

Align Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Align: TAlign;
```

Description

The Align property determines how the controls align within their container (or parent control). These are the possible values:

Value	Meaning
alNone	The component remains where you place it in the form. This is the default value.
alTop	The component moves to the top of the form and resizes to fill the width of the form. The height of the component is not affected.
alBottom	The component moves to the bottom of the form and resizes to fill the width of the form. The height of the component is not affected.
alLeft	The component moves to the left side of the form and resizes to fill the height of the form. The width of the component is not affected.
alRight	The component moves to the right side of the form and resizes to fill the height of the form. The width of the component is not affected.
alClient	The component resizes to fill the client area of a form. If a component already occupies part of the client area, the component resizes to fit within the remaining client area. The Client area of a MegaPanel is defined as the area inside all the surrounding Bevels, Borders and Margins . If the Caption of the MegaPanel is set Inside the panel frame , then the captions area is also Excluded from the total client area. This prevents other controls with Align Properties set to alClient from covering the MegaPanel Caption. GroupBox style MegaPanels set the top of the client area to the Bottom of the Caption plus any CaptionSpace that has been assigned.

Using the Align property is useful when you want a MegaPanel to stay in one position on the form or within another Panel or MegaPanel, even if the size of the form or the panel changes. For example, you could use a panel component with a various controls on it as a tool palette. By changing Align to alLeft, you guarantee that the tool palette remains on the left side of the form and always equals the client height of the form.

Note that if you set the Align property to alClient the component fills the entire client area so that it will not be possible to select the parent form by clicking on it. In this case use the Object Inspector to access the form or press ESC.

Any number of components can have the same Align value in which case they will stack up along the edge of the containing form or panel. To adjust the order in which the controls stack up, drag the controls into their desired positions.

AutoCenter Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property AutoCenter: TAutoCenter;
```

Default Value

acNone

Description

Specifies the alignment of controls within the MegaPanel whenever the MegaPanel is Resized. These are the possible values:

Value	Meaning
acNone	Controls are NOT automatically centered when the MegaPanel is resized.
acControls	Controls are centered based on the Width of the controls only
acControlsAndLabels	Controls are centered taking into account the Width of the controls and the width of any automatically generated labels.

Caption Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Caption: string;
```

Default Value

Empty string

Description

The Caption property is the text that is used to create the MegaPanel's heading. The Caption property may be empty if not MegaPanel is required.

MegaPanel Caption Properties

CaptionAngle

CaptionBevelInner

CaptionBevelOuter

CaptionBevelMargin

CaptionBevelWidth

CaptionBorderStyle

CaptionBorderWidth

CaptionClearance

CaptionFont

CaptionFontStyle

CaptionHorzJustify

CaptionLocation

CaptionPosition

CaptionSpace

CaptionStyle

ColorCaptionBkgnd

Label Properties

Panel Properties

Picture Properties

Resize Properties

Events

CaptionAngle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionAngle: integer;
```

Default Value

0

Description

CaptionAngle specifies the rotation angle in Degrees of the Caption text. The default value of 0 displays the caption text in its normal orientation. The angle entered is normalized within the range of -180 to +180 degrees.

CaptionBevelInner Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionBevelInner: TPanelBevel;
```

Default Value

bvNone

Description

The style of the Inner Caption Bevel, may be set to bvNone, bvRaised or bvLowered.

CaptionBevelMargin Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionBevelMargin: integer;
```

Default Value

2

Description

If [CaptionLocation](#) is set to clOutsideFrame then this is the space between the outer edge of the MegaPanel and the first CaptionBevel.

If [CaptionLocation](#) is set to clInsideFrame then this is the space between the Inner Panel Bevel and the first Caption Bevel.

CaptionBevelOuter Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionBevelOuter: TPanelBevel;
```

Default Value

bvRaised

Description

The style of the Outer Caption Bevel, may be set to bvNone, bvRaised or bvLowered.

CaptionBevelWidth Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionBevelWitdh: integer;
```

Default Value

1

Description

The width of each of the Inner and Outer Caption Bevels.

CaptionBorderStyle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionBorderStyle: TBorderStyle;
```

Description

The style of the outer Caption Border. May be set to bsSingle which will draw a one pixel wide black frame around the Caption or bsNone which omits the border frame.

CaptionBorderWidth Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionBorderWidth: integer;
```

DefaultValue

1

Description

The width of the border Between the [Outer](#) and [Inner](#) Caption Bevels.

CaptionClearance Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionClearance: integer;
```

Default Value

2

Description

This is the amount of blank space to allow between the Caption's frame (Bevels and Borders) and the Caption's Text.

CaptionFont Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionFont: TFont;
```

Description

The font used to display the caption.

Note

If you wish to use a [Rotated](#) Caption then this font must be a TrueType font.

CaptionFontStyle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionFontStyle: TFontStyle;
```

Default Value

fsNormal

Description

The special effect style of the caption's font, may be one of the following settings:

Value	Meaning
fsNormal	No special effects
fsRaisedLight	The caption will appear slightly raised
fsRaisedHeavy	The caption will appear completely raised
fsInsetLight	The caption will appear slightly inset
fsInsetHeavy	The caption will appear completely inset
fsDropShadow	A shadow of the captions text is displayed behind it

Note

The colors [ColorHighlight](#) and [ColorShadow](#) are used to create the light and shadow of the 3D effects.

CaptionHorzJustify Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionHorzJustify: THorzJust;
```

Default Value

hjLeft

Description

The justification of the caption's text within the caption's frame. May be set to one of the following values:

Value	Meaning
hjLeft	The left sides of the caption's lines of text will be aligned
hjCenter	The caption's lines of text will have their centers aligned
hjRight	The right side of the caption's lines of text will be aligned

Note

If the Caption is rotated then this setting should initially be set to its default value of hjLeft.

CaptionLocation Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionLocation: TCaptionLoc;
```

Default Value

```
clOutsideFrame
```

Description

This property sets the relative location of the caption with respect to the MegaPanel's frame. It can be set to one of the following:

Value	Meaning
clInsideFrame:	The caption's frame is located inside the MegaPanel's frame
clOutsideFrame:	The caption's frame is located outside the MegaPane's frame

CaptionPosition Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionPosition: TCaptionPos;
```

Default Value

```
cpTopLeft
```

Description

The position of the caption within the frame as set by [CaptionLocation](#) above. This property can be set to one of the following values: `cpTopLeft`, `cpTopCenter`, `cpTopRight`, `cpLeftTop`, `cpLeftCenter`, `cpLeftBottom`, `cpRightTop`, `cpRightCenter`, `cpRightBottom`, `cpBottomLeft`, `cpBottomCenter`, `cpBottomRight`);

Note

To correctly position the caption text as per this property's description, start with the [CaptionHorzJustify](#) property set to `hjLeft`.

CaptionSpace Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionSpace: integer;
```

Default Value

2

Description

If [CaptionLocation](#) is set to `clOutsideFrame` then this is the space between the Caption's frame and the MegaPanel's frame.

If [CaptionLocation](#) is set to `clInsideFrame` then this is the space between the Caption's frame and the `alClient` Alignment area for child controls placed in the MegaPanel.

This property is also effective when the [psGroupBox](#) style is selected.

CaptionStyle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CaptionStyle: TCaptionStyle;
```

Default Value

csNone

Description

The style of the caption. This property may be set to one of the following values:

Value	Meaning
csNone:	There is no caption and no caption frame
csPanel:	The caption will be aligned adjacent to the MegaPanel's frame edge, See also CaptionLocation and CaptionPosition
csGroupBox:	The caption will be located in the top frame of the MegaPanel similar to a standard GropuBox control

ColorCaptionBkgnd Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ColorCaptionBkgnd: TMegaColor;
```

Description

This is the color of the Background of the caption area of the MegaPanel. Set it to cINone if you want to make the caption's background [Transparent](#).

ColorDisabledLabel Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ColorDisabledLabel: TMegaColor;
```

Default Value

```
clInactiveCaptionText
```

Description

This color is used for labels associated with Disabled controls. This color overrides both the LabelFont's color and any color set using the control's [Tag](#) property.

ColorHighlight Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ColorHighlight: TMegaColor;
```

Default Value

```
clBtnHighlight
```

Description

The color to be used as the Highlighted edge for 3D effects such as Bevels and Inset or Raised text.

ColorLabelBkgnd Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ColorLabelBkgnd: TMegaColor;
```

Default Value

clNone

Description

The default color for all [Label](#) backgrounds. Does not apply to [Quicken Style Labels](#). This colour can be overridden on a control by control basis by using the control's [Tag](#) property.

ColorPanel Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ColorPanel: TMegaColor;
```

Default Value

clBtnFace

Description

The color of the MegaPanel background. To create a [Transparent](#) MegaPanel set this property to clNone.

ColorShadow Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ColorShadow: TMegaColor;
```

Default Value

clBtnShadow

Description

The color to be used as the Shadowed edge for 3D effects such as Bevels and Inset or Raised text.

MegaPanel Component

[Properties](#)

[Events](#)

[Tasks](#)

Unit

[MegaPanel](#)

Description

[MegaPanel](#) is a Delphi control that provides all the functionality of the standard Delphi TPanel control plus:

- Supports both Delphi 1.x and Delphi 2.x
- Automatically generated Panel Labels.
- Automatic Quicken Style Labels.
- Multi-line Labels and Captions
- Rotation of Labels and Captions
- 16 different Label positioning options
- 12 different Caption positioning options
- Selectable Label background colors
- Selectable Caption background colors
- Each label may be assigned a separate foreground or background color
- Transparent Labels and Panels
- 3 different Captioning styles
- Many options to control the Caption location
- Background bitmaps for Panel or Caption or Both
- 14 different Bitmap positioning options including Stretch and Tile
- 7 different Bitmap location options
- Transparent Bitmaps.
- Automatic Panel Dividers
- Automatic Child Control Centering and Resizing
- Automatic Child Control Font Resizing
- Automatic Caption Resizing and Label Resizing
- Gradient Fills in many different styles for Panel or Caption or Both
- Gradient Fills and Pictures may be used at the same time
- Automatic Keyboard Fixup to allow the Enter or Arrow keys to work like the Tab key on a Panel by Panel basis
- Special Property Editor for editing control class fixups
- Automatic Control Drop Shadow effects
- Automatic Hot Key Label generation from hint text
- Automatic Hot Keyed Control Focusing
- Label color effects for Focused control
- Label 3D effects for Focused control
- Use Any Form as a Panel. This allows you to modularize your code

Ctl3D Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Ctl3D: boolean;
```

Default Value

True

Description

The Ctl3D property determines whether a control has a three-dimensional (3-D) or two-dimensional look. If Ctl3D is True, the control has a 3-D appearance. If Ctl3D is False, the control appears normal or flat.

Note

If you wish to use the [CtlShadow](#) effect then all controls in the MegaPanel should have Ctl3d set to False.

At the time of writing, there is a **Bug in the Delphi 2.0 ComboBox** control that does not allow the Ctl3d effect to be turned off for ComboBoxes.

CtlShadow Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property CtlShadow: TShadowEffect;
```

Description

The CtlShadow property has several Sub-Properties that allow you to set a Drop Shadow effect for controls in the MegaPanel. You can specify whether a control uses the Drop Shadow effect on a Panel by Panel, Class by Class basis using the [Class Fixup Editor](#).

