called instead. Runtime only.

## **TVrMatrix.Spacing**

[TVrMatrix](#)

[see also](#)

Determines the space between each single led segment.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single led segment.

## TVrMatrix.Text

[TVrMatrix](#) [see also](#)

Specifies a text string that is displayed within the Matrix.

**property** Text: string;

### Description

Specifies a text string that is displayed within the Matrix, using scrolling and multi-color coded strings.

### Multi-color character support:

Each color code is made out of a percentage character and a combination of digits: for example "%CHello" will display all characters in Yellow. Combinations are also possible: %4H%0ello. This will only display the "H" in a different color.

% + Code:

1 = clBlack	A = clRed
2 = clMaroon	B = clLime
3 = clGreen	C = clYellow
4 = clOlive	D = clBlue
5 = clNavy	E = clFuchsia
6 = clPurple	F = clAqua
7 = clTeal	G = clWhite
8 = clGray	
9 = clSilver	

**Note:** in order to reset to the default palette color use "%0", or to display a % character use %%.

## TVrMatrix.TextStyle

[TVrMatrix](#) [see also](#)

Formats the text string.

```
type TVrMatrixTextStyle = (tsUpperCase, tsLowerCase, tsAsIs, tcProperCase);  
property TextStyle: TVrMatrixTextStyle;
```

### Description

Formats the text string.

Value	Meaning
tsUpperCase	All uppercase charaters
tsLowerCase	All lowercase charaters
tsAsIs	No formatting
tcProperCase	First character is in uppercase

## **TvrMatrix.Threaded**

[TvrMatrix](#)

[see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TvrMatrix.TimeInterval**

[TvrMatrix](#)

[see also](#)

Determines the scrolling speed of the text.

**property** TimeInterval: Integer;

### **Description**

Use TimeInterval to increase or decrease the scrolling speed of the text. Only applies when the AutoScroll property is set to true.

## **TVrMatrixGroup**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrMatrixGroup is a multi row LCD control to display text.

### **Unit**

vrmatrix

### **Description**

TVrMatrixGroup is a dot - matrix control to display text. It is a nonwindowed control and is derived from TVrGraphicControl . TVrMatrixGroup can be used as a normal text display component but it contains additional features in order to create some special effects.



## **TVrMatrixGroup.Alignment**

[TVrMatrixGroup](#) [see also](#)

Controls the horizontal placement of the text within the matrixgroup.

**type** TAlignment = (taLeftJustify, taRightJustify, taCenter)

**property** Alignment: TAlignment;

### **Description**

TAlignment is the type of the Alignment property. The following table lists the values of the TAlignment type:

<b>Value</b>	<b>Meaning</b>
taLeftJustify	Align text to the left side of the control
taCenter	Center text horizontally in the control
taRightJustify	Align text to the right side of the control

## **TVrMatrixGroup.AutoScroll**

[TVrMatrixGroup](#) [see also](#)

Enables or disables scrolling.

**property** AutoScroll: Boolean;

### **Description**

Set AutoScroll to True in order to start the scrolling of the text. AutoScroll is only activated during runtime sessions. The scroll speed is defined by the [TimeInterval](#) property.

## **TVrMatrixGroup.Bevel**

[TVrMatrixGroup](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrMatrixGroup.CharSpacing**

[TVrMatrixGroup](#) [see also](#)

Describes the free space between each led character.

**property** CharSpacing: Integer;

### **Description**

CharSpacing describes the free space between each led character.

## **TVrMatrixGroup.Cols**

[TVrMatrixGroup](#) [see also](#)

Describes the number of visible leds.

```
type TVrColInt = 1..MaxInt;
```

```
property Cols: TVrColInt;
```

### **Description**

Number of visible horizontal leds.

## TVrMatrixGroup.Lines

[TVrMatrixGroup](#) [see also](#)

Contains the text to display.

**property** Lines: TStrings;

### Description

Lines is a list of strings which contains the text to display in TVrMatrixGroup.

### Multi-color character support:

Each color code is made out of a percentage character and a combination of digits: for example "%CHello" will display all characters in Yellow. Combinations are also possible: %4H%0ello. This will only display the "H" in a different color.

% + Code:

1 = clBlack	A = clRed
2 = clMaroon	B = clLime
3 = clGreen	C = clYellow
4 = clOlive	D = clBlue
5 = clNavy	E = clFuchsia
6 = clPurple	F = clAqua
7 = clTeal	G = clWhite
8 = clGray	
9 = clSilver	

**Note:** in order to reset to the default palette color use "%0", or to display a % character use %%.

## **TVrMatrixGroup.LineSpacing**

[TVrMatrixGroup](#) [see also](#)

Describes the free space between each row.

**property** LineSpacing: Integer;

### **Description**

LineSpacing describes the free space between each row in the matrix.

## TVrMatrixGroup.Palette

[TVrMatrixGroup](#) [see also](#)

Defines the color attributes for TVrMatrixGroup.

**property** Palette: TVrPalette;

### Multi-color character support:

Each color code is made out of a percentage character and a combination of digits: for example "%CHello" will display all characters in Yellow. Combinations are also possible: %4H%0ello. This will only display the "H" in a different color.

% + Code:

1 = clBlack	A = clRed
2 = clMaroon	B = clLime
3 = clGreen	C = clYellow
4 = clOlive	D = clBlue
5 = clNavy	E = clFuchsia
6 = clPurple	F = clAqua
7 = clTeal	G = clWhite
8 = clGray	
9 = clSilver	

**Note:** in order to reset to the default palette color use "%0", or to display a % character use %%.



## **TvrMatrixGroup.PixelSize**

[TvrMatrixGroup](#) [see also](#)

Defines the size in pixels of each led segment.

**property** PixelSize: Integer;

### **Description**

Each segment is painted as a small square (1x1, 2x2). PixelSize defines the size in pixels of each led segment.

## **TVrMatrixGroup.PixelSpacing**

[TVrMatrixGroup](#) [see also](#)

Determines the number of pixels between each led pixel.

**property** PixelSpacing: Integer;

### **Description**

PixelSpacing determines the number of pixels between each led pixel. The visible area defined by spacing is filled with the normal background color of the control.

## **TVrMatrixGroup.Reset**

[TVrMatrixGroup](#) [see also](#)

Brings the control back in it's original state.

**procedure** Reset;

### **Description**

Reset terminates the scrolling process and brings the control back in it's original state.

## **TVrMatrixGroup.Rows**

[TVrMatrixGroup](#) [see also](#)

Describes the number of visible rows within the controls boundaries.

**type** TVrRowInt = 1..MaxInt;

**property** Rows: TVrRowInt;

### **Description**

Number of visible vertical leds.

## **TVrMatrixGroup.ScrollDirection**

[TVrMatrixGroup](#) [see also](#)

Describes the scrolling direction.

**type** TVrScrollDirection = (sdRightToLeft, sdLeftToRight, sdTopToBottom, sdBottomToTop);

**property** ScrollDirection: TVrScrollDirection;

## TVrMatrixGroup.TextStyle

[TVrMatrixGroup](#) [see also](#)

Formats the text string.

```
type TVrMatrixTextStyle = (tsUpperCase, tsLowerCase, tsAsIs, tcProperCase);
```

```
property TextStyle: TVrMatrixTextStyle;
```

### Description

Formats the text string.

Value	Meaning
tsUpperCase	All uppercase charaters
tsLowerCase	All lowercase charaters
tsAsIs	No formatting
tcProperCase	First character is in uppercase

## **TVrMatrixGroup.Threaded**

[TVrMatrixGroup](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the scroll sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrMatrixGroup.TimeInterval**

[TVrMatrixGroup](#) [see also](#)

Determines the scrolling speed of the text.

**property** TimeInterval: Integer;

### **Description**

Use TimeInterval to increase or decrease the scrolling speed of the text. Only applies when the AutoScroll property is set to true.



## **TVrMediaButton**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrMediaButton is a push button control with an 3d effect.

### **Unit**

vrnavigator

### **Description**

TVrMediaButton is a button control with an 3d effect. Use TVrMediaButton to put a standard Windows push button on a form. TVrMediaButton introduces several properties to control its behavior. Users choose button controls to initiate actions.

## **TVrMediaButton.BorderColor**

[TVrMediaButton](#) [see also](#)

Describes the attributes of a small rectangle painted around the client area.

**property** BorderColor: TColor;

### **Description**

BorderColor defines the color which is used to paint a small outline rectangle around the button.

## **TVrMediaButton.ButtonType**

[TVrMediaButton](#) [see also](#)

Defines the type of graphical image.

```
type TVrButtonType = (btPower, btPlay, btPause, btStop, btPrev, btBack,  
btStep, btNext, btRecord, btEject);  
property ButtonType: TVrButtonType;
```

### **Description**

ButtonType defines the type of graphical image which is painted onto the button.

## **TVrMediaButton.FocusColor**

[TVrMediaButton](#) [see also](#)

Color for the outline when the button becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the button receives focus, a small rectangle is painted around the button in the color defined by FocusColor.

## **TVrMeter**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrMeter is a meter control which contains a needle and a separate scale.

### **Unit**

vrmeter

### **Description**

TVrMeter is a meter control which contains a needle and a separate scale. It can be used to display different kind of signals by changing the position property.

## **TVrMeter.Angle**

[TVrMeter](#)

[see also](#)

Describes the starting angle of the scale and labels inside the controls boundaries.

**property** Angle: Integer;

### **Description**

Angle describes the starting angle of the scale and labels inside the controls boundaries.

## **TVrMeter.BackImage**

[TVrMeter](#)

[see also](#)

Is used as a background image.

**property** BackImage: TBitmap;

### **Description**

BackImage is used as a background image.

## **TVrMeter.Bevel**

[TVrMeter](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;



## **TVrMeter.LabelOffsetX**

[TVrMeter](#) [see also](#)

Describes the horizontal gap between the scale and the text labels.

**property** LabelOffsetX: Integer;

### **Description**

LabelOffsetX describes the horizontal gap between the scale and the text labels.

## **TVrMeter.LabelOffsetY**

[TVrMeter](#)

[see also](#)

Describes the vertical gap between the scale and the text labels.

**property** LabelOffsetY: Integer;

### **Description**

LabelOffsetY describes the vertical gap between the scale and the text labels.

## **TVrMeter.Labels**

[TVrMeter](#) [see also](#)

Defines the number of text labels displayed along the scale of the meter control.

**property** Labels: Integer;

### **Description**

Labels defines the number of text labels displayed along the scale of the meter control. All labels are automatically positioned.

## **TvrMeter.MaxValue**

[TvrMeter](#)

[see also](#)

Used to set an upper limit to the value that can be represented using the Meter.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set an upper limit to the value that can be represented using the Meter. The position property can never exceed this value.

## **TvrMeter.MinValue**

[TvrMeter](#) [see also](#)

Used to set a lower limit to the value that can be represented using the Meter.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the Meter. The position property can never be smaller than this value.

## **TVrMeter.NeedleColor**

[TVrMeter](#) [see also](#)

Used to change the color of the needle.

**property** NeedleColor: TColor;

### **Description**

NeedleColor is used to change the color attribute of the needle. The needle points to the current position within the scale.

## **TvrMeter.NeedleWidth**

[TvrMeter](#)

[see also](#)

Size in pixels of the needle.

**property** NeedleWidth: Integer;

### **Description**

Size in pixels of the needle.

## **TVrMeter.OnChange**

[TVrMeter](#)

[see also](#)

Occurs when the position property is changed.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange occurs when the position property is changed.



## **TVrMeter.Position**

[TVrMeter](#) [see also](#)

Points to the current state or progress made.

**property** Position: Integer;

### **Description**

Position points to the current state or progress made. The position can never exceed the values defined by the MinValue and MaxValue properties. The needle indicates the current position in a range between Min and Max.

## TVrMeter.Scale

[TVrMeter](#) [see also](#)

Defines the attributes of the scale.

```
property Scale: TVrMeterScale;
```

### Description

#### **Color1: TColor;**

The scale used by TVrMeter is build out of three areas, High, Medium and Low. Color1 defines the color used to paint the lower area of the scale.

#### **Color2: TColor;**

The scale used by TVrMeter is build out of three areas, High, Medium and Low. Color2 defines the color used to paint the medium area of the scale.

#### **Color3: TColor;**

The scale used by TVrMeter is build out of three areas, High, Medium and Low. Color3 defines the color used to paint the high area of the scale.

#### **Enlarge: Integer;**

Enlarge is used to paint the scale of TVrMeter. Each single dot within the scale can be upsized, defined by the HeightMax property. If Enlarge is set to Zero all dots are painted with a HeightMin size, in pixels.

#### **Percent1: Integer;**

Percent1 and Percent2 define the levels for the scale. Percent1 describes the percentage of the scale which is filled with the lower segment colors. Percent1 and Percent2 together can never exceed 100%.

#### **Percent2: Integer;**

Percent1 and Percent2 define the levels for the scale. Percent2 describes the percentage of the scale which is filled with the medium segment colors. Percent1 and Percent2 together can never exceed 100%.

#### **Ticks: Integer;**

The scale used by TVrMeter is build out of three areas, High, Medium and Low. Ticks defines the number of ticks of the scale.

#### **HeightMax: Integer;**

HeightMax describes the size in pixels of each enlarged scale position.

#### **HeightMin: Integer;**

HeightMin describes the default size in pixels of each scale position.

#### **Visible: Boolean;**

Hides or shows the scale.

## **TVrMeter.Spacing**

[TVrMeter](#)

[see also](#)

Describes the gap between the top of the control and the scale and labels.

**property** Spacing: Integer;

### **Description**

Spacing describes the gap between the top of the control and the scale and labels.

## TVrNavigator

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrNavigator consists out of a set of buttons.

### Unit

vrnavigator

### Description

TVrNavigator consists out of a set of buttons (Play, Stop, Eject, and so on) that can be used to control a multi media device such as a CD-ROM drive, a MIDI sequencer, or a VCR. The navigator component consists of multiple buttons. These buttons can be clicked with the mouse or selected with the space bar.

## **TVrNavigator.Bevel**

[TVrNavigator](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrNavigator.BorderColor**

[TVrNavigator](#) [see also](#)

Describes the attributes of a small rectangle painted around each button.

**property** BorderColor: TColor;

### **Description**

BorderColor defines the color which is used to paint a small outline rectangle around each button.

## **TVrNavigator.ButtonIndex**

[TVrNavigator](#) [see also](#)

Translates a buttontype to a numeric value from 1 to 10.

```
function ButtonIndex(Button: TVrButtonType): Integer;
```

### **Description**

ButtonIndex translates a buttontype to a numeric value from 1 to 10.

## **TvrNavigator.EnabledButtons**

[TvrNavigator](#) [see also](#)

Controls which buttons on the navigator are enabled and usable.

**type** TvrButtonSet = set of TvrButtonType;

**property** EnabledButtons: TvrButtonSet;

### **Description**

EnabledButtons controls which buttons on the navigator are enabled and usable.

An enabled button is colored and usable. A disabled button is dimmed and not usable. If a button is not enabled with EnabledButtons, it is disabled. By default, all buttons are enabled.



## **TVrNavigator.FocusColor**

[TVrNavigator](#) [see also](#)

Color for the outline when a button becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the navigator receives focus, a small rectangle is painted around the active button in the color defined by FocusColor.

## **TVrNavigator.Numeric**

[TVrNavigator](#) [see also](#)

Display numeric images on the navigator buttons.

**property** Numeric: Boolean;

### **Description**

Set numeric to True in order to display numeric image instead of the play, pause etc images.

## TVrNavigator.OnButtonClick

[TVrNavigator](#) [see also](#)

Event handler specifying the button which was pressed

```
type TVrButtonClickEvent = procedure(Sender: TObject; ButtonType: TVrButtonType) of object;  
property OnButtonClick: TVrButtonClickEvent;
```

### Description

When a button on the navigator is pressed this event handler is called specifying the button which was pressed. All types are defined in the ButtonType parameter.

```
TVrButtonType = (btPower, btPlay, btPause, btStop, btPrev, btBack, btStep, btNext, btRecord, btEject);
```

```
procedure TForm1.VrNavigator1ButtonClick(Sender: TObject;  
    Button: TVrButtonType);  
begin  
    if Button = btPlay then MediaPlayer1.Play;  
end;
```

## TVrNavigator.Orientation

[TVrNavigator](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrNavigator.Spacing**

[TVrNavigator](#) [see also](#)

Defines the space between each individual button.

**property** Spacing: Integer;

### **Description**

Increases or decreases the space between each individual button.

## **TVrNavigator.VisibleButtons**

[TVrNavigator](#) [see also](#)

Determines the buttons on the navigator that are visible.

**type** TVrButtonSet = set of [TVrButtonType](#);

**property** VisibleButtons: TVrButtonSet;

### **Description**

VisibleButtons determines the buttons on the navigator that are visible. If a button is not made visible with VisibleButtons, it does not appear on the navigator control. By default, all buttons are visible when a navigator component is added to a form.

## TVrNum

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrNum is used to display numeric values in LCD style.

### Unit

vrlcd

### Description

TVrNum is derived from TGraphicControl. TVrNum is used to display numeric values in LCD style. There are several properties available to customize the control. TVrNum currently supports different led styles. Use the digits property to specify the number of visible leds. Negative values are also allowed. Use the min and max properties to restrict the range of allowed values. The minus sign only becomes visible when negative values are enabled by setting the min property below zero.

## TVrNum.Alignment

[TVrNum](#) [see also](#)

Controls the horizontal placement of the leds within VrNum.

```
type TVrNumAlignment = (naLeftJustify, naCenter, naRightJustify);  
property Alignment: TVrNumAlignment;
```

### Description

TVrNumAlignment is the type of the Alignment property. The following table lists the values of the TVrNumAlignment type:

Value	Meaning
naLeftJustify	Align leds to the left side of the control
naCenter	Center leds horizontally in the control
naRightJustify	Align leds to the right side of the control

**Note:** Only applies when AutoSize is False.



## **TVrNum.AutoSize**

[TVrNum](#) [see also](#)

Determines if the control automatically changes it's boundaries to fit all digits.

