Object Library Version 1.1

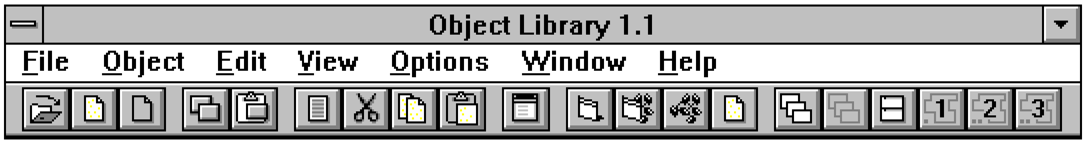
Contents

Overview

The Object Library consists of a **Main** Window, a **Contents** Window, and 4 **Code** Windows whose behavior is patterned after that of the VB design environment.

Several other windows provide utility services.

Main Window



The Main Window consists of a menu and a toolbar whose function and behavior is similar to the Main window of Visual Basic.
The toolbar provides quick access to editing and window placement functions. Click the buttons for a description of their function.
The toolbar may be hidden by selecting **Option | Toolbar**.
The Main menu may be hidden by double-clicking the space between the buttons.
The following is a brief description of the commands available from the Main menu:

File | Open Library... - opens a library database
File | Open File - opens a VB source file
File | Save File - saves a VB source file
File | Print to File - dumps the current record to a text file
File | Project Browser - utility for viewing symbols and dependencies in projects

Object | New Group - creates a new Table in the Library database
Object | Rename Group - renames a Table in the database
Object | Delete Group - deletes a Table and all its Object records
Object | New - creates a new Object record in the current Group
Object | Delete - deletes the current Object record
Object | Save performs an immediate update on the current record
Object | Import Wizard - utility for pasting code to the Library
Object | Export Wizard - utility for copying code from the Library

Edit | Copy Controls - copies binary control data from the library to clipboard
Edit | Paste Controls - pastes binary control data from clipboard to the library
Edit | Revert - reloads the current record from the database
Edit | Select All - selects all visible text in the active code window
Edit | Cut - cuts selected text from a code window
Edit | Copy - copies selected text to the clipboard
Edit | Paste - pastes text from the clipboard to the active code window
Edit | Find - invokes the Library search dialog

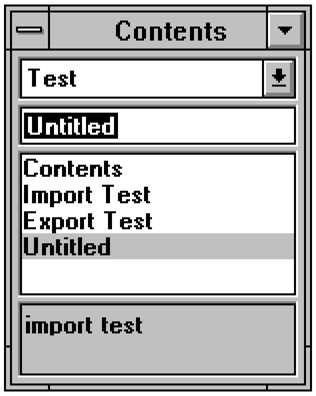
View | Contents - shows the Contents window
View | Form - shows the Form window
View | Form Code - shows the Form Code window
View | Module - shows the Module window
View | Text - shows the Text window
View | New Procedure - creates a new procedure in the active window
View | Next Procedure - shows next procedure in the active window
View | Previous Procedure - shows previous procedure in the active window

Options | Toolbar - toggles visibility of the toolbar

Options | Menu - toggles visibility of the menu
Options | Font - sets the fonts in the various windows

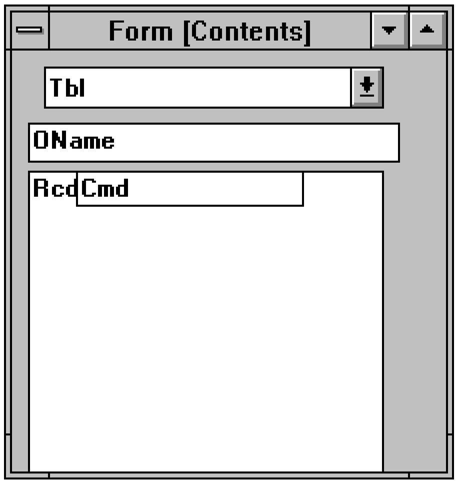
Window | Cascade - cascades the 4 Library code windows
Window | Tile - tiles the 4 Library code windows
Window | Close All - closes the 4 Library code windows and the Contents window
Window | Show All - shows the 4 Library code windows and the Contents window
Window | Screen Manager - utility to arrange VB design environment windows
Window | <named desktop> - placeholder for user-defined desktops created with Screen Manager