Cursor Property

[See Also](#)

Applies to















[TMegaPanel](#)

Declaration

```
property Cursor: TCursor;
```

Description

The Cursor property is the image used when the mouse passes into the region covered by the control. These are the possible images:

Value	Image	Value	Image	Value	Image
crDefault		crSizeNESW			
		crHourglass			
					
crArrow		crSizeNS			
		crDrag			
					
crCross		crSizeNWSE			
		crNoDrop			
					
crIBeam		crSizeWE			
		crHSplit			
					
crSize		crUpArrow			
		crVSplit			





DBFilterCombo for Delphi

DBFilterCombo for Delphi

DBFilterCombo is a Data Aware Lookup ComboBox control for Borland's Delphi. It looks and works just like the standard Delphi DBLookupCombo with a couple of notable exceptions;

DBFilterCombo will not only create a lookup list from a second database it allows the items in the lookup list to be Filtered by a third and independent LookupFilterField. The LookupFilterField has both Minimum and Maximum values which may be Inclusive or Exclusive. The LookupFilterField does NOT have to be a key.

DBFilterCombo adds records to the Lookup Datasource if the user enters a value that is not already contained in the lookup list. DBFilterCombo alerts you to this operation and allows you to programatically accept or cancel the record addition on a record by record basis.

DBFilterCombo supports Incremental List Searching with or without Case Sensitivity to allow the user to enter the first letters of an item in the list and have the rest of the field filled in automatically.

DBFilterCombo has intelligent Drop Down List placement when the drop down list is near to a screen edge.

DBFilterCombo is really great for those little Helper Tables that all Databases need. You know the ones that store things like Titles (Mrs, Mrs, Colonel etc) or Shipping Method (Air, Sea, Courier etc etc). In fact you can now store all of these little helpers in a single table and use DBFilterCombo to provide Dropdown lists Filtered by category (eg, Titles, Shipping Method, Religion etc). I am sure that you will find it equally as useful.

Try out the Shareware version available for download from the Compuserve Delphi Forum. Search for DBFLTCOM.ZIP.

Source code is available as a separate registration from Version 2.0 onwards.

DBLookupText

DBLookupText Component

The idea behind DBLookupText is simple.

Its a Label control that sources its text from a Database Field. The contents of this field is not actually displayed, but is used as a Lookup reference for a second Database. The result of the Lookup operation is used as the Text that is displayed in the Label.

DBLookupText is very useful for Display Only data. This data may have been entered into a Database using one of the Lookup components like DBLookupCombo or DBFilterCombo etc.

DBLookupText is free of charge to all registered DBFilterCombo users.

DBZipMemo

DBZipMemo Component

DBZipMemo is two components in one. In its inactive state it looks and works just like an ordinary DBMemo control. When you double click on the DBMemo control a window pops up complete with a larger Memo control to allow easier editing and viewing of the Memo details. The window is complete with Caption and Hint lines taken from the original DBMemo's Hint property. The window also has OK and Cancel buttons to allow you to accept or reject changes. When one of the buttons is pressed the window disappears and the focus returns to the original DBMemo control (or optionally to the next control in the Tab order). If the changes were accepted then the windows contents are copied to the original DBMemo control otherwise they are discarded.

DBZipMemo is free of charge to all registered DBFilterCombo users.



DataSorcerer for Delphi

DataSorcerer for Delphi

DataSorcerer gives Delphi Database Programmers the much sought after feature of a Global Datasource: The main features of DataSorcerer are:

- A single DataSorcerer and its Dataset can be accessed Globally from all form within an application.
- Links between DataSorcerers on different Forms can be created at Design Time.
- All data linkages are present at Design time to allow testing before compilation.
- There is No Code required on any Form to make the the data available.
- There is No Code required on any Form to keep the data synchronized.
- All Data Aware controls on all forms are automatically kept synchronized.
- Direct access is available to the underlying DataSet, no matter what type it is.
- Abbreviated program syntax to access database fields.
- All public TDataSource methods and properties are directly accessible.
- All public TDataSet methods and properties are directly accessible.
- All public TDBDataSet methods and properties are directly accessible.
- All public TTable methods and properties are directly accessible.
- All public TQuery methods and properties are directly accessible.
- All public TStoredProc methods and properties are directly accessible.
- The TTable Fields Editor is available from any linked DataSorcerer.
- The TQuery/TStoredProc Params Editor is available from any linked DataSorcerer.
- The Master Fields Designer is available from any linked DataSorcerer.
- Full support for InfoPower Data Access components
- The InfoPower Property Editors are available from any linked DataSorcerer.
- Exception trapping of Database errors
- Support for both Delphi V1.x 16bit and Delphi V2.x 32bit platforms
- Support for additional Delphi V2.0 database component Events, Properties and Methods
- Availability of Source Code
- Delphi Online Help compatible Help File

DataSorcerer is a descendant of the Delphi TDataSource component. If InfoPower support is enabled then DataSorcerer descends from the TwwDataSource component, which is itself a direct descendant of TDataSource.



Directory Sorter for Windows 95

Directory Sorter for Windows 95

Do you use the DOS prompt in Windows 95 ?

Are you annoyed when you archive, copy or backup files and they are not processed in Alphabetic order ?

Are you annoyed that the latest version of Norton Utilities for Windows 95 does not include a DirSort ?

Then get Directory Sorter!

Directory Sorter will physically store all the files and directories on your drives in Alphabetic order. This means that whenever you do a dir, copy, xcopy, backup or archive operation etc, all the files and directories will be processed in Alphabetic order.

The shareware version of Directory Sorter is available in the WINUTILITY and MSWIN95 forums on Compuserve. If you like the shareware version then registration is also available on-line on Compuserve.

Grab a copy and try it out.

STOP PRESS: Directory Sorter V2.0 with tons of new features is due out in May 96. Look for it in the Compuserve WINUTILITY and MSWIN95 forums, filename DIRSORT2.EXE.

DividerBevel Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property DividerBevel: TPanelBevel;
```

Default Value

bvLowered

Description

The style used for MegaPanel Dividers. This can be set to one of the following values: bvNone, bvLowered, bvRaised.

DividerWidth Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property DividerWidth: integer;
```

Default Value

2

Description

The width of each Divider Bevel.

DividersHorz Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property DividersHorz: integer;
```

Default Value

0

Description

The number of Horizontal dividers to place on the MegaPanel. The dividers are always equally spaced, so for example, if the DividersHorz was set to 2 then there would be two dividers displayed and the MegaPanel would be divided into equal thirds from top to bottom.

DividersVert Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Dividers Vert: integer;
```

Default Value

0

Description

The number of Vertical dividers to place on the MegaPanel. The dividers are always equally spaced, so for example, if the DividersVert was set to 3 then there would be three dividers displayed and the MegaPanel would be divided into equal quarters from left to right.

DragCursor Property

[See Also](#)

Applies to















[TMegaPanel](#)

Declaration

```
property DragCursor: TCursor;
```

Description

The DragCursor property determines the shape of the mouse pointer when the pointer is over a component that will accept an object being dragged. These are the possible images:

Value	Image	Value	Image	Value	Image
crDefault		crSizeNESW			
	crHourglass				
					
crArrow		crSizeNS			
	crDrag				
					
crCross		crSizeNWSE			
	crNoDrop				
					
crIBeam		crSizeWE			
	crHSplit				
					
crSize		crUpArrow			
	crVSplit				



DragMode Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property DragMode: TDragMode;
```

Description

The DragMode property determines the drag and drop behavior of a control. These are the possible values:

Value	Meaning
dmAutomatic	If dmAutomatic is selected, the control is ready to be dragged; the user just clicks and drags it.
dmManual	If dmManual is selected, the control can't be dragged until the application calls the BeginDrag method.

If a control's DragMode property value is dmAutomatic, the application can disable the drag and drop capability at run time by changing the DragMode property value to dmManual.

Enabled Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Enabled: boolean;
```

Description

The Enabled property controls whether the control responds to mouse, keyboard events. If Enabled is True, the control responds normally. If Enabled is False, the control ignores mouse and keyboard events.

MegaPanel Events

See Also

[OnClick](#)

[OnDbClick](#)

[OnDragDrop](#)

[OnDragOver](#)

[OnEndDrag](#)

[OnEnter](#)

[OnExit](#)

[OnMouseDown](#)

[OnMouseMove](#)

[OnMouseUp](#)

[OnResize](#)

MegaPanel Events

OnClick

OnDbClick

OnDragDrop

OnDragOver

OnEndDrag

OnEnter

OnExit

OnMouseDown

OnMouseMove

OnMouseUp

OnResize

Caption Properties

Label Properties

Panel Properties

Picture Properties

Resize Properties

FixClasses Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property FixClasses: TStrings;
```

Description

The FixClasses property allows you to set Enter and Arrow Key conversions and Drop Shadow effects for individual control Classes on a panel by panel basis. The [Class Fixup Editor](#) allows this to be done easily and is activated by either Double Clicking on the property or by pressing the "..." button.

FixKeys Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property FixKeys: TFixKeys;
```

Default Value

```
[fkEnter, fkUpDown]
```

Description

The FixKeys property provides a Panel Global Override for the individual Class settings for Enter and Arrow Key conversion as set using the [Class Fixup Editor](#).

If the FixKeys property does not contain the value fkEnter then the Enter Key is Not converted to the Tab Key for Any control in the MegaPanel regardless of the individual control Class Fixup settings.

If the FixKeys property does not contain the value fkUpDown then the Arrow Keys are Not converted to the Shift+Tab and Tab Keys for Any control in the MegaPanel regardless of the individual control Class Fixup settings.

GradientFill Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property GradientFill: TGradient;
```

Description

The GradientFill property selects the Type, Placement, and Colors used for gradient fills in the MegaPanel.

Height Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Height: integer;
```

Description

Height property of a control is the vertical size of the control, form, or graphic in pixels.