**property** AutoSize: Boolean;

### **Description**

When AutoSize is true the control automatically changes it's boundaries to fit all digits. AutoSize only applies when Align is set to alNone.

## **TVrNum.Digits**

[TVrNum](#) [see also](#)

Used to increase or decrease the number of visible leds/digits.

**property** Digits: Integer;

### **Description**

Use digits to increase or decrease the number of visible leds/digits. When AutoSize is true the digits property is also used to calculate the new size of the control.

## **TVrNum.LeadngZero**

[TVrNum](#) [see also](#)

Determines if leds are automatically activated when zero.

**property** LeadingZero: Boolean;

### **Description**

When leadingzero is enabled all digits are automatically activated when zero otherwise all leds stay dimmed.

## **TVrNum.Max**

[TVrNum](#) [see also](#)

Defines the upper limit in the range of valid values.

**property** Max: Integer;

### **Description**

Max is the upper limit in the range of valid values. All values assigned above Max are ignored.

## **TVrNum.Min**

[TVrNum](#) [see also](#)

Defines the lower limit in the range of valid values.

**property** Min: Integer;

### **Description**

Min is the lower limit in the range of valid values. All values assigned below Min are ignored.

## **TVrNum.OnChange**

[TVrNum](#)

[see also](#)

Events which is called when the Value property is modified.

**property** OnChange: TNotifyEvent;

### **Description**

The OnChange event is triggered when the value property is modified.

## **TVrNum.Palette**

[TVrNum](#) [see also](#)

Defines the color attributes for TVrNum.

**property** Palette: TVrPalette;

## **TVrNum.Spacing**

[TVrNum](#) [see also](#)

Defines the space between each single digit.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single digit.



## **TVrNum.Style**

[TVrNum](#) [see also](#)

Describes the size of the digits.

**type** TVrNumStyle = (ns13x24, ns11x20, ns7x13, ns12x17, ns5x7);

**property** Style: TVrNumStyle;

## **TVrNum.Transparent**

[TVrNum](#)

[see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TvrNum.Value**

[TvrNum](#) [see also](#)

Contains the actual integer value which is being highlighted.

**property** Value: Integer;

### **Description**

The Value property contains the actual integer value which is being highlighted. Value can never exceed the limits defined by the min and max properties.

## **TVrNum.ZeroBlank**

[TVrNum](#) [see also](#)

Determines if the value 0 is displayed as a single digit.

**property** ZeroBlank: Boolean;

### **Description**

When ZeroBlank is true, the value 0 is displayed as a single digit otherwise all digits will be set to their off position.

## TVrNumEdit

[properties](#)[methods](#)

[events](#)

[see also](#)

A wrapper for a Windows single-line edit control.

### Unit

vredit

### Description

Use a TVrNumEdit object to put a standard Windows edit control on a form. Edit controls are used to retrieve text that users type. Edit controls can also display text to the user.

TVrNumEdit implements the generic behavior introduced in TCustomEdit. TVrNumEdit publishes many of the properties inherited from TCustomEdit and introduce new behavior. TVrNumEdit only accept numeric values as input!

## **TVrNumEdit.Alignment**

[TVrNumEdit](#) [see also](#)

Specifies how text is aligned within the edit control.

**property** Alignment: TAlignment;

### **Description**

Use Alignment to specify whether the text should be left-justified, right-justified, or centered.

### **taLeftJustify**

Align text on the left side of the edit control.

### **taCenter**

Center text in the edit control.

### **taRightJustify**

Align text on the right side of the edit control.

## **TVrNumEdit.Decimals**

[TVrNumEdit](#) [see also](#)

Defines the number of decimals allowed.

**property** Decimals: Integer;

### **Description**

Decimals defines the number of decimals allowed.

## **TVrNumEdit.MaxValue**

[TVrNumEdit](#) [see also](#)

Defines the upper limit in the range of valid values.

**property** MaxValue: Double;

### **Description**

MaxValue is the upper limit in the range of valid values. All values assigned above MaxValue raise an OnError exception.



## **TVrNumEdit.MinValue**

[TVrNumEdit](#) [see also](#)

Defines the lower limit in the range of valid values.

**property** MinValue: Double;

### **Description**

MinValue is the lower limit in the range of valid values. All values assigned below MinValue raise an OnError exception.

## **TvrNumEdit.OnError**

[TvrNumEdit](#) [see also](#)

Input error event.

```
type TaErrorEvent = procedure(Sender: TObject; var CanExit: Boolean) of  
object;  
property OnError: TaErrorEvent;
```

### **Description**

Event which is triggered if the value entered exceeds the limits defined by MinValue and MaxValue or when an invalid value is specified.

## **TVrNumEdit.OnMouseEnter**

[TVrNumEdit](#) [see also](#)

Occurs when the mouse is moved within the controls boundaries.

**property** OnMouseEnter: TNotifyEvent;

### **Description**

Use the OnMouseEnter event handler to cause any special processing to occur when the mouse is moved within the controls boundaries.

## **TVrNumEdit.OnMouseLeave**

[TVrNumEdit](#) [see also](#)

Occurs when the mouse is moved outside the controls boundaries.

**property** OnMouseLeave: TNotifyEvent;

### **Description**

Use the OnMouseLeave event handler to cause any special processing to occur when the mouse is moved outside the controls boundaries.

## **TVrNumEdit.RestoreOnEsc**

[TVrNumEdit](#) [see also](#)

Allow to retrieve the previous value with ESC.

**property** RestoreOnEsc: Boolean;

### **Description**

Set RestoreOnEsc to true in order to use the ESC key to retrieve the previous value before the user entered a new value.

## **TVrNumEdit.Value**

[TVrNumEdit](#) [see also](#)

Contains the value entered by the user.

**property** Value: Double;

### **Description**

Value contains the value entered by the user.

## **TVrPalette**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrPalette is used to define 2 types of colors which indicate the state of an led control.

### **Unit**

vrclasses

### **Description**

TVrPalette is derived from TPersistent. TVrPalette is used to define 2 types of colors which indicate the state of a led control (Active = True). It has it's own property editor to set the high and low color members. Six custom palettes are already defined for easy selection.

## **TVrPalette.High**

[TVrPalette](#) [see also](#)

One of the colors which make up the palette.

**property** High: TColor;



## **TVrPalette.Low**

[TVrPalette](#) [see also](#)

One of the colors which make up the palette.

**property** Low: TColor;

## TVrPercentBar

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrPercentBar is a simple bar graph component.

### Unit

vrgraphs

### Description

TVrPercentBar is a simple bar graph component using a variable number of segments each with its own color settings.

## **TVrPercentBar.OnChange**

[TVrPercentBar](#) [see also](#)

Is called when one of the segment values is changed.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called when one of the segment values is changed. See also [Segments](#).

## **TVrPercentBar.Orientation**

[TVrPercentBar](#) [see also](#)

Specifies whether the control is horizontal or vertical.

```
type TVrOrientation = (voVertical, voHorizontal);  
property Orientation: TVrOrientation;
```

## **TVrPercentBar.OutlineColor**

[TVrPercentBar](#) [see also](#)

Describes the color attribute surrounding each segment.

**property** OutlineColor: TColor;

### **Description**

OutlineColor describes the color attribute surrounding each segment.

## TVrPercentBar.Segments

[TVrPercentBar](#) [see also](#)

```
type TVrPercentGraphItem = class;  
type TVrPercentGraphItems = class;  
property Segments: TVrPercentGraphItems;
```

### Description

property Color: TColor;  
The actual fill color for the current segment.

property Value: Double;  
The actual value for the current segment. Each segment has a value. The total used to paint the bar is calculated by adding all values from all segments.

property Pattern: TVrBrushPattern;  
Fill style of current segment.

TVrBrushPattern = (bpSolid, bpClear, bpHorizontal, bpVertical, bpFDiagonal, bpBDiagonal, bpCross, bpDiagCross);

## TVrPercentPie

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrPercentPie is a simple pie graph component.

### Unit

vrgraphs

### Description

TVrPercentPie is a simple pie graph component using a variable number of segments each with its own color settings.

## **TVrPercentPie.OnChange**

[TVrPercentPie](#) [see also](#)

Is called when one of the segment values is changed.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called when one of the segment values is changed. See also [Segments](#).



## **TVrPercentPie.OutlineColor**

[TVrPercentPie](#) [see also](#)

Describes the color attribute surrounding each segment.

**property** OutlineColor: TColor;

### **Description**

OutlineColor describes the color attribute surrounding each segment.

## TVrPercentPie.Segments

[TVrPercentPie](#) [see also](#)

```
type TVrPercentGraphItem = class;  
type TVrPercentGraphItems = class;  
property Segments: TVrPercentGraphItems;
```

### Description

property Color: TColor;  
The actual fill color for the current segment.

property Value: Double;  
The actual value for the current segment. Each segment has a value. The total used to paint the pie is calculated by adding all values from all segments.

property Pattern: TVrBrushPattern;  
Fill style of current segment.

TVrBrushPattern = (bpSolid, bpClear, bpHorizontal, bpVertical, bpFDiagonal, bpBDiagonal, bpCross, bpDiagCross);

## **TVrPercentPie.Transparent**

[TVrPercentPie](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## TVrPieGraph

[properties](#)[methods](#)

[events](#)

[see also](#)

Represents a windows like percentage pie graph.

### Unit

vrrocker

### Description

TVrPieGraph is derived from TVrGraphicControl and represents a windows like percentage pie graphic. It uses two areas to show a state or progress "Free and Used".

## **TVrPieGraph.Depth**

[TVrPieGraph](#) [see also](#)

Describes the height of the piegrah

**property** Depth: Integer;

### **Description**

Depth describes the height of the piegrah (3d effect).

## **TVrPieGraph.FreeColor**

[TVrPieGraph](#) [see also](#)

Is used to fill the segment described by the MaxValue property.

**property** FreeColor: TColor;

### **Description**

The piegraph is build out of two segments Used and Free. FreeColor is used to fill the segment described by the MaxValue property minus Value.

## **TvrPieGraph.FreeShadowColor**

[TvrPieGraph](#) [see also](#)

Is used to fill the segment described by the MaxValue property. (Border)

**property** FreeShadowColor: TColor;

### **Description**

The piegraph is build out of two segments Used and Free. FreeColor is used to fill the segment described by the MaxValue property minus Value (border of the piegraph).

## **TVrPieGraph.MaxValue**

[TVrPieGraph](#) [see also](#)

Determines the maximum value which can be assigned to the Value property.

**property** MaxValue: Integer;

### **Description**

MaxValues determines the maximum value which can be assigned to the Value property. Make sure to set MaxValue high enough.



## **TVrPieGraph.OutlineColor**

[TVrPieGraph](#) [see also](#)

Describes the outline color of each pie segment.

**property** OutlineColor: TColor;

### **Description**

OutlineColor describes the outline color of each pie segment.

## **TVrPieGraph.Transparent**

[TVrPieGraph](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrPieGraph.UsedColor**

[TVrPieGraph](#) [see also](#)

Is used to fill the segment described by the Value property.

**property** UsedColor: TColor;

### **Description**

The piegraph is build out of two segments Used and Free. UsedColor is used to fill the segment described by the Value property.

## **TVrPieGraph.UsedShadowColor**

[TVrPieGraph](#) [see also](#)

Is used to fill the segment described by the Value property (border)

**property** UsedShadowColor: TColor;

### **Description**

The piegraph is build out of two segments Used and Free. UsedShadowColor is used to fill the segment described by the Value property (border of the piegraph).

## **TVrPieGraph.Value**

[TVrPieGraph](#) [see also](#)

Describes the used pie segment.

**property** Value: Integer;

### **Description**

Value describes the used pie segment. The piegraph will show Value against the total pie which is described by MaxValue.

## TVrPowerButton

[properties](#)[methods](#)

[events](#)

[see also](#)

A standard windows button control which contains a small led image.

### Unit

vrpowerbutton

### Description

TVrPowerButton is derived from TVrCustomControl and represents a standard windows button control. Only TVrPowerButton contains a small led image which will be toggled on and off while pressed.

## **TVrPowerButton.Active**

[TVrPowerButton](#) [see also](#)

Describes the current state of the powerbutton.

**property** Active: Boolean;

### **Description**

Active describes the current state of the powerbutton.

## **TVrPowerButton.BevelWidth**

[TVrPowerButton](#) [see also](#)

Describes the size of the 3d border of the button.

```
type TVrByteInt = 0..255;  
property BevelWidth: TVrByteInt;
```

### **Description**

Describes the size of the 3d border of the button.



## **TVrPowerButton.FocusColor**

[TVrPowerButton](#) [see also](#)

Color for the outline when the PowerButton becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the PowerButton receives focus, a small rectangle is painted around the button with the color defined by FocusColor.

## **TVrPowerButton.HighlightColor**

[TVrPowerButton](#) [see also](#)

HighlightColor is one of the two colors which make the 3d effect.

**property** HighlightColor: TColor;

### **Description**

HighlightColor is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ShadowColor and HighlightColor.

## TVrPowerButton.Layout

[TVrPowerButton](#) [see also](#)

Determines where the image or text appears on the button.

**type** TVrImageTextLayout = (ImageLeft, ImageRight, ImageTop, ImageBottom);

**property** Layout: TVrImageTextLayout;

### Description

Value	Meaning
ImageLeft	The image or caption appears near the left side of the button.
ImageRight	The image or caption appears near the right side of the button.
ImageTop	The image or caption appears near the top of the button.
ImageBottom	The image or caption appears near the bottom of the button.

## **TVrPowerButton.LedHeight**

[TVrPowerButton](#) [see also](#)

Use LedHeight to increase or decrease the size of the led image.

**property** LedHeight: TVrByteInt;

### **Description**

TVrPowerButton uses a led shaped image to change state. This occurs when a user clicks the button with a mouse. Use LedHeight to increase or decrease the size of the led image.

## **TVrPowerButton.LedWidth**

[TVrPowerButton](#) [see also](#)

Use LedWidth to increase or decrease the size of the led image.

**property** LedWidth: TVrByteInt;

### **Description**

TVrPowerButton uses a led shaped image to change state. This occurs when a user clicks the image with a mouse. Use LedWidth to increase or decrease the size of the led image.

## **TVrPowerButton.Margin**

[TVrPowerButton](#) [see also](#)

Use Margin to specify the indentation of the image or the text specified by the Caption property.

**property** Margin: Integer;

### **Description**

Use Margin to specify the indentation of the image or the text specified by the Caption property. The edges that Margin separates depends on the Layout property. If Layout is ImageLeft, the margin appears between the left edge of the image or caption and the left edge of the control. If Layout is ImageRight, the margin separates the right edges. If Layout is ImageTop, the margin separates the top edges, and if Layout is ImageBottom, the margin separates the bottom edges.

If Margin is -1 then the image or text are centered on the button.

## **TVrPowerButton.OutlineColor**

[TVrPowerButton](#) [see also](#)

Describes the attributes of a small rectangle painted around the button image.

**property** OutlineColor: TColor;

### **Description**

OutlineColor defines the color which is used to paint a small outline rectangle around the button image.

## **TVrPowerButton.OutlineWidth**

[TVrPowerButton](#) [see also](#)

Describes the size in pixels of the outline which is painted as a small rectangle.

**property** OutlineWidth: TVrByteInt;

### **Description**

OutlineWidth describes the size in pixels of the outline which is painted as a small rectangle.



## **TVrPowerButton.Palette**

[TVrPowerButton](#) [see also](#)

Defines the color attributes for the led of TVrPowerButton.

**property** Palette: TVrPalette;

## **TVrPowerButton.ShadowColor**

[TVrPowerButton](#) [see also](#)

ShadowColor is one of the two colors which make the 3d effect.

**property** ShadowColor: TColor;

### **Description**

ShadowColor is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ShadowColor and HighlightColor.

## **TVrPowerButton.Spacing**

[TVrPowerButton](#) [see also](#)

Defines the space between the Image and Caption text.

**property** Spacing: Integer;

### **Description**

Defines the space between the Image and Caption text.

## TVrPowerMeter

[properties](#)[methods](#) [events](#) [see also](#)

Represents a temperature/progress like meter.

### Unit

vrpowermeter

### Description

TVrPowerMeter is derived from TVrGraphicControl and represents a temperature/progress like meter. Instead of Integer control values, float values are used, type of Double. This way precise measurements are possible.

## **TVrPowerMeter.BarColor**

[TVrPowerMeter](#) [see also](#)

Describes the color attribute used to paint the bar image.

**property** BarColor: TColor;

### **Description**

BarColor describes the color attribute used to paint the bar image.

## **TVrPowerMeter.BarWidth**

[TVrPowerMeter](#) [see also](#)

Defines the width of the bar.

**property** BarWidth: Integer;

### **Description**

Use BarWidth to define the width of the bar.

## **TVrPowerMeter.BottomOffset**

[TVrPowerMeter](#) [see also](#)

Defines the space between the bar image and the bottom of the control.

**property** BottomOffset: Integer;

### **Description**

Use BottomOffset to define the space between the bar image and the bottom of the controls boundaries.

## **TVrPowerMeter.ColorHighlight**

[TVrPowerMeter](#) [see also](#)

ColorHighlight is one of the two colors which make the 3d effect.

**property** ColorHighlight: TColor;

### **Description**

ColorHighlight is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ColorShadow and ColorHighlight.



## **TVrPowerMeter.ColorShadow**

[TVrPowerMeter](#) [see also](#)

ColorShadow is one of the two colors which make the 3d effect.

**property** ColorShadow: TColor;

### **Description**

ColorShadow is one of the two colors which make the 3d effect. This 3d effect is composed of two colors: ColorShadow and ColorHighlight.

