Visual Basic Office

For Help on Help, Press F1

Visual Basic Office

Visual Basic Office (**VBOffice**) is an add-on to the Visual Basic for Windows development environment versions 2.0 and 3.0. **VBOffice** features a compact buttonbar and numerous popup windows which provide quick file access, programmable undo capability, monitoring of window handles, SpyMode, a Code Librarian, a Code Typist and more. **VBOffice**'s features include...

 Quick File Access

 The Handle List

 The Code Librarian

 One-Click Save & Run

 The Code Typist

 HotKey Functionality

 Project File Access

 Full Configurability

 File Import/Export

 The ClipBoard Monitor File Manager

See Also:

Quick Start

Quick Start

If you're viewing this help file you've already managed to extract the program from it's source ZIP file. All other program files can also be located in the **VBOffice** program directory. You can place VBOFFICE.EXE in the Program Manager group of your choice using the File New command from Program Manager. If you place a MAK file as the command line parameter in your Program Manager item, **VBOffice** will load that project into Visual Basic after it's started. Double click the icon to start Visual Basic and **VBOffice**.

The Buttonbar is the key to accessing all **VBOffice** features. It works like any other buttonbar, with each button yielding a different feature. A single click on a buttonbar button pops up the corresponding window, which you can use to perform the desired action. Each pushed button stays down as long as its window is visible; If you click on a pushed button the corresponding window will receive the focus.

You can also access each button using the HotKey function. **VBOffice** has twelve buttons arranged in three groups of four, just like the F1 through F12 keys at the top of a standard 101-key enhanced keyboard. Pressing [Shift]+[Alt] plus the corresponding function key pushes down the corresponding button and pops up the desired window. (You can disable this functionality if **VBOffice**'s HotKey capability conflicts with other similar applications on your system.)

To save resources you can unload a popup window by double-clicking the system box; to save time you can simply hide it, leaving it loaded for quicker access again later.

The Help Files, Document Files and Program windows can launch a help, document or program file when you double-click a list item or highlight it and press [Enter]. You can add, delete and modify list items from the List menu on these windows using menu command or accelerator keys.

The Code Typist and Code Librarian windows allow you to retrieve text from the current VB code window or to type text into it. The Code Typist works with small text fragments, while the Code Librarian works with entire Subs or Functions. Either is suitable for blocks of declarations code.

You can also edit the saved LIB and TYP files easily using the List | Edit command, and you can import and export LIB and TYP files from other directories, making it possible to share Code Library and Code Typist files with other users of **VBOffice**. (See Importing and Exporting Files .)

The Options menu on each popup window contains a number of configuration selections you can choose to make each window work the way you want it to. You can set the font size and boldness, the behavior of the box after accessing a file and other list management and display options.

You can work with saved ClipBoard text using the Clipboard Monitor File Manager . You can view a list of window handles or "spy" on each window using the various operating modes of the Handle List .

The Configuration window allows you to set the style and location of the toolbox, the maximum number of ClipBoard Monitor files to retain and whether Visual Basic closes when **VBOffice** is shut down.

See Also:

Technical Notes

The Save/Run Button

The Save/Run button saves all the files in your project before running it. This can be helpful if you're working with API calls in your program or if other aspects of your system are unstable. While your program is running the button remains down and the small right arrow is colored red. Another click on this button switches your program into break mode and the button pops up. Whenever your application is stopped or shut down or in break mode the button pops back up and the arrow turns green.

Visual Basic 3.0 provides a save before run option in it's Options | Environment section. It is recommended that you leave this option disabled so you can decide on a case-by-case basis whether to save your project before running it. If you wish to save your project before running it, use the **VBOffice** Save/Run button; if not, use VB's run button.

Accessing Help Files

The Help Files popup window provides you with a list of help files which can be instantly opened with a double-click or by highlighting the desired file and pressing [Enter]. **VBOffice** initially adds a few help files to this window when you first run it, but you can add, delete or modify these entries as you wish. The files are displayed in alphabetical order. Double-clicking on the entry for a help file which is already open will set focus to that help file.

The List Menu:

Add:	Pops up a common dialog box allowing you to locate and select one or more help files for addition to the list. You are then prompted to enter a list caption for each item. You can also invoke this action by keying the [Ins] key.
Delete:	Deletes the currently highlighted Help file item from the list.
Modify:	Allows you to modify the file name and caption for the currently highlighted help file item using the Modify dialog box. The Browse button also allows you to use a common dialog box to locate a new file.
Open:	Opens the selected help file
Help:	Displays this help screen.
Printout:	Creates and opens a text file containing a printout of all list items managed by **VBOffice**. This can be helpful in evaluating your system configuration.
About:	Displays the current version number and other information about **VBOffice**.
Hide:	Hides the Help Files popup window. The window remains loaded for quicker access. (This is the same as clicking the "Hide" item on the window's main menu.)
Close:	Closes (unloads) the Help Files popup window. This saves on system resources but is a little slower.

See Also:

The Popup Window Options Menu

Technical Notes: Popup Windows

Accessing Document Files

The Document Files popup window provides you with a list of document files which can be instantly opened with a double-click. **VBOffice** initially adds a few files to this window when you first run it, but you can add, delete or modify these entries as you wish. The files are displayed in alphabetical order. Double-clicking on the entry for a document file which is already open will set focus to that help file.

The Document Files window attempts to find the program associated with the file by looking in WIN.INI. For example, TXT files are normally opened by NotePad, WRI files by Windows Write, DOC files by Word for Windows and so on. You're not limited to document/text files of this type; you can directly add any file to the list which has a file association recorded in WIN.INI. (See **Notes**, below.)

The List Menu:

Add: Pops up a common dialog box allowing you to locate and select one or more document files for addition to the list. You are then prompted to enter a list caption for each item. You can also invoke this action by keying the [Ins] key.

Delete: Deletes the currently highlighted Document file item from the list.

Modify: Allows you to modify the file name and caption for the currently highlighted Document file item using the Modify dialog box. The Browse button also allows you to use a common dialog box to locate a new file.

Edit: Opens the selected Document file using the associated program if possible. Usually this means opening a document with it's associated word processor or text editor.

Help: Displays this help screen.

Printout: Creates and opens a text file containing a printout of all list items managed by **VBOffice**. This can be helpful in evaluating your system configuration.

About: Displays the current version number and other information about **VBOffice**.

Hide: Hides the Document Files popup window. The window remains loaded for quicker access. (This is the same as clicking the "Hide" item on the window's main menu.)

Close: Closes (unloads) the Document Files popup window. This saves on system resources but is a little slower.