HelpContext Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property HelpContext: THelpContext;
```

Description

The HelpContext property provides a context number for use in calling context-sensitive online Help. Each screen in the Help system should have a unique context number. When a component is selected in the application, pressing F1 displays a Help screen. Which Help screen appears depends on the value of the HelpContext property.

If HelpContext is zero, then the control will "inherit" the help context of its parent control. For example, if a button with zero help context is contained in a form with a non-zero help context and F1 is pressed, the context screen for the form will be shown.

MegaPanel Installation

MegaPanel Installation

Now that there are two versions of Delphi in concurrent use by many programmers to support both Windows 3.x and Windows 95, it is not possible to specify exact directory locations for any of Delphi's sub-directories. The Delphi Installation program allows you to install Delphi into a directory of your own choice. The installation instructions below refer to your "Delphi LIB" directory. This LIB directory will be a sub-directory of your chosen Delphi installation directory. For example, if you installed Delphi V1.0 in the C:\DELPHI10 directory, the LIB directory is C:\DELPHI10\LIB and if you installed Delphi V2.0 in the C:\DELPHI20 directory the LIB directory is C:\DELPHI20\LIB. The Delphi BIN and HELP directories can be located in the same way.

If you have installed both Delphi V1.0 and Delphi V2.0 then you will need to install MegaPanel in the LIB sub-directories of BOTH Delphi directories. You will need to follow the instructions for installation for both versions of Delphi.

The method for adding and/or removing components from your Delphi Component Library is different for each version of Delphi. Please refer to your manuals or Delphi Help files for precise instructions on adding and/or removing components from the Delphi Component Library.

As always, before installing or removing a component from the Component Library, it is prudent to make a backup copy of the Component Library file, just in case something goes wrong and you need to revert to the old version.

Installing the Source Code Version of MegaPanel

To install the Source Code Version of MegaPanel follow these steps:

1. If you installed a previous version of MegaPanel then you will first need to Delete the MEGAPANL.DCU, MEGAPROP.DCU and MEGAPANL.DCR files from your Delphi LIB directory.
2. Copy the Source Code distribution program EXE file to a temporary directory and run this program from the Windows File Manager / Program Manager / Explorer or Windows 95 DOS Prompt. This will Un-Zip the distribution files.
3. Copy the MEGAPANL.PAS, MEGAPROP.PAS and MEGAPROP.DFM files to your Delphi LIB directory.
4. Copy the MEGAPANL.INI file to your WINDOWS directory.
- 5a. If you are using Delphi V1.0. Copy the MEGAPANL.16 file to the Delphi LIB directory.
- 5b. If you are using Delphi V2.0. Copy the MEGAPANL.32 file to the Delphi LIB directory.
6. If you previously installed V1.x, the Shareware or Component only version of MegaPanel then Remove the MEGAPANL.DCU and MEGAPROP.DCU files from the Component Library.
7. Add BOTH the MEGAPANL.PAS and MEGAPROP.PAS files to your Delphi Component.
8. When the Library is Rebuilt the MegaPanel Icon will appear in the Standard Tool Palette.

Installing the Shareware & Component Only Versions of MegaPanel

To install MegaPanel Shareware or Component Only versions follow these steps:

1. Copy the distribution MEGAPANL.EXE file to a temporary directory and run the MEGAPANL.EXE program from the Windows File Manager/Program Manager/Explorer. This will Un-Zip the distribution files.
- 2a. If you are using Delphi V1.0. Unzip the 16BIT.ZIP file.
- 2b. If you are using Delphi V2.0. Unzip the 32BIT.ZIP file.
3. Now copy the MEGAPANL.DCU, MEGAPROP.DCU and MEGAPROP.DFM files to your Delphi LIB directory.
4. Copy the MEGAPANL.INI file to your WINDOWS directory.

- 5a. If you are using Delphi V1.0. Copy the MEGAPANL.16 file to the Delphi LIB directory. If you have installed any previous versions of MegaPanel then delete the MEGAPANL.DCR file from you Delphi LIB directory.
- 5b. If you are using Delphi V2.0. Copy the MEGAPANL.32 file to the Delphi LIB directory. If you have installed any previous versions of MegaPanel then delete the MEGAPANL.DCR file from you Delphi LIB directory.
6. Add BOTH the MEGAPANL.DCU and MEGAPROP.DCU to your Delphi Component Library.
7. When the Library is Rebuilt the MegaPanel Icon will appear in the Standard Tool Palette.

Adding the MegaPanel Help file to Delphi's Online Help

To add the MegaPanel Help file to the Delphi Online Help system follow these steps:

- 1a. If you are using Delphi V1.0. Copy the MEGAPANL.HLP file to the Delphi BIN directory and copy the MEGAPANL.KWF file to the Delphi HELP directory.
- 1b. If you are using Delphi V2.0. Copy both the MEGAPANL.HLP and MEGAPANL.KWF files to the Delphi HELP directory.
2. Now add the MEGAPANL.KWF file to the Delphi Online Help keyword list using the Help Installer program supplied with Delphi. Please refer to the Delphi documentation or Online Help for precise instructions on adding keyword files to the keyword index.

Note:

The first time that you use MegaPanel's Help file within Delphi, you may need to tell the system where to find the Help file. Just use the path where you have installed it above. You should only have to do this once.

Installing the Sample Program

To install the sample program follow these steps:

1. Unzip the SAMPLE.ZIP file supplied as part of the distribution archive into a directory of your own choice.
2. Load Delphi and open the TEST.DPR project you have just unzipped.
3. Many of the features of MegaPanel can be tested at Design Time. However if you wish to see features like Resizing and Focus Effects you will need to compile the TEST.DPR Project, Run the program and have a "Play".

Note:

The sample assumes that you installed the Delphi Sample Database files when you installed Delphi. If you did not install the sample database files then you will need to use the Delphi Installer to add them to your system. The sample code assumes that the database Alias for the sample database files is DBDEMOS.

MegaPanel Label Properties

<u>ColorLabelBkgnd</u>	<u>LabelControlY</u>	<u>LabelPosition</u>
<u>LabelAngle</u>	<u>LabelFocusEffect</u>	<u>LabelSpace</u>
<u>LabelBevel</u>	<u>LabelFont</u>	<u>LabelStyle</u>
<u>LabelBevelWidth</u>	<u>LabelFontStyle</u>	<u>LabelVertJustify</u>
<u>LabelClearance</u>	<u>LabelHeight</u>	<u>LabelWidth</u>
<u>LabelControlX</u>	<u>LabelHorzJustify</u>	

Caption Properties

Panel Properties

Picture Properties

Resize Properties

Events

LabelAngle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelAngle: integer;
```

Default Value

0

Description

The rotation angle in degrees of each Label. Does not apply to [Quicken Style Labels](#). You may need to experiment a little with the [LabelPosition](#), [LabelHorzJustify](#) and [LabelVertJustify](#) properties to position the Labels exactly where you want.

LabelBevel Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelBevel: TPanelBevel;
```

Default Value

bvNone

Description

The Bevel style of the frame around each Label. This property may be set to one of the following values: bvNone, bvLowered, bvRaised.

LabelBevelWidth Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelBevelWidth: integer;
```

Default Value

1

Description

The thickness of the Bevel around each label.

LabelClearance Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelClearance: integer;
```

Default Value

2

Description

The amount of blank space between the Label's bevel and the Label's text.

LabelControlX Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelControlX: integer;
```

Default Value

3

Description

Applies only to [Quicken Style Labels](#). This is the offset in the horizontal direction of the controls Label from the left edge of the control.

LabelControlY Property

See Also

Applies to

TMegaPanel

Declaration

```
property LabelControlY: integer;
```

Default Value

3

Description

Applies only to Quicken Style Labels. This is the offset in the vertical direction of the controls Label from the top edge of the control.

LabelFocusEffect Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelFocusEffect: TFocusEffect;
```

Description

The LabelFocusEffect property allows you to set special effects for the label of the control which has the input focus.

LabelFont Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelFont: TFont;
```

Description

The font used for the labels.

Note

If you want to [Rotate the labels](#) this must be a TrueType Font. Bitmap fonts like System, MS Serif, or MS Sans Serif can not be rotated.

LabelFontStyle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelFontStyle: TFontStyle;
```

Default Value

fsNormal

Description

The special effect style of the Label font, may be one of the following settings:

Value	Function
fsNormal:	No special effects
fsRaisedLight:	The labels will appear slightly raised
fsRaisedHeavy:	The labels will appear completely raised
fsInsetLight:	The labels will appear slightly inset
fsInsetHeavy:	The labels will appear completely inset
fsDropShadow:	A shadow of the label's text is displayed behind it
fsHeavy	The label is displayed very bold

Note

The colors [ColorHighlight](#) and [ColorShadow](#) are used to create the light and shadow of the 3D effects.

LabelHeight Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelHeight: integer;
```

Default Value

0

Description

The height of all labels. Set this value to 0 to allow MegaPanel to calculate each individual Label's height based on the LabelFont and the number of lines in each individual Label.

LabelHorzJustify Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelHorzJustify: THorzJust;
```

Default Value

hjRight

Description

The horizontal justification of the all Label text. May be set to one of the following values:

Value	Function
hjLeft:	The left sides of each Label's lines of text will be aligned
hjCenter:	Each Labels's lines of text will have their centers aligned
hjRight:	The right side of each Label's lines of text will be aligned

LabelPosition Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelPosition: TLabelPos;
```

Default Value

lpLeftTop

Description

The position of the Labels relative to their owner control. May be set to one of the following values:
lpTopLeftCorner, lpTopLeft, lpTopCenter, lpTopRight, lpTopRightCorner, lpLeftTop, lpLeftCenter, lpLeftBottom, lpRightTop, lpRightCenter, lpRightBottom, lpBottomLeftCorner, lpBottomLeft, lpBottomCenter, lpBottomRight, lpBottomRightCorner

LabelSpace Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelSpace: integer;
```

Default Value

4

Description

The space between the Label's Bevel and the Label's owner control.

LabelStyle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelStyle: TLabelStyle;
```

Description

The type of Label automatically created by the MegaPanel. This property may be set to one of the following values:

Value	Function
IsNone:	No Labels will be displayed
IsInPanel:	Labels are displayed in the MegaPanel adjacent to the owner controls
IsInControl:	Quicken Style Labels. Labels are displayed inside the area of the owner controls only if the owner control has NO text entered into it. As soon as the owner control has any text entered into it then the Label will disappear. If the text is deleted from the owner control then the label will reappear. We recommend that labels using this style should be set to a different color than the text of the owner control and perhaps italicized or in a different font. This will help the user to distinguish from a label and the actual owner control's text.

LabelVertJustify Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelVertJustify: TVertJust;
```

Default Value

vjCenter

Description

The vertical justification of the all Label text. May be set to one of the following values:

Value	Function
vjTop:	The top line of text is adjacent to the top of the label frame
vjCenter:	The label's lines are centered vertically in the label frame
vjBottom:	The bottom line of text is adjacent to the bottom of the label frame

LabelWidth Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property LabelWidth: integer;
```

Default Value

75

Description

The width of the label frame. If you set this value to 0 then the MegaPanel will calculate each Label's frame width on a label by label basis based on the LabelFont.

Left Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Left: integer;
```

Description

The Left property determines the horizontal coordinate of the left edge of a component relative to its parent.

Locked Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Locked: boolean;
```

Description

The Locked property determines whether a panel is replaced by an in-place active OLE object. If Locked is False, the OLE server can replace the panel. If Locked is True and panel is aligned to one of the edges of the form (its Align property is alTop, alBottom, alLeft, or alRight), then the panel remains when an OLE object in a TOLEContainer component is activated in place.

Use Locked to prevent status bars and the like from being replaced.

MegaPanel for Delphi

MegaPanel Copyright © 1995-1996 Steve Flynn

[Component Overview](#)

[New Features in Version 2](#)

[Installation](#)

[Finding the MegaPanel Software Version](#)

[Shareware Limitations](#)

[Sample Program](#)

[Ordering the Registered Version of MegaPanel](#)

[Software License Agreement](#)

[Other products by Steve Flynn](#)

[Getting Technical support](#)

Delphi is a trademark of Borland.
Windows and Windows 95 are trademarks of Microsoft.
Norton Utilities is a trademark of Symantec.