## **TVrPowerMeter.Decimals**

[TVrPowerMeter](#) [see also](#)

Describes the actual number of decimals displayed.

**property** Decimals: Integer;

### **Description**

Decimals describes the actual number of decimals displayed for all floating point values within the control.

## **TVrPowerMeter.FillColor**

[TVrPowerMeter](#) [see also](#)

Is used to show the area pointed out by the Value property.

**property** FillColor: TColor;

### **Description**

FillColor is used to show the area pointed out by the Value property.

## **TVrPowerMeter.Increment**

[TVrPowerMeter](#) [see also](#)

Is used to paint the scale.

**property** Increment: Double;

### **Description**

Increment is used to paint the scale. It defines the values between MinValue and MaxValue which are displayed.

## **TVrPowerMeter.MarkerImage**

[TVrPowerMeter](#) [see also](#)

Contains the actual bitmap used as a marker of current position.

**property** MarkerImage: TBitmap;

### **Description**

MarkerImage contains the actual bitmap used as a marker of current position.

## **TVrPowerMeter.MarkerVisible**

[TVrPowerMeter](#) [see also](#)

Determines if the marker defined by MarkerImage is visible.

**property** MarkerVisible: Boolean;

### **Description**

MarkerVisible determines if the marker defined by MarkerImage is visible.

## **TVrPowerMeter.MaxValue**

[TVrPowerMeter](#) [see also](#)

Use MaxValue to set a upper limit to the value that can be represented using the meter.

**property** MaxValue: Double;

### **Description**

Use MaxValue to set a upper limit to the value that can be represented using the meter. The highlighted area indicates the current position in a range between MinValue and MaxValue.

## **TVrPowerMeter.MinValue**

[TVrPowerMeter](#) [see also](#)

Use MinValue to set a lower limit to the value that can be represented using the meter.

**property** MinValue: Double;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the meter. The highlighted area indicates the current position in a range between MinValue and MaxValue.



## **TVrPowerMeter.OnChange**

[TVrPowerMeter](#) [see also](#)

Occurs when the position value changes.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange occurs when the position value changes.

## **TVrPowerMeter.ScaleVisible**

[TVrPowerMeter](#) [see also](#)

Determines if the scale is visible.

**property** ScaleVisible: Boolean;

### **Description**

ScaleVisible determines if the scale is visible.

## **TVrPowerMeter.TitleAlignment**

[TVrPowerMeter](#) [see also](#)

**type** TVrTitleAlignment = (tiaTop, tiaBottom);

**property** TitleAlignment: TVrTitleAlignment;

## **TVrPowerMeter.TitleFont**

[TVrPowerMeter](#) [see also](#)

Font style used while displaying the components title.

**property** TitleFont: TFont;

### **Description**

See also TFont. Font style used while displaying the components title.

## **TVrPowerMeter.TopOffset**

[TVrPowerMeter](#) [see also](#)

Defines the space between the bar image and the top of the control.

**property** TopOffset: Integer;

### **Description**

Use TopOffset to define the space between the bar image and the top of the controls boundaries.

## **TVrPowerMeter.Transparent**

[TVrPowerMeter](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## **TVrPowerMeter.UnitsText**

[TVrPowerMeter](#) [see also](#)

Describes the type of entity to measure.

**property** UnitsText: string;

### **Description**

Use UnitText to describes the type of entity to measure.

## **TVrPowerMeter.Value**

[TVrPowerMeter](#) [see also](#)

Points to the current state or progress made.

**property** Value: Double;

### **Description**

Position points to the current state or progress made. The "Value" can never exceed the values defined by the MinValue and MaxValue properties.



## **TVrProgressBar**

[properties](#)[methods](#)

[events](#)

[see also](#)

Use TVrProgressBar to add a progress bar to a form.

### **Unit**

vrprogressbar

### **Description**

Use TVrProgressBar to add a progress bar to a form. Progress bars provide users with visual feedback about the progress of a procedure within an application. As the procedure progresses, the rectangular progress bar gradually fills from left to right with the system highlight color or a gradient fill.

## **TVrProgressBar.Bevel**

[TVrProgressBar](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrProgressBar.Bitmap**

[TVrProgressBar](#) [see also](#)

Used to fill the bar image when style is set to pfBitmap.

**property** Bitmap: TBitmap;

### **Description**

Bitmap is used to fill the bar image when style is set to pfBitmap.

## **TVrProgressBar.EndColor**

[TVrProgressBar](#) [see also](#)

Is one of the two colors used to create the gradient fill.

**property** EndColor: TColor;

### **Description**

The EndColor property is one of the two colors used to create the gradient fill. The gradient fill is composed of two colors: [StartColor](#) and EndColor.

## TVrProgressBar.FillType

[TVrProgressBar](#) [see also](#)

Determines the fillstyle.

```
type TVrProgressBarFill = (pfPlain, pfGradient, pfBitmap);  
property FillType: TVrProgressBarFill;
```

### Description

FillType defines the style used to fill the progressbar.

pfPlain, uses normal system colors

pfGradient, uses gradient fill defined by startcolor and endcolor

pfBitmap, uses a bitmap image (tiled).

Note: pfPlain is used when FillType is set to pfBitmap and no bitmap is defined.

## **TVrProgressBar.MaxValue**

[TVrProgressBar](#) [see also](#)

Defines the upper limit of the value that can be represented using the ProgressBar.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set an upper limit to the value that can be represented using the ProgressBar. The position property can never exceed this value.

## **TVrProgressBar.MinValue**

[TVrProgressBar](#) [see also](#)

Defines a lower limit of the value that can be represented using the ProgressBar.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the ProgressBar. The position property can never be smaller than this value.

## **TVrProgressBar.Onchange**

[TVrProgressBar](#) [see also](#)

Occurs when the position value changes.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange occurs when the position value changes.



## **TVrProgressBar.Orientation**

[TVrProgressBar](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrProgressBar.PercentDone**

[TVrProgressBar](#) [see also](#)

Defines the current position in the range between min and max.

**property** PercentDone: Integer;

### **Description**

Percentage of the position value calculated in the range defined by the min and max properties. This is a runtime and read-only property.

## **TVrProgressBar.Position**

[TVrProgressBar](#) [see also](#)

Points to the current state or progress made.

**property** Position: Integer;

### **Description**

Position points to the current state or progress made. The position can never exceed the values defined by the Min and Max properties. The colored area indicates the current position between Min and Max.

## **TVrProgressBar.Smooth**

[TVrProgressBar](#) [see also](#)

Determines whether the progress bar is smooth or segmented.

**property** Smooth: Boolean;

### **Description**

Use Smooth to specify whether the progress bar is smooth or segmented.

## **TVrProgressBar.StartColor**

[TVrProgressBar](#) [see also](#)

Is one of the two colors used to create the gradient fill.

**property** StartColor: TColor;

### **Description**

The StartColor property is one of the two colors used to create the gradient fill. The gradient fill is composed of two colors: StartColor and [EndColor](#).

## **TVrProgressBar.Step**

[TVrProgressBar](#) [see also](#)

Specifies the amount that Position increases when the StepIt method is called.

**property** Step: Integer;

### **Description**

Specifies the amount that Position increases when the StepIt method is called.

Set Step to specify the granularity of the progress bar. Step should reflect the size of each step in the process tracked by the progress bar, in the logical units used by the Max and Min properties.

When a progress bar is created, Min and Max represent percentages, where Min is 0 (0% complete) and Max is 100 (100% complete). If these values are not changed, Step is the percentage of the process completed before the user is provided with additional visual feedback.

When the StepIt method is called, the value of Position increases by Step.

## **TVrProgressBar.StepBy**

[TVrProgressBar](#) [see also](#)

Advances the Position of the progress bar by a specified amount.

**procedure** StepBy(Delta: Integer);

### **Description**

Advances the Position of the progress bar by a specified amount. Call StepBy to increase the value of Position by the value of the Delta parameter. To advance Position by a default amount that represents a single step in the process, use the StepIt method.

## **TVrProgressBar.StepIt**

[TVrProgressBar](#) [see also](#)

Call the StepIt method to increase the value of Position by the value of the Step property.

```
procedure StepIt;
```

### **Description**

Advances Position by the amount specified in the Step property.

Call the StepIt method to increase the value of Position by the value of the Step property. If Step represents the size of one logical step in the process tracked by the progress bar, call Step after each logical step is completed.



## **TVrRaster**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrRaster consists of rows and columns of led typed cells.

### **Unit**

vrraster

### **Description**

TVrRaster consists of rows and columns of led typed cells. Each cell has it's own individual active property. TVrRaster is derived from TCustomVrRaster which inherits everything from TGraphicControl.

## **TVrRaster.Bevel**

[TVrRaster](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## TVrRaster.Columns

[TVrRaster](#) [see also](#)

Defines the number of visible horizontal columns.

```
type TVrColInt = 1..MaxInt;  
property Columns: TVrColInt;
```

### Description

Use columns to set the number of visible columns.

## **TVrRaster.Count**

[TVrRaster](#) [see also](#)

Contains the number of cells.

**property** Count: Integer;

### **Description**

At runtime you can determine the number of leds/cells by calling the count property. This is the result from Columns \* Rows. Remember that the cell structure is zero-based when referencing individual cells.

```
procedure TForm1.VrRaster1Click(Sender: TObject);
var
  I: Integer;
begin
  with VrRaster1 do
    for I := 0 to Count - 1 do Items[I].Active := True;
end;
```

## TVrRaster.Items

[TVrRaster](#) [see also](#)

Used to access each single cell.

**property** `Items[Index: Integer]: TVrRasterLed;`

### Description

To access each single led/cell in TVrRaster you can use the Items property. Remember that the cell structure is zero-based when referencing individual cells. Each item is derived from TObject.

The first cell: `Items[0]`

The last cell: `Items [Count-1]`

### property Active

Changes the state of each led/cell to it's on/off state.

**Example:** `Items[0].Active := true;`

## **TVrRaster.MultiSelect**

[TVrRaster](#) [see also](#)

Determines if only one cell can be active at any time.

### **Declaration**

```
property MultiSelect: Boolean;
```

### **Description**

Determines if only one cell can be active at any time. If MultiSelect is false, each active cell is deactivated before the new cell becomes active.

## TVrRaster.Palette

[TVrRaster](#) [see also](#)

Defines the color attributes for TVrRaster.

**property** Palette: TVrPalette;

## **TVrRaster.PlainColors**

[TVrRaster](#) [see also](#)

Determines the use of gradient paint procedures.

**property** PlainColors: boolean;

### **Description**

Draws a gradient pattern on the surface of each led/cell or use the default 16 color palette. Using only the 16 color palette will increase the general performance of the component.



## **TVrRaster.Rows**

[TVrRaster](#) [see also](#)

Describes the number of visible vertical rows.

**type** TVrRowInt = 1..MaxInt;

**property** Rows: TVrRowInt;

### **Description**

Rows define the number of visible vertical cells.

## **TVrRaster.Spacing**

[TVrRaster](#)

[see also](#)

Defines the space between each single led.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single led.

## **TVrRaster.Style**

[TVrRaster](#) [see also](#)

Defines the type of outline of each single led.

```
type TVrRasterStyle = (rsRaised, rsLowered, rsNone, rsFlat);  
property Style: TVrRasterStyle;
```

## **TVrRocker**

[properties](#)[methods](#) [events](#) [see also](#)

Represents a lightswitch styled switch control.

### **Unit**

vrrocker

### **Description**

TVrRocker is derived from TVrGraphicControl and represents a lightswitch styled switch control.

## TVrRocker.Depth

[TVrRocker](#) [see also](#)

Determines the size of the 3d effect.

**property** Depth: TVrByteInt;

### Description

Depth can have values between 0 and 255. It determines the size of the 3d effect of TVrRocker.

## **TVrRocker.FocusColor**

[TVrRocker](#) [see also](#)

Color for the outline when the PowerButton becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the Rocker receives focus, a small rectangle is painted around the button with the color defined by FocusColor.

## **TVrRocker.HighlightColor**

[TVrRocker](#) [see also](#)

HighlightColor is one of the three colors which make the 3d effect.

**property** HighlightColor: TColor;

### **Description**

HighlightColor is one of the three colors which make the 3d effect. This 3d effect is composed of three colors: ShadowColor, ShadowLightColor and HighlightColor.

## **TVrRocker.OnLowerClick**

[TVrRocker](#) [see also](#)

Is triggered as soon as the lower section of the switch is pressed.

**property** OnLowerClick: TNotifyEvent;

### **Description**

OnUpperClick is triggered as soon as the lower section of the rocker switch is pressed.



## **TVrRocker.OnUpperClick**

[TVrRocker](#) [see also](#)

Is triggered as soon as the upper section of the switch is pressed.

**property** OnUpperClick: TNotifyEvent;

### **Description**

OnUpperClick is triggered as soon as the upper section of the rocker switch is pressed.

## **TVrRocker.OutlineColor**

[TVrRocker](#) [see also](#)

Describes the attributes of a small rectangle painted around the button image.

**property** OutlineColor: TColor;

### **Description**

OutlineColor defines the color which is used to paint a small outline rectangle around the button image.

## **TVrRocker.OutlineWidth**

[TVrRocker](#) [see also](#)

Describes the size in pixels of the outline which is painted as a small rectangle.

**property** OutlineWidth: TVrByteInt;

### **Description**

OutlineWidth describes the size in pixels of the outline which is painted as a small rectangle.

## **TVrRocker.ShadowColor**

[TVrRocker](#) [see also](#)

ShadowColor is one of the three colors which make the 3d effect.

**property** ShadowColor: TColor;

### **Description**

ShadowColor is one of the three colors which make the 3d effect. This 3d effect is composed of three colors: ShadowColor, ShadowLightColor and HighlightColor.

## **TVrRocker.ShadowLightColor**

[TVrRocker](#) [see also](#)

ShadowLightColor is one of the three colors which make the 3d effect.

**property** ShadowLightColor: TColor;

### **Description**

ShadowLightColor is one of the three colors which make the 3d effect. This 3d effect is composed of three colors: ShadowColor, ShadowLightColor and HighlightColor.

## TVrRocker.State

[TVrRocker](#) [see also](#)

**type** TVrRockerState = (stUpperDown, stLowerDown, stNone);

**property** State: TVrRockerState;

### Description

Use state to retrieve or set the current state of the rocker button. If the style of the rocker is set to [<rsButton>](#) State will always be stNone ad only the Upperclick and LowerClick events are triggered.

## **TVrRocker.Style**

[TVrRocker](#) [see also](#)

**type** TVrRockerStyle = (rsButton, rsSwitch);

**property** Style: TVrRockerStyle;

### **Description**

Determines how the rocker switch handles mouse clicks. If Style is rsButton the rocker always flips back to a normal "stNone" state.

## **TVrRunOnce**

[properties](#)[methods](#)

[events](#)

[see also](#)

Makes sure only one instance of an application is active at the time.

### **Unit**

vrsystem

### **Description**

TVrRunOnce will make sure only one instance of an application is active at the time. Terminate the second instance or just show a warning message.



## **TvrRunOnce.MessageText**

[TvrRunOnce](#) [see also](#)

Contains the actual message showed to the user.

**property** MessageText: string;

### **Description**

MessageText contains the actual message showed to the user. See also [ShowMessageText](#).

## **TVrRunOnce.OnExists**

[TVrRunOnce](#) [see also](#)

Is triggered as soon as another instance of the current application is detected.

**property** OnExists: TNotifyEvent;

### **Description**

OnExists is triggered as soon as another instance of the current application is detected.

## **TVrRunOnce.RestorePrevInst**

[TVrRunOnce](#) [see also](#)

Determines if a previous instance of the application should be restored.

**property** RestorePrevInst: Boolean;

### **Description**

Determines if a previous instance of the application should be restored (becomes the active application on the windows desktop).

## TVrRunOnce.ShowMessage

[TVrRunOnce](#) [see also](#)

Applies only when a previous instance of the application already exists.

**property** ShowMessage: Boolean;

### Description

Set ShowMessage to true if a message should be displayed by the TVrRunOnce component. Applies only when a previous instance of the application already exists.

## **TVrRunOnce.Terminate**

[TVrRunOnce](#) [see also](#)

Determines if the current application should be closed and terminated.

**property** Terminate: Boolean;

### **Description**

Determines if the current application should be closed and terminated when another instance already exists.

## **TVrScale**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrScale is used to display a scale.

### **Unit**

vrscale

### **Description**

TVrScale is used to display a separate scale, with numeric values and scale ticks, along another component. TVrScale is derived from TVrGraphicControl and contains several properties.

## **TVrScale.Alignment**

[TVrScale](#) [see also](#)

Controls the horizontal placement of the text within the scale.

**property** Alignment: TAlignment;

### **Description**

Controls the horizontal placement of the text within the scale. Set Alignment to specify how the text of the scale is aligned within the ClientRect of the scale control.

## **TVrScale.Digits**

[TVrScale](#) [see also](#)

Defines the minimum number of digits for each value.

**property** Digits: Integer;

### **Description**

The scale consists out of values in the range between Min and Max. Digits defines the minimum number of digits for each value.



## **TVrScale.Layout**

[TVrScale](#) [see also](#)

Specifies the vertical placement of the text within the scale.

**property** Layout: TTextLayout;

### **Description**

Specifies the vertical placement of the text within the scale. Set Layout to specify how the text of the scale is placed within the ClientRect of the scale control.

## **TVrScale.LeadingZero**

[TVrScale](#)

[see also](#)

Determines if numeric values are left padded with zero's.

**property** LeadingZero: Boolean;

### **Description**

Depending on the Digits property, numeric values are left padded with zero's.

## **TVrScale.MaxValue**

[TVrScale](#) [see also](#)

Used to define an upper limit of the value that can be represented using the Scale.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set an upper limit to the value that can be represented using the Scale.