Contents Window



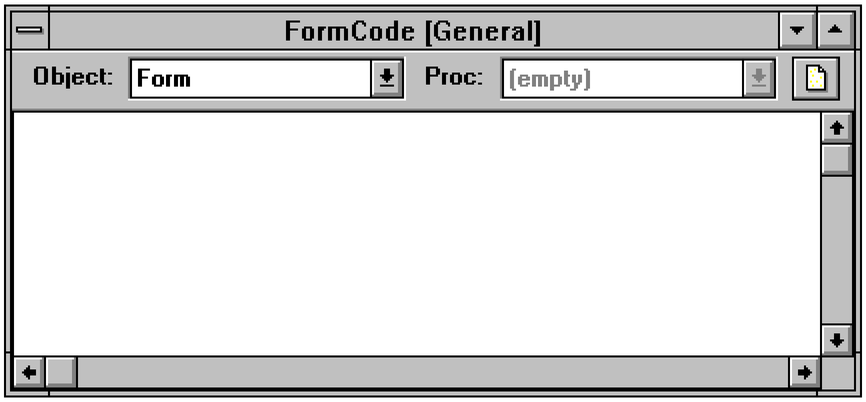
The Contents Window provides the means to select the current Object record. The window consists of a combobox, a textbox, a listbox and a text field for notes.
The **Group** combobox lists all the Group Tables in the code library.
The **Name** textbox below it allows you to edit the name of the current Object record.
The **Object** listbox shows each Object record in the current Group.
The **Notes** textbox holds a short description of the selected Object.
The illustration shows Group **Test** with the Object **Untitled** selected and the description **import test**.

Form Window



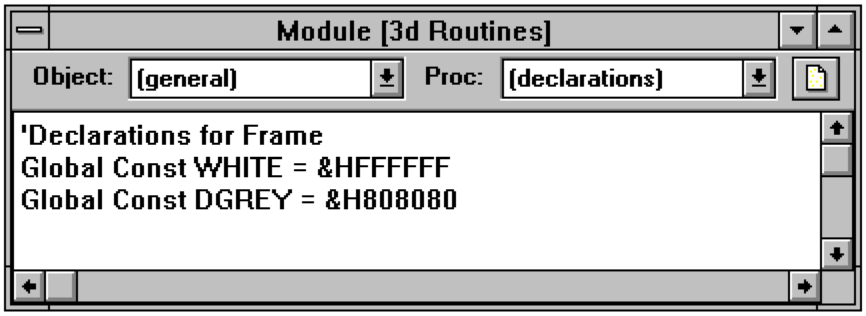
The 4 Code Windows display the contents of the current Object record. The 2 types of Code Windows correspond to Visual Basic Forms and Modules: The **Form** Window, shown above, displays controls; the 3 other **Code** Windows display text.
The Form Window displays a stylized graphic representation of the current Object's control data. Controls are represented without regard to their Z-order; they are drawn in an XRay style to allow you to view as many of them as possible.   If the control has a caption, the caption is printed over it, otherwise the name is used.
Controls are positioned according to their positions when pasted into the library.
The illustration shows the controls used in the design of the Library **Content** window.
Note: Certain controls may be misrepresented or omitted in the Form window. 3rd party VBX's will be represented by a blank rectangle. Line controls may not be visible. Do not be alarmed if what you see does not exactly match what the record should contain. If you need to see a more exact rendering of the control data, add a new Form to a VB project and paste the controls to the Form.   If you need to edit control data, paste the controls to a Form, edit them, then paste them back into the library.   Control data cannot be edited from within the library.

FormCode Window



The 3 Code windows display the source code and other text in the current Object record.
The **FormCode** Window, shown above, shows the Control procedures and Form procedures in the current Object.
If control data exists in the current record, the FormCode window will list the names of the controls in the **Object** combobox. The FormCode window will always add an entry for **Form** to the Object combobox. Event procedures for controls are listed in the **Procedures** combobox. Event procedures are not listed unless a procedure exists. If a control has no procedures, the Procedures combobox will be disabled. This differs from the behavior of VB code window. To add a procedure, paste it from the clipboard or select **View | New Procedure**.
If event procedures for controls are pasted to the FormCode window and no controls with the same name exist in the current record, then the event procedures will be displayed in the Procedures combobox under the (general) section. Otherwise, they will be displayed when the control is selected from the Object combobox. This is the same behavior as a VB Form code window.
The illustration shows the FormCode Window of the Object record **General**. The **Form** control is selected, but no procedure code has been added, so the procedure combobox is disabled and bears the caption **(empty).**

Module Window



```
'Declarations for Frame
Global Const WHITE = &HFFFFFF
Global Const DGREY = &H808080
```

The **Module** Window shows the general procedures in the current Object.
The Module window does not recognize control procedures. All procedures are treated alike as general procedures and listed in the Procedures combobox. This is the same behavior as a VB code window.
The illustration shows the Module window of the Object record **3d Routines**. The (general) (declarations) section has had some constants added.

Text Window

The Text window is identical to the Module window.

The **Text** Window shows any code, comments, or other textual material which has been added to the current Object, but which you wish to keep separate. You can use this field for documentation purposes if the Notes field in the Content window is too small.

This field is optional.   When you create a new Group table, this field is not created by default and you will probably not need it for most Group tables.

Import Wizard

The Import Wizard automates the process of selecting declarations and procedures from VB source files and pasting them into the Library.

The wizard allows you to select declarations and procedures from a list and assign the Library window (field) to which each will be pasted. You may select from individual source files or from a list of all the files in a project. You may preview the material to be imported and then accomplish the copy/paste operation in a single click, thus reducing the liklihood of errors.

Selecting an Object to import to

When the wizard is launched, it shows the Object record currently selected in the library.   The comboboxes display a list of all the controls and procedures in that Object.