Name Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Name: TComponentName;
```

Description

The Name property contains the name of the component as referenced by other components. By default, Delphi assigns sequential names based on the type of the component, such as 'Button1', 'Button2', and so on. You may change these to suit your needs.

Note

Change component names only at design time.

New Features of MegaPanel V2.0

New Features of MegaPanel V2.0

MegaPanel includes many added features and enhancements including the following

- Supports both Delphi 1.x and Delphi 2.x
- Gradient Fills in many different styles for Panel or Caption or Both
- Gradient Fills and Pictures may be used at the same time
- Automatic Keyboard Fixup to allow the Enter or Arrow keys to work like the Tab key on a Panel by Panel basis
- Special Property Editor for editing control class fixups
- Automatic Control Drop Shadow effects
- Automatic Hot Key Label generation from hint text
- Automatic Hot Keyed Control Focusing
- Label color effects for Focused control
- Label 3D effects for Focused control
- Use Any Form as a Panel. This allows you to modularize your code

OnClick Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnClick: TNotifyEvent;
```

Description

The OnClick event occurs when the user clicks the MegaPanel. Typically, this is when the user presses and releases the primary mouse button with the mouse pointer over the MegaPanel.

OnDblClick Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnDblClick: TNotifyEvent;
```

Description

The OnDblClick event occurs when the user double-clicks the mouse button while the mouse pointer is over the MegaPanel.

OnDragDrop Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnDragDrop: TDragDropEvent;
```

Description

The OnDragDrop event occurs when the user drops an object being dragged. Use the OnDragDrop event handler to specify what you want to happen when the user drops an object. The Source parameter of the OnDragDrop event is the object being dropped, and the Sender is the control the object is being dropped on. The X and Y parameters are the coordinates of the mouse positioned over the control.

The TDragDropEvent type points to a method that handles the dropping of a dragged object. The Source parameter is the object being dragged, Sender is the object the Source is being dropped on, and X and Y are screen coordinates in pixels.

OnDragOver Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnDragOver: TDragOverEvent;
```

Description

The OnDragOver event occurs when the user drags an object over a component. You use an OnDragOver event to accept an object so the user can drop it.

The OnDragOver event accepts an object when its Accept parameter is True.

Usually, you will want the cursor to change shape, indicating that the control can accept the dragged object if the user drops it. You can change the shape of the cursor by changing the value of the [DragCursor](#) property for the control either at design time or at run time before an OnDragOver event occurs.

The TDragOverEvent type points to a method that handles the dragging of one object over another. The Source parameter is the object being dragged. Sender is the object the Source is being dragged over. X and Y are the coordinates within the control in pixels. State is the state of the drag object in relationship to the object being dragged over. And Accept determines whether the Sender recognizes the drag object. Accept defaults to False.

OnEndDrag Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnEndDrag: TEndDragEvent;
```

Description

The OnEndDrag event occurs whenever the dragging of an object ends, either by dropping the object or by canceling the dragging. Use the OnEndDrag event handler to specify any special processing you want to occur when dragging stops. If the dragged object was dropped and accepted by the control, the Target parameter of the OnEndDrag event is set to the object that accepted the drop. If the object was not dropped successfully, the value of Target is nil.

The TEndDragEvent type points to a method that handles the stopping of the dragging of an object. The Sender is the object being dragged, Target is the object Sender is dragged to, and X and Y are screen coordinates in pixels.

OnEnter Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnEnter: TNotifyEvent;
```

Description

The OnEnter event occurs when a component becomes active. Use the OnEnter event handler to specify any special processing you want to occur when a component becomes active.

Note

The OnEnter event does not occur when switching between forms or between another Windows application and your application.

Note

When switching between container controls such as the TPanel and the TGroupBox, an OnEnter event is generated for the container when the control in the container receives focus. This event is generated before the OnEnter event for the control itself is generated. An OnExit event of the container is generated after the OnExit event of the control in a container if focus is given to another control not contained by the container. For example, if you have a group box that contains three radio buttons on a form with an OK button, if focus is currently on the OK button and you click one of the radio buttons, an OnExit is generated for the button. If you then click on the OK button, an OnExit is generated for the radio button followed by an OnExit for the group box, and then the button will receive the OnEnter.

OnExit Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnExit: TNotifyEvent;
```

Description

The OnExit event occurs when the input focus shifts away from one control to another. Use the OnExit event handler when you want special processing to occur when this control ceases to be active.

Note

The OnExit event does not occur when switching between forms or between another Windows application and your application.

Note

When switching between container controls such as the TPanel and the TGroupBox, an OnEnter event is generated for the container when the control in the container receives focus. This event is generated before the OnEnter event for the control itself is generated. An OnExit event of the container is generated after the OnExit event of the control in a container if focus is given to another control not contained by the container. For example, if you have a group box that contains three radio buttons on a form with an OK button, if focus is currently on the OK button and you click one of the radio buttons, an OnExit is generated for the button. If you then click on the OK button, an OnExit is generated for the radio button followed by an OnExit for the group box, and then the button will receive the OnEnter.

Note

The ActiveControl property is updated before an OnExit event occurs.

OnMouseDown Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnMouseDown: TMouseEvent;
```

Description

The OnMouseDown event occurs when the user presses a mouse button with the mouse pointer over a control. Use the OnMouseDown event handler when you want some processing to occur as a result of pressing a mouse button.

The Button parameter of the OnMouseDown event identifies which mouse button was pressed. By using the Shift parameter of the OnMouseDown event handler, you can respond to the state of the mouse buttons and shift keys. Shift keys are the Shift, Ctrl, and Alt keys.

The TMouseEvent type points to a method that handles mouse-button events. The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) and mouse buttons were down when the user pressed or released the mouse button that generated the mouse-button event. X and Y are the pixel coordinates of the mouse pointer, within the client area of the Sender.

OnMouseMove Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnMouseMove: TMouseMoveEvent;
```

Description

The OnMouseMove event occurs when the user moves the mouse pointer while the mouse pointer is over a control. Use the OnMouseMove event handler when you want something to happen when the mouse pointer moves within the control.

By using the Shift parameter of the OnMouseDown event handler, you can respond to the state of the mouse buttons and shift keys. Shift keys are the Shift, Ctrl, and Alt keys.

The TMouseMoveEvent type points to a method that handles mouse-move events. The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) and mouse buttons were down when the user moved the mouse. X and Y are pixel coordinates of the new location of the mouse pointer, in the client area of the Sender.

OnMouseUp Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnMouseUp: TMouseEvent;
```

Description

The OnMouseUp event occurs when the user releases a mouse button that was pressed with the mouse pointer over a component. Use the OnMouseUp event handler when you want processing to occur when the user releases a mouse button.

The OnMouseUp event handler can respond to left, right, or center mouse button presses and shift key plus mouse-button combinations. Shift keys are the Shift, Ctrl, and Alt keys.

The TMouseEvent type points to a method that handles mouse-button events. The Button parameter determines which mouse button the user pressed. Shift indicates which shift keys (Shift, Ctrl, or Alt) and mouse buttons were down when the user pressed or released the mouse button that generated the mouse-button event. X and Y are the pixel coordinates of the mouse pointer, in the client area of the Sender.

OnResize Event

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property OnResize: TNotifyEvent;
```

Description

The OnResize event occurs whenever the MegaPanel is resized while an application is running. Use the OnResize event handler when you want something to happen in your application when the MegaPanel is resized.

Ordering the Registered Version of MegaPanel

Registration Benefits

Registered users of MegaPanel are free to use it in their own Delphi programs without payment of additional license fees or royalties in accordance with the terms of the [MegaPanel Software License Agreement](#).

Registered users are entitled to bug fixes and minor version upgrades at no additional cost. All bug fixes and upgrades will be distributed via Compuserve.

By using the MegaPanel software you are agreeing to be bound by the terms of the [MegaPanel Software License Agreement](#).

Registering Online via Compuserve

You may order any of the registered version of MegaPanel via Compuserve's SWREG facility. On Compuserve use the Services menu, GO command then enter SWREG as the service name and follow Compuserve's instructions. There are no shipping charges when you order via Compuserve.

Registering via Internet

We expect that on-line registration via Internet should be available during April/May 1996. The actual date is out of our control and up to the registration service provider. Announcements of this service should be made by the provider on Compuserve, Internet and other on-line services when the registration service comes on line. There is a \$3 additional shipping charge to cover mailing of diskettes when you register via Internet. Unfortunately we have found that E-Mail delivery of Binary files via Internet is troublesome at best and we are therefore not providing E-Mail delivery via Internet at this time (until we can do it reliably to all our customers).

There are three ways to Register MegaPanel

1. Component Only Version

This version of MegaPanel IS NOT supplied with Source Code.

The Compuserve SWREG Registration ID Number for the Component Only Version of MegaPanel is **10927**, the price of this version is \$19.95 and includes delivery by Compuserve E-Mail. There is an additional \$3 shipping charge when registering via Internet.

2. Source Code Version

This version of MegaPanel IS supplied in Source Code form and provides all the same features as the Component Only version. Because this version is supplied as source code you get both the security of having access to the source and the ability to recompile or adapt the source code for different versions of Delphi or other third party products. Any changes you make to the source are still subject to the [MegaPanel Software License Agreement](#).

The Compuserve SWREG Registration ID number for the Source Code Version of MegaPanel is **10928**, the price of this version is \$69.95 and includes delivery by Compuserve E-Mail. There is an additional \$3 shipping charge when registering via Internet.

3. Component Only Upgrade Version

This version of MegaPanel IS NOT supplied with Source Code. It provides registered users of V1.x of MegaPanel with an upgrade to the Component only version of MegaPanel V2.0 at a reduced price.

The Compuserve SWREG Registration ID number for the Component Only Upgrade Version of MegaPanel is **10930**, the price of this version is \$10 and includes delivery by Compuserve E-Mail. There is an additional \$3 shipping charge when registering via Internet.

4. Source Code Upgrade Version

This version of MegaPanel IS supplied in Source Code form. It provides registered users of V1.x of MegaPanel with an upgrade to the Source Code version of MegaPanel V2.0 at a reduced price.

The Compuserve SWREG Registration ID number for the Source Code Upgrade Version of MegaPanel is **11122**, the price of this version is \$59.95 and includes delivery by Compuserve E-Mail. There is an additional \$3 shipping charge when registering via Internet.

Credit Card Orders

At the time of writing we are not providing direct purchase via credit card. This may change should demand for this service increase.

Other products by Steve Flynn

Other products by Steve Flynn

[DataSorcerer for Delphi](#)

[DBFilterCombo for Delphi](#)

[DBLookupText for Delphi](#)

[DBZipMemo for Delphi](#)

[Directory Sorter for Windows 95](#)

MegaPanel Panel Properties

ColorPanel

PanelBevelOuter

PanelBorderWidth

PanelBevelInner

PanelBevelWidth

PanelFormName

PanelBevelMargin

PanelBorderStyle

Caption Properties

Label Properties

Picture Properties

Resize Properties

Events

PanelBevelInner Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PanelBevelInner: TPanelBevel;
```

Default Value

bvNone

Description

The style of the Inner Panel Bevel may be set to bvNone, bvRaised or bvLowered.

PanelBevelMargin Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PanelBevelMargin: integer;
```

Default Value

0

Description

The space between the outer edge of the MegaPanel and the first Panel Bevel.