## **TVrScale.MinValue**

[TVrScale](#) [see also](#)

Used to define a lower limit to the value that can be represented using the Scale.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the Scale.

## **TVrScale.Orientation**

[TVrScale](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrScale.PeakColor**

[TVrScale](#) [see also](#)

Used to differ the color of the scale values from a certain starting point.

**property** PeakColor: TColor;

### **Description**

PeakColor is used to differ the color of the scale values from a certain starting point defined by PeakLevel.

## **TVrScale.PeakLevel**

[TVrScale](#) [see also](#)

Used the differ the color of the scale values from a certain starting point.

**property** PeakLevel: Integer;

### **Description**

PeakLevel contains a value within the range defined by Min and Max. PeakLevel is used the differ the color of the scale values from a certain starting point.

## **TVrScale.ScaleColor**

[TVrScale](#) [see also](#)

Defines the color attribute of the scale ticks.

**property** ScaleColor: TColor;

### **Description**

ScaleColor defines the color of the scale ticks.



## **TVrScale.ScaleOffset**

[TVrScale](#) [see also](#)

Defines the position of the TickMarks with the control.

**property** ScaleOffset: Integer;

### **Description**

ScaleOffset defines the position of the TickMarks within the control.

## **TVrScale.ShowSign**

[TVrScale](#)

[see also](#)

Displays the + sign with positive values.

**property** ShowSign: Boolean;

### **Description**

If ShowSign is set to True all positive values are displayed with the + sign prefix.

## **TVrScale.TickMarks**

[TVrScale](#) [see also](#)

Specifies how tick marks are placed on the scale.

**type** TVrTickMarks = (tmNone, tmBoth, tmBottomRight, tmTopLeft);

**property** TickMarks: TVrTickMarks;

### **Description**

Set TickMarks to specify whether the scale should display tick marks, and if so, how those tick marks are set.

## **TVrScale.Ticks**

[TVrScale](#) [see also](#)

Defines the number of scale ticks.

**property** Ticks: Integer;

### **Description**

Ticks defines the number of scale ticks.

## **TVrScale.TicksHeight**

[TVrScale](#)

[see also](#)

Defines the height of each scale tick.

**property** TicksHeight: Integer;

### **Description**

TicksHeight defines the height of each scale tick.

## **TVrScale.TicksWidth**

[TVrScale](#) [see also](#)

Defines the Width of each scale tick.

**property** TicksWidth: Integer;

### **Description**

TicksWidth defines the width of each scale tick.

## **TVrScale.Transparent**

[TVrScale](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## TVrScanner

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrScanner is a tool to display a row of horizontal leds which can be highlighted.

### Unit

vrscanner

### Description

TVrScanner is derived from TVrLedGroup which is derived from TVrGraphicControl. It's a tool to display a row of horizontal leds which can be highlighted. TVrScanner contains a timer component which is assigned to the scanning procedure. With scanning enabled one of the Leds is highlighted along the row of leds in a direction defined by Direction. The speed of the animation can be set with TimeInterval.



## **TVrScanner.Active**

[TVrScanner](#) [see also](#)

Enables the automatic scanning feature or animation.

**property** Active: Boolean;

### **Description**

Enables the automatic scanning feature or animation. The highlighted led moves in the direction defined by the ScanDirection property.

## **TVrScanner.ColorWidth**

[TVrScanner](#) [see also](#)

Describes the size of each color transition.

**property** ColorWidth: TVrInt10;

### **Description**

ColorWidth can be used to speed up the painting process. Assigning a small value will make the gradient look smoother but will take a longer time to refresh on screen.

## **TVrScanner.Direction**

[TVrScanner](#) [see also](#)

Defines the direction to move the activated led.

```
type TVrScanDirection = (sdBoth, sdLeftRight, sdRightLeft);
```

```
property Direction: TVrScanDirection;
```

### **Description**

Direction defines the direction to move the activated led when the Active property is set to True.

## **TVrScanner.Leds**

[TVrScanner](#) [see also](#)

Defines the number of visible leds.

**property** Leds: Integer;

### **Description**

Defines the number of visible leds. There must be at least 1 led.

## **TVrScanner.LedStyle**

[TVrScanner](#) [see also](#)

**type** TVrLedStyle = class;

**property** LedStyle: TVrLedStyle;

### **Description**

LedStyle contains several color attributes which make up the led image.

## **TVrScanner.OnChange**

[TVrScanner](#) [see also](#)

This event is invoked each time the position value changes.

### **Declaration**

```
property OnChange: TNotifyEvent;
```

### **Description**

The scanning is internally done by a timer component. The timer interval is defined by the TimeInterval property. This event is invoked each time the position value changes.

## **TVrScanner.Palette**

[TVrScanner](#) [see also](#)

Defines the color attributes for TVrScanner.

**property** Palette: TVrPalette;

## **TVrScanner.Position**

[TVrScanner](#) [see also](#)

Used to handle the led states manually.

**property** Position: Integer;

### **Description**

Position allows you to set a led in it's active state. Only one led can be active at a time. To disable this feature set position to -1. This property is used to handle the led states manually.



## **TVrScanner.Spacing**

[TVrScanner](#) [see also](#)

Defines the space between each individual led.

**property** Spacing: Integer;

### **Description**

Increases or decreases the space between each individual led.

## **TVrScanner.Threaded**

[TVrScanner](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrScanner.TimeInterval**

[TVrScanner](#) [see also](#)

Defines the animation speed when scanning is enabled.

**property** TimeInterval: Integer;

### **Description**

Increases or decrease the animation speed when scanning is enabled.

## **TVrScanner.Transparent**

[TVrScanner](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Transparent specifies whether the background of the image obscures objects below the image object. Set Transparent to True to allow objects behind the object to show through the background of the control. Set Transparent to False to make the background opaque.

## **TVrScope**

[properties](#)[methods](#)

[events](#)

[see also](#)

An graphic oscilloscope control like the ones used in Windows NT Taskmanager.

### **Unit**

vrscope

### **Description**

An graphic oscilloscope control like the ones used in Windows NT Taskmanager. It draws a line graph within a predefined grid. The scope is fully scalable and can be resized during runtime sessions. When adding new values to the controls buffer the line graphic is shifted from right to left which produces the animated effect. VrScope is derived from TVrGraphicControl.

## **TVrScope.Active**

[TVrScope](#) *see also*

Determines if the scope is enabled or disabled.

**property** Active: Boolean;

### **Description**

Set Active to True in order to start the scope. Although Active can be set in the designer, it is only activated during runtime sessions.

## **TVrScope.AnimateGrid**

[TVrScope](#) [see also](#)

Makes the grid background image move from right to left.

**property** AnimateGrid: Boolean;

### **Description**

Set AnimateGrid to true to make the grid background image move from right to left. Otherwise a static grid background is used.

## **TvrScope.BaseColor**

[TvrScope](#) *see also*

Describes the pen color attribute which is used to draw the BaseOffset.

**property** BaseLineColor: TColor;

### **Description**

BaseLineColor describes the pen color which is used to draw the BaseOffset.



## **TvrScope.BaseOffset**

[TvrScope](#) *see also*

Defines the starting point of the line graphic.

```
type TVrBaseOffsetInt = 0..100;  
property BaseOffset: TVrBaseOffsetInt;
```

### **Description**

BaseOffset is the starting point of the line graphic. It is a percentage value in the range defined by the Min and Max properties.

## **TvrScope.BufferSize**

[TvrScope](#) [see also](#)

Points to the actual size of the buffer.

**property** BufferSize: Integer;

### **Description**

TvrScope uses an internal buffer to store the values added through the OnNeedData event. The buffer works like a ring buffer and uses FIFO (First In First Out). Make sure to set the buffersize not to small or the graphic will become corrupted when the control is resized during runtime. The size needed depends on the Frequency property and the Width of the Scope control in pixels.

## **TVrScope.Channels**

[TVrScope](#) [see also](#)

**property** Channels: TVrChannels;

**type** TVrChannel = class;

### **Description**

Channels contains all separate channel objects. A TVrChannel object has several independent properties.

**property** TVrChannel.Color: TColor;

Defines the color of the line used to display the graph.

**property** TVrChannel.Width: Integer;

Defines the linewidth of the line used to display the graph.

**type** TGraphStyle = (gsLine, gsBar);

**property** TVrChannel.Style: TGraphStyle;

Defines the type of graph to use.

example:

```
VrScope1.Channels[0].Color := clRed;
```

Use Channels.Count to determine the number of defined channels.

## **TVrScope.Clear**

[TVrScope](#) [see also](#)

Method which purges the data buffer.

```
procedure Clear;
```

### **Description**

Clear method purges the data buffer.

## **TVrScope.Frequency**

[TVrScope](#) [see also](#)

Frequency describes the intervals in pixels which are used to display each value.

**property** Frequency: TVrMaxInt;

### **Description**

Frequency describes the intervals in pixels which are used to display each value. Assigning a higher value will make the graphic more readable and produces faster animation speed.

## **TvrScope.GridColor**

[TvrScope](#) *see also*

Defines the default color of the grid.

**property** GridColor: TColor;

### **Description**

GridColor defines the default color of the grid.

## **TvrScope.GridLineWidth**

[TvrScope](#) [see also](#)

Specifies the width (in pixels) of the lines that separate the cells of the grid.

**property** GridLineWidth: Integer;

### **Description**

Specifies the width (in pixels) of the lines that separate the cells of the grid. Set GridLineWidth to make the lines that separate the cells in the grid heavier or lighter. When GridLineWidth is zero, no separators are drawn between the cells of the grid.

**Note:** Values greater than 3 pixels are not recommended for applications that will run on Windows 95 because of distortions that can appear.

## **TVrScope.GridSize**

[TVrScope](#) [see also](#)

Defines the size of each grid cell in pixels.

**property** GridSize: TVrMaxInt;

### **Description**

GridSize defines the size of each grid cell in pixels.



## **TVrScope.Interval**

[TVrScope](#) *see also*

Determines the amount of time, in milliseconds, that passes before the internal timer initiates another event.

**property** Interval: Integer;

### **Description**

Interval determines how frequently a timer event occurs. Each time the specified interval passes, a timer event occurs. Use Interval to specify any integer value as the interval between timer events. The default value is 1000 (one second).

## **TVrScope.MaxValue**

[TVrScope](#) *see also*

Used to set an upper limit to the value that can be represented using the Scope.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set an upper limit to the value that can be represented using the Scope. Values added into the scope's buffer can never be smaller than MinValue.

## **TvrScope.MinValue**

[TvrScope](#) [see also](#)

Used to set a lower limit to the value that can be represented using the Scope.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the Scope. Values added into the scope's buffer can never be below MinValue.

## **TVrScope.OnNeedData**

[TVrScope](#) [see also](#)

type TVrNeedDataEvent = procedure(Sender: TObject; Channel: Integer; var Value: Integer) of object;  
property OnNeedData: TVrNeedDataEvent;

### Description

The OnNeedData event is called as soon as the scope is activated. The parameter Channel defines the channel number needing a new value. Use the Value parameter to assign this new value.

## **TvrScope.Threaded**

[TvrScope](#) [see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## TVrShadowButton

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrShadowButton is a push button control with an 3d effect.

### Unit

vrbuttons

### Description

TVrShadowButton is a 3d button control with a shadow area. Use TVrShadowButton to put a Windows push button on a form. TVrShadowButton introduces several properties to control its behavior. Users choose button controls to initiate actions.

## **TVrShadowButton.Depth**

[TVrShadowButton](#) [see also](#)

Describes the offset of the shadow area

**property** Depth: Integer;

### **Description**

Depth describes the offset of the shadow area within the controls boundaries. Clicking the shadow area does NOT result in an OnClick event.

## **TVrShadowButton.DepthAdjust**

[TVrShadowButton](#) [see also](#)

Is used when the button is pressed/down

**property** DepthAdjust: Integer;

### **Description**

DepthAdjust is used when the button is pressed/down. The shadow image and the button image are moved along side defined by DepthAdjust.



## **TVrShadowButton.Direction**

[TVrShadowButton](#) [see also](#)

Defines the placement of the shadow area.

**type** TVrShadowDirection = (sdTopLeft, sdTopRight, sdBottomLeft, sdBottomRight)

**property** Direction: TVrShadowDirection;

### **Description**

Direction defines the placement of the shadow area.

## **TVrShadowButton.DisabledTextColor**

[TVrShadowButton](#) [see also](#)

Describes the color for disabled text.

**property** DisabledTextColor: TColor;

### **Description**

DisabledTextColor describes the color for disabled text. DisabledTextColor overrides the existing font color when enabled is set to false.

## **TVrShadowButton.ShadowColor**

[TVrShadowButton](#) [see also](#)

Defines the color in which the shadow area appears.

**property** ShadowColor: TColor;

### **Description**

ShadowColor defines the color in which the shadow area appears.

## **TVrShadowButton.ShadowOutline**

[TVrShadowButton](#) *see also*

Defines the outlinecolor of the shadow rectangle.

**property** ShadowOutline: TColor;

### **Description**

ShadowOutline defines the outlinecolor of the shadow rectangle.

## **TVrShadowButton.Style**

[TVrShadowButton](#) [see also](#)

Defines the outline of the button shape.

**type** TVrShadowButtonStyle = (ssRectangle, ssRoundRect);

**property** Style: TVrShadowButtonStyle;

### **Description**

Style defines the outline of the button shape.

## **TVrShadowButton.TextAlign**

Controls the placement of the text within the image control.

```
type TVrTextAlignment = (vtaLeft, vtaCenter, vtaRight, vtaTopLeft, vtaTop, vtaTopRight, vtaBottomLeft, vtaBottom, vtaBottomRight);  
property TextAlign: TVrTextAlignment;
```

### **Description**

Set TextAlign to specify how the text is aligned within the ClientRect of the image control.

## **TVrShadowButton.Transparent**

[TVrShadowButton](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

When Transparent is true, all unused area's within the controls boundaries are made transparent.

## TVrShapeBtn

[properties](#)[methods](#)

[events](#)

[see also](#)

Is used to transform a single bitmap image in a 3D rendered image.

### Unit

VrButtons

### Description

TVrShapeBtn is derived from TVrGraphicControl. It is used to transform a single bitmap image in a 3D rendered image which makes the button image. Try different images to get the best results possible. Not every image is suitable for 3d rendering.



## **TVrShapeBtn.Bitmap**

[TVrShapeBtn](#) [see also](#)

Contains the actual bitmap which is rendered to a full 3D image.

**property** Bitmap: TBitmap;

### **Description**

Bitmap contains the actual bitmap which is rendered to a full 3D image.

## **TVrShapeBtn.Transparent**

[TVrShapeBtn](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## TVrSlider

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrSlider is a smooth slider control.

### Unit

vrslider

### Description

TVrSlider is LCD control. It's a smooth slider control which allows the user to change certain values. The values are changed by dragging the controls handle or thumb with a mouse. The cursor automatically changes it's appearance when moved over the handle. It supports two directions horizontal and vertical. TVrTracker is derived from TVrCustomControl.

## **TVrSlider.Bevel**

[TVrSlider](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrSlider.BitmapList**

[TVrSlider](#)

[see also](#)

Contains the graphical images

**property** BitmapList: TVrBitmapList;

### **Description**

A BitmapList is separate buffer component which contains a list of unformatted TBitmap objects. These graphical images are used by TVrSlider.

## **TVrSlider.BorderColor**

[TVrSlider](#)

[see also](#)

Describes the attributes of a small rectangle painted around the client area.

**property** BorderColor: TColor;

### **Description**

BorderColor defines the color which is used to paint a small outline rectangle around the slider control.

## **TVrSlider.BorderWidth**

[TVrSlider](#) [see also](#)

Is used in combination of FocusColor.

**property** BorderWidth: Integer;

### **Description**

BorderWidth is used in combination of FocusColor and BorderColor. BorderWidth therefore defines a focus rectangle which is painted along the sides of the slider control when the slider receives focus.

## **TVrSlider.FocusColor**

[TVrSlider](#)

[see also](#)

Color for the outline when the slider becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the slider receives focus, a small rectangle is painted around the slider with the color defined by FocusColor.



## **TVrSlider.KeyIncrement**

[TVrSlider](#)

[see also](#)

Increment value when the thumb is moved with the arrow keys.

**property** KeyIncrement: Integer;

### **Description**

Increment value when the thumb is moved with the arrow keys. Each key stroke will increase or decrease the position value.

## **TVrSlider.MaxValue**

[TVrSlider](#)

[see also](#)

Used to set an upper limit to the value that can be represented using the Slider.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set an upper limit to the value that can be represented using the slider. The thumb indicates the current Position in a range between MinValue and MaxValue.

## **TVrSlider.MinValue**

[TVrSlider](#)

[see also](#)

Used to set a lower limit to the value that can be represented using the Slider.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the Slider. The thumb indicates the current Position in a range between MinValue and MaxValue.

## **TVrSlider.OnChange**

[TVrSlider](#)

[see also](#)

Occurs when the position value changes.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange occurs when the position value changes.

## **TVrSlider.Options**

[TVrSlider](#) [see also](#)

Specifies various display and behavioral properties of the slider.

```
type TVrSliderOption = (soActiveClick, soMouseClip, soHandPoint,  
soThumbOpaque);  
type TVrSliderOptions = set of TVrSliderOption;  
property Options: TVrSliderOptions;
```

### **Description**

#### **soActiveClick**

Clicking inside the controls boundaries will update the position of the thumb.

#### **soMouseClip**

When the thumb is being dragged the mouse cannot leave the controls boundaries.

#### **soHandPoint**

Use a special handcursor for the thumb.

#### **soThumbOpaque**

if not ThumbOpaque then the Thumb is painted transparent using the transparentcolor of the thumb image.

## **TVrSlider.Orientation**

[TVrSlider](#)

[see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrSlider.Position**

[TVrSlider](#) [see also](#)

Returns the actual position of the thumb pad.

