Christopher Greaves The Landfall Garden House
60 Canon Bayley Road; PO BOX 1452 Bonavista; CANADA A0C 1B0

709-218-7927

Contents

What This Application Does For You

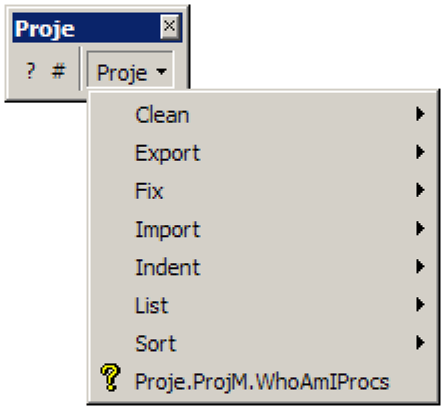
This application automates the laborious process of project management in your Microsoft Office applications.

Typically you have completed development of an end-user application in Excel, PowerPoint or Word, and realize that many experimental procedures are scattered around. They will be of no use in the final product, but how to track them down?

Project manager is the answer!

One-Click Use

After installing the application template Proje.dot, load Word, run one of the macros from the Proje toolbar.



Session Environment

Proje stores your preferences in a file called Proje.INI within your Documents And Settings folder tree:

```
[Proje]
ClassModules=True
CodeModules=True
FormsModules=True
DocumentObjectsModules=True
Macros=False
PrivateProc=True
```

```
PublicProc=True
ExportFolder=B:\
FlushFolder=True
FlushProject=True
ImportFolder=B:\
CleanFolder=B:\
IndentationCharacter=SPACE
IndentationIncrement=4
DocumentationModuleName=True
DelimiterDocumentation=
ForceUpperCase=False
Indent=4
```

Reporting Suggestions

Please use the Contact details from the web site www.chrisgreaves.com.

Summary of Operations

Task	Description	W	E	P
Clean	Reduce bloat by Exporting and immediately re-importing your source code modules.	X	X	X
Export	Export your modules to an archive folder	X	X	X
Fix	Correct elements of poor coding style while maintaining correct syntax	X	X	X
Import	Import your modules from an archive folder	X	X	X
Indent	Carefully re-indent your code so that all constructions are neatly nested.	X	X	X
List	Document your program code, producing listings in documents.	X	X	X
Sort	Sort your procedures in ascending or descending alphabetic sequence within each module	X	X	X

Generate	Re-generate a copy of your template based on all source modules in a prescribed set of folders.			
Harvest	Trawl your hard drive looking for VBA code and assemble it into a searchable source library.			
Locate	Search for and report on VBA code across your hard drive			
Migrate	Copy or Move procedures from one application project to another			
PXref	Build a cross-reference array of your project and provide you with the tools to build your own project analysis tools!			
Strip	Remove deadwood – un-referenced identifiers (data and procedures) from your project.	X	X	X

The GUI Forms

Modules

At least one of the four modules switches must be checked ON.

Public/Private

At least one of these switches must be turned ON.

Macros

I believe that macros (Subroutine procedures with no parameters) are meant solely for the user interface. All other procedures should be functions with parameters that return a result.

Clean

Reduce bloat by Exporting and immediately re-importing your source code modules.

Rob Bovey's Word Code Cleaner is still the best at http://word.mvps.org/FAQs/MacrosVBA/TemplateBloat.htm

Rob has an Excel cleaner, too, and I leave the stripping-of-comments to him.

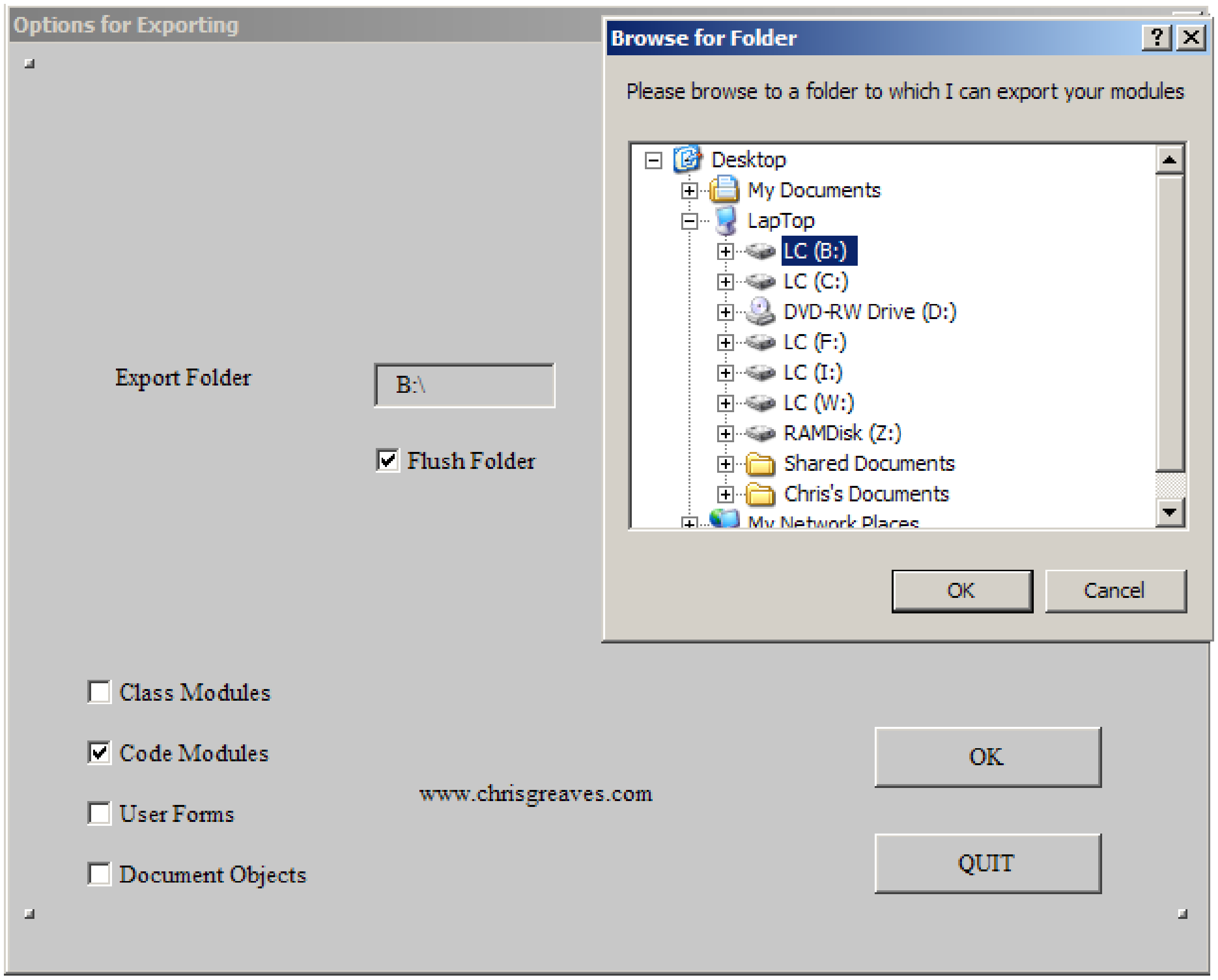
However, I will export and re-import your source modules from Excel, Word *and* PowerPoint.

Why did I bother? Easy! I already had an "Export" module and an "Import" module. It was the work of a lazy five minutes to hook them together.

Please read the two sections "Export" and "Import" for more details.

Export

Export your modules to an archive folder.



Please browse to a folder to which I can export your modules

Export Folder: B:\

☑ Flush Folder

☐ Class Modules

☑ Code Modules

☐ User Forms

☐ Document Objects

Double-click the export folder text box to browse to an existing folder where your source code will be saved.

Check ON the check box if you would like Exporter to flush the folder contents before starting the export process.

You must choose to export at least one of the four types of modules.

Choose OK to continue, at which time you will browse to a VBA application. The appropriate modules of that (unlocked!) application will be written to the folder.