PanelBevelOuter Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PanelBevelOuter: TPanelBevel;
```

Default Value

bvRaised

Description

The style of the Outer Panel Bevel may be set to bvNone, bvRaised or bvLowered.

PanelBevelWidth Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PanelBevelWidth: integer;
```

Default Value

1

Description

The width of each of the Inner and Outer Panel Bevels.

PanelBorderStyle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PanelBorderStyle: TBorderStyle;
```

Default Value

bsNone

Description

The style of the outer MegaPanel Border. May be set to bsSingle which will draw a one pixel wide black frame around the MegaPanel or bsNone which omits the border frame.

PanelBorderWidth Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PanelBorderWidth: integer;
```

Default Value

0

Description

The width of the border Between the Outer and Inner Panel Bevels.

PanelFormName Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PanelFormName: string;
```

Description

The PanelFormName property allows you to select the name of a Form to be used as a Panel. The form will be automatically aligned to fit the client area of the Megapanel and, in use, looks and works just like a panel.

To achieve this result all Border Styles and Border Icons are stripped from the form before inserting it into the panel and the Form is resized to fit the client area of the MegaPanel. A menu can not be used in a form assigned to a Megapanel in this way.

This has a major advantage when working with controls like TabbedNotebook. It allows you to develop and test the individual pages for the tabbed control separately, then assign them to their correct pages automatically at run time.

For tabbed controls this also has the added advantage that each separate page comes from a separate source file and so you don't end up with one huge source PAS or DFM file but several smaller and easier to maintain files. These smaller files may also be reusable in the same or another application.

To make it easier to design your form to fit a MegaPanel, the [MegaPanel Component Editor Menu](#) now contains the Overall and Client area dimensions of the MegaPanel as well as the software Copyright and Version number information.

Note

Unfortunately it is not possible to display the form inside the MegaPanel at Design Time as this causes major problems with Delphi. Therefore the MegaPanel will remain blank at Design Time. However the PanelForm will have its Design Time Size and Position changed to imitate its placement at Run Time. At Design Time the PanelForm will also still have its borders and caption displayed, since it's not possible to remove these at design time. The imitated placement allows for the border and caption. The PanelForm's Size and Placement is updated whenever the MegaPanel is updated.

The form is of course displayed correctly in the panel at run time, without any borders or caption.

Note

A form assigned to a MegaPanel can only be displayed on the screen once. If a second MegaPanel is displayed containing the same Form then the Form will "Switch" to the most recently displayed MegaPanel. You can however assign the same form to as many MegaPanels as you like providing only one of these is displayed at one time.

ParentColor Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ParentColor: boolean;
```

Description

The ParentColor property determines where a control looks for its color information. If ParentColor is True, the control uses the color in its parent component's Color property. If ParentColor is False, the control uses its own Color property.

By using ParentColor, you can ensure that all the controls on a form have a uniform appearance. For example, if you change the background color of your form to gray, by default, the controls on the form will also have a gray background.

To specify a different color for a particular control, specify the desired color as the value of that control's Color property, and ParentColor becomes False automatically.

ParentCtl3D Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ParentCtl3D: boolean;
```

Description

The ParentCtl3D property determines where a component looks to determine if it should appear three dimensional. If ParentCtl3D is True, the component uses the dimensionality of its parent component's [Ctl3D](#) property. If ParentCtl3D is False, the control uses its own Ctl3D property. The default value is True.

By using ParentCtl3D, you can ensure that all the components on a form have a uniform appearance. For example, if you want all components on a form to appear three dimensional, set the form's Ctl3D property to True and each component's ParentCtl3D property to True. Not only will all components have a three-dimensional appearance, but if you decide you prefer a two-dimensional appearance, you only have to change the Ctl3D property of the form and all the components will become two dimensional.

To specify a different dimensionality for a particular component, specify the dimensionality (True for 3D or False for 2D) as the value of that control's Ctl3D property, and ParentCtl3D becomes False automatically

ParentFont Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ParentFont: boolean;
```

Description

The ParentFont property determines where a control looks for its font information. If ParentFont is True, the control uses the font in its parent component's Font property. If ParentFont is False, the control uses its own [Font](#) property.

By using ParentFont, you can ensure that all the controls on a form have a uniform appearance. For example, if you want all the controls in a form to use 12-point Courier for their font, you can set the form's Font property to that font. By default, all the controls on that form will use the same font.

To specify a different font for a particular control, specify the desired font as the value of the control's Font property, and ParentFont becomes False automatically.

When the ParentFont is True for a form, the form uses the value of the application's Font property.

Note

If this property is used in conjunction with the [ResizeControls](#) and [ResizeFonts](#) properties, the resulting re-sized control sizes and/or placements may be unpredictable. Do not use the ResizeControls, ResizeFonts and ParentFont properties together.

Note

Child controls which have their ParentFont property set to True will display using the MegaPanel's Parent control font. In other words the childs Parent.Parent.Font.

ParentShowHint Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ParentShowHint: boolean;
```

Description

The ParentShowHint property determines where a control looks to find out if Help Hint, specified as the value of the Hint property for the control, should be shown. If ParentShowHint is True, the control uses the ShowHint property value of its parent. If ParentShowHint is False, the control uses its own ShowHint property.

By using ParentShowHint, you can ensure that all the controls on a form either show their Help Hints or don't show them. By default, ParentShowHint is True.

If don't want all the controls to have Help Hints, set the ShowHint property for those controls you do want to have Help Hints to True, and ParentShowHint becomes False automatically.

You can enable or disable all Help Hints for the entire application using the ShowHint property of the application.

Picture Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Picture: TBitmap;
```

Description

The picture to be displayed in the MegaPanel. If you double click on this property or press the "..." button then the installed Picture Property Editor will appear allowing you to load a Bitmap (.BMP file) as the MegaPanel Picture.

A good source for Megapanel background pictures are the Texture or Tile files that come with packages like Corel, Picture Publisher, 3D-FX, TrueSpace and other similar graphics and rendering programs. Some of these programs supply files in TIF or PCX formats. We recommend Paint Shop Pro V3 as an excellent program for converting and enhancing these types of images for use with MegaPanel. When converting HiColor images for use as backgrounds we suggest that you should convert them using the Standard Windows 256 color Palette. This will prevent color palette switching when the focus changes to different windows.

MegaPanel Picture Properties

Picture

PictureStyle

PictureTransparent

PicturePosition

Caption Properties

Label Properties

Panel Properties

Resize Properties

Events

PicturePosition Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PicturePosition: TPicturePos;
```

Default Value

```
ppTile
```

Description

The Position of the Picture in the MegePanel. May be set to one of the following values: ppTopLeft, ppTopCenter, ppTopRight, ppLeftCenter, ppCenter, psRightCenter, ppBottomLeft, ppBottomCenter, ppBottomRight, ppStretch, ppTile, ppTileFast.

The options ppTile and ppTileFast both produce a tiled panel background, the difference is in the way the background is painted and the amount of memory required. The ppTile option is the slowest and most memory efficient method. In some cases where the Picture is small, the MegaPanel is large and the machine running the program is slow the ppTile option produces an unacceptable result, since you can see each tile being painted in position. The ppTileFast option paints the entire panel background in one operation and is extremely fast by comparison to ppTile. The additional memory requirement for this option is roughly (The Width of the Panel X The Height of the Panel X The Number of Bytes Per Pixel of the Picture). This is roughly 768K for a 1024 x 768 MegaPanel using a 256 color tile. The memory is only required while the MegaPanel is being painted and is freed immediately afterwards. This is also a worst case example of an entire full screen panel being repainted. I many cases the repainted area is only a portion of the panel and in these cases MegaPanel only uses the amount of memory required for the repainted area. Even so this memory overhead may be unacceptable for large MegaPanels with small tiles on machines with only a small amount of RAM and in particular machines that are only "Just Running" Windows 95.

PictureStyle Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PictureStyle: TPictureStyle;
```

Default Value

```
psAll
```

Description

The PictureStyle property defines the Caption or Panel style of the Picture and may be set to allow the Picture to fill an area as follows:

Value	Result
psAll:	The entire area of the MegaPanel
psPanel:	Only the panel area
psPanelFrame:	The area inside the panel frame including the frame
psPanelInsideFrame:	The area inside the panel frame excluding the frame
psCaption:	The entire caption area.
psCaptionFrame:	The area inside the caption frame including the frame
psCaptionInsideFrame:	The area inside the caption frame excluding the frame
psCaptionText:	The area of the caption's text only

The area inside the

The term "frame" above refers to the bevels and borders that make up the bounding frame of the Panel or Caption. The description of the picture placement above assumes that the picture is tiled or stretched, if this is not so then the picture may not entirely fill the area depending on the picture style.

PictureTransparent Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property PictureTransparent: boolean;
```

Default Value

False

Description

If this property is set to True then the MegaPanel Picture will be transparent. That is, the Picture will be displayed as defined by the other Picture Properties but any of the pixels of the Picture which are the Transparent Color will be masked out. The Picture's Transparent Color is defined as the color of the Top Left hand corner pixel of the Picture. This is the same as the Glyphs used with the standard Delphi BitBtn or SpeedButton controls.

MegaPanel Properties

See Also

<u>AutoCenter</u>	<u>ColorPanel</u>	<u>LabelControlX</u>	<u>PanelFormNam</u>
<u>Align</u>	<u>ColorShadow</u>	<u>LabelControlY</u>	
<u>Caption</u>	<u>Ctl3D</u>	<u>LabelFocusEffec</u>	<u>ParentColor</u>
<u>CaptionAngle</u>	<u>CtlShadow</u>		<u>ParentCtl3D</u>
<u>CaptionBevell</u>	<u>Cursor</u>	<u>LabelFont</u>	<u>ParentFont</u>
	<u>DividerBevel</u>	<u>LabelFontStyle</u>	<u>ParentShowHint</u>
<u>CaptionBevel</u>	<u>DividersHorz</u>	<u>LabelHeight</u>	<u>Picture</u>
	<u>DividersVert</u>	<u>LabelHorzJustify</u>	<u>PicturePosition</u>
<u>CaptionBevel</u>	<u>DividerWidth</u>	<u>LabelPosition</u>	<u>PictureStyle</u>
	<u>DragCursor</u>	<u>LabelSpace</u>	<u>PictureTranspar</u>
<u>CaptionBevel</u>	<u>DragMode</u>	<u>LabelStyle</u>	
	<u>Enabled</u>	<u>LabelVertJustify</u>	<u>ResizeEnabled</u>
<u>CaptionBorder</u>	<u>Enabled</u>	<u>LabelWidth</u>	<u>ResizeControls</u>
	<u>FixKeys</u>	<u>Left</u>	<u>ResizeFonts</u>
<u>CaptionBorder</u>	<u>FixClasses</u>	<u>Locked</u>	<u>ResizeWhileDes</u>
	<u>GradientFill</u>	<u>Name</u>	
<u>CaptionCleara</u>	<u>Height</u>	<u>PanelBevelInner</u>	<u>TabOrder</u>
	<u>HelpContext</u>	<u>PanelBevelMarg</u>	<u>TabStop</u>
<u>CaptionFont</u>	<u>LabelAngle</u>		<u>Tag</u>
<u>CaptionFontSt</u>	<u>LabelBevel</u>	<u>PanelBevelOute</u>	<u>TagUsage</u>
	<u>LabelBevelWidt</u>		<u>ShowHint</u>
<u>CaptionHorzJu</u>	<u>LabelClearance</u>	<u>PanelBevelWidt</u>	<u>Top</u>
		<u>PanelBorderStyl</u>	<u>Visible</u>
<u>CaptionLocati</u>			<u>Width</u>
<u>CaptionPositio</u>		<u>PanelBorderWid</u>	
<u>CaptionSpace</u>			
<u>CaptionStyle</u>			
<u>ColorCaptionB</u>			
<u>ColorDisabled</u>			
<u>ColorHighlight</u>			
<u>ColorLabelBkg</u>			

MegaPanel Resize Properties

ResizeEnabled

ResizeFonts

ResizeWhileDesigning

ResizeControls

Label Properties

Panel Properties

Picture Properties

Events

ResizeControls Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ResizeControls: TResizeConts;
```