**property** Position: Integer;

### **Description**

Position returns the actual position of the thumb pad.

## **TVrSlider.SolidFill**

[TVrSlider](#) [see also](#)

Determines if each single step is filled with a dither pattern.

**property** SolidFill: Boolean;

### **Description**

When SolidFill is false each single step is filled with a dither pattern.



## **TVrSlider.Spacing**

[TVrSlider](#)

[see also](#)

Defines the space between each single segment within the image with Spacing.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single step within the image with Spacing.

## **TVrSlider.Style**

[TVrSlider](#) [see also](#)

Determines the direction of the slider.

**type** TVrProgressStyle = (psBottomLeft, psTopRight);

**property** Style: TVrProgressStyle;

### **Description**

Position points to the current state or progress made. With psBottomLeft progress is displayed from Bottom-to-Top or from Left-to-Right (default). Use style to change the starting point of the display.

## **TVrSlider.ThumbImageIndex**

[TVrSlider](#) [see also](#)

Points to the glyph index contained in the BitmapList.

**property** ThumbImageIndex: Integer;

### **Description**

ThumbImageIndex is used in combination with the BitmapList property. ThumbImageIndex points to the glyph index contained in the BitmapList. Each glyph can contain multiple images. See also [ThumbStates](#).

## **TVrSlider.ThumbIndent**

[TVrSlider](#) [see also](#)

Describes the offset of the thumb from the defined border.

**property** ThumbIndent: Integer;

### **Description**

ThumbIndent describes the offset of the thumb from the defined border. Therefore the thumbs start and end positions can be adjusted.

## TVrSlider.ThumbStates

[TVrSlider](#) [see also](#)

Describes the number of thumb images.

**property** ThumbStates: Integer;

### Description

ThumbStates defines the number of thumb images contained in the HThumb and VThumb properties. Each thumb represents a certain state. Atleast 1 thumb image must be defined.

Value	Meaning
1	Normal
2	Disabled
3	Thumb is pressed and is being dragged
4	Mouse is moved over the thumb

## **TVrSlider.TickWidth**

[TVrSlider](#) [see also](#)

Defines the size of each single step.

**property** TickWidth: Integer;

### **Description**

TVrSlider is build out of steps. TickWidth defines the size of each single step.

## **TVrSlideShow**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrSlideShow component is a bitmap transition component.

### **Unit**

vrslideshow

### **Description**

A TVrSlideShow component is a bitmap transition component to create animated banners in your Delphi programs. There are 19 different transition types. TVrSlideShow is derived from TGraphicControl.

## **TVrSlideShow.Active**

[TVrSlideShow](#)

[see also](#)

Determines if animation is enabled or disabled.

**property** Active: Boolean;

### **Description**

Set Active to True in order to show the current animation. Although Active can be set in the designer, it is only activated during runtime sessions.



## **TVrSlideShow.BitmapList**

[TVrSlideShow](#)

[see also](#)

Contains the graphical images

**property** BitmapList: TVrBitmapList;

### **Description**

A BitmapList is separate buffer component which contains a list of unformatted TBitmap objects. These graphical images are used by TVrSlideShow. The bitmaplist used must be exclusive for TVrSlideShow because it cycles through all existing bitmaps.

## **TVrSlideShow.ImageIndex1**

[TVrSlideShow](#)

[see also](#)

Describes the original bitmap image.

**property** ImageIndex1: Integer;

### **Description**

ImageIndex1 describes the original bitmap image. During animation ImageIndex1 will be combined with the Image defined by ImageIndex2 (destination). The resulting effect is defined by Transition. ImageIndex1 is always the first image displayed at runtime.

## **TVrSlideShow.ImageIndex2**

[TVrSlideShow](#)

[see also](#)

Describes the destination bitmap.

**property** ImageIndex2: Integer;

### **Description**

ImageIndex2 describes the destination bitmap image. During animation ImageIndex1 will be combined with ImageIndex2 (destination). The resulting effect is defined by [Transition](#).

## **TVrSlideShow.Interval**

[TVrSlideShow](#)

[see also](#)

Determines the animation speed.

**property** Interval: integer;

### **Description**

Interval describes the animation speed in milliseconds. A lower interval result in higher animation speed.

## **TVrSlideShow.Loop**

[TVrSlideShow](#)

[see also](#)

Determines if the animation will play continuously or terminate.

**property** Loop: Boolean;

### **Description**

The Loop property determines if the animation will play continuously or terminate when the last frame is displayed.

## **TVrSlideShow.OnNextSlide**

[TVrSlideShow](#)

[see also](#)

Called before the next transition starts.

**property** OnNextSlide: TNotifyEvent;

### **Description**

OnNextSlide is called before the next transition starts.

## **TVrSlideShow.OnNotify**

[TVrSlideShow](#)

[see also](#)

Called when the last sequence of the animation is displayed.

**property** OnNotify: TNotifyEvent;

### **Description**

OnNotify is called when the last sequence of the animation is displayed. OnNotify only applies when the Loop property is set to false.

## **TVrSlideShow.Steps**

[TVrSlideShow](#)

[see also](#)

Describes the number of animation frames.

**property** Steps: integer;

### **Description**

Steps describes the number of animation frames.



## **TVrSlideShow.Threaded**

[TVrSlideShow](#)

[see also](#)

Use a threaded timer for the animation sequence.

**property** Threaded: Boolean;

### **Description**

Set threaded to true to use a threaded timer for the animation sequence. Otherwise no Win32 threads are used, only message based timers.

## **TVrSlideShow.Transition**

[TVrSlideShow](#)

[see also](#)

Describes the type of transition.

```
type TVrTransitionEffect = (StretchFromLeft, StretchFromRight, StretchFromTop,  
StretchFromBottom, StretchFromTopLeft, StretchFromBottomRight,  
StretchFromXcenter, StretchFromYcenter, PushFromBottom, PushFromLeft,  
PushFromRight, PushFromTop, SlideFromLeft, SlideFromRight, SlideFromTop,  
SlideFromBottom, SlideFromTopLeft, SlideFromBottomRight, Zoom);  
property Transition: TVrTransitionEffect;
```

### **Description**

Transition defines the type of transition for the animation sequence.

## TVrSpectrum

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrSpectrum consists of a row of vertical bars to display a collection of signals.

### Unit

vrspectrum

### Description

TVrSpectrum consists of a row of vertical bars to display a collection of signals. Each individual bar can be accessed through the items property. The number of bars can be set by Columns. TVrSpectrum is derived from TVrGraphicControl.

## **TVrSpectrum.BarSpacing**

[TVrSpectrum](#) [see also](#)

Determines the space between each bar.

**property** BarSpacing: Integer;

### **Description**

BarSpacing determines the space in pixels between each separate bar.

## **TVrSpectrum.BarWidth**

[TVrSpectrum](#) [see also](#)

Determines the width of each bar.

**property** BarWidth: Integer;

### **Description**

BarWidth determines the width of each separate bar.

## **TVrSpectrum.Bevel**

[TVrSpectrum](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## TVrSpectrum.Columns

[TVrSpectrum](#) [see also](#)

Defines the number of bars used by the spectrum component.

**property** Columns: Integer;

### Description

Number of bars used by the spectrum component. Each bar has it's own properties. See the [Items](#) property for more information.

## **TVrSpectrum.Count**

[TVrSpectrum](#) [see also](#)

Returns the number of visible bars

**property** Count: Integer;

### **Description**

Returns the number of visible bars. Runtime only property. Remember that the structure is zero-based.



## TVrSpectrum.Items

[TVrSpectrum](#) [see also](#)

With the items property you can access each bar separately.

```
type TVrSpectrumBar = class(TVrCollectionItem);  
property Items[Index: Integer]: TVrSpectrumBar;
```

### Description

With the items property you can access each bar separately. The structure is zero based. Items is a runtime only property.

the first bar:     Items[0]  
the last bar:     Items[Count - 1]

### property Position

Defines the highlighted area within the bar in the range between Min and Max.

```
procedure TForm1.SpectrumTimerTimer(Sender: TObject);  
var  
    I, Value: Integer;  
begin  
    with VrSpectrum1 do  
        for I := 0 to Pred(Count) do  
            begin  
                Value := Random(Max + 1);  
                Items[I].Position := Value;  
            end;  
    end;  
end;
```

## **TVrSpectrum.MarkerColor**

[TVrSpectrum](#) [see also](#)

Defines the color of the tooltip.

**property** MarkerColor: TColor;

### **Description**

Each bar has a small tool tip to highlight it's maximum position. With MarkerColor you can change the used color.

## **TVrSpectrum.MarkerVisible**

[TVrSpectrum](#) [see also](#)

Hide or show the tooltip defined by MarkerColor.

**property** MarkerVisible: Boolean;

### **Description**

Hide or show the tooltip defined by MarkerColor.

## **TVrSpectrum.MaxValue**

[TVrSpectrum](#) [see also](#)

Used to set a upper limit to the value that can be represented using each bar.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set a upper limit to the value that can be represented using each bar. A highlighted area indicates the current Position in a range between MinValue and MaxValue.

## **TVrSpectrum.MinValue**

[TVrSpectrum](#) [see also](#)

Used to set a lower limit to the value that can be represented using each bar.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using each bar. A highlighted area indicates the current Position in a range between MinValue and MaxValue.

## **TVrSpectrum.Palette1**

[TVrSpectrum](#) [see also](#)

Color attributes for the lower bar area.

**property** Palette1: TVrPalette;

### **Description**

Used to fill part of the bar described by Percent1.

## **TVrSpectrum.Palette2**

[TVrSpectrum](#) [see also](#)

Color attributes for the medium bar area.

**property** Palette2: TVrPalette;

### **Description**

Used to fill part of the bar described by Percent2.

## **TVrSpectrum.Palette3**

[TVrSpectrum](#) [see also](#)

Color attributes for the higher bar area.

**property** Palette3: TVrPalette;

### **Description**

The Percent1 and Percent2 properties describe the lower and medium bar areas. The rest is filled with Palette3.



## TVrSpectrum.Percent1

[TVrSpectrum](#) [see also](#)

Describes the size of the bar used for the lower segment palette.

**property** Percent1: Integer;

### Description

Percent1 and Percent2 make up the levels in TVrSpectrum. Percent1 describes the percentage of the bar which is filled with the lower segment palette1 colors. Percent1 and Percent2 together can never exceed 100%.

## **TVrSpectrum.Percent2**

[TVrSpectrum](#) [see also](#)

Describes the size of the bar used for the medium segment palette.

**property** Percent2: Integer;

### **Description**

Percent1 and Percent2 make up the levels in TVrSpectrum. Percent2 describes the percentage of the bar which is filled with the medium segment palette2 colors. Percent1 and Percent2 together can never exceed 100%.

## **TVrSpectrum.PlainColors**

[TVrSpectrum](#) [see also](#)

Enables or disables the use of gradient paint procedures.

**property** PlainColors: Boolean;

### **Description**

Enables or disables the use of gradient paint procedures.

## **TVrSpectrum.Reset**

[TVrSpectrum](#) [see also](#)

Resets all bar positions to the Value specified (Default state).

**procedure** Reset(Value: Integer);

### **Description**

Resets all bar positions to the Value specified (Default state). The value parameter must be between the min and max values.

## **TVrSpectrum.ShowInactive**

[TVrSpectrum](#) [see also](#)

Determines if inactive bars are visible.

**property** ShowInactive: Boolean;

### **Description**

Set ShowInactive to false in order to hide inactive bars.

## **TVrSpectrum.Spacing**

[TVrSpectrum](#) [see also](#)

Defines the space between each single segment within the image with Spacing.

**property** Spacing: Integer;

### **Description**

Increase or decrease the space between each single step within the image with Spacing.

## **TVrSpectrum.TickHeight**

[TVrSpectrum](#) [see also](#)

Defines the height of each bar segment.

**property** TickHeight: Integer;

### **Description**

Each bar consists out of bar segments. Tickheight defines the height of these ticks.

## TVrSpinner

[properties](#)[methods](#)

[events](#)

[see also](#)

Use TVrSpinner to add an up-down control to a form.

### Unit

vrspinner

### Description

Use TVrSpinner to add an up-down control to a form. Up-down controls consist of a pair of arrow buttons, such as the arrows that appear in a spin box. Up-down controls allow users to change the size of a numerical value by clicking on arrow buttons. After pressing one of the buttons the OnDownClick or the OnUpClick event is triggered.



## **TVrSpinner.FocusControl**

[TVrSpinner](#) [see also](#)

Designates a windowed control associated with the label.

**property** FocusControl: TWinControl;

### **Description**

When TVrSpinner is pressed or receives focus it will automatically reassign the focus to the control defined by the FocusControl property.

## **TVrSpinner.OnDownClick**

[TVrSpinner](#) [see also](#)

This event is called when the DownButton is pressed.

**property** OnDownClick: TNotifyEvent;

### **Description**

OnDownClick is called when the DownButton is pressed.

## **TVrSpinner.OnUpClick**

[TVrSpinner](#) [see also](#)

This event is called when the UpButton is pressed.

**property** OnUpClick: TNotifyEvent;

### **Description**

OnUpClick is called when the UpButton is pressed.

## **TVrSpinner.Orientation**

[TVrSpinner](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrSpinner.Palette**

[TVrSpinner](#) [see also](#)

Defines the color attributes for TVrSpinner.

**property** Palette: TVrPalette;

## **TVrStrEdit**

[properties](#)[methods](#)

[events](#)

[see also](#)

A wrapper for a Windows single-line edit control.

### **Unit**

vredit

### **Description**

Use a TVrStrEdit object to put a standard Windows edit control on a form. Edit controls are used to retrieve text that users type. Edit controls can also display text to the user.

TVrStrEdit implements the generic behavior introduced in TCustomEdit. TVrStrEdit publishes many of the properties inherited from TCustomEdit and introduce new behavior.

## **TVrStrEdit.Alignment**

[TVrStrEdit](#) [see also](#)

Specifies how text is aligned within the edit control.

**property** Alignment: TAlignment;

### **Description**

Use Alignment to specify whether the text should be left-justified, right-justified, or centered.

### **taLeftJustify**

Align text on the left side of the edit control.

### **taCenter**

Center text in the edit control.

### **taRightJustify**

Align text on the right side of the edit control.

## **TVrStrEdit.OnMouseEnter**

[TVrStrEdit](#) [see also](#)

Occurs when the mouse is moved within the controls boundaries.

**property** OnMouseEnter: TNotifyEvent;

### **Description**

Use the OnMouseEnter event handler to cause any special processing to occur when the mouse is moved within the controls boundaries.



## **TVrStrEdit.OnMouseLeave**

[TVrStrEdit](#) [see also](#)

Occurs when the mouse is moved outside the controls boundaries.

**property** OnMouseLeave: TNotifyEvent;

### **Description**

Use the OnMouseLeave event handler to cause any special processing to occur when the mouse is moved outside the controls boundaries.

## **TVrStrEdit.ValidKeys**

[TVrStrEdit](#)

[see also](#)

Contains a string of characters which the user may enter in the editbox

**property** ValidKeys: string;

### **Description**

ValidKeys contains a string of characters which the user may enter in the editbox.

## **TVrStringList**

[properties](#)[methods](#)

[events](#)

[see also](#)

Non visual TStringList container component.

### **Unit**

vrsystem

### **Description**

TVrStringList is a wrapper component which contains a list of strings. Use a string list object to store and manipulate a list of strings.

## **TVrStringList.OnChange**

[TVrStringList](#) [see also](#)

Occurs immediately after the list of strings changes.

**property** OnChange: TNotifyEvent;

### **Description**

Write an OnChange event handler to respond to changes in the list of strings.

Whenever strings in the list are added, deleted, moved, or modified, the following events take place:

- 1 First, an OnChanging event occurs before the change.
- 2 The strings are added, deleted, moved, or modified.
- 3 Finally, an OnChange event occurs.

## **TVrStringList.OnChanging**

[TVrStringList](#) [see also](#)

Occurs immediately before the list of strings changes.

```
property OnChanging: TNotifyEvent;
```

### **Description**

Write an OnChanging event handler to prepare for changes in the list of strings.

Whenever strings in the list are added, deleted, moved, or modified, the following events take place:

- 1 First, an OnChanging event occurs.
- 2 The strings are added, deleted, moved, or modified.
- 3 Finally, an OnChange event occurs after the changes are complete.

## **TVrStringList.Sorted**

[TVrStringList](#) [see also](#)

Specifies whether the strings in the list should be automatically sorted.

**property** Sorted: Boolean;

### **Description**

Set Sorted to True to cause the strings in the list to be automatically sorted in ascending order. Set Sorted to False to allow strings to remain where they are inserted. When Sorted is False, the strings in the list can be put in ascending order at any time by calling the Sort method.

When Sorted is True, do not use Insert to add strings to the list. Instead, use Add, which will insert the new strings in the appropriate position. When Sorted is False, use Insert to add strings to an arbitrary position in the list, or Add to add strings to the end of the list.

## TVrSwitch

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrSwitch is a switch component.

### Unit

vrswitch

### Description

TVrSwitch is a switch component. The switch is activated by moving the thumb up or down. TVrSwitch is derived from TVrCustomImageControl.

## **TVrSwitch.BackImageIndex**

[TVrSwitch](#) [see also](#)

Describes the image used to fill the background of the control.

**property** BackImageIndex: Integer;

### **Description**

BackImageIndex is used in combination of the BitmapList property. BackImageIndex describes the image used to fill the background of the control.



## **TVrSwitch.Bevel**

[TVrSwitch](#) [see also](#)

Defines the attributes of the border which is painted around the client area.

**property** Bevel: TVrBevel;

## **TVrSwitch.BitmapList**

[TVrSwitch](#)

[see also](#)

Contains the graphical images

**property** BitmapList: TVrBitmapList;

### **Description**

A BitmapList is separate buffer component which contains a list of unformatted TBitmap objects. These graphical images are used by TVrSwitch.

