Greatis Form Designer

Runtime Form Designer Component
for Borland Delphi / C++ Builder

User Manual

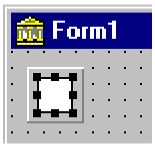
New in TFormDesigner

- **DFM support**
  Now you can read/write your form from/to DFM file both in text and binary format by LoadFromDFM and SaveToDFM procedures
- **Form streaming**
  Now you can read/write your form from/to any stream by LoadFromStream and SaveToStream procedures
- **Clipboard operations**
  Now you can copy and cut selected controls to clipboard and paste they from clipboard
- **FormData property**
  Read-write property for set/get form property (including all controls and components) by simple string.
- **ClearBeforeLoad property**
  Set this property to True if you want to delete all controls from form before loading it from DFM-file, stream or FormData property.
- **OnReadError event**
  This event handler is called when error is occured when reading the form or control from DFM-file, stream, clipboard or FormData property.

About TFormDesigner

TFormDesigner is a hi-tech component for Borland Delphi and Borland C++ Builder. It allows to move all types of controls on a form and change their size at runtime. The form and behaviour of the edited controls completely confirm the specifications of the standard form designer, which serves a component in Delphi and C++ Builder IDE.

TFormDesigner puts no additional demands to the form and no special preparation of the form needed, it's amply sufficient to place the component on a form and activate it in runtime:



```
procedure ActivateButtonClick(Sender: TObject);
begin
  FormDesigner.Active := True;
end;
```

TFormDesigner

Hierarchy

TObject – TPersistent – TComponent – TCustomFormDesigner – TFormDesigner

Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```

Standard contstructor and destructor overriden for creating/destroying internal objects and initializing properties.

```
procedure Update; virtual;
```

Call this method to update TFormDesigner after manual changing controls on the edited form.

```
function IsLocked(AControl: TControl): Boolean;
function IsProtected(AControl: TControl): Boolean;
function IsTransparent(AControl: TControl): Boolean;
```

These methods allow to determine lock and protected state and event transparency of the control.

See also

- LockedControls
- LockChildren
- LockInverse
- ProtectedControls
- ProtectChildren
- ProtectedInverse
- TransparentControls
- TransparentChildren
- TransparentInverse

```
function FindNextControl(GoForward: Boolean): TControl;
```

This function returns next or previous editable (non-protected) control.

```
function ControlAtPos(AParent: TWinControl;
     P: TPoint): TControl;
```

This function returns the child control located at a specified position within the AParent.

```
function FindWinControl(Wnd: HWND): TWinControl;
```

This function returns the TWinControl descendance from window handle.

```
procedure SaveToFile(FileName: string);
procedure LoadFromFile(FileName: string);
```

These procedures save/load controls size and position to/from INI-file. All loaded controls must be registered with RegisterClasses procedure. All components must be registered by RegisterClasses procedure.

See also

- OnSaveControl
- OnLoadControl
- RegisterClasses

```
procedure SaveToDFM(FileName: string;
     DFMFormat: TDFMFormat);
procedure LoadFromDFM(FileName: string
     DFMFormat: TDFMFormat);
```

These procedures save/load form to/from DFM-file in text and binary format. All components must be registered by RegisterClasses procedure.

```
TDFMFormat = (dfmBinary,dfmText);
```

dfmBinary	binary DFM-format
dfmText	text DFM-format

See also

- SaveToStream
- LoadFromStream
- ClearBeforeLoad
- FormData
- RegisterClasses

```
procedure SaveToStream(Stream: TStream;
     DFMFormat: TDFMFormat);
procedure LoadFromStream(Stream: TStream;
     DFMFormat: TDFMFormat);
```

These procedures save/load form to/from stream in text or binary format. All components must be registered by RegisterClasses procedure.

TDFMFormat is defined as:

```
TDFMFormat = (dfmBinary,dfmText);
```

dfmBinary	binary DFM-format
dfmText	text DFM-format

See also

- SaveToDFM
- LoadFromDFM
- ClearBeforeLoad
- FormData
- RegisterClasses

```
procedure CopyToClipboard;
procedure CutToClipboard;
procedure PasteFromClipboard;
```

These procedures are intended to work with clipboard. Clipboard format is text and is compatible with Delphi and C++ Builder IDE clipboard format, so you can copy controls between your Form Designer form and IDE form designer. All components must be registered by RegisterClasses procedure.