Fix

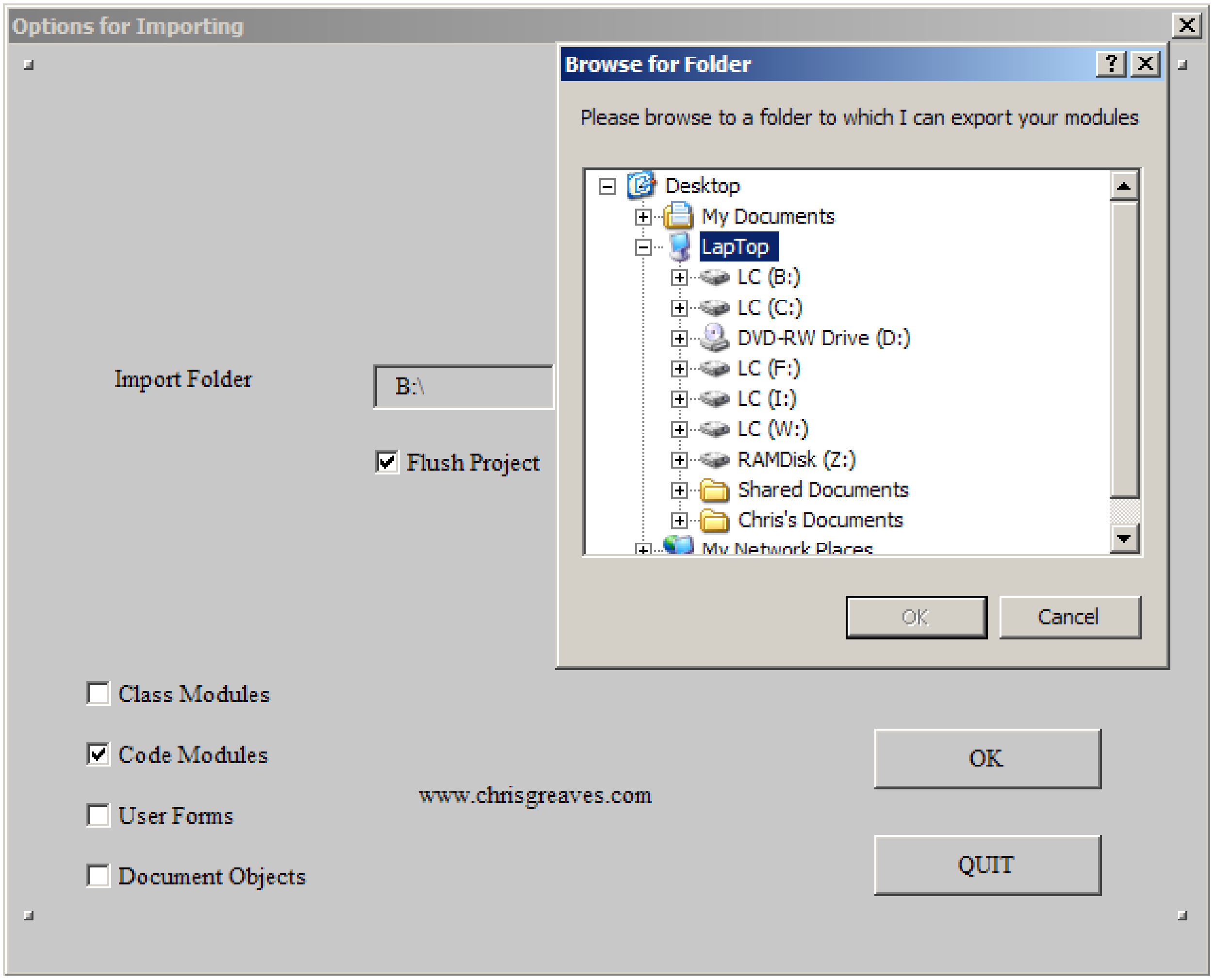
Correct elements of poor coding style while maintaining correct syntax.

History: In 1999 I wrote “Nestr”, a rules-based device for applying correct nesting (indentation) of source code. Nester took its rules from a text file, and could therefore be customized to any nestable language, not just Visual Basic, but ALGOL, C, Plasyd, and so on.

I used Nester to indent client projects, and found that weak syntax was used, typically “If” and “Then” on the same line of code, or joined lines with the colon character.

I wrote some VB-specific code to cope with this, and that code is embedded in Proje as “Fixer”; it tries to “fix” poor VB coding patterns.

Import



Import your modules from an archive folder

Double-click the import folder text box to browse to an existing folder from which your source code will be loaded. This may well be the folder you chose for export purposes.

Check ON the check box if you would like Exporter to flush the project file of all modules before starting the import process.

You must choose to import at least one of the four types of modules.

Choose OK to continue, at which time you will browse to a VBA application. The appropriate modules of that (unlocked!) application will receive the contents of the folder.

Indent

Carefully re-indent your code so that all constructions are neatly nested.

But see also the section titled “Fix”.

Pasting code often results in poor indentation, and that makes code difficult to read, and that increases the probability of error.

Perhaps you have inherited some “stripped” code from the Word Code Cleaner, with all white space (including indentation) removed.

Indent will indent your code according to a set of rules that defines the various statement blocks (such as While-Wend, For-next, If-Then-Else-Elseif-Endif).

26				
DO	0		4	
LOOP	-4	DO	0	
IF	0		4	
ELSE	-4	IF/ELSEIF	4	Y
ELSEIF	-4	IF/ELSEIF	4	Y
END IF	-4	IF/ELSE/ELSEIF	0	
FOR	0		4	
NEXT	-4	FOR	0	
SELECT	0		4	
CASE	-4	SELECT/CASE	4	Y
END SELECT	-4	SELECT/CASE	0	
WHILE	0		4	
WEND	-4	WHILE	0	
WITH	0		4	
END WITH	-4	WITH	0	
SUB	0		4	
PUBLIC SUB	0		4	
PRIVATE SUB	0		4	
FUNCTION	0		4	
PUBLIC FUNCTION	0		4	
PRIVATE FUNCTION	0		4	
END SUB	-4	SUB/PUBLIC SUB/PRIVATE SUB	0	
END FUNCTION	-4	FUNCTION/PUBLIC FUNCTION/PRIVATE FUNCTION	0	
TYPE	0		4	
PUBLIC TYPE	0		4	
PRIVATE TYPE	0		4	

END TYPE	-4	TYPE/PUBLIC TYPE/PRIVATE TYPE	0	

Above is a Rules Table for Office 2003 VBA code.

Column (1) The string that signals the start of a nested structure in the language.

Column (2) Indentation increment for following statements.

Column (3) One or more strings that signal the end of a nested structure in the language.

Column (4) Indentation increment for following statements.

Column (5) Whether the start of structure is to be re-pushed onto the stack

Row (1) The zero-based number of rules in the table. The example table has 27 rules numbered 0, 1, 2 ..., 25, 26.

Example (1) The strings “SUB”, “PUBLIC SUB” and “PRIVATE SUB” signal the start of a procedure known as a subroutine in VBA.

When we encounter any one of these strings (as the leading string in a line) we change the indentation by zero (column 2) because we do not indent the start of a subroutine.

All lines following the first line of a subroutine are to be indented by 4 spaces (column 4).

When we encounter the string “END SUB” as the leading string in a line, we immediately reduce the indentation by 4 (column 2) before emitting the End SUB string.

Example (2) The string “IF” signals the start of an IF-statement in Office VBA.

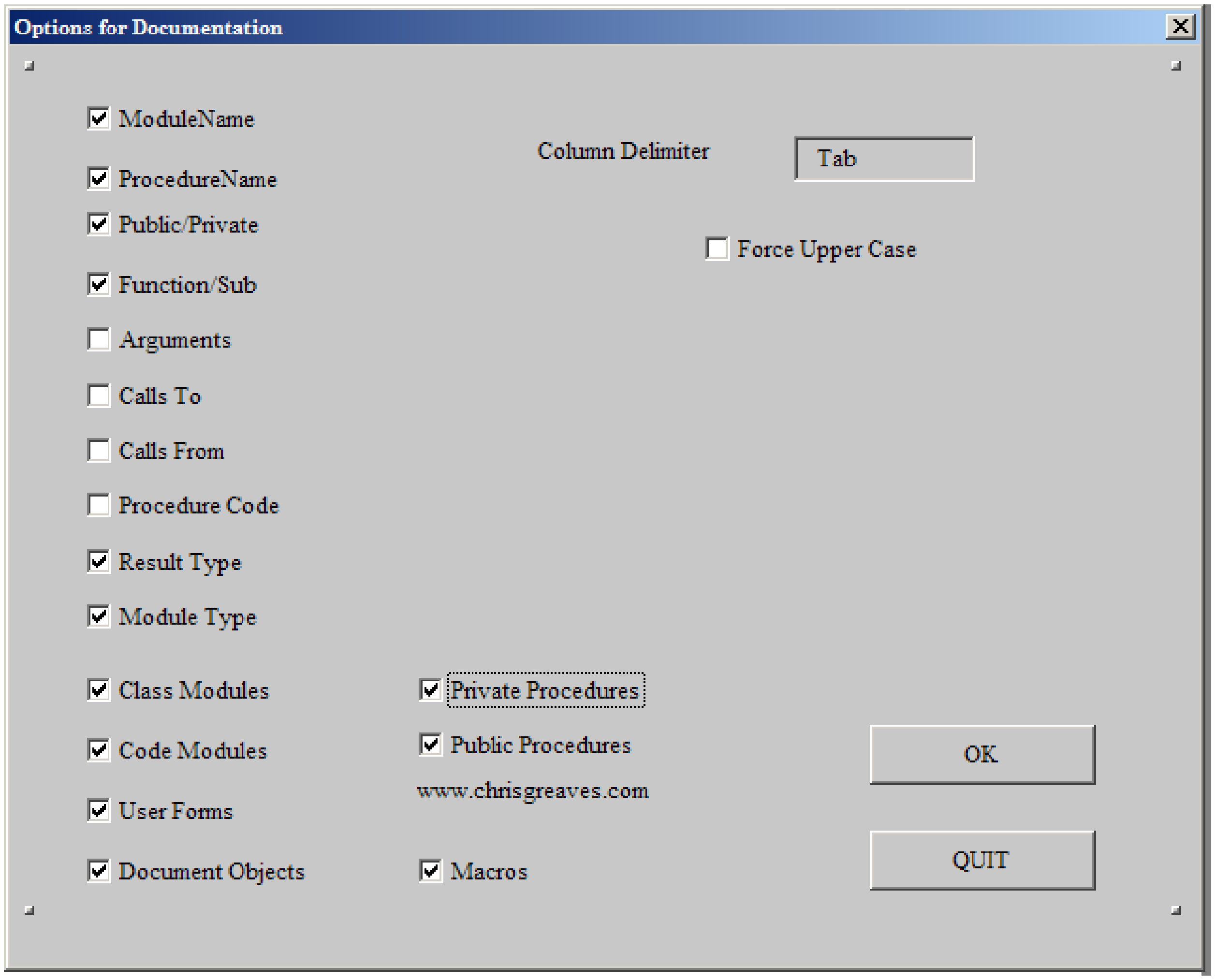
The string “ENDIF” signals the end of an IF-statement in Office VBA.