## **TVrSwitch.BorderColor**

[TVrSwitch](#)

[see also](#)

Describes the attributes of a small rectangle painted around the client area.

**property** BorderColor: TColor;

### **Description**

BorderColor defines the color which is used to paint a small outline rectangle around the switch control.

## **TVrSwitch.BorderWidth**

[TVrSwitch](#) [see also](#)

Is used in combination of FocusColor.

**property** BorderWidth: Integer;

### **Description**

BorderWidth is used in combination of FocusColor and BorderColor. BorderWidth therefore defines a focus rectangle which is painted along the sides of the slider control when the switch receives focus.

## **TVrSwitch.FocusColor**

[TVrSwitch](#) [see also](#)

Color for the outline when the slider becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the slider receives focus, a small rectangle is painted around the switch with the color defined by FocusColor.

## **TVrSwitch.Offset**

[TVrSwitch](#) [see also](#)

Defines the current switch position.

**property** Offset: Integer;

### **Description**

Offset defines the current switch position. Offset can never exceed the value defined by the Positions property.

## **TVrSwitch.OnChange**

[TVrSwitch](#)

[see also](#)

This event is called when the offset value changes or the thumb is dragged.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called when the offset value changes or the thumb is dragged.

## **TVrSwitch.Options**

[TVrSwitch](#) [see also](#)

Specifies various display and behavioral properties of the switch.

```
type TVrSwitchOption = (soActiveClick, soMouseClip, soHandPoint,  
soThumbOpaque);  
type TVrSliderOptions = set of TVrSliderOption;  
property Options: TVrSliderOptions;
```

### **Description**

#### **soActiveClick**

Clicking inside the controls boundaries will update the position of the thumb.

#### **soMouseClip**

When the thumb is being dragged the mouse cannot leave the controls boundaries.

#### **soHandPoint**

Use a special handcursor for the thumb.

#### **soThumbOpaque**

if not ThumbOpaque then the Thumb is painted transparent using the transparentcolor of the thumb image.



## **TVrSwitch.Orientation**

[TVrSwitch](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrSwitch.Positions**

[TVrSwitch](#) [see also](#)

Describes the number of intervals or options within the switch.

**property** Positions: Integer;

### **Description**

Positions describes the number of intervals or options within the switch.

## **TVrSwitch.Style**

[TVrSwitch](#) [see also](#)

Determines the direction of the switch.

**type** TVrProgressStyle = (psBottomLeft, psTopRight);

**property** Style: TVrProgressStyle;

### **Description**

Offset contains the current state of the switch. With psBottomLeft switch states are displayed from Bottom-to-Top or from Left-to-Right (default). Use style to change the starting point of the display.

## **TVrSwitch.ThumbImageIndex**

[TVrSwitch](#)

[see also](#)

Points to the glyph index contained in the BitmapList.

**property** ThumbImageIndex: Integer;

### **Description**

ThumbImageIndex is used in combination with the BitmapList property. ThumbImageIndex points to the glyph index contained in the BitmapList. Each glyph can contain multiple images. See also [ThumbStates](#).

## **TVrSwitch.ThumbIndent**

[TVrSwitch](#) [see also](#)

Describes the offset of the thumb from the defined border.

**property** ThumbIndent: Integer;

### **Description**

ThumbIndent describes the offset of the thumb from the defined border. Therefore the thumbs start and end positions can be adjusted.

## TVrSwitch.ThumbStates

[TVrSwitch](#) [see also](#)

Describes the number of thumb images.

**property** ThumbStates: Integer;

### Description

ThumbStates defines the number of thumb images contained in the HThumb and VThumb properties. Each thumb represents a certain state. Atleast 1 thumb image must be defined.

Value	Meaning
1	Normal
2	Mouse is moved over the thumb
3	Thumb is pressed and is being dragged
4	Disabled

## **TVrTextOutline**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrTextOutline provides some additional text formatting styles.

### **Unit**

vrclasses

### **Description**

TVrTextOutline provides some additional text formatting styles.

## **TVrTextOutline.Color**

[TVrTextOutline](#) [see also](#)

Color used for painting a outline around the text.

**property** Color: TColor;



## **TVrTextOutline.Visible**

[TVrTextOutline](#) [see also](#)

Determines if the outline attributes are used.

**property** Visible: Boolean;

## **TVrTextOutline.Width**

[TVrTextOutline](#) [see also](#)

Size of the outline around the text.

**property** Width: Integer;

## TVrThread

[properties](#)[methods](#)

[events](#)

[see also](#)

Wrapper component which contains a TThread object.

### Unit

vrthreads

### Description

TVrThread is a wrapper component which contains a TThread object. Use TVrThread to represent an execution thread in a multi-threaded application. Each new instance of TVrThread is a new thread of execution. Multiple instances of a TVrThread class make a Delphi application multi-threaded.

When an application is run, it is loaded into memory ready for execution. At this point it becomes a process containing one or more threads that contain the data, code and other system resources for the program. A thread executes one part of an application and is allocated CPU time by the operating system. All threads of a process share the same address space and can access the process's global variables.

Use threads to improve application performance by

- Managing input from several communication devices.
- Distinguishing among tasks of varying priority.

## **TVrThread.Enabled**

[TVrThread](#) [see also](#)

Determines if the enclosed thread is active.

**property** Enabled: Boolean;

### **Description**

Enabled determines if the enclosed thread is active and the OnExecute event will be triggered.

## **TVrThread.OnExecute**

[TVrThread](#) [see also](#)

Called when the Active property is set to True and the enclosed Thread is called by the CPU.

**property** OnExecute: TNotifyEvent;

### **Description**

OnExecute is called when the Active property is set to True and the enclosed Thread is called by the CPU.

## **TvrThread.Priority**

[TvrThread](#) [see also](#)

Enumerated type whose default is tpNormal; adjust the priority higher or lower as needed.

```
type TThreadPriority = (tpIdle, tpLowest, tpLower, tpNormal, tpHigher,  
tpHighest, tpTimeCritical);  
property Priority: TThreadPriority;
```

### **Description**

The Priority property is an enumerated type whose default is tpNormal; adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the TThread component, as defined in the following table. Windows schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

**Warning:** Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events

## **TvrThread.SyncEvent**

[TvrThread](#) [see also](#)

Executes Method within the main VCL thread.

**property** SyncEvent: Boolean;

### **Description**

The Synchronize method causes the call specified by Method to be executed by the main VCL thread, avoiding multi-thread conflicts. If you are unsure whether a method call is thread-safe, call it from within the main VCL thread by setting SyncEvent to True.

## TVrTimer

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrTimer is an object that encapsulates some Windows timer functions.

### Unit

vrthreads

### Description

Use one timer component for each timer in the application. Timer properties and methods affect the functionality of the timer by providing information for the timer event. This information includes the timer interval. The actual execution of the timer occurs through its OnTimer event.



## **TVrTimer.Enabled**

[TVrTimer](#) [see also](#)

Controls whether the timer responds to timer events.

**property** Enabled: Boolean;

### **Description**

Use Enabled to enable or disable the timer. If Enabled is True, the timer responds normally. If Enabled is False, the timer ignores the OnTimer event. The default is True.

## **TVrTimer.Interval**

[TVrTimer](#) [see also](#)

Determines the amount of time, in milliseconds, that passes before the timer component initiates another OnTimer event.

**property** Interval: Integer;

### **Description**

Interval determines how frequently the OnTimer event occurs. Each time the specified interval passes, the OnTimer event occurs. Use Interval to specify any integer value as the interval between OnTimer events. The default value is 1000 (one second).

## **TVrTimer.OnTimer**

[TVrTimer](#) [see also](#)

Occurs when a specified amount of time, determined by the Interval property, has passed.

**property** OnTimer: TNotifyEvent;

### **Description**

Use OnTimer to write an event handler to execute an action at regular intervals. The Interval property of a timer determines how frequently the OnTimer event occurs. Each time the specified interval passes, the OnTimer event occurs.

## **TVrTimer.Priority**

[TVrTimer](#) [see also](#)

Enumerated type whose default is tpNormal; adjust the priority higher or lower as needed.

```
type TThreadPriority = (tpIdle, tpLowest, tpLower, tpNormal, tpHigher,  
tpHighest, tpTimeCritical);  
property Priority: TThreadPriority;
```

### **Description**

The Priority property is an enumerated type whose default is tpNormal; adjust the priority higher or lower as needed.

TThreadPriority type defines the possible values for the Priority property of the TThread component, as defined in the following table. Windows schedules CPU cycles to each thread based on a priority scale; the Priority property adjusts a thread's priority higher or lower on the scale.

**Warning:** Boosting the thread priority of a CPU intensive operation may "starve" the other threads in the application. Only apply priority boosts to threads that spend most of their time waiting for external events

Only applies when TimerType is set to ttThread.

## **TVrTimer.SyncEvent**

[TVrTimer](#) [see also](#)

Executes Method within the main VCL thread.

**property** SyncEvent: Boolean;

### **Description**

The Synchronize method causes the call specified by Method to be executed by the main VCL thread, avoiding multi-thread conflicts. If you are unsure whether a method call is thread-safe, call it from within the main VCL thread by setting SyncEvent to True.

Only applies when TimerType is set to ttThread.

## **TVrTimer.TimerType**

[TVrTimer](#) [see also](#)

Determines if the internal timer is created by a thread or by a standard message based timer.

```
type TVrTimerType = (ttThread, ttSystem);
```

```
property TimerType: TVrTimerType;
```

### **Description**

TimerType determines if the internal timer is created by a thread or by a standard message based timer. This way it is not required to use threads in an application.

## TVrTrackBar

[properties](#)[methods](#)

[events](#)

[see also](#)

Use TVrTrackBar to put a track bar on a form.

### Unit

vrtrackbar

### Description

Use TVrTrackBar to put a track bar on a form. A track bar represents a position along a continuum using a slider and, optionally, tick marks. A track bar can also display a selected range marked by triangular ticks at the starting and ending positions of the selection.

During program execution, the slider can be moved to the desired position by dragging it with the mouse or by clicking the mouse on the bar. To use the keyboard to move the slider, press the arrow keys or the PageUp and PageDown keys.

## **TVrTrackBar.BackImageIndex**

[TVrTrackBar](#) [see also](#)

Describes the image used to fill the background of the control.

**property** BackImageIndex: Integer;

### **Description**

BackImageIndex is used in combination of the BitmapList property. BackImageIndex describes the image used to fill the background of the control.



## **TVrTrackBar.BitmapList**

[TVrTrackBar](#) [see also](#)

Contains the graphical images

**property** BitmapList: TVrBitmapList;

### **Description**

A BitmapList is separate buffer component which contains a list of unformatted TBitmap objects. These graphical images are used by TVrTrackBar.

## **TVrTrackBar.BorderWidth**

[TVrTrackBar](#) [see also](#)

Describes the offset from the client area.

**property** BorderWidth: Integer;

### **Description**

Use BorderWidth to increase or decrease the space inside the controls boundaries.

## **TVrTrackBar.FocusColor**

[TVrTrackBar](#) [see also](#)

Color for the outline when the TrackBar becomes the active control.

**property** FocusColor: TColor;

### **Description**

When the TrackBar receives focus, a small rectangle is painted around the control with the color defined by FocusColor.

## **TVrTrackBar.FocusOffset**

[TVrTrackBar](#) [see also](#)

Describes the overall offset from the controls boundaries

**property** FocusOffset: Integer;

### **Description**

FocusOffset describes the overall offset from the controls boundaries from which the focus rectangle is painted. Setting FocusOffset less than zero means no focus rectangle is painted.

## **TVrTrackBar.Frequency**

[TVrTrackBar](#) [see also](#)

Specifies the increment between tick marks on the track bar.

**property** Frequency: Integer;

### **Description**

Use Frequency to specify the spacing of the tick marks, using the logical units used by the Position property. For example, a Frequency of 5 sets a tick mark at every fifth possible increment.

## **TVrTrackBar.GutterBevel**

[TVrTrackBar](#) [see also](#)

Defines the attributes of the border which is painted inside the client area.

**property** GutterBevel: TVrBevel;

## **TVrTrackBar.GutterColor**

[TVrTrackBar](#) [see also](#)

Defines the Color of the gutter in it's normal state

**property** GutterColor: TColor;

### **Description**

GutterColor defines the Color of the gutter.

## **TVrTrackBar.GutterWidth**

[TVrTrackBar](#) [see also](#)

Defines the height of the gutter in pixels.

**property** GutterWidth: Integer;

### **Description**

GutterWidth defines the height of the gutter in pixels.



## **TVrTrackBar.MaxValue**

[TVrTrackBar](#) [see also](#)

Used to set an upper limit to the value that can be represented using the TrackBar.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set an upper limit to the value that can be represented using the TrackBar. The position property can never exceed this value.

## **TVrTrackBar.MinValue**

[TVrTrackBar](#) [see also](#)

Used to set a lower limit to the value that can be represented using the TrackBar.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the TrackBar. The position property can never be smaller than this value.

## **TVrTrackBar.OnChange**

[TVrTrackBar](#) [see also](#)

This event is called whenever the position value changes.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called whenever the position value changes.

## TVrTrackBar.Options

[TVrTrackBar](#) [see also](#)

Specifies various display and behavioral properties of the slider.

```
type TVrTrackBarOption = (toActiveClick, toMouseClip, toHandPoint,  
toFixedPoints, toThumbOpaque);
```

```
type TVrTrackBarOptions = set of TVrTrackBarOption;
```

```
property Options: TVrTrackBarOptions;
```

### Description

Value	Meaning
toActiveClick	Clicking inside the controls boundaries will update the position of the thumb.
toMouseClip	When the thumb is being dragged the mouse cannot leave the controls boundaries.
toHandPoint	Use a special handcursor for the thumb.
toFixedPoints	Enable sliding of the thumb with predefined intervals.
toThumbOpaque	if not ThumbOpaque then the Thumb is painted transparent using the transparentcolor of the thumb image.

## **TVrTrackBar.Orientation**

[TVrTrackBar](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## **TVrTrackBar.Position**

[TVrTrackBar](#) [see also](#)

Points to the current state or progress made.

**property** Position: Integer;

### **Description**

Position points to the current state or progress made. The position can never exceed the values defined by the MinValue and MaxValue properties. The thumb indicates the current position between MinValue and MaxValue.

## **TVrTrackBar.ScaleOffset**

[TVrTrackBar](#) [see also](#)

Defines the position of the TickMarks with the control.

**property** ScaleOffset: Integer;

### **Description**

ScaleOffset defines the position of the TickMarks within the control.

## TVrTrackBar.Style

[TVrTrackBar](#) [see also](#)

Determines the direction of the trackbar.

**type** TVrProgressStyle = (psBottomLeft, psTopRight);

**property** Style: TVrProgressStyle;

### Description

Position points to the current state or progress made. With psBottomLeft progress is displayed from Bottom-to-Top or from Left-to-Right (default). Use style to change the starting point of the display.



## **TVrTrackBar.ThumbImageIndex**

[TVrTrackBar](#) [see also](#)

Points to the glyph index contained in the BitmapList.

**property** ThumbImageIndex: Integer;

### **Description**

ThumbImageIndex is used in combination with the BitmapList property. ThumbImageIndex points to the glyph index contained in the BitmapList. Each glyph can contain multiple images. See also [ThumbStates](#).

## TVrTrackBar.ThumbStates

[TVrTrackBar](#) [see also](#)

Describes the number of thumb images.

**property** ThumbStates: Integer;

### Description

ThumbStates defines the number of thumb images. Each thumb represents a certain state. Atleast 1 (normal) thumb image must be defined.

Value	Meaning
1	Normal
2	Disabled
3	Thumb is pressed and is being dragged
4	Mouse is moved over the thumb

## **TVrTrackBar.TickColor**

[TVrTrackBar](#) [see also](#)

Describes the color of the scale.

**property** TickColor: TColor;

### **Description**

TickColor describes the color of the scale.

## **TVrTrackBar.TickMarks**

[TVrTrackBar](#) [see also](#)

Specifies how tick marks are placed on the scale.

**type** TVrTickMarks = (tmNone, tmBoth, tmBottomRight, tmTopLeft);

**property** TickMarks: TVrTickMarks;

### **Description**

Set TickMarks to specify whether the scale should display tick marks, and if so, how those tick marks are set.

## TVrTrayGauge

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrTrayGauge is a progress indicator which appears in the Windows system tray as an icon.

### Unit

vrtraygauge

### Description

TVrTrayGauge is a progress indicator which appears in the Windows system tray as an icon.

## **TVrTrayGauge.Enabled**

[TVrTrayGauge](#) [see also](#)

Controls whether the control responds to mouse, keyboard, and timer events.

**property** Enabled: Boolean;

### **Description**

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. If Enabled is False, the control ignores mouse and keyboard events.

To re-enable a control, set Enabled to True. The user can use the control.

## **TVrTrayGauge.HideTaskBtn**

[TVrTrayGauge](#) [see also](#)

Hide the application button on the Windows TaskBar.

**property** HideTaskBtn: Boolean;

### **Description**

When a form is minimized, HideTaskBtn will hide the application button on the Windows TaskBar.

## **TVrTrayGauge.Max**

[TVrTrayGauge](#) [see also](#)

Used to set an upper limit to the value that can be represented using the TrayGauge.

**property** Max: Integer;

### **Description**

Use Max to set an upper limit to the value that can be represented using the TrayGauge. The [position](#) property can never exceed this value. A highlighted area indicates the current position in a range between Min and Max.



## TVrTrayGauge.Min

[TVrTrayGauge](#) [see also](#)

Used to set a lower limit to the value that can be represented using the TrayGauge.

**property** Min: Integer;

### Description

Use Max to set a lower limit to the value that can be represented using the TrayGauge. The position property can never be smaller than this value. A highlighted area indicates the current Position in a range between Min and Max.