Default Value

```
[rcHorizontal, rcVertical, rcCaptionFrames,  
rcPanelFrames, rcLabelFrames, rcCtlShadow]
```

Description

This property defines how the MegaPanel's child controls will be resized when the size of the MegaPanel changes. This property contains a set of boolean values as follows:

Value	Result
rcHorizontal:	When set to True all child controls will have their Widths and Left starting points resized in proportion with the new width of the MegaPanel.
rcVertical:	When set to True all child controls will have their Heights and Top starting points resized in proportion to the new MegaPanel Height.
rcScrollBars:	When set to true All TScrollBar descendant controls will have their width and height resized the same as other controls and in accordance with the above to settings. It is usually desirable to keep a scrollbar's width (for vertical scrollbars) or height (for horizontal scrollbars) constant. When this property is set to False then Vertical scrollbars will Not have their Width adjusted and Horizontal scrollbars will Not have their Height adjusted.
rcCaptionFrames:	When set to True the size of all Caption Bevels, Borders and Margins are scaled in proportion to the new MegaPanel Height.
rcPanelFrames:	When set to True the size of all Panel Bevels, Borders and Margins are scaled in proportion to the new MegaPanel Height.
rcLabelFrames:	When set to True the size of all Label Bevels, Borders and Margins are scaled in proportion to the new MegaPanel Height.
rcCtlShadow	When set to True the size of the Drop Shadows is scaled in proportion to the new Width and Height of the MegaPanel.

Set both of the rcHorizontal and rcVertical properties to True if you want to scale child control Widths and Heights simultaneously when the MegaPanel changes size.

This property does not use the Delphi ScaleControl, ScaleControls or ChangeScale methods since these methods are only able to scale a control by one factor and cannot handle separate and/or unrelated changes to the MegaPanel's Width and/or Height.

This property may be used in combination with the AutoCenter property in which case the new Left side starting point of any resized child controls will be overridden by the AutoCenter property.

Note

Many of the standard Delphi controls, for example TComboBox, have their Height bound to their font size. This means that some controls when Resized will only change their Width because their Height is locked to their font size. To overcome this problem set the [ResizeFonts](#) rfControls property to True.

Note

If this property is used in conjunction with the ParentFont property, the resulting re-sized control sizes and/or placements may be unpredictable. Do not use the ResizeControls, [ResizeFonts](#) and [ParentFont](#) properties together.

Note

[ResizeEnabled](#) overrides this and ALL other Resize properties and causes them to be ignored if it is set to False.

ResizeEnabled Property

[See Also](#)

Applies to

[IMegaPanel](#)

Declaration

```
property ResizeEnabled: boolean;
```

Default Value

True

Description

This property globally enables or disables ALL other Resize properties. When this property is set to False, ALL other Resize properties are ignored. When this property is set to True ALL other properties function as per their individual settings.

ResizeFonts Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ResizeFonts: TResizeFonts;
```

Default Value

```
[rfCaption, rfLabels, rfControls]
```

Description

This property is only effective if the [ResizeControls](#) rcVertical property is set to True. Since it is not possible to scale fonts arbitrarily by width and height, all font size changes are made by changing the font's Height in proportion to the new MegaPanel Height. The resulting new width of the font is dependant on the style of the font being used. This property contains a set of boolean values as follows:

Value	Result
rfCaption:	When set to True the MegaPanel Caption font and Caption area will be resized in proportion with the new MegaPanel Height.
rfLabels:	When set to True all automatically generated Labels will have their fonts and framing areas resized in proportion to the new MegaPanel Height.
rfControls:	When set to True all child controls will have their fonts resized in proportion to the new MegaPanel Height

Note

For this property to be effective, any font that you plan to scale must be either a TrueType font like Arial or a bitmap font with a selection of sizes like MS Sans Serif. TrueType fonts always give the best result, but bear in mind that the user must have the correct TrueType font installed on their system. Its safest to stick to the standard TrueType fonts Arial, Times New Roman, Courier, Symbol and Wingdings unless you plan to ship additional fonts with your application. Some stunning results can be achieved however by using the correct TrueType fonts to match panel background pictures. Bitmap fonts can look rather "Chunky" when scaled to large sizes (depending on the actual bitmap sizes available).

Note

If this property is used in conjunction with the [ParentFont](#) property, the resulting re-sized control sizes and/or placements may be unpredictable. Do not use the [ResizeControls](#), [ResizeFonts](#) and [ParentFont](#) properties together.

Note

[ResizeEnabled](#) overrides this and ALL other Resize properties so that they are ignored if it is set to False.

ResizeWhileDesigning Property

See Also

Applies to

TMegaPanel

Declaration

```
property ResizeWhileDesigning: boolean;
```

Default Value

False

Description

This property globally enables or disables ALL other Resize properties while in Design mode. When this property is set to False, ALL other Resize properties are ignored in Design mode. When this property is set to True, ALL other properties function as per their individual settings in Design mode.

Note

ResizeEnabled overrides this and ALL other Resize properties so that they are ignored if it is set to True.

The MegaPanel Sample Program

The MegaPanel Sample Program

A sample application is included in the distribution archive. To use the sample application just open the TEST.DPR project file from within Delphi. The sample application has only a tiny amount of additional code added to it to display all the forms at once when the program is run. All code and Forms were generated by Delphi using MegaPanels.

The sample is for the specific purpose of demonstrating some of the features of MegaPanel. You may use any of the samples in your own work if you wish. The samples were created on a HiColor display (64K Colors) at a resolution of 1024 x 768. Some of the colors may display Dithered on a 256 color display, particularly the Gradient Fills. The best results will be obtained using HiColor or TrueColor displays at the same or greater resolution. If you are using a resolution of less than 1024x768 portions of the sample forms may be off screen and you will need to drag them back on screen to see them fully.

The "Stucco" form has four MegaPanels demonstrating the use of various features like Transparent Labels, Multi-Line Labels, Multi-line Captions, background Pictures for entire Panels or Captions or Frames, different Label Styles and Locations, Tag Coloring, Gradient Fills and Focus Effects plus much more. All MegaPanels on this form use the csGroupBox Caption Style.

The "AutoCenter" form is an example of Quicken Style Labels and uses Data Aware controls. This sample assumes that you have installed the Database Demo files into their default location when you installed Delphi. If you did not do this and you wish to use this sample form then you will need to install the database demo samples using the Delphi setup program. This form demonstrates the AutoCenter property. Whenever the Form is resized, all the child controls are Automatically Re-Centered.

The "Resize and AutoCenter" form is a basically a copy of the AutoCenter example except that it has its ResizeControls Property set to True. All controls on this form will be Resized and Automatically Centered when you change the size of the form.

The "Resize Test" form contains almost all the standard Delphi V1.0 controls and is used to demonstrate and test the ResizeControls and ResizeFonts properties.

The "Tranparent Form" demonstrates the use of Transparent MegaPanels and Panel Dividers and the ppTileFast PicturePosition property.

The "Panel Captions" form demonstrates some of the MegaPanel Captioning options (other than csGroupBox), Gradient Fills, Form Panels and uses transparent MegaPanels as containers for SpeedButtons which have automatically generated Labels.

The "Group Box" form is a very simple demonstration of Quicken Style Labels in standard Edit, Memo and ComboBox controls. AutoCenter is also enabled.

The "Form 8" form is used as a Form Panel for the "Panel Captions" example above.

Shareware Limitations

Shareware Limitations

The Shareware version of MegaPanel is fully functional except that it will only operate while Delphi is running. There are no other limitations.

The registered version of MegaPanel has this limitation removed.

ShowHint Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property ShowHint: boolean;
```

Description

The ShowHint property determines if the control should display a Help Hint when the user's mouse pointer rests momentarily on the control. The Help Hint is the value of the [Hint](#) property, which is displayed in a box just beneath the control. If ShowHint property is True, the Help Hint will appear.

If ShowHint is False, the Help Hint may or may not appear. If [ParentShowHint](#) is False also, the Help Hint won't appear. If, however, ParentShowHint is True, whether or not the Help Hint appears depends on the setting of the ShowHint property of the control's parent. For example, imagine a check box within a group box. If the ShowHint property of the group box is True and the ParentShowHint property of the check box is True, but the ShowHint property of the check box is False, the check box will still display its Help Hint.

The default value is False.

Changing the ShowHint value to True automatically sets the ParentShowHint property to False.

MegaPanel Software License Agreement

MegaPanel Software License Agreement

Use of the Software

Steven M. Flynn (the "Author") grants you, the end user, a non-exclusive single-user license to use the supplied software program and all associated materials including Source Code where supplied (the "Software"). The Software and all Intellectual Property embodied in the Software always remains the property of the Author. Your use of the Software indicates your acceptance in full of the terms and conditions of this agreement.

Copying the Software

You may make copies of the Software for backup or archival purposes only. You may use the Software on one or more computers provided there is no possibility of it being used concurrently by more than one person. A separate license is required for each concurrent user of the Software.

Distribution of the Shareware Version of the Software

The Shareware version of the Software, which is limited in functionality and does not include Source Code, may be distributed royalty free in any form providing that no charge is made for the Software itself. Only a nominal fee for media and/or copying services is allowed. All files that are supplied with the Shareware version that you have received must also be included.

Distribution of the Registered Version of the Software

The Licensed or Registered version of the Software may not be distributed in any form without prior written consent from the Author. You are granted permission to include the Software in your own original works in its Compiled Form Only. Specifically this means that you may not distribute the Source Code Files or the Delphi Library Files (DCU or OBJ files) without paying additional License Fees to the Author. No further License Fee or Royalty is required for including the Software in your own original works in its Compiled form.

Derived Works

You may NOT use the Software to derive any works which are based in part or in full on the Software or portion of the Software..

Modified Versions of the Software

All updated, revised, modified or improved versions of the Software are also subject to the conditions of this license agreement and shall remain the property of the Author.

Termination of this Agreement

You may terminate this License Agreement at any time by destroying the Software along with all copies in any form.

Limited Warranty

The Software is distributed and licensed "AS IS". The Author specifically disclaims all other warranties, express or implied, including but not limited to, implied warranties of merchantability and fitness for a particular purpose, with regard to the Software.

Damages

By using the Software you do so at your own risk. In no event shall the Author be responsible for any damages whatsoever (including but not limited to, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss or any other real or consequential damages) arising out of the use or inability to use this product.