See also

SaveToDFM
LoadFromDFM
SaveToStream
LoadFromStream
FormData
RegisterClasses

```
procedure AlignToGrid(TheControl: TControl);
```

This procedure aligns the selected controls to the closest grid point.

See also

GridStep
SnapToGrid

```
function EditControlLists(
     DefaultList: TListType): Boolean;
```

This function calls editor for LockedControls and ProtectedControls lists. DefaultList is determines default list. TListType is defined as:

```
TListType = (ltLocked, ltProtected);
```

```
procedure AlignDialog;
```

This procedure calls dialog for align selected controls.

```
procedure SizeDialog;
```

This procedure calls dialog for resize selected controls.

```
procedure ShowAlignmentPalette;
procedure HideAlignmentPalette;
```

These procedures show/hide alignment palette window.

```
procedure AlignControls(Hor,Ver: TAlignMode);
```

This procedure aligns selected controls, using values of Hor and Ver parameters. TAlignMode is defined as

```
TAlignMode = (
     amNoChange,
     amLeftTop,
     amCenters,
     amRightBottom,
     amSpace,
     amWindowCenter);
```

amNoChange	Does not change the alignment of the controls
amLeftTop	Lines up the left/top edges of the selected controls
amCenters	Lines up the centers of the selected controls
amRightBottom	Lines up the right/bottom edges of the selected controls
amSpace	Lines up the selected controls equidistant from each other
amWindowCenter	Lines up the selected controls with the center of the window

```
procedure SizeControls(WMode: TSizeMode;
     WValue: Integer; HMode: TSizeMode;
     HValue: Integer);
```

This procedure resize all selected controls. TSizeMode is defined as

```
TSizeMode = (
     smNoChange,
     smToSmallest,
     smToLargest,
     smValue);
```

smNoChange	Does not change the size of the controls
smToSmallest	Resizes the group of controls to the height or width of the smallest selected controls
smToLargest	Resizes the group of controls to the height or width of the largest selected controls
smValue	Sets width/height of selected controls to WValue/HValue.

```
procedure Lock;
procedure Unlock;
```

These procedures lock/unlock message processing in TFormDesigner.

See also

- Locked
- OnMessage

```
procedure LeaveMouseAction;
```

This procedure stops control dragging or selecting by area immediately.

See also

- MouseAction

```
procedure AddControl(AControl: TControl);
procedure DeleteControl(AControl: TControl);
procedure SelectAll;
procedure ClearControls;
```

AddControl and DeleteControl procedures add/delete control into/from selected controls list SelectAll procedures adds all controls into the list and ClearControls deletes all controls from selected controls list.

See also

Control
Controls
ControlIndex
OnSelectControl
OnAddControl
OnDeleteControl

```
function ControlIndex(AControl: TControl): Integer;
```

This function returns control index in selected controls list. If the list does not contain AControl, function returns –1.

See also

AddControl
DeleteControl
ClearControls
Control
Controls
OnSelectControl
OnAddControl
OnDeleteControl

```
procedure DrawGrab(Canvas: TCanvas; R: TRect;
     GrabType: TGrabType); virtual;
```

This virtual procedure draws grab handles. By default this procedure draws rectangles using NormalGrabBorder, NormalGrabFill, LockedGrabBorder and LockedGrabBorder properties.

TGrabType is defined as

```
TGrabType = (gtNormal,gtLocked);
```

gtNormal	normal grabs
gtLocked	grabs around locked control

```
procedure UpdateGrabs;
```

This procedure reinitializes grabs.

Properties

public

`property Active: Boolean;`

This property contains activity state of TFormDesigner. Set Active to True to activate it.

`property Locked: Boolean;`

This read-only property contains lock state of TFormDesigner.

See also

- Lock
- Unlock
- OnMessage

`property ControlCount: Integer;`

This read-only property contains count of selected controls.

See also

- Control
- Controls
- AddControl
- DeleteControl
- ClearControls
- OnSelectControl
- OnAddControl
- OnDeleteControl

`property MouseAction: TMouseAction;`

This read-only property contains current mouse action. TMouseAction is defined as:

`TMouseAction = (maNone, maDragging, maSelecting);`

maNone	no action - simple mouse moving
maDragging	user drags control(s) now
maSelecting	user selects control(s) by area now

See also

- LeaveMouseAction

`property Control: TControl;`

This property contains current selected control or nil if no controls are selected. You can assign value to this property directly. When user selects more than one control, this property contains last selected control.

