Welcome to **"ZieglerCollection one"** - one of the best Delphi component collections
Version 1.20 ©1996-97 ZieglerSoft
Helpfile version 0.90 ©1996-97 ZieglerSoft

Contents in this helpfile

This helpfile is version 0.90 for "ZieglerCollection one". The latest helpfile can, at any time, be downloaded from our web-pages at (http://www.zieglersoft.dk). The helpfile will be updated more often than the "ZieglerCollection one" itself.

Remember: ZieglerSoft is making tailor-made components.
If You have a wish, please contact ZieglerSoft

Components

Included in ZieglerCollection one is the following components:

TzMinMax	A component for very easy control of the form-size.
TzBigLabel	A Label-component that can hold more data than the build-in version (Delphi 1). In Delphi 2, 3 and C++Builder the TLabel component can be used to hold large texts, but to make things the same in all versions, we use TzBigLabel as the "mother" component of many of our Label-components.
Tz3DLabel	A TzBigLabel descendant, that can show its caption in 3D (Raised or lowered).
TzAngleLabel	A TLabel descendant, that can show its caption in any angle You want (Works only if the caption is displayed in True-Type fonts). It is NOT a TzBigLabel descendant.
TzTabListBox	An extended tListBox, where You can used tabs to align the displayed text. It has what the tListBox needs: an OnChange event.
TzBitmap	A bitmap, where You can decide what color is transparent (You can see the background-color through). Now has a "AutoBackColor" property where it uses Delphi's own way to get the transparent color (Left, bottom of bitmap).
TzAnimated	A TzBitmap descendant, which can be used to play small cartoons (no sound).
TzBackground	A background-component, that can tile a bitmap, or draw a color-blend. This component works on all Form-types, and only on forms. Use TzBlendPaint or TzTileMap if You need to color a part of a component, or a component that is not a form. This component will not use any Window-resources permanent while showing. This component has an extra option, in which it can be used to draw anything on the surface of a MDIForm, by using the OnPaint event, that supplies a Canvas, just for that.
TzBlendPaint	A backdrop that can be started and stopped on any color, and will make a gradient fill between the two colors. It works in the same way as a tPanel (it is a windowed control).
TzTileMap	A backdrop that can have any bitmap tiled to fill the component complete. It works in the same way as a tPanel (it is a windowed control).
TzLed	A led component, that show any of eight colors, can be on or off, or can blink (If You use blink, it will use one windows-timer pr. TzLed -component, so it will be wise to use one TzLed to control the rest, if they are supposed to blink at the same time).
TzSegment	A 16-part led-segment, that can show most characters. All the characters that it can show can be customized (All Segment-components will change at the same time).
TzSegmentLabel	A label-like component, that uses TzSegments to display the caption. This component is a child of a standard-component, called TzCustomSegmentLabel, that can be used as mother to other label-like components.
	A label-like clock-component, that uses TzSegments to show the time. It can show a clock with or without seconds. Every time it is updated (once per

TzSegmentClock	second if seconds are shown, else once per minute) it can call an event-handler, and thereby be used to control other components. The component is a child of TzCustomSegmentLabel.
TzGauge	A component that can be used to show progress in any operation. Can use a glyph or a colorbar to show the progress. Can be displayed in a lot of different styles.
TzSlideBar	A component that can be used anywhere You want to be able to scroll through a range of numbers. One use could be as a volume-control. It can take a large number of shapes.
TzFrame	A component that can be used to make frames around other components, or on its own just like the tBevel component found in the VCL-component collection. The big difference is the control You have over the component.
TzDivider	The most simple component in the whole collection: A divider-line, that can be horizontal or vertical. It is used to separate groups of other components on forms (But can be used for other things too, like dropzone for drag-drop components, because You can set it to accept drops below the line itself, or to the right of it).
TzMovePanel	A tPanel component, that can be moved around with the mouse, useful for building floating toolbars.
TzTitleBar	A smart component, useful if You want to set and remove the titlebar on a form, while Your program is running. It has one more function, and that's to make sure You can move Your form, even when no caption exists.
TzHint	A component, that make it a lot easier to manipulate the hint-window. It also allows You to have multiline hints. Easy change of the font used to write the hint-text.
TzShowApp	A component, that makes it a lot easier to control Your tApplication component. You can even assign event-handlers at design-time. Includes a lot of information about the environment too.
TzVerSpilt	A component that makes it easy to make a vertical splitter-window. It is possible to have any number of splitter-component on a form.
TzHorSplit	A component that makes it easy to make a horizontal splitter-window. It is possible to have any number of splitter-component on a form.
TzMouseSpot	A component, useful for adding mouseevnts to pictures, maps and drawings. It don't use a window-handle, so You use as many TzMouseSpots on a single picture as You want. It also makes it easy to prevent mouseevents at some parts of the picture in some situations, and in other parts in other situations.
TzCalc	A non-visible component, used to do calculations. You pass it a string with the calculation, and this component will try to solve it for You.
TzShapeBtn	A non-windowed button-component, that does not receive focus. It can take almost any shape, by using the build-in types, or by using the owner-draw option. It will only be pressed when the mouse is in the shape of the button, not when the mouse is in the rectangle that surrounds the button.
TzColorBtn	Nearly a standard Tbutton, with one very big exception, it has the possibility of changing colors, the face-color, highlight-color and shadow-color. Can have a caption with more than one line of text too.
TzGradBtn	A TzColorBtn with a little extra. Will paint a gradient fill from Color to Endcolor on the surface of the button instead of just one color as the TzColorBtn. It can be used instead of a TzColorBtn, because it will be the same if both colors are set to the same value.

TzBitColBtn	A TzColorBtn with a little extra (Can include a glyph-bitmap on the button surface). If the Glyph includes more than one picture (it can hold from 1-4) the first will be used for the normal Picture, the second will be the disabled picture and the third is used for down. The fourth is not used but is accepted for compatibility reasons. It is not fully compatible with the standard tBitBtn. You decide which color is to be used for the transparent parts of the bitmap, it will not do it automatic as the tBitBtn does, because we think You may need more control over it. Has a "AutoBackColor" property where it uses Delphi's own way to get the transparent color.
TzIconColBtn	A TzColorBtn with a little extra (Can include 3 icons for Enabled/disabled/down button).
TzScope	A simple way to display a lot of data visible. Can show one or two graphs, looking just like an oscilloscope. Can set all the data at once, can put one value in at a time and can automatically push all the data already on the scope forward when a new value comes in.
TzPanelMeter	A component used to display values in a way that looks like Panel meters. This is a very simple component, but it is very useful to give quick and dirty displays of values, easy to read. Can show in many different shapes, so it should be possible to use it in nearly any type of program.
TzKnob	A component that looks and works like a dial (knob) on a stereo. Very easy to use. Is a windowed control, and can have focus, and use the keyboard for adjusting the value.
TzDblKnob	A TzKnob component. The difference is that this component can be used to select two values in the space of one component.
TzTripKnob	A TzKnob component. The difference is that this component can be used to select three values in the space of one component.
TzTrayIcon	A component for easy handling of programs, that need to show itself in the tray (Windows'95 and NT 4.x and later only). In cooperation with TzShowApp, it can even remove the program from the normal statusbar, and make sure the main-window is not shown.
TzNWColorBtn	A Non-windowed button. It functions in most ways as a TzColorBtn. Can show a flat and/or a transparent look if wanted. Can of course not have the focus.
TzNWBitColBtn	A Non-windowed button. It functions in most ways as a TzBitColBtn. Can show a flat and/or a transparent look if wanted. Can of course not have the focus.
TzNWIconColBtn	A Non-windowed button. It functions in most ways as a TzIconColBtn. Can show a flat and/or a transparent look if wanted. Can of course not have the focus.
TzNWBlendPaint	A backdrop that can be started and stopped on any color, and will make a gradient fill between the two colors it is a non-windowed control. It works in most ways like TzBlendPaint.
TzNWTileMap	A backdrop that can be started and stopped on any color, and will make a gradient fill between the two colors it is a non-windowed control. It works in most ways like TzTileMap.
TzResBitmap	A tBitmap descendant, that will load its glyph from the resourcefile. It is not a component with an entry in the Component-Palette, but can be created at runtime when needed. It is used in some of the "ZieglerCollection one"'s components.

TzDeskTop	A tCanvas descendant, that holds the complete desktop. This makes it very easy to write directly on the desktop. Just create a TzDesktop, and use this for drawing. Also useful for reading everything on the desktop.

Functions and procedures

Included in ZieglerCollection one is the following functions and procedures:

Procedure BMPRotate(Pic1,Pic2:tBitmap;Angle:Integer);
Unit ANIMATE.PAS
Takes the picture in Pic1, rotates it by Angle degrees, and return it in Pic2.