If the current Object is not the Object you wish to import to, select the desired Object in the **Contents** window or create a new one.

Click **Refresh** and the wizard will be updated to show the current Object record.

Selecting the Source files to import from

If VB is running and the loaded project can be found in the current directory, then the project name will be displayed in the textbox.

If no name is displayed and you wish to select from a project, click **Browse** and select the project makefile.

If VB is not running, or if you wish to import code from a different project, click the **Browse** button and select the makefile.

It is not necessary to select a project to import from; you may select files individually in the next step. Importing from a project gives you a list of that projects source files to select from.

Selecting Procedures and Declarations to import

This discussion describes selecting a procedure. Follow the same steps to select declarations.

1. Select a file from the combobox or click **Files...** and select a file.
2. A list of the procedures which the file contains is shown in the **Procedures** listbox.
3. Select either the **Form** or **Module** list by clicking it. The active list is shown with a white background.
4. Double-click a procedure name in the **Procedures** list to add it to either the Form or Module list, whichever is active.
5. Double-click the procedure name in the Form or Module list to remove any procedures you decide you do not wish to import.
6. If you want to view a procedure, select it in any of the lists and then right click. A textbox will pop up with the procedure displayed. Click the textbox to return to selecting procedures.

Selecting the Field to import to

The procedures and declarations you have selected are shown in the **Form** and **Module** lists, reflecting which Library window (field) the code will be appended to.

To make the task of reusing the code later a little easier, pay attention to which list you add the procedures and declarations to.

When a **.FRM** file is selected, the wizard will make the Form list active, otherwise the Module list. Check to be sure the correct list is active.

Try to add control procedures to the Form list, and global declarations and constants to the Module list. That way, if you use the Export Wizard later to transfer the code to a project, you will be able to do so with just a couple of button clicks.

Importing the selected code

When you have finished selecting items to import, move to the final step.

Click the **Field** combobox to preview the selections you have assigned to the Form or Module fields.

If the wizard detects that any procedures you have selected conflict with the names of procedures already in the Object record, it will notify you and give you a chance to abort the import.

Name conflicts within the Library are not critical, but you will want to rename the procedures after they are imported to prevent possible confusions.

If you need to edit the code you have selected, you can edit the code after it has been imported. Any changes made to the text at this point will have no effect.

When you are satisfied with the selections, click **Import**.

The selections for the Form field will be added to the FormCode window, the selections for the Module field will be added to the Module window.

Export Wizard

The Export Wizard automates the selection, renaming, and copying of the procedures in an Object record for use in a VB project.

The wizard allows you to select from an Object record the code you wish to export, and separate it for copying to several targets.

The wizard will scan the targets for name conflicts at the procedural level and advise you of potential problems.

If conflicts exist between control or procedure names, the wizard will allow you to provide aliases for the procedures and controls before copying them to a VB project. It will also give you a chance to edit procedure text before copying it without affecting the code in the Library.

Selecting Procedures to export
Renaming Procedures and Controls
Checking for conflicts
Previewing and Editing the code

Selecting Procedures to export

When the wizard opens, it provides you a list of all the procedures and controls in the current Object. If this is not the Object containing the code you wish to export, select the Object from the **Contents** window, then click **Refresh**.

The **Field** combobox allows you to view the Controls, Form procedures, and Module procedures available for export.

Controls must be pasted all together or not at all; however, you can deselect procedures by unchecking them in the list.

Renaming Controls and Procedures

If the names of the controls or procedures in the Library conflict with those in your project, you can rename them before pasting.

To Rename a Control:

1. Select a Control from the list. Click **Rename**.

2. Fill in a new name for the Control. The new name must consist of the same number of characters as the original. The textbox will not allow extra characters. If the new name has too few characters, it will pad the name.

To Rename a Procedure:

1. Select one or more procedures from the list. Click **Rename**.

2. If you selected more than one procedure for renaming, you now have 2 options. (For Control procedures, the suffix option will be disabled).   Enter a suffix or prefix and select the option button, to apply suffixes or prefixes to several procedures at once.

3. If you only selected one procedure from the list, the **Rename** option will also be available. Enter a new name. (It need not be the same length as the original.)

Click **Apply** to create the alias.

Click **Reset** to remove aliases from selected items.

Checking for Name Conflicts

If the wizard can find the current VB project, its files will be listed in the Form and Module listboxes.

If the current project is not found, click **Project** and select the project makfile.

To check for name conflicts, select a Target file in either or both of the comboboxes.

The selected procedures in the Form list (step 1) will be compared to the procedures in the file you select in the Form combobox; likewise the Module list to the Module combobox.

If name conflicts are found, they will be shown in the Results list below. Go back and provide an alias for the Control or Procedure needing one. Then try again until no conflicts are found.

Note: Variable name conflicts and scoping issues are not analyzed at this time. The wizard only searches for conflicts which will cause pasting errors.

Previewing and editing

Click the **Field** combobox to view the code you have selected for export. Controls cannot be previewed, but a message in the textbox will indicate if any controls are selected for export.