The strings “ELSE” and “ELSEIF” signal a portion of, but not the end of, an IF-statement in Office VBA. As such we need to display them with a decremented indentation but immediately push back the indentation to its previous level. Only an “ENDIF” statement can brings us back to the previous level for good.

List

Document your program code, producing listings in documents.

Choose a List GUI macro. You will be shown an options screen:


Options for Documentation

- [✓] ModuleName
- [✓] ProcedureName
- [✓] Public/Private
- [✓] Function/Sub
- [ ] Arguments
- [ ] Calls To
- [ ] Calls From
- [ ] Procedure Code
- [✓] Result Type
- [✓] Module Type
- [✓] Class Modules
- [✓] Code Modules
- [✓] User Forms
- [✓] Document Objects

Column Delimiter: Tab

- [ ] Force Upper Case
- [✓] Private Procedures
- [✓] Public Procedures
- [✓] Macros

www.chrisgreaves.com

In the example above I am asking for columns of data to represent, for each procedure in my project, the module and procedure names, whether the procedure is public or private, whether the procedure is a function or a subroutine, the type of result it returns (if it is a function) and the type of module which houses it.

I am asking that all types of modules be reported.

The results will be written to a Tab-delimited form in a new document.

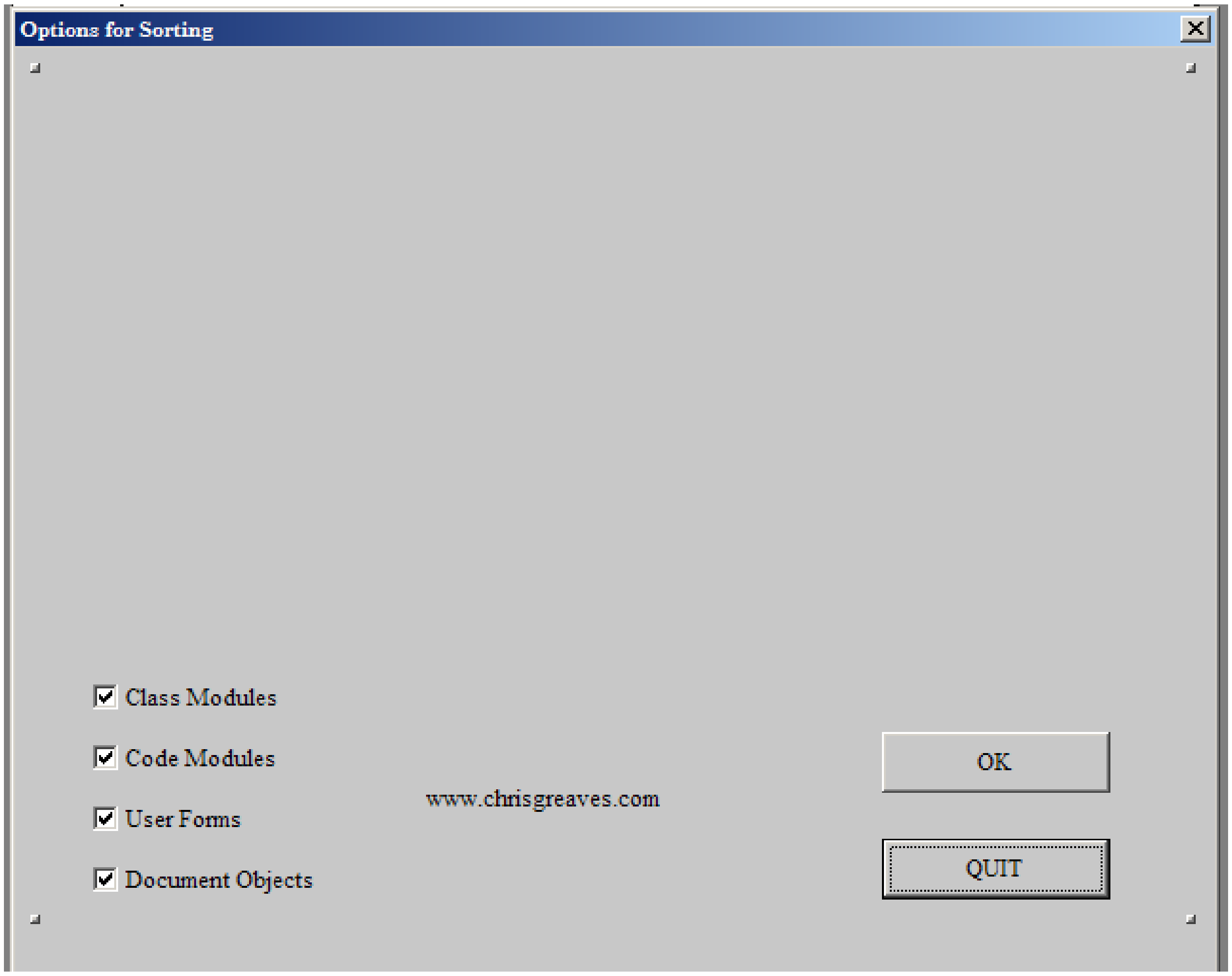
The paragraphs can then be converted to a table form, or exported for further analysis:

Module name	Procedure name	Public/Private	Function/Sub	Result type	Module type
Browsers	strBrowseFile	PUBLIC	FUNCTION		1
Browsers	strBrowseFolder	PUBLIC	FUNCTION	STRING	1
DriveMapping	GetUNCPath	PUBLIC	FUNCTION	STRING	1
DriveMapping	MapDrive	PUBLIC	FUNCTION		1
PassWMacros	EncipherText	PRIVATE	FUNCTION	STRING	1
PassWMacros	strAlphaDigitsOnly	PRIVATE	FUNCTION	STRING	1
PassWMacros	strPasswordOfTheTime	PUBLIC	FUNCTION	STRING	1
PassWMacros	strUnique	PRIVATE	FUNCTION	STRING	1
PassWMacros	strPasswordOfTheDay	PUBLIC	FUNCTION	STRING	1

Sort

Sort your procedures in ascending or descending alphabetic sequence within each module.

Choose a Sort GUI. You will be presented with a set of options:

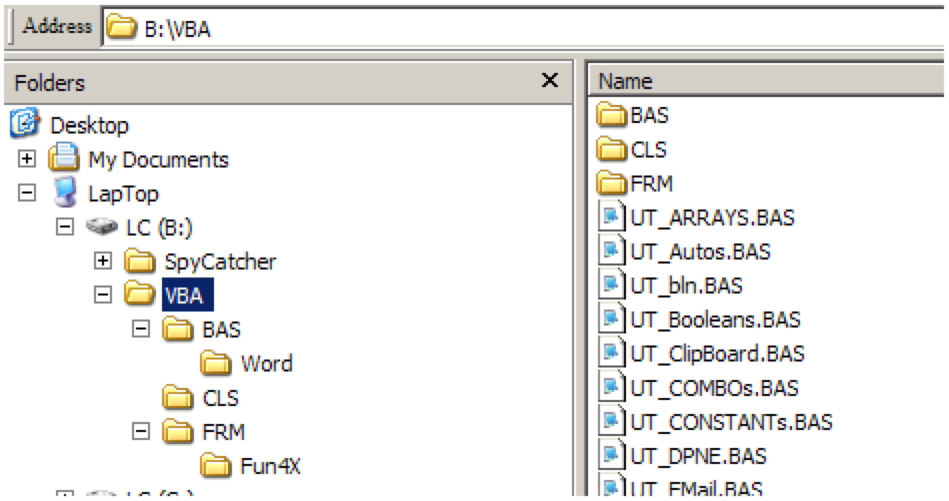


Browse to a project file. You will observe the status as blank lines are removed and each module is sorted.