Notes:

Technically a "document file" is any file which Windows associates with a program. This window was originally designed to accommodate text files and files created by Windows Write and Word for Windows, but if a file association exists for a file there's no reason why it can't be added directly to this list and accessed the same way.

See Also:

The Popup Window Options Menu

Technical Notes: Popup Windows

Running Programs

The Programs popup window provides you with a list of programs which can be launched instantly with a double-click. **VBOffice** initially adds a few files to this window when you first run it, but you can add, delete or modify these entries as you wish. You can also specify command-line options for these programs using the List | Modify command. The files are displayed in alphabetical order. Double-clicking on the entry of a program which is already open will open another instance of the program.

The List Menu:

Add:	Pops up a common dialog box allowing you to locate and select one or more program files for addition to the list. You are then prompted to enter a list caption for each item. You can also invoke this action by keying the [Ins] key.
Delete:	Deletes the currently highlighted Program item from the list. You can also invoke this action by keying the [Del] key.
Modify:	Allows you to modify the file name and caption for the currently highlighted Program item using the Modify dialog box. The Browse also button allows you to use a common dialog box to locate a new file.
Run:	Runs the selected program.
Help:	Displays this help screen.
Printout:	Creates and opens a text file containing a printout of all list items managed by **VBOffice**. This can be helpful in evaluating your system configuration.
About:	Displays the current version number and other information about **VBOffice**.
Hide:	Hides the Programs popup window. The window remains loaded for quicker access. (This is the same as clicking the "Hide" item on the window's main menu.)
Close:	Closes (unloads) the Programs popup window. This saves on system resources but is a little slower.

See Also:

The Popup Window Options Menu

Technical Notes: Popup Windows

The Project Files Window

The Project Files popup window helps you manage Visual Basic project files. These files come in two categories: Project Files which you can add to your project and Make Files which can be opened with Visual Basic.

The Project Files Window provides you with a single list containing both types of files. Make files are indicated by the [MAK] keyword after the file caption; all other files display the caption only in the menu list.

Project Files (*.VBX, *.FRM, *.BAS, etc.) are files which can be added to your project with a double-click. Most commonly this includes custom control (VBX) files, but you can also have module (BAS) and form (FRM) files in this list as well. The files are displayed in alphabetical order by caption.

Upon initial setup **VBOffice** scans your WINDOWS\SYSTEM directory (or another directory you specify) and adds all the custom control files found to this window. You can add, delete or modify the filenames and captions of these entries as you wish

Make Files (*.MAK) are the files which you open in Visual Basic to open a programming project. Make files have the suffix [MAK] entered after their list caption.

If you double click on a Make File in the Project Files window list, **VBOffice** will instruct Visual Basic to open that project. If your current project has changed, you'll be prompted to save it.

The List Menu:

Add: Pops up a common dialog box allowing you to locate and select one or more Project or Make Files for addition to the list. You are then prompted to enter a list caption for each item. You can also invoke this action by keying the [Ins] key.

Delete: Deletes the currently highlighted Project File from the list. You can also invoke this action by keying the [Del] key.

Modify: Allows you to modify the file name and caption for the currently highlighted Project File item using the Modify dialog box. The Browse also button allows you to use a common dialog box to locate a new file.

Add to: Adds the Project File to the current project.

Help: Displays this help screen.

Printout: Creates and opens a text file containing a printout of all list items managed by **VBOffice**. This can be helpful in evaluating your system configuration.

About: Displays the current version number and other information about **VBOffice**.

Hide: Hides the Project Files popup window. The window remains loaded for quicker access. (This is the same as clicking the "Hide" item on the window's main menu.)

Close: Closes (unloads) the Project Files popup window. This saves on system resources but is a little slower.

See Also:

The Popup Window Options Menu

Technical Notes: Popup Windows

The Code Typist

The Code Typist popup window presents you with a list of Code Fragment items. Double-clicking on a list item types it's text into the currently active Visual Basic code window at the current cursor location.

The first item in the list, "[Get Selected Text]", allows you to add Code Fragment items to the Code Typist list. Simply select the desired text and double click this item. You'll then be prompted for a list caption for this item, after which it will be added to the list.

The second list item, "[New Code Fragment]", allows you to create a new code fragment using your default text editor. When you're done, save the file and **VBOffice** does the rest, prompting you for a caption and adding the new fragment to the list.

All code fragment files are stored as text files with the TYP file extension in the VBOFFICE.LIB directory and are managed by **VBOffice**. You should not attempt to alter or delete these files unless you do so through **VBOffice**. All items are displayed in alphabetical order.

The List Menu:

Delete: Deletes the currently highlighted Code Fragment item from the list. You can also invoke this action by keying the [Del] key. **NOTE: This also deletes the corresponding TYP file from the VBOFFICE.LIB directory!**

Modify: Allows you to modify the caption for the currently highlighted Code Fragment item using the Modify dialog box. Note that you can't modify the file name, since this is managed and controlled by **VBOffice**.

Edit: This allows you to edit the file containing the code fragment using NotePad. The first two lines in the file are header lines, each ending with "DO NOT MODIFY THIS LINE." These lines should **not** be modified, moved or erased.

Type To: This actually types the contents of the Code Fragment File to the current VB Code Window. Text is inserted at the cursor location within the current sub, function or declarations section.

File: Provides access to three submenu items; Import, Export and Printout. The Printout function creates and opens a text file which reports on all file information currently tracked by **VBOffice**, including their full location and the popup window menu caption assigned. For more information on the Import and Export functions, see Importing and Exporting TYP or LIB files.

Help: Displays this help screen.

About: Displays the current version number and other information about **VBOffice**.

Hide: Hides the Code Typist popup window. The window remains loaded for quicker access. (This is the same as clicking the "Hide" item on the window's main menu.)

Close: Closes (unloads) the Code Typist popup window. This saves on system resources but is a little slower.

See Also:

The Popup Window Options Menu

Technical Notes: Popup Windows

The Code Librarian

The Code Librarian popup window presents you with a list of Subs and Functions which have been placed into the Code Library. Double-clicking on an item pastes the sub or function to the bottom of the [Declarations] section of the currently active Visual Basic code window.

The first item in the list, [Get Current Procedure], allows you to add Subs, Functions and blocks of Declarations text to the Code Library. When you double click this item the entire text of the currently active Sub, Functions or block of Declarations text is obtained for addition to the Code Librarian List. You'll then be prompted for a list caption for this item, after which it will be added to the Library.

All Code Librarian files are stored as text files with the LIB file extension in the VBOFFICE.LIB directory and are managed by **VBOffice**. You should not attempt to alter or delete these files unless you do so through **VBOffice**. All items are displayed in alphabetical order by caption.

