VBMap - DoDi's VB Project Browser

In big projects, you may loose control over where variables or user types are declared. VB 3.0 can find the location of subroutines, but other declarations are not supported. Also it's impossible to tell where the modules are stored. It may also happen, that the interpreter saves a modified module to a different path. Sometimes you may want to review similar functions from another project and include parts of a module into your current project. This may be impossible if the desired module is stored in binary format and therefore can be displayed only in the interpreter. **VBMap** supports all these functions, shows lists of all global and local symbols and their source code, even if the modules are saved in binary format.

VBMap reads a make file and then looks into the modules found there. The modules may be saved as either text or **binaries**, and even modules from **VB 2.0** are handled.

All symbols, e.g. **types** (type), **constants** (const), **variables** (var) and **subroutines** (sub, fun, ext for Declare) are shown in global and a local list.

The planned display of local variables in subroutines and of all references to a symbol was cancelled, because the tables for these informations may become very large, possibly preventing other programs from running while the tool is active, and I found no practical way to display that many informations. Mail me, if you have an idea how to implement this.

The global list shows all global declarations. For a selected entry, **VBMap** shows its type and the module containing the declaration. With a **double click** on the entry, all the declarations in the module are displayed in the local list.

With a **double click** on a global or local symbol, the source code is shown in a **module window**.

For big projects, you can **reduce** the amount of the information shown. Clicking on the simple buttons (labels) located above the lists toggles the display state of classes of symbols, and the label is greyed in hidden state.

In the global list, you can hide variables, constants, types and subroutines as desired, clicking on '*' reverts to the full display.

In the local list, you can suppress the global symbols already shown in the global list.

In the menu, you can **save** the complete map to a disk file (*.map).

To **view a specific module**, select it in the module list and use the menu to show it.

The **modules are displayed** in splitted windows, just like the interpreter does. From the left ComboBox you can select a **module**, from the right combo box the different **sections** of the module. There are no different lists for the controls of a module, all subs and events are contained in the section box.

You can **copy** selected lines to the **clipboard** or save the whole module **as text** (*.txt) from the module menu in the main window, but these features are disabled in the demo version.

VBDiff - DoDi's VB File Compare

Comparing different versions of a source file requires a tool that can handle situations, where the subroutines are stored in **different order**, or where the case of the names differs.

VBDiff compares any text files, stopping on differences. Then you can select matching lines to continue the comparison.

The files to compare are **selected** from two related lists, where all changes in the left column are reflected in the right column. Selecting a file from the left list automatically selects the file with the same name in the right list, if one is found there. This way you can easily compare similar projects contained in different directories. If there is no corresponding file, you can select any other file from the right list. To start the comparison, double click on one of the desired files.

The differences found are shown in three lists, the list on the top contains the lines common to both files, the lists below show the file contents starting with the different line.

You select from the menu, whether leading spaces are ignored and whether the comparison should be done case sensitive.

To continue the search after differences were found, select matching lines from both files and double click on any of these.

If the interpreter has reordered subroutines in the files, double click on any line with the definition (Sub or Function). Then the matching definition is searched in the other file, and comparison continued from that point. To completely compare such modules, you should always use the same list to restart the comparison, so all sections are compared in the order they occur in this file.

Comparison is implemented only for text files, to compare modules saved in **binary** format, you must convert them to text format before.