Procedure DarkenBMP(Pic:TBitmap;Percent:TPercentType;SaveBack:Boolean;BColor:tColor);
Unit ANIMATE.PAS
Takes the picture in Pic, makes it Percent percents darker, returns the result in Pic. If SaveBack is TRUE, it will not touch anything in the BColor color.

Procedure LightenBMP(Pic:TBitmap;Percent:TPercentType;SaveBack:Boolean;BColor:tColor);
Unit ANIMATE.PAS
Takes the picture in Pic, makes it Percent percents lighter, returns the result in Pic. If SaveBack is TRUE, it will not touch anything in the BColor color.

Procedure GreyBMP(Pic:TBitmap;SaveBack:Boolean;BColor:tColor);
Unit ANIMATE.PAS
Takes the picture in Pic, and greyscale it. If SaveBack is TRUE, it will not touch anything in the BColor color.

Procedure TransparentBlt(Dest:TCanvas;Bmp:TBitmap;X,Y:Integer;TransColor:TColor);
Unit ANIMATE.PAS
Makes a real transparent blit of the picture in Bmp onto Dest at position X,Y. The color given in TransColor will be removed, and whatever is below parts painted with that color will show trough.

Function FindForm(ThisComponent:tComponent):tForm;
Unit MYSTD.PAS
Given a component, ThisComponent, this function will return the Form the component is placed on. It will go the full way back, to find the form.

Function IsPrevius(GoToPrevius:Boolean):Boolean;
Unit MYSTD.PAS
If GoToPrevius is true and this returns true, then the previous version IS started, and You must end this one, WITHOUT doing anything more. If You only want to test if a previous version is running, then set GoToPrevius to false before calling this function. If You want to start a normal Delphi program, only once, then the mainprogram will look something like this:

```
If Not(IsPrevius(True)) then Begin
  Application.CreateForm(TZieglerSetupForm, ZieglerSetupForm);
  Application.Run;
End;
```

Function DlgUnitsToPixelsX(DlgUnits: word): word;
Unit MYSTD.PAS
Translate Dialogunits for X axis to Pixels.

Function DlgUnitsToPixelsY(DlgUnits: word): word;
Unit MYSTD.PAS
Translate Dialogunits for Y axis to Pixels.

Function PixelsToDlgUnitsX(PixUnits: word): word;
Unit MYSTD.PAS
Translates pixels to Dialogunits for X axis.

Function PixelsToDlgUnitsY(PixUnits: word): word;
Unit MYSTD.PAS
Translates pixels to Dialogunits for Y axis.

Function CpuID:Integer;
Unit MYSTD.PAS
Returns CPU type for this machine.

Function ArcSin(Nummer:Extended):Extended;
Unit MYSTD.PAS
Returns the ArcSin to the number Nummer.

Function ArcCos(Nummer:Extended):Extended;
Unit MYSTD.PAS
Returns the ArcCos to the number Nummer.

Function Log10(Nummer:Extended):Extended;
Unit MYSTD.PAS
Returns the Log10 number to the number Nummer.

Function Power(Nummer,Eksponent:Extended):Extended;
Unit MYSTD.PAS
Returns the number Nummer lifted to the power of Eksponent.

Function Factorial(Nummer:Integer):Extended;
Unit MYSTD.PAS
Returns the Factorial number to the number Nummer.

Function IsPrime(Nummer:Integer):Boolean;
Unit MYSTD.PAS
Returns TRUE if the number Nummer is a primenumber.

Function Root(x,y:Extended):Extended;
Unit MYSTD.PAS
Returns the root:

$$\sqrt[x]{y}$$

Function WindowsType:WTypes;
Unit ZHELPER.PAS
Returns the main type of windows this machine is running.

Function MajorVersion:LongInt;
Unit ZHELPER.PAS
Returns the major number of the windowsversion running.

Function MinorVersion:LongInt;
Unit ZHELPER.PAS
Returns the minor number of the windowsversion running.

Function BuildVersion:LongInt;
Unit ZHELPER.PAS
Returns the build number of the windowsversion running.

Function IsWorkgroup:Boolean;
Unit ZHELPER.PAS
Returns true if we are running on Windows for WorkGroups.

Function Is311:Boolean;
Unit ZHELPER.PAS
Returns true if Windows version is 3.11.

Procedure ChangeACharLook(TheChar:zChar;TheLook:zSet);
Unit ZSEG.PAS
This will change ALL chars used in ZSEG components at once. Given the char You want to change in TheChar, and a set, telling which parts of the 16-segmented display are to be lid when the char is shown in TheLook, will change the look, not only new ZSEG's, men all shown ZSEG's on screen.

Files

In which file is the wanted object, component, procedure or function? And what is in the rest of the files in ZieglerCollection one?

NOSALE.INC	Only used to make it easier for ZieglerSoft to control the demoversions of "ZieglerCollection one".
ZS_VCL32.PAS ZS_VCL32.DCR	This file is used to register all components in all 32-bit Delphi and C++ Builder versions. It is not used in Delphi 1.
ZS_VCL.PAS ZS_VCL.DCR	This file is used to register all components in 16-bit Delphi. It is not used in 32-bit Delphi and C++ Builder versions.
REG.FIL	Common parts of ZS_VCL.PAS and ZS_VCL32.PAS.
ZEXP1.PAS ZEXP1.DFM	Parts of custom expert.
ZANGEDIT.PAS ZANGEDIT.DFM	Parts of custom property-editor.
ZBAREDIT.PAS ZBAREDIT.DFM	Parts of custom property-editor.
ZBITEDIT.PAS ZBITEDIT.DFM	Parts of custom property-editor.
ZBTNEDIT.PAS ZBTNEDIT.DFM	Parts of custom property-editor.
ZHNTEDIT.PAS ZHNTEDIT.DFM	Parts of custom property-editor.
ZIEGLERCOLLECTION.DPK .DPL .DCP	The "ZieglerCollection one" as a Delphi 3 package, designtime only. .DPK is the sourcecode, used to build the two others.
ZCOL.HLP	This helpfile (It will be expanded in future versions of "ZieglerCollection").
BLEND.PAS	One of the source-files, making up the "ZieglerCollection one".
ANIMATE.PAS	One of the source-files, making up the "ZieglerCollection one".
SLIDEBAR.PAS	One of the source-files, making up the "ZieglerCollection one".
ZGAUGE.PAS	One of the source-files, making up the "ZieglerCollection one".
ZLED.PAS	One of the source-files, making up the "ZieglerCollection one".
MYSTD.PAS	One of the source-files, making up the "ZieglerCollection one".
ZSPLIT.PAS	One of the source-files, making up the "ZieglerCollection one".
ZHELPER.PAS	One of the source-files, making up the "ZieglerCollection one".
STD2.PAS	One of the source-files, making up the "ZieglerCollection one".
ZPANEL.PAS	One of the source-files, making up the "ZieglerCollection one".
ZSEG.PAS	One of the source-files, making up the "ZieglerCollection one".

TzMinMax

Unit STD2.PAS

This component makes it very easy to control the size of a form. Drop it on the form You want to control and set the properties as wanted.

property AutoTaskBarAjust
If set to TRUE, all resizing of the form will take the taskbar (Windows95 and NT) into account. If FALSE, it don't matter where the taskbar is located. This property works in Delphi 1 too.

property MaxDragHeight
Set it to the maximum height the user can resize the form to, when resizing with the mouse.

property MaxDragWidth
Set it to the maximum width the user can resize the form to, when resizing with the mouse.

property MaxHeight
Set it to the maximum height the form can be.

property Maxleft
Set it to the position where the left edge of the form must be when maximizing the form.

property MaxTop
Set it to the position where the top edge of the form must be when maximizing the form.

property MaxWidth
Set it to the maximum width the form can be.

property UseDefaultSize
If TRUE, the form will use the normal Windows default maxiumsize, minimumsize and so on. If FALSE, the values out in the other properties will be used.

property MinDragWidth
Set it to the minimum width the user can resize the form to.

property MinDragHeight
Set it to the minimum height the user can resize the form to.

event OnSizeChange
Called when the form is resized.

TzBigLabel

Unit MYSTD.PAS

A Label-component that can hold more data than the build-in version (Delphi 1). In Delphi 2, 3 and C++Builder the TLabel component can be used to hold large texts, but to make things the same in all versions, we use TzBigLabel as the "mother" component of many of our Label-components.

Use SetTextBuf and GetTextBuf to set and get the text. For text longer than 256 chars, You will need to set WordWrap to TRUE.

event OnMouseEnter
Called when the mouse enters the TzBigLabel control.

event OnMouseLeave
Called when the mouse leaves the TzBigLabel control.

Tz3DLabel

Unit MYSTD.PAS

A TzBigLabel descendant, that can show its caption in 3D (Raised or lowered).

property HighlightColor
Set it to the color You want the highlighted part of the caption to be.

property ShadowColor
Set it to the color You want the shadowed part of the caption to be.

property Raised
Set to TRUE if You want the caption to be raised above the background, to FALSE if You want it to be lower than the background.

property Use3D
TRUE turns the 3D on, FALSE turns it off.