The List Menu:

Delete: Deletes the currently highlighted Code Library item from the list. You can also invoke this action by keying the [Del] key. **NOTE: This also deletes the corresponding LIB file from the VBOFFICE.LIB directory!**

Modify: Allows you to modify the caption for the currently highlighted Code Fragment item using the Modify dialog box. Note that you can't modify the file name, since this is managed and controlled by **VBOffice**.

Edit: This allows you to edit the file containing the code fragment using NotePad. The first two lines in the file are header lines, each ending with "DO NOT MODIFY THIS LINE." These lines should **not** be modified, moved or erased.

File: Provides access to three submenu items; Import, Export and Printout. The Printout function creates and opens a text file which reports on all file information currently tracked by **VBOffice**, including their full location and the popup window menu caption assigned. For more information on the Import and Export functions, see Importing and Exporting TYP or LIB files.

Add To: This adds the selected Sub or Function to the current VB Code Window. The text is inserted at the bottom of the module's [Declarations] section; if it's a new sub or function it'll add itself to the list for that module; if the block of text is declarations it'll remain at the bottom of the declarations section for that module.

Help: Displays this help screen.

About: Displays the current version number and other information about **VBOffice**.

Hide: Hides the Code Librarian popup window. The window remains loaded for quicker access. (This is the same as clicking the "Hide" item on the window's main menu.)

Close: Closes (unloads) the Code Librarian popup window. This saves on system resources but is a little slower.

See Also:

The Popup Window Options Menu

Technical Notes: Popup Windows

The ClipBoard Monitor File Manager

When the ClipBoard Monitor feature is enabled, **VBOffice** stores each piece of text copied to the Clipboard in a text file known as a ClipBoard Monitor (CLM) File. CLM files are stored in the VBOFFICE.LIB directory and are managed using the ClipBoard Monitor File Manager.

Managing CLM Files

The ClipBoard Monitor File Manager.allows you to manage ClipBoard Monitor (CLM) files saved by **VBOffice**. The ClipBoard Monitor File Manager displays all these files in date order, newest first. The resizeable split window display allows you to view a list of CLM files in the top pane and the contents of the currently selected file in the bottom pane. As each new piece of text is cut or copied to the Clipboard, the ClipBoard Monitor File Manager adds the corresponding file name to the top of the list.

To delete one or more CLM files, select the file or files to delete and click the Delete button (or key the [Del] key). This will delete the currently highlighted file(s) from the VBOFFICE.LIB directory.

Viewing and Retrieving Code

You can use the ClipBoard Monitor File Manager to view CLM files and retrieve some or all of their text to the ClipBoard. Simply click on (or key to) a file in the list and it's contents are displayed in the text window beneath. Select the text you wish to retrieve and click the Copy button to re-copy the text back to the ClipBoard. (If you wish to retrieve all of the text, simply click the Copy button or double-click the list item without selecting any text.) Once the text has been copied to the Clipboard you can paste it back into your project.

Searching for Code

You can also search through all your ClipBoard Monitor files to find the module containing a code fragment you remember. You initiate a search the same way you search for code in Visual Basic. Key [Ctrl+F] or [F3] to initiate a search, [F3] to search again, and [Shift+F3] to search backwards. As each phrase is located it's file is highlighted in the file list and it's contents are displayed in the bottom pane where the found text is highlighted.

Configuring the Clipboard Monitor

The Clips... button allows you to specify the maximum number of ClipBoard Monitor files **VBOffice** should retain. This allows you to control how much disk space is consumed by CLM files. The higher the number, the more files you retain and the greater your undo capability. The lower the number, the fewer files you retain and the more disk space you save. If you reset the maximum clipfile count to a number lower than the quantity of files currently in storage, **VBOffice** will **delete** the oldest files in excess of the new number. If you set this value to zero, Clipboard Monitoring is disabled. (You can also specify this value in the Configuration Window.)

As with all **VBOffice** popup windows, the ClipBoard Monitor File Manager remembers it's location, size and configuration from session to session. You can also set options from the Options menu which are similar to those in the Options menu of the various popup windows.

See Also:

Configuration

The Popup Window Options Menu

Technical Notes: The ClipBoard Monitor File Manager

The Handle List

The Handle List form provides a quick way to peek under the hood of Windows. This form has two modes of operation; Handle List Mode and SpyMode.

Handle List Mode

When in Handle List mode this form displays a list of all visible windows, their task IDs and handles. This can be helpful in evaluating and debugging subs or functions which make Windows API calls.

The Handle List can help you find a specific window. A single click on the window caption desired will set focus to that window. Conversely, the Handle List always highlights the active window in its list.

You can set the Handle List to view all visible captioned windows, or only those belonging to Visual Basic. Certain windows are always excluded from the Handle List, including whatever program is specified as the Task Manager and the hidden parent windows of **VBOffice** and Visual Basic.

SpyMode

You can also set the Handle List to operate in SpyMode by clicking the SpyMode main menu item. SpyMode allows you to look at details about each window as the mouse pointer passes over it. Information about the parent window is also displayed. This information includes the window's handle, caption and class name. The coordinates of the mouse pointer as it moves about the screen are also displayed.

When you click on a window the information about that window is retained in the Handle List and SpyMode ends. You can click on the Handle List menu option to return the form to Handle List mode.

See Also:

Technical Notes: The Handle List

Configuration

Configuring **VBOffice** is done in two places; the Configuration Window and the Options menu of each popup window. Configuration options in the Configuration Window are global in scope, while the changes you make in each Options menu only affect that particular window.

The Configuration Window allows you to position the **VBOffice** button box on your screen in one of the six locations shown. You can also specify whether **VBOffice** will be in ToolBox style or ToolBar style.

The Configuration Window also offers you another place to specify the number of ClipBoard Monitor (CLM) files to store in the VBOFFICE.LIB directory. You can disable capturing of CLM files in this window by setting this value to zero.

If **VBOffice**'s hotkeys conflict with hotkey combinations you already have defined on your system, or if the hotkeys interfere with the smooth functioning of your menus (this happens on some systems) you can disable hotkey operation by unchecking the "Use HotKeys" check box. If you wish **VBOffice** to close Visual Basic when you shut it down, check the "Close VB on Exit" checkbox.

If you wish to disable the chimes which play whenever certain forms are displayed, uncheck the "Play Chimes" checkbox. You can also disable the VBOffice opening screen by checking the "No Opening Screen" option. **The "Play Chimes" and "No Opening Screen" options are available to registered users only.**

Changes made to the Configuration Window are saved to VBOFFICE.INI in the **VBOffice** directory.

See Also:

The Popup Window Options Menu

The Options Menu

Each popup window has an Options Menu which allows you to configure the behavior and display of that window uniquely. Settings made in the Options menu apply only to that popup window. Each popup window also remembers it's size and location between sessions.