If you applied aliases, the text should reflect those changes. Check to be sure the code is as you wanted.

If you wish to further edit the code before copying it, do so now.

To Copy Code to a VB Project:

Copying the code involves 3 steps, one copy for each field.

1. First select **Controls**, then click **Copy Field**. Open the VB Form you are copying to and paste the controls.

2. Next, select either **Form Procedures** or **Module Procedures**. Click **Copy Field** to copy all the text in the field. (You may also copy selections. Select the code and click **Copy Selection**.)

Open a VB code window and paste the code.

3. Repeat step 2 for the other field.

Project Browser

The **Project Browser** gives you an quick overview of a VB project. It will display all the Controls, Variables, Types, Dlls, VBXs, and Procedures in a VB project by category. It will display the same information on a by-file basis.

You can quickly locate and view Variables and Procedures by double-clicking them in the Project Browser lists. You can also scan the project for occurences of or references to Variables and Procedures and save the information to a file.

Selecting a Project
Viewing by Element
Viewing by File
Searching a Project

Selecting a Project

The Project Browser reads disk files. If the project is open in the VB design environment, select **File | Save Project ...** from the VB menu to be sure the results are accurate.

Select **File | Open Project**.

The Project Browser must analyze the project files. This can take 10 - 20 seconds for a project the size of this one.

Viewing by Element

In this mode, select from the **Element** menu to see a listing of all the symbols in the project by category.

Viewing by File

In this mode, select from the combobox to see a listing of all the symbols in a particular file.

If the file is a Form file, Controls will be shown in the large list. If it is a Module file, public procedures will be shown.

Searching a Project

You can search a project for occurences of a variable or procedure reference.

The results of the search are displayed in a list which allows you to click and view the item in context.

To Search a Project:

1. Select **View | Search...** and enter a variable or procedure name.
2. The files to be searched are listed to the right of the textbox. Right-click the list to select or deselect all items. Click **Find...** to initiate the search.
3. When the search is complete, the results are displayed in the list. Click an entry to view it.
4. Click Save to write the search results to a file in the projects makefile directory.

Screen Manager

The VB desktop can get really cluttered, and Object Library adds several more windows. Help!

The **Screen Manager** maps the windows of the VB design environment to a desired arrangement of locations and sizes to which you may later move them with a single button click. You may create up to 10 desktop arrangements with the Screen Manager. Three of the desktop arrangements can be accessed from buttons on the toolbar, the remainder from the Main menu.

To create a desktop you simply drag and size the windows to the way you want them to appear when you invoke the desktop. Then give the arrangement a name and save it.

Window placement
Creating and saving desktops
Invoking a desktop arrangement

Window placement

When the Screen Manager opens, several windows are placed at default locations on the screen. These represent VBs windows, all Help files, all Applications, and the Object Library.

Drag and size the windows to the positions at which you want them to appear when you invoke the desktop. You may elect to minimize the window or close it.

The Form, Form Module, and Module windows refer to **VB Form** and **Code** windows. If these windows are not closed or minimized, they will be cascaded from the position you select.

Click **Save** to save this arrangement.

Note: Windows which are not open will not be affected. Certain actions are not available for some windows. Applications will be minimized, not closed. The Main VB window only be moved. The toolbar will not be resized, and so on.

Creating desktops

Select **Window | Screen Manager** from the menu.   When the Screen Manager opens, it fills the entire screen and displays several windows.

Click **New** and give the desktop a name. Drag the windows to create an arrangement to suit your purposes.

Be sure to click **Save** when you are finished.

Invoking desktop arrangements

Three buttons labeled **1**,**2**,**3** appear at the right of the Main window toolbar.

If you have created any desktops, the will be listed in the Main menu below **Window | Screen Manager**. The buttons represent the first 3 items, if any, in the list.

Select from either the menu or the toolbar to invoke the desktop.   VB windows will be moved to the locations you selected.

Using the Library

Some notes on using the Library

Object Library is a code and control library designed to make code reuse easier. Two fundamental requirements of this objective are easy access to the code you have already written (or collected) and the ability to modify your code for new tasks without a major rewrite..

Unfortunately, neither VBXs nor OCXs offer any solutions to code reuse because of the the impracticality of distributing them, their occasional unreliability, and the problems of modifying them when you need slightly different features. You may find that the only code you can reliably use and reuse in VB is VB source code. At least this is the premise the Object Library is based upon: that in many cases you can best create modular components for use in VB by implementing them in native VB code, and that these components will be more reliable, better or comparable in performance, easier to modify, and (hopefully) easier to reuse, than those created by any other method. But to do so requires a little planning.

To create a modular, reusable component, or Object, in VB, you must design your code with a view to modularity and the requirements of extracting the code for storage and reuse. However you approach object design, some common factors to consider are:

naming conventions

the sites at which data is stored

the interactions between events, controls, and genaral procedures

the methods by which data and control properties are manipulated

These factors cause the greatest problems when you attempt to copy working code from one project to a new project.

