Documentation for TASKLIST

Original release: 16 August 1994
Minor bug fixes: 24 September 1994

0. What is it?

TASKLIST is a sample program for displaying a task list (like Window's TASKMAN) or a module list (like Microsoft's unsupported WPS), targetted at Visual BASIC programmers who want to add such capabilities to their own programs. **TASKLIST is NOT a fully-tested task list application that is ready for release into an unsuspecting world.**

To do anything useful with it, you'll need:

○ Microsoft Visual BASIC v2.0 Standard Edition or better;

○ Windows 3.1 or 3.11. 3.0 won't work, Chicago won't do.

○ The MLIST.VBX multi-column listbox control written by Robin W. McKean and the MSGBLAST.VBX subclass control written by Ed Staffin. For your convenience, these are included in this package. Both add-ons are available from CompuServe's MSBASIC forum. If you want documentation, examples and more information or registering these (shareware) controls, you can download the complete packages as MLIST4.ZIP and MSGBLA.ZIP, respectively.

TASKLIST's source code will show you how to do the following things:

○ Using Microsoft's CTL3D.DLL to give your MsgBoxes a 3D look;

○ Creating a cute small window caption* and left-aligned title while allowing your window to be moved, closed, etc.;

○ Adding an 'always on top' option to youw window;

○ Creating and displaying your own system menu; *

○ Retrieving a list of running tasks the way Windows' Task Manager does this; **

○ Using the TOOLHELP library to retrieve a list of running tasks using the internal Windows TASKENTRY structure;

○ Using the TOOLHELP library to retrieve the module list using the internal Windows MODULEENTRY structure;

○ Closing a task the "Windows way". Closing tasks and modules the "hard way";

○ Determining if a certain window is a DOS Box; **

○ Cascading/tiling and arranging windows and icons on your Desktop; **

○ Extracting icons from module files;

○ Using the TOOLHELP library to retrieve information about window classes.

Features marked with an * are based on the SMALLCAP example program that comes with the

MessageBlaster subclassing control. The original source code for that was written by Randall Kern, Jim Cash and Ed Staffin. It was significantly modified and enhanced for use in TASKLIST.

Features marked with two *'s are based on information from the excellent book "Undocumented Windows" by Andrew Schulman, David Maxey and Matt Pietrek as published by Addison Wesley (ISBN 0-201-60834-0). I wouldn't be surprised if there was a second edition out by now.

1. Legal Stuff

You may freely use and distribute TASKLIST, its source code and documentation as long as:

□ You use the abovementioned materials in a friendly and lawful way;

□ You don't sell TASKLIST or use it as added value in a sales transaction;

□ You leave all copyright notices in distributed TASKLIST code and documentation intact;

□ You don't distribute modified copies of TASKLIST in any way, other as a part of a source code distribution of your own programs.

There is no license fee for TASKLIST for personal use. If it is useful in any way to you, however, you're invited to send a nice postcard to the author (see section 3 for address details).

Corporate and other commercial users are **required** to pay a US$ 10 license fee for using TASKLIST or any part of TASKLIST in their applactions. This fee should be paid via CompuServe's SWREG service (GO SWREG at any prompt and register using ID# 3195). Corporate and commercial users who don't have access to CompuServe will be able to get away with just sending a postcard as mentioned above.

PRODUCT WARRANTY: "If you break it, you own both parts". The author provides no warranty whatsoever about functionality, stability, usability, merchantability or suitability for a certain purpose. IN NO EVENT WILL THE AUTHOR, AND/OR THE AUTHORS OF THE 3RD-PARTY ADD-ONS USED IN THIS PRODUCT, BE LIABLE FOR ANY DAMAGES RESULTING FROM THE PERFORMANCE, USE OF, OR INABILITY TO USE THIS CODE. ***Anyone using this code in their own applications is solely responsible for the consequences of doing so.***

Also, if you decide to use this code in your own programs, you are **solely responsible for complying with the license agreements of the 3rd-party products included with this code**. Instructions for obtaining the full versions of these products are at the beginning of this document. Please note that registering MLIST.VBX is **required** before you can redistribute it and that MSGBLAST.VBX has specific license terms regarding redistribution.

Microsoft, Windows and Visual Basic are registered trademarks of Microsoft, Inc.; CompuServe is a registered trademark of H&R Block.

2. TASKLIST implementation notes

TASKLIST uses a relatively large number of API calls and custom controls to achieve its goals. Below is a short summary of how the most important things in TASKLIST are done, and where in the program the implementation is hiding:

CTL3D processing
Is set up in the main() procedure and de-initialized in the DoEnd() procedure (both in TASKLIST.BAS). The reason we can't just use End to exit the program while CTL3D is active is that the CTL3D library expects to be able to subclass a number of windows. By just destroying those windows without telling CTL3D about it, we'll generate a GPF or a stack fault.