Generate

Re-generate a copy of your template based on all source modules in a prescribed set of folders.

Start by storing (use "Export") modules of code in folders on your hard drive:



Generation is extremely useful when you maintain several sets of source modules.

In one folder you can maintain pure VB code that will satisfy any Microsoft office desktop application. In a separate folder maintain a set of source modules with VBA code that works on Word objects; another folder holds code for Excel objects, and so on. You spot some handy code for PowerPoint? Paste it into a text file and store that text file as a module in your PowerPoint folder.

Assembly of a library of code is now a matter of pointing to the folders that contain the appropriate files or modules of VBA source.

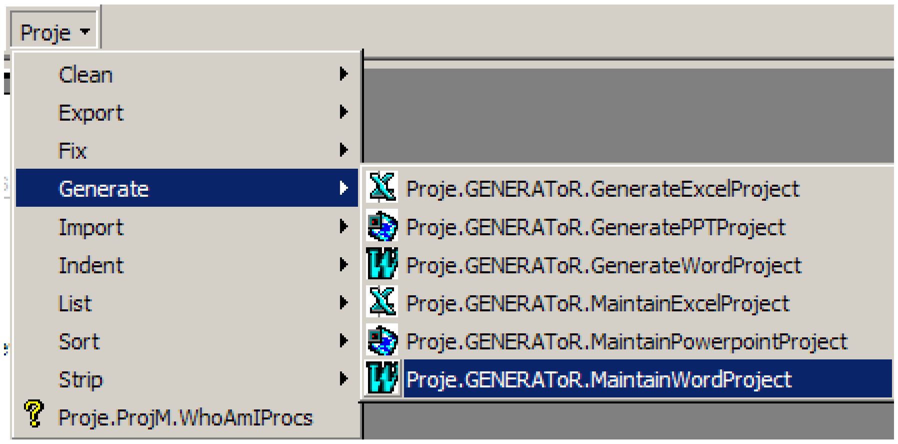
Update your pure VBA code, and the next generation of any library will obtain that updated source!

Maintenance

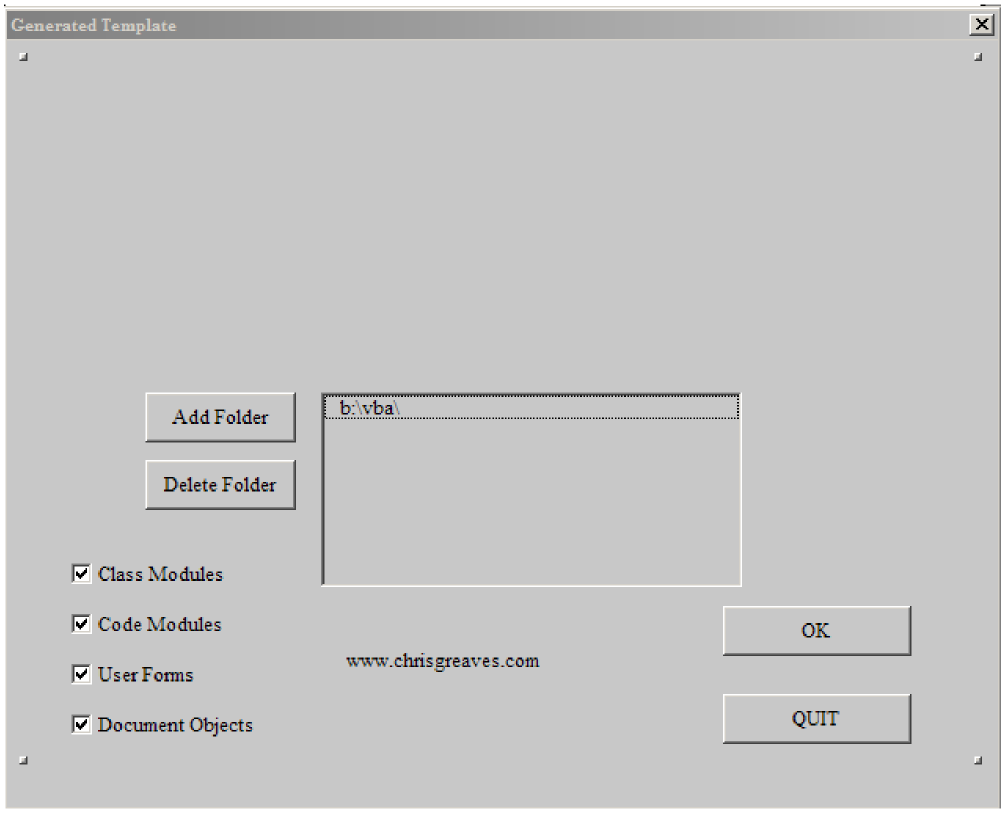
We maintain a list of folders whose files (source modules) will contribute to the project.

Create a new project file.

Run the appropriate generator macro


Proje
Clean
Export
Fix
Generate
Import
Indent
List
Sort
Strip
Proje.ProjM.WhoAmIProcs
Proje.GENERAToR.GenerateExcelProject
Proje.GENERAToR.GeneratePPTProject
Proje.GENERAToR.GenerateWordProject
Proje.GENERAToR.MaintainExcelProject
Proje.GENERAToR.MaintainPowerpointProject
Proje.GENERAToR.MaintainWordProject

You will be asked to browse to your new (empty) template, and then will be offered a GUI form with which you can Add folders to your template.



b:\vba\

☑ Class Modules

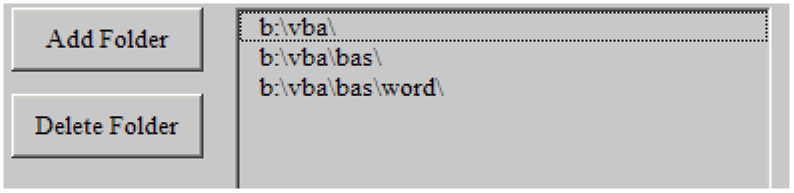
☑ Code Modules

☑ User Forms

☑ Document Objects

Choose the Add Folder command or double-click in the list box to browse to any folder you would like to add. You'll need to identify a genuine module file before the folder will be accepted.

Add every folder that contains modules you need in your project.



b:\vba\
b:\vba\bas\
b:\vba\bas\word\

Choose OK to assemble the modules from your chosen folders into your project.

Generation

Once your project has a set of folder identifiers, you can (re-)generate the folder by using the Generator macros.

Each generator macro obtains the list of folders stored in your project during the maintenance phase, and imports each appropriate source module from each folder in your list.

Harvest

Trawl your hard drive looking for VBA code and assemble it into a searchable source library.

A Computer Network is an array of Storage Devices.

A Storage Device is an array of Folders

A Folder is an array of Projects (a.k.a. Documents, Workbooks, Presentations etc.)

A ProJect is an array of Modules (Module, Class Module, User form).

A Module is an array of Procedures (Functions, Subroutines)

A Procedure is an array of Lines, but since we deal in procedures we can treat a procedure as a string of lines of text, each line terminated by a vbCrLf.

If we maintain the entire set of procedure lines in a single string, we can quickly search that string for text, and for each found item, we must be able to locate the parentage of procedure, Module, project, Folder and Storage Device.

Before you run the macro HarvestCode, be warned that turning it loose on a single hard drive can take a long time to run. You may instead want to browse to a small folder while you experiment with *Harvesting* and (using the library so generated) *Searching*.

Many of your files may be difficult for Harvester to handle. If Harvester baulks, consider running the macro HarvestCodePreProcess, which works like Harvester except that once it has opened a file for inspection, it does not process it. This is a fast way of making sure that all your files can be opened by Harvester. Once HarvestCodePreProcess has run to conclusion, run HarvestCode on the safe set of files.

When either macro baulks, you will find at the foot of your LogFile.txt the name of the file that caused the problem. Either fix that file and restart, or else append that file's name to the file Exclude.txt so that HarvestCodePreProcess will ignore it in the future. You can work your way through the files in Exclude.txt at a later date to determine why they won't open.

Both HarvestCodePreProcess and HarvestCode permit you to browse to a folder.