TzAngleLabel

Unit MYSTD.PAS

A TLabel descendant, that can show its caption in any angle You want (Works only if the caption is displayed in True-Type fonts). It is NOT a TzBigLabel descendant.

property HighlightColor
Set it to the color You want the highlighted part of the caption to be.

property ShadowColor
Set it to the color You want the shadowed part of the caption to be.

property Raised
Set to TRUE if You want the caption to be raised above the background, to FALSE if You want it to be lower than the background.

property Use3D
TRUE turns the 3D on, FALSE turns it off.

property Angle
Set it to the wanted angle (It only works if the font used to display the label is a TRUE-TYPE font).

event OnPaint
Called every time the control is repainted.

TzTabListBox

Unit MYSTD.PAS

An extended tListBox, where You can used tabs to align the displayed text. It has what the tListBox needs: an OnChange event.

procedure SetTabStops(a:array of words)
Pass this procedure an array of words, in order from smallest to largest, of the positions in pixels, where You want the tabstops to be.

procedure SetFromHeader(a:tHeader)
Pass this procedure a tHeader control, and the TzTabListBox will set its tabstops so that they matches the dividers in the tHeader.

property SizeAfterdel
Set to TRUE, and the control will reset itself everytime a line is deleted, set to FALSE, it will keep its tabstops even when lines are deleted.

event OnChange
Called everytime the user selects a new line in the TzTabListBox.

event OnScroll
Called everytime the use has to use the scrollbars on the TzTabListBox.

event OnMouseEnter
Called when the mouse enters the TzTabListBox control.

event OnMouseLeave
Called when the mouse leaves the TzTabListBox control.

TzBitmap

Unit ANIMATE.PAS

A bitmap, where You can decide what color is transparent (You can see the background-color through). Now has a "AutoBackColor" property where it uses Delphi's own way to get the transparent color (Left, bottom of bitmap).

procedure Rotate(degrees)
Rotate the picture a number of degrees.

procedure Darken(SaveBack:Boolean;Percent:TPercentType)
Darken the picture a number of percents. If SaveBack is TRUE, the background is not touched.

procedure Lighten(SaveBack:Boolean;Percent:TPercentType)
Lighten the picture a number of percents. If SaveBack is TRUE, the background is not touched.

procedure GreyScale(SaveBack:Boolean)
Grayscale the picture. If SaveBack is TRUE, the background is not touched.

procedure BW
Make the picture Black & White.

property AutoBackColor
If Set to TRUE, then the background will be chosen in the same way as Delphi uses. If FALSE, You can set Your own backgroundcolor by using TransparentColor.

property RealTransparant
If Set to true, the drawing of the picture will be a lot slower, but anything in the background will shine through, where the backgroundcolor is on the picture. If FALSE, only the formcolor behind the picture is used in place of the backgroundcolor.

property Bitmap
Set it to the wanted bitmap.

property TransparentColor
If AutoBackColor is FALSE, set this to the color You want to use as background.

event OnPaint
Called every time the picture is repainted.

event OnMouseEnter
Called when the mouse enters the TzBitmap control.

event OnMouseLeave
Called when the mouse leaves the TzBitmap control.

TzAnimated

Unit ANIMATE.PAS

A TzBitmap descendant, which can be used to play small cartoons (no sound).

property Bitmap
Set it to the wanted bitmap. The bitmap is a long string of small bitmaps, pasted together to one. The small bitmaps has to be exactly the same size, so that the final bitmap is devided into a number of equally smaller bitmaps. Take a note on how many small bitmaps is used. You need that number in FrameCount.

property Interval
How many milliseconds between every frame in the cartoon?

property FrameCount
Set this to the number of frames in the cartoon. Remember to set the Width-property to the width of the bitmap divided by the number of frames in the cartoon.

property Frame
What frame is shown now?

property Play
Set to TRUE, and the cartoon is playing, FALSE and the cartoon is stopped.

property Reverse
Set to TRUE, and the cartoon is playing from the last frame to the first, Set to FALSE, and the cartoon is playing from the first to last frame.

property Loop
Set to TRUE, and the cartoon will keep on playing from the beginning every time it is through. Set To FALSE, and the cartoon will only play once.

event OnChangeFrame
Is called every time a new frame is about to be displayed.

TzBackGround

Unit BLEND.PAS

A background-component, that can tile a bitmap, or draw a color-blend.
This component works on all Form-types, and only on forms. Use TzBlendPaint or TzTileMap if You need to color a part of a component, or a component that is not a form. This component will not use any Window-resources permanent while showing.
This component has an extra option, in which it can be used to draw anything on the surface of a MDIForm, by using the OnPaint event, that supplies a Canvas, just for that.

property BackType
Decides how to paint the background. Choose between the following: btBitmap: for a bitmap that is tiled on the complete background of the form, btBlend: for at blend paint (the style of this can be set in Color, EndColor, Bands and FillType), btNormal: for a normal background (use this if You want to draw on the background yourself, using the OnPaint event) and btSteel: which gives a gray steel look on the background.

property FillType
Decides how a fill will look if the property BackType is set to btBlend. Choose between gHorz: if You want the fill to be of the horizontal version, gVert: if You want the fill to be of the vertical version and gCenter: if the fill should be of the rectangled version.

property Bitmap
If BackType is set to btBitmap, this property decides which bitmap to use for the background.

property Color
If Backtype is set to btBlend, this decides the starting color of the blend-fill.

property EndColor
If BackType is set to btBlend, this decides the ending color of the blend-fill.

property Bands
If Backtype is set to btBlend, this decides how many bands to split the blend-fill into.

event OnPaint
Use it to draw Your own things on the background. Called every time the background is updated.

event OnResize
Called every time the form this background is on, is resized.

TzBlendPaint

Unit BLEND.PAS

A backdrop that can be started and stopped on any color, and will make a gradient fill between the two colors. It works in the same way as a tPanel (it is a windowed control).

propert SteelLook
If set to TRUE, the control will fill with gray steel-look, if FALSE this control will fill as set in FillType.

property FillType
Choose between gHorz: if You want the fill to be of the horizontal version, gVert: if You want the fill to be of the vertical version and gCenter: if the fill should be of the rectangled version.

property Color
This decides the starting color of the blend-fill.

property EndColor
This decides the ending color of the blend-fill.

property Bands
This decides how many bands to split the blend-fill into.

event OnPaint
Called every time the control is repainted.

TzTileMap

Unit BLEND.PAS

A backdrop that can have any bitmap tiled to fill the component complete. It works in the same way as a tPanel (it is a windowed control).

property Bitmap
Set the bitmap that You want to show.

event OnPaint
Called every time the control is repainted.

TzLed

Unit ZLED.PAS

A led component, that show any of eight colors, can be on or off, or can blink (If You use blink, it will use one windows-timer pr. TzLed -component, so it will be wise to use one TzLed to control the rest, if they are supposed to blink at the same time).

property SteelLook
If TRUE, and LedStyle is set to lstRound, the remaining space around the led is painted in a light steel-look. If FALSE it is drawn in the background-color.

property LedType
Decides if the led is round or square.

property LedColor
Sets the color of the led.

property Enabled
TRUE if the led is on, FALSE if off.

property Blink
If TRUE the led will blink, if FALSE it will not.

property BlinkInterval
Set the number of milliseconds between blink (if Blink is TRUE).

event OnBlink
Called every time the status changes (from on to off, or the reverse).

event OnPaint
Called every time the led is redrawn.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzSegment

Unit ZSEG.PAS

A 16-part led-segment, that can show most characters. All the characters that it can show can be customized (All Segment-components will change at the same time).

property Char
Set the char You want the TzSegment to show.

property Color
Set the background-color of the TzSegment.

property Height
Adjust the size of the TzSegment.

property LitColor
The color a segment-part should have when lit.

property Punktur
Tell the TzSegment if any punktur should be displayed.

property Transparent
Set to TRUE if You want the form to show trough where the background of the TzSegment is displayed.

property Width
Adjust the size of the TzSegment.

property UnLitColor
The color a segment-part should have when not lit.

property Size
Adjust the size of the TzSegment.

event OnPaint
Called every time the TzSegment is redrawn.

TzSegmentLabel (and TzCustomSegmentLabel)

Unit ZSEG.PAS

A label-like component, that uses TzSegments to display the caption
Included in the "ZieglerCollection one" is a component called TzCustomSegmentLabel, that can be used as "mother component" for new types of lTzSegmentlabel components, and this component is itself a child of this component, without any new properties and such.

For same of the properties, have a look at TzSegment.

procedure DoUpdate
Call this procedure everytime You have changed the look of the letters the TzSegements can show, to update the display instantly.

property Caption
Sets the caption of the label-component.

property NumberOfChars
How many chars should be displayed by this component?