Liability

In the event of failure of the Software for any reason, the Author's sole liability shall be to refund the purchase price of the Software or to provide a corrected version of the Software, at the option of the Author.

Copyright

The Software and the Intellectual Property embodied in the Software shall always remain the property of the Author and is protected by Australian Copyright Law and International Treaty Provisions.

Agreement

Your use of the Software indicates your agreement with the above terms and conditions. If you do not agree with these terms and conditions then you should destroy all copies of the software and return the original copy of the software (where it has been supplied on diskette) within 14 days for a refund.

Consumer Rights

If any clause in the above Software License Agreement is in violation of your consumer rights under your local laws then that clause will be deemed to be excluded in your specific case without effect on the remainder of the agreement.

Finding the MegaPanel Software Version

Finding the MegaPanel Software Version

To find out the version number of your MegaPanel software just:

1. In Design mode select any Megapanel component by Left Clicking on it.
2. Now Right Click on the same MegaPanel. This brings up the Component Editor Menu.
3. Choose About MegaPanel from the popup menu.

Note

Also included on the Component Editor Menu is a useful display of the Overall and Client Area dimensions of the Megapanel. This is handy if you are using the PanelFormName property to assign a Form to a MegaPanel.

TFocusEffect Class Type

Applies to

TMegaPanel

Declaration

```
TFocusEffect = class(TPersistent)
published
    property UseEffects: boolean default False;
    property LabelBackground: TMegaColor default clNone;
    property LabelBevel: TPanelBevel default bvNone;
    property LabelFontColor: TColor default clBlack;
    property LabelFontStyle: TFontStyle default fsNormal;
end;
```

Description

TFocusEffect is the class type for the MegaPanel LabelFocusEffect property. It has the following Sub-properties and functions:

Value	Function
UseEffects	True if focus effects are used, False if not used
LabelBackground	The color of the label background for the focused control
LabelBevel	The <u>label bevel style</u> for the focused control
LabelFontColor	The color of the focused control's label font
LabelFontStyle	The <u>special effects style</u> of the focused control's label

Note

The Focus Effect settings override any label foreground or background colors set using the Tag coloring properties.

TGradient Class Type

Applies to

TMegaPanel

Declaration

```
TGradient = class(TPersistent)
published
    property Bands: integer default 64;
    property EndColor: TColor default clBlue;
    property FillType: TGradientType default gtNone;
    property StartColor: TColor default clBlack;
    property Style: TPictureStyle default psAll;
end;
```

Description

TGradient is the class type for the MegaPanel GradientFill property. It has the following Sub-properties and functions:

Value	Function
Bands	The number of color bands displayed to achieve the gradient effect
EndColor	The final color to display at the end (or center) of the gradient
FillType	The gradient fill type used. Can be one of the following: gtNone, gtHorizontal, gtVertical, gtHorzSplit, gtVertSplit.
StartColor	The initial color to display at the start (or outside) of the gradient
Style	The style placement of the gradient fill. Can be one of the following values: psAll, psPanel, psPanelFrame, psPanelInsideFrame, psCaption, psCaptionFrame, psCaptionInsideFrame, psCaptionText

TMegaColor Type

Applies to

TMegaPanel

Declaration

The TMegaColor type is used for all MegaPanel Colors.

For Delphi V1.x the TMegaColor type is a superset of the TColor type. All the standard color names of TColor are included plus the following Windows 95 specific values which are set in the Windows 95 Display properties: c3DDkShadow, c3DLight, cInfoText, cInfoBk.

For Delphi 1.x TMegaColor also defines a color name of cNone which is used for Transparent Labels and Panels.

For Delphi V2.x TMegaColor is defined as identical to the standard TColor property since the Windows 95 colors and cNone are included in the Delphi 2.x TColor type.

When developing Delphi programs to be compiled under both the Delphi 1.x 16 bit and Delphi 2.x 32 bit platforms, MegaPanel automatically adjusts for the difference in the declarations of the System Colors used by Delphi.

TShadowEffect Class Type

Applies to

TMegaPanel

Declaration

```
TShadowEffect = class(TPersistent)
published
    property Color: TColor default clGray;
    property UseShadow: boolean default False;
    property XOffset: integer default 4;
    property YOffset: integer default 4;
end;
```

Description

TShadowEffect is the Class type for the MegaPanel CtlShadow property. TShadowEffect has the following Sub-Properties:

Sub-Property	Function
Color	The Color of the Drop Shadow
UseShadow	True if Drop Shadow effect is used, False if not
XOffset	The horizontal offset of the Drop Shadow from the control
YOffset	The vertical offset of the Drop Shadow from the control

TabOrder Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property TabOrder: TTabOrder;
```

Description

The TabOrder property indicates the position of the control in its parent's tab order, the order in which controls receive the focus when the user presses the Tab key.

Initially, the tab order is always the order in which the components were added to the form, but you can change this by changing the TabOrder property. The value of the TabOrder property is unique for each component on the form. The first component added to the form has a TabOrder value of 0, the second is 1, the third is 2, and so on. These values determine where a control is in the tab order.

Each component has a unique tab-order value within its parent. If you change the TabOrder property value of one component to be the same as the value of a second component, Delphi automatically renumbers the TabOrder value for all the other components. For example, suppose a component is sixth in the tab order. If you change the component's TabOrder property value to 3 (making the component fourth in the tab order), the component that was originally fourth in the tab order now becomes fifth, and the component that was fifth becomes sixth.

If you attempt to give a component a TabOrder value greater than the number of components on the form minus one (because numbering starts with 0), Delphi won't accept the new value, but will enter the value that assures the component will be the last in the tab order.

The control with the TabOrder value of 0 is the control that will have the focus when the form first appears.

TabOrder is meaningful only if the [TabStop](#) property is True.

TabStop Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property TabStop: boolean;
```

Description

The TabStop property determines if the user can tab to a control. If TabStop is True, the control is in the tab order. If TabStop is False, the control is not in the tab order; therefore, the user can't press the Tab key to move to the control. The default value for most controls is True.

Tag Coloring

Tag Coloring

Any control which has an Automatic Label created for it by a MegaPanel can override either the Label Background or Text color. That is the Color as set by the LabelFont or the ColorLabelBkgnd properties can be overridden on a control by control basis. To do this MegaPanel uses the Tag Property of the control. If the TagUsage Property is set to tuLabelColor then the Tag is used to represent the color of the Label Text for that control. If the TagUsage Property is set to tuLabelBackground then the Tag is used to represent the Background Color of the Label for that control. This applies not only to Panel Labels but also to Quicken Style Labels.

Note

By default Delphi sets all control Tag Properties to 0. This is the equivalent of clBlack and if left unchanged all controls will override the the default Label Text or Label Background color depending on the setting of the TagUsage Property. This of course does not matter if TagUsage is set to tuNone, since in this case the control Tags are ignored. If you are using the Tag Coloring feature then please be aware that when you change TagUsage to tuLabelColor or tuLabelBackground, you must set the Tag property of ALL controls that are not intended to override the Label color to clNone. By setting a control's Tag property to clNone the MegaPanel will display the control's Label with the color as set by the MegaPanel's LabelFont or ColorLabelBkgnd properties. You can do this easily by group selecting all the desired controls (hold down the Shift Key while clicking on the controls) and setting all their Tag properties to clNone at the same time.

To allow you to more easily set the color using the Tag property a Color Property Editor is now available by either double clicking on the Tag property or pressing the "..." button that appears in the Object Inspector when the Tag Property is selected.

When TagUsage is set to tuNone all Tag properties for the MegaPanel and its child controls will be displayed in decimal, as they were before. If TagUsage is set to tuLabelColor or tuLabelBackground then all Tag properties for the MegaPanel and its child controls will be displayed in Color notation. You may enter values in the Tag property field of any control in hexadecimal, decimal or as a color name regardless of the TagUsage setting, only the way in which the values are displayed is changed when the TagUsage property is changed.

Tag Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Tag: longint;
```

Description

The Tag property is available to store an integer value as part of a component. While the Tag property has no meaning to Delphi, your application can use the property to store a value for its special needs.

MegaPanel can use the Tag property of controls to store Color information to allow an individual control to override either the Label Text color or the Label Background color. This [Tag Coloring](#) depends on the setting of the TagUsage property.

TagUsage Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property TagUsage: TTagUsage;
```

Description

This property defines how the [Tag Property](#) of the child controls of the MegaPanel will be interpreted. It can be set to one of the following values:

Value	Result
tuNone:	The value of all child controls Tag is ignored
tuLabelColor:	If any child control has a Tag with a value other than cINone then the value of that child controls Tag will be interpreted as the Color of the Label Text for that child control.
tuLabelBackground:	If any child control has a Tag with a value other than cINone then the value of that child controls Tag will be interpreted as the Color of the Label Background for that child control.

MegaPanel Technical support

Technical Support

We are always interested in your suggestions for improvements to our products. If you would like to contact us with your ideas, or if you are experiencing any problems, please E-Mail us at one of the following addresses

CompuServe: 100351,2607.

Internet: 100351,2607@compuserve.com

Because we are located in Australia we have a time difference problem with the rest of the world and can therefore only provide technical support via E-Mail.

Since it is often difficult to describe and/or duplicate problem conditions, we may ask you to E-Mail us a program sample to allow us to investigate any reported problems.

The Class Fixup Editor

[See Also](#)

The Class Fixup Editor

The Class Fixup Editor allows you to specify certain settings used by MegaPanel on a Panel by Panel and Class by Class basis. The information is displayed in a table form as follows:

Column	Function
Class Name	This column lists the Class Names of all the controls used in the selected MegaPanel.
Enter	A Check Mark in this column means that All Controls of this Class Type in the selected MegaPanel will have the Enter Key converted to a Tab. This is useful, for example, on Database Entry forms when you wish to make the Enter Key move the focus to the next Entry Field when pressed. Unlisted Class Types Default to Automatic Enter to Tab conversion. All Enter to Tab conversions can be overridden by the MegaPanel's FixKeys property
Arrow	A Check Mark in this column means that All Controls of this Class Type in the selected MegaPanel will have the Up and Down Arrow keys converted to Shift+Tab and Tab respectively. This is useful, for example, on Database Entry forms when you wish to make the Arrow Keys move the focus to the next Previous or Entry Field when pressed. Unlisted Class Types Default to Automatic Arrow Key conversion. All Arrow Key conversions can be overridden by the MegaPanel's FixKeys property
Shadow	A Check Mark in this column means that All Controls of this Class Type in the selected MegaPanel will display with a Drop Shadow effect if the MegaPanel's CtlShadow properties have been set to display Drop Shadows.