Start by pointing HarvestCodePreProcess to the root folder of your hard drive C:\ and let it run. When it baulks, copy the file name to Exclude.txt but make a note of the path; there may be other faulty files within that folder. Restart HarvestCodePreProcess by browsing to that specific folder – there is no point in wading through previously-accessed "good" files if we don't need to. Once that folder is processed, revert to the attack on the root drive of C:.

You will find the auxiliary macros HarvestCodePreProcessRepeat and HarvestCodeRepeat save you browsing to the same folder in succession – they repeat the operation based on the most recently browsed folder.

Locate

Search for and report on VBA code across your hard drive

Migrate

Copy or Move procedures from one application project to another

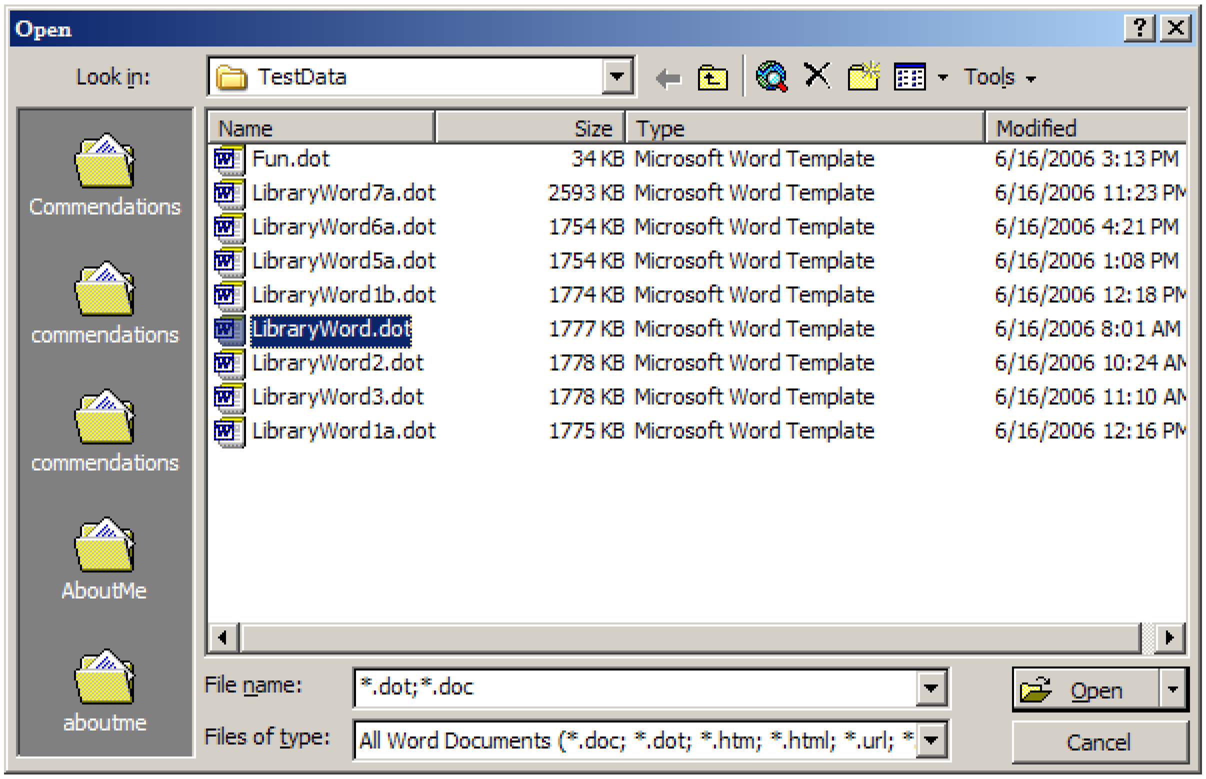
PXref

Build a cross-reference array of your project and provide you with the tools to build your own project analysis tools!

Strip

Remove deadwood – un-referenced identifiers (data and procedures) from your project.

Run one of the Stripper macros. You will be prompted to open an unlocked project file, preferably free of syntax errors:



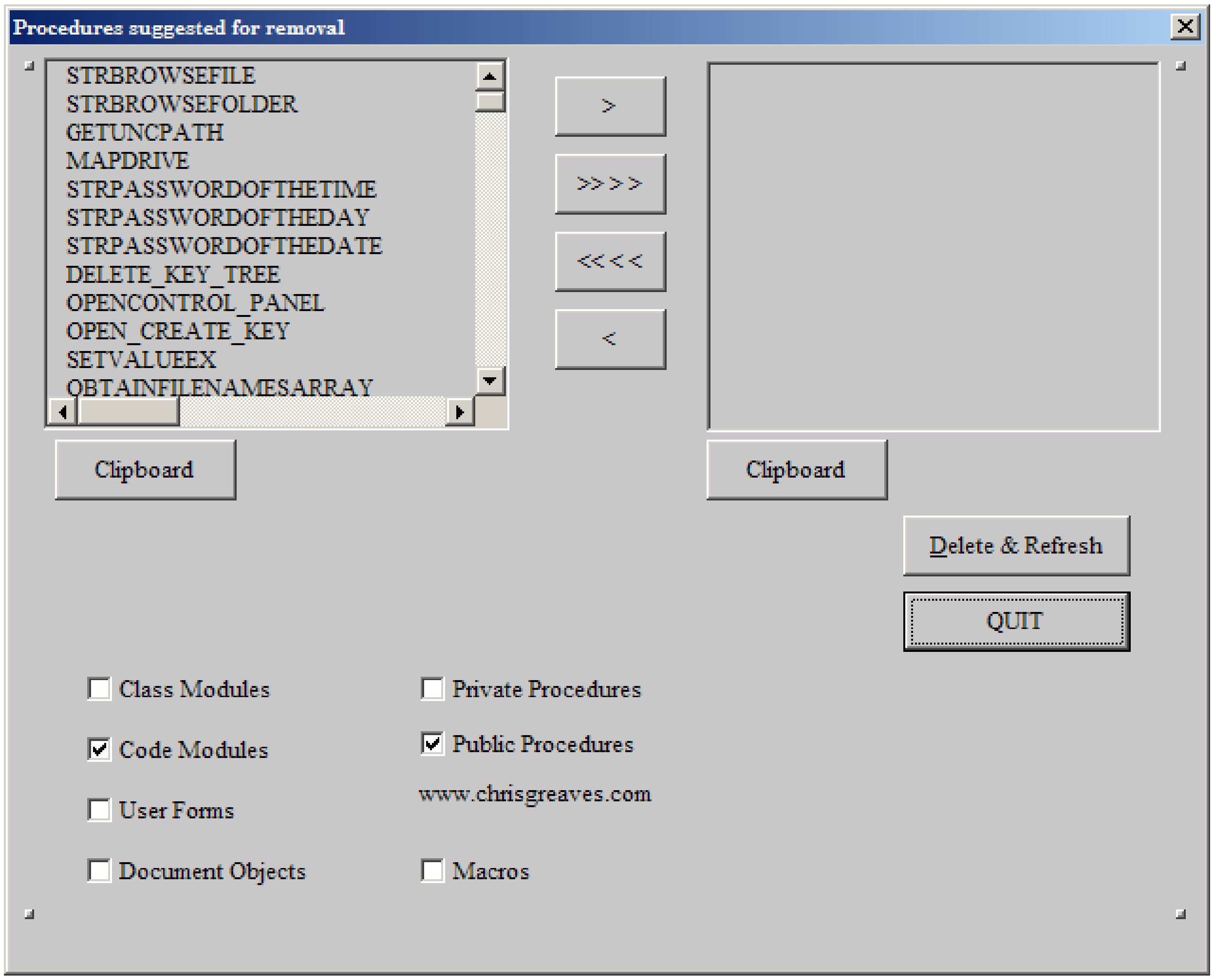
Look in: TestData

Name	Size	Type	Modified
Fun.dot	34 KB	Microsoft Word Template	6/16/2006 3:13 PM
LibraryWord7a.dot	2593 KB	Microsoft Word Template	6/16/2006 11:23 PM
LibraryWord6a.dot	1754 KB	Microsoft Word Template	6/16/2006 4:21 PM
LibraryWord5a.dot	1754 KB	Microsoft Word Template	6/16/2006 1:08 PM
LibraryWord1b.dot	1774 KB	Microsoft Word Template	6/16/2006 12:18 PM
LibraryWord.dot	1777 KB	Microsoft Word Template	6/16/2006 8:01 AM
LibraryWord2.dot	1778 KB	Microsoft Word Template	6/16/2006 10:24 AM
LibraryWord3.dot	1778 KB	Microsoft Word Template	6/16/2006 11:10 AM
LibraryWord1a.dot	1775 KB	Microsoft Word Template	6/16/2006 12:16 PM

File name: *.dot;*.doc

Files of type: All Word Documents (*.doc; *.dot; *.htm; *.html; *.url; *

Stripper will open the project file and analyze its contents. You will observe the status meter on the lower left end of your status bar.