Other factors complicating the implementation of Objects in VB are:

using file scope to effect encapsulation

methods of data storage for Object instances

message passing techniques

VB hides most of its own implementations of objects, and has more than a few design features and quirks which make object design difficult. All told, the possibilities are not that great, and working around VBs helpful features and bugs are sometimes the greatest impediment to the task.

The following topics will explain how you can use the Object Library to help with some of these problems, and will suggest some possible approaches to designing VB Objects. Several models of object design for VB have been proposed, see the Readme file for a list of sources.

Creating Groups

A **Group** is a Table in the Library database.

Groups contain the individual records or Objects where you store code and controls. You may find it useful to create simple Groups based on control classes or operations, such as **Listboxes**, **Textboxes**, **Mouse**, **Printing**, and so on. You may also use a Group to store an implementation of a base Object and several variations of the code which may function as derived Objects.

To create a group:

1. Select **Object | New Group** from the menu.
2. A dialog box appears asking you to enter a name for the Group. Enter a unique name.
3. Select the **Fields** you want all Objects in this Table to possess.

If you wish to store controls in any of the records of the table, you must select the **Form** field. Selecting Form also includes the FormCode field. The **Module** field will only store code. You may also add an auxillary Text field for notes, comments, etc. The FormCode, Module, and Text fields are functionally identical, with the following exceptions:

the FormCode field will display event procedures for controls as control procedures if a control of the same name exists on the Form

the Text field is ignored by the Wizards.

4. Click **OK**. A Table will be created in the Library database and the Group will be added to the Group combobox in the Contents Window.

To Select a Group:

1. Click the combobox in the **Contents** Window. The names of all the Groups appear in this list. The names of all the Object records in the selected Group appear in listbox below it.

To Rename a Group:

1. Select **Object | Rename Group** from the menu. An inputbox will prompt you for the new name.
2. A new Table is created and the current Group's records are copied to it. The current Group is then deleted. This is a fairly slow operation, so avoid it if possible.

To Delete a Group:

1. Deleting a Group deletes the Table ***and all the Object records*** it contains. All the items shown in the Object listbox of the Contents window will be deleted when the Table is deleted.
2. Select **Object | Delete Group** from the menu. You will be prompted to confirm. Groupscannot be undeleted.

Creating Objects

An **Object** is a single record in the current Group. It consists of up to 6 fields: its Name, a short description, its binary Control data, its Form code, its Module code, and any supplemental Text you have stored with it.

An Object record is displayed in a Form/Module format similar to that of VB. The Name and description appear in the Contents window. The binary control data is drawn on the Form window. The Form code and Module code are displayed in code windows similar to VB code windows.

You can use an Object record in several ways;

to store related coding techniques, for example, a collection of functions which do numeric conversions, perhaps by alternate methods

to store a number of functions related to a particular task

to store entire forms and all their related code, such as dialogs or custom messageboxes

to store modular components which comprise a block of functionality, such as the code and controls for creating tabs, owner-draw listboxes, custom labels, etc.

To create an Object record:

1. Select **Object | New** from the menu, or click the button with a new page icon.

2. A new record is immediately created with a default name and added to the database. The new item appears in the Object listbox in the Contents window.

To Rename an Object:

1. The **Name** field appears between the Group combobox and the Object listbox in the Contents Window. Change the text and press **Enter** and the name will be immediately updated. This change will be reflected in the Object listbox.

To Update an Object:

You may Update an Object record at any time by selecting **Object | Save** from the menu, but this is seldom necessary, as the record will be updated automatically whenever a new Group or Object is selected, whenever the library is unloaded, or ***whenever the Name is changed***.

To Delete an Object:

1. Select **Object | Delete** from the menu, or click the button with a faded page icon.

2. When you delete an Object, all the fields are cleared and the record is removed from the database. If the Object is the last record in the current Group, all the fields will be cleared and the record renamed **Untitled**, but the empty record will not be deleted until you delete the Table.

3. An Object record cannot be undeleted.

Finding Things

Manually browse the library from the Contents window.

You may search the Library from the Find dialog.

1. Select **Edit | Find**, then select the Fields you wish to search.
2. Enter an expression and click **Find...** . The results will be displayed in a list showing the Group (Table) and Object (record) where the expression was found.
3. Click an item in the **Search Results** list and that item will be displayed by the Library.

For Example:

Assuming you have some code in the Library, enter **Get** or something equally common and search the Name field. You will probably get a fairly long list of items.

Click an item at random. You will see that the Library makes this the current record.

See also Searching a project

Using the Wizards

Import Wizard

Manually copying code from source files to the Library is quite easy, but for situations where the code is scattered, or contains many Constant and DLL declarations, or consists of a large number of event procedures, the Import Wizard can save you some work.

Create a new Object record to avoid importing procedures with name conflicts.

Pay attention to which field you store imported procedures in (Form or Module), as this will make it easier to export the code for reuse. Code which has been written with modularity in mind will be much easier to find and import than code which has not. If necessary, reorganize and edit the code before importing it.

Export Wizard

The Analysis feature will check for name conflicts at the procedural level, but detection of conflicts with global and local variables and constants has not been implemented in this version, so you should consider pasting to a new Module or Form when adding code to a VB project.