Button	Function
+	Adds a blank line to the table, allowing you to manually enter a Class Name and other details
-	Deletes the row of the table that currently holds the selection focus
Get	The Get Button will populate the table with the class names and default settings for all the TWinControl control Class Types in the selected MegaPanel. Settings that have been previously saved are not changed, only classes not currently listed are added with default settings, thus preserving any previous settings Default settings are read from the MEGAPANL.INI file
OK	Saves the settings to the selected MegaPanel
Cancel	Discards any changes made including any additions made to the table

The Relationship Between the Class Fixup Editor and the MEGAPANL.INI file

You may set up default Class Type settings in the [MEGAPANL.INI](#) file. Default settings are entered in this file under the [Class Fixups] heading as follows:

```
ClassName=xyz
```

Where:

ClassName = the name of the Class to apply the Fixups to

x = 0 for No Enter Key conversion or 1 to Enable Enter Key conversion

y = 0 for No Arrow Key conversion or 1 to Enable Arrow Key conversion

z = 0 for No Drop Shadow effect or 1 to Enable Drop Shadow effects

For unlisted classes the default settings are:

- Enter Key Conversion Enabled
- Arrow Key Conversion Enabled
- Drop Shadow Effect Enabled

The supplied [MEGAPANL.INI](#) has default settings for all the standard Delphi Class Names and several popular third party components.

The Hint Editor

The Hint Editor

MegaPanel's Hint Property Editor is based on an original idea and Freeware component by Richard Hansen.

MegaPanel's Hint Property Editor works with the inbuilt features of Delphi to allow you to produce "Tool Tip" or "Fly Over" Hints which contain multiple lines. No additional programming is required, the multi-line feature is available as a built in Delphi feature. All you need to do to use the multi-line hints (or any hint for that matter) is to set the control's ShowHint property to True.

The MegaPanel Hint Property Editor allows you to easily create Multi-Line Hints, Labels and Captions. Whenever you select the Hint property for any control or a MegaPanel Caption property from the Delphi Object Inspector a "..." button will be displayed. When you press the "..." button or double click on the property the Hint/Caption Property Editor will be displayed. The property editor has two Memo style controls which allow you to enter the required string for the Hint/Label or Caption.

For Captions, use the top Memo Box control only.

For Hints or Automatic MegaPanel Labels, the top Memo Box represents the Short Hint (Fly Over or Tool Tip hint) and the bottom Memo Box represents the Long Hint (Status Bar Hint or Automatic MegaPanel Label). You can press the Enter key in these boxes to create a new line and thus a Multi-Line Hint, Label or Caption.

The Copy button allows you to copy the contents of the Top Memo Box to the Bottom one.

When you have entered or edited the information, press the OK button (or Cancel to discard changes) and the Property Editor will copy the strings to the Hint or Caption property of the control and automatically insert Carriage Returns (#13 characters) and '|' characters in the appropriate places.

If you leave the mouse cursor sitting over the top Memo box a Tool Tip Hint will be displayed using the current settings from the Hint Editor to allow you to edit the Hint to achieve the desired Tool Tip layout.

The Status Panel at the bottom of the Hint Editor will show you what the Status Hint looks like when displayed by Delphi.

The MEGAPANL.INI File

[See Also](#)

The MEGAPANL.INI File

The MEGAPANL.INI file is located in your WINDOWS Directory and is used to store the default values for all components for use with the [Class Fixup Editor](#).

You may set up default Class Type settings in the MEGAPANL.INI file. Default settings are entered in this file under the [Class Fixups] heading as follows:

```
ClassName=xyz
```

Where:

ClassName = the name of the Class to apply the Fixups to

x = 0 for No Enter Key conversion or 1 to Enable Enter Key conversion

y = 0 for No Arrow Key conversion or 1 to Enable Arrow Key conversion

z = 0 for No Drop Shadow effect or 1 to Enable Drop Shadow effects

For unlisted classes the default settings are:

```
Enter Key Conversion Enabled  
Arrow Key Conversion Enabled  
Drop Shadow Effect Enabled
```

The supplied MEGAPANL.INI has default settings for all the standard Delphi Class Names and several popular third party components.

Note

MEGAPANL.INI is a Design Time Only file. You Do Not need to supply a copy of this file to your customers, nor is it used in any way at Run Time.

The supplied MEGAPANL.INI file contains the following entries:

```
[Class Fixups]  
TBevel=000  
TBitBtn=010  
TButton=010  
TCalendar=111  
TCheckBox=110  
TComboBox=101  
TDBCheckBox=110  
TDBComboBox=101  
TDBCtlGrid=101  
TDBEdit=111  
TDBFieldGrid=101  
TDBFilterCombo=101  
TDBGrid=101  
TDBImage=001  
TDBListBox=101  
TDBLookupCombo=101  
TDBLookupComboBox=101  
TDBLookupList=101  
TDBLookupListBox=101  
TDBLookupText=000  
TDBMemo=001  
TDBNavigator=000  
TDBRadioGroup=000  
TDBText=000  
TDBZipMemo=001  
TDirectoryListBox=101  
TDrawGrid=101  
TDriveComboBox=101  
TEdit=111  
TFileListBox=101  
TFilterComboBox=101  
TGroupBox=000
```

THeader=000
THeaderControl=000
THotKey=101
TImage=001
TLabel=000
TListBox=101
TListView=101
TMaskEdit=111
TMediaPlayer=001
TMegaPanel=000
TMemo=001
TNoteBook=000
TOleContainer=001
TOutline=001
TPageControl=101
TPanel=000
TRadioButton=110
TRadioGroup=000
TRichEdit=111
TScrollBar=000
TScrollBar=000
TShape=000
TSpeedButton=010
TSpinButton=111
TSpinEdit=111
TStatusBar=000
TStringGrid=101
TTabbedNoteBook=001
TTabControl=101
TTabSet=000
TTrackBar=100
TTreeView=101
TUpDown=101
TwwDBComboBox=101
TwwDBGrid=001
TwwDBLookupCombo=101
TwwIncrementalSearch=111
TwwKeyCombo=101

Top Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Top: integer;
```

Description

The Top property determines the Y coordinate of the top left corner of the MegaPanel, relative to its parent or containing control in pixels.

Transparent MegaPanels and Labels

Transparent MegaPanels and Labels

MegaPanels can be made Transparent, that is they do not have any background color of their own and their Parent MegaPanel's background, which would normally be obscured, can be seen thru them. This is useful if the parent MegaPanel has a Picture background that you wish to keep consistent over the whole form. Normally if each MegaPanel was assigned the same background picture, the picture would be repainted in each MegaPanel, this may cause misalignment of, for example, a textured background. Transparent MegaPanels solve this problem. The [sample application](#) has two forms, "Panel Captions" and "Transparent" which demonstrate Transparent MegaPanels. The "Stucco" form has examples of Transparent Labels and Captions.

Captions and Labels can also be made Transparent. This means that any features of the MegaPanel that would normally be hidden by the Label or Caption can be seen. (Its only the Label or Caption's background that is transparent, there is obviously no benefit in having transparent letters).

To create a transparent MegaPanel set its [ColorPanel](#) Property to clNone.

To create a transparent [Label](#) or [Caption](#) set its corresponding ColorBackground Property to clNone.

Note

Because of the way in which Windows and Delphi handle the painting of TWinControl descendant controls, a Transparent MegaPanel that is placed Over another control will cause a "Hole" to be displayed in the control. For example If a MegaPanel had a Memo control in it and you placed a small Transparent MegaPanel inside the Memo control, the Memo control would appear to have a Hole in it. The Parent Panel's background would show thru the Memo control.

In practise this is not really a problem since you are not likely to stack controls in exactly this way. The problem can be fixed by placing the transparent MegaPanel Behind the other control.

Using MegaPanel

Using MegaPanel

MegaPanel provides all the basic functionality of the Delphi TPanel component plus Mega Enhancements.

Pedigree

MegaPanel is a descendant of TCustomPanel. It declares no additional Public or Protected methods and no additional Events.

Automatic Panel Labels and Automatic Quicken Style Labels

A MegaPanel can Automatically generate Labels within the MegaPanel client area (Panel Label) or within the space of a control placed on the MegaPanel (Quicken Style Labels).

MegaPanel uses the HINT property of a control placed on it to generate a Label for that control.

Delphi provides two Hints using a control's Hint property, these are called a Short Hint and a Long Hint. The Short Hint is the part of the Hint String before the '|' character and the Long Hint is the part of the string after it. These are provided in Delphi to allow the control's Hint Property to define a "Fly Over" or "Tool Tip" style Hint (the Short Hint) and a Status Bar Hint (the Long Hint) using a single string property.

MegaPanel uses the Long Hint portion of the control's Hint Property to automatically generate a Label for the control. This means that the portion of the control's Hint Property following the '|' symbol will be used as a label.

Note

If there is no '|' in the Help Property string the the Hint Property is regarded by Delphi as BOTH the Short Hint and the Long Hint. This means that if there is no '|' symbol the "Fly Over" hint and the control's Label will be exactly the same. If you want to have only a Short Hint then place a '|' character at the end of the Hint Property string. If you want to have only a Long Hint then place a '|' character at the beginning of the Hint Property string.

MegaPanel supports Multi-Line Labels AND Captions. These are created by including Carriage Return characters in the control's Hint Property (#13 characters). MegaPanel includes a special Hint Property Editor for Hints and Captions that allows you to do this easily.

Quicken Style Labels

Automatic Quicken Style Labels are displayed when the LabelStyle Property is set to IsInControl. Quicken Style Labels are displayed inside the area of the owner controls only if the owner control has NO text entered into it. As soon as the owner control has any text entered into it then the Label will disappear. If the text is deleted from the owner control then the label will reappear. We recommend that labels using this style should be set to a different color than the text of the owner control and perhaps italicized or in a different font. This will help the user to distinguish between the label and the actual owner control's text.

Quicken Style Labels are only generated for controls which are Descendants of TWinControl. This means that some controls like SpeedButtons and Graphics Shapes CAN NOT have Quicken Style Labels. But then they dont really need them since no Data can be entered into this type of control.

Automatic Quicken Style Labels also work with Data Aware controls like TDBEdit and TDBLookupCombo etc etc.

Quicken Style labels may also be Multi-Line.

Colors

MegaPanel defines a new type of Color property which is a Superset of TColor, called TMegaColor. All Color Properties used by MegaPanel are TMegaColor Properties (except for the colors specified for Fonts using the Font Property Editor). TMegaColor includes all the colors defined by TColor plus the following additional colors:

cl3DDkShadow	Windows 95 specific
cl3DLight	Windows 95 specific
clInfoText	Windows 95 specific
clInfoBk	Windows 95 specific
clNone	MegaPanel Specific

Of these new colors the first four define colors for the corresponding features of Windows 95, for example; clInfoBk is the Windows 95 color name for the background color of "Fly Over" or "Tool Tip" windows.

The color name clNone is used by MegaPanel for two purposes:

1. MegaPanels, Panel Labels and Captions which have their Background colors set to clNone are "Transparent". See Also [Transparent Labels and Panels](#).
2. Control Tags which are set to clNone do not override the Label Color or Label Background settings for a MegaPanel. See Also [Tag Coloring](#).

Note

TMegaColor and TColor define exactly the same color set in Delphi V2.0. MegaPanel automatically handles the differences between the Delphi V1.0 version of TColor and the Delphi V2.0 version of TColor.

Visible Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Visible: boolean;
```

Description

The Visible property determines whether the component appears onscreen. If Visible is True, the component appears. If Visible is False, the component is not visible.

Width Property

[See Also](#)

Applies to

[TMegaPanel](#)

Declaration

```
property Width: integer;
```

Description

The Width property determines the horizontal size of the MegaPanel in pixels.