Chosen settings: [✓ Code Modules], [✓ Public Procedures]; Class Modules, User Forms, Document Objects, Private Procedures and Macros unchecked.

The left-hand panel represents procedures which are orphaned; that is, procedures which are called by no other procedures in the project.

You could move them to the right-hand panel and choose “Delete and Refresh” (the left-hand panel) providing that there is no reason to retain these procedures.

However, if one of your procedures is an end-user macro, then it may well not be called at all from your project, but you would not want to delete it. Turning the Macros switch OFF will inhibit the display of such procedures.

Likewise you can restrict display (and hence potential deletion) to only Public or only Private procedures. You can restrict display by type of module.

Note too that if your application is meant to be called from other applications, some of your procedures may not be referenced from this application and may therefore appear to Stripper as orphans!

You may need several passes of “Delete and Refresh” because deleting an orphaned procedure that calls other procedures may well orphan those procedures!

Compare

You have two projects, typically two versions of the same application, and you would like to know which is the more current in terms of content.

The comparison processor accepts two projects and performs comparisons at succeedingly intimate levels.

Do the module names match on a 1-1 basis between the two projects?

If not, explore the mis-matched modules in greater details.

Do the procedure names within each module match on a 1-1 basis between the two projects?

If not, explore the mis-matched procedures in greater details.

Do the procedure bodies (content) within each module match on a 1-1 basis between the two projects?

If not, explore the mis-matched procedure bodies in greater details.

A summary report available in batch processing mode should show:

A list of mis-matched module names

A list of mis-matched module.procedure names

A list of mis-matched module.procedure bodies.

Problems

Please see also "Messages from Microsoft Word"

Any automated process that covers hundreds or thousands of documents is likely to run into problems.

Problems fall into two categories:

1. My problems
2. Not my problems (your problems!)

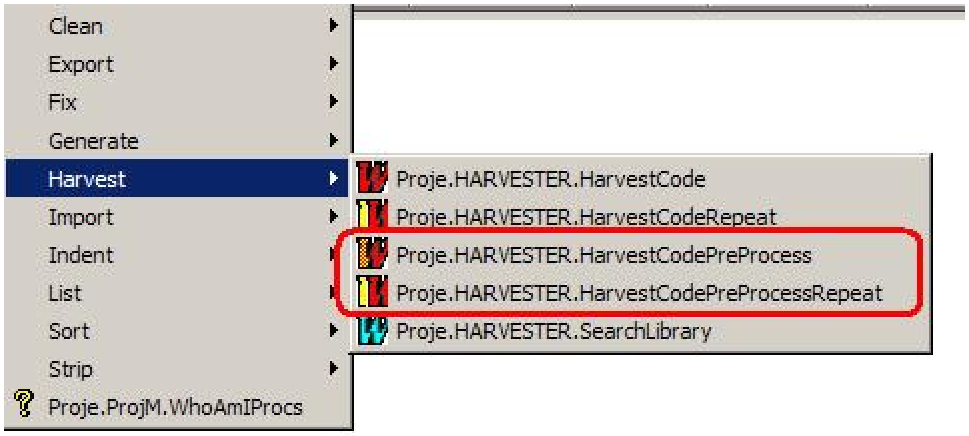
My problems are errors in my programming that cause one of my programs to crash. Please notify me of such problems. I strive to fix them as soon as possible, if only because they are likely to affect me if they have already affected you!

Your problems are most likely caused by faults within your project documents and templates. In this section I will try to show the most common symptoms and the best methods for circumventing them.

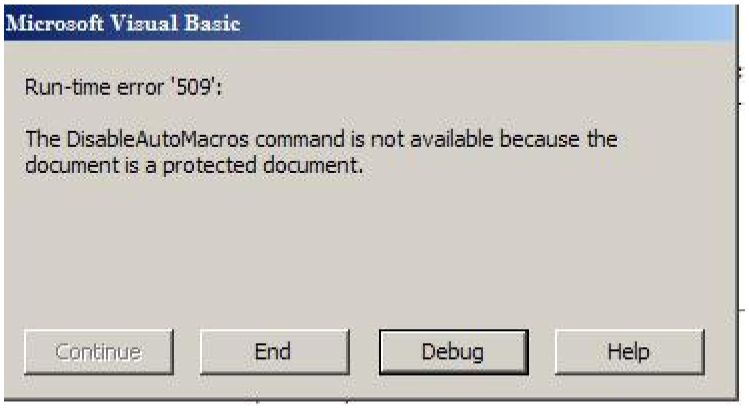
In most cases the problems arise because projects are unclean, that is, they have syntax errors, broken references, and similar characteristics.

Be aware of the two quick-running macros HarvestCodePreProcess and HarvestCodePreProcessRepeat. HarvestCodePreProcess will work its way through your files, attempting to open each one, without incurring the CPU and disk cost of analyzing and writing library code. HarvestCodePreProcessRepeat will save you the task of navigating to the folder of the previous run.

I find it helpful, after a problem is resolved, to run the HarvestCodePreProcess macros from the most local (lowest-level) folder in an attempt to locate any other problem documents in that folder, then I run HarvestCodePreProcess from the next higher, parent level, and so on, working my way back to the original level. This can be a low-cost way of ferreting out other problem documents.


Clean
Export
Fix
Generate
Harvest
Import
Indent
List
Sort
Strip
Proje.ProjM.WhoAmIProcs
Proje.HARVESTER.HarvestCode
Proje.HARVESTER.HarvestCodeRepeat
Proje.HARVESTER.HarvestCodePreProcess
Proje.HARVESTER.HarvestCodePreProcessRepeat
Proje.HARVESTER.SearchLibrary

Protected Document

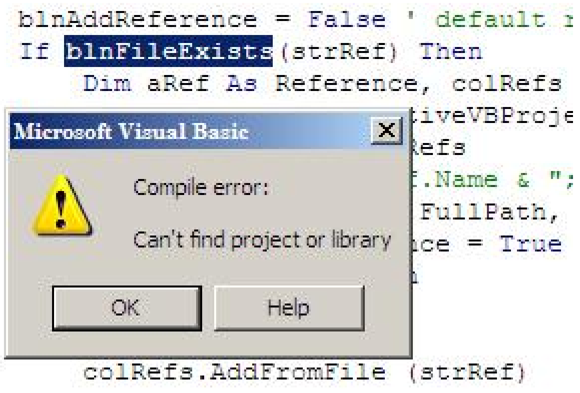

Microsoft Visual Basic

Run-time error '509':

The DisableAutoMacros command is not available because the document is a protected document.

In Word, choose Tools, unProtect document, save the document and restart processing.

Broken References



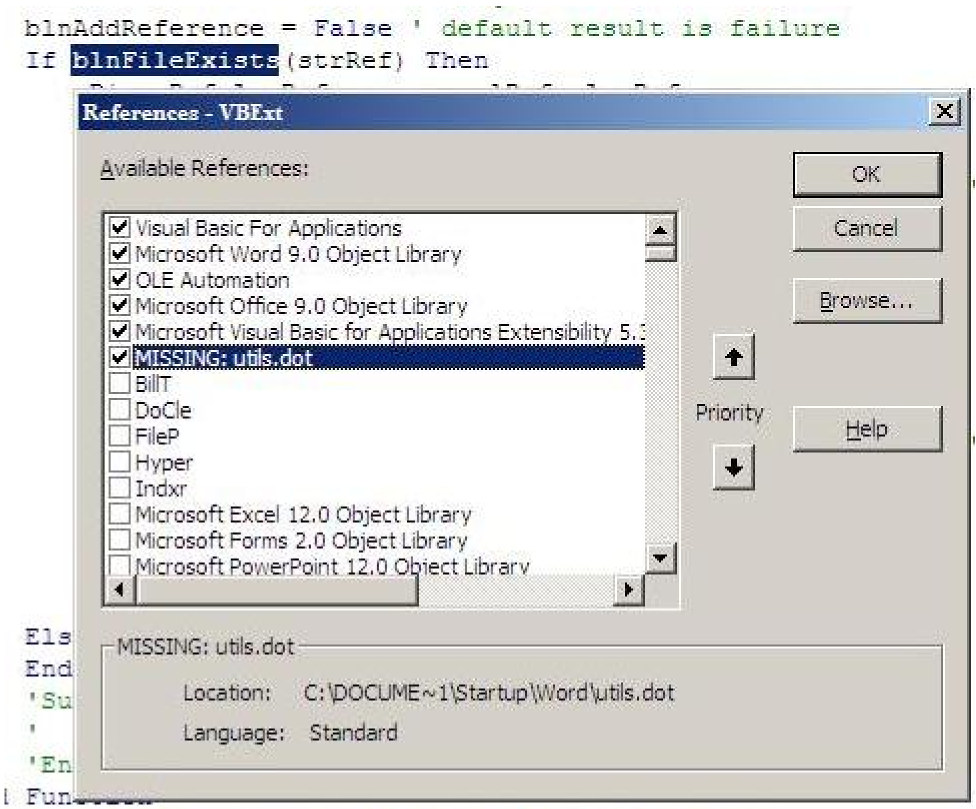
```
blnAddReference = False ' default r
If blnFileExists(strRef) Then
    Dim aRef As Reference, colRefs
                         tiveVBProje
                         Refs
                         f.Name & ";
                         FullPath,
                         nce = True
                         n


    colRefs.AddFromFile (strRef)
```