Add needed VBXs and DLLs to a project before pasting code and controls in.

Immediately check the results of pasting code to a project by attempting to run it; errors will be easier to correct or remedy if you address them at once.

Designing Objects

One way to implement an Object is to place all the code for a block of functionality in a single separate module.. If the object has multiple instances, the data for each instance may be stored with the instance, and the code to manipulate it placed in the module. This is particularly convenient if the instances of an object are associated with a **Form**. The Visual Basic MDI sample is an example of this kind of implementation.

A more general approach is to define a **Type** structure which holds all the data for each object instance. Allocate a structure for each instance and pass it to the Module when you call its functions. The Module need hold no data at all; it manipulates the data passed to it.

This model works well when objects are not created dynamically, when their number is known in advance, or when they are created in association with a Form. In some cases it is possible to create an array of structures, perhaps maintained by the Module, which contain each objects data. However, if each object uses an array to store data, this will not work, because VB structures cannot contain arrays.

It is generally impractical to create derived Objects by implementing wrappers for one module in other modules (simulating derived classes), but this apparently can be done by message dispatching.

Beware of bugs in VB. VB sometimes loads new instances of a Form when a Form calls a method (implemented by messaging) in another Form. Use a global data structure to pass data between Forms, and call the method from a Module, or use a global structure to pass data between Forms.

Editing Operations

Importing Controls

Controls are brought into the Library by pasting from the Clipboard

To Paste Controls:

1. From the VB design environment, select the controls and copy them to the clipboard.
2. Select **Edit | Paste Controls** or click the button with a clipboard icon.
3. If the Object record already contains control data, you will be prompted to either **Overwrite** or **Abort**. This version of the Library does not support appending the new control data to previously existing data.
4. Open the **Form** window to view the controls and verify that the new data was pasted.

Importing Code

Code text may be pasted from the clipboard directly to the active code window.

Importing Event Code for Controls

1. First, paste the controls to the Form window as described previously. (optional)
2. Next, select the event code in VB and copy it to the clipboard.
3. Click the FormCode window to make it the active window. Then select **Edit | Paste** or click the button with a paste icon.

Importing General Procedures into the Library

1. Copy the procedure to the clipboard, making sure you copy the header and footer (Sub XXX(), End Sub)
2. Click either the FormCode or the Module window to make it active. Then select **Edit | Paste** or click the button with a paste icon.

Importing Code with Import Wizard

The Clipboard is not used when Importing code with the wizard. In this case, the Library will scan a VB source file and extract the procedures and declarations which you select. The VB source file must be up-to-date (issue **Save Project** from VB before begining if the files are open in the VB design environment) and must be saved as text (this version does not read binary source files.)

Selecting an Object to import to
Selecting the Source files to import from
Selecting declarations and procedures to import
Selecting the Object field to store the imported code
Importing the code

Printing

Printing is not directly supported.
To print, select **File | Print to File**.   You will be prompted for a file name.
The current Object record will be saved as text to the file.
This version may not support dumping Control data as text. Check the Readme file.

Editing Code

Editing Text:

The Menu provides **Select All**, **Cut**, **Copy**, and **Paste** functions for manipulating text in the Code windows. These functions work much like the same commands in the VB design environment. When pasting code into a code window, procedures will be seperated and added as new procedures; other text will be inserted at the caret location; ambiguous code will be placed in the (general) (declarations) section.

Adding Procedures:

To add a new procedure, select **View | New Procedure**.

Subs and Functions are added as in VB.

In order to ease the task of locating code, the Library does not provide stubs for control procedures, so to add a control procedure, select the **Event** option and then select the **Control** and the **Event** from the choices offered. For custom events of VBX's, you must type in the name of the event or edit the name of a standard event after you add it.

If you add a procedure for one of the listed controls, it will be recognized as a control procedure by the FormCode window and placed in the Object combobox.

If you are unsure, just add any name or event. You can edit it afterwards in the code window.

Undoing Mistakes:

To undo changes, select **Edit | Revert** before doing anything to cause an Update. The current record will be reloaded from the database and all changes made since the last Update will be lost. (Updates occur whenever a new record is selected, when **Object | Save** is selected from the menu, ***and whenever the Name is edited***.)

Note: The Revert operation applies only to text within a record. Deleted Object records or Groups cannot be recovered.

Moving Code:

There is no way to copy or move an entire record from one Group to another at this writing. You must create a new destination Object record and copy the contents of the one to the other by copying the control data and the text to the Clipboard and then pasting them into the new Object record.

Exporting Controls

To copy Controls:

1. Find the Object record with the controls you wish to copy.
2. Select **Edit | Copy Controls**, or click the button with a control icon.
3. The control data will be copied to the clipboard. Paste it to a VB Form in the normal manner.

Keep in mind that the control data in the library is just like any other VB control data; if controls of the same name already exist on the Form, VB will prompt you to create a control array. If you try to paste VBX controls to a Form and the VBX does not exist in the project, you will generate an error. In both cases, VB will alter properties of the controls you are pasting. The Library will alert you if the controls you are pasting require a VBX, but it does not check for name or other conflicts unless you use the Export Wizard.