The Options Menu:

Hide Upon Open:	When this item is checked the popup window automatically hides itself after performing it's specified action. When not checked, the popup window remains open for additional work after it finishes. This appears as "Hide After Typing" in the Code Typist and "Hide After Adding" in the Code Librarian.
Confirm Deletions:	Prompts you for confirmation when you delete an item from a popup window list using List | Delete or the [Del] key. If this item is not checked, list items are deleted instantly. (This item is also found in the Options Menu of the Clipboard Monitor File Manager.) **NOTE:** When you delete items from the Code Librarian, Code Typist or Clipboard Monitor File Manager list, you also delete the corresponding file stored in the VBOFFICE.LIB directory.
AutoMax List:	When this option is checked the list height will automatically adjust to display all items in the list, up to the available screen height. When this item is not checked a scroll bar will appear whenever there are more list items than can be displayed in the available height.
Bold Font:	Allows you to turn font bolding on or off for the current list.
Font Size:	Allows you to set the font size for the list to 8.25, 9.75 or 12 point Arial.
Cascade Menus:	Clicking this menu item 'cascades' all open popup windows to the top left corner of your screen.
Unload All Menus:	Clicking this menu item will unload all popup windows from memory. This can be handy if you're running low on system resources and wish to reclaim some RAM quickly and easily.

Modifying List Entries

When you click List Modify or key Ctrl+M the Modify dialog box appears. This form allows you to modify the List caption, program filename and/or command line parameter of the current list items. You move from field to field using the Tab key.

The Browse button allows you to select a file if a file name property is highlighted. This button pops up a common dialog box to select the file desired. This dialog box defaults to the file currently in the textbox.

Which properties can be modified depends on the type of list you're working with.

For Help Files you can modify the Caption and File Name.

For Document Files you can modify the Caption and File Name

For Program Files you can modify the Caption, File Name and Command Line Parameter.

For Project Files you can modify the Caption and File Name.

For Code Librarian and Code Typist Files you can only modify the caption.

Importing and Exporting TYP or LIB Files

VBOffice makes it possible for you to share code fragments (TYP files) and code library subs and functions (LIB files) with other programmers. **T**hese files are stored with a two line header containing both an identifying string and the caption to use for the file. **VBOffice** can import and export these files to and from the VBOFFICE.LIB directory.

To Export Files: click List | File | Export; this brings up the Export Dialog Box. This dialog allows you to select the code items you wish to export. The multiselect list box allows you to select one or more items in the list for exporting. You then type the full destination drive and path name into the textbox and click the Export button (or key [Enter]). **VBOffice** then copies all files related to the selected items to that directory.

To Import Files: Click List | File | Import...; this brings up the Import dialog box. This dialog allows you to select the source drive and directory, displaying any TYP or LIB files which may be present. (The file list box is for display only and performs no function.) Use the File Import dialog box to locate the source drive/directory you desire; If source files are present, you can click OK to import them. **VBOffice** will then copy and rename every LIB or TYP file in that directory to the VBOFFICE.LIB directory.

If you have both LIB and TYP files to import or export you must perform the process twice, once from the Code Typist and once from the Code Librarian.

NOTE: It is strongly recommended that you let VBOffice perform and manage all file import functions. When bulk importing files, **VBOFFICE** renames each file as it's copied in, making sure you don't overwrite files you already have. If you copy the files yourself you could lose code if you overwrite a current file with a new file of the same name.

Technical Notes

Copyright Notice

I wrote **VBOffice** in Microsoft Visual Basic 2.0, then I recompiled it under version 3.0 when that program became available. Consequently, the concepts I used in designing the program will be familiar to many Visual Basic programmers. Many Windows API calls were used to ensure that the program could find the current code window, evaluate the status of Visual Basic and so on. An explanation of some of these concepts may help in understanding exactly how the program behaves.

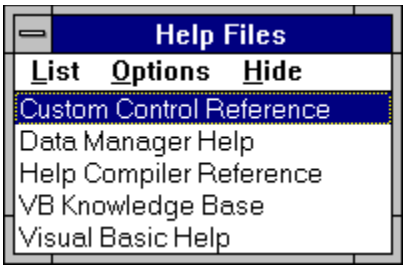
The Renamed VBX Files

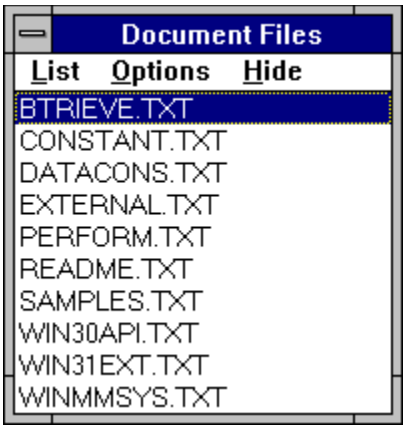
The ButtonBar

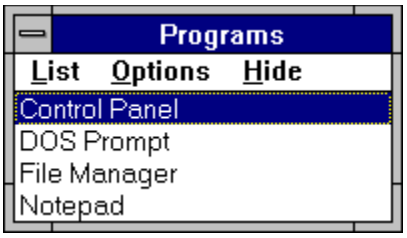
Popup Windows

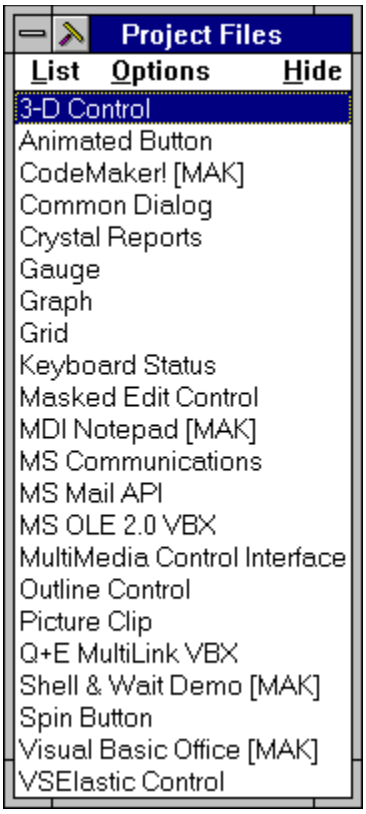
The ClipBoard Monitor File Manager

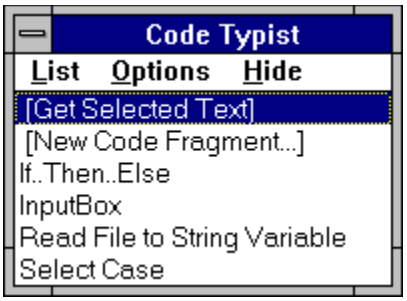
The Handle List

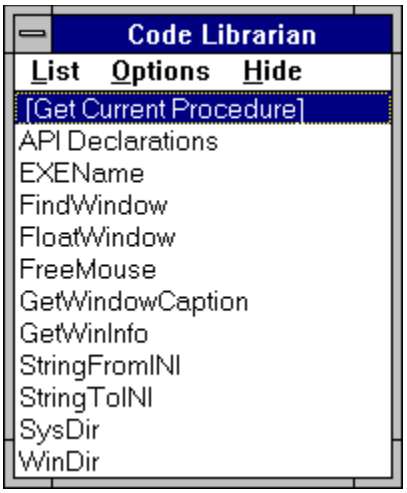

Help Files
List Options Hide
Custom Control Reference
Data Manager Help
Help Compiler Reference
VB Knowledge Base
Visual Basic Help

