VBDis3 - DoDi's Discompiler for Visual Basic 2.0 and 3.0

This decompiler generates the complete source files (including the forms, modules and the MAK file, ready to run the program in the VB environment) from compiled Visual Basic (VB) executables. It is not designed to "crack" someone else's programs but to demonstrate how many parts of the source code are included in the executables by the compiler. I find this behaviour alarming - for what program are these pieces of information meant? The runtime system does not need them, and yet they were intentionally taken over in the program.

The installation process of the Discompiler is easy:
If you got an **archive**, just copy it to a new directory (e.g. VBDIS). If it is an EXE (e.g. VBDISARC*.EXE), simply start it to get the files, else extract the files using the appropriate unarchiver. If you want to save disk space, you may delete the archive now.

Now you should find the programs VBDIS3E.EXE, VBCTRLE.EXE and several data files that are needed for decompiling. A special installation under Windows is not required; the Discompiler needs no entries in WIN.INI etc. The programs **require Visual Basic 3.0** to be installed on your system, the file VBRUN300.DLL is part of this and therefore not included.

If you have installed **Visual Basic 4.0**, some further steps might be necessary: The program reads the **vbpath=** entry in [Visual Basic] from VB.INI, and in that path looks for the file AUTOLOAD.MAK. The first line in this file is expected to contain the name and possibly the path of a VBX file. If you had to edit the vbpath entry in VB.INI, you should restart Windows to flush the INI cache!

Sporadic occurring errors during the start-up phase of the program were reported when running under **Windows 95**. If any errors occur before the first window is displayed they are caused by an insufficient co-operation between Visual Basic (VBRUN300.DLL) and Windows 95. The programs run fine with Windows 3.x, therefore such errors are due to Windows 95 and should disappear with one of the next releases of Windows 95.

The usage of the Discompiler is easy as well:

- Start the program, choose 'Open' from the File menu and select the desired program. If the program was created with neither VB2 nor VB3, try another one.

- The 'Select Project Directory' dialogue will pop up and you are asked to specify the directory where the source files shall be generated. Pressing 'New' will allow you to create a new directory. Note: The chosen directory must be displayed as opened before clicking on 'OK'.

- Now the Discompiler starts up and creates the source files. Don't worry about the error-messages that might pop up - just choose OK or IGNORE. Note: The Discompiler won't decompile itself or other protected programs.

- In pass #1 the Discompiler generates several BAS files in **text** format, and the FRM files in **binary** format. As soon as the Discompiler completed pass #1 (it will display "Now save *.FRM as text, please"), you should run VB, load the created MAK file (don't care about possible error messages) and save all forms **as text**. This step is unfortunately necessary, since it is often impossible to create the forms as text without a little help from the custom controls (VBX). Then you should save the project and leave the VB environment.

- Now switch back to the Discompiler and choose 'Combine forms' from the File menu to merge forms and code. (This could also be done using an editor of course, but not that fast)

- Finished! The source code can now be executed within the interpreter.

If untyped variables occur in the sources, you can request another pass through the program, using Improve from the menu. Then you repeat the above steps, saving and combining the forms, and check the result with the interpreter. You can repeat this step until all variables are typed, or until no more changes occur. The remaining untyped variables are almost really unused in the program, it is very unlikely that the parameters of unused subroutines can be typed this way.

Errors might be reported during the process of decompiling, but they should (hopefully) not cause the program to crash. Please pay attention to the comments: The underlying tables were generated without decompiling the runtime system, so they might not be complete.
Some constructions in your programs might also cause errors, if they are not yet known and the Discompiler therefore does not know how to handle them. Special custom controls cannot be handled in the shareware version of **VBDis**, use **VBMDis** instead, with many more capabilities to optimise your programs

The descriptions of **custom controls** are stored in files with the extension '300' (for VB 3.0). If a program makes use of additional VBXs which are not part of the VBXs shipped with VB 3.0, the Discompiler can't analyse this program properly. That's the job of **VBCtrl**, a program that will analyse VBX files and save information about them for the Discompiler. For details refer to **VBMDIS.WRI**, please. In the registered version of **VBDis** however, you can use the Ignore button to continue execution, but the Discompiler will create erroneous code in all places, where the properties and events of an unknown control are accessed.

When running the decompiled program in VB for the first time, the interpreter might find some errors, mainly not defined or wrong data type errors, and wrong arguments when calling functions. You'll have to find out the correct declarations/arguments yourself, an extensive search for missing data and function types is not part of this version, and also the naming of variables and functions is only possible with the professional version.
Comparing the code with your sources, you may miss variables and constants, or find strange parameter lists in subroutine declarations of the form (p,p,p); all this normally stems from unused variables or subroutines never called in the program, and therefore the Discompiler couldn't locate them or assign them a data type.

In the sources created by the Discompiler, pay attention to the names of forms and controls that the compiler included with much expenditure exactly like they are in the source code, as well as the exact indentation of all lines. With every usage of a variable is accurately noted whether it was written with or without a type character. You can prove all this easily with your own test programs, comparing the created code with your sources.

Notes

The registered version of the Discompiler offers you some advantages, mainly you won't get stuck on unknown custom controls, and you get the descriptions of all controls shipped with VB 3.0. Registration is possible in CompuServe with SWREG (VBDIS3, ID 7807), a direct registration with the author from outside the European Community can cause a considerable charge for handling and money transfer (refer to REGISTER.WRI for details).

For the **optimisation** of your programs try **VBMDis**, with features like
- Handling of very large programs
- Handling of all custom controls
- Direct comparison of your sources with the created code
- Display of the interpreter and compiler tokens
- Module and subroutine names taken over from your source code
- Free definition of the names and types of variables

with **possible** extensions (not guaranteed to come)
- Hints for program optimisation
- Optimisation of programs after compilation
- Code display in other languages (C++, Pascal)

For details, see **VBMDIS.WRI**

Please don't flame Lutz Lesener <les@uni-muenster.de> for mistakes in the English docs.
He has done his best, but I introduced new mistakes when I updated his translation ;-)
Therefore, flames are welcome to Hans-Peter Diettrich <100752.3277@compuserve.com> or <Hans-Peter_Diettrich@s.maus.de>