Microsoft Visual Basic

Compile error:

Can't find project or library

Enter VBE, resolve the missing references. This has occurred frequently in the past after I have made a major reorganization of library code, typically abandoning a library called U.DOT and embarking on UW.DOT. Very old projects pine for U.DOT.



```
blnAddReference = False ' default result is failure
If blnFileExists(strRef) Then
```

- [✓] Visual Basic For Applications
- [✓] Microsoft Word 9.0 Object Library
- [✓] OLE Automation
- [✓] Microsoft Office 9.0 Object Library
- [✓] Microsoft Visual Basic for Applications Extensibility 5.3
- [✓] MISSING: utils.dot
- [ ] BillT
- [ ] DoCle
- [ ] FileP
- [ ] Hyper
- [ ] Indxr
- [ ] Microsoft Excel 12.0 Object Library
- [ ] Microsoft Forms 2.0 Object Library
- [ ] Microsoft PowerPoint 12.0 Object Library

MISSING: utils.dot

Location: C:\DOCUME~1\Startup\Word\utils.dot

Language: Standard

Word Collapses

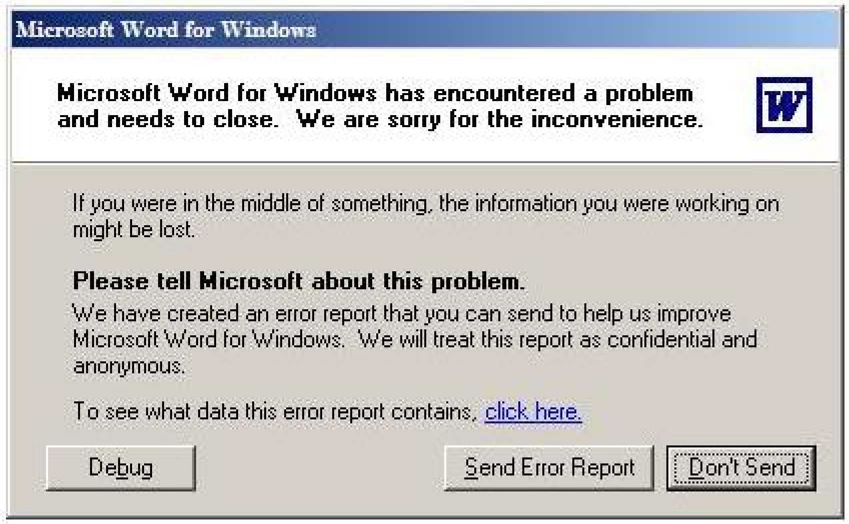
Annoying as it is that Word just DIES on us, it happens.

Examine the log file (look under ..\Greaves\Proje\ in your Documents And Settings folder) to determine the file being processed. It will be the last entry in the log file.

Reload Word (you may want to reboot to remove the lock placed on the errant file) and try to see what is odd about the project.

Fix any obvious problems (syntax errors, broken references and so on), then restart the process.

If all else fails, I have had some success by creating a new project document, pasting the text across, and export/importing the code modules, saving the project with a new name (UserGuide2.doc), deleting the old file (UserGuide.doc), then renaming the new file back to the old file name.



Microsoft Word for Windows

Microsoft Word for Windows has encountered a problem and needs to close. We are sorry for the inconvenience.

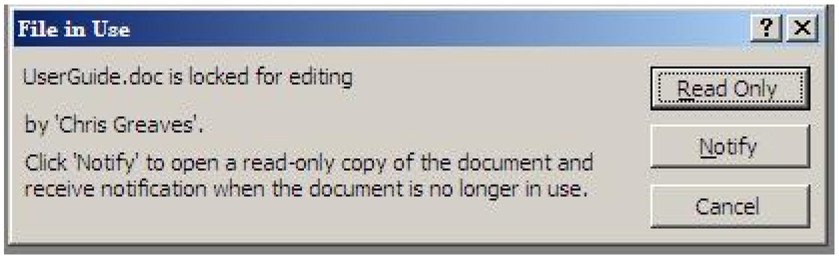
If you were in the middle of something, the information you were working on might be lost.

Please tell Microsoft about this problem.
We have created an error report that you can send to help us improve Microsoft Word for Windows. We will treat this report as confidential and anonymous.

To see what data this error report contains, click here.

Locked For Editing

After a crash, Word leaves the problem file locked for editing. No, you can't delete it. Yes, you will have to reboot the computer.



File in Use

UserGuide.doc is locked for editing

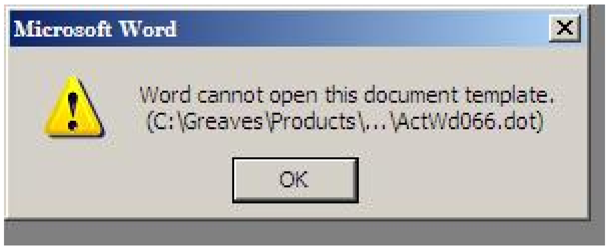
by 'Chris Greaves'.

Click 'Notify' to open a read-only copy of the document and receive notification when the document is no longer in use.

Can't Find Document

If you are attempting a "ReRun" of a previous project, and have deleted or renamed a file since the initial run, Proje will be confused, as in "That's odd! The file was here last time I ran!??!!!".

Restart the process, for example, choose HarvestCode, rather than HarvestCodeRepeat.



```
Word cannot open this document template.
(C:\Greaves\Products\...\ActWd066.dot)
```

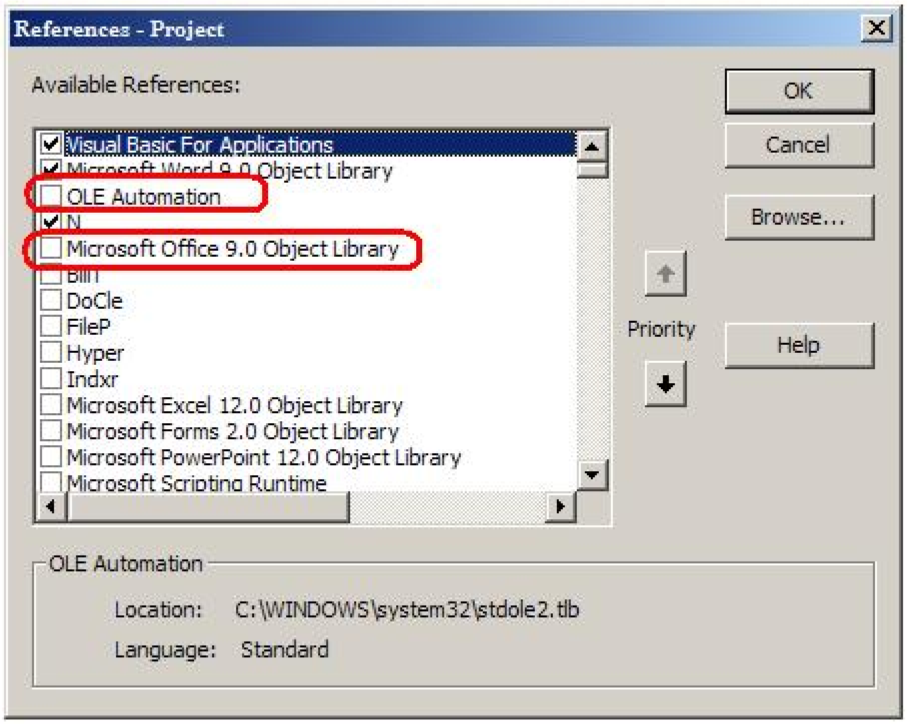
Word Collapses

Here is another cause: I had a project that crashed Word despite my rebuilding it completely. Somehow or other, two redundant references crept in, circled below. I removed the references, saved with a new name, rebooted, deleted the original and renamed the new, and the Harvester ran quite successfully through the file on the next pass. Go figure!

Later: No. Word still collapses. I resolved to save it as an RTF file and delete the DOC form to see what happens next.

Later: No. I rename the folder ..\Libraries\Word\ to be b:\Libraries_Word, that is, I move it right out of the path of processing temporarily.

I ran with a sub-folder of Libraries\ and the run completed.