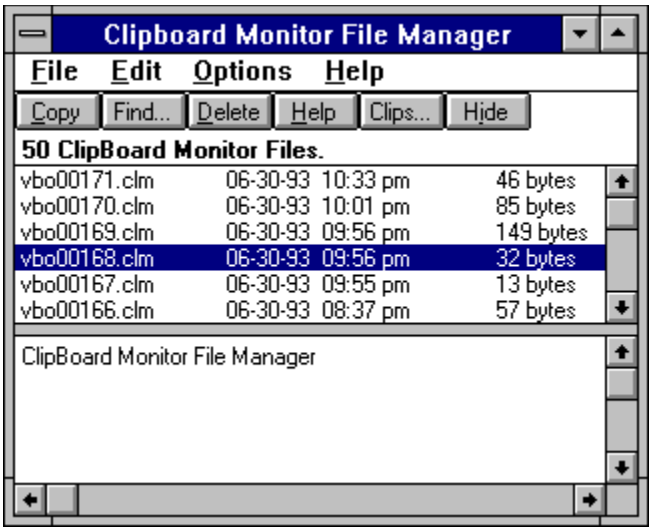

Document Files
List Options Hide
BTRIEVE.TXT
CONSTANT.TXT
DATACONS.TXT
EXTERNAL.TXT
PERFORM.TXT
README.TXT
SAMPLES.TXT
WIN30API.TXT
WIN31EXT.TXT
WINMMSYS.TXT

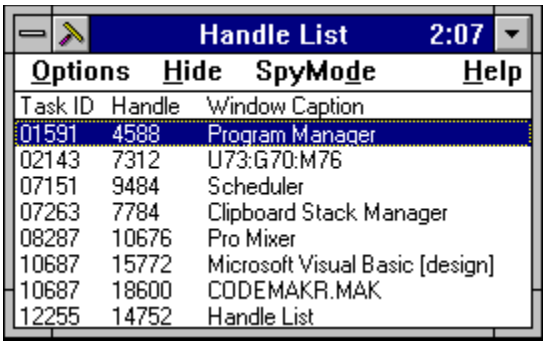

Programs
List Options Hide
Control Panel
DOS Prompt
File Manager
Notepad

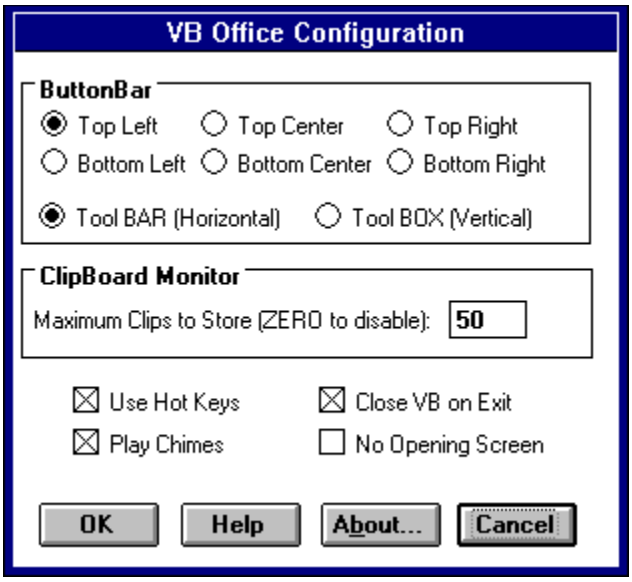

Project Files
List
Options
Hide
3-D Control
Animated Button
CodeMaker! [MAK]
Common Dialog
Crystal Reports
Gauge
Graph
Grid
Keyboard Status
Masked Edit Control
MDI Notepad [MAK]
MS Communications
MS Mail API
MS OLE 2.0 VBX
MultiMedia Control Interface
Outline Control
Picture Clip
Q+E MultiLink VBX
Shell & Wait Demo [MAK]
Spin Button
Visual Basic Office [MAK]
VSElastic Control


Code Typist
List Options Hide
[Get Selected Text]
[New Code Fragment...]
If..Then..Else
InputBox
Read File to String Variable
Select Case


Code Librarian
List Options Hide
[Get Current Procedure]
API Declarations
EXEName
FindWindow
FloatWindow
FreeMouse
GetWindowCaption
GetWinInfo
StringFromINI
StringToINI
SysDir
WinDir


Clipboard Monitor File Manager
File Edit Options Help
Copy Find... Delete Help Clips... Hide
50 ClipBoard Monitor Files.
vbo00171.clm 06-30-93 10:33 pm 46 bytes
vbo00170.clm 06-30-93 10:01 pm 85 bytes
vbo00169.clm 06-30-93 09:56 pm 149 bytes
vbo00168.clm 06-30-93 09:56 pm 32 bytes
vbo00167.clm 06-30-93 09:55 pm 13 bytes
vbo00166.clm 06-30-93 08:37 pm 57 bytes
ClipBoard Monitor File Manager


Handle List
2:07
Options
Hide
SpyMode
Help
Task ID Handle Window Caption
01591 4588 Program Manager
02143 7312 U73:G70:M76
07151 9484 Scheduler
07263 7784 Clipboard Stack Manager
08287 10676 Pro Mixer
10687 15772 Microsoft Visual Basic [design]
10687 18600 CODEMAKR.MAK
12255 14752 Handle List