Check carefully for potential conflicts before pasting. If a problem arises, abort or undo the operation in VB. Take appropriate action such as renaming the controls on the VB Form or adding needed VBX or DLL files.

To edit the Control data in the Library, paste it to an empty VB Form, edit it, and then copy it back to the Clipboard. Then repeat the copy / paste procedure.   Renaming of controls is supported if you use theExport Wizard.

Exporting Code

To Copy code:
1. Select the code you wish to copy in the Code window.
2. Select **Edit | Copy**, or click the button with a copy icon.

To paste text into a project, open a code window and select **Edit | Paste** from the VB menu.

You may also use the Export Wizard to automate complex copying operations.

Exporting Code with Export Wizard

View Options

Menu and Toolbar

The Menu contains commands for all the operations available in the Library.   The Toolbar provides access to the most frequently needed operations.

The Toolbar may be removed from the display by unchecking the appropriate selection under **Options**.

The **Menu** may also be removed.   To show or hide the menu, double-click the space between the buttons.

Form and Code Windows

Setting the Fonts:

Select **Options | Font** and select a Typeface.   You can also change the background color of windows. The font and colors of code windows containing open source code files may be set independenly of the library code windows.   Click either the **Library** or **File** option, then make adjustments to the font and colors.   Click **Apply** to save the changes.

Code/Text mode:

The Form and Code windows are only shown when a field of their type exists in the current Group table.

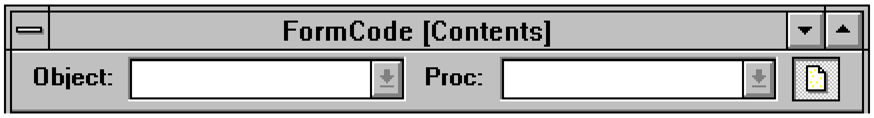
Code windows can be viewed   in **text** mode if they do not exceed 45K. Click the page icon to the right of the Procedure combobox.

FormCode windows can display a graphic representation of Controls if a Form icon appears

Arranging the windows:

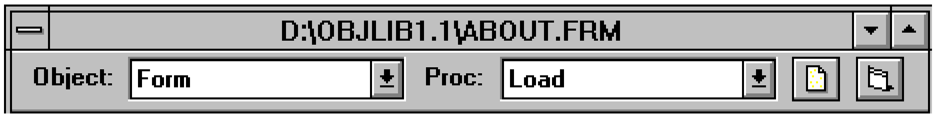
Windows may be cascaded or tiled from the **Window** menu or from the toolbar. When cascading windows, the Library attempts to keep them on the screen and immediately below the Main window.

Toggling between Code/Text views



At the right of each code window is a page icon. When you click the icon, the code window is toggled into text mode and the comboboxes are disabled. If the Library suspects the textbox may be unable to hold all the text in the field, it may refuse to go to text mode.
Warning! You can edit the text in this mode, but you cannot paste text. If you exceed the capacity of the textbox (somewhat less than 50K), some text will be lost and the Library will not know! When you toggle back to code mode, the text will be lost!   See the README file for more information.
Also, be attentive to the possibility of introducing parsing errors when editing text in this mode. Avoid producing text without proper procedure headers (Sub XXX ()) or footers (End Sub).

Toggling between Controls/Text views



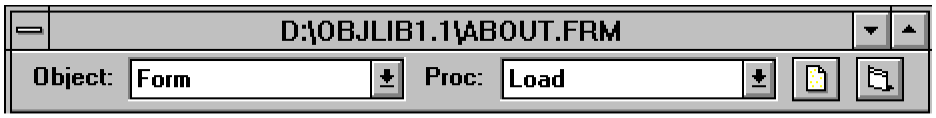
At present, this feature only exists when opening a **.FRM** file.
When you open a .FRM file, you can view a graphic representation of the controls by clicking the form icon (which will display the controls as text), then the page icon (which will display the controls graphically).

See Viewing VB Files

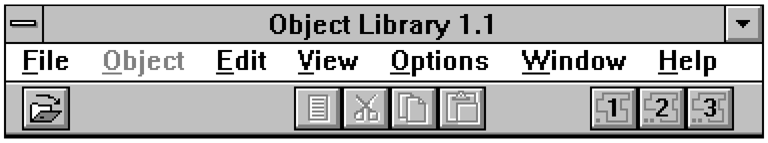
Browsing Files and Projects

Viewing VB Files

You can open and view VB source code files which have been saved as text. Files are opened in code windows similar to those used by VB.



When a .FRM file is opened, an extra **Form** button appears on the window toolbar. Click the Form icon to view the **Controls** section of the source file. While the Control text is displayed, click the **textmode** button (the page icon) to view a graphical representation of the controls. (This feature may be disabled for files whose controls are too long to fit in a textbox. See alsoToggling between Code/Text views.)