See also

- Controls
- OnSelectControl

`property Controls[Index: Integer]: TControl;`

This property contains selected controls.

See also

- Control
- ControlCount
- AddControl
- DeleteControl
- ClearControls
- OnSelectControl
- OnAddControl
- OnDeleteControl

`property ParentForm: TCustomForm;`

This read-only property contains parent form or nil if TFormDesigner has no parent form.

`property AlignmentPaletteForm: TForm;`

This read-only property contains alignment palette form. Do not free this object yourself.

`property MenuControl: TControl;`

This read-only property contains the control popup menu was called up for.

See also

- Control
- PopupMenu

`property FormData: string;`

Use this read-write property to get/set all form properties including all children controls and component. All components must be registered by RegisterClasses procedure.

See also

- SaveToDFM
- LoadFromDFM
- SaveToStream
- LoadFromStream
- RegisterClasses

published

`property MessageProcessor: TMessageProcessor;`

This property contains message processor type. TMessageProcessor is defined as:

`TMessageProcessor = (mpEvent, mpHook);`

mpEvent	TFormDesigner gets messages by Application.OnEvent handler
mpHook	TFormDesigner gets messages by Windows hook

Use mpHook only for ActiveX forms, because this algorythm has some conflicts with main form menu.

`property LockedControls: TStrings;`

This property contains list of names of controls, which cannot be moved or sized. To lock control in runtime just add his name into this list:

```
LockedControls.Add('Button1');
```

`property LockedInverse: Boolean;`

Set this property to True to inverse LockedControls contents. If LockedInverse is True, all controls are not included in LockedControls are locked.

`property LockChildren: TChildrenMode;`

This property determines lock logic for children of locked controls. TChildrenMode is defined as:

`TChildrenMode = (cmNone, cmNormal, cmRecurse);`

cmNone	only listed controls are locked
cmNormal	listed controls and all their children are locked
cmRecurse	listed controls, all children with all subchildren are locked

`property ProtectedControls: TStrings;`

This property contains list of names of controls, which cannot be edited. Controls listed in this list act in a regular way independently from TFormDesigner activity. To protect control in runtime just add his name into this list:

```
ProtectedControls.Add('Button1');
```

`property ProtectedInverse: Boolean;`

Set this property to True to inverse ProtectedControls contents. If ProtectedInverse is True, all controls are not included in ProtectedControls are locked.

`property ProtectChildren: TChildrenMode;`

This property determines protect logic for children of protected controls. TChildrenMode is defined as:

`TChildrenMode = (cmNone, cmNormal, cmRecurse);`

cmNone	only listed controls are protected
cmNormal	listed controls and all their children are protected
cmRecurse	listed controls, all children with all subchildren are protected

`property ProtectMode: TProtectMode;`

This property determines TFormDesigner action when protected control got focus. TProtectedMode is defined as:

`TProtectMode = (pmUnselect, pmLockKeyboard);`

pmUnselect	TFormDesigner unselects control(s)
pmLockKeyboard	TFormDesigner locks keyboard events for selected control(s)

`property TransparentControls: TStrings;`

This property contains list of names of controls, which cannot be selected. When user clicks in "transparent" controls, click is transferred to its parent. For example you can use this feature to move parent control when it contains child control with Align=alClient.

`property TransparentInverse: Boolean;`

Set this property to True to inverse TransparentControls contents. If TransparentInverse is True, all controls are not included in TransparentControls are "transparent".

`property TransparentChildren: TChildrenMode;`

This property determines "transparency" logic for children of "transparent" controls. TChildrenMode is defined as:

`TChildrenMode = (cmNone, cmNormal, cmRecurse);`

cmNone	only listed controls are "transparent"
cmNormal	listed controls and all their children are "transparent"
cmRecurse	listed controls, all children with all subchildren are "transparent"

`property PopupMenu: TPopupMenu;`

PopupMenu contains the pop-up menu associated with the TFormDesigner. This pop-up menu is called when user click right mouse button on the selected control.

See also
- Control
- MenuControl

`property GridStep: Integer;`

This property contains grid spacing in pixels.

`property SnapToGrid: Boolean;`

If this property is set to True minimal step of moving or sizing is GridStep.

`property DisplayGrid: Boolean;`

This property contains visibility of design grid.