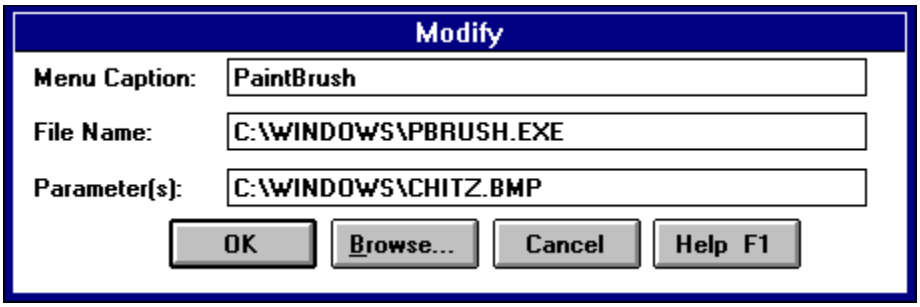

VB Office Configuration
ButtonBar
Top Left
Top Center
Top Right
Bottom Left
Bottom Center
Bottom Right
Tool BAR (Horizontal)
Tool BOX (Vertical)
ClipBoard Monitor
Maximum Clips to Store (ZERO to disable):
50
Use Hot Keys
Close VB on Exit
Play Chimes
No Opening Screen
OK
Help
About...
Cancel

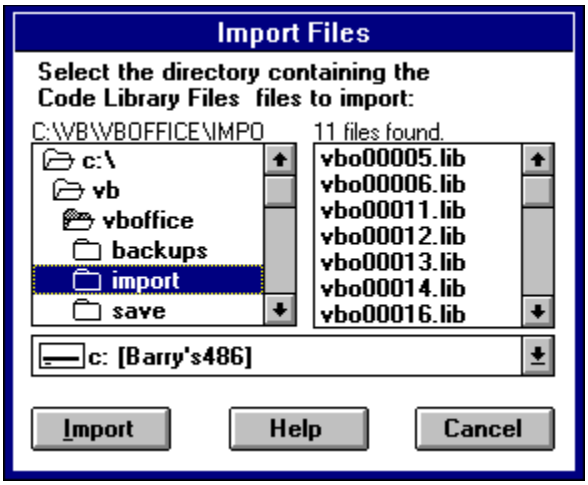

Options
✓ Hide Upon Run
✓ Confirm Deletions
✓ AutoMax List
Bold Font
Font Size
Cascade Menus
Unload All Menus


EXE
EXIT

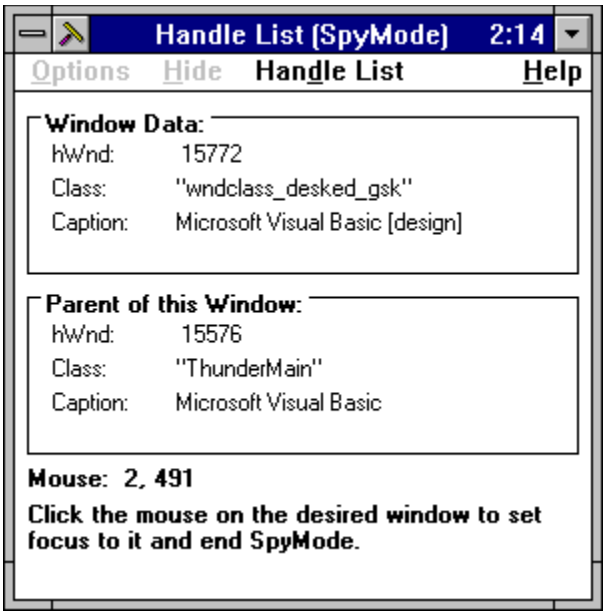

EXE
EXIT

Shift + Alt + ... F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F11 F12


Modify
Menu Caption:
PaintBrush
File Name:
C:\WINDOWS\PBRUSH.EXE
Parameter(s):
C:\WINDOWS\CHITZ.BMP
OK
Browse...
Cancel
Help F1


Import Files
Select the directory containing the
Code Library Files files to import:
C:\VB\VBOFFICE\IMPO
11 files found.
c:\
vb
vboffice
backups
import
save
vbo00005.lib
vbo00006.lib
vbo00011.lib
vbo00012.lib
vbo00013.lib
vbo00014.lib
vbo00016.lib
c: [Barry's486]
Import
Help
Cancel


File Export
5:09
Select the Code Typist items to export
from the multiselect list:
Do...Loop
For...Next
If..Then..Else
InputBox
On Error Resume Next
Open File
Select Case
Enter the Destination Drive/Directory:
A:\
Export
Help
Cancel


Handle List (SpyMode) 2:14
Options Hide Handle List Help
Window Data:
hWnd: 15772
Class: "wndclass_desked_gsk"
Caption: Microsoft Visual Basic [design]
Parent of this Window:
hWnd: 15576
Class: "ThunderMain"
Caption: Microsoft Visual Basic
Mouse: 2, 491
Click the mouse on the desired window to set focus to it and end SpyMode.

Visual Basic Office

Marquette Computer Consultants
22 Sirard Lane
San Rafael, CA 94901-1066
(415) 459-0835
CompuServe 70413,3405

THANKS TO KATHERINE AND WALTER
FOR THEIR PATIENCE AND SUPPORT
DURING THE DEVELOPMENT OF **VBOFFICE!**

Packing List

The following files comprise this release of **VBOffice**:

VBOFFICE.EXE	Program File
VBOFFICE.HLP	Help File
README.TXT	Last minute notes.
FILE_ID.DIZ	A text file which describes VBOffice
VBOFC001.VBX*	Renamed Visual Basic 3.0 Custom Control CMDIALOG.VBX
VBOFC02.VBX*	Renamed Visual Basic 3.0 Custom Control PICCLIP.VBX
HOTKEY.DLL	DLL for HotKey functionality
REGISTER.WRI	Windows Write file containing the mail-in registration form
VBRUN300.DLL	Visual Basic 3.0 Run-time DLL (Required, but not included)

*** Note: The two VBX files provided with VBOffice are RENAMED from the custom control files provided with Visual Basic version 3.0.** This is not an attempt to steal credit from Microsoft. Rather, this is to ensure compatibility with users who may still using Visual Basic 2.0. For more details on the why and how of this see Renamed VBX Files.

Tech Notes: The Button Bar

The PicClip control was used to create the buttons, thus saving on GDI resources. I used a lot of API calls to determine the presence and operating status of Visual Basic. **GetProfileString** was used to obtain the location of Visual Basic from WIN.INI in the event that **VBOffice** had to start it. I used **GetWindow** to obtain all window handles and **GetWindowCaption** to identify them. **GetParent** obtained the handle of VB's "hidden" parent window; if the hidden parent is iconized, all **VBOffice** forms are hidden. If the hidden parent ceases to exist, **VBOffice** shuts down too.

GetActiveWindow, GetClassName and **IsIconic** were used by a timer in the Buttonbar to evaluate each window as it became active and to determine if it was a valid code window. If it was, any subsequent code pasting or copying operations would be performed on that window. **IsWindow** was used to ensure the existence of the "current" code window prior to pasting to it.

