

TMPTaskBar

Ver. 1.1

Author: Michael Powers

taskbar@gyrix.com

Document revision: A

Contents:

1. Overview
 - 1.1 Suggested implementation
2. Properties
3. Methods
4. Events
5. License
6. Registration

1.0 Overview

Welcome to TMPTaskBar! This component will make it much easier for the users of your software to navigate various windows and documents in your interface. Unlike other taskbar components which only work with MDI-style programs, TMPTaskBar is far more versatile, enabling you to focus-switch any descendant of TForm. This control descends from TPanel, and no TPanel functionality is obfuscated.

If you have any questions or comments, please contact the author at taskbar@gyrix.com.

1.1 Suggested Implementation

The flexibility of this component allows for many configurations but this will demonstrate a common implementation, whether for MDI environments or not.

1. To start, place the component on a form and configure to you liking.
2. To the OnCreate() event (or constructor) call NewButton():

Delphi:

```
procedure NewButton(correspondingForm: TForm; buttonCaption:
                    string; glyphIndex: Integer);
```

C++Builder:

```
void NewButton(TForm* correspondingForm, AnsiString
               buttonCaption, int glyphIndex);
```

This will create a task button for the form, when it is created. Use “this” or “self” for the correspondingForm parameter. Do this for each form class that requires representation on the taskbar.

3. In the OnClose() event (or destructor) call RemoveButton():

```
procedure RemoveButton(correspondingForm: TForm);
void RemoveButton(TForm* correspondingForm);
```

Again, use “this” or “self” to indicate the button’s form association.

4. In the OnActivate() event call ButtonActive():

```
procedure ButtonActive(correspondingForm: TForm);
void ButtonActive(TForm* correspondingForm);
```

With the same technique above.

5. That’s it! TMPTaskBar does not detect MDI events, meaning you have to do a bit more work (if you call 3 lines of code work!) in order to get up and running, but it doesn’t limit you to use only in a MDI environment. (Note that above we used to MDI references.)

2.0 Properties

FlatButtons	When set to true, the buttons will have a flat appearance, a style introduced in Microsoft Office 97. When set to false, the task buttons will have the traditional appearance.
Images	Images is assigned to a TImageList which contains a list of images referenced by calls to NewButton() and UpdateIcon().
HotImages	HotImages also contains a list of images that will be displayed when UseHotGlyphs is true and the mouse is hovering over the button. The corresponding images in HotImages must have the same respective index as the images in Images.
ButtonWidth	ButtonWidth sets the default width (in pixels) of the task buttons. If there are too many buttons on screen to be accommodated by the bar at a given button width, the width will be automatically reduced.
UseHotGlyphs	When true, the task button glyph will change to the respective glyph in HotImages. When false, no action will take place during a mouse hover.

3.0 Methods

NewButton	<pre>procedure NewButton(correspondingForm: TForm; buttonCaption: string; glyphIndex: Integer); void NewButton(TForm* correspondingForm, AnsiString buttonCaption, int glyphIndex);</pre> <p>Creates a new button, and sets it to correspond with “correspondingForm”. The button caption will be set to “buttonCaption” and the Hot and Regular glyphs used from the image lists will be indexed by “glyphIndex”.</p>
RemoveButton	<pre>procedure RemoveButton(correspondingForm: TForm); void RemoveButton(TForm* correspondingForm);</pre> <p>Removes the button that is associated with “correspondingForm”.</p>
ButtonActive	<pre>procedure ButtonActive(correspondingForm: TForm); void ButtonActive(TForm* correspondingForm);</pre> <p>Signals to the taskbar that the form indicated by “correspondingForm” is now active (has focus) and will depress the associated button.</p>
UpdateIcon	<pre>procedure UpdateIcon(correspondingForm: TForm; newIndex: Integer); void UpdateIcon(TForm* correspondingForm, int newIndex);</pre> <p>Updates the glyph index on the button corresponding to the form indicated by “correspondingForm”, and sets the new image index to “newIndex”.</p>
UpdateCaption	<pre>procedure UpdateCaption(correspondingForm: TForm; newCaption: string); void UpdateCaption(TForm* correspondingForm, AnsiString newCaption);</pre> <p>Updates the caption on the button corresponding to the form indicated by “correspondingForm”, and sets the new caption to “newCaption”.</p>
Count	<pre>function Count(): Integer; int Count();</pre> <p>Returns the number of buttons on the taskbar.</p>

4.0 Events

There are no events extended beyond TPanel events.

5.0 License

Upon registration you will receive a license for the registered version. Please feel free to distribute the unregistered version, provided you do so free of charge. The Author reserves all ownership rights to this software.

6.0 Registration

Please register at <http://www.gyrix.com/taskbar> or send a message to taskbar@gyrix.com. Thank you for registering.