TzSegmentClock

Unit ZSEG.PAS

A label-like clock-component, that uses TzSegments to show the time. It can show a clock with or without seconds. Every time it is updated (once per second if seconds are shown, else once per minute) it can call an event-handler, and thereby be used to control other components.

The component is a child of TzCustomSegmentLabel.

For same of the properties, have a look at TzSegment.

property ShowSeconds
If TRUE, the label will show seconds, if FALSE it will not.

event OnTimeChange
Called every time the time is updated (Once every second if seconds are shown, else once every minute).

TzGauge

Unit ZGAUGE.PAS

A component that can be used to show progress in any operation. Can use a glyph or a colorbar to show the progress. Can be displayed in a lot of different styles.

procedure AddValue (Value:LongInt);
If Value is positive, the Value will be added to the value already stored in the gauge. If Value is negative, the value already in the gauge will be reduced with Value.

property Glyph
If a bitmap is put in this property, this bitmap will be used to show the progressbar.

property Percent
Set or read how many percents of the maximum value, the gauge is displaying now.

property DrawOnlyWhenPCTChange
If TRUE, the gauge will only be updated everytime the pct is changed, not everytime the Value is changed. If FALSE, the gauge will be updated everytime the Value is changed.

property CaptionStyle
Set the style of the caption (if any).

property BarColor
If no Glyph is used, this is used to decide what color the gauge-bat should have.

property Caption
In the caption You can use format-specifier. The text is displayed with functions like this one: Format(fCaption,[GetPercent]) (for the CaptionStyle gaPctBar), this one: Format(fCaption,[gValue]) (for gaTotalBar), this one: Format(fCaption,[gValue,GetPercent]) (for gaTotPct) or this one: Format(fCaption,[GetPercent,gValue]) (for gaPctTot). As can be seen, the programmer is very much in control over the look of the caption.

property Min
Sets the minimum value the gauge can take.

property Max
Sets the maximum value the gauge can take.

property Value
The value the gauge has right now.

property SpaceBetweenLed
If NumberOfleds is different from zero, this property decides how big a gap should exist between the led-elements.

property NumberOfLed
If different from zero, the progressbar is cut into pieces.

event OnMinimum
Called everytime the value reaches Min.

event OnMaximum
Called everytime the value reaches Max.

event OnPaint
Called every time the control is redrawn.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzSlideBar

Unit SLIDEBAR.PAS

A component that can be used anywhere You want to be able to scroll through a range of numbers. One use could be as a volume-control. It can take a large number of shapes.

function CurrentLabel: String;
Every position on the slidebar can have a string connected. This function returns the string, connected to the value right now.

property SteelLook
If TRUE, the background is replaced with a light gray steel-look.

property TickSpace
This is the distance from the slider-center, and out to the start of the tick-marks.

property TickSize
How big is the tick-marks?

property TickWhere
Where should we display tick-marks?

property StepPerTick
Distance between the tick-marks.

property FocusColor
The color of the slider when this control is in focus.

property NonFocusColor
The color of the slider when this control is not in focus.

property Labels
A list of strings connected to the values the slidebar can show.

property Max
The maximum value the slidebar can show.

property Min
The minimum value the slidebar can show.

property Orientation
Is the Slidebar Horizontal or vertical?

property Value
The value of the slidebar right now.

property Thickness
The thickness of the slider.

property ThumbStyle
How should the thumb look.

property Ticks
If TRUE, then the control shows tick-marks, if FALSE then it don't.

property Style
How should the control look? Lowered or raised?

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

event OnMinimum
Called everytime the value reaches Min.

event OnMaximum
Called everytime the value reaches Max.

event OnChange
Called everytime the value changes.

TzFrame

Unit ANIMATE.PAS

A component that can be used to make frames around other components, or on its own just like the tBevel component found in the VCL-component collection. The big difference is the control You have over the component.

property PassMouseOn
If TRUE, this control don't answer to mouseevents, but instead they are passed on to the parent of this control. If set to FALSE, this component will receive mouseevents.

event OnPaint
Called everytime this control is redrawn.

TzDivider

Unit ANIMATE.PAS

The most simple component in the whole collection: A divider-line, that can be horizontal or vertical. It is used to separate groups of other components on forms (But can be used for other things too, like dropzone for drag-drop components, because You can set it to accept drops below the line itself, or to the right of it).

property Orientation
Is the TzDivider Horizontal or vertical?

property Raised
Is the line raised above the surface, or is it engraved into it?

event OnPaint
Called everytime this control is redrawn.

TzMovePanel

Unit MYSTD.PAS

A tPanel component, that can be moved around with the mouse, useful for building floating toolbars.

procedure DoUpdate
Call it whenever You move the panel yourself.

property StickWhere
Set where You want the panel to stick.

property StickOldStyle
For compatibility with earlier version of ZieglerCollection, set this to TRUE.

event OnBeforeUpdate
Called just before the panel is updated. Can be used to move the panel or whatever is wanted.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzTitleBar

Unit MYSTD.PAS

A smart component, useful if You want to set and remove the titlebar on a form, while Your program is running. It has one more function, and that's to make sure You can move Your form, even when no caption exists.

property ShowTitleBar
If TRUE, the titlebar will be shown, if FALSE it will not.

property AllowMoveWhitoutTitle
If TRUE, then You can move the form by dragging it (grab it outside every control/component). If You use this, You can't have things like SpeedButtons on the form, because they will be treated as the form itself. If FALSE, the form can't be moved (with the mouse) when no titlebar is shown.

property TitleBarOnFormShow
If TRUE, then the titlebar will be shown when the form displays, if FALSE, it will be removed before the form shows up.

event OnChange
Called when the status changes.

TzHint

Unit MYSTD.PAS

A component, that make it a lot easier to manipulate the hint-window. It also allows You to have multiline hints. Easy change of the font used to write the hint-text.

procedure UpDate;
If You change anything about the hint-system, calling this procedure will make sure the changes will be in effect.

property HighLightColor
If Use3D is set to TRUE, this is the color used to draw the highlight parts of the hint-window.

property ShadowColor
If Use3D is set to TRUE, this is the color used to draw the shadow parts of the hint-window.

property HintPosition
Where should the hint be shown?

property PartOfScreen
How many parts should the screen be devided into? One of these parts will be used for the Hint-window.

property MoveBeforeOff
How many pixels can the mouse be moved before the hint-windows is closed?

property MultiLine
TRUE if You want to use multi lines in the hint-window, FALSE if not.

property Use3D
TRUE if the Hint-window should be shown in 3D, FALSE if not.

property Font
What font should the text in the hint-window be shown in?

event OnHintPaint
Called when the hint-window is painted.

event OnHintShow
Called when the hint-window is about to be shown.

TzShowApp

Unit MYSTD.PAS

A component, that makes it a lot easier to control Your tApplication component. You can even assign event-handlers at design-time. Includes a lot of information about the environment too.

Normally You only use this component on the main-form of Your program.

property EnvironmentLines
A read-only property, which has all the environment variables.

property OnlyOneInstance
If set to TRUE, only one instance of the program can be run. If more than one instance is started, then the newly started instance is closed again, and the first instance is brought to top.

property WindowKind
Under what type of Windows is this program running? Even in 16-bit Delphi (Delphi 1) program running on Windows95, it will return the right type (W_95). Read-only.

property WinVerMajor
A read-only property containing the major version number of the Windows version the program is run under.

property WinVerMinor
A read-only property containing the minor version number of the Windows version the program is run under.

property WinVerBuild
A read-only property containing the build-number of the Windows version the program is run under.

property WindowsString
A read-only property containing a string, which tells what windows version the program is run under. This string is suitable for display.

property CPUType
A read-only property containing a number which represents the CPU in the computer the program is run on.

property CPUString
A read-only property containing a string, telling what CPU the computer has. This string is suitable for display.

property UseCmdShow
If set to TRUE, the program will use the information passed over when the program starts (Should the program start minimized, maximized or normalized?). A Delphi program normally don't care about this stuff, so if You need to be sure the program starts the way the user want, then this is the property to use. If FALSE the normal Delphi method is used.

property HelpFile
Set the application helpfile.

property Hint

Set the application hint.

property HintColor
Set the application HintColor.

property HintPause
Set the application HintPause.

property HintHidePause
ONLY 32-BIT. Set the application HintHidePause.

property ShowOnStatusBar
ONLY 32-BIT. If set to TRUE (default), the application shows up in the statusbar, as normal programs do. If FALSE, the program will be removed from the statusbar. (useful for programs that uses the tray).

property ShowMainForm
ONLY 32-BIT. If set to TRUE (default), the application will show itself as normal. If FALSE, the program will not show its main-form on start (useful for programs that uses the tray).

property Icon
Set the application Icon.

property ShowHint
Set the application ShowHint.

property Title
Set the application Title.

event OnActivate
The application OnActivate event.

event OnDeActivate
The application OnDeactivate event.

event OnExecption
The application OnExecption event.

event OnHelp
The application OnHelp event.

event OnHint
The application OnHint event.

event OnIdle
The application OnIdle event.

event OnMessage
The application OnMessage event.

event OnMinimize
The application OnMinimize event.

event OnRestore
The application OnRestore event.

event OnShowHint

The application OnShowHint event.

event OnAskEndSession
Called every time the user tries to close windows. The closedown can be stopped in this event.

event OnEndSession
Called every time Windows closes down.

event OnSecondInstance
If this instance of the program is not the first, and OnlyOneInstance is TRUE, this event will be called, so the program can do anything needed before it closes down again.

event OnAnotherInstance
If another instance is started, and OnlyOneInstance is TRUE, this event is called, just before the second instance is closed down again.