For purposes of testing, **VBOffice** was configured to consider Windows Notepad a code window as well. This functionality was left intact, so you can still copy and paste to and from NotePad if you wish.

I used the API call **GetWindowRect** with the **GetWindowPlacement** data structure to determine the exact position and size of the visible VB window. This information allowed me to correctly position the buttonbar under the VB window. **SetWindowPos** was also used to bring the buttonbar to the top each time VB's visible main window received the focus.

See Also:

Technical Notes

Tech Notes: Popup Windows

The various popup windows (except for the Handle List, ClipMon Manager and Configuration Window) are actually multiple instances of the same form with different internal names. Since much of the functionality of these windows is duplicated it was decided to consolidate most of them into one form, making for a smaller program.

I used the API call **SetFocus** as **SetFocusAPI** to give the Code Typist and Code Librarian the ability to activate the current code window prior to operating on it. If no code windows are available or the project is in run mode, **VBOffice** pops up a message to the user if they try to paste text.

See Also:

Technical Notes

Tech Notes: The ClipBoard Monitor File Manager

This window presented a number of interesting challenges. Keeping a sorted list of files with date and time information required using an array of records of a user-defined type, one for each file found in the VBOFFICE.LIB directory. I obtained data for each file record using the **Dir$( )**, **GetAttr( )**, **FileLen( )** and **FileDateTime( )** functions. I then wrote code to sort the array in date order using a bubble sort. Any files over the maximum allowable amount were deleted (from oldest to newest) and the array redimensioned with a **Redim Preserve**.

As each new CLM file appears, the records in the array are 'bumped up,' making room for the newest. If the array is full, the last file is deleted and it's record is deliberately 'lost.' This data is retained regardless of whether the ClipBoard Monitor File Manager is loaded or not.

The resizing bar separating the file list and the text box is actually a picture control which is sized to the scalewidth of the form plus two pixels. When you click on it to drag it up or down, the API call **ClipCursor** ensures that you can only move the mouse vertically. When you drop the bar, ClipCursor frees the mouse and the Textbox and List controls are resized accordingly.

At present the ClipBoard Monitor File Manager monitors text only and uses a VB Timer. In the future I hope to add bitmap (BMP) file saving and a better method of monitoring Clipboard activity.

See Also:

Technical Notes

Tech Notes: The Handle List

This form was adapted from another stand-alone applet I developed to help me test API calls I was making in a custom application. The Handle List uses the API calls **GetWindow** and **GetNextWindow** to cycle through all windows currently in existence in the Windows environment. **IsWindowVisible** and **GetWindowCaption** were used to filter out unwanted windows, since only visible windows with captions should appear in the list. Surprisingly the 'hidden' parent window of Visual Basic evaluates as visible, so I had to exclude it from the list via customized code. I also excluded the caption for the hidden "Visual Basic Office" parent window, as well as the caption for any active Task Manager program.

Originally the Handle List used **GetNumTasks** to determine whether or not to update it's window list, but this no longer worked under Visual Basic. VB code windows can be opened and closed without changing the number of tasks running in Windows, so I had to find a better approach. The solution was to use the Windows 3.1 API call **GetFreeSystemResources**, passing it the GFSR_USERRESOURCES parameter. Windows' user resources indicate how much memory is available to create handles and windows. When the result of that function changes it means windows have been opened or closed.

For **SpyMode** I used the WndPeek program provided in "Visual Basic How-To" from Waite Group Press. (Thanks to Robert Arnson, Daniel Rosen, Mitchell Waite and Jonathan Zuck for an excellent book which has as much use under VB 3.0 as it did under version 1.0!) I performed minor modifications in order to display additional information about each window highlighted.

I used **SetCapture** to redirect all information regarding mouse movements to the Handle List form, while **ReleaseCapture** was called when exiting SpyMode. **GetMessagePos** and **WindowFromPoint** were used to determine which window the mouse was over. After that it was a simple matter to obtain the window's caption, class and other information.

The highlighting of each window was peformed using a number of API calls to draw a box around the current window. The window's coordinates were determined using **GetWindowRect**. The pen to draw the box box was set to use the inverse of the colors currently displayed, ensuring that the box would be visible regardless of the background color. This also enabled me to 'erase' the box by issuing the same call to the same window, drawing the inverse of the inverse, thus restoring the original color. **SetObject** was used to tell Windows which object to draw upon, while the API call **Rectangle** actually drew the box.

See Also:

Technical Notes

Tech Notes: The Renamed VBX Files

VBOffice was intended to support Visual Basic versions 2.0 and 3.0, but this presented a dilemma. If a user was using Version 2.0, the Version 3.0 VBX files provided with **VBOffice** might cause compatibility problems. Microsoft Technical Support would only go so far as to guess 80% compatibility between VB 2.0 and the custom controls from VB 3.0 I used to develop this program. If I went back to VB 2.0 incompatibility would be certain for users of Version 3.0. (I tried it; believe me, it doesn't work!)

I ended up renaming the VBX files to avoid a conflict with the installed Custom Controls of either version of VB. This involved making a copy of each file and editing it in binary mode. I had to replace every instance of CMDIALOG with VBOFC001 in the CMDIALOG.VBX file, and each PICCLIP became VBOFC02 in the PICCLIP.VBX file. (Kids, don't try this at home.)

The renamed VBX files worked just fine in my testing and there seems to be no collision between my installed copy of CMDIALOG.VBX and PICCLIP.VBX. The only drawback is the extra disk space taken up by these files; I hope you agree it's a small price to pay; it's certainly a better alternative than having **VBOffice Setup** copying over installed VBX files and creating incompatibilities. In the future, when the installed base of version 3.0 is larger I'll probably return to using CMDIALOG.VBX and PICCLIP.VBX, at which point you'll be able to remove VBO*.VBX from your WINDOWS\SYSTEM directory.

This is in no way intended to 'steal' technology from Microsoft. These files are already free for distribution with applications developed in Visual Basic, and I make no secret of the original authorship of these files, as you can see from the Copyright Notice of this help file and the **VBOffice** About box.

See Also:

Technical Notes

VBOffice lets you quickly access any help file, document file or program you wish without switching away from the VB environment. Individual popup windows containing the files to access can be easily configured, allowing you to access files, documents and programs in any directory, on any drive.

The **VBOffice** Code Librarian stores and manages all your favorite subroutines and functions in a single directory, where they can be easily accessed for addition to your project. Now you can maintain a single library of your favorite code modules for easy addition to any project.

The **VBOffice** Code Typist stores often-used code fragments for quick insertion into your code. You can create your own or import other code fragments, depending on how you work. With **VBOffice** you always have a speedy typing assistant only a mouse-click away.