`property DesignerColor: TColor;`

This property contains form background color in design mode. Set this property to clNone to save default background color.

`property GridColor: TColor;`

This property contains grid dots color. If you set this property to clNone grid color will be clWindowText.

`property ShowMoveSizeHint: Boolean;`

If this property is set to True TFormDesigner shows hint window with size/coodinate of selected control when they are moved/resized.

`property GrabSize: Integer;`

Size of grab handle. Value of this property may be between 3 and 32 pixels (5 by default).

```
property AlignmentPalette: TAlignmentPaletteOptions;
```

This property determines options of alignment palette window.

TAlignmentPaletteOptions is defined as

```
TAlignmentPaletteOption = (
     apAutoShow,
     apStayOnTop,
     apShowHints,
     apFlatButtons);
TAlignmentPaletteOptions = set of
     TAlignmentPaletteOption;
```

apAutoShow	Shows alignment palette window when TFormDesigner becomes active
apStayOnTop	Places alignment palette window above all other window of the application
apShowHints	Show hints for palette buttons
apFlatButtons	Makes palette buttons "flat"

```
property NormalGrabBorder: TColor;
property NormalGrabFill: TColor;
property LockedGrabBorder: TColor;
property LockedGrabFill: TColor;
```

Colors for border and fill of the grab handles.

```
property ClearBeforeLoad: Boolean;
```

Set this property to True if you want to delete all controls from your form before loading from DFM-file, stream or FormData property.

See also

LoadFromDFM
LoadFromStream
FormData

Events

published

```
property OnLoadControl: TLoadSaveEvent;
property OnSaveControl: TLoadSaveEvent;
```

These events occur when the control is loading/saving. You can save any additional properties. TLoadSaveEvent is defined as:

```
TLoadSaveEvent = procedure (Sender: TObject;
     TheControl: TControl; IniFile: TIniFile) of object;
```

```
property OnMoveSizeControl: TControlNotifyEvent;
```

This event occurs after the control position or/and size was changed.

See also

- MouseAction
- LeaveMouseAction
- OnMoveLimit
- OnSizeLimit

```
property OnMoveLimit: TMoveLimitEvent;
```

This event occurs when designer needs to determine limit rectangle for moving the control. By default this rectangle is bounds rectangle of control's parent control. TMoveLimitEvent is defined as:

```
TMoveLimitEvent = procedure(Sender: TObject;
     TheControl: TControl;
     var LimitRect: TRect) of object;
```

See also

- OnMoveSizeControl
- OnSizeLimit

```
property OnSizeLimit: TSizeLimitEvent;
```

This event occurs when designer sets bounds of the resized control. By default MinSize is Point(0,0) and MaxSize is Point(MaxInt,MaxInt). TSizeLimitEvent is defined as:

```
TSizeLimitEvent = procedure(Sender: TObject;
     TheControl: TControl;
     var MinSize,MaxSize: TPoint) of object;
```

See also

- OnMoveSizeControl
- OnMoveLimit

```
property OnSelectControl: TControlNotifyEvent;
```

This event occurs after the control was selected. When user unselect all selected controls, TheControl parameter is nil.

See also

- Control

```
property OnAddControl: TControlNotifyEvent;
property OnDeleteControl: TControlNotifyEvent;
```

These events occur after the control was added/deleted into/from selected controls list.

See also

- Controls
- AddControl
- DeleteControl
- ClearControls

```
property OnControlDblClick: TControlNotifyEvent;
```

This event occurs after the user double-click left mouse button on the control.

See also

- Controls

TControlNotifyEvent is defined as:

```
TControlNotifyEvent = procedure (Sender: TObject;
     TheControl: TControl) of object;
```

```
property OnActivate: TNotifyEvent;
property OnDeactivate: TNotifyEvent;
```

These events occur when TFormDesigner is activated/deactivated.

See also

- Active

```
property OnKeyDown: TKeyEvent;
property OnKeyUp: TKeyEvent;
```

These events occur when user presses/releases a key. Events handlers are called after TFromDesigner message processing, thus some keys, like cursor keys cannot be processed by these events.