The Library can be used as a file viewer without opening the Library database. To do so, select **File | Close Library**. When no library database is open, some toolbar buttons are hidden. You can launch Object Library without opening a database by dropping a file on **OBJLIB.EXE** in File Manager. You can also launch it with the switch **-** (minus sign) to supress opening of the default database.

Object Library supports drag and drop from the File Mamager. Drag a file from File Manager onto the Main window to open the file in a code window.

Editing VB Files

When you open a VB file in a code window, the same kinds of operations are used to edit text as are used for library code windows.   (see Editing code)

To save changes to a VB file, select **File | Save File**.   You will be prompted for a filename.   The default choice is to save the file as text with the extension .TXT.   Although the library will not prevent it, it is not recommended that you overwrite VB source files from the library.

Viewing VB Projects

Use the **Project Browser** to browse a VB project without having to open the project in the VB design environment (perhaps because you already have a project open). The Project Browser will let you get a quick feel for the layout of an unfamiliar project.

You can use the Browser to assist in adding code to a VB project or to the Library. Open a project, then click in any of the lists to view the projects source code. Select some text, then click **Copy** to copy it to the clipboard.

The Project Browser can assist you in determining vital information about projects you are developing. For example, you can see at a glance how many global variables you are using, or what the scope of a particular variable might be. You can see the total number of controls used by a form or by the project, and you can check the data types of functions and variables.

You can also search a project for a variable or function name and determine how many times that item is referenced and from where it is referenced.

See also Project Browser

Keyboard Shortcuts

Editing

CTL+L	Open a library database
CTL+F	Open a file
CTL+P	Print the current Object record to file
CTL+G	Create a new Group table
CTL+O	Create a new Object record
CTL+D	Delete the current Object record
CTL+S	Save the current Object record
CTL+Q	Copy control data to clipboard
CTL+R	Paste control data from clipboard
CTL+Z	Revert
CTL+A	Select text in active window
CTL+X	Cut selected text to clipboard
CTL+C	Copy selected text to clipboard
CTL+V	Paste text from clipboard
CTL+N	Create a new procedure
CTL+DOWN	Previous procedure
CTL+UP	Next procedure
Shift+F3	Open the Find dialog

View

F4	Contents
F5	Form
F6	FormCode
F7	Module
F8	Text
F9	Cascade
F11	Tile
F12	Show all windows
CTL+F12	Hide all windows
CTL+T	Toggle the Toolbar
CTL+M	Toggle the Menu

Dialogs

ESC	Dismiss the dialog
CTL+F5	Open the Project Browser
CTL+F6	Open the Import wizard
CTL+F7	Open the Export wizard

Library Database

Libraries
Fields

Libraries

Other Libraries

The Object Library looks in its directory for **OBJLIB.MDB** when it starts. If it fails to find or open its database, or if you wish to create multiple Library databases, you can open a database by selecting **File | Open Library...**.

Compacting and Repairing the Database

Use Access 1.1 to occasionally compact the database.

The Visual Data sample included with VB can also be used to compact and repair the database.

This version of the Library makes no provision for these operations from within the application itself.

Fields

The Library database contains Tables for each Group you have created. The Tables contain a record for each Object you have created. All the Object records in a particular Table have the same number of fields, but depending on the selections you made when you created the group, the Tables may not all have the same number of fields.

The records contain the code and controls you enter into the Library database. Each record has two required and several optional Fields.

FIELD NAME	**TYPE**	**SIZE**	**WINDOW**
Required fields:			
1. "Name"	text	40	Contents
2. "Notes"	text	256	Contents
Optional fields:			
3. "Form"	binary	0	Form
4. "FormCode"	text	0	FormCode
5. "Module"	text	0	Module
6. "Text"	text	0	Text

The Tables have no Indexes.

You may create new Tables or edit the text fields of records with Access 1.1 or the Visual Data sample app

Warning! Do not alter the names of the fields! The Library will not be able to read a Table which has had its fields incorrectly renamed.

If you wish to experiment with the Tables, create a 'Test' Table or a copy of the database to use for these purposes.

Open File - opens a VB source code file

New - creates a new Object record

Delete - deletes the current Object record

Copy Controls - copies controls from the Library to the clipboard

Paste Controls - pastes controls from the clipboard to the Library

Select - selects the text in the active code window

Cut - cuts the selected text from the active code window to the clipboard

Copy - copies the selected text to the clipboard

Paste - pastes text from the clipboard to the active code window

Shows the **Contents** window

Shows the **Form** window

Shows the **FormCode** window

Shows the **Module** window

Show the **Text** window

Shows all the Library code windows

Hides all the Library code windows

Tiles the Library code windows

Arrange the VB design windows

Arrange the VB design windows

Arrange the VB design windows

Groups - lists all the Groups (Tables) in the Library database

Name - edit the Name of the current Object

Object - list of all the Object records in the current Group

Notes - brief description of the current Object

Object - lists the control procedures

Procedure - lists the general procedures

Code/Text mode - toggles the code window between code and text mode

Code text - displays the text of the current procedure

Combo box

Textbox

Listbox

Label