Small window caption
The caption itself is simply implemented in the Form_Draw procedure. The window title (left aligned) is just a label near the top of our (captionless) window. Its BackStyle is set to 'Transparent', so it will look good no matter what color the caption is.

The difficult part about the caption stuff is not the drawing itself, but getting our 'fake' caption to work like a real one. To do this, we'll need to intercept some messages that Windows sends to our window. Here is where the MessageBlaster control comes into play. If you look at its setup-code (in Form_Load) , you'll see that we 'trap' a number of messages. These messages are handled in the Msg1_Message event. Looking at the comments in that code should give you a pretty good idea of what is going on.

Window 'floating'
The key to making our window 'float' above all the others ('always on top') is the SetWindowPos API call. We use this to set the Z-Order of our window either to TOPMOST or NON-TOPMOST. The 'toggling' of the relevant menu option is handled in Msg1_Message.

Building a task list
TASKLIST shows two ways to build a task list. The Sub BuildTaskList in TASKLIST.FRM does it almost exactly the same way as Window's TASKMAN. The output, however, is different because TASKLIST also shows 'hidden' or invisible windows (this is sometimes confusing, but very interesting).

Sub BuildTaskList2 takes another approach: instead of enumerating the top-most windows, it walks the internal Windows' list of tasks. This list is available through the TOOLHELP API, and is explained by source comments.

Both BuildTaskList and BuildTaskList2 use the Sub AddTaskEntryToList once they've retrieved a usable TASKENTRY structure.

Building a module list
BuildModuleList is very similar to BuildTaskList2 (which is to be expected), only the displayed information is different (which should not come as a complete surprise either). BuildModuleList handles the adding of an item to the listbox itself.

Extracting an icon from a module file
In order to display a task's or module's icon, AddTaskEntryToList and BuildModuleList use the Windows API function ExtractIcon to retrieve the handle to this icon. The icon is then drawn onto a VB Picture control and the Image property of that control (which is a persistent bitmap) is added to the MList (the MList does not 'understand' the Picture property directly). After using it, the icon it is destroyed again, freeing up its memory.

Switching to a task, cascading and tiling windows
Just like Windows' TASKMAN.EXE, TASKLIST simply uses three undocumented Windows functions to achieve these tasks. Take a look at the _Click events to see how trivial it is.

Closing and 'nuking' tasks
'Closing' a task posts a WM_QUIT message to the application's window, which should be sufficient in most cases. If the user is trying to close a DOS box (this is checked using the

undocumented IsWinOldApTask Windows API call) he/she is asked for confirmation first, as closing a DOS task without its cooperation is generally not a good idea.

For those tasks that prefer to stick around even after the WM_QUIT message, TASKLIST offers the 'Nuke' option. This will call the TOOLHELP TerminateApp function, which will cause the app to do a 'FatalAppExit'. This might leave some system resources unfreed and is thus not recommended.

Window class information
Displaying the window class information for a certain task or module involves looking up all entries in the window class list that have an instance handle that matches the hModule of the specified task. The code that shows how to do this is in the ClassInfo_Click event and should be self-explaining.

Notes on Task List refreshes
Because it is very possible that processes are started and/or terminated while TASKLIST is running, it refreshes its list every time it is activated ('gets the focus'). Due to the 'kludged' nature of our window, the normal VB _GetFocus event doesn't work reliably: therefore the refresh is done upon receiving a WM_NCACTIVATE message.

As an experiment, TASKLIST also does a refresh after receiving the (undocumented) WM_OTHERWINDOWCREATED or WM_OTHERWINDOWDESTROYED message. This does not seem to work, however. Please let me know about your experiences with receiving this messages!

Release notes for version 1.01
Version 1.01 corrects a number of problems with the 1.0 code. There was one major problem with the "Get Class Info" code: because I forgot a 'byval' clause in a function declaration, the information this function displayed was partially incorrect. Thanks to Jonathan Wood for pointing this out, as well as attending me to the fact that I could eliminate the use of VBASM by just passing the wc structure directly instead of creating a pointer to it first.

All other problems were relatively minor. One issue was that you could not end the program by pressing the stop button from VB. I corrected this by removing the call to RefreshList_Click from the WM_NCACTIVATE handler and adding the NO3D command line parameter, which disables CTL3D handling. This parameter is in the supplied TASKLIST.MAK by default.

3. Contacting the author

Send all your bug-reports, comments, postcards, etc. to:

michiel de bruijn
PO Box 22396
NL-3003 DJ ROTTERDAM
The Netherlands

Internet E-mail: mdb@vbd.nl
CompuServe: 100021,1061

<EOF>