See also

- ProtectMode

```
property OnMessage: TDesignerMessageEvent;
```

This event calls in internal message handler only if TFormDesigner is locked.
TDesignerMessageEvent is defined as:

```
TDesignerMessageEvent = procedure(Sender: TObject;
     var Msg: TMsg) of object;
```

See also

- Lock
- Unlock
- Locked

```
property OnReadError: TReaderError;
```

This event handler is called when reading error occurs. TReaderError is defined in Classes unit. If this event handler is not defined Form Designer do nothing when error occurs, but set Handled to True, therefore no error messages are displayed. Read error may occurs when non-registered class is reading or property value is invalid, etc.

See also

- LoadFromDFM
- LoadFromStream
- FormData
- RegisterClasses
- TReaderError
- TReader.OnError

Other declarations

DESIGNER.PAS

`EFormDesigner = class(Exception);`

Exception class used for all exceptions in TFormDesigner. You can use this type to smart exception processing:

```
try
  FormDesigner.Active := True;
except
  on E: EFormDesigner do
    ShowMessage(
      'TFormDesigner exception:'#13+
      E.Message);
end;
```

`function Designing: Boolean;`

This function returns True if any TFormDesigner in application is Active otherwise False.

Installing and uninstalling

Registered version:

Installing

1) Delphi
    a) Use "File|Open..." menu item of Delphi IDE to open Form Designer runtime package GFDLIBx0.DPK
    b) In "Package..." window click "Compile" button to compile packages GFDLIBx0.DPK
    c) Put compiled BPL file into directory that is accessible through the search PATH (i.e. DOS "PATH" environment variable; for example, in the Windows\System directory)
    d) After compiling From Designer runtime package you must install Form Designer design-time package into the IDE
    e) Use "File|Open..." menu item to open Form Designer design-time package GFDSTDx0.DPK
    f) In "Package..." window click "Compile" button to compile the package and then click "Install" button to register Form Designer component on the component palette

2) C++ Builder
    a) Be sure that linker option "Use dynamic RTL" is unchecked
    b) Use "File|Open..." menu item of C++Builder IDE to open Form Designer runtime package GFDLIBx0.DPK
    a) Use "Project|Make..." or "Project|Build..." menu item to compile package GFDLIBx0.DPK
    b) Put compiled BPL file into directory that is accessible through the search PATH (i.e. DOS "PATH" environment variable; for example, in the Windows\System directory)
    c) Use "Component|Install packages..." menu item to open "Packages" dialog box
    d) Click "Add..." button, locate GFDSTDx0.DPK
    e) Click "OK" to install package into IDE

Uninstalling

1) Use "Component|Install packages..." to open "Packages" dialog box
2) Choose "Greatis Form Designer Design-Time Package" in "Design packages" list
3) Click "Remove" button
4) Click "Yes" button to confirm removing
5) Click "OK" button to close dialog

Trial version

Installing

1) Delphi
    a) Use "File|Open..." to open GFDTRIAL.DPK
    b) Click "Install" button in "Package" window

2) C++ Builder
    a) Use "Component|Install packages..." to open "Packages" dialog box
    b) Click "Add..." button
    c) Locate GFDTRIAL.BPL
    d) Click "OK" button to install package into IDE

Uninstalling

1) Use "Component|Install packages..." to open "Packages" dialog box
2) Choose "Greatis Form Designer Trial Package" in "Design packages" list
3) Click "Remove" button
4) Click "Yes" button to confirm removing
5) Click "OK" button to close dialog

Trial limitations

In trial version of TFormDesigner:

1) Information message is displayed when
    a) installing component
    b) starting application, used TFormDesigner
    c) starting Delphi or C++ Builder with installed TFormDesigner

2) Only controls inserted into form are editable, you cannot select, move or resize controls inserted into other controls.

License Agreements

The following actions ARE ALLOWED:

1) The distribution of the products or source codes produced utilizing TFormDesigner.
2) The distribution of products or source codes produced utilizing a modified version of TFormDesigner.
3) The creation and distribution of descendant components in source code or compiled form.

The following actions ARE NOT ALLOWED:

1) The distribution of TFormDesigner, in either source or compiled form.
2) The inclusion of TFormDesigner, in source or compiled form, within other software distributions.

TFormDesigner is ROYALTY-FREE software.

Contact information

Greatis TFormDesigner

Home page:	www.greatis.com/formdes.htm
Newest documentation	www.greatis.com/formdes.pdf
Free demo	www.greatis.com/formdesdemo.zip
Order page	www.greatis.com/formdesbuy.htm
Support e-mail:	b-team@greatis.com

Greatis Software

Home page:	www.greatis.com