TzVerSplit

Unit ZSPLIT.PAS

A component that makes it easy to make a vertical splitter-window. It is possible to have any number of splitter-component on a form.

When put on a form, You can start drop components on the right side of the splitter-bar. To put component on the left side, set WhereSplit to a large number (Larger than the form-width), and it is possible to drop the wanted components on the left side. For this to work, the following two properties must be set to TRUE: MoveOnChangeLeft and MoveOnChangeRight. When done dropping components on the left side, set WhereSplit to 0 (zero) again (or to the wanted value).

In the ZSPLIT unit a component called TzSplit is declared. This component is ONLY for internal use, and is used by TzVerSplit and TzHorSplit, to show the splitter-bar. DO NOT USE THIS COMPONENT FOR ANYTHING ELSE.

procedure DoSize
This procedure is used internally, to make sure the splitterbar and the split-window is in the right size, according to the other part.

procedure GoMinimum
If called, the splitter-bar goes to the leftmost position (or if this is a TzHorSplit, it goes to the topmost position).

procedure GoMaximum
If called, the splitter-bar goes to the rightmost position (or if this is a TzHorSplit, it goes to the bottom).

procedure GoMedium
If called, the splitterbar goes to the position evenly between the leftmost- and the rightmost position (or if this is a TzHorSplit, between the top and bottom).

procedure SetLeftTop(A:tControl)
Align the upper-left part of the splitter-control to the given control.

procedure SetRightBottom(A:tControl)
Align the lower-right part of the splitter-control to the given control.

property PassMouseOn
If TRUE, the mouseevents don't get processed in this component, but is passed on the parent. If FALSE, the splitter-component will handle mouse-events.

property MoveOnChangeLeft
If TRUE (default), the components on the left side (or top side) are moved with the splitter-bar. This is the normal function of a splitter-window. If You want to be in control yourself, then set this to FALSE, and the splitter-bar will not move anything on the left-side (or top-side).

property MoveOnChangeRight
If TRUE (default), the components on the right side (or bottom side) are moved with the splitter-bar. This is the normal function of a splitter-window. If You want to be in control yourself, then set this to FALSE, and the splitter-bar will not move anything on the right side (or bottom-side).

property SplitterStyle

How should the splitter-bar look?

property SplitterWidth
Width of the splitter-bar.

property SplitterInner
How should the splitter-bar look?

property SplitterOuter
How should the splitter-bar look?

property WhereSplit
The position of the splitter-bar.

property NewLook
If TRUE, the splitter-bar will look like the new Microsoft-look, if FALSE You decide with the SplitterInner and SplitterOuter.

event OnSplitChange
Called every time the splitter-bar is moved.

event OnBeforeShow
If You want to set something just before the splitter is shown the first time (position of controls and splitter-bar and such stuff), this is the event where this should be done.

TzHorSplit

Unit ZSPLIT.PAS

A component that makes it easy to make a horizontal splitter-window. It is possible to have any number of splitter-component on a form.

To see how this component works, have a look at TzVerSplit.

TzMouseSpot

Unit MYSTD.PAS

A component, useful for adding mouseevnts to pictures, maps and drawings. It don't use a window-handle, so You use as many TzMouseSpots on a single picture as You want. It also makes it easy to prevent mouseevents at some parts of the picture in some situations, and in other parts in other situations. The component can only sit on top of other non-windowed controls.

event OnParentPaint
Called when the component below the TzMouseSpot is repainted.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzCalc

Unit MYSTD.PAS

A non-visible component, used to do calculations. You pass it a string with the calculation, and this component will try to solve it for You.

property CalcResult
How did it go the last time we did a calculation?

property WhereIsLastError
If the last calculation did not go well, where in the string (in number of chars) did the error happened?

property CalcLine
The string we want to calculate. This could be something like "4^6*log(25)".
It is possible to use the following build-in functions in the string:
A (This is the result placed in the A-register),
B (This is the result placed in the B-register),
C (This is the result placed in the C-register),
D (This is the result placed in the D-register),
PI, E, EXP, SIN, COS, ARCTAN, ABS, FRAC, ARCSIN, ARCCOS, LN,
LOG, SQR, SQRT and ROUND
Every time You put a new calculation into this property, the result from the last one is moved one register down (from NumberNum to registerANum, from registerANum to RegisterBNum and so on).

property FormatLine
In this property You can use the normal format-specifiers to decide how all the string-version of the registers should look.

property Number
read-Only. This is the result of the last calculation, formatted according to FormatLine.

property RegisterA
property RegisterB
property RegisterC
property RegisterD
Read-Only. The results of the calculations before the last one, formatted according to FormatLine.

property RegisterANum
property RegisterBNum
property RegisterCNum
property RegisterDNum
Read-Only. The result of the calculations before the last one.

property NumberNum
Read-Only. The last result.

event OnError
Called every time an calculation error exists.

event OnOK
Called when a new calculation is done, and this calculation don't have any errors.

TzShapeBtn

Unit ANIMATE.PAS

A non-windowed button-component, that does not receive focus. It can take almost any shape, by using the build-in types, or by using the owner-draw option. It will only be pressed when the mouse is in the shape of the button, not when the mouse is in the rectangle that surrounds the button.

procedure DrawRectAngle
procedure DrawTriangle(Direction:byte)
procedure DrawPlus
procedure DrawMinus
procedure DrawCircle
Procedures that does the painting of the build-in shapes. Normally only used internally.

property Region
Used by all drawing routines, to uptain the region that makes the button-shape.

property IsDown
Used by all drawing routines, to decide if the button should be drawn down or up.

property Shape
What should the button look like? This is also the place where You set the owner-drawn style.

event OnPaint
Called every time the button is redrawn.

event OnGetRegion
If ownerdrawn, this is called when the button needs to now its region.

event OnOwnerDraw
If ownerdrawn, this is called when the button needs to be redrawn.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzColorBtn

Unit ANIMATE.PAS

Nearly a standard Tbutton with one very big exception, it has the possibility of changing colors, the face-color, highlight-color and shadow-color. Can have a caption with more than one line of text too. For all properties not mentioned here, look at TButton in the Delphi help.

property Flat
If set to TRUE, the button-outline is not visible when the mouse is not over the button.

property HorzAlign
Set the alignment of the text, in the horizontal plane.

property MultipleLines
Set to TRUE if You want to have more than one textline on the button. This will affect how it is possible to adjust the look, because Windows don't allow all combinations of adjustment when more than one line of text is displayed.

property WordWrap
If more than one line of text, this decides if it should have wordwrap turned on or off.

property Color
The face-color of the button.

property HighLightColor
The highlight color of the button.

property ShadowColor
The shadow color of the button.

event OnPaint
Called every time the button is redrawn.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzGradBtn

Unit ANIMATE.PAS

A TzColorBtn with a little extra. Will paint a gradient fill from Color to Endcolor on the surface of the button instead of just one color as the TzColorBtn. It can be used instead of a TzColorBtn, because it will be the same if both colors are set to the same value.

property FillType
How du we want to see the gradient fill?

property EndColor
The end-color of the fill. The start-color is the one in Color.

property Bands
How many parts do we want to split the color-blend into?

TzBitColBtn

Unit ANIMATE.PAS

A TzColorBtn with a little extra (Can include a glyph-bitmap on the button surface). If the Glyph includes more than one picture (it can hold from 1-4) the first will be used for the normal Picture, the second will be the disabled picture and the third is used for down. The fourth is not used but is accepted for compatibility reasons. It is not fully compatible with the standard tBitBtn. You decide which color is to be used for the transparent parts of the bitmap, it will not do it automatic as the tBitBtn does, because we think You may need more control over it. Has a "AutoBackColor" property where it uses Delphi's own way to get the transparent color.

property Glyph
The bitmap used on the button.

property NumGlyphs
How many bitmaps is in the Glyph? (1-4).

property TransparentColor
What color should be used as transparent (where the background shines trough)? Don't use it if You set AutoBackColor to TRUE.

property BitmapWhere
Where on the button should the bitmap go?

property AutoBackColor
If TRUE, use Delphi's normal way to find the background color, FALSE if we want to decide it ourselves with the help of TransparentColor.