The Project Files popup window allows you to to quickly add custom controls, forms and modules to your project. Specify their location once and these files will remain quickly accessible to your project no matter where they are on your disk. Double-click on one and it's added to your current project.

You can also open any MAK file from this window! Specify the location once and it'll be instantly accessible for opening at any time. Double click on a MAK file name and Visual Basic opens that project instantly.

VBOffice can save a copy of each fragment of code you copy to the clipboard for later retrieval. You can specify as many or as few ClipMon files as you like if you wish to conserve disk space. Using the **ClipBoard Monitor File Manager** you can view and delete these files. You can even perform text searches through the saved files if you want to find a piece of code you deleted. If it was copied to the ClipBoard, the ClipBoard Monitor can help you find it again! The ClipBoard Monitor provides an undo capability limited only by the free space on your hard disk.

The **VBOffice** Handle List displays a list of all visible windows with captions, and includes their Task ID and Handle in the list. A single click on a list entry activates that window. Conversely, the Handle List always highlights the list item which is currently active.

The Handle List also features **SpyMode**, where you can discover information about any visible window by passing the mouse cursor over it. When in SpyMode each window is highlighted while information about the window's handle, class name, caption and parent are shown in the Handle List form.

These Handle List functions are handy for evaluating API calls (where Window handles are often used), for quickly finding a window if it's buried under others and to determine exact information about a window using SpyMode.

VBOffice's Save/Run button automatically saves your entire project before you run it, protecting you from code loss in the event of a General Protection Fault (GPF). The button can also be used to break program execution, and acts as a quick visual reminder of your current run mode.

The **VBOffice** toolbar matches the twelve function keys on your keyboard. Holding down [Shift]+[Alt] and pressing the appropriate function key is the same as pushing the button. This provides quick access to all **VBOffice** tools, even if you're editing in a full-screen code window.

VBOffice can be configured to match the way you work. The ClipBoard Monitor can be set to store as many or as few files as you wish, and the ToolBar can be positioned in horizontal or vertical mode in any one of six different screen positions. The various **VBOffice** popup windows can be configured to remain open or disappear after they do their work and can be configured to display a font style suitable for your screen resolution. All **VBOffice** windows remember their location, size and configuration from session to session, and can be either hidden for greater speed or unloaded (closed) to conserve system resources.

VBOffice allows you to share Code Typist and Code Librarian files with other programmers; just use the File Export function to copy your files to another drive or directory, or use the File Import function to bring files into your library.

Registering VBOffice

Registering by Mail **Registering On Line**

VBOffice is shareware; that means you have a no-risk opportunity to test the software and see if you like it. It does **NOT** mean it's free. If you like it and continue to use it, you are expected to pay for it.

What You Get for Registering...

Code Module Files: Users who register via CompuServe will receive via CISMail a ZIP file containing a collection of self-contained Code Typist and Code Librarian files, many which use API calls. Just import them into your system and you'll have some great code routines which work 'right out of the box.' Users who register by mail can recieve these files if they enclose a floppy disk and mailer with their registration form. We'll copy the files to the diskette and mail it back to you.

Unlimited Technical Support: Since the quality of technical support is an integral part of the product you are evaluating, **all** users of **VBOffice** will receive support from Marquette Computer Consultants, either via CompuServe Mail or by telephone. However, **unregistered users will be limited to only three support incidents**. Registered users will receive unlimited technical support.

Upgrade Notices and Discounts: As new versions of **VBOffice** are released you'll be informed, either by CISMail, Postal Mail or fax. Registered users will also be eligible to upgrade their software at a cost savings; unregistered users will have to pay full price to register newer versions of **VBOffice**.

Long-Term Assurance. If shareware is paid for, the author has incentive to improve the product and continue to support it. By paying for **VBOffice** you help to assure continued support and upgrades of this product.

A Clear Conscience. I've put in eight months on this project, and it's only the beginning. Don't let me down!

About the Unregistered Version...

An unregistered copy of **VBOffice** is not crippled. It performs all the program functions of a registered copy, with only a few minor exceptions. The opening screen of an unregistered program stays open for five seconds before the OK button is enabled.

If your program is unregistered you cannot reconfigure it to eliminate the opening screen, nor can you disable the playing of CHIMES.WAV when certain forms are opened. Each form and many menus in VBOffice will also contain items which both remind you to register and give you quick access to the registration dialog box.

Once you purchase **VBOffice**, Marquette Computer Consultants will send you a registration code which you enter into the program. Once this code is entered all registration reminders (menus, buttons, etc.) will be removed and the above mentioned two configuration options will be enabled.

There are two ways to register: Click above on the method you're interested in to jump to that topic.

Registering by Mail

To register by mail, open the REGISTER.WRI file provided with **VBOffice** and fill it in as completely as you can. Print out the form and attach a check payable to **Marquette Computer Consultants**. Enclose payment of $40.00 US for each copy ordered. Mail the form to...

Marquette Computer Consultants
22 Sirard Lane
San Rafael, CA 94901-1066

Once Marquette Computer Consultants receives your order form and check you will be mailed or faxed the registration code required to remove all registration reminders. In addition you'll be informed of the latest upgrade versions, if any, along with other related information.

If you wish to receive the sampler of Code Librarian and Code Typist files, please include a floppy diskette and mailer with your registration form. We'll copy the files to the diskette and mail it back to you.

See Also: **Registering On Line**

Registering On Line

If you have a CompuServe account you can register **VBOffice** on line using the Shareware Registration database in the SWREG forum. The Shareware Registration database contains descriptions of hundreds of shareware programs designed for a variety of platforms and purposes.

GO SWREG to access this forum and select "Register Shareware." A number of search criteria will be presented; you want to search by Registration ID.

NOTE: The REGISTRATION ID of VBOffice is 1262.

You can use this number to find VBOffice. Once the file is found, you are shown the program's description, with a prompt: "Would You Like to Register? (Y/N)."

Enter "Y" at the prompt to register; enter "N" at the prompt to return to the previous menu. When you enter "Y" to register, you will be prompted to enter your full name, company name (optional), your complete address, your phone number (optional) and the total number of copies of the program that you wish to register. When you finish entering this information, you can VIEW, CHANGE, SEND or CANCEL it. When you SEND your registration, your CompuServe account will be billed for the amount required to pay for the software.

Once Marquette Computer Consultants receives notification of your registration you will receive via CISMail the registration code required to remove all registration reminders. You'll also receive a ZIP file containing a sampler of Code Librarian and Code Typist files which you can import into your Code Typist and Code Librarian lists. In addition you'll be informed of the latest upgrade versions, if any, along with other related information.

See Also: **Registering by Mail**