## TVrTrayGauge.OnChange

[TVrTrayGauge](#) [see also](#)

Occurs when the position property is changed.

**property** OnChange: TNotifyEvent;

### Description

OnChange occurs when the position property is changed.

## **TVrTrayGauge.OnRightClick**

[TVrTrayGauge](#) [see also](#)

Standard mouse handling event for the right mouse button click.

**property** OnRightClick: TMouseEvent;

### **Description**

Standard mouse handling event for the right mouse button click. This event is only called when there is no popup menu assigned.

## TVrTrayGauge.Palette

[TVrTrayGauge](#) [see also](#)

Defines the color attributes for TVrTrayGauge.

**property** Palette: TVrPalette;

## TVrTrayGauge.Position

[TVrTrayGauge](#) [see also](#)

Points to the current state or progress made.

**property** Position: Integer;

### Description

Position points to the current state or progress made. The position can never exceed the values defined by the min and max properties. A highlighted area indicates the current position in the range between Min and Max.

## TVrTrayGauge.Style

[TVrTrayGauge](#) [see also](#)

Describes the style in which the trayGauge appears in the windows system tray.

**type** TVrTrayGaugeStyle = (gsSingle, gsDual);

**property** Style: TVrTrayGaugeStyle;

### Description

Optional property which is by default set to isSingle. It describes the style in which the trayGauge appears in the windows system tray.

## **TVrTrayGauge.Visible**

[TVrTrayGauge](#) [see also](#)

Describes if the TrayGauge is visible during runtime sessions.

**property** Visible: Boolean;

### **Description**

Visible describes if the TrayGauge is visible during runtime sessions.

## **TVrTrayIcon**

[properties](#)[methods](#)

[events](#)

[see also](#)

Adds an icon to the Windows system tray.

### **Unit**

vrsystem

### **Description**

Drop this component on a form, and the application automatically adds an icon to the Windows system tray. You can add a popup menu to be invoked when the user right-clicks the icon. Event handlers give you finer control over mouse events.



## **TVrTrayIcon.Enabled**

[TVrTrayIcon](#) [see also](#)

Controls whether the control responds to mouse, keyboard, and timer events.

**property** Enabled: Boolean;

### **Description**

Use Enabled to change the availability of the control to the user. To disable a control, set Enabled to False. If Enabled is False, the control ignores mouse and keyboard events.

To re-enable a control, set Enabled to True. The user can use the control.

## **TVrTrayIcon.HideTaskBtn**

[TVrTrayIcon](#) [see also](#)

Hide the application button on the Windows TaskBar.

**property** HideTaskBtn: Boolean;

### **Description**

When a form is minimized, HideTaskBtn will hide the application button on the Windows TaskBar.

## **TVrTrayIcon.Icon**

[TVrTrayIcon](#) [see also](#)

Contains the icon which is put in the system tray.

**property** Icon: TIcon;

### **Description**

Contains the icon which is put in the system tray. If visible is set to True the icon assigned to the icon property will appear on the system tray.

## **TVrTrayIcon.OnRightClick**

[TVrTrayIcon](#) [see also](#)

Standard mouse handling event for the right mouse button click.

**property** OnRightClick: TMouseEvent;

### **Description**

Standard mouse handling event for the right mouse button click. This event is only called when there is no popup menu assigned.

## **TVrTrayIcon.Visible**

[TVrTrayIcon](#) [see also](#)

Describes if the TrayIcon is visible during runtime sessions.

**property** Visible: Boolean;

### **Description**

Visible describes if the TrayIcon is visible during runtime sessions. If visible is set to True the icon assigned to the icon property will appear on the system tray.

## TVrUpDown

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrUpDown is a push button control with additional features.

### Unit

vrupdown

### Description

Use TVrUpDown to add an up-down control to a form. Up-down controls consist of a pair of arrow buttons, such as the arrows that appear in a spin box. Up-down controls allow users to change the size of a numerical value by clicking on arrow buttons. TVrUpDown cannot receive focus because it is derived from TGraphicControl.

## **TVrUpDown.GlyphsDown**

[TVrUpDown](#)

[see also](#)

Contains the graphical button image.

**property** GlyphsDown: TBitmap;

### **Description**

GlyphsDown can contain up to 4 separate images each indicating a different state. See also [NumGlyphs](#).

## **TVrUpDown.GlyphsUp**

[TVrUpDown](#)

[see also](#)

Contains the graphical button image.

**property** GlyphsUp: TBitmap;

### **Description**

GlyphsUp can contain up to 4 separate images each indicating a different state. See also [NumGlyphs](#).



## TVrUpDown.NumGlyphs

[TVrUpDown](#)

[see also](#)

Describes the number of images.

**type** TVrNumGlyphs = 1..4;

**property** NumGlyphs: Integer;

### Description

NumGlyphs defines the number of button images contained in the GlyphUp and GlyphDown properties. Each button image represents a certain state. Atleast 1 button image must be defined.

Value	Meaning
1	Normal
2	Mouse is moved over the button
3	Button is pressed
4	Button is disabled

## **TVrUpDown.OnDownClick**

[TVrUpDown](#)

[see also](#)

This event is called when the DownButton is pressed.

**property** OnDownClick: TNotifyEvent;

### **Description**

OnDownClick is called when the DownButton is pressed.

## **TVrUpDown.OnUpClick**

[TVrUpDown](#)

[see also](#)

This event is called when the UpButton is pressed.

**property** OnUpClick: TNotifyEvent;

### **Description**

OnUpClick is called when the UpButton is pressed.

## **TVrUpDown.Orientation**

[TVrUpDown](#) [see also](#)

Specifies whether the control is horizontal or vertical.

**type** TVrOrientation = (voVertical, voHorizontal);

**property** Orientation: TVrOrientation;

## TVrUpDown.RepeatClick

[TVrUpDown](#)

[see also](#)

Trigger the click events while one of the buttons is being pressed.

**property** RepeatClick: Boolean;

### Description

Set RepeatClick to true to trigger the click-events with intervals while one of the buttons is hold down.

## TVrUpDown.RepeatPause

[TVrUpDown](#)

[see also](#)

Describes the intervals in which the onclick-events are triggered.

**property** RepeatPause: TVrMaxInt;

### Description

RepeatPause describes the intervals in which the onclick-events are triggered while one of the buttons is being pressed.

## **TVrUpDown.Transparent**

[TVrUpDown](#)

[see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## TVrUserLed

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrUserLed is an user definable led control which represents a small light bulb.

### Unit

vrleds

### Description

TVrUserLed is an user definable led control which represents a small light bulb. Use the active property to change state. The Led is fully scalable and will become larger or smaller when the control is resized.



## **TVrUserLed.Active**

[TVrUserLed](#) [see also](#)

used to toggle the state of the control (true/false).

**property** Active: Boolean;

### **Description**

Use the active property to toggle the state of the control (true/false).

## **TVrUserLed.Bevel**

[TVrUserLed](#) [see also](#)

Describes the outline of the led component.

**property** Bevel: TVrBevel;

### **Description**

Bevel contains the attributes which define the actual led shape.

## TVrUserLed.DrawStyle

[TVrUserLed](#) [see also](#)

Implements custom drawing.

```
type TVrDrawStyle = (dsOwnerDraw, dsNormal);
```

```
property DrawStyle: TVrDrawStyle;
```

### Description

Set DrawStyle to dsOwnerDraw to override any default painting behavior in [OnDraw](#).

## **TVrUserLed.OnChange**

[TVrUserLed](#) [see also](#)

This event is called whenever the Active property value changes.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange is called whenever the Active property value changes.

## TVrUserLed.OnDraw

[TVrUserLed](#) [see also](#)

Implements custom painting within TVrUserLed.

```
type TVrUserLedDrawEvent = procedure(Sender: TObject; Canvas: Canvas; Rect: TRect) of object;  
property OnDraw: TVrUserLedDrawEvent;
```

### Description

Write code in an OnDraw handler to draw to the controls canvas before it is painted.

## **TVrUserLed.OutlineColor**

[TVrUserLed](#) [see also](#)

Describes the color of the outline which is painted as a small rectangle.

**property** OutlineColor: TColor;

### **Description**

OutlineColor describes the color of the outline which is painted as a small rectangle.

## **TVrUserLed.OutlineWidth**

[TVrUserLed](#) [see also](#)

Describes the size in pixels of the outline which is painted as a small rectangle.

**property** OutlineWidth: Integer;

### **Description**

OutlineWidth describes the size in pixels of the outline which is painted as a small rectangle.

## TVrUserLed.PaletteEx

[TVrUserLed](#) [see also](#)

Defines the color attributes for TVrUserLed.

**property** PaletteEx: TVrPaletteEx;

### Description

PaletteEx contains the color attributes used to define the fill style.



## **TVrWave**

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrWave is a wave player component.

### **Unit**

vraudio

### **Description**

VrWave is a wave player component. With the supplied property editor you can load any wave file and play it within your application. The audio data is contained within the application itself, so no need to supply any separate audio files on disk.

## TVrWave.Options

[TVrWave](#) [see also](#)

```
type TVrWaveOption = (vwoAsync, vwoNoDefault, vwoLoop, vwoNoStop, vwoNoWait);  
type TVrWaveOptions = set of TVrWaveOption;  
property Options: TVrWaveOptions;
```

### Description

#### **vwoLoop**

The sound plays repeatedly. You must also specify the vwoAsync flag to indicate an asynchronous sound event.

#### **vwoNoDefault**

No default sound event is used. If the sound cannot be found, TVrWave returns silently without playing the default sound.

#### **vwoNoStop**

The specified sound event will yield to another sound event that is already playing. If a sound cannot be played because the resource needed to generate that sound is busy playing another sound, the function immediately returns FALSE without playing the requested sound. If this flag is not specified, TVrWave attempts to stop the currently playing sound so that the device can be used to play the new sound.

#### **vwoNoWait**

If the driver is busy, return immediately without playing the sound.

#### **vwoAsync**

Asynchronous playback of a sound event. TVrWave returns immediately when the sound event starts.

## **TVrWave.Play**

[TVrWave](#) [see also](#)

Playback of the sound contained in the Sound property.

**procedure** Start;

### **Description**

Begins playback of the sound contained in the Sound property.

## **TVrWave.Sound**

[TVrWave](#) [see also](#)

a special class derived from TPersistent.

**property** Sound: TVrAudioData;

### **Description**

TVrAudioData is a special class derived from TPersistent. It is used to hold the Audio data which is included in the resource of the application (no need for single on disk files). During runtime use LoadFromFile or SaveToFile to load and save wave files.

## **TVrWave.Stop**

[TVrWave](#) [see also](#)

Cancels the current sound playing.

**procedure** Stop;

### **Description**

Stop cancels the current sound playing.

## **TVrWebLabel**

[properties](#)[methods](#)

[events](#)

[see also](#)

Adds real internet functionality through the URL property.

### **Unit**

VrHyperCtrls

### **Description**

TVrWebLabel is derived from [TVrHyperLink](#) and inherits all its properties, methods and events.

TVrWebLabel adds real internet functionality through an URL property.

## **TVrWebLabel.OnError**

[TVrWebLabel](#) [see also](#)

Is called when an error occurred while executing the URL.

**property** OnError: TNotifyEvent;

### **Description**

The event OnError is called when an error occurred while executing the URL.

## **TVrWebLabel.URL**

[TVrWebLabel](#) [see also](#)

Is executed as soon as the label is clicked.

**property** URL: string;

### **Description**

The URL property is executed as soon as the label is clicked. All addresses and internet transport names are allowed if your current browser supports them.

eg.

<http://www.tmssoftware.com/>

or

[Mailto:info@tmssoftware.com](mailto:info@tmssoftware.com)

etc



## TVrWheel

[properties](#)[methods](#)

[events](#)

[see also](#)

TVrWheel is a rounded progress indicator or knob.

### Unit

vrwheel

### Description

TVrWheel is a rounded progress indicator or knob. TVrWheel is not scalable. TVrWheel needs bitmap images for it's background and handle to paint itself.

## **TVrWheel.BackImage**

[TVrWheel](#) [see also](#)

Used to define the knob image.

**property** BackImage: TBitmap;

### **Description**

BackImage is used to define a custom knob image. This could be any graphical image representing a wheel/knob. The transparentcolor of the bitmap itself is used to make it transparent.

## **TVrWheel.BaseAngle**

[TVrWheel](#) *see also*

Describes the starting angle from which the position of the handle image is calculated.

**property** BaseAngle: Integer;

### **Description**

BaseAngle describes the starting angle from which the position of the handle image is calculated.

## TVrWheel.HandleImage

[TVrWheel](#) [see also](#)

Used to define the handle image.

**property** HandleImage: TBitmap;

### Description

HandleImage is used to define a custom handle image. This could be any graphical image representing a handle. The transparentcolor of the bitmap itself is used to make it transparent.

## **TVrWheel.MaxValue**

[TVrWheel](#) *see also*

Use MaxValue to set a upper limit to the value that can be represented using the Wheel control.

**property** MaxValue: Integer;

### **Description**

Use MaxValue to set a upper limit to the value that can be represented using the Wheel control.

## **TVrWheel.MinValue**

[TVrWheel](#) *see also*

Use MinValue to set a lower limit to the value that can be represented using the Wheel control.

**property** MinValue: Integer;

### **Description**

Use MinValue to set a lower limit to the value that can be represented using the Wheel control.

## **TVrWheel.OnChange**

[TVrWheel](#) [see also](#)

OnChange event occurs when you assign another value to the position property.

**property** OnChange: TNotifyEvent;

### **Description**

OnChange event occurs when you assign another value to the position property.

## **TVrWheel.PercentDone**

[TVrWheel](#) [see also](#)

Returns the current position calculated against MinValue and MaxValue.

**property** PercentDone: Integer;

### **Description**

PercentDone returns the current position calculated against MinValue and MaxValue. This is a runtime only property.



## **TVrWheel.Position**

[TVrWheel](#) *see also*

Points to the current state.

**property** Position: Integer;

### **Description**

Position points to the current state. The position can never exceed the values defined by the MinValue and MaxValue properties.

## **TVrWheel.Radius**

[TVrWheel](#) [see also](#)

Used to position the Handle image.

**property** Radius: Integer;

### **Description**

Radius is used to position the Handle image.

## **TVrWheel.Transparent**

[TVrWheel](#) [see also](#)

Specifies whether other controls on a form can be seen through the background.

**property** Transparent: Boolean;

### **Description**

Set Transparent to True to prevent the control from obscuring other controls on the form. For example, if the control is put onto a graphic, set Transparent to True so that the control does not stand out as a separate object.

## How to contact us

Support - EMail [Info@tmssoftware.com](mailto:Info@tmssoftware.com)  
[see also](#)

Support:

[help@tmssoftware.com](mailto:help@tmssoftware.com)

Internet:

<http://www.tmssoftware.com>

## **Our Software Licence Agreement**

Support - EMail [Info@tmssoftware.com](mailto:Info@tmssoftware.com)  
[see also](#)

BEFORE PROCEEDING WITH THE INSTALLATION AND/OR USE OF THIS SOFTWARE, CAREFULLY READ THE FOLLOWING TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT AND LIMITED WARRANTY.

BY INSTALLING OR USING THIS SOFTWARE YOU INDICATE YOUR ACCEPTANCE OF THIS AGREEMENT. IF YOU DO NOT ACCEPT OR AGREE WITH THESE TERMS, YOU MAY NOT INSTALL OR USE THIS SOFTWARE!

TMS Software reserves the right as the sole distributor of the library source code. Hence although we encourage you to change and modify the library to suit your needs, you may not distribute derivative works based on the library source code without express written permission from TMS Software. Worthwhile changes and modifications to the libraries may be submitted to TMS Software for integration into a future release of the product.

### **LICENSE**

This software, including documentation, source code, object code and/or additional materials (the "Software") is owned by TMS Software. This Agreement does not provide you with title or ownership of Product, but only a right of limited use as outlined in this license agreement. TMS Software hereby grant you a non-exclusive, royalty free license to use the Software as set forth below:

- Integrate the software with your applications, subject to the redistribution terms below.
- Modify or adapt the software in whole or in part for the development of Applications based on the Software.
- Use portions of TMS Instrumentation Workshop source code or demo applications in your own products and libraries.

### **REDISTRIBUTION RIGHTS**

You are granted a non-exclusive, royalty-free right to reproduce and redistribute executable files created using the Software (the "Executable Code") in conjunction with software products that you develop and/or market (the "Applications").

### **RESTRICTIONS**

Without the expressed, written consent of TMS Software, you may NOT: distribute modified versions of the Software, in whole or in part. . rent or lease the Software. . sell any portion of the Software on its own, without integrating it into your Applications as Executable Code.

### **SELECTION AND USE**

You assume full responsibility for the selection of the Software to achieve your intended results and for the installation, use and results obtained from the Software.

### **LIMITED WARRANTY**

THIS SOFTWARE IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PRODUCT IS WITH YOU. SHOULD THE PRODUCT PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING OR ERROR CORRECTION. TMS SOFTWARE DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE SOFTWARE WILL MEET YOUR REQUIREMENTS OR THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR

ERROR FREE.

No oral or written information given by TMS Software authors shall create a warranty.

**LIMITATION OF REMEDIES AND LIABILITY.**

IN NO EVENT SHALL TMS SOFTWARE, OR ANY OTHER PARTY WHO MAY HAVE DISTRIBUTED THE SOFTWARE AS PERMITTED ABOVE, BE LIABLE FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE SOFTWARE (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR FAILURE OF THE SOFTWARE TO OPERATE WITH ANY OTHER PRODUCTS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## Ordering Information

Support - EMail [Info@tmssoftware.com](mailto:Info@tmssoftware.com)  
[see also](#)

### TMS Instrumentation Workshop Registration

#### **TMS Instrumentation Workshop 1.x Professional Edition \$75.00**

TMS Instrumentation Workshop comes packed with all of the great components and features. In addition, the disk is full of example programs and comes with complete documentation in the form of a Windows Help file. The professional edition comes with the complete source code of the TMS Instrumentation Workshop component library.