TzIconColBtn

Unit ANIMATE.PAS

A TzColorBtn with a little extra (Can include 3 icons for enabled/disabled/down button).

property AutoAllIcons
If TRUE, the same icon will be used (if anotherone is not put in) in all positions (enabled/disabled/down).

property GlyphEnabled
The Icon used for Enabled.

property GlyphDown
The Icon used for Down.

property GlyphDisabled
The Icon for Disabled.

property IconWhere
Where on the button should the Icon's be shown?

TzScope

Unit ZPANEL.PAS

A simple way to display a lot of data visible. Can show one or two graphs, looking just like an oscilloscope. Can set all the data at once, can put one value in at a time and can automatically push all the data already on the scope forward when a new value comes in.

procedure Push
Moves all data forward. This can be called by a timer, so that the data is moved in a constant speed, or You can set AutoPush to TRUE, so that You don't have to call this procedure yourself.

procedure SetAllChannel1(Const Values:BeamArray)
Set all values (max. 1000) for the first channel at once.

procedure SetAllChannel2(Const Values:BeamArray)
Set all values (max. 1000) for the second channel at once.

property AutoPush
Set to TRUE, if You want the data to move forward every time new values enters the control, FALSE if You want to call Push yourself (or has a timer to do it).

property Min
The minimum value that can be displayed.

property Max
The maximum value that can be displayed.

property GridXstep
Space between grid-lines in x-direction.

property GridYstep
Space between grid-lines in y-direction.

property ShowGrid
TRUE if You want the grid to show, else FALSE.

property GridColor
The color of the grid.

property Beam1Color
The color of the first beam.

property Beam2Color
The color of the second beam.

property ShowBeam1
TRUE if the first beam should be visible, else FALSE.

property ShowBeam2
TRUE if the second beam should be visible, else FALSE.

property Channel1

The last value put into the first beam (it is here You deliver the data You want to show).

property Channel2
The last value put into the second beam (it is here You deliver the data You want to show).

property Ajust1
Move the point of zero for the first beam.

property Ajust2
Move the point of zero for the second beam.

property FaceColor
The color of the background in the scope.

event OnPaint
Called every time the control is redrawn.

event OnBeforePush
Called every time the data is about to be pushed one place.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzPanelMeter

Unit ZPANEL.PAS

A component used to display values in a way that looks like Panel meters. This is a very simple component, but it is very useful to give quick and dirty displays of values, easy to read. Can show in many different shapes, so it should be possible to use it in nearly any type of program.

property Min
The minimum value that can be displayed.

property Max
The maximum value that can be displayed.

property ShowAll
One of the options, related to the look of the control.

property UseAlert
One of the options, related to the look of the control.

property PanelType
How should the panel-meter look? The main option, related to the look of the control.

property HouseColor
The color of the panel-meter-house.

property HouseStyle
How should we paint the house?

property ShadowColor
property HighColor
property FaceColor
property NormalColor
property AlertColor
property Hand
Colors for the various parts of the panel-meter-control.

property Ticks
TRUE if we want to show tics, FALSE if not.

property StepPerTick
If Ticks is TRUE, how much space should be between the ticks?

property Alert
On which value should the "Alert-zone" start?

property Value
The value.

property ScaleWidth
Width of the scale.

property SteelLook

Paint the house in SteelLook if set to TRUE.

event OnPaint
Called when the control is redrawn.

event OnAlert
Called every time the value goes from normal to Alert.

event OnNotAlert
Called every time the value goes from Alert to normal.

event OnChange
Called when the value changes.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzKnob

Unit ZPANEL.PAS

A component that looks and works like a dial (knob) on a stereo. Very easy to use. Is a windowed control, and can have focus, and use the keyboard for adjusting the value.

When the knob has focus, the arrow-keys is used to select the wanted value. (If Dbl- or TripKnobs, then use the arrows in company with Shift and Ctrl for selecting values).

property HideReflex
If TRUE, the knob don't show a moving reflex when the value is changes, if FALSE, it will show the reflex.

property Min
The minimum value that can be selected with this knob.

property Max
the maximum value that can be selected with this knob.

property Ticks
If TRUE, show tick-marks, if FALSE, don't.

property TickColor
Color of tick-marks.

property StepPerTick
Distance (in value) between the ticks.

property PointColor
The color of the point, which shows the selected value of the knob.

property PointWidth
Width of the select-point of the knob.

property KnobColor
The color of the knob.

property Value
The selected value of the knob.

property SteelLook
If TRUE, then the background is shown in a gray steel-look.

event OnPaint
Called when the knob is redrawn.

event OnChange
Called when the value changes.

event OnMinimum
Called when value enters the minimum value.

event OnMaximum

Called when the value enters the maximum value.

event OnMouseEnter
Called when the mouse enters the control.

event OnMouseLeave
Called when the mouse leaves the control.

TzDblKnob

Unit ZPANEL.PAS

A TzKnob component. The difference is that this component can be used to select two values in the space of one component. Look at TzKnob for most of the functionality.

property Value2
The value of the second knob part.

event OnMinimum2
Called when value2 enters the minimum value.

event OnMaximum2
Called when the value2 enters the maximum value.

TzTripKnob

Unit ZPANEL.PAS

A TzKnob component. The difference is that this component can be used to select three values in the space of one component. Look at TzKnob for most of the functionality, and in TzDblKnob for more.

property Value3
The value of the third knob part.

event OnMinimum3
Called when value3 enters the minimum value.

event OnMaximum3
Called when the value3 enters the maximum value.

TzTrayIcon

Unit MYSTD.PAS

A component for easy handling of programs, that need to show itself in the tray (Windows'95 and NT 4.x and later only). In cooperation with TzShowApp, it can even remove the program from the normal statusbar, and make sure the main-window is not shown.

Can hide the form it is placed on, and together with TzShowApp, it can hide the program altogether, so that only the icon in the tray is visible. If the form it is placed on, is the main-form of the application, then use a TzShowApp on the form and set "ShowMainForm" to false because "HideParentForm" is only for forms that are not mainforms. (and it don't work on main-forms, to avoid the flicker). You will need a TzShowApp anyway, if You want to remove the program from the statusbar.

If You are using TzTrayIcon on other forms than the mainform of Your program, then just set "HideParentForm" to true to hide the form. You will still need a TzShowApp on the mainform of Your program, if You want to remove the program from the statusbar.

If more than one TzTrayIcon is used on the same form, You will have to set the "HideParentForm" to the same value for all then Icons.

property HideParentForm
TRUE if the form this component is on, should be non-visible (se above for explanation).

property ShowDesigning
TRUE if You want to see the icon in the tray in designtime too (remember to set it to FALSE when You are done with the program).

property Active
Should the icon be displayed?

property Icon
The icon You want in the tray. If set to nothing, a blank icon will be used.

property ToolTip
The ToolTip, You want the user to see when the mouse is over the icon in the tray.

property OnClick
property OnDblClick
property OnRightClick
property OnRightDblClick
property OnMidClick
property OnMidDblClick
property OnMouseMove
proeprty OnMouseDown
property OnMouseUp
Called when the appropriate mouseevent happens.

TzNWColorBtn

Unit ANIMATE.PAS

A Non-windowed button. It functions in most ways as a TzColorBtn. Can show a flat and/or a transparent look if wanted. Can of course not have the focus.

For most of the properties, events and so on, have a look at TzColorBtn.

property Transparent
Set to TRUE if the background should shine trough.

property GlassValue
A value, that describes how transparent the button should be (a good value is 145).

TzNWBitColBtn

Unit ANIMATE.PAS

A Non-windowed button. It functions in most ways as a TzBitColBtn. Can show a flat and/or a transparent look if wanted. Can of course not have the focus.

For most of the properties, events and so on, have a look at TzBitColBtn, the rest can be found in TzNWColorBtn.

TzNWIconColBtn

Unit ANIMATE.PAS

A Non-windowed button. It functions in most ways as a TzIconColBtn. Can show a flat and/or a transparent look if wanted. Can of course not have the focus.

For most of the properties, events and so on, have a look at TzIconColBtn, the rest can be found in TzNWColorBtn.

TzNWBlendPaint

Unit BLEND.PAS

A backdrop that can be started and stopped on any color, and will make a gradient fill between the two colors it is a non-windowed control. It works in most ways like TzBlendPaint.

For most of the procedure, events and properties, look at TzBlendPaint.

property PassMouseOn
If TRUE, the mouseevents is not handled, but passed on to the parent. If FALSE, this component handles the mouseevents itself.

TzNWTileMap

Unit BLEND.PAS

A backdrop that can be started and stopped on any color, and will make a gradient fill between the two colors it is a non-windowed control. It works in most ways like TzTileMap.