☑ Visual Basic For Applications
☑ Microsoft Word 9.0 Object Library
☐ OLE Automation
☑ N
☐ Microsoft Office 9.0 Object Library
☐ Bill
☐ DoCle
☐ FileP
☐ Hyper
☐ Indxr
☐ Microsoft Excel 12.0 Object Library
☐ Microsoft Forms 2.0 Object Library
☐ Microsoft PowerPoint 12.0 Object Library
☐ Microsoft Scripting Runtime

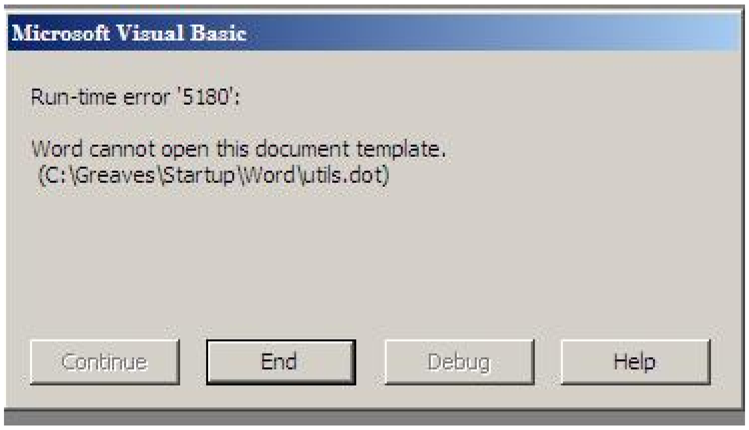
OLE Automation

Location: C:\WINDOWS\system32\stdole2.tlb

Language: Standard

Can't Open Template

Fix the references in VBE by choosing Tools, References.

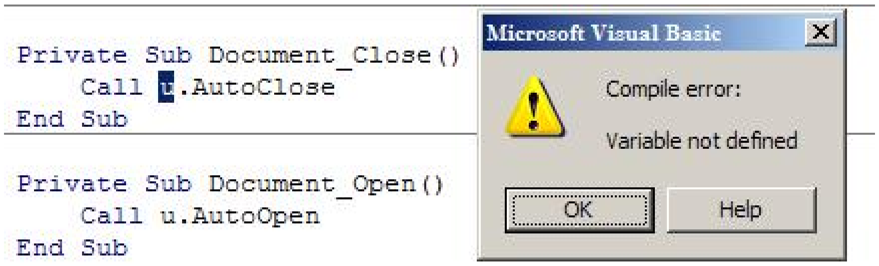


```
Run-time error '5180':

Word cannot open this document template.
 (C:\Greaves\Startup\Word\utils.dot)
```

Variable Not Defined

Fix the references in VBE by choosing Tools, References.

In my case I had also to remove the U. or replace it with the new project identifier UW.

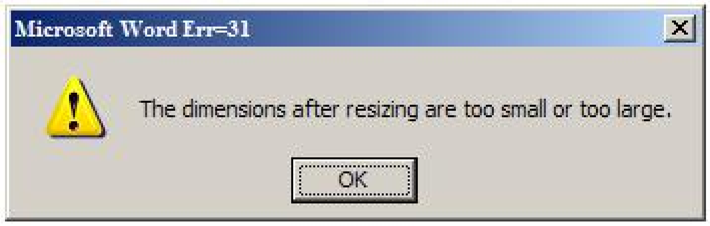


```
Private Sub Document_Close()
    Call u.AutoClose
End Sub

Private Sub Document_Open()
    Call u.AutoOpen
End Sub
```

Microsoft Visual Basic

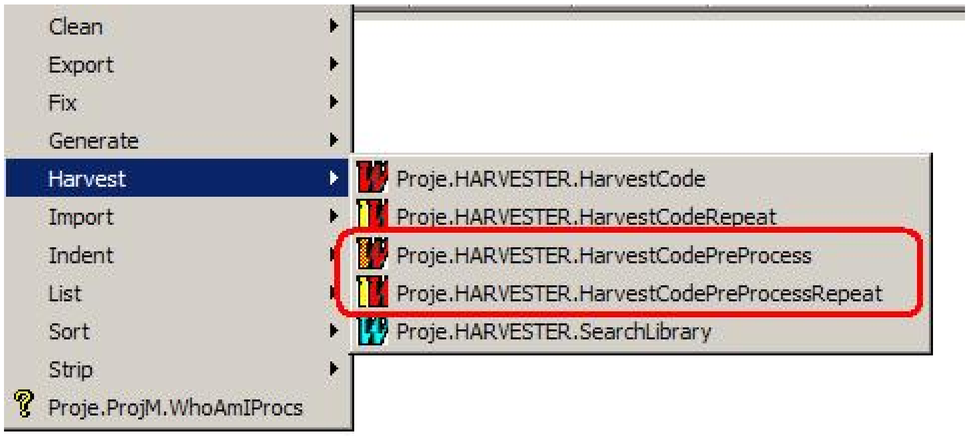
Compile error:

Variable not defined



Microsoft Word Err=31

The dimensions after resizing are too small or too large.

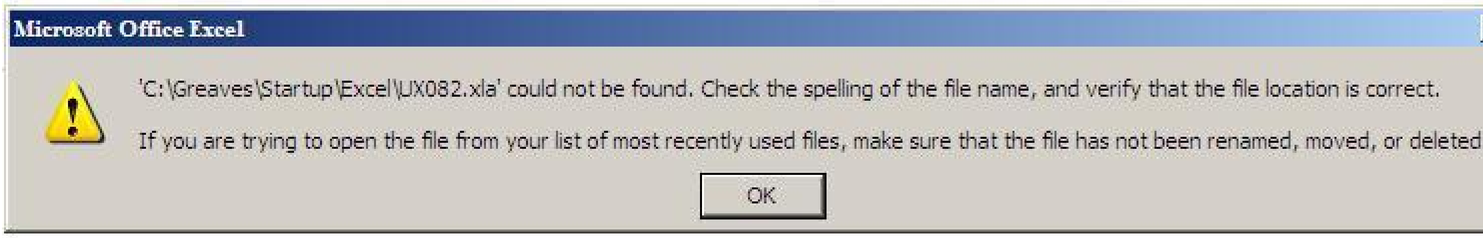


Clean
Export
Fix
Generate
Harvest
Import
Indent
List
Sort
Strip
Proje.ProjM.WhoAmIProcs
Proje.HARVESTER.HarvestCode
Proje.HARVESTER.HarvestCodeRepeat
Proje.HARVESTER.HarvestCodePreProcess
Proje.HARVESTER.HarvestCodePreProcessRepeat
Proje.HARVESTER.SearchLibrary

Could Not Be Found

This came from a Word document with a link to an Excel sheet. I had earlier removed the library call from Excels 97, 2000 and 2003, but of course, Word2000 linked this time to Excel2007 and found that the XLA was referenced but not available.

A quick trip to Excel2007 Options-land, remove the reference to the non-existent add-in, and all is well. Again.



Microsoft Office Excel

'C:\Greaves\Startup\Excel\UX082.xla' could not be found. Check the spelling of the file name, and verify that the file location is correct.

If you are trying to open the file from your list of most recently used files, make sure that the file has not been renamed, moved, or deleted

```
{·LINK·Excel.Sheet.8·C:\\Greaves\\Clients\\E\\EdTech\\20050805Costs.xls·
Sheet1!R1C1:R25C5·\a·\p·}¶
```

The Exclude File

Harvester references a small text file called "Exclude.txt" and avoids any file whose name appears in this file.

Thus, rather than deleting a troublesome file such as "C:\Greaves\Admin\20002001\Letters\AdultEducation.doc", merely paste the full name into Exclude .txt; Harvester will ignore that file on the next pass.

You can use a Folder name such as "" to exclude all files within that folder tree.

```
C:\Greaves\Products\DEVEL\WbWrd\WebWordRevised.doc
C:\Greaves\Products\
C:\Greaves\Clients\
C:\Greaves\Courses\Spreadsheets\Excel\Vba\ExcelVBA.doc
C:\Greaves\Courses\Programming\VBA\Libraries9.doc
C:\Greaves\
```

In the example above, I have been appending exclusions to the file Exclude.txt.

Once C:\Greaves has been processed, I excluded that folder and turned my attention back to pre-processing C:\, this time avoiding C:\Greaves and all its subsidiary folders.

Initially I excluded two specific files.

Came the day (!) that I had managed to process all the C:\Greaves\Products files, I temporarily excluded the folder C:\Greaves\Products from the processing, and turned my attention to C:\Greaves again.

I struggled through C:\Greaves\Clients - many problem documents there(!), but once through that folder sub-tree, I temporarily excluded it from processing.

I was then able to focus on C:\Greaves again, excluding C:\Greaves\products and C:\Greaves\Clients, which have been processed to completion.

Index