---

We have contracted another company, Shareit! Inc, to process any orders you may wish to place with your Visa, American Express, Eurocard/Mastercard or Diners Club. As soon as Shareit! proceeds the order and sends us the required information, we will provide you with a password and the url to our download page.

All prices are in US dollars. Registering entitles you to unlimited free support via E-mail, minor version updates and major version updates for 1 year from date of purchase. The right to use the components, royalty free, in any software product. You can also order by mail, phone or fax. Visit the Shareit website at [www.shareit.com](http://www.shareit.com) for more information.

If you would like to register TMS Instrumentation Workshop, you can do the registration online on the Internet at:

<http://www.tmssoftware.com/orders.htm>

**Please note that for each additional user a license is required.**

If you do not have access to the Internet, you can register via phone, fax or postal mail. Please print out the following form, and fax or mail it to:

Shareit!  
element 5 AG  
Vogelsanger Str. 78  
D-50823 Köln/Cologne  
Germany  
tel: +49 - 221 - 31088-20  
fax: +49 - 221 - 31088-29

US customers may also order by calling 1-800-903-4152 (orders only please!).  
US check and cash orders can be sent to our US office at

Shareit! Inc.  
PO Box 844  
Greensburg, PA 15601-0844  
USA  
tel: +1 (724) 850 - 8186  
fax: +1 (724) 850 - 8187  
(Please pay special attention to the country code!)

Registration form for TMS Instrumentation Workshop

Program No.: 154178

Number of licenses: \_\_\_\_\_

Last name: \_\_\_\_\_

First name: \_\_\_\_\_

Company: \_\_\_\_\_

Street and #: \_\_\_\_\_

City, State, postal code: \_\_\_\_\_

Country: \_\_\_\_\_

Phone: \_\_\_\_\_

Fax: \_\_\_\_\_

E-Mail: \_\_\_\_\_

How would you like to pay the registration fee:

credit card - wire transfer - EuroCheque - cash

Credit card information (if applicable)

Credit card: Visa - Eurocard/Mastercard - American Express - Diners Club

Card holder: \_\_\_\_\_

Card No.: \_\_\_\_\_

Date of Expiration : \_\_\_\_\_

Date / Signature \_\_\_\_\_



**Introduction**

**What's New**

**VCL Overview**

**Ordering Information**

**Our License Agreement**

[None]

OnAlarm

OnHoursChanged

OnMinutesChanged

OnSecondsChanged

Active  
AlarmTime  
EnableAlarm  
Glyph  
HandsColor  
HourMarks  
Hours  
Minutes  
SecHandColor  
Seconds  
SecondsIndicator  
Threaded  
TickColor  
TickOutline  
TickWidth  
Transparent

OnChange

Angle  
CenterDotWidth  
ColorZone3  
LabelsFont  
MinValue  
NeedleLength  
Position  
Ticks  
TicksMax

AngleOffset  
ColorZone1  
Decimals  
LabelsOffset  
NeedleBaseWidth  
Percent1  
Radius  
TicksColor  
TicksMin

CenterDotColor  
ColorZone2  
Labels  
MaxValue  
NeedleColor  
Percent2  
Spacing  
TicksEnlarge  
Transparent

AutoSize  
BitmapIndex  
BitmapList  
Down  
ImageCount  
Interval  
SwitchStyle  
Threaded  
Transparent

OnNotify



Active  
AutoSize  
Bitmap  
CurrentFrame  
FrameCols  
FrameRows  
Interval  
Loop  
Stretch  
Threaded  
Transparent

Active

Direction

Palette

TrackMouse

Transparent

OnScrollDone

AutoScroll  
Bevel  
Bitmap  
Direction  
PixelColor  
PixelMode  
PixelSize  
Spacing  
Threaded  
TimeInterval

BorderColor  
BorderWidth  
InnerColor  
InnerHighlight  
InnerOutline  
InnerShadow  
InnerSpace  
InnerStyle  
InnerWidth  
OuterColor  
OuterHighlight  
OuterOutline  
OuterShadow  
OuterSpace  
OuterStyle  
OuterWidth  
Visible

AutoSize

Glyph

HIndent

NumGlyphs

Transparent

TransparentMode

VIndent

OnChange

Toggle



AllowGrayed  
BitmapList  
Checked  
DisabledGlyphIndex  
EnabledGlyphIndex  
FocusColor  
FocusOffset  
Font3D  
Layout  
Margin  
NumGlyphs  
Spacing  
State  
TextureIndex  
TextureStyle  
TransparentColor

OnChange

AutoSize  
BitmapIndex  
BitmapList  
HideCursor  
ImageCount  
Increment  
MaxValue  
MinValue  
Position  
Style  
Transparent

AutoSize  
BitmapIndex  
BitmapList  
Center  
Stretch  
Transparent

OnChange

GetBitmap

## Bitmaps

OnChange



BitmapList  
Checked  
DisabledGlyphIndex  
EnabledGlyphIndex  
FocusColor  
FocusOffset  
Font3D  
Layout  
Margin  
NumGlyphs  
Spacing  
TextureIndex  
TextureStyle  
TransparentColor

Clear

Add

Delete

Exchange

IndexOf

Insert

LoadFromFile

LoadFromStream

Move

SaveToFile

SaveToStream

Bitmaps  
Count

Bevel  
BackImage

HighlightColor  
ShadowColor  
Shape  
Style

OnDraw

Reset

Alignment  
Bevel  
Columns  
Count  
Digits  
DrawStyle  
Grid  
Items  
ItemIndex  
NextStep  
Options  
Orientation  
Palette  
Rows



OnChange

CheckWidth  
CheckHeight  
Checked  
CheckStyle  
Layout  
Margin  
Palette  
Spacing

OnHoursChanged  
OnMinutesChanged  
OnSecondsChanged

Active  
AutoSize  
Blink  
ClockType  
Hours  
Hours24  
Minutes  
Seconds  
ShowTimeZone  
Style  
Palette  
ShowSeconds  
Threaded  
Transparent

OnChange

AutoSize  
BackImage  
CircleColor  
CircleOutlineColor  
CircleOutlineWidth  
CircleWidth  
Heading  
NeedleColor  
NeedleLength  
NeedleTransparent  
NeedleWidth  
Transparent

AfterCopy  
BeforeOpen  
BeforeOverwrite  
OnProgress

Execute  
Terminate



BufferSize  
CopyDateTime  
DestFile  
SourceFile  
Overwrite

OnChange

AutoSize  
Bitmap  
Digits  
Spacing  
Stretch  
Transparent  
Value

BevelWidth  
Bitmap  
DisabledTextColor  
Flat  
FocusColor  
Font3D  
FontEnter  
FontLeave  
HighlightColor  
OutlineColor  
OutlineWidth  
ShadowColor  
TextAlignment

FormDrag  
Glyph

OnChange

ActiveOnly  
OutlineColor  
Palette  
Transparent  
Value  
ValueBinary

OnLocate  
OnNotify  
OnPathChange



Cancel  
Execute

Attributes  
Mask  
Path  
Recursive

Bevel

ShadowColor1

ShadowColor2

ShadowLayout

ShadowWidth

HighlightColor  
HighlightDepth  
ShadowColor  
ShadowDepth  
Style

Mask

MaskColor

OnChange  
OnMaxValue  
OnMinValue

PercentDone  
StepBy  
StepIt

ActiveClick  
Bevel  
MinValue  
MaxValue  
Palette  
Position  
Orientation  
SolidFill  
Spacing  
Step  
Style  
TickHeight



ColorWidth

Direction

EndColor

FormDrag

StartColor

SwapColors

OnFontChanged  
OnMouseEnter  
OnMouseLeave

ColorEnter  
ColorLeave  
DrawStyle  
HotRect  
ImageEnter  
ImageLeave  
TextAlignment

Color  
Visible  
Width

OnMouseEnter  
OnMouseLeave

BorderColor  
BorderHighlight  
BorderShadow  
ColorEnter  
ColorLeave  
DisabledAnimate  
DisabledText  
Glyph  
Layout  
Margin  
Spacing  
Transparent

OnMouseEnter  
OnMouseLeave

Alignment  
ColorEnter  
ColorLeave  
FontEnter  
FontLeave  
TextOutline  
Transparent  
WordWrap



OnChange

Active

Blink

ImageType

Inverted

Palette

TimeInterval

Transparent

OnChange

StepBy  
StepIt

ColorWidth  
Step  
MaxValue  
MinValue  
LedStyle1  
LedStyle2  
LedStyle3  
Percent1  
Percent2  
PercentDone  
Position  
Spacing  
Transparent

Direction

Palette

Spacing

Transparent

VisibleArrows

OnChange

Keys  
MonitorEvents



Alignment

Angle

AutoSize

Bitmap

ColorHighlight

ColorShadow

Layout

ShadowDepth

Style

Transparent

OnChange

Active  
Glyphs  
Layout  
LedType  
Margin  
Palette  
Spacing  
Transparent

OnChange  
OnMaxValue  
OnMinValue

StepBy  
StepIt

Bevel  
MinValue  
MaxValue  
Orientation  
Palette1  
Palette2  
Palette3  
Percent1  
Percent2  
Position  
Spacing  
Step  
Style  
TickHeight

OnChange

LedState  
LedsVisible  
LedType  
Order  
Orientation  
Spacing  
Transparent



OnScrollDone

Alignment  
AutoScroll  
Bevel  
Leds  
LedStyle  
LedsVisible  
Orientation  
Palette  
ScrollDirection  
Spacing  
Threaded  
Text  
TextStyle  
TimeInterval

ScrollText

Reset

Alignment  
AutoScroll  
Bevel  
CharSpacing  
Cols  
Lines  
LineSpacing  
Palette  
PixelSize  
PixelSpacing  
Rows  
ScrollDirection  
TextStyle  
TimeInterval  
Threaded

BorderColor  
ButtonType  
FocusColor

Angle  
BackImage  
Bevel  
Labels  
LabelOffsetX  
LabelOffsetY  
MinValue  
MaxValue  
NeedleColor  
NeedleWidth  
Position  
Scale  
Spacing

OnChange



OnButtonClick

ButtonIndex

Bevel  
BorderColor  
EnabledButtons  
FocusColor  
Numeric  
Orientation  
Spacing  
VisibleButtons

OnChange

Alignment  
AutoSize  
Digits  
LeadingZero  
Min  
Max  
Palette  
Spacing  
Style  
Transparent  
Value  
ZeroBlank

OnError  
OnMouseEnter  
OnMouseLeave

Alignment  
Decimals  
MaxValue  
MinValue  
RestoreOnEsc  
Value

High  
Low



OnChange

OutlineColor  
Orientation  
Segments

OnChange

OutlineColor  
Segments  
Transparent

Depth

FreeColor

FreeShadowColor

MaxValue

OutlineColor

Transparent

UsedColor

UsedShadowColor

Value

Active  
BevelWidth  
FocusColor  
HighlightColor  
Layout  
LedHeight  
LedWidth  
Margin  
OutlineColor  
OutlineWidth  
Palette  
ShadowColor  
Spacing

OnChange

BarColor  
BarWidth  
BottomOffset  
ColorHighlight  
ColorShadow  
Decimals  
FillColor  
Increment  
MarkerImage  
MarkerVisible  
MaxValue  
MinValue  
ScaleVisible  
TitleAlignment  
TitleFont  
TopOffset  
Transparent  
UnitsText  
Value



StepBy  
StepIt

Bevel  
Bitmap  
EndColor  
FillType  
MinValue  
MaxValue  
Orientation  
PercentDone  
Position  
Smooth  
StartColor  
Step

OnChange

Bevel

Columns

Count

Items

MultiSelect

Palette

PlainColors

Rows

Spacing

Style

OnLowerClick  
OnUpperClick

Depth  
FocusColor  
HighlightColor  
OutlineColor  
OutlineWidth  
ShadowColor  
ShadowLightColor  
State  
Style

OnExists

MessageText  
RestorePrevInst  
ShowMessage  
Terminate



Alignment  
Layout  
MinValue  
Orientation  
PeakLevel  
ScaleOffset  
Ticks  
TicksWidth  
Transparent

Digits  
LeadingZero  
MaxValue  
PeakColor  
ScaleColor  
ShowSign  
TicksHeight  
TickMarks

OnChange

Active  
ColorWidth  
Direction  
Leds  
LedStyle  
Palette  
Position  
Spacing  
Threaded  
TimeInterval  
Transparent

OnNeedData

Clear

Active  
AnimateGrid  
BaseColor  
BaseOffset  
BufferSize  
Channels  
Frequency  
GridColor  
GridLineWidth  
GridSize  
Interval  
MaxValue  
MinValue  
Threaded

Depth  
DepthAdjust  
Direction  
DisabledTextColor  
ShadowColor  
ShadowOutline  
Style  
TextAlign  
Transparent

Bitmap  
Transparent



OnChange

Bevel  
BitmapList  
BorderColor  
BorderWidth  
FocusColor  
KeyIncrement  
MaxValue  
MinValue  
Options  
Orientation  
Position  
SolidFill  
Spacing  
Style  
ThumbImageIndex  
ThumbIndent  
ThumbStates  
TickWidth

OnNextSlide  
OnNotify

Active  
BitmapList  
ImageIndex1  
ImageIndex2  
Interval  
Loop  
Steps  
Threaded  
Transition

Reset

BarSpacing  
BarWidth  
Columns  
Count  
Items  
MarkerColor  
MarkerVisible  
MaxValue  
MinValue  
Palette1  
Palette2  
Palette3  
Percent1  
Percent2  
PlainColors  
ShowInactive  
Spacing  
TickHeight

OnDownClick  
OnUpClick

FocusControl  
Orientation  
Palette



OnMouseEnter  
OnMouseLeave

Alignment  
ValidKeys

OnChange  
OnChangeing

Sorted

OnChange

BackImageIndex  
Bevel  
BitmapList  
BorderColor  
BorderWidth  
FocusColor  
Offset  
Options  
Orientation  
Positions  
Style  
ThumbImageIndex  
ThumbIndent  
ThumbStates

Color  
Visible

OnExecute



Enabled  
Priority  
SyncEvent

OnTimer

Enabled  
Interval  
Priority  
SyncEvent  
TimerType

OnChange

BackImageIndex  
BitmapList  
BorderWidth  
FocusColor  
FocusOffset  
Frequency  
GutterBevel  
GutterWidth  
GutterColor  
Orientation  
MinValue  
MaxValue  
Options  
Position  
ScaleOffset  
Style  
ThumbImageIndex  
ThumbStates  
TickMarks  
TickColor

OnChange  
OnRightClick

Enabled  
HideTaskBtn  
Min  
Max  
Palette  
Position  
Style  
Visible

OnRightClick



Enabled  
HideTaskBtn  
Icon  
Visible

OnUpClick  
OnDownClick

GlyphsDown  
GlyphsUp  
NumGlyphs  
Orientation  
RepeatClick  
RepeatPause  
Transparent

OnChange  
OnDraw

Active

Bevel

DrawStyle

OutlineColor

OutlineWidth

Palette

Play

Stop

Options  
Sound

OnError



URL

OnChange

BackImage  
BaseAngle  
HandleImage  
MaxValue  
MinValue  
PercentDone  
Position  
Radius  
Transparent

## VCL Overview

### TMS Instrumentation Workshop

#### [ A ]

[TVrAnalogClock](#)  
[TVrAniButton](#)  
[TVrAnimate](#)  
[TVrArrow](#)  
[TVrAngularMeter](#)

#### [ B ]

[TVrBanner](#)  
[TVrBitmapButton](#)  
[TVrBitmapCheckBox](#)  
[TVrBitmapDial](#)  
[TVrBitmapImage](#)  
[TVrBitmapList](#)  
[TVrBitmapRadioButton](#)  
[TVrBlotter](#)  
[TVrBorder](#)

#### [ C ]

[TVrCalendar](#)  
[TVrCheckLed](#)  
[TVrClock](#)  
[TVrCompass](#)  
[TVrCopyFile](#)  
[TVrCounter](#)

#### [ D ]

[TVrDemoButton](#)  
[TVrDeskTop](#)  
[TVrDigit](#)  
[TVrDirScan](#)  
[TVrDisplay](#)

#### [ F ]

[TVrFormShape](#)

#### [ G ]

[TVrGauge](#)  
[TVrGradient](#)

#### [ H ]

[TVrHotImage](#)  
[TVrHyperButton](#)  
[TVrHyperLink](#)

#### [ I ]

[TVrImageLed](#)  
[TVrIndicator](#)

#### [ J ]

[TVrJoyPad](#)

#### [ K ]

TVrKeyStatus

**[ L ]**

TVrLabel

TVrLed

TVrLevelBar

TVrLights

**[ M ]**

TVrMatrix

TVrMatrixGroup

TVrMediaButton

TVrMeter

**[ N ]**

TVrNavigator

TVrNum

TVrNumEdit

**[ P ]**

**TVrPieGraph**

**TVrPowerButton**

TVrPowerMeter

TVrProgressBar

TVrPercentBar

TVrPercentPie

**[ R ]**

TVrRaster

TVrRocker

TVrRunOnce

**[ S ]**

TVrScale

TVrScanner

TVrScope

TVrShadowButton

TVrShapeBtn

TVrSlider

TVrSlideShow

TVrSpectrum

TVrSpinner

TVrStrEdit

TVrStringList

TVrSwitch

**[ T ]**

TVrThread

TVrTrackBar

TVrTrayGauge

TVrTrayIcon

**[ U ]**

TVrUpDown

TVrUserLed

**[ W ]**

TVrWave

TVrWebLabel

TVrWheel