For most of the procedure, events and properties, look at TzTileMap, and for the rest look at TzNWBlendPaint.

TzResBitmap

Unit ANIMATE.PAS

A tBitmap descendant, that will load its glyph from the resourcefile. It is not a component with an entry in the Component-Palette, but can be created at runtime when needed. It is used in some of the "ZieglerCollection one"'s components.

To use it, do something like this:

```
Var
  TestBitmap : TzResBitmap;
Begin
  TestBitmap:=TzResBitmap.Create('TheResourceName');
...
...
  TestBitmap.Free;
End;
```

It has no new properties or functions, so just look at tBitmap in the Delphi help-file for information.

TzDeskTop

Unit ANIMATE.PAS

A tCanvas descendant, that holds the complete desktop. This makes it very easy to write directly on the desktop. Just create a TzDesktop, and use this for drawing. Also useful for reading everything on the desktop. For most of the use, have a look at TCanvas in the Delphi help-file.

property Width
The width of the Desktop.

property Height
The height of the Desktop.

BLEND.PAS

Types:

TzBackType=(btBitmap,btBlend,btNormal,btSteel);
How the background is filled in various backdrops.

TzBackPaintEvent=procedure(Canvas:tCanvas;Sender:TObject) of Object;
Event-type, used by backdrops, called when painting of the backdrop takes place.

Components/Classes:

TzBackground
TzBlendPaint
TzNWBlendPaint
TzTileMap
TzNWTileMap

ANIMATE.PAS

Types:

TPercentType = 0..100;
Used whenever a percent value is needed.

TzGlassValue=0..255;
A number used to decide how "glass-lookalike" a button should be.

doType = (doHorz,doVert);
Is a TzDivider horizontal or vertical.

```
sbType = (sbRectangle,
      sbUpTriangle,
      sbDnTriangle,
      sbLTriangle,
      sbRTriangle,
      sbTLTriangle,
      sbTRTriangle,
      sbBLTriangle,
      sbBRTriangle,
      sbPlus,
      sbMinus,
      sbCircle,
      sbOwnerDraw);
```
The looks of the TzShapeBtn.

zbHorzAlign = (zbha_Center,zbha_Top,zbha_Bottom);
How to adjust multiline buttons.

```
TsbDrawEvent=Procedure(Sender:TObject;Canvas:TCanvas;Down:Boolean;Width,
      Height:Integer;Region:HRGN;Shadow,Highlight:tColor) of Object;
```
Called when an ownerdrawn TzShapeBtn needs to be painted.

TsbGetRegEvent=Procedure(Sender:TObject;Var Region:HRGN;Width,Height:Integer) of Object;
Called when an ownerdrawn TzShapeBtn needs to know how it looks.

tBitWhere=(z_BWLeft,z_BWRight);
On buttons with bitmaps, where should the bitmap be?

tBitNumber=1..4;
How many bitmaps is used on a bitmap button?

Components/Classes:

TzColorBtn
TzGradBtn
TzShapeBtn
TzResBitmap
TzDesktop
TzBitmap
TzAnimated
TzDivider
TzFrame

TzBitColBtn
TzIconColBtn
TzNWColorBtn
TzNWBitColBtn
TzNWIconColBtn

Procedures:
BMPRotate
DarkenBMP
LightenBMP
GreyBMP
TransparentBlt

SLIDEBAR.PAS

Types:

```
TBarStyle       = (bsLowered,bsRaised);
```
How should the slidebar look?

```
TBarTick        = 1..MaxInt;
```
How many units between each tick on the slidebar?

```
TBarTickStyle= (ttsLeft,ttsRight,ttsBoth);
```
Where should the slidebar display tickmarks?

```
TOrientation = (orVertical,orHorizontal);
```
Is the slidebar horizontal or vertical?

```
TThumbStyle    = (tsHorzHigh,
      tsVertHigh,
      tsHorzLow,
      tsVertLow,
      tsSquare,
      tsSquarehollow,
      tsLeft,
      tsRight,
      tsTop,
      tsBottom,
      tsRound,
      tsHorzRound);
```
How should the thumb of the slidebar look?

Components:

TzSlideBar

ZGAUGE.PAS

Types:
gaCaptionStyle = (gaPctBar,gaTotalBar,gaTotPct,gaPctTot);
In what style should the caption of the gague be shown?

Components/Classes:
TzGauge

ZLED.PAS

Types:
LedColorType=(lctRed,lctBlue,lctGreen,lctyellow,lctMangenta,lctGrey,lctCyan,lctWhite);
What color should the led have?

LedStyleType=(lstRound,lstSquare);
Should the led be round or square?

Components/Classes:
TzLed

MYSTD.PAS

Types:

EzTabListboxError = Class(Exception);
An exception, used when something goes wrong in the TzTabListbox.

TzStickType = (stNone,stleft,stTop,stRight,stBottom, stBoth);
How should a TzMovePanel behave?

TzMoveDist=2..20;
How much can the mouse be moved before hint disappears?

TzAngle=0..359;
Used when we need an angle-value.

TzHintPos=(hpLeftAbove,hpLeftUnder,hpRightAbove,hpRightUnder,hpNormal);
Where should the hint be shown?

TzHintDiv=1..10;
The hint uses this to decide how many parts the screen is divided into, and will use exactly one of these parts to decide how wide the hint can be on the screen.

THintShowEvent=Procedure(Sender:TObject;Var ShowHint:Boolean) of Object;
Called when the hint shows.

TChangeEvent=Procedure(Sender:TObject;PrevItemIndex:Integer) of Object;
Called when the TzTabListboc changes.

TCalcErrorEvent=Procedure(Sender:TObject;CalcLine:String;Where:Byte) of Object;
When an error happens in a TzCalc.

TUpdateEvent=Procedure(Sender:TObject;Var X,Y,W,H:Integer) Of Object;
Called when the TzMovePanel is about to be redrawn.

TAskEndSession=Procedure(Var DoEnd:Boolean) Of Object;
Called when the user tries to close windows.

TEndSession=Procedure Of Object;
Called when Windows is closing.

TOneInstance=Procedure(Sender:TObject;Var BringFirstInstanceToFront:Boolean) Of Object;
Called when user tries to start more than one instance of this program.

ScrollDirection=(zScrollHor,zScrollVer);
Decide which scrollbar was used in TzTabListBox.

```
TScrollEvent=Procedure(Sender:TObject;Position:SmallInt;ScrollCode:SmallInt;
    ScrollDir:ScrollDirection) of Object;
```
Called when user scrolls the TzTabListBox.

TzToolTiptext = String[62];
Used for TrayIcons.

TzCalcResult = (TzcOK,TzcNumberMissing,TzcFunctionMissing,TzcLeftPar, TzcRightPar,TzcDivZero,TzcOverFlow,TzcFormatError);

The result of the last calculation in a tzCalc.

Components/Classes:

TzTrayIcon
TzCalc
TzTitleBar
TzTabListbox
TzMovePanel
TzHint
TzShowApp
TzBigLabel
Tz3DLabel
TzAngleLabel
TzMouseSpot

Functions:

FindForm
IsPrevius
DlgUnitsToPixelsX
DlgUnitsToPixelsY
PixelsToDlgUnitsX
PixelsToDlgUnitsY
CpuID
ArcSin
ArcCos
Log10
Power
Factorial
IsPrime
Root

ZSPLIT.PAS

Components/Classes:
TzSplit
TzVerSplit
TzHorSplit

ZHELPER.PAS

Types:
WTypes = (z_Windows,z_WFW,z_Win32S,z_NT,z_95);
What type of Windows is running? (The real one).

(The rest is only available in Delphi 1, and is used internally to call 32-bit functions from 16-bit Delphi. It is not ment to be used by anyone else beside "ZieglerCollection one", and is not supported)

```
Handle32 = LongInt;
WOW_HANDLE_TYPE=(
    WOW_TYPE_HWND,
    WOW_TYPE_HMENU,
    WOW_TYPE_HDWP,
    WOW_TYPE_HDROP,
    WOW_TYPE_HDC,
    WOW_TYPE_HFONT,
    WOW_TYPE_HMETAFILE,
    WOW_TYPE_HRGN,
    WOW_TYPE_HBITMAP,
    WOW_TYPE_HBRUSH,
    WOW_TYPE_HPALETTE,
    WOW_TYPE_HPEN,
    WOW_TYPE_HACCEL,
    WOW_TYPE_HTASK,
    WOW_TYPE_FULLHWND);
```

Functions:
WindowsType
MajorVersion
MinorVersion
BuildVersion
IsWorkgroup
Is311

(The rest is only available in Delphi 1, and is used internally to call 32-bit functions from 16-bit Delphi. It is not ment to be used by anyone else beside "ZieglerCollection one", and is not supported)

```
WOWHandle32(Ind:tHandle;hT:WOW_HANDLE_TYPE):Handle32;
WOWHandle16(Ind:Handle32;hT:WOW_HANDLE_TYPE):tHandle;
Declare32(Name,Lib,Arg:pchar):longint;
GetVDMPointer32W(name:pchar;Length:word):longint;
GetLastError:LongInt;
```

Procedures:
(The rest is only available in Delphi 1, and is used internally to call 32-bit functions from 16-bit Delphi. It is not ment to be used by anyone else beside "ZieglerCollection one", and is not supported).

```
Call32(iProc:longint);
SetLastError(ErrorCode:LongInt);
```

STD2.PAS

Types:
TAskSize=Procedure(Sender:TObject;Var MinDragWidth,MinDragHeight,
MaxWidth,MaxHeight,MaxDragWidth,MaxDragHeight,
Maxleft,MaxTop:Integer) Of Object;
Called when the form is about to be changed in size/position.

Variables:
TSteel : tBitmap;
Used by all drawing-functions in "ZieglerCollection one" which have a "Steel-look".

Components/Classes:
TzMinMax

ZPANEL.PAS

Types:

BeamArray = Array[0..1000] of Integer;
Used for filling of a complete TzScope control in one go.

tPushEvent = procedure(Sender: TObject; var DoPush:Boolean) of object;
Called when the TzScope is about to move one step.

KnobColorType=(kcRed,kcBlue,kcGreen,kcYellow,kcMagenta,kcSilver,kcCyan,kcWhite);
The color of the knob(s).

KnobPointWidth=1..5;
The size of the point on the knob.

zPanelType = (zpStandard, zpEdgeHorz, zpEdgeVert);
How should the TzPanelMeter look+.

zScaleWidthType = 0..5;
The width of the scale on a TzPanelmeter.

Components/Classes:

TzScope
TzKnob
TzDblKnob
TzTripKnob
TzPanelMeter

ZSEG.PAS

Types:
zSegSize = 1..15;
The allowed sizes of the Segment.

zSegPunktur = (spPunktum,spColon,spNone);
Is any punktur light in a segment? Which one?

zChar = ' '..#255;
What chars can be shown/changed in a segment.

zSet   = Set of 1..16;
Used when changing the look of a char in all the segments.

Components/Classes:
TzSegment
TzCustomSegmentLabel
TzSegmentLabel
TzSegmentClock

Procedures:
ChangeACharLook

History

ZieglerCollection one

Version	00.01	First draft -
	00.09	- last draft
	00.90	Beta version, only released in 250 copies.
	01.00	First release **This version (E-mail) is free to all registered users of ZieglerCollection** Added our own Hint-editor to make life a bit easier, when working with multiline hints.
	01.01	Bug fixes (sorry about that). **This version (E-mail) is free to all registered users of ZieglerCollection.**
	01.10	Updated to run under Delphi 3 **This version (E-mail) is free to all registered users of ZieglerCollection** New unit: zhelper.pas (Functions to handle 32-bit functions under 16 bit, and version-functions). Added TzGradBtn Added TzBitColBtn Added TzDesktop Added TzTrayIcon Added TzMinMax Can now hide the application from the statusbar, and (in 32-bit) hide the mainform too. Can now find the "Real" windows version the application is run under Can now show a bitmap as "Gauge" in TzGauge.
	01.11	Internal release with some stuff added Added more styles to TzBlendPaint and TzBackGround Added more styles to TzLed Added TSteel (bitmap) Added TzPanelMeter Buttons now paint a lot faster "Autobackcolor" property added to TzBitmap, TzBitColBtn and TzAnimated Buttons now can have more than one line in Caption, has wordwrap and such.
	01.12	One more internal version, released to a few external persons Less flicker in TzLed Added "SteelLook" to the following components: TzBlendPaint TzBackGround TzSlideBar TzLed TzPanelMeter Added TzKnob Added TzDblKnob Added TzTripKnob Added TzScope
	01.15	Many new functions added TzLed can now be shown in all sizes The bitmap-loaded now remembers last used directory (in this session) .Added help-file (separate, free, product, but included in ZieglerCollection). Added "Flat" property to buttons. Many new bitmap-handling functions (Rotate, Darken, Lighten, graying and

blitting).
Added "OnMouseEnter" and "OnMouseLeave" to the following components:
   TzBitmap
   TzKnob
   TzDblKnob
   TzTripKnob
   TzPanelMeter
   TzScope
   TzAnimated
   TzShapeBtn
   TzColorBtn
   TzBitColBtn
   TzGradBtn
   TzLed
   TzIconColBtn
   TzBigLabel
   Tz3DLabel
   TzMovePanel
   TzMouseSpot
   TzTabListBox
   TzSlideBar
   TzGauge
Fixed some palette-errors
TzHint can now change font
Added OnScroll-event to TzTabListBox
Added new styles to TzSlideBar, colors to the thumb and a much faster drawing without flicker.

01.20 Late Summer 1997 release
This version (E-mail) is free to all registered users of ZieglerCollection.
Updated to run in C++ Builder too.
Added TzNWColBtn
Added TzNWBitColBtn
Added TzNWIconColBtn
Added TzNWBlendPaint
Added TzNWTileMap
New Event in TzApplication: OnEndSession
"New look" added to TzVerSplit and TzHorSplit
New "Stick" possibilities added to TzMovePanel
Added a new "Expert" for making longer Library-search-path in 32-bit.
Demo-versions now exists for Delphi 1, 2, 3 and C++ Builder.

Helpfile

Version 0.90 First version of help-file

Software License Agreement

Rights and limitations:

This agreement becomes effective when You install all or any part of the software contained on the disk or is part of the zip-file, included with this document. By using our software You agree to the terms of this license. If the terms of this agreement are unacceptable, You may return the package, including this document, diskettes (if any), within 8 days of the purchase date for a refund (E-mail versions can not be returned).

Two versions of the software exists: A demo version, without source-code, called DV in this document, and a full version, including source-code, called FV in this document (The FV version exist in two versions, one as a ZIP-file sent to You by E-mail from ZieglerSoft, and one sent to You, or bought by You in a store on diskette. There is no difference in the functionality of those two versions, so they are named the same in this document).

The DV version (Demo version):

ZieglerSoft gives the right to distribute the DV version of this product in an electronic form, as long as there is no direct charge for this distribution. This includes posting the software on WWW-sites, BBS's, CD-rom or diskette(s). The DV must be distributed in its original form, without any modifications. The DV version is normally distributed as a ZIP-file with the name zcd1xxxx.ZIP (or .EXE) (Delphi 1), zcd2xxxx.ZIP (or .EXE) (Delphi 2), zcd3xxxx.ZIP (or .EXE) (Delphi 3) or zcc1xxxx.ZIP (or .EXE) (C++ Builder 1) where xxxx is replaced with the versionnumber. You may not include the DV version in any programs You distribute outside Your own computer.

You may not use the DV version for writing software for sale, only for Your own testing of the units, before buying the FV version. If You want to use the unit in programs outside You own computer, You will have to buy the FV version of the product.

The DV version don't contain sourcecode.

You may use the DV version for testing for three weeks. After three weeks, if You still want to use the unit, then You must buy the FV version of the product.

The FV version (Full version):

ZieglerSoft provides the software and grants non-exclusive use of its contents. The software which accompanies this license ("ZieglerCollection one") is the property of ZieglerSoft and is protected by copyright law. By using "ZieglerCollection one" (the FV version) You agree to abide by the terms of this agreement. Only one user may use this software on a single computer. This software may not be used on a network or any other multi-user platform. Contact ZieglerSoft if You need a SITE or network license. You may make backup copies for Your own use only.

When "ZieglerCollection one" is being compiled into an executable with the extensions ".exe" or ".dll" then there are no licensing fees or royalties for distribution of the executable file. Should any part of ZieglerCollection one be used in a noncompiled application, such as: a value added VCL, VBX, OCX or anything like that, royalties do apply.

Failure to comply with the terms outlined in this agreement will result in termination of Your software license.

How to get in touch with ZieglerSoft

Address:	ZieglerSoft Rughaven 25,2 DK-9000 Aalborg Denmark
Phone:	+45 9811 3772
Home page:	http://www.zieglersoft.dk

E-Mail:

- Sales department: sales@zieglersoft.dk
- Support department: support@zieglersoft.dk
- Support for this product: zieglercollection@zieglersoft.dk
- Backup E-mail address: zieglersoft@compuserve.com

Sale:

- From our Home-page (http://www.zieglersoft.dk)
- From Compuserve (GO SWREG, select 14655)
- By sending an Euro-Check to our address (in DKK).
- By sending a Check drawn on a Danish bank to our address

The price for an E-Mail version of this product is DKK 240,00 or $52.00 (US) or £31.00 (UK). The diskette version (can't be ordered on Compuserve) is DKK 360,00 or $73.00 (US) or £43.00 (UK). The cheapest way to buy is in DKK, and